Development of intelligent systems (RInS)

Colours

Danijel Skočaj
University of Ljubljana
Faculty of Computer and Information Science

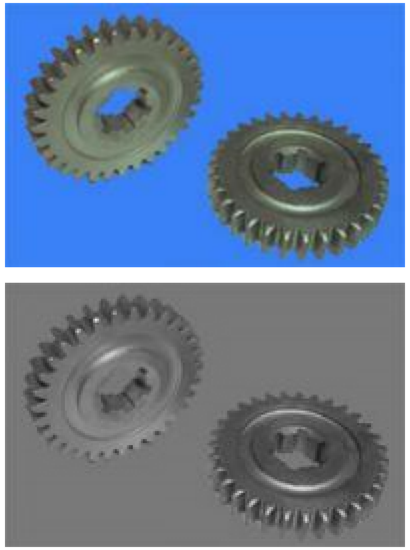
Literature: W. Burger, M. J. Burge (2008).
Digital Image Processing, chapter 12

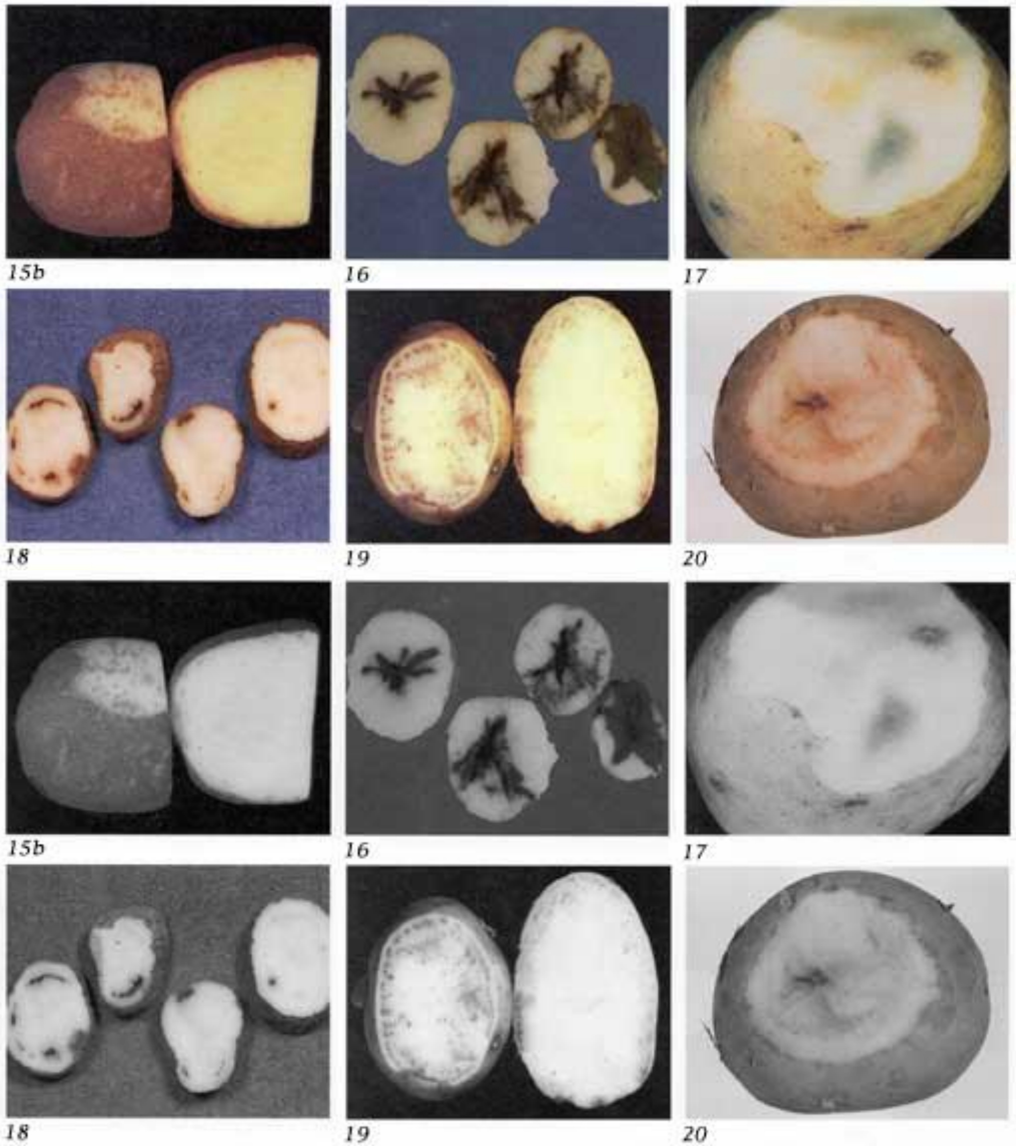
Academic year: 2021/22

Colour images

Colour images

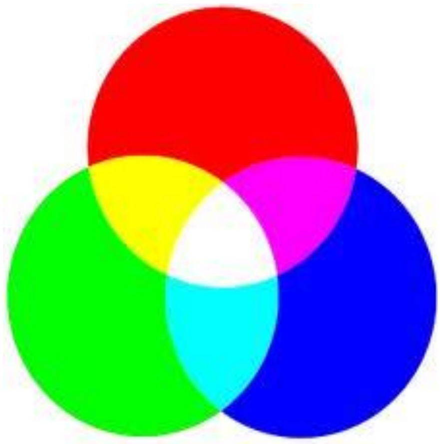
- Sometimes colours include meaningful information!

