Introduction to Image Processing

Lecturer:
Dr. mat. nat. Christian Kehl

Week 2 – Image enhancement, Geometric Transformation and Colours

November 15, 2024

Image Enhancement

Image enhancement

Process of improving image quality so that the result is more suitable for a specific application.

Image enhancement

Process of improving image quality so that the result is more suitable for a specific application.

Sample image enhancement methods:

- contrast stretching
- histogram processing
- smoothing
- sharpening

Classification of image operations

- *Point operation*: output pixel value depends only on corresponding input pixel (e.g. contrast- and brightness): $f_{out}(x, y) = \mathcal{O}(\{f_{in}(x, y)\})$ where $\mathcal{O}$ denotes the *grayscale transformation* function.

Classification of image operations

- *Point operation*: output pixel value depends only on corresponding input pixel (e.g. contrast- and brightness): $f_{out}(x, y) = \mathcal{O}(\{f_{in}(x, y)\})$ where $\mathcal{O}$ denotes the *grayscale transformation* function.
- *Local operation*: output pixel value depends on neighborhood $\mathcal{N}(x, y)$ of input pixel: $f_{out}(x, y) = \mathcal{O}(\{f_{in}(x', y') : (x', y') \in \mathcal{N}(x, y)\})$.

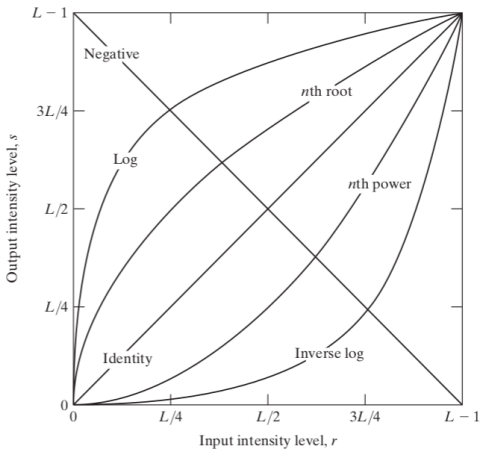
Classification of image operations

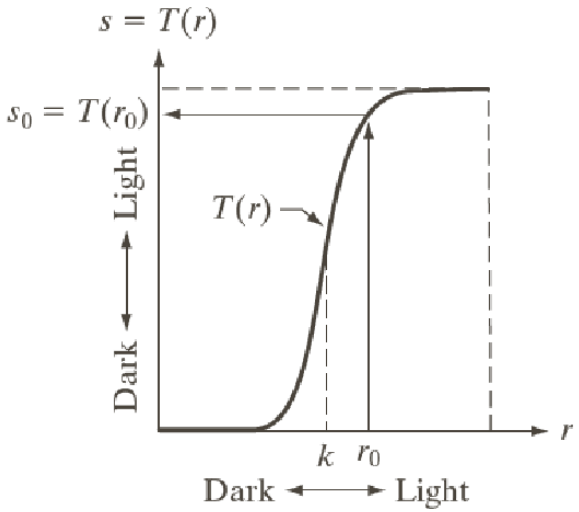
- *Point operation*: output pixel value depends only on corresponding input pixel (e.g. contrast- and brightness): $f_{out}(x, y) = \mathcal{O}(\{f_{in}(x, y)\})$ where $\mathcal{O}$ denotes the *grayscale transformation* function.
- *Local operation*: output pixel value depends on neighborhood $\mathcal{N}(x, y)$ of input pixel: $f_{out}(x, y) = \mathcal{O}(\{f_{in}(x', y') : (x', y') \in \mathcal{N}(x, y)\})$.
- *Global operation*: output pixel value depends on all input pixels. Example: frequency transforms.

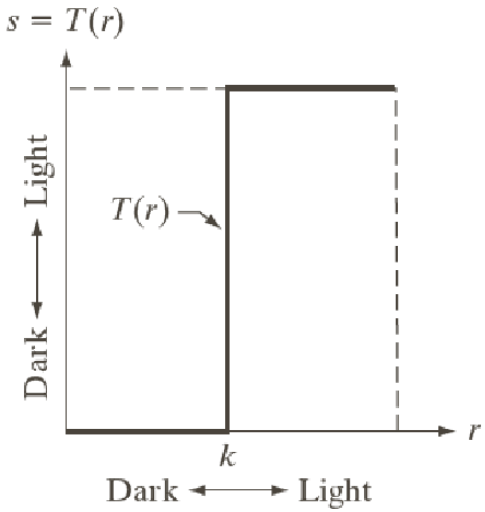
Classification of image operations

- *Point operation*: output pixel value depends only on corresponding input pixel (e.g. contrast- and brightness): $f_{out}(x, y) = \mathcal{O}(\{f_{in}(x, y)\})$ where $\mathcal{O}$ denotes the *grayscale transformation* function.
- *Local operation*: output pixel value depends on neighborhood $\mathcal{N}(x, y)$ of input pixel: $f_{out}(x, y) = \mathcal{O}(\{f_{in}(x', y') : (x', y') \in \mathcal{N}(x, y)\})$.
- *Global operation*: output pixel value depends on all input pixels. Example: frequency transforms.
- *Geometric operation*: spatial transformation (scaling, translation, rotation)

Point operations







(mid) Non-linear Contrast stretching. (right) Thresholding.

Implementation: **forall pixels** p **do** $image[p] = T[image[p]]$

Brightness adaptation

- Image brightness: average intensity of an image
- visible in histogram → too dim / bright
  audience Q: how would that look like (in the histogram) ?

Brightness adaptation

- Image brightness: average intensity of an image
- visible in histogram → too dim / bright
  audience Q: how would that look like (in the histogram) ?
- Enhancement: raise (i.e. add) or lower (i.e. subtract) average intensity a

$$f_{out}(x, y) = f_{in}(x, y) + a.$$

- rounding or clamping (i.e. truncation) may be required

(left) Over-exposed aerial image.
(right) Brightness-corrected aerial image.

Linear contrast stretching

- Used if features of interest occupy only a *small range* of the available gray levels → see in histogram

Linear contrast stretching

- Used if features of interest occupy only a *small range* of the available gray levels → see in histogram
- Let the input image f_{in} have minimum and maximum gray level min and max ($0 \leqslant min < max$), respectively. Then the following operation stretches the gray level range of the image to [0..255]:

$$f_{out}(x, y) = \frac{255}{max - min} \left(f_{in}(x, y) - min\right)$$

To convert the output to an integer image, a rounding (or truncation) operation needs to be performed.

Contrast stretch

input

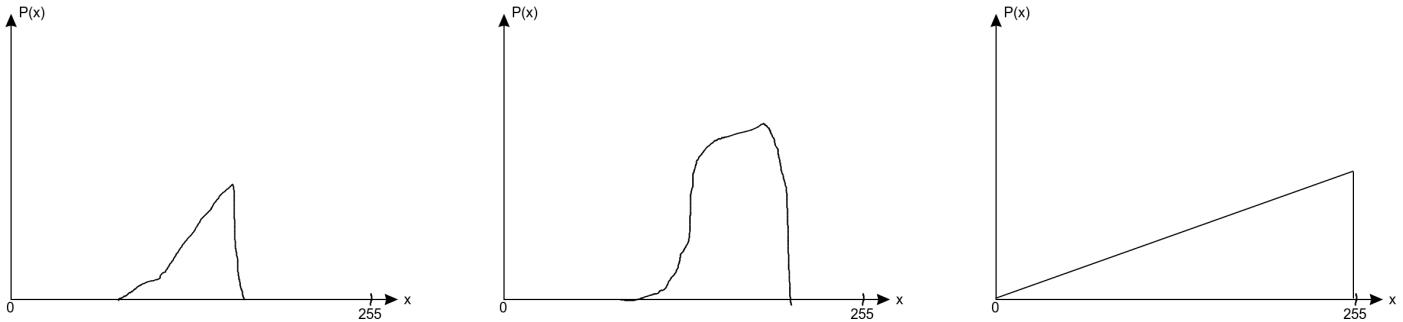
contrast stretched

Gamma correction

Brightness- & contrast adaptation have some issues:

- Two separate, yet interdependent functions to be adapted
- Only a *linear* stretch (i.e. transformation) possible

Some distortions need a non-linear correction ...



(left) Linear histogram (in). (mid) Non-linear histogram (in). (right) Target histogram stretch (out).

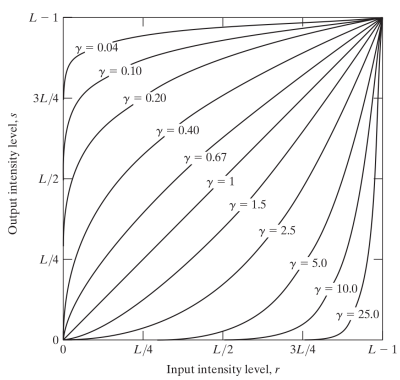
Gamma correction

Solution: Gamma correction

- Non-linear transformation function
- both control parameters for offset c and stretch γ in 1 equation

$$f_{out}(x, y) = c\ f_{in}(x, y)^{\gamma}, \gamma > 0$$

Example correction: atmosperic haze



(left) Effect of different γ levels. (mid-left) Original aerial image. (mid-right) Correction $\gamma = 3.0$. (right) Correction $\gamma = 5.0$.

