African Institute for Mathematical Sciences — Cameroon
Structured Master's, Data Science Stream · Computer Vision

From Light to Numbers

Week 1 · Day 2: Image Formation, Sampling, Quantisation and Colour

Yusuf Brima, Ph.D.

Introduction to Computer Vision · AIMS Cameroon, 2026–2027

Today

1. **Image formation**: how light from a 3D scene lands on a 2D sensor
2. **Sampling**: how many pixels, and what goes wrong with too few
3. **Quantisation**: how many grey levels, and how big an image is
4. **Colour**: RGB, grayscale, HSV and other colour spaces
5. **Images in code**: arrays, types, the classic bugs, normalisation, neighbourhoods

By the end of today you should be able to

- project a 3D point with the pinhole model;
- explain sampling vs. quantisation, and predict aliasing;
- compute storage, grayscale and HSV values by hand;
- choose a colour space for a task;
- avoid the classic image-loading bugs.

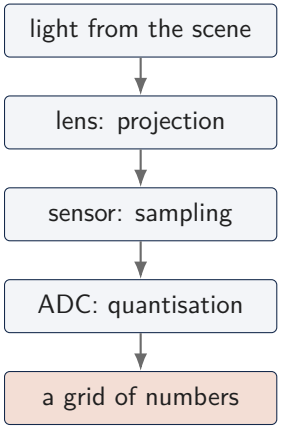
Where we are

Yesterday

- Vision turns images into descriptions.
- It is hard: pose, occlusion, scale, light, clutter.
- The eye samples light with three cone types and filters it already in the retina.

Today

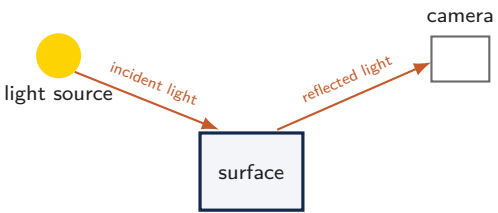
- Before we can *interpret* an image, we must understand exactly what the numbers *are*.
- Every choice made by the camera (lens, sensor, pixel count, bit depth, colour filters) shapes what any algorithm can later see.



Part 1

Image formation: from a 3D scene to a 2D image

What a camera actually measures

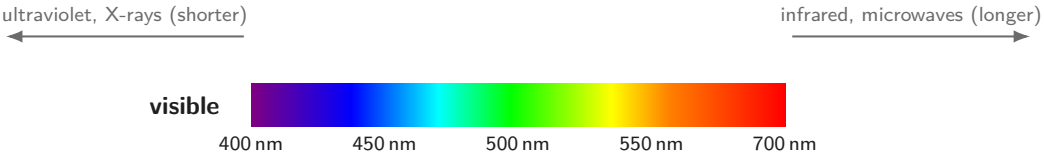


- Light leaves a source, bounces off surfaces, and some of it enters the camera.
- Each pixel measures **how much light arrived** from one small direction, during the exposure.
- That amount depends on the *light*, the *surface* (colour, shininess) and the *geometry* (angles, distance).

Intuition

A pixel value is a *measurement of light*, not a property of the object. A white shirt in shadow and a grey shirt in sunlight can give the same pixel. (Remember the checker-shadow illusion from Day 1.)

Light is a wave: the visible spectrum

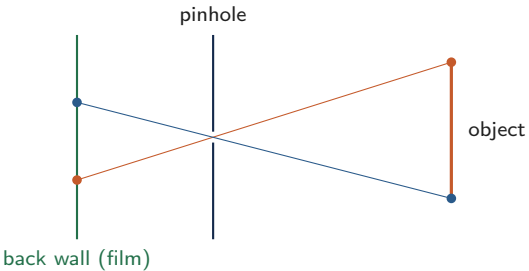


- Visible light is a tiny slice of the electromagnetic spectrum: about 400–700 nm.
- Our three cone types sample this slice at three overlapping bands (Day 1).
- Ordinary cameras copy the eye: three filters, red, green, blue.

Practitioner's Note

"Images" in vision go far beyond visible light: X-rays (radiography), near-infrared (crop health from satellites), thermal infrared (night surveillance, fever screening), radio waves (MRI, radar). The maths is the same; only the physics of the sensor changes.

The simplest camera: a box with a hole

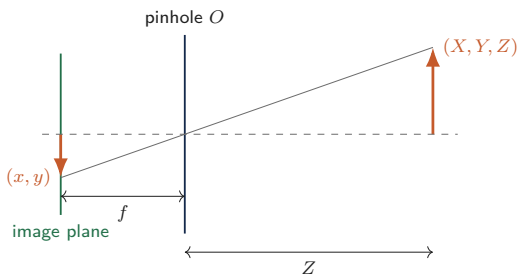


- Each point of the object sends light in all directions; the hole lets through *one ray* from each point.
- The rays cross at the hole, so the image is **upside down**.
- Known for over 2000 years as the *camera obscura* ("dark room"); you can build one with a box and foil.

Intuition

One ray per scene point means one scene point per image point: a sharp image with no lens at all. Every real camera is a pinhole camera with a lens added to let in more light.

The pinhole camera model



Perspective projection

$$x = f\,\frac{X}{Z}, \qquad y = f\,\frac{Y}{Z}$$

f: focal length (hole to image plane). Z: depth of the point.

- **Similar triangles**: the small triangle behind the hole has the same shape as the big one in front.
- We usually flip the image plane to the front so the image is upright; the maths is unchanged.

Worked example: how many pixels tall is a person?