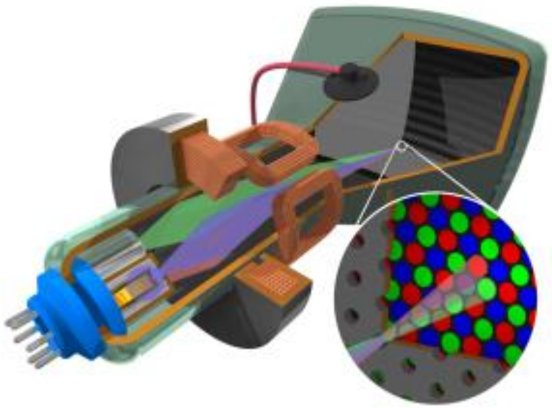




RGB colour images

- RGB colour scheme encodes colours as combinations of three basics colours: red, green and blue
- Very frequently used
- Additive colour system

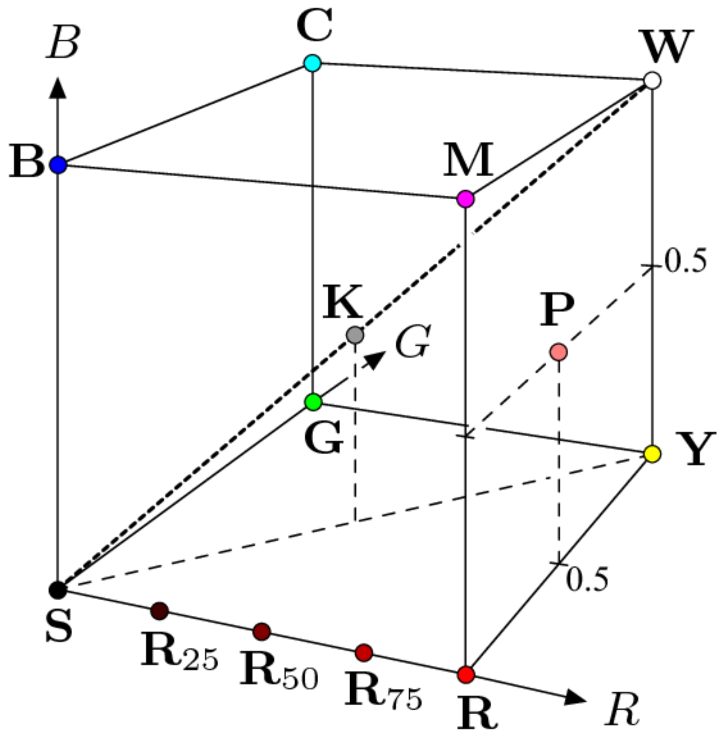




RGB colour space

- Every colour is a point in the 3D RGB space

$$\mathbf{C}_i = (R_i, G_i, B_i)$$



	RGB Value				
Point	Color	R	G	B
S	Black	0.00	0.00	0.00
R	Red	1.00	0.00	0.00
Y	Yellow	1.00	1.00	0.00
G	Green	0.00	1.00	0.00
C	Cyan	0.00	1.00	1.00
B	Blue	0.00	0.00	1.00
M	Magenta	1.00	0.00	1.00
W	White	1.00	1.00	1.00
K	50% Gray	0.50	0.50	0.50
$\mathbf{R}_{75}$	75% Red	0.75	0.00	0.00
$\mathbf{R}_{50}$	50% Red	0.50	0.00	0.00
$\mathbf{R}_{25}$	25% Red	0.25	0.00	0.00
P	Pink	1.00	0.50	0.50

RGB channels

R G B

Conversion to grayscale images

- Simple conversion:

$$Y = \mathrm{Avg}(R, G, B) = \frac{R + G + B}{3}$$

- Human eye perceives red and green as brighter than blue, hence we can use the weighted average:

$$Y = \mathrm{Lum}(R, G, B) = w_R \cdot R + w_G \cdot G + w_B \cdot B$$

$$w_R = 0.299 \qquad w_G = 0.587 \qquad w_B = 0.114$$

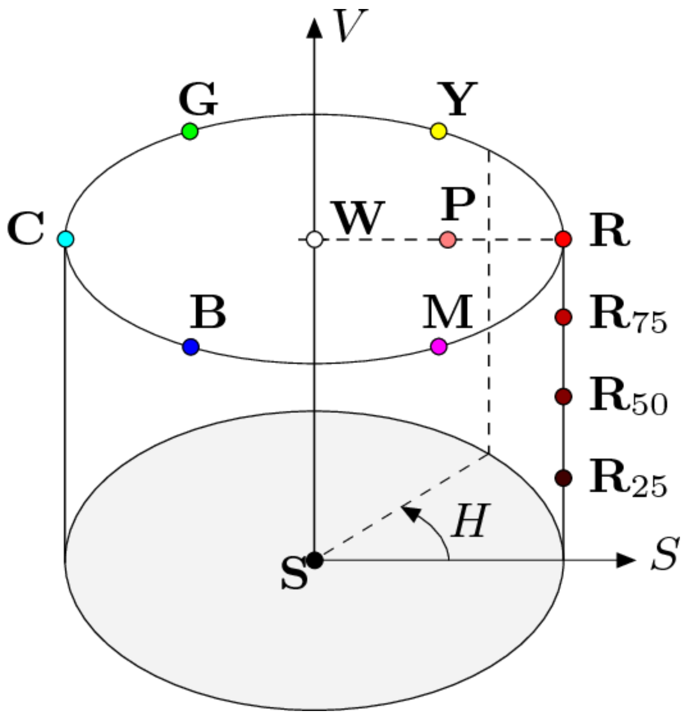
$$w_R = 0.2125 \qquad w_G = 0.7154 \qquad w_B = 0.072$$

- Grayscale RGB images have all three components equal:

$$R = G = B \qquad \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} \leftarrow \begin{pmatrix} Y \\ Y \\ Y \end{pmatrix}$$

HSV colour space

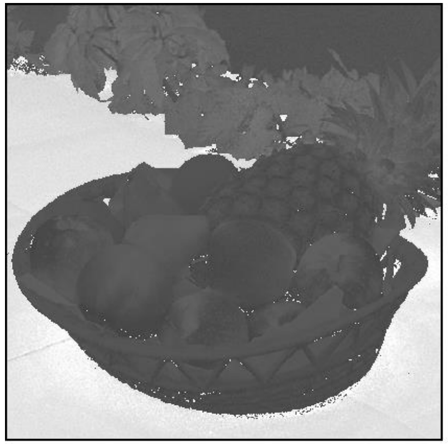
- Hue, Saturation, Value



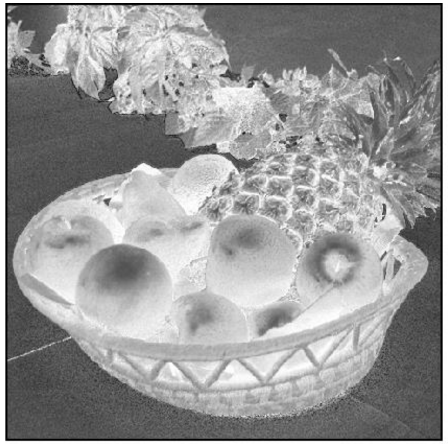
RGB/HSV Values

Pt.	Color	R	G	B	H	S	V
S	Black	0.00	0.00	0.00	—	0.00	0.00
R	Red	1.00	0.00	0.00	0	1.00	1.00
Y	Yellow	1.00	1.00	0.00	1/6	1.00	1.00
G	Green	0.00	1.00	0.00	2/6	1.00	1.00
C	Cyan	0.00	1.00	1.00	3/6	1.00	1.00
B	Blue	0.00	0.00	1.00	4/6	1.00	1.00
M	Magenta	1.00	0.00	1.00	5/6	1.00	1.00
W	White	1.00	1.00	1.00	—	0.00	1.00
$\mathbf{R}_{75}$	75% Red	0.75	0.00	0.00	0	1.00	0.75
$\mathbf{R}_{50}$	50% Red	0.50	0.00	0.00	0	1.00	0.50
$\mathbf{R}_{25}$	25% Red	0.25	0.00	0.00	0	1.00	0.25
P	Pink	1.00	0.50	0.50	0	0.5	1.00

HSV channels



H_{HSV}



S_{HSV}

V_{HSV}

Conversion from RGB to HSV

$$C_{\text{high}} = \max(R, G, B) \quad C_{\text{low}} = \min(R, G, B) \quad C_{\text{rng}} = C_{\text{high}} - C_{\text{low}}$$

$$S_{\text{HSV}} = \begin{cases} \frac{C_{\text{rng}}}{C_{\text{high}}} & \text{for } C_{\text{high}} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$V_{\text{HSV}} = \frac{C_{\text{high}}}{C_{\text{max}}}$$

255

$$R' = \frac{C_{\text{high}} - R}{C_{\text{rng}}} \qquad G' = \frac{C_{\text{high}} - G}{C_{\text{rng}}} \qquad B' = \frac{C_{\text{high}} - B}{C_{\text{rng}}}$$

$$H' = \begin{cases} B' - G' & \text{if } R = C_{\text{high}} \\ R' - B' + 2 & \text{if } G = C_{\text{high}} \\ G' - R' + 4 & \text{if } B = C_{\text{high}} \end{cases}$$

$$H_{\text{HSV}} = \frac{1}{6} \cdot \begin{cases} (H' + 6) & \text{for } H' < 0 \\ H' & \text{otherwise} \end{cases}$$