Histogram equalization - 1/6

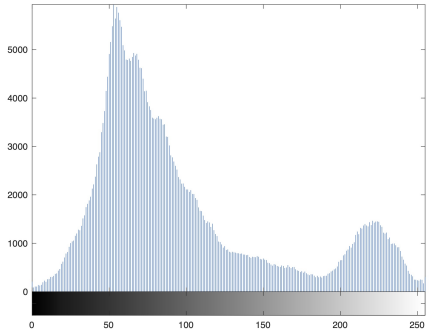
What γ-correction doesn't resolve: unequal histogram distribution.
In order to fix that, we need **information theory:**

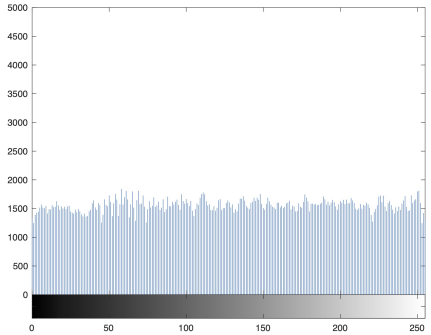
- occurence probability of value $f(x)$ in image I
- predictable value distribution: $P(x) = \frac{1}{L}\ \forall x \in [0..L-1]$ (equiprobability)
  $\rightarrow$ *information maximization*

If the gray levels are far from equally distributed, then the image exhibits a low contrast.

Histogram equalization - 2/6

(left) before and (right) after histogram equalization.





Histogram equalization - 3/6

- Goal: *flat histogram* in the output, i.e., on average an equal number of pixels at each gray level.
- If the image has N pixels and gray level range $[0, L-1]$, the output image will have (on average) $N/(L-1)$ pixels for each gray level.

Histogram equalization - 3/6

- Goal: *flat histogram* in the output, i.e., on average an equal number of pixels at each gray level.
- If the image has N pixels and gray level range $[0, L-1]$, the output image will have (on average) $N/(L-1)$ pixels for each gray level.
- The grayscale transformation achieving histogram equalization is $\mathcal{O}(f(x,y)) = (L-1)\,\mathrm{P}(f(x,y))$, where

$$\mathrm{P}(\ell) = \frac{1}{N}\sum_{m=0}^{\ell} h(m)$$

is the *normalized cumulative histogram* function.

Histogram equalization - 3/6

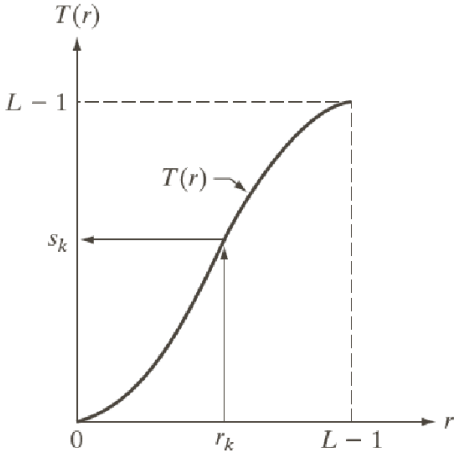
- Goal: *flat histogram* in the output, i.e., on average an equal number of pixels at each gray level.
- If the image has N pixels and gray level range $[0, L-1]$, the output image will have (on average) $N/(L-1)$ pixels for each gray level.
- The grayscale transformation achieving histogram equalization is $\mathcal{O}(f(x,y)) = (L-1)\,\mathrm{P}(f(x,y))$, where

$$\mathrm{P}(\ell) = \frac{1}{N}\sum_{m=0}^{\ell} h(m)$$

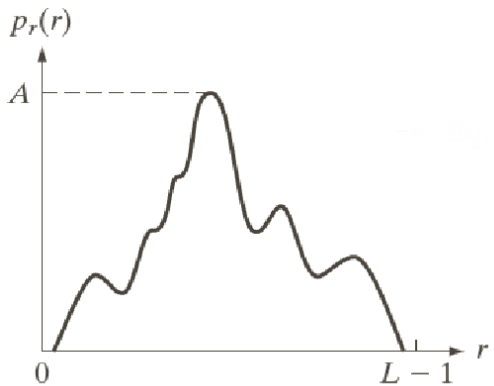
is the *normalized cumulative histogram* function.
(**hint**: compute $\mathrm{P}(\ell)$ first ...)

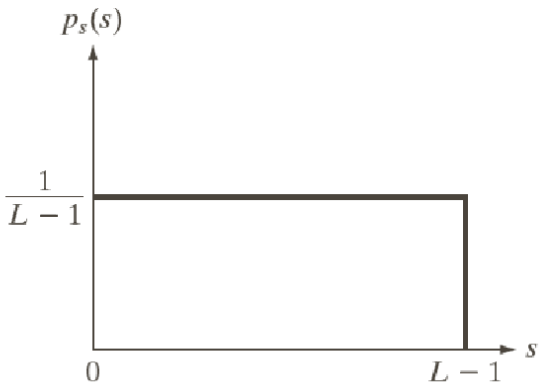
Histogram equalization: details - 4/6

We apply a *strictly monotonically increasing* intensity mapping $s = T(r)$, with r = input level and s = output level.



We want to choose $T(r)$ in such a way that the histogram $\tilde{P}(s)$ after the transformation is flat (i.e. constant).

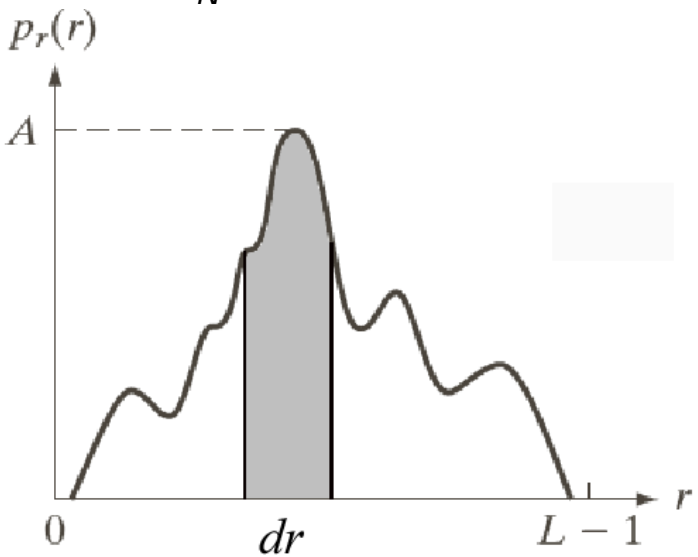


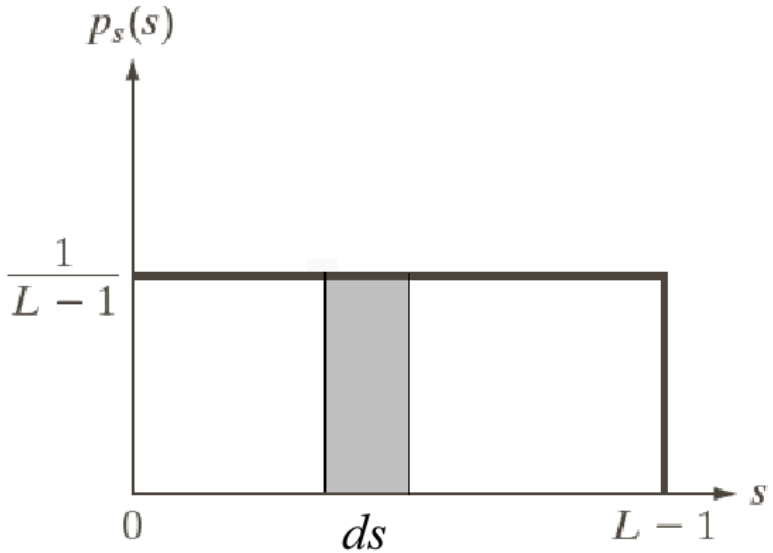


(hint: P(x) = *probability* $\rightarrow \{p_r(r), p_s(s)\} \in [0..1]$)

Histogram equalization: details - 5/6

Definition: no. pixels $\|x \in X | f(x) = r\|$ of intensity value $r \rightarrow h(r)$. Thus, $P(r) = \frac{h(r)}{N}$.