A phone photo

A person of height $Y = 1.7\,\text{m}$ stands $Z = 10\,\text{m}$ from a phone camera with $f = 4\,\text{mm}$. Pixels are $1.4\,\mu\text{m}$ wide.

$$y = f\frac{Y}{Z} = 4\,\text{mm} \times \frac{1.7}{10} = 0.68\,\text{mm}$$

$$\frac{0.68\,\text{mm}}{1.4\,\mu\text{m}} = \frac{680}{1.4} \approx 486 \text{ pixels}$$

What this tells us

- At 20 m: 243 pixels. At 100 m: 49 pixels. Size in the image falls as $1/Z$.
- A detector that must find people at 100 m needs to work on objects about 50 pixels tall: the *scale* challenge from Day 1, now with numbers.
- The same formula tells a drone how high it may fly and still see a diseased leaf.

Perspective in a real photograph



Guaka, CC BY-SA 4.0

- Cars of the same real size shrink steadily up the street, as $1/Z$.
- The road edges, kerbs and rooflines are parallel in the world but **converge** in the image towards a *vanishing point*.
- Equal steps along the road become ever smaller steps in the image.

Intuition

Perspective is not a distortion; it is *information*. Your visual system reads depth from it, and so do monocular depth networks (the Depth Anything example on Day 1 used this very photo).

What projection destroys

Depth is lost

(X, Y, Z) and $(2X, 2Y, 2Z) \mapsto \left(f\frac{X}{Z},\ f\frac{Y}{Z}\right)$

Every point on a ray through the pinhole lands on the *same* pixel.

- Lengths and angles are not preserved.
- Parallel lines may meet.
- Circles become ellipses.

What survives

- **Straight lines** stay straight (for an ideal lens).
- **Order along a line** is preserved.
- **Occlusion**: what is in front stays in front.

Practitioner's Note

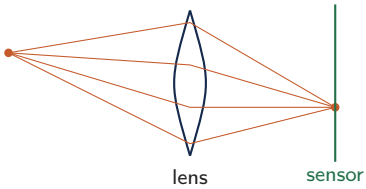
To recover depth we need *more than one view* (stereo, video) or *priors* learned from data. A second camera a few centimetres away is how your two eyes do it.

Why real cameras need lenses

The pinhole dilemma

- **Large hole**: lots of light, but each scene point spreads into a blurry disc.
- **Tiny hole**: sharp, but very dark, and below about 0.3 mm diffraction blurs it again.

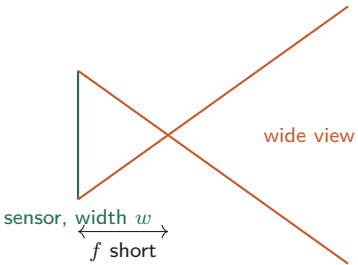
A lens solves both: it collects light over a large opening and bends all rays from one scene point back to one image point.



Intuition

A lens only focuses *one depth* perfectly. Points nearer or farther blur slightly: this is **depth of field**. Portrait photos blur the background on purpose; microscopes have very shallow depth of field, which is why smear images can be partly out of focus.

Focal length and field of view



Field of view

$$\text{FOV} = 2\arctan\left(\frac{w}{2f}\right)$$

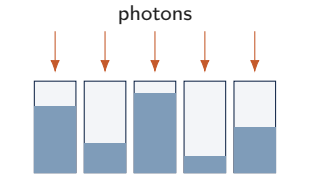
- **Short focal length**: wide angle, objects small (phone main camera, dashcams).
- **Long focal length**: narrow angle, objects large (telephoto, wildlife, surveillance).

A phone camera

Sensor width $w = 5.6\,\text{mm}$, $f = 4\,\text{mm}$: FOV $= 2\arctan(0.7) \approx 70^\circ$. Doubling f to 8 mm: FOV $\approx 39^\circ$, and every object is twice as tall in pixels.

From photons to electrons: the image sensor

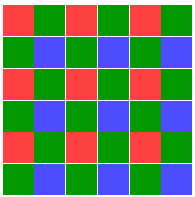
1. The sensor is a grid of **photosites**, millions of tiny light buckets.
2. During the exposure, each photon that hits a site may release an electron.
3. At the end, the charge in each bucket is read out as a voltage.
4. An analogue-to-digital converter (ADC) turns the voltage into an integer.



Intuition

Photon arrival is random, like raindrops on a roof. Even a perfectly uniform grey wall gives slightly different counts in neighbouring buckets: this **shot noise** is fundamental physics, not a defect. It is worst in dim light, which is why night photos look grainy. Day 3 deals with it.

One sensor, three colours: the Bayer mosaic



RGGB Bayer pattern:

50% green, 25% red, 25% blue

- A photosite counts photons but cannot tell their colour.
- So each site gets a tiny colour filter: red, green *or* blue. Each pixel measures **only one** of the three colours.
- **Twice as many green** sites, because green carries most of the brightness detail the eye cares about (Day 1, cones).
- **Demosaicing** estimates the two missing colours at every pixel from its neighbours.

Intuition

Two-thirds of the colour values in every photo you have ever taken were *interpolated*, not measured.

Demosaicing on a real photograph



Minette Lontsie, CC BY-SA 4.0

We simulated the sensor on a crop of a Douala crossroads: kept only the colour each photosite would measure (second panel), stored one number per pixel (third), then demosaiced with OpenCV (fourth).

Practitioner's Note

Look at fine, high-contrast detail such as stripes, text and thin wires: demosaicing produces false colour fringes there. Medical and scientific cameras often use monochrome sensors to avoid this.