Algorithm

```
static float[] RGBtoHSV (int R, int G, int B, float[] HSV) {
  // R, G, B ∈ [0, 255]
  float H = 0, S = 0, V = 0;
  float cMax = 255.0f;
  int cHi = Math.max(R,Math.max(G,B)); // highest color value
  int cLo = Math.min(R,Math.min(G,B)); // lowest color value
  int cRng = cHi - cLo;          // color range

  // compute value V
  V = cHi / cMax;

  // compute saturation S
  if (cHi > 0)
    S = (float) cRng / cHi;

  // compute hue H
  if (cRng > 0) { // hue is defined only for color pixels
    float rr = (float)(cHi - R) / cRng;
    float gg = (float)(cHi - G) / cRng;
    float bb = (float)(cHi - B) / cRng;
    float hh;
    if (R == cHi)                    // R is highest color value
      hh = bb - gg;
    else if (G == cHi)               // G is highest color value
      hh = rr - bb + 2.0f;
    else                             // B is highest color value
      hh = gg - rr + 4.0f;
    if (hh < 0)
      hh= hh + 6;
    H = hh / 6;
  }

  if (HSV == null) // create a new HSV array if needed
    HSV = new float[3];
  HSV[0] = H; HSV[1] = S; HSV[2] = V;
  return HSV;
}
```

Conversion from HSV to RGB

$$H' = (6 \cdot H_{\mathrm{HSV}}) \bmod 6$$

$$\begin{aligned} c_1 &= \lfloor H' \rfloor & x &= (1 - S_{\mathrm{HSV}}) \cdot v \\ c_2 &= H' - c_1 & y &= (1 - (S_{\mathrm{HSV}} \cdot c_2)) \cdot V_{\mathrm{HSV}} \\ & & z &= (1 - (S_{\mathrm{HSV}} \cdot (1 - c_2))) \cdot V_{\mathrm{HSV}} \end{aligned}$$

$$(R', G', B') = \begin{cases} (v, z, x) & \text{if } c_1 = 0 \\ (y, v, x) & \text{if } c_1 = 1 \\ (x, v, z) & \text{if } c_1 = 2 \\ (x, y, v) & \text{if } c_1 = 3 \\ (z, x, v) & \text{if } c_1 = 4 \\ (v, x, y) & \text{if } c_1 = 5. \end{cases}$$

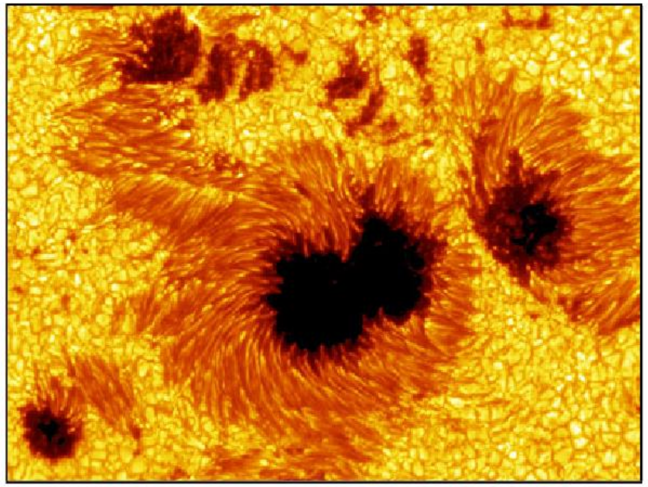
$$R = \min\big(\mathrm{round}(N \cdot R'), N - 1\big)$$

