Introduction to Image Analysis - Lecture 3

Sampling, Quantization & Interpolation

Dr. Hugo Armando GUILLEN RAMIREZ

Department of Visceral Surgery and Medicine
Department of Digital Medicine
University of Bern

Spring 2026

Outline

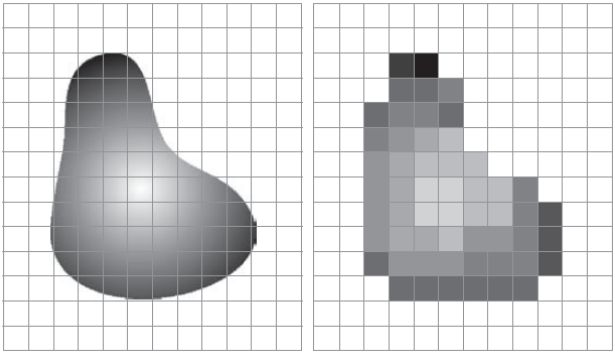
Learning Goals

After this lecture, you should be able to:

1. **Explain** how a continuous scene becomes a digital image (sampling → quantization pipeline).
2. **State** the Nyquist–Shannon sampling theorem, identify when aliasing occurs, and describe how anti-aliasing filters prevent it.
3. **Distinguish** uniform from optimal (Lloyd–Max) quantization and explain when each is appropriate.
4. **Read** an image histogram and diagnose common defects (low contrast, saturation, compression artefacts).
5. **Compare** interpolation methods (nearest-neighbour, bilinear, bicubic, spline) and select the right one for a given application.
6. **Describe** the Gaussian and Laplacian image pyramids and their role in multi-scale analysis.

Sampling

Intuition for sampling and quantization



- **Sampling** determines the shape and number of pixels
- **Quantization** determines the number of possible colors or shades each pixel can take
- We use the Fourier Transform to analyze if the sampling parameters (the pixel grid density) are sufficient to capture the image's details without causing artifacts like **aliasing**.

Formal Image Definition

Definition

A **digital image** is a function of two discrete coordinates x and y:

$$f : X \times Y \longrightarrow V$$

$X = \{0, 1, \ldots, M-1\}$	row indices
$Y = \{0, 1, \ldots, N-1\}$	column indices
$V = \{0, \ldots, 2^B - 1\}$	grey levels ($L = 2^B$)

Represented as an $M \times N$ **numerical array**: The value $f(x, y)$ is the intensity at **row** x and **column** y.

Matrix Representation:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Origin (0,0) is Top-Left

Common bit depths

Bits B	Levels L	Typical use
8	256	Standard photography
12	4 096	CT, Digital X-ray
16	65 536	MRI, Scientific Imaging

Cartesian vs. Row-Major Convention

Cartesian (Math/Geometry)

Used in geometry and signal processing.

- **Variables**: (x, y)
- **Axis**: x is horizontal, y is vertical.
- **Origin**: Bottom-left.
- **Intuition**: x is horizontal distance, y is vertical height.

Row-Major (Matrix/CV)

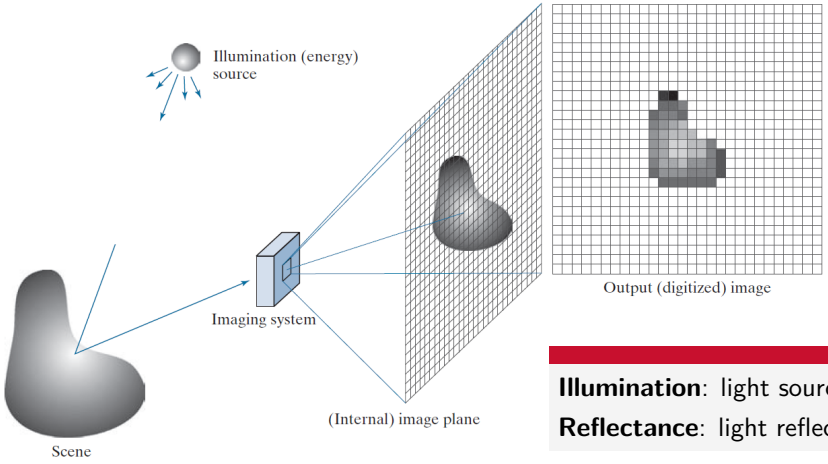
Standard for NumPy, OpenCV, and C++.