The total number of pixels covered by interval dr is the same as the number of pixels covered by interval ds after transformation:
$h(r)\ dr = \tilde{h}(s)\ ds$

Histogram equalization: details - 6/6

- $\tilde{h}(s) = constant = N/(L-1)$. $\rightarrow$

$$T'(r) = \frac{ds}{dr} = \frac{L-1}{N} h(r) \qquad \text{(reverse transform)}$$

$$T(r) = \frac{L-1}{N} \int_0^r h(r)\, dr \qquad \text{(forward transform)}$$

Histogram equalization: details - 6/6

- $\tilde{h}(s) = constant = N/(L-1)$. $\rightarrow$

$$T'(r) = \frac{ds}{dr} = \frac{L-1}{N} h(r) \qquad \text{(reverse transform)}$$

$$T(r) = \frac{L-1}{N} \int_0^r h(r)\, dr \qquad \text{(forward transform)}$$

- **Discrete space $\rightarrow$ the integral becomes a sum**

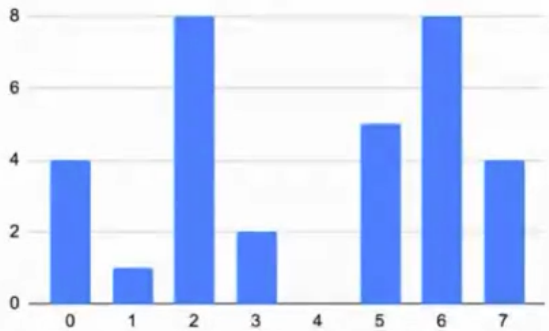
$$T(r) = \frac{L-1}{N} \sum_{m=0}^{r} h(m),$$

with $\sum_{m=0}^{r} h(m) = H(m)$ (*cumulative histogram*)

- compare:
$\mathcal{O}(f(x,y)) = (L-1)\ \mathrm{P}(f(x,y)); \mathrm{P}(\ell) = \frac{1}{N} \sum_{m=0}^{\ell} h(m)$

Histogram equalization: concrete example

Histogram Equalization

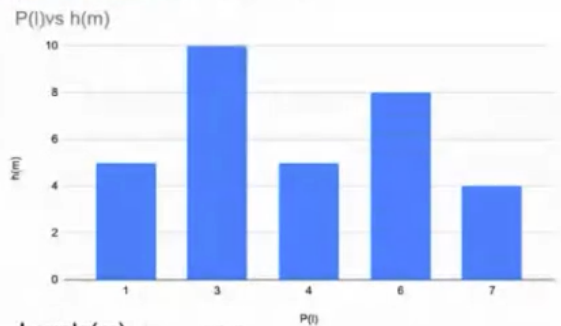


The grey scale transformation achieving histogram equalisation is $\mathcal{O}(f(x,y)) = (L-1)\,\mathrm{P}(f(x,y))$, where

$$\mathrm{P}(\ell) = \frac{1}{N}\sum_{m=0}^{\ell} h(m)$$

is the normalised cumulative histogram function.

Gray Levels(m)	h(m)	PDF	H(m)	CDF	$(L-1)$CDF	$\mathcal{O}$(m)
0	4	0.13	4	0.13	0.88	1.00
1	1	0.03	5	0.16	1.09	1.00
2	8	0.25	13	0.41	2.84	3.00
3	2	0.06	15	0.47	3.28	3.00
4	0	0.00	15	0.47	3.28	3.00
5	5	0.16	20	0.63	4.38	4.00
6	8	0.25	28	0.88	6.13	6.00
7	4	0.13	32	1	7.00	7.00
	32	1.00				



Levels(m)	h(m)
1	5
3	10
4	5
6	8
7	4
	32

Contrast stretching & histogram equalization

(a)

(b)

(c)

(left): input image.
(middle): contrast stretch of image. (more sensitive to outliers)
(right): histogram equalization of image. (less sensitive to outliers)

Local histogram equalization

- global histogram equalisation resuls in better value distribition (i.e. bit allocation) *over the entire image*
- Sometimes, the goal is to improve *regional* value distribution
- reasons:
  - make *local* features visible
  - counter locally-varying illumination

Local histogram equalization

- global histogram equalisation resuls in better value distribition (i.e. bit allocation) *over the entire image*
- Sometimes, the goal is to improve *regional* value distribution
- reasons:
  - make *local* features visible
  - counter locally-varying illumination
  - treat different areas of the same intensity differently ($\rightarrow$ context information)
- regional filters (e.g. histogram equalization) require a *neighbourhood*

Local histogram equalization

Approach:

- iterate over all pixels $x \in [0..M], y \in [0..N]$

Local histogram equalization

Approach:

- iterate over all pixels $x \in [0..M], y \in [0..N]$
- calculate local histogram $h(\mathcal{N}_{xy})$ (ex.: $N(\mathcal{N}_{xy}) = 9$ for $\mathcal{N}_{xy} = 3 \times 3$)

Local histogram equalization

Approach:

- iterate over all pixels $x \in [0..M], y \in [0..N]$
- calculate local histogram $h(\mathcal{N}_{xy})$ (ex.: $N(\mathcal{N}_{xy}) = 9$ for $\mathcal{N}_{xy} = 3 \times 3$)
- set $\tilde{f}(x, y)$ by calculating $\mathcal{O}(f(x, y), \mathcal{N}_{xy})$:

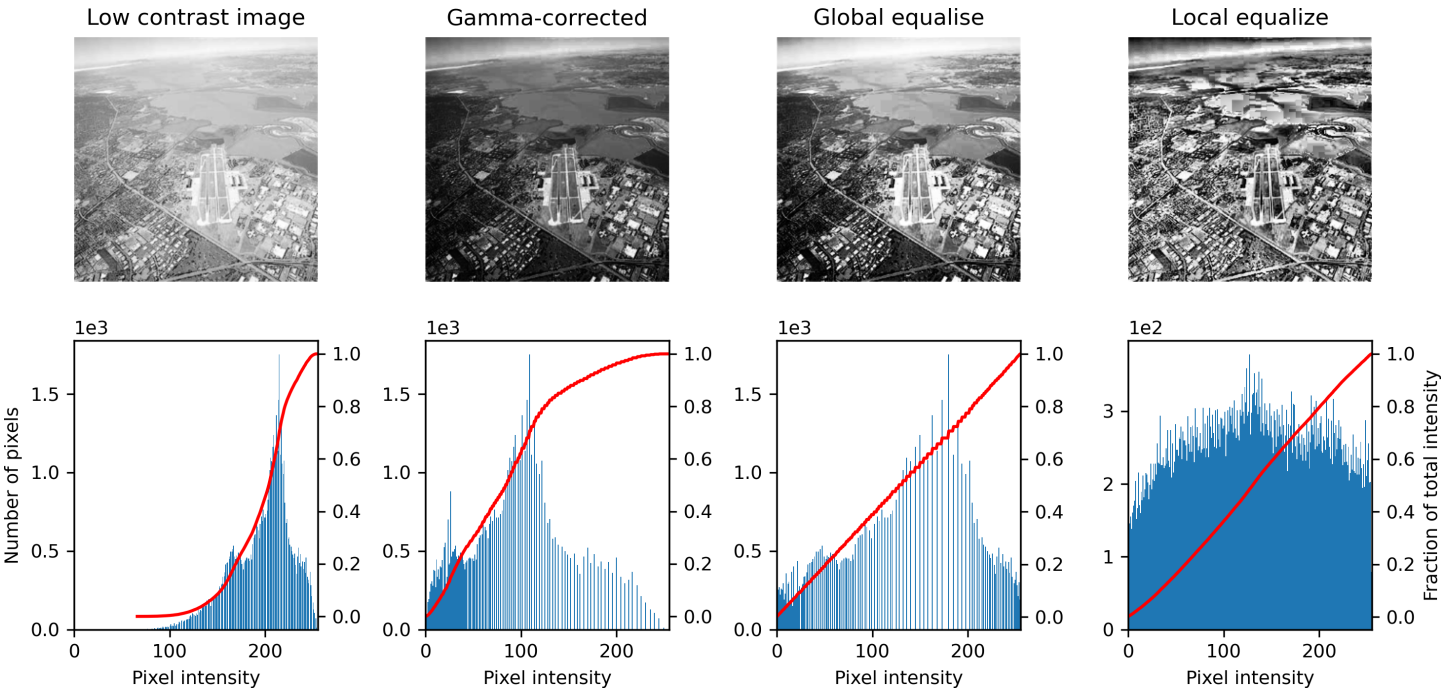
$$\mathcal{O}(f(x, y), \mathcal{N}_{xy}) = (L - 1)\ \mathrm{P}(f(x, y)); \mathrm{P}(\ell) = \frac{1}{N(\mathcal{N}_{xy})} \sum_{m=0}^{\ell} h(\mathcal{N}_{xy})$$

Local histogram equalization

Approach:

- iterate over all pixels $x \in [0..M], y \in [0..N]$
- calculate local histogram $h(\mathcal{N}_{xy})$ (ex.: $N(\mathcal{N}_{xy}) = 9$ for $\mathcal{N}_{xy} = 3 \times 3$)
- set $\tilde{f}(x, y)$ by calculating $\mathcal{O}(f(x, y), \mathcal{N}_{xy})$:

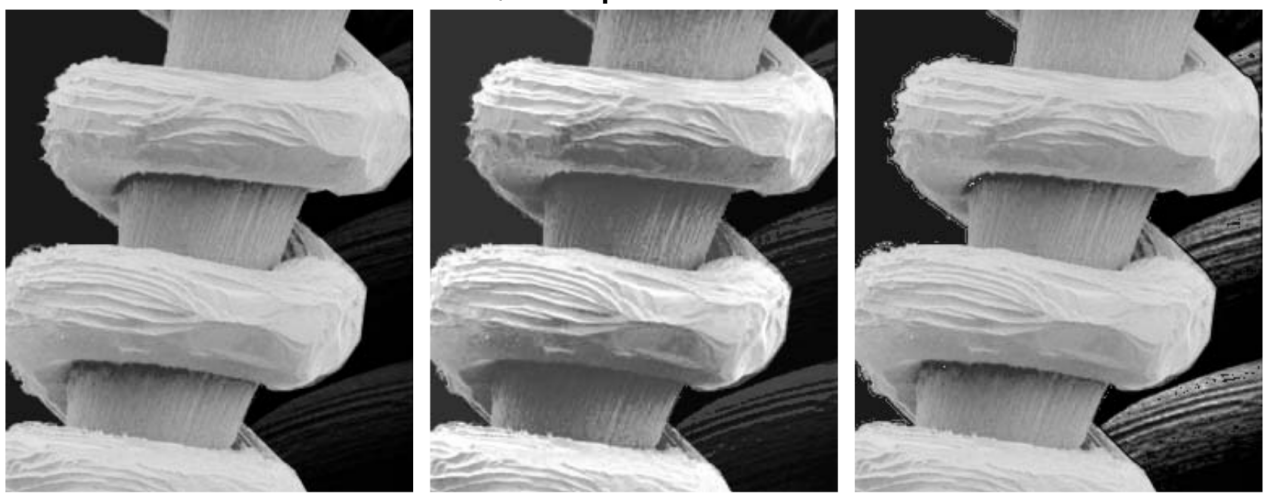
$$\mathcal{O}(f(x,y),\mathcal{N}_{xy}) = (L-1)\ \mathrm{P}(f(x,y)); \mathrm{P}(\ell) = \frac{1}{N(\mathcal{N}_{xy})} \sum_{m=0}^{\ell} h(\mathcal{N}_{xy})$$



Inspiration for more ...

- You like image enhancement ?
- You wanna try it out, and explore more options ?

Have a look into the course book, chapter *3.3ff*



a b c

FIGURE 3.27 (a) SEM image of a tungsten filament magnified approximately 130×. (b) Result of global histogram equalization. (c) Image enhanced using local histogram statistics. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

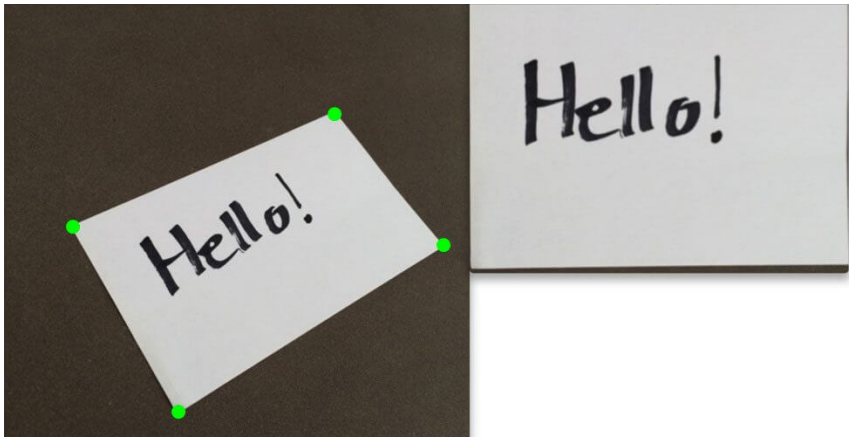
Fundamentals: Geometric image transformation

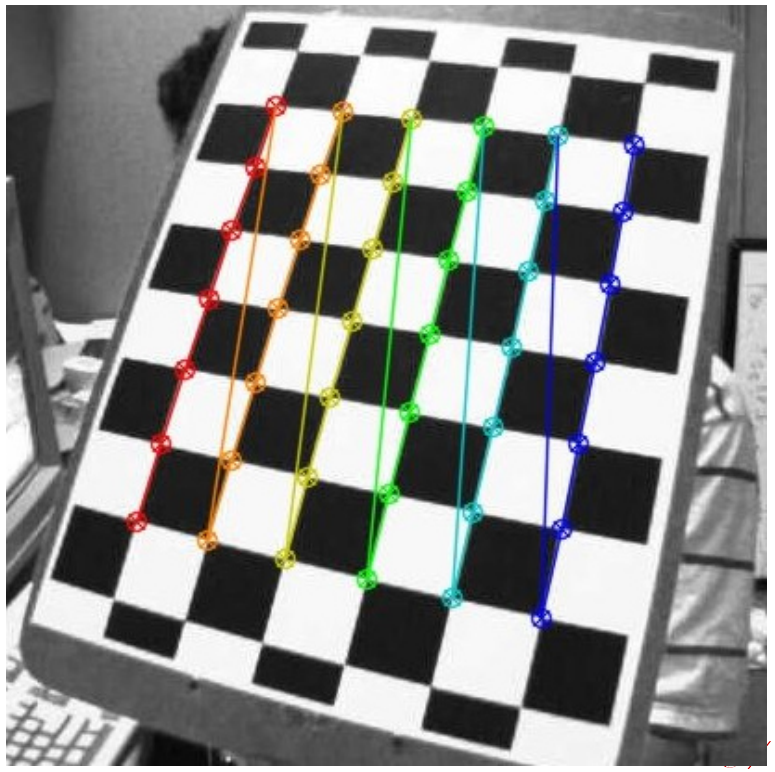
Geometric transformation - Motivation

- common image operations: zoom in or out; horizon alignment (landscape photos); image centering

Geometric transformation - Motivation

- common image operations: zoom in or out; horizon alignment (landscape photos); image centering
- advanced operations: image calibration; correcting lense distortions (in photos); image projection





Geometric transformation - Motivation

- common image operations: zoom in or out; horizon alignment (landscape photos); image centering
- advanced operations: image calibration; correcting lense distortions (in photos); image projection & registration

Generally: moving original pixel coordinates (v, w) to a target coordinate system as (x, y) via a transformation function $\mathbf{T}$

$$(x, y) = T\{(v, w)\}$$

Example - "shrink / zoom-out": $(x, y) = (\frac{1}{2}v, \frac{1}{2}w)$

Linear transformation

- transformations per pixel and coordinate component are *tedious* and *slow*
- improvement: matrix notation

$$x = t_x v \quad \text{becomes } [x, y] = [v, w] \begin{bmatrix} t_x & 0 \\ 0 & t_y \end{bmatrix}$$
$$y = t_y w$$

- example shrinking: $[x, y] = [v, w] \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$

Linear transformation

- transformations per pixel and coordinate component are *tedious* and *slow*
- improvement: matrix notation

$$x = t_x v \quad \text{becomes } [x, y] = [v, w] \begin{bmatrix} t_x & 0 \\ 0 & t_y \end{bmatrix}$$
$$y = t_y w$$

- example shrinking: $[x, y] = [v, w] \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$
- Linear cooordinate combination (general 2D case):

$$x = t_{11} v + t_{21} w$$
$$y = t_{12} v + t_{22} w$$

becomes

$$[x, y] = [v, w] \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \text{ or } \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{21} \\ t_{12} & t_{22} \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

Linear transformation

Linear cooordinate combination (general 2D case):

$$[x, y] = [v, w] \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}$$

Transforms that are linear coordinate combinations:

- scale
- skew
- rotate
- reflect
- rotate

Audience Q: How about moving (i.e. translating) pixels ?

Affine transformation & homogeneous coordinates

- cannot represent a scalar coordinate offset as linear combination of coordinates:

$$x = t_x v + a\_\_$$
$$y = t_y w + b\_\_$$

- solution: augmenting the matrix with scalar dimension:

$$[x, 1] = [v, 1] \left[\begin{array}{c|c} tx & a \\ 0 & 1 \end{array}\right]$$

then: represent all affine transformations as matrix combination of:

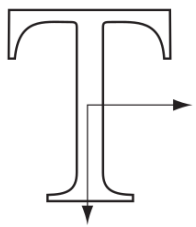
$$[x, y, 1] = [v, w, 1]\mathbf{T}$$

Geometric transformation - scaling

Scaling

$$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x = c_x v$$
$$y = c_y w$$

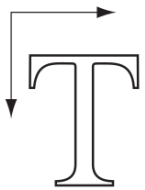


Geometric transformation - translation

Translation

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

$$x = v + t_x$$
$$y = w + t_y$$

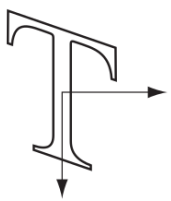


Geometric transformation - shear

Shear (vertical)

$$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

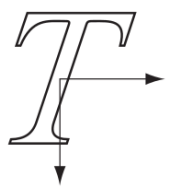
$$x = v + s_v w$$
$$y = w$$



Shear (horizontal)

$$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x = v$$
$$y = s_h v + w$$

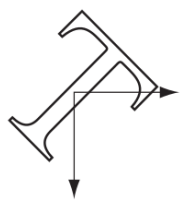


Geometric transformation - rotation

Rotation

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x = v\cos\theta - w\sin\theta$$
$$y = v\cos\theta + w\sin\theta$$



Geometric transformation chains

Consider image below:

Original

Rotated

Audience Q: Is it the result of $T = \begin{bmatrix} cos(15^\circ) & sin(15^\circ) & 0 \\ -sin(15^\circ) & cos(15^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix}$?