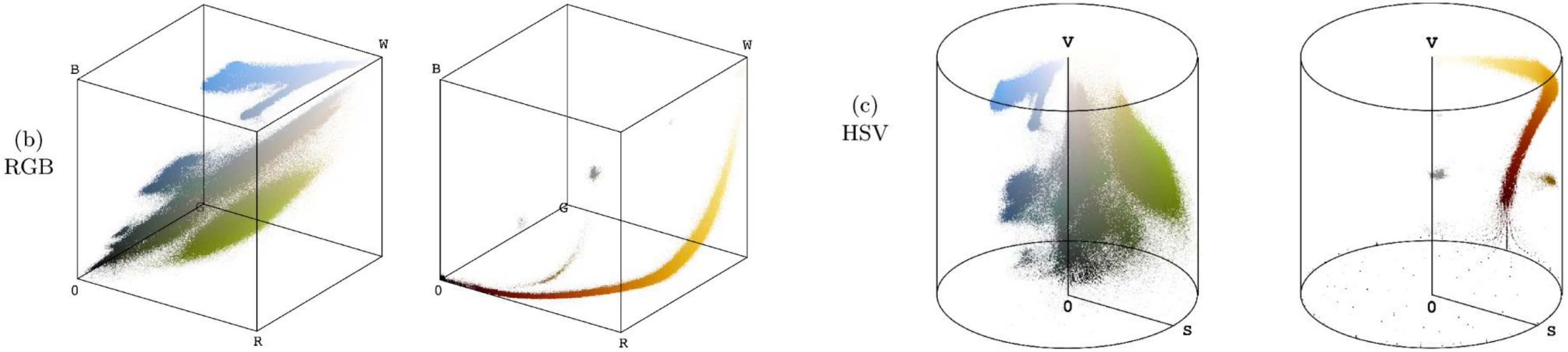
256

$$G = \min\big(\mathrm{round}(N \cdot G'), N - 1\big)$$

$$B = \min\big(\mathrm{round}(N \cdot B'), N - 1\big)$$

Examples



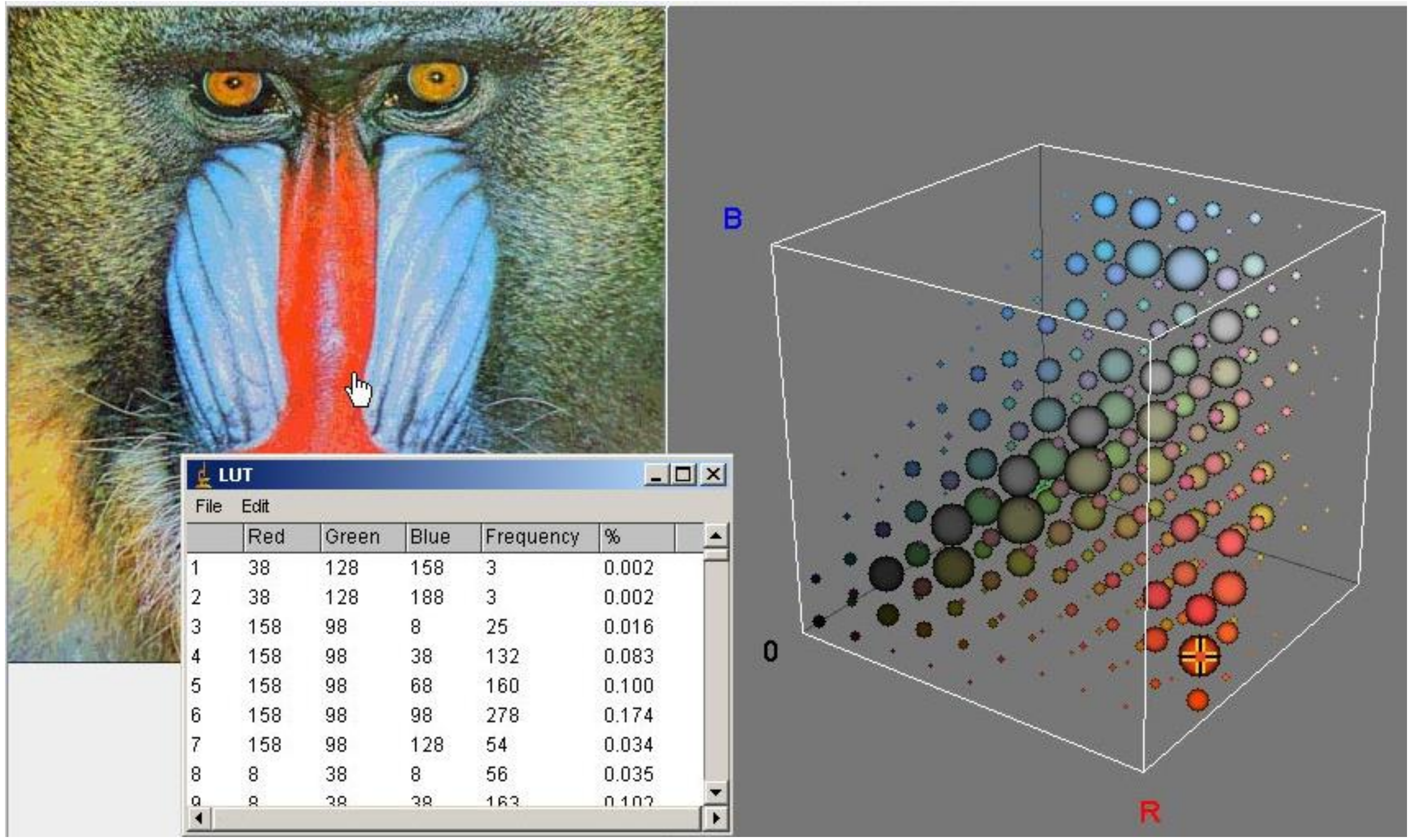


Other colour spaces

- HLS
- TV colour spaces
  - YUV
  - YIQ
  - YCbCr
- Colour spaces for print
  - CMY
  - CMYK
- Colorimetric colour spaces
  - CIE XYZ
  - CIE YUV, YU‘V‘, L*u*v, YCbCr
  - CIE L*a*b*
  - sRGB