- **Variables**: (x, y) or (i, j)
- **Axis**: x denotes rows, y denotes columns.
- **Origin**: Top-left.
- **Intuition**: Row index i goes down, Column index j goes right.

Notational clarity

Gonzales' book uses the Row-Major notation.

Digital image acquisition



Illumination: light source incidence

Reflectance: light reflected

From World to Pixel

Two independent factors govern image intensity:

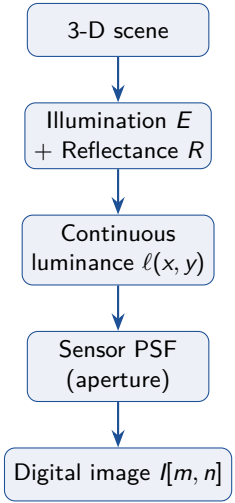
$$I(x, y) \;=\; E(x, y) \;\cdot\; R(x, y)$$

Symbol	Name	Range
E	**Illumination**	$[0, \infty)$
R	**Reflectance**	$[0, 1]$
I	**Recorded intensity**	$[0, \infty)$

Log-space **decouples** the two:

$$\log I \;=\; \log E \;+\; \log R$$

$\Rightarrow$ basis of *homomorphic filtering* (covered in later lectures).

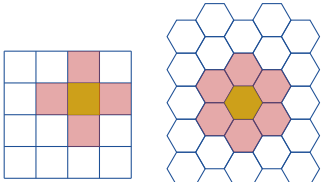


Sampling: Tessellation

- A pattern of regions that cover the image so that there are **no holes** and **no overlapping** regions.
- A cell is a picture element: pixel that should cover the entire image and accumulate brightness
- Only three regular polygons tile the plane gaplessly: squares, hexagons and triangles.

Standard cameras use **square grids** for hardware simplicity. Powers of 2 in size.

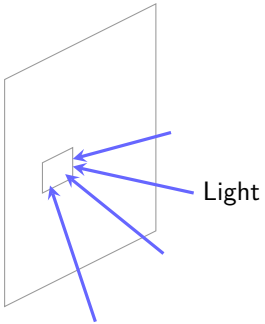
Hexagonal grids appear in some specialized sensors (e.g. certain retinal chips) because every neighbour is *equidistant*.



Square (4-nb) Hexagonal (6-nb)

Sampling: Model

- We need to:
  1. Integrate brightness over a region of a pixel
  2. Read out values at pixel locations



Define a function

$$p(x,y) = \begin{cases} 1 & \textit{if } (x,y) \textit{ is in pixel } (u,v) \\ 0 & \textit{otherwise} \end{cases}$$

Then a pixel value $I(u,v)$ can be defined as

$$I(u,v) = \iint I(x,y)p(u-x, v-y)dxdy$$

Sampling is best understood as a **frequency-dependent operation**. To model it accurately, we need a quick refresher on signal analysis.

Fourier Transform — Refresher

Core idea: a signal can be decomposed into its frequencies.

1-D Fourier Transform:

$$F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i \omega x}\, dx$$

$$f(x) = \int_{-\infty}^{\infty} F(\omega)\, e^{+2\pi i \omega x}\, d\omega$$

2-D extension:

$$F(u,v) = \iint f(x,y)\, e^{-2\pi i (ux+vy)}\, dx\, dy$$

Key properties:

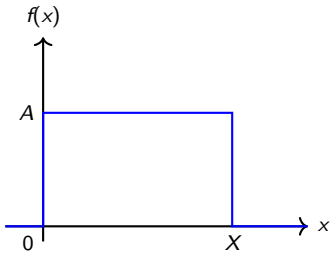
Property	Spatial	Frequency
Convolution	$f * g$	$F \cdot G$
Multiplication	$f \cdot g$	$F * G$
Shift	$f(x - x_0)$	$e^{-2\pi i \omega x_0} F$
Scaling	$f(ax)$	$\frac{1}{\lvert a \rvert} F(\omega/a)$

Rectangle → Sinc pair:

$$\mathrm{rect}_W(x) \xrightarrow{\mathcal{F}} W \mathrm{sinc}(W\omega)$$

Wider aperture ⇔ narrower spectrum.

Fourier Transform: Rectangular Pulse (box)



Rectangular pulse:

$f(x) = A$ for $0 \leq x \leq X$, 0 otherwise.