Geometric transformation chains

Consider image below:

Original

Rotated

Audience Q: Is it the result of $T = \begin{bmatrix} cos(15^\circ) & sin(15^\circ) & 0 \\ -sin(15^\circ) & cos(15^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix}$?

Issue: rotation center = coordinate center $[x, y] = [0, 0]$

Geometric transformation chains

Consider image below:

Original

Rotated

Audience Q: Is it the result of $T = \begin{bmatrix} cos(15^\circ) & sin(15^\circ) & 0 \\ -sin(15^\circ) & cos(15^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix}$?

Issue: rotation center = coordinate center $[x, y] = [0, 0]$

Result: $T =$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ c_x & c_y & 1 \end{bmatrix} \begin{bmatrix} cos(-15^\circ) & sin(-15^\circ) & 0 \\ -sin(-15^\circ) & cos(-15^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -c_x & -c_y & 1 \end{bmatrix}$$

Remember: 1^{st} transformation done comes *last* in the matrix chain

...

Geometric transformation - projective transform

Perspective project - see an image from another *vantage point*

Define input quadrilateral

Define base quadrilateral

After projective transformation

Cropped image

Geometric transformation - projective transform

Perspective project - see an image from another *vantage point*

- Euclidean transformations (i.e. rotate, translate) preserve: Distance, Angles, Parallelism
- Rigid transformations (i.e. scale, rotate, translate) preserve: Angles, Parallelism
- Affine transformations preserve only parallelism
- (Perspective) Projection ? *straight lines* (i.e. vantage lines)

Geometric transformation - projective transform

Perspective project - see an image from another *vantage point*

- matrix form: 9 unknows, 8 free parameters

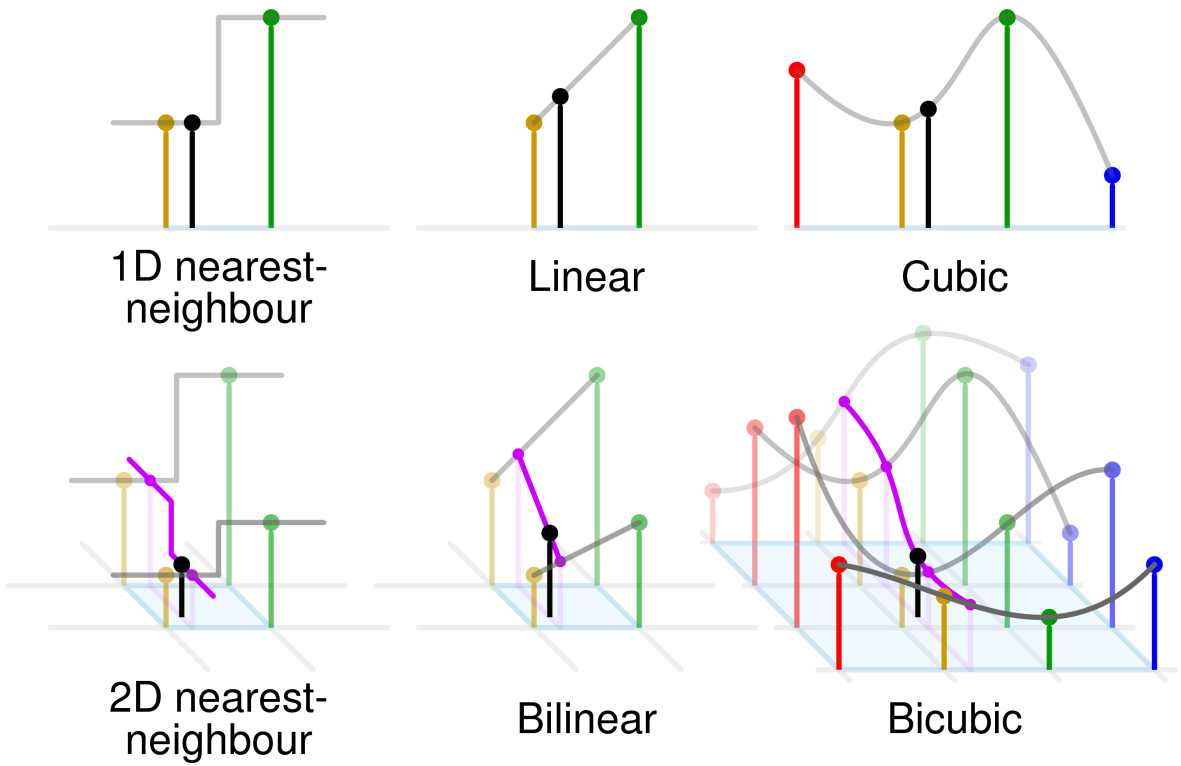
$$S' = ST \rightarrow \begin{bmatrix} x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \\ x'_3 & y'_3 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$x' = a_{11}x + a_{12}y + a_{13}$$
$$y' = a_{21}x + a_{22}y + a_{23}$$
$$1 = a_{31}x + a_{32}y + a_{33}$$

- Challenge:
  detect $[x', y']$ & $[x, y]$ so accurately that $S' = ST$ can be solved as linear system $y = Ax$

Interpolation

- Result of coordinate transform: sub-pixel image coordinate
- task: determine value at inter-pixel positions
- You remember the previous lecture ?



Forward- and inverse mapping

Transformation order:

1. Coordinate transform of all pixels
2. Value interpolation at new pixel positions

Trivial idea: *forward mapping* → for each source pixel, determine its new position and value.

Forward- and inverse mapping

Transformation order:

1. Coordinate transform of all pixels
2. Value interpolation at new pixel positions

Trivial idea: *forward mapping* → for each source pixel, determine its new position and value. Issues:

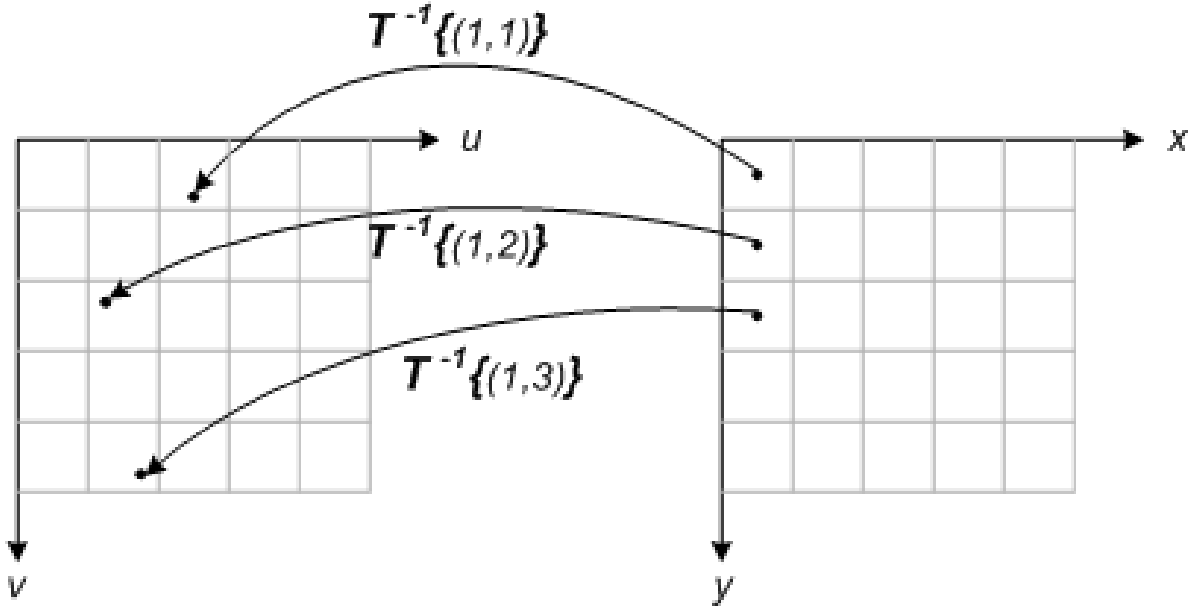
- N source pixels → 1 destination pixel, require weighted average contribution
- Some source pixels map outside destination image boundaries
- Some destination pixels do not have a source pixel correspondence (i.e. empty pixels

Forward- and inverse mapping

Transformation order:

1. Coordinate transform of all pixels
2. Value interpolation at new pixel positions

Common approach: *inverse mapping* → for each target pixel, determine its original position and value in source image.