Inside the camera: the image-processing pipeline

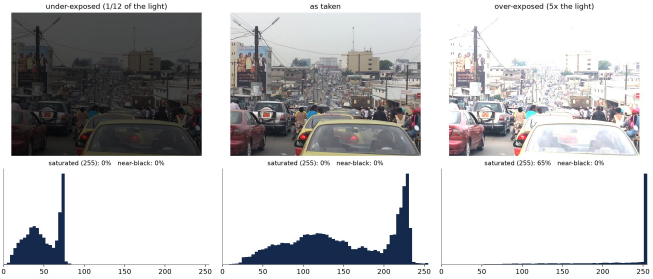


- **White balance**: makes a white wall look white under yellow bulbs or blue shade.
- **Gamma**: stores brightness non-linearly, to match the eye (in a few slides).
- **Denoise, sharpen, compress**: each camera maker does this differently.

Practitioner's Note

Your "original" photo is already heavily processed, differently by every phone brand. A crop-disease model trained on one phone model can lose accuracy on another: an example of **dataset shift**.

Exposure and dynamic range



Guaka, CC BY-SA 4.0

Dynamic range: the ratio between the brightest and darkest light a sensor can record. Outside it, values are **clipped**: too little light gives 0 (crushed shadows), too much gives 255 (blown highlights).

Intuition

Clipping destroys information permanently: every overexposed pixel in the sky is 255, so no algorithm can recover the clouds. The histogram (Day 3) is the quickest way to check exposure.

Part 2

Sampling: how many pixels?

An image is a function, then a matrix

Continuous to discrete

$$I(x, y) \xrightarrow{\text{sample}} I[m, n] \xrightarrow{\text{quantise}} \{0, \ldots, L-1\}$$

$m = 0, \ldots, M-1$ indexes rows; $n = 0, \ldots, N-1$ indexes columns.

- **Sampling**: *where* we measure. A grid of $M \times N$ positions.
- **Quantisation**: *how finely* we record each value. $L = 2^b$ levels for b bits.
- Convention: row m runs top to bottom (y), column n runs left to right (x). NumPy indexes `img[row, col]`.

Graph paper over a painting

Sampling decides how fine the graph paper is. Quantisation decides how many shades of grey you may use to colour each square. Both throw information away, and neither can be undone.

Zooming into a real photo: it is numbers all the way down



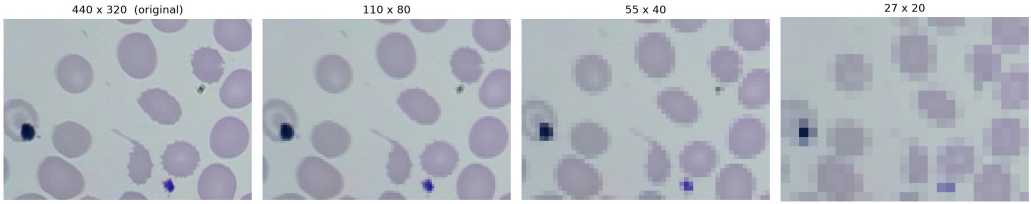
Bdx, CC BY-SA 4.0

A 1.2-megapixel street photo from Yaoundé. Zoom in three times and the smooth scene becomes squares; each square stores three integers, one for red, green and blue.

Intuition

A pixel is not a little square of colour in the world. It is *one sample* of the light that fell on one photosite. Squares are just how we choose to display samples.

Spatial resolution: what can you still see?



OpenFlexure Project, CC BY-SA 4.0

The same blood smear at four resolutions. At 110×80 the parasites are still visible; at 55×40 they are a few dark pixels; at 27×20 the small parasite on the right has merged into a red cell.

Practitioner's Note

The right resolution is set by the *smallest thing you must see*, not by the image. Rule of thumb: the object of interest should span at least a few pixels, ideally ten or more.

Resolution trades detail against cost

512 x 512 samples



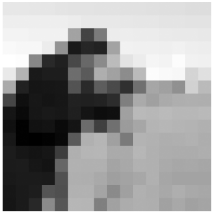
64 x 64 samples



32 x 32 samples



16 x 16 samples

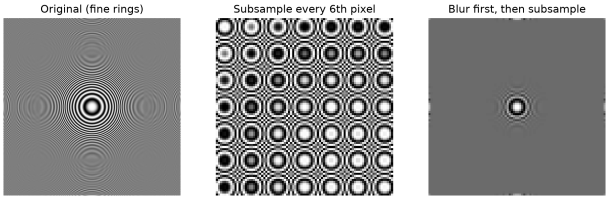


- Halve the resolution in each direction and you have **four times** fewer pixels.
- Memory, storage and compute all grow with $M \times N$.

Practitioner's Note

This is why classification CNNs are trained at 224×224, why gigapixel pathology slides and satellite scenes are cut into *tiles*, and why detectors struggle with tiny distant objects.

Aliasing: when sampling lies



- Sample a fast wave too slowly, and the samples trace out a **slow wave that was never there**.
- The wrong frequency is called an **alias**: a false identity.
- In films, car wheels sometimes seem to spin backwards: the camera samples 24–30 frames per second, too slow for the spokes.

Intuition

Under-sampled fine detail does not vanish: it *disguises itself as coarse detail*, and no algorithm can tell the fake from the real.

Aliasing in a real photo: moiré on a lace curtain



Famartin, CC BY-SA 4.0

Middle: keeping every 7th pixel (`img[::7, ::7]`) turns the fine lace into noisy speckle and false patterns. **Right**: blurring first, then shrinking, gives a softer but honest picture.