Its Fourier Transform magnitude:

$$|F(u)| = AX \left| \frac{\sin(\pi u X)}{\pi u X} \right| = AX |\mathrm{sinc}(uX)|$$

Key insight: a wider spatial pulse $\Leftrightarrow$ a narrower frequency spectrum (and vice versa).

Full derivation: see supplementary notes / Gonzalez Ch. 4.

The Impulse Train (Dirac Comb)

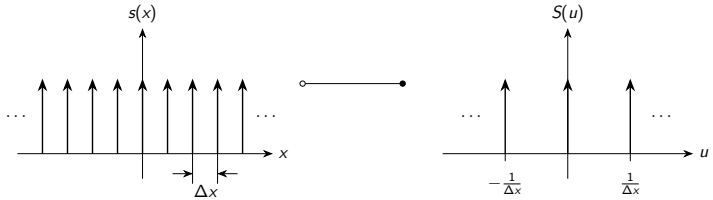
The Dirac Delta Function $\delta(x)$: An infinitely narrow, infinitely tall spike with an area of 1. It acts as our mathematical "snapshot" at a single point in space or time.

The Sampling Function $s(x)$: An infinite series of these spikes spaced by Δx is called an **Impulse Train** or **Dirac Comb**:

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - n\Delta x)$$

The Fourier Transform of an impulse train is **another impulse train** in the frequency domain!
If the spatial spikes are spaced by Δx, the frequency spikes $S(u)$ are scaled and spaced by $\frac{1}{\Delta x}$:

$$S(u) = \mathcal{F}\{s(x)\} = \frac{1}{\Delta x} \sum_{k=-\infty}^{\infty} \delta\left(u - \frac{k}{\Delta x}\right)$$

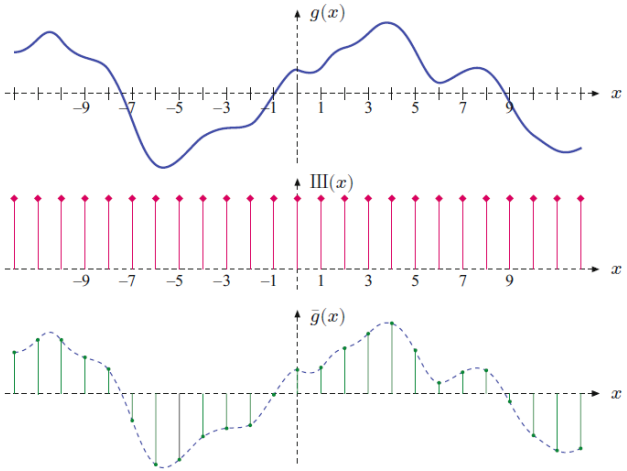


Sampling

Sampling

To sample a continuous signal $f(x)$, we multiply it by a sampling grid $s(x)$ (an impulse train spaced by Δx):

$$f_{\text{sampled}}(x) = f(x) \cdot \sum_{n=-\infty}^{\infty} \delta(x - n\Delta x)$$



Spectral replication and aliasing

The Frequency Domain

Recall that multiplication in the spatial domain becomes **convolution** in the frequency domain.

$$F_{\text{sampled}}(u) = F(u) * S(u)$$

Infinite Clones

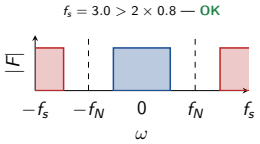
The Fourier Transform of an impulse train is another impulse train, spaced by the sampling frequency $f_s = \frac{1}{\Delta x}$.

Convolving our original spectrum $F(u)$ with this frequency grid creates **infinite, repeating copies** of $F(u)$ shifted by integer multiples of f_s:

$$F_{\text{sampled}}(u) = f_s \sum_{k=-\infty}^{\infty} F(u - k f_s)$$

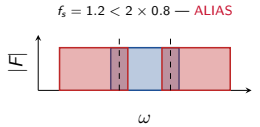
Copies do not overlap:

$$f_s \geq 2\,\omega_{\max} \quad \Leftrightarrow \quad \Delta x \leq \frac{1}{2\,\omega_{\max}}$$



Copies overlap: **aliasing**

$$f_s < 2\,\omega_{\max} \quad \Rightarrow \quad \text{ALIASING}$$



The red copies overlap the blue original $\Rightarrow$ false low-frequency content is injected.