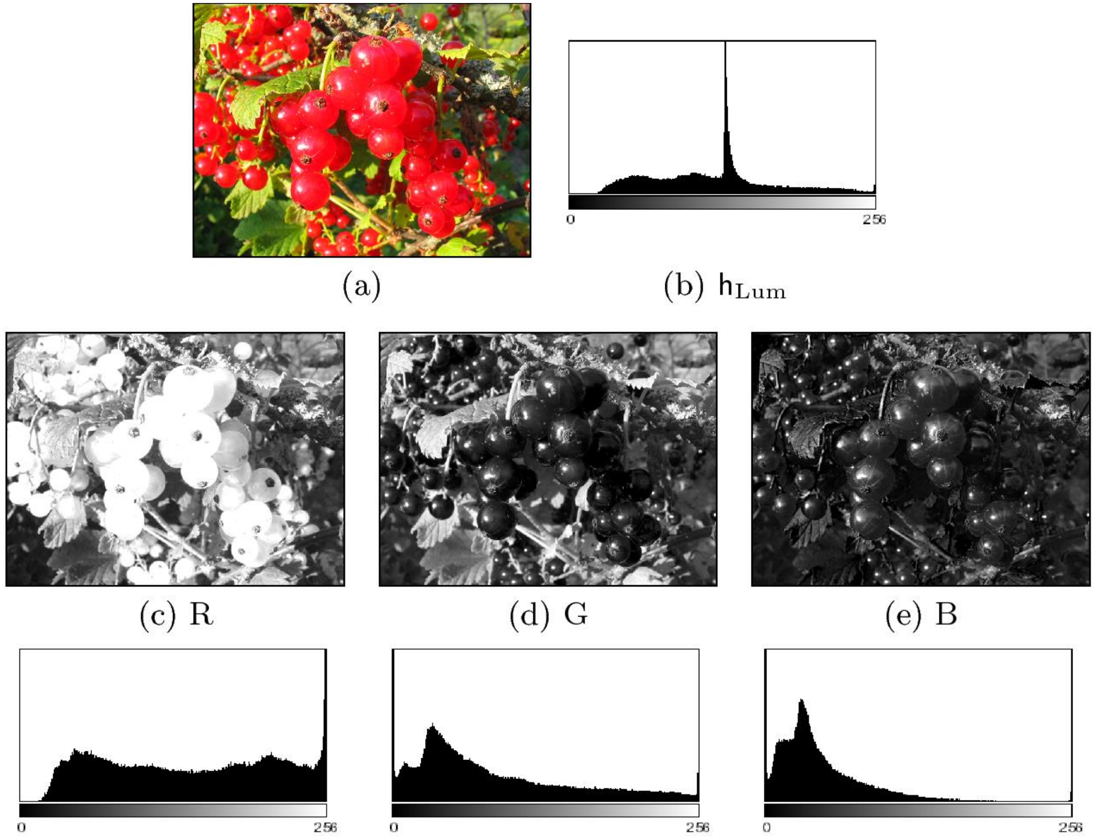
3D colour histograms

- 3 components -> 3D histogram
  - High space complexity, „sparse“



1D colour histograms

- 1 D histograms of the individual components
- Do not model correlations between individual colour components