Practitioner's Note

You see moiré when you photograph a TV screen, a striped shirt or a fine fence. Datasets resized carelessly contain it, and networks can learn it as if it were real texture.

The sampling theorem

Nyquist–Shannon

A signal whose fastest variation has frequency $f_{\max}$ can be reconstructed exactly from samples taken at rate

$$f_s > 2\, f_{\max}.$$

- In words: you need **at least two samples per cycle** of the finest pattern.
- One sample for the bright part, one for the dark part.

Stripes on a shirt

Black-and-white stripes repeat every 4 pixels in the original image (one cycle = 4 pixels).

- Keep every 2nd pixel: 2 samples per cycle. Borderline.
- Keep every 3rd pixel: fewer than 2 per cycle: the stripes alias into wider, false stripes.

Intuition

The fix is to remove what you cannot represent *before* sampling: a low-pass (blur) filter, Day 3.

Resizing correctly in code

Shrinking (downsampling)

```
small = img[::4, ::4]          # aliases!

small = cv2.resize(img, (w//4, h//4),
          interpolation=cv2.INTER_AREA)   #
    averages first

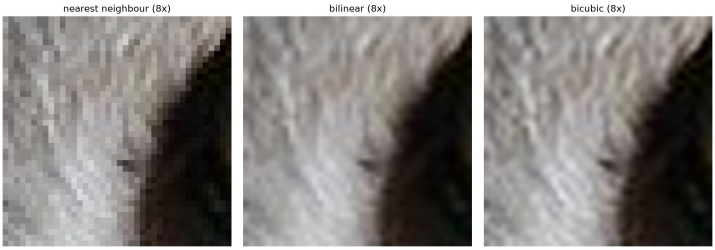
from skimage.transform import rescale
small = rescale(img, 0.25, anti_aliasing=True,
                channel_axis=-1)
```

- `INTER_AREA` averages all pixels that fall into each new pixel: a built-in blur.
- scikit-image blurs with a Gaussian when `anti_aliasing=True`.

Practitioner's Note

CNNs downsample too (strided convolutions and pooling), and can alias: shift a test image by one pixel and the prediction can change. Week 2 returns to this.

Enlarging an image: interpolation



Gourdo 1 at en.wikipedia, Public domain

- **Nearest neighbour**: copy the closest pixel. Blocky, but never invents new values: use it for *label masks*.
- **Bilinear**: weighted average of the 4 nearest pixels. Smooth.
- **Bicubic**: uses 16 neighbours. Sharper edges.

Intuition

Interpolation makes images *bigger*, not *more detailed*. Detail that was never sampled cannot be recovered by averaging. Learned super-resolution (Week 3) instead *guesses* plausible detail from experience.

Part 3

Quantisation: how many grey levels?

Quantisation: rounding brightness to a fixed set of levels

b = 8 bits (L = 256 levels)



b = 4 bits (L = 16 levels)



b = 2 bits (L = 4 levels)



b = 1 bits (L = 2 levels)



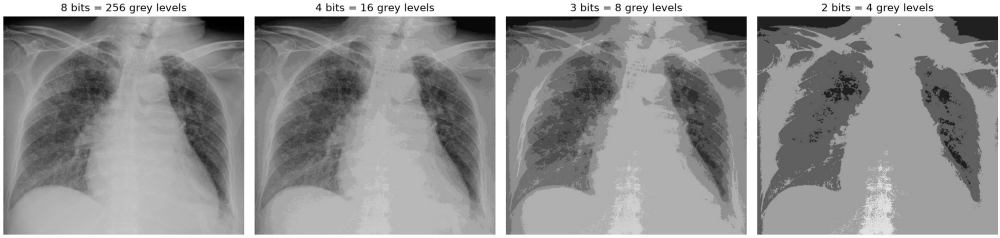
Levels and quantiser

$$L = 2^b, \qquad q(v) = \left\lfloor \frac{v}{256} L \right\rfloor$$

b bits per value give L levels.

- 8 bits (256 levels) is the default for photos and screens.
- Too few levels create **false contours**: smooth gradients become visible steps (banding).

Quantising a chest X-ray



Herreros et al., CC BY 4.0

B.

At 16 levels the X-ray still looks fine to us. At 8 levels the smooth soft tissue breaks into bands. At 4 levels, the fine nodules of miliary TB, the very thing a doctor needs to see, are gone.

Practitioner's Note

Medical images are usually stored with 12 or 16 bits per pixel (4096 or 65 536 levels). Converting a CT scan to 8-bit for a standard image library throws away most of that range. Keep the original bit depth until you know what you need.

Bit depths you will meet

Bits per value	Levels	Where
1	2	binary masks, scanned documents
8	256	photos, video, the web, most CNN inputs
10–12	1024–4096	phone raw files, HDR video, X-ray
16	65 536	CT, MRI, scientific cameras, satellite bands
32 (float)	continuous	computation: after normalisation, inside networks

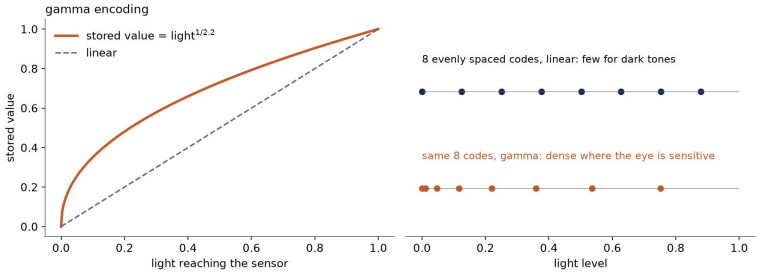
A 12-bit X-ray

$L = 2^{12} = 4096$ grey levels: sixteen times finer than an 8-bit photo.