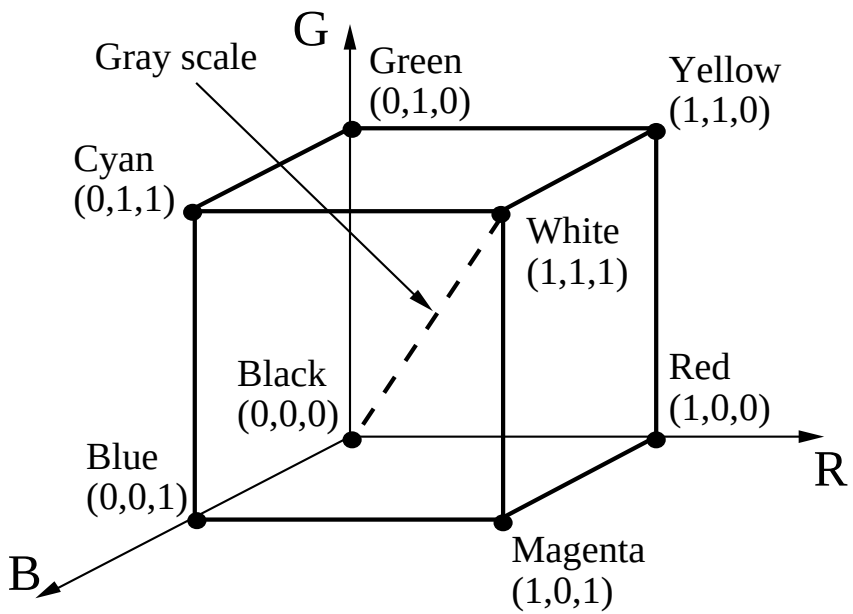


Coloured images

Motivation of colours

Obviously, photos are just half as fun without colour ...

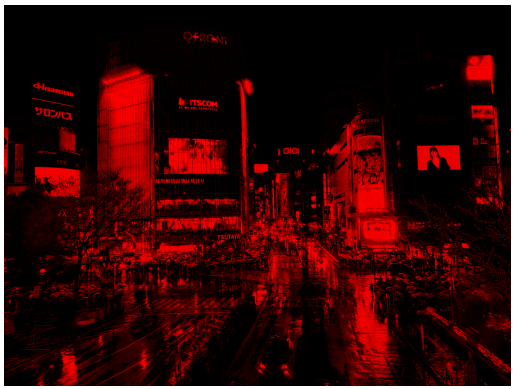
RGB color model

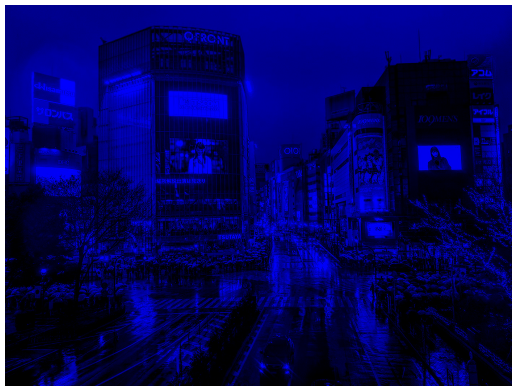


- any color is written as a sum of the primary colors R(ed), G(reen) and B(lue):

$$Color = r\,R + g\,G + b\,B, \quad r, g, b \in [0, 1]$$

RGB color model





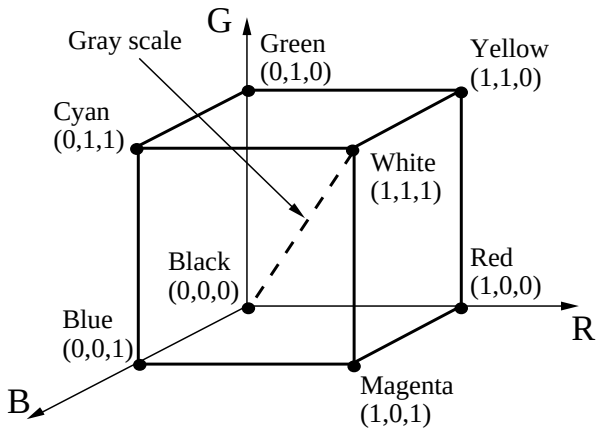
split 1x 3-channel image in 3x 1-channel images (+colour-code)

RGB color model

split 1x 3-channel image in 3x 1-channel images (+colour-code)

- *additive* colour model:
  - channel intensity *illuminates* the pixel further
  - base: lights off, i.e. black
  - saturation: full lights on, i.e. white

CMY model



- colour is sum of the primary colours C(yan), M(agenta) and Y(ellow):

$$Color = c\,C + m\,M + y\,Y,$$

- *subtractive* colour model:
  - channel intensity *attenuation* light by material (e.g. paper)
  - base: no attenuation, i.e. white
  - saturation: full attenuation, i.e. black

Colour conversion: RGB to CMY

- *Linear* transformation:

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

  (interchanging colours across the main diagonals)
- CMY to CIE: apply CMY to RGB followed by RGB to CIE.

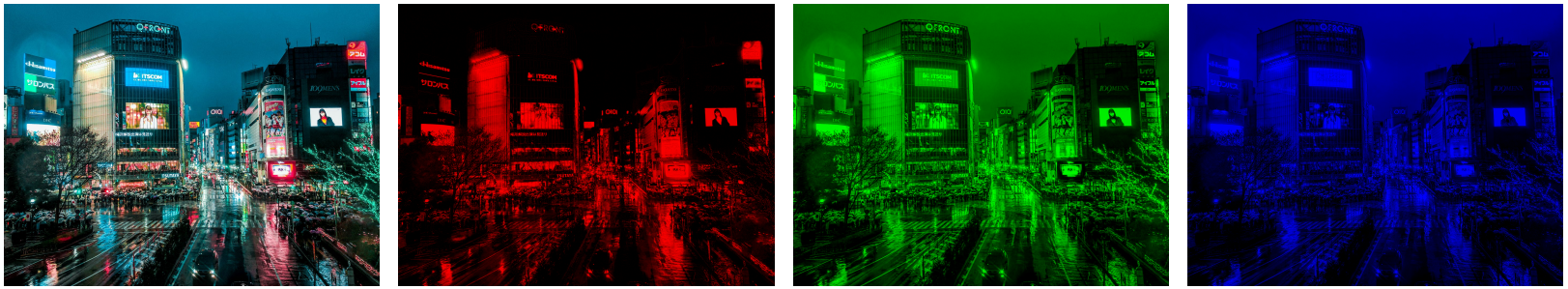
Colour conversion: RGB to CMY

- *Linear* transformation:

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

  (interchanging colours across the main diagonals)
- CMY to CIE: apply CMY to RGB followed by RGB to CIE.
- What is CIE ? Good that you ask ...

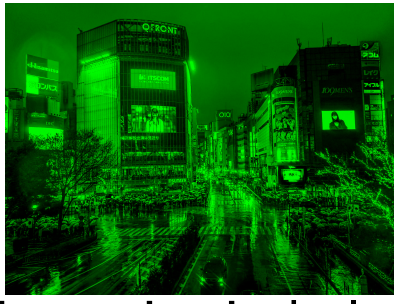
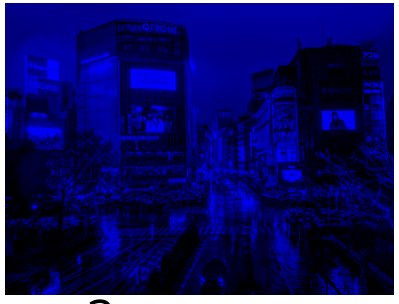
Colour sensitivity



Do you recognize the green-red intensity imbalance ?

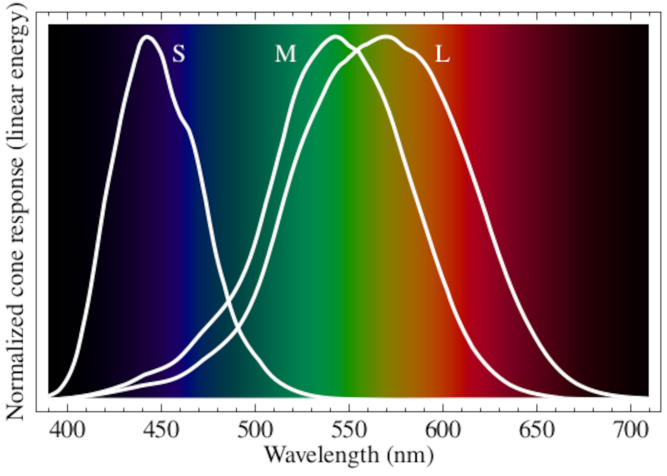
Colour sensitivity

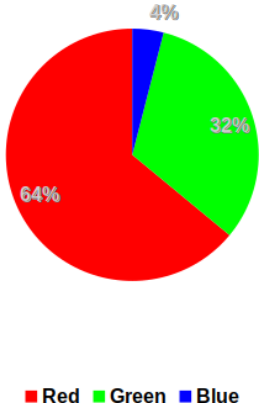


Do you recognize the green-red intensity imbalance ?

Reason: human physiology.