(a) (b) h_{Lum}

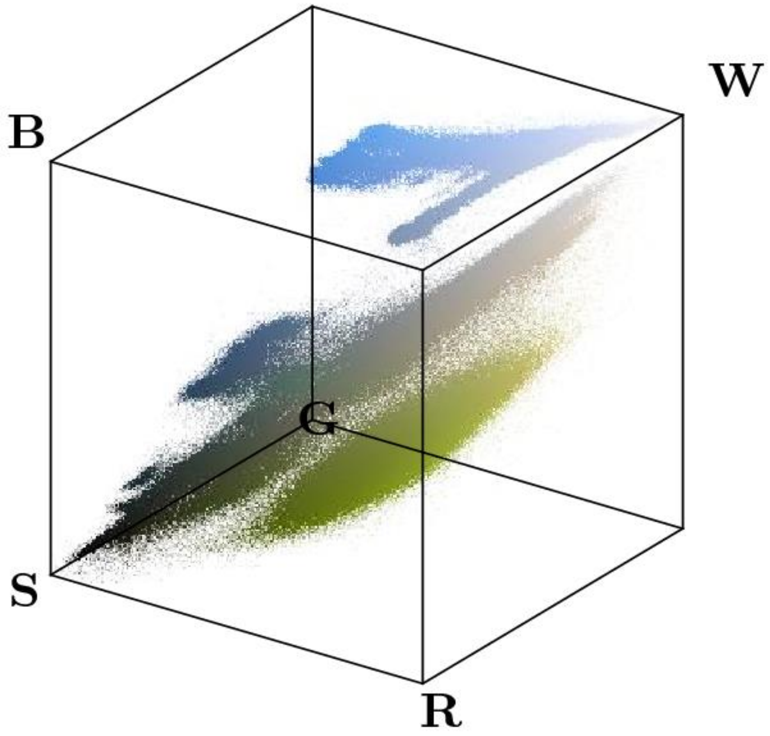
(c) R (d) G (e) B

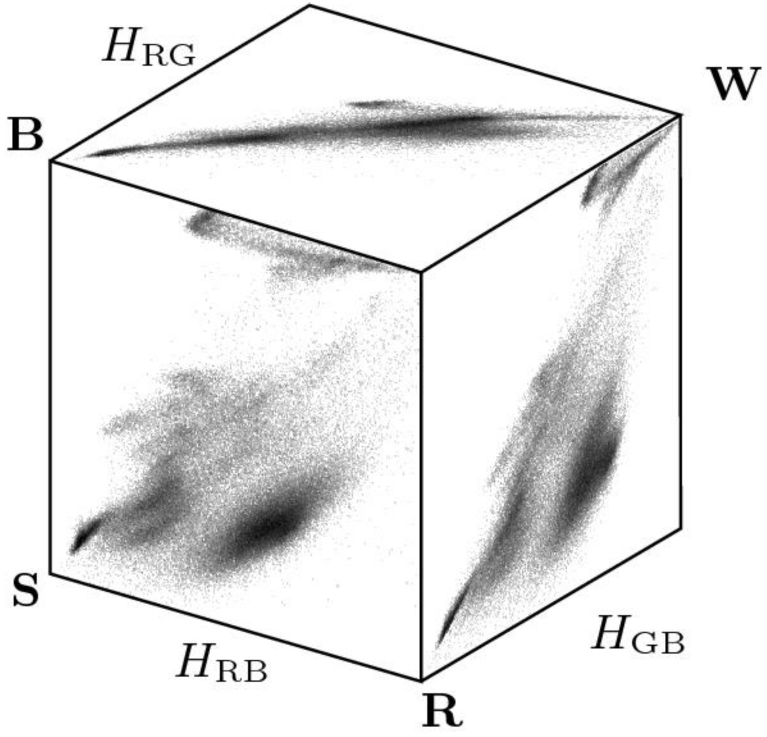
(f) h_R (g) h_G (h) h_B

2D colour histograms

- Calculate pairs of 2D histograms
  - Encompass at least a partial correlation between the individual components

$$
\begin{aligned}
H_{\mathrm{RG}}(r,g) &\leftarrow \text{number of pixels with } I_{\mathrm{RGB}}(u,v)=(r,g,*)\\
H_{\mathrm{RB}}(r,b) &\leftarrow \text{number of pixels with } I_{\mathrm{RGB}}(u,v)=(r,*,b)\\
H_{\mathrm{GB}}(g,b) &\leftarrow \text{number of pixels with } I_{\mathrm{RGB}}(u,v)=(*,g,b)
\end{aligned}
$$