Intuition

More bits help only if the sensor's *noise* is smaller than one level. Otherwise the extra bits just record noise more precisely.

Gamma: spending the levels where the eye looks



Your eye is far more sensitive to differences between dark tones than between bright ones. Cameras therefore store roughly $\text{light}^{1/2.2}$, not light itself, and screens undo it.

Practitioner's Note

Pixel values in a JPEG are *not* proportional to light. Doubling a pixel value does not mean twice the light. For physically meaningful maths (e.g. averaging exposures) convert to linear first.

How big is an image?

Raw storage

$$\text{bits} = M \times N \times C \times b \qquad \text{(rows} \times \text{columns} \times \text{channels} \times \text{bits per value)}$$

A Full-HD video frame

1920×1080, RGB ($C = 3$), 8 bits:

$1920 \cdot 1080 \cdot 3 = 6\,220\,800$ bytes ≈ 6.2 MB

At 30 frames per second: about 187 MB every second.

A CT slice

512×512, one channel, 16 bits $= 2$ bytes:

$512 \cdot 512 \cdot 2 = 524\,288$ bytes $= 512$ KB

A full scan of 300 slices: about 150 MB.

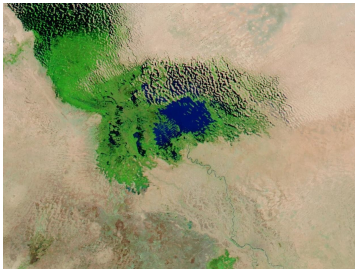
Worked example: one satellite scene

A Sentinel-2 tile

One tile covers 110×110 km at up to 10 m resolution: $10\,980 \times 10\,980$ pixels. The satellite records 13 spectral bands at 16 bits.

If all 13 bands were stored at 10 m:

$$10\,980^2 \times 13 \times 2 \text{ bytes} \approx 3.1 \text{ GB}$$



NASA MODIS Rapid Response, Public domain

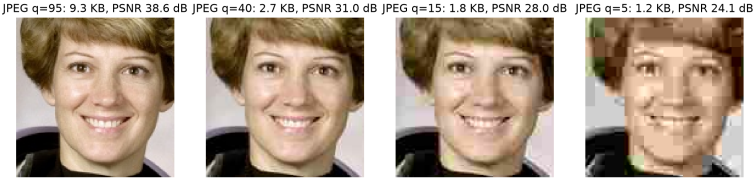
Practitioner's Note

Cameroon is covered by dozens of such tiles, re-imaged every few days. Earth observation is a *big data* problem first: you will tile, compress, and process lazily, never "load the whole image".

Compression and file formats

Format	Compression	Typical use
PNG	lossless	masks, labels, screenshots
JPEG	lossy	photographs
TIFF	either	microscopy, satellite
WebP	either	the web
DICOM	+ patient metadata	clinical imaging
GeoTIFF	+ map coordinates	remote sensing

- **Lossless**: you get back exactly the same numbers (like a ZIP file).
- **Lossy**: throws away what the eye is least likely to miss: fine detail and colour detail.



JPEG: what is thrown away, and why it matters

- JPEG cuts the image into 8×8 blocks and describes each block by its slow and fast *patterns* (frequencies, Day 3).
- It keeps the slow patterns and coarsely rounds the fast ones; it also stores colour at lower resolution than brightness.
- Strong compression shows **blocking** and **ringing** around edges.

Never store labels as JPEG

A segmentation mask has values $\{0, 1, 2\}$. JPEG rounding at boundaries invents values like 1.4 or 0.7, i.e. labels that do not exist. Store masks as PNG.

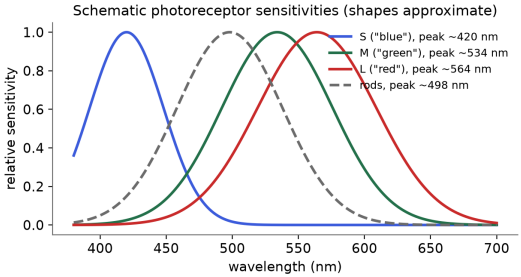
Keep a master copy

Every re-save as JPEG loses a little more. Keep one lossless original of every dataset.

Part 4

Colour

Why three numbers per pixel?

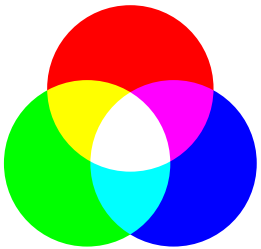


- Light has a whole *spectrum*: an intensity at every wavelength.
- Your eye reduces it to **three numbers**: the responses of S, M and L cones.
- Two different spectra that give the same three responses look *identical* to you (**metamers**).
- So a screen only needs three primaries to fool you into seeing any colour in its range.

Intuition

RGB is a property of human vision, not of light. A bee (with UV receptors) would need a different camera.

Additive colour mixing



- Screens *add* light: red + green = yellow, green + blue = cyan, red + blue = magenta, all three = white.
- An RGB pixel says how much of each primary to emit:

$$(255, 255, 0) \text{ is yellow}, \quad (0, 0, 0) \text{ is black}.$$

- Printing is *subtractive* (inks absorb light): cyan, magenta, yellow, black (CMYK).

Intuition

The yellow taxis of Yaoundé are stored as "lots of red, lots of green, little blue". Keep that in mind for the next slide.

RGB: three stacked images



Bdx, CC BY-SA 4.0

Each channel is a grayscale image. The yellow taxis are **bright in R and G** but **dark in B**. The woman's red dress is bright only in R. The grey road is similar in all three.