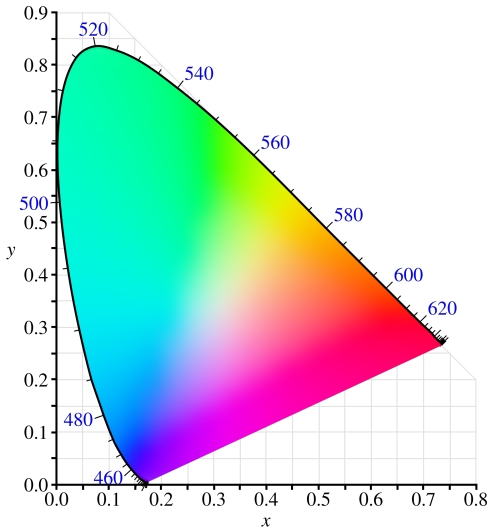
left: EM spectrum - cone distribution. right: cone amount per band.

→ Doesn't mean we see different colours more ...

but does mean we can distinguish less levels on comparison.

CIE colour space

Result: new colour space accounting for conal distribution



CIE 1927 colour space

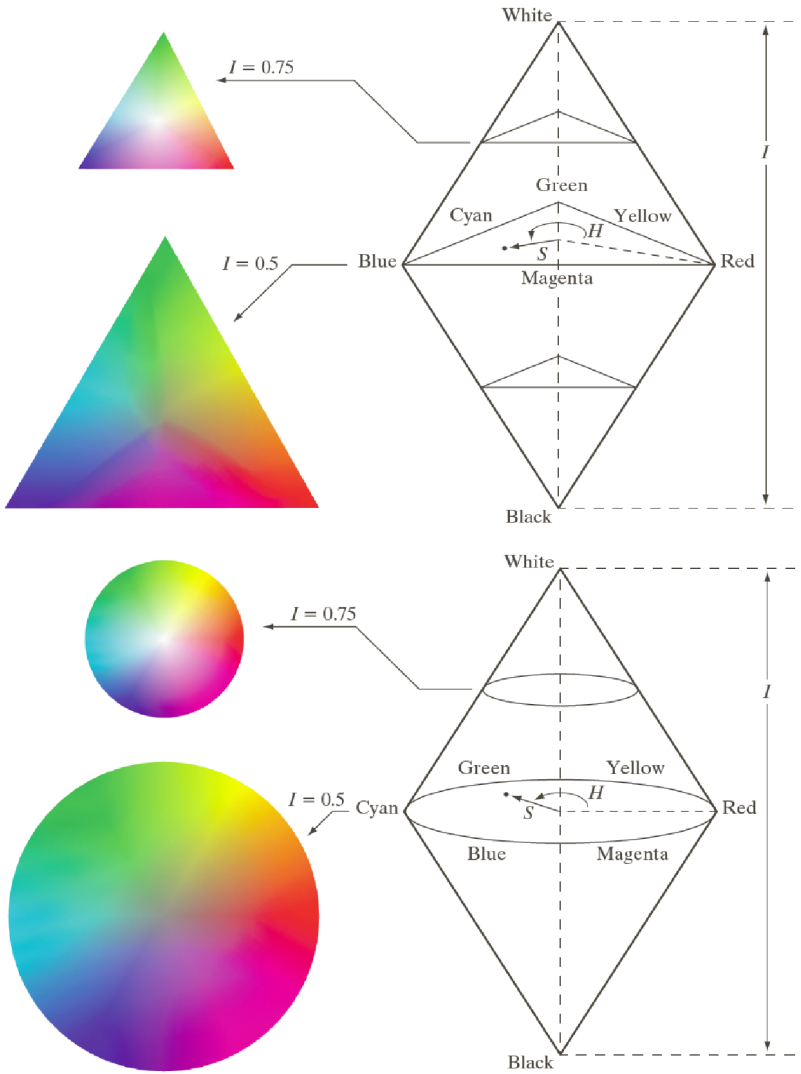
- accounts for cone distribution and sensitivity
- issue: RGB from-/to colour conversion is non-linear
  $\longrightarrow$ depends on look-up data (i.e. ICC profiles)
- use in raw image formats
  (e.g. smartphone *digital negative (DNG)* via CIE XYZ)

HSI model

- start from a pure color = *hue*, then add black to obtain *shades*, or white to obtain *tones* of that color
- Parameters: **Hue** (a pure color), **Saturation** (purity of the color), and **Intensity** (intensity of a color).
- HSI coordinates can be *converted* to RGB coordinates, and vice versa, but not by a simple linear transformation.
- Related to HSV (Hue-Saturation-Value) color model

HSI model

I along vertical axis, H an angle around this axis, S radial distance from it



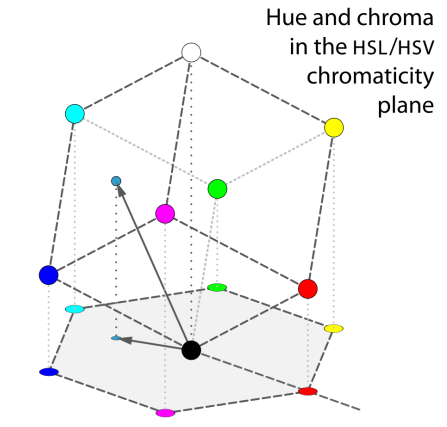
Conversion RGB to HSI

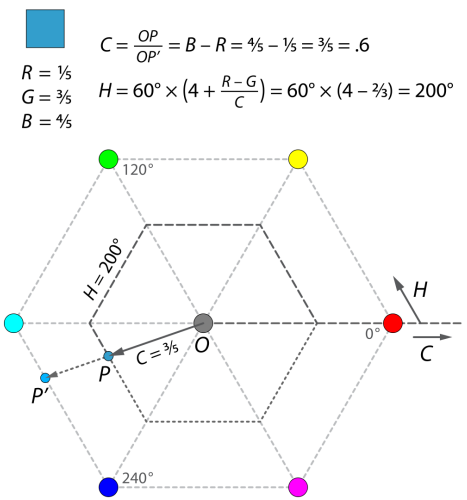
- difference *intensity*, *value* and *light* → cone shape
- step 1: get *Hue*

$$
\begin{aligned}
M &= max(R, G, B) \\
m &= min(R, G.B) \\
C &= M - m
\end{aligned}
$$

$$
H' = \begin{cases}
\text{undefined}, & \text{if } C = 0 \\
\frac{G-B}{C} \text{mod} 6, & \text{if } M = R \\
\frac{B-R}{C} + 2, & \text{if } M = G \\
\frac{R-G}{C} + 4, & \text{if } M = B
\end{cases}
$$

$$
H = 60^\circ \times H'
$$

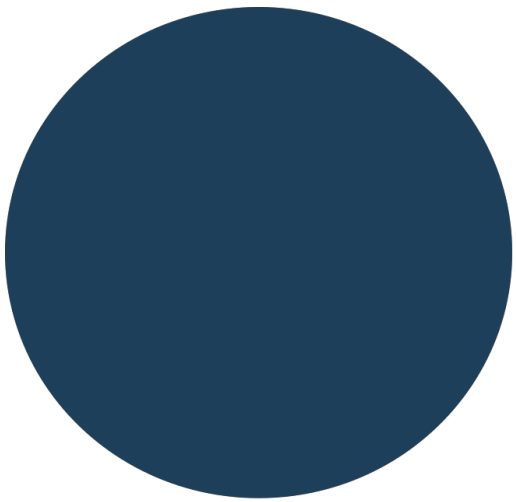




Conversion RGB to HSI

- difference *intensity*, *value* and *lightness* → cone shape
- step 2: get I / V / L

$$
\begin{aligned}
I &= avg(R, G, B) &&= \frac{1}{3}(R + G + B) \\
V &= max(R, G.B) &&= M \\
L &= mid(R, G.B) &&= \frac{1}{2}(M + m)
\end{aligned}
$$



R = 30; G = 63; B = 90

intensity I = 61
value V = 90
light L = 60

[0..255]

Conversion RGB to HSI

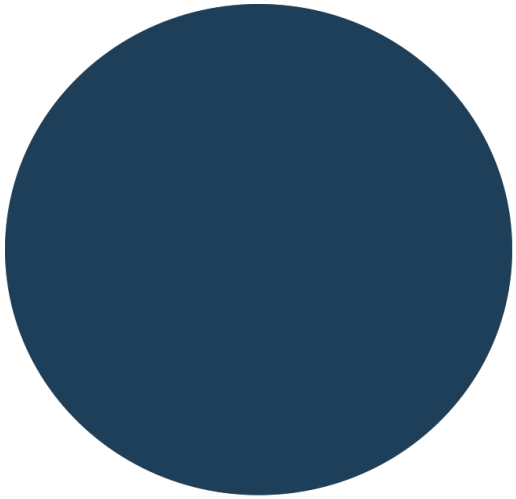
- difference *intensity*, *value* and *lightness* → cone shape
- for *HSI*/*HSL*: convert $\{C, M, m, L, I\}$ to $[0..1]$
- step 3: get *Saturation*

$$S_I = \begin{cases} 0, & \text{if } I = 0 \\ 1 - \frac{m}{I}, & \text{otherwise} \end{cases}$$

$$S_V = \begin{cases} 0, & \text{if } V = 0 \\ \frac{C}{V}, & \text{otherwise} \end{cases}$$

$$S_L = \begin{cases} 0, & \text{if } L = 1 \text{ or } L = 0 \\ \frac{C}{1-|2L-1|}, & \text{otherwise} \end{cases}$$

Last: convert S and I/L to $[0..100]$



R = 30; G = 63; B = 90

S (in HSI) = 0.51
S (in HSV) = 0.67
S (in HSL) = 0.49

[0..1]

Compressed colour spaces - YUV/YC_BC_R family

- application to real-time colour image capture- & transmission
- requirements and critique of previous spaces
  - high throughput, minimal data → RGB / CYMK ?
  - fast digital conversion → HSI / HSV / HSL ?
  - reduced data, acceptable bitrate → CIE ?