Nyquist–Shannon Sampling Theorem

Theorem (Shannon, 1949 / Nyquist, 1928)

A band-limited signal with highest frequency $f_{\max}$ can be **perfectly reconstructed** from its samples *if and only if*

$$f_s \geq 2\, f_{\max} \iff \Delta x \leq \frac{1}{2\, f_{\max}} \iff \Delta x \leq \frac{1}{2\, \omega_{\max}}$$

- $f_N = f_{\max}$ is the **Nyquist frequency**.
- Sampling below Nyquist introduces **aliasing**.
- Aliased frequency of a component at ω:

$$\omega_{\text{alias}} = \omega - \left\lfloor \frac{\omega}{f_s} + 0.5 \right\rfloor f_s$$

1-D example (8 Hz signal)

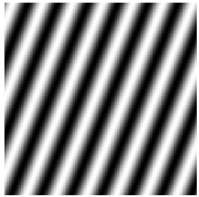
f_s	$\geq 2 \times 8$?	Alias
40 Hz	**Yes**	none
16 Hz	**Borderline**	none
9 Hz	No	$8 - 9 = -1$ Hz

Aliasing Example: Sampling Below Nyquist Rate

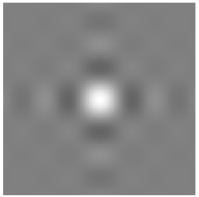
In this example the sampling rate ($\Delta x = 15$) is below the Nyquist rate.

- **(a)** the 128x128 sample rolling pattern
- **(b)** the sinc function for a sampling rate of $\Delta x = 15$. The grey background is zero, brighter is positive, and darker is negative
- **(c)** the original pattern is sampled with $\Delta x = 15$
- **(d)** the reconstructed rolling pattern is no longer valid. It is interesting that not only the frequency changed, but even the orientation of the pattern.

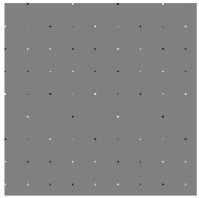
(a) Original pattern



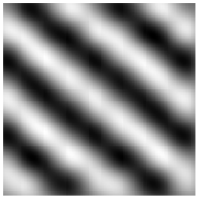
(b) Sinc size 15



(c) Sampled pattern



(d) Reconstruction

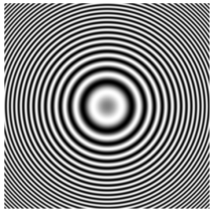


Aliasing: Variable Frequency Content

The input image contains regions with clearly different frequency content. Going from the centre to boundary, the frequency increases. It can be seen that once the Nyquist rate is higher than the actual sampling, aliasing occurs.

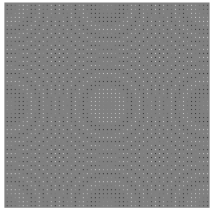
- **(a)** the 256x256 sample pattern
- **(b)** the sinc function for a sampling rate of $\Delta x = 5$ (grey is zero, brighter is positive, and darker is negative)
- **(c)** the original pattern is sampled with $\Delta x = 5$
- **(d)** the reconstructed pattern. In regions where the Nyquist rate is higher strong aliasing artefacts are present

(a) Original pattern

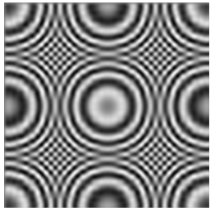


(b) Sinc size 5

(c) Sampled pattern



(d) Reconstruction



Moiré Patterns: Aliasing in the Real World

What is a Moiré Pattern?

A **Moiré pattern** is a visible artifact (an interference pattern) that appears when two fine, repetitive grids are overlaid at an angle or with slightly different spacing.

The Digital Imaging Cause

- **The Two Grids:** The camera's digital sensor (pixel grid) interacts with high-frequency details in the real world.
- **Under-sampling:** When the object's spatial frequency exceeds the sensor's Nyquist limit ($\Delta x \leq \frac{1}{2\omega}$ rule broken).
- **The Result:** High frequencies "fold back" into false low frequencies, creating wavy, artificial structures that do not actually exist in the scene.

Real-World Examples

- Photographing a TV or laptop screen (pixel grid on pixel grid).
- A rotating wheel filmed at too low a frame rate appears to spin backwards (wagon-wheel effect).
- Taking a picture of a brick wall or tiled roof from far away.

The Solution: Anti-Aliasing (Low-Pass) Filters

The Golden Rule of Sampling

If you cannot increase your sampling rate (f_s) to meet the Nyquist limit, you must **decrease the maximum frequency** (ω_{max}) of your signal before sampling.