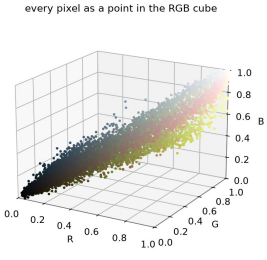
Shape of a colour image

$$I[m, n] = (R, G, B) \in \{0, \ldots, 255\}^3$$

NumPy shape: $M \times N \times 3$ (height, width, channels).

Every pixel is a point in a colour cube





Chroanch, CC BY-SA 4.0

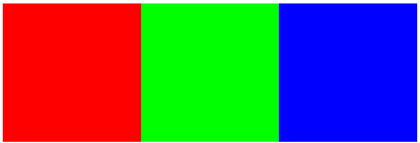
The pixels of the maize photo form a long, thin cloud from black to white. Moving along the cloud mostly changes *brightness*; moving across it changes *colour* (green leaves vs. tan lesions).

Intuition

In RGB, brightness and colour are tangled together along the diagonal. A shadow moves a pixel along the diagonal, not across it. That suggests a better coordinate system: separate brightness from colour.

Grayscale: not a plain average

Pure red, green, blue



Average (R+G+B)/3: 85, 85, 85



BT.601 0.299R+0.587G+0.114B: 76, 150, 29



Luminance (ITU-R BT.601)

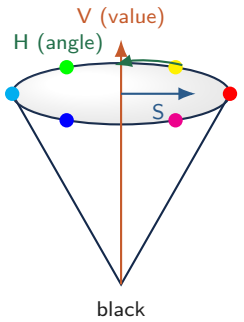
$$Y = 0.299\,R + 0.587\,G + 0.114\,B$$

- The weights follow the eye's sensitivity: green dominates perceived brightness, blue contributes little.
- A plain average would make pure blue look as bright as pure green, which no human agrees with.

One red pixel

$(R, G, B) = (220, 40, 30)$: $Y = 65.8 + 23.5 + 3.4 = 92.7 \approx 93$.
Plain average: 96.7. For a pure blue $(0, 0, 255)$: $Y = 29$, but the average gives 85.

HSV: separating *which colour* from *how bright*

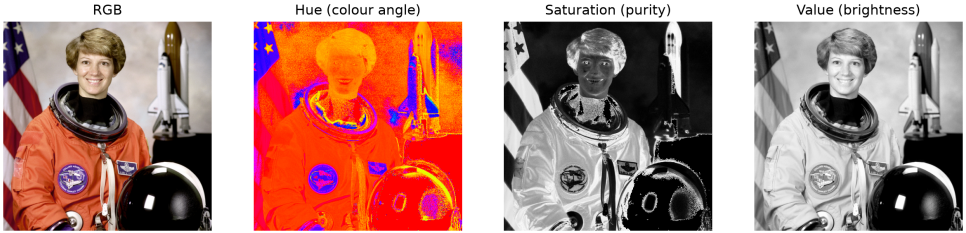


- **H, hue**: *which* colour, as an angle on the colour wheel: red $0°$, yellow $60°$, green $120°$, cyan $180°$, blue $240°$.
- **S, saturation**: how *pure* the colour is. 0 = grey, 1 = vivid.
- **V, value**: how *bright*. 0 = black.

Intuition

HSV is how you would describe colour in words: "a bright, vivid yellow". Shadows mainly change V; the hue stays put.

HSV on a real image



Hue shows *which* colour regardless of brightness. Saturation separates vivid objects from the grey background. Value is close to a grayscale image.

Practitioner's Note

Hue is meaningless for greys ($S \approx 0$) and very noisy for dark pixels ($V \approx 0$). Always threshold on S and V as well as H.

Converting RGB to HSV

RGB $\in [0,1]^3 \to$ HSV

$$V = \max(R,G,B), \quad \Delta = V - \min(R,G,B)$$

$$S = \Delta / V \quad (S = 0 \text{ if } V = 0)$$

$$H = 60^\circ \times \begin{cases} \left(\frac{G-B}{\Delta}\right) \bmod 6 & \text{if } V = R \\ \frac{B-R}{\Delta} + 2 & \text{if } V = G \\ \frac{R-G}{\Delta} + 4 & \text{if } V = B \end{cases}$$

Reading it: V is the largest channel; S is how far the smallest channel falls below it; H says which channel is largest (which third of the wheel) and where between its neighbours.

A yellow taxi pixel

$(R,G,B) = (200, 170, 60)$, divide by 255:

$(0.784, 0.667, 0.235)$.

$V = 0.784, \quad \Delta = 0.784 - 0.235 = 0.549$

$S = 0.549 / 0.784 = 0.70$

$V = R$, so $H = 60^\circ \times \frac{0.667 - 0.235}{0.549} = 47^\circ$

A fairly vivid orange-yellow.