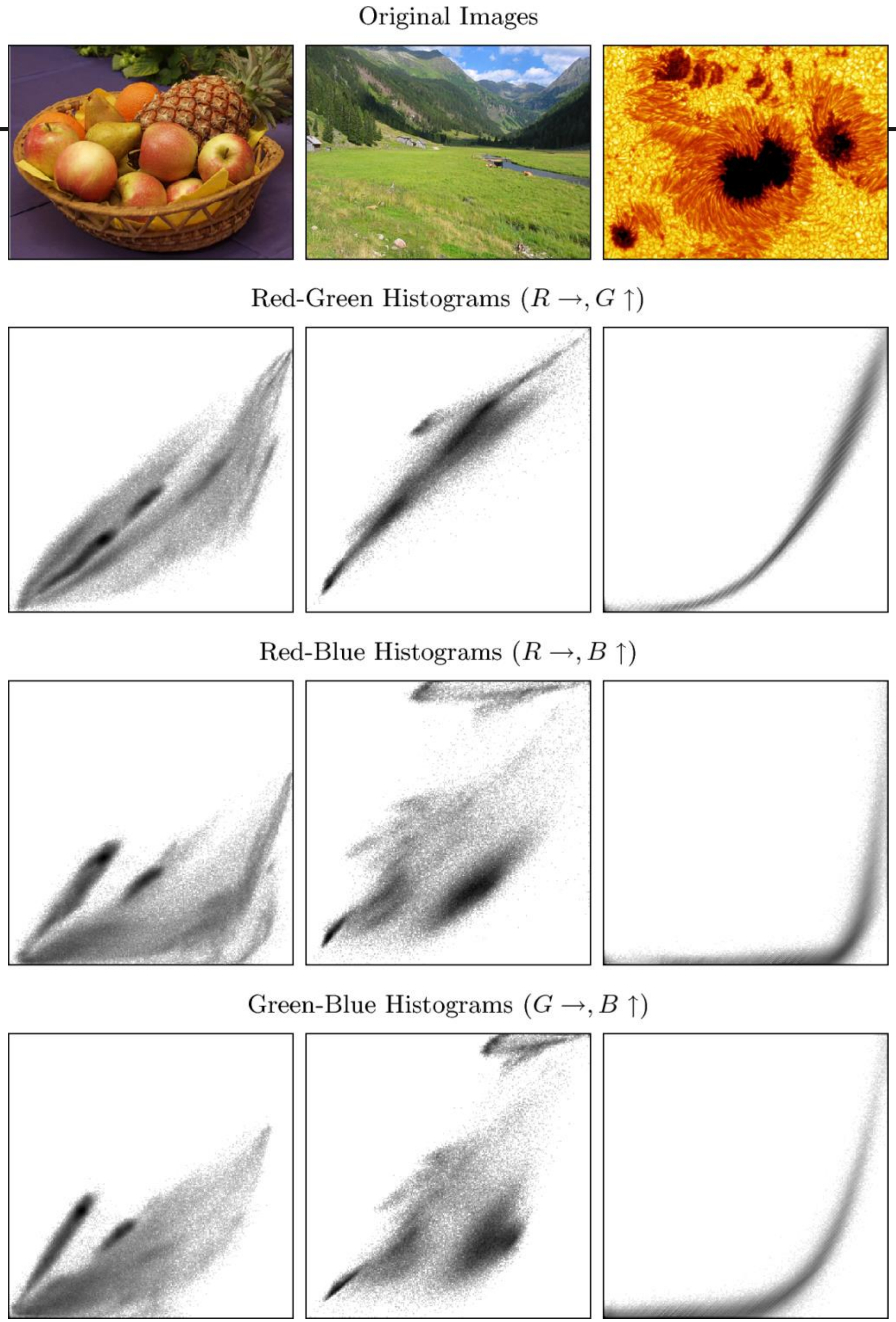




Algorithm

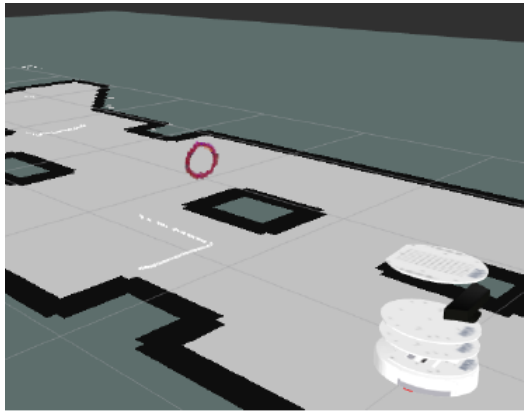
```
static int[][] get2dHistogram
                (ColorProcessor cp, int c1, int c2) {
  // c1, c2: R = 0, G = 1, B = 2
  int[] RGB = new int[3];
  int[][] H = new int[256][256]; // histogram array H[c1][c2]

  for (int v = 0; v < cp.getHeight(); v++) {
    for (int u = 0; u < cp.getWidth(); u++) {
      cp.getPixel(u, v, RGB);
      int i = RGB[c1];
      int j = RGB[c2];
      // increment corresponding histogram cell
      H[j][i]++; // i runs horizontal, j runs vertical
    }
  }
  return H;
}
```

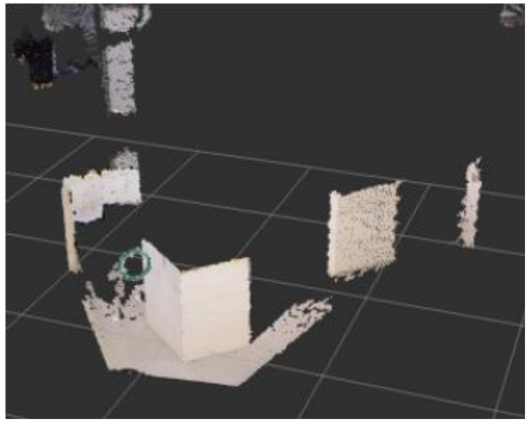
Examples

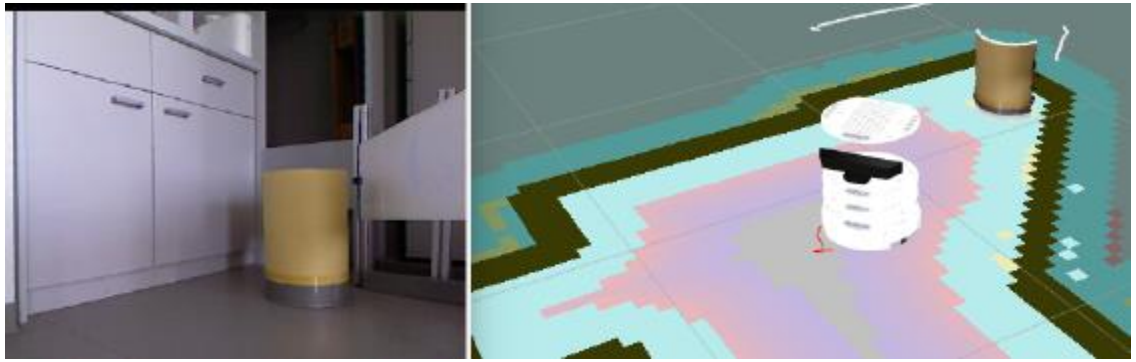

Original Images
Red-Green Histograms ($R \rightarrow, G \uparrow$)
Red-Blue Histograms ($R \rightarrow, B \uparrow$)
Green-Blue Histograms ($G \rightarrow, B \uparrow$)

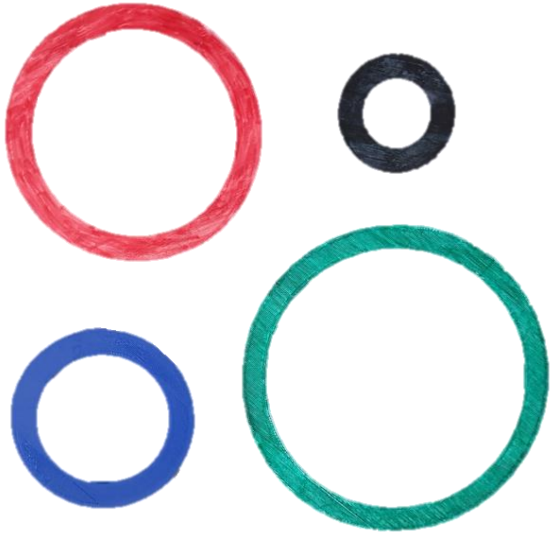
Object colours

- Rings of different colours
- Cylinders of different colours









Colour recognition

- Detect and segment the object
  - in 2D or 3D
- Modelling colours
  - Probability distribution
  - Gaussian,
    mixture of Gaussians
- Train a classifier
  - SVM, ANN, kNN,…
- In 1D, 2D or 3D space
- RGB, HSV and other colour spaces
- Working with the individual pixels or histograms
- Working with images