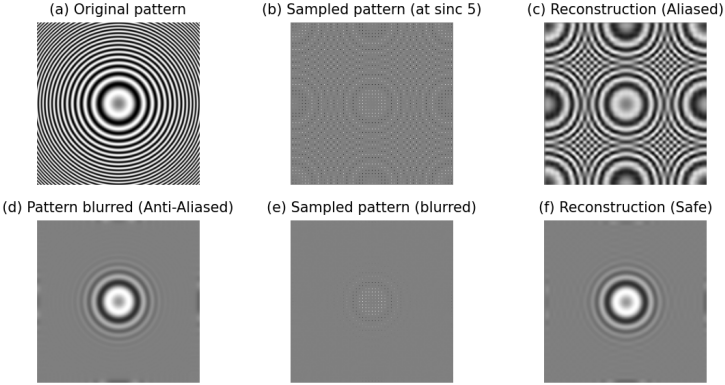
The Process (Pre-Filtering)

1. **Filter:** Apply a low-pass filter (like a Gaussian blur) to the continuous image to erase any details finer than $2\Delta x$.
2. **Sample:** Take your digital snapshots safely.
3. **Result:** The high frequencies are gone, so they cannot fold back and create Moiré patterns.

The Inevitable Trade-off

- By cutting off the high frequencies, we are permanently discarding the sharpest details of the image.
- **The Choice:** We trade crisp, corrupted details (aliasing/Moiré) for soft, mathematically accurate details (blur).
- *Note: Digital cameras achieve this physically using an optical low-pass filter (OLPF) placed directly over the sensor*

Anti-Aliasing



- Sampling the raw image captures frequencies higher than the Nyquist limit. The reconstruction **(c)** hallucinates false structures (Moiré patterns) where the rings get too tight.
- A strong Gaussian blur **(d)** destroys the high frequencies *before* sampling. The reconstruction **(f)** loses sharp detail, but gracefully fades to gray instead of corrupting the image.

Anti-Aliasing: Algorithm

Standard downsampling pipeline

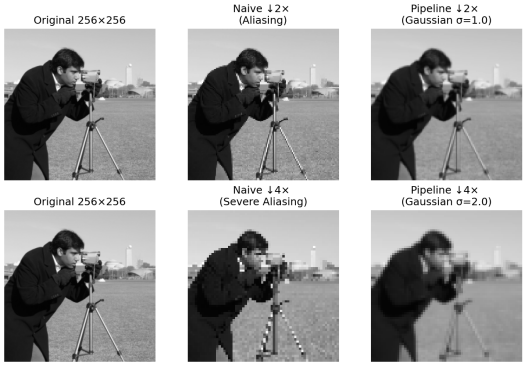
1. **Gaussian blur** with $\sigma \approx$ scale/2 pixels.
2. **Subsample** (take every n-th pixel).

Why Gaussian? It is the *only* separable, rotationally symmetric, non-negative filter. Scale-space theory guarantees it introduces no spurious features.

Rule of thumb

$\sigma \approx$ downsampling factor/2

For $\downarrow 2\times$: $\sigma = 1$. For $\downarrow 4\times$: $\sigma = 2$.



Exercise 1
Checkerboard Aliasing (~10 min)

Quantization

What is quantization?

After sampling, each pixel value must be stored as an integer.
This mapping from a continuous range to a finite set of levels is **quantization**.

$$f_{\text{continuous}} \longrightarrow Q(f) \in \{q_1, q_2, \ldots, q_L\}$$

Uniform quantization ($L = 2^B$ equal bins)

$$\Delta = \frac{f_{\max} - f_{\min}}{L}$$

$$Q(f) = \Delta \left\lfloor \frac{f - f_{\min}}{\Delta} + 0.5 \right\rfloor + f_{\min}$$

$$|e| \leq \tfrac{\Delta}{2}, \quad \sigma_q = \frac{\Delta}{\sqrt{12}}$$

Δ: Quantization step size (bin width)
e: Quantization error ($f - Q(f)$)
σ_q: Quantization noise (std. dev.)

Visual artefacts from low bit depths

Bits	Levels	Artefact
8	256	None visible
4	16	Slight banding
3	8	Visible banding
2	4	Severe false contours

False contours (banding) appear where the quantization step Δ is large relative to the local image gradient.

Uniform quantization on a smooth gradient