Why HSV is robust to illumination

Scale the light by $k > 0$

A shadow multiplies all three channels by roughly the same k:

$$(R, G, B) \mapsto (kR, kG, kB)$$

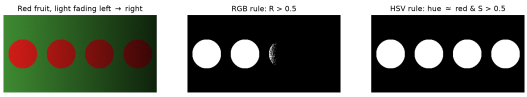
Then $V \mapsto kV$ and $\Delta \mapsto k\Delta$, so

$$S = \frac{k\Delta}{kV} = \frac{\Delta}{V} \quad \text{unchanged.}$$

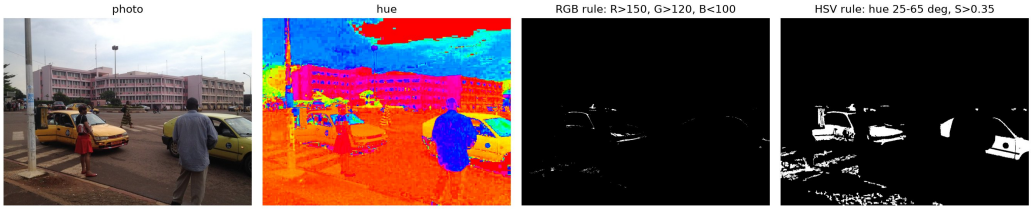
H uses ratios like $\frac{kG-kB}{k\Delta} = \frac{G-B}{\Delta}$: also unchanged.

When the rule breaks

- **Clipping**: if kR would exceed 255 it is stored as 255; the ratios change.
- **Very dark pixels**: Δ is tiny and dominated by noise.
- **Coloured light**: sunset or a yellow bulb scales channels *differently*: hue shifts.



HSV in practice: finding yellow taxis



Bdx, CC BY-SA 4.0

A rule in RGB ("R and G high, B low") finds only the brightest, sunlit parts of the taxis. A rule in HSV ("hue between $25°$ and $65°$, reasonably saturated") finds most of both taxis, including their shaded sides.

Practitioner's Note

But look at the bottom: the reddish-brown laterite road also has a yellow-orange hue. Colour alone is rarely enough: you add shape, size or texture. This is exactly why we will want learned features.

Other colour spaces you will meet

Space	Idea	Typical use
RGB / BGR	sensor-like additive primaries	storage, display, CNN input
HSV / HSL	hue, saturation, brightness	colour thresholding, hue augmentation
YCbCr	brightness Y + two colour-difference channels	JPEG and video; enhance Y only (Day 3)
CIE L*a*b*	approximately *perceptually uniform*: equal distances look equally different	colour matching, quality control, clustering
Multispectral	many bands, including near-infrared	remote sensing, agriculture

Intuition

Choosing a colour space is choosing a *representation*, the central theme of this course. The right representation can turn a hard problem into a simple threshold.

Beyond RGB: seeing plant health from space

- Healthy leaves *absorb* red light (for photosynthesis) and strongly *reflect* near-infrared (NIR), which we cannot see.
- Stressed or sparse vegetation reflects less NIR.
- Satellites record both bands, so we can compute a vegetation index per pixel.

Normalised difference vegetation index

$$\text{NDVI} = \frac{\text{NIR} - \text{Red}}{\text{NIR} + \text{Red}} \in [-1, 1]$$

Two pixels

Dense forest: NIR $= 0.50$, Red $= 0.05$: NDVI $= 0.45/0.55 = 0.82$.

Bare soil: NIR $= 0.30$, Red $= 0.25$: NDVI $= 0.05/0.55 = 0.09$.

Open water: NIR $<$ Red, so NDVI < 0.

Intuition

Like HSV, NDVI is a *ratio*, so it is largely unaffected by overall brightness (sun angle, haze).

Part 5

Images in code

An image is a NumPy array

```
import numpy as np
from skimage import io
img = io.imread("yaounde_taxis.jpg")
img.shape      # (960, 1280, 3): rows, cols,
    channels
img.dtype      # uint8: integers 0..255
img[500, 300]           # one pixel: array([R, G,
     B])
img[500, 300, 0]        # its red value
img[400:600, 100:400]   # a crop (rows, then cols
    )
img[:, ::-1]            # mirror left-right
img[..., 1]             # the green channel
```

- **Rows first**: `img[y, x]`, not `img[x, y]`. The single most common indexing mistake.
- Slicing gives a *view*, not a copy: changing a crop changes the original image.

Intuition

Everything this week (filters, histograms, edges) is arithmetic on such arrays. If you are fluent with NumPy slicing, you are halfway there.

Bug 1: BGR vs. RGB

```
import cv2
img = cv2.imread("leaf.jpg")      # BGR order!
plt.imshow(img)                   # blue leaves
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(rgb)                   # correct
```

- OpenCV stores channels as **B, G, R** for historical reasons; almost everything else uses R, G, B.
- Silent and dangerous: a model trained on RGB and fed BGR still runs, just worse.

Displayed as RGB (correct)



Same array read as BGR



Bug 2: integer overflow

```
a = np.full((2, 2), 200, np.uint8)
b = np.full((2, 2), 100, np.uint8)
a + b                # 44   (wraps around mod
    256)
cv2.add(a, b)        # 255  (saturates)
a.astype(float) + b # 300.0

img = img.astype(np.float32) / 255.0  # safest
```

Why 44?

uint8 holds 0–255. $200 + 100 = 300$, and $300 - 256 = 44$: the counter rolls over like a car's odometer.

Practitioner's Note

Brightening an image by adding 100 in uint8 turns bright pixels *dark*. Convert to float, compute, then clip and convert back.

Bugs 3 and 4: layout and silent failure

3. HWC vs. CHW

```
img.shape                          # (H, W, 3)
     NumPy, OpenCV
x = img.transpose(2, 0, 1)         # (3, H, W)
     PyTorch
x = x[None]                        # (1, 3, H, W) a
      batch
```

NumPy puts channels *last*; PyTorch puts them *first* and adds a batch dimension in front.

4. Silent failure

```
img = cv2.imread("wrong/path.jpg")    #
     None!
assert img is not None, "file not found"
```

OpenCV returns `None` instead of raising an error. The crash comes later, with a confusing message.