Compressed colour spaces - YUV/YC_BC_R family

- application to real-time colour image capture- & transmission
- requirements and critique of previous spaces
  - high throughput, minimal data → RGB / CYMK ?
  - fast digital conversion → HSI / HSV / HSL ?
  - reduced data, acceptable bitrate → CIE ?
- result: common building blocks for real-time applications
  - split luminance (Y) from chrominance channels
  - linear-algebra use (→ conversion- & quantization matrices)
  - chromatic subsampling

Compressed colour spaces - YUV/YC_RC_B family

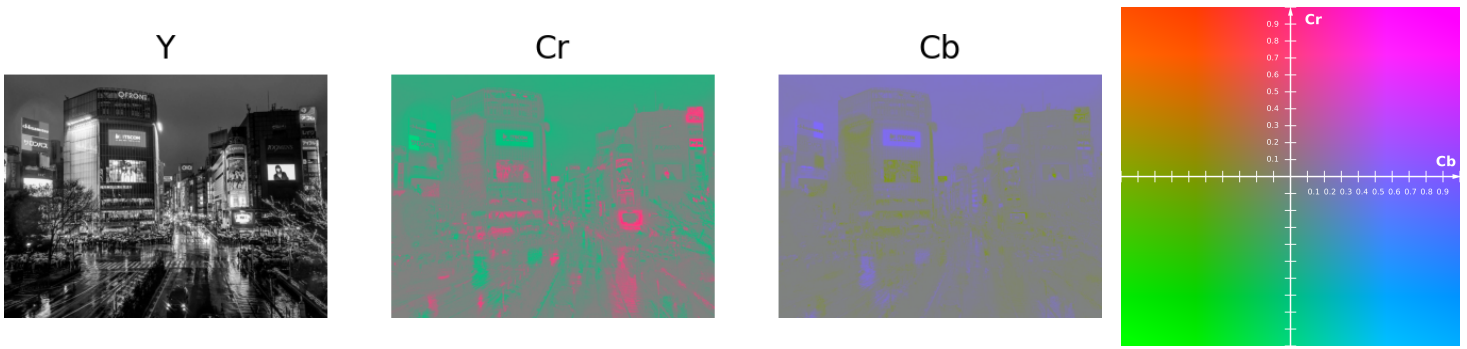
- perceptual / physiological motivation: rods >> cones
  $\rightarrow$ split luma (Y) and chroma
- chroma: green as prime contribution, with blue-chroma (C_B) and red-chroma (C_R) as difference factor

Compressed colour spaces - YUV/YC_RC_B family

- perceptual / physiological motivation: rods >> cones
  $\rightarrow$ split luma (Y) and chroma
- chroma: green as prime contribution, with blue-chroma (C_B) and red-chroma (C_R) as difference factor

$$Y' = K_R R' + K_G G' + K_B B',$$

with K_x being free parameters for colour conversion



Compressed colour spaces - YUV/YC_RC_B family

- example definition (ITU-R BT.601 standard):

$$
\begin{aligned}
Y' &= 16+ && (65.481R'+128.553G'+ && 24.966B') \\
C'_B &= 128+ && (-37.797R'+74.203G'+ && 112.0B') \\
C'_R &= 128+ && (112R'+93.786G'+ && 18.214B')
\end{aligned}
$$

Compressed colour spaces - YUV/YC_RC_B family

- example definition (ITU-R BT.601 standard):

$$\begin{aligned} Y' &= 16+ & (65.481R'+128.553G'+ & 24.966B') \\ C'_B &= 128+ & (-37.797R'+74.203G'+ & 112.0B') \\ C'_R &= 128+ & (112R'+93.786G'+ & 18.214B') \end{aligned}$$

- advantage: can be reforumated in linear-algebraic form $\mathbf{y} = \mathbf{Ax} + \mathbf{t}$
- consider SIMD & vectorized PU's nowadays: $\mathbf{y} = \mathbf{Ax} + \mathbf{t}$ is `fma` – operation $\rightarrow$ hardware-accelerated

Compressed colour spaces - YUV/YC_RC_B family

- *how to do compression in* YC_BC_R *?*

Compressed colour spaces - YUV/YC_RC_B family

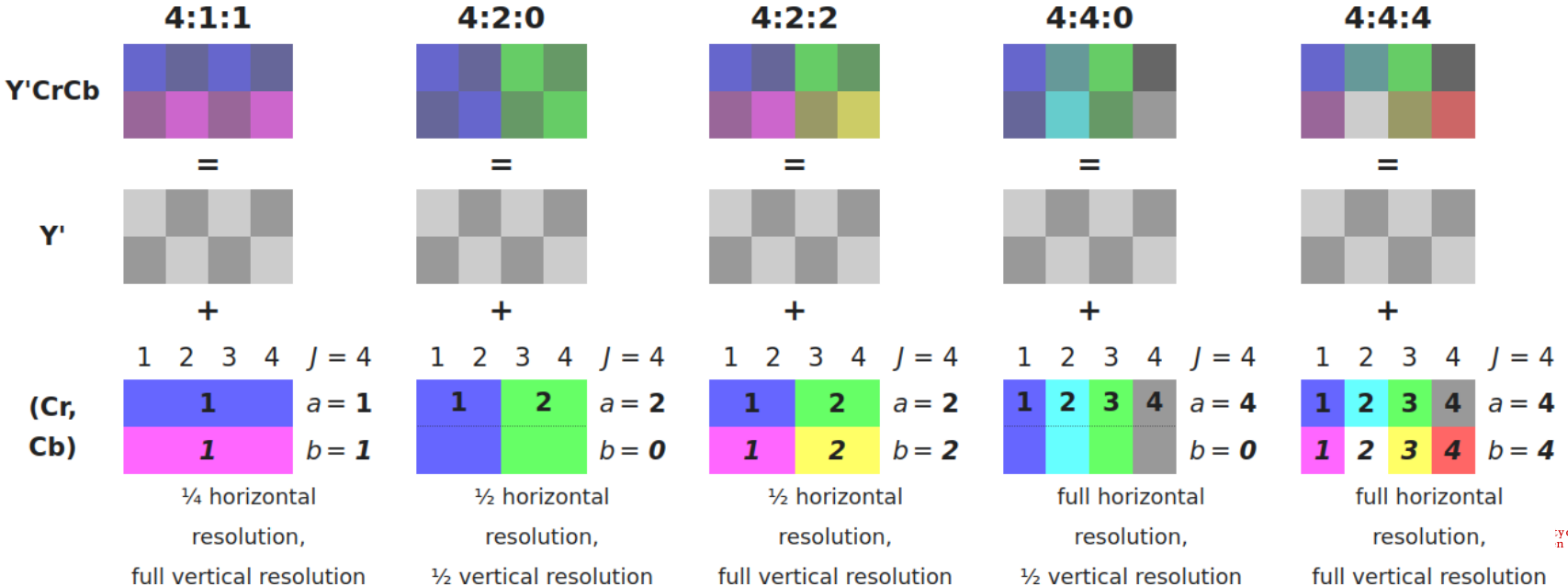
- *how to do compression in* YC_BC_R *?*
- perceptual rationale: humans less sensitive to colour changes
- consequence: less samples in C_B and $C_R \rightarrow$ interleave
- idea: no. pixel samples in Y:C_R (column):C_B (row), e.g. 4:2:2

Compressed colour spaces - YUV/YC_RC_B family

- *how to do compression in* YC_BC_R *?*
- perceptual rationale: humans less sensitive to colour changes
- consequence: less samples in C_B and $C_R \rightarrow$ interleave
- idea: no. pixel samples in Y:C_R (column):C_B (row), e.g. 4:2:2 (compression rate ?)

Compressed colour spaces - YUV/YC_RC_B family

- *how to do compression in* YC_BC_R *?*
- perceptual rationale: humans less sensitive to colour changes
- consequence: less samples in C_B and C_R → interleave
- idea: no. pixel samples in Y:C_R (column):C_B (row), e.g. 4:2:2 (compression rate ?)
- specific packing can differ - example:



That's it for this week!