A habit that prevents all four

After loading any image, print `shape`, `dtype`, `min()`, `max()` and *look at it*. Ten seconds that save hours.

Normalisation before learning

Two common normalisations

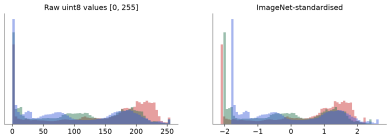
Min–max to $[0, 1]$:

$$\hat{I} = \frac{I - I_{\min}}{I_{\max} - I_{\min}}$$

Per-channel standardisation:

$$\hat{I}_c = \frac{I_c/255 - \mu_c}{\sigma_c}$$

ImageNet statistics: $\mu = (0.485, 0.456, 0.406)$, $\sigma = (0.229, 0.224, 0.225)$.



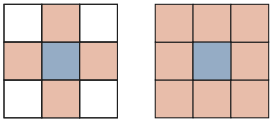
A red value of 180

$180/255 = 0.706$; $(0.706 - 0.485)/0.229 \approx 0.965$.

Practitioner's Note

Use the *same* statistics at training and test time: a pre-trained model expects the ones it was trained with.

Neighbours and connectivity



- **4-neighbours**: up, down, left, right.
- **8-neighbours**: also the four diagonals.
- Two foreground pixels are **connected** if a path of foreground neighbours joins them.
- A **connected component** is a maximal connected set: "one object".

Intuition

Connectivity is how a computer decides what counts as *one object*. Two pixels touching only at a corner are one object under 8-connectivity and two objects under 4-connectivity.

Distances between pixels

Three pixel distances

With $\Delta m = |m_1 - m_2|$ and $\Delta n = |n_1 - n_2|$:

$d_E = \sqrt{\Delta m^2 + \Delta n^2}$	Euclidean
$d_4 = \Delta m + \Delta n$	city block
$d_8 = \max(\Delta m, \Delta n)$	chessboard

d_4 counts steps of a rook-like walker that moves only along rows and columns; d_8 counts steps of a chess king.

Pixels $(2, 3)$ and $(5, 7)$

Differences: 3 rows, 4 columns.

$d_E = \sqrt{9 + 16} = 5$

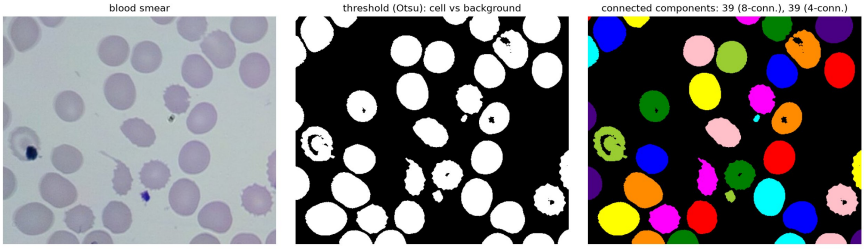
$d_4 = 3 + 4 = 7$

$d_8 = \max(3, 4) = 4$

Practitioner's Note

Distance transforms built on these are used to separate touching cells and to measure how far each pixel is from a boundary.

Connected components on a real blood smear



OpenFlexure Project, CC BY-SA 4.0

A global threshold separates cells from background, then connected-component labelling gives each object a number. In this crop, 4- and 8-connectivity agree: the cells are well separated.

Practitioner's Note

Count the coloured blobs: some contain two touching cells, and cut-off cells at the border count too. Real cell counters add splitting (watershed, Day 3 ideas) and border rules. Simple pipelines are easy to build and easy to fool.

Part 6

Wrapping up

Key takeaways

1. A pixel measures **light**, which depends on illumination, surface and geometry.
2. **Projection** $x = fX/Z$ discards depth; size falls as $1/Z$.
3. Each photosite sees one colour; **demosaicing** fills in the rest.
4. **Sampling** sets resolution; too coarse sampling **aliases**. Blur before shrinking.
5. **Quantisation** sets $L = 2^b$ levels; too few cause banding and lose detail.
6. Three colour channels because we have three cone types; **HSV** separates colour from brightness.
7. Know your `dtype`, channel order and layout; normalise consistently.
8. **Connectivity** defines what counts as one object.

Before the next lecture

Practise (tutorial notebook `W1D2_Representation_and_Colour`)

- Simulate sampling and quantisation; watch aliasing appear and disappear.
- Implement `rgb_to_gray` and `rgb_to_hsv` from the formulas and check them against scikit-image.
- Build a hue-based segmenter and test it on a photo you take yourself, in sun and in shade.

Read

- Course notes, Chapter 2
- Gonzalez & Woods, §2.4 and §6.1–6.2

Next lecture: processing images

- Where noise comes from, and how to measure it
- Convolution: the single most important operation in this course
- Blurring, denoising, sharpening
- The frequency view of images
- Histograms and contrast enhancement

Credits

Photographs from Wikimedia Commons under the licences shown. Other images: scikit-image sample data (`astronaut`, `camera`).

`yaounde_taxis`: Bdx, CC BY-SA 4.0
`douala_ndokoti`: Minette Lontsie, CC BY-SA 4.0
`douala_2014`: Guaka, CC BY-SA 4.0
`malaria_smear`: OpenFlexure Project, CC BY-SA 4.0
`cat_lace`: Famartin, CC BY-SA 4.0
`cat_persian`: Gourdo 1 at en.wikipedia, Public domain
`xray_tb`: B. Herreros et al., CC BY 4.0
`lake_chad_2024`: NASA MODIS Rapid Response, Public domain
`maize_blight`: Chroanch, CC BY-SA 4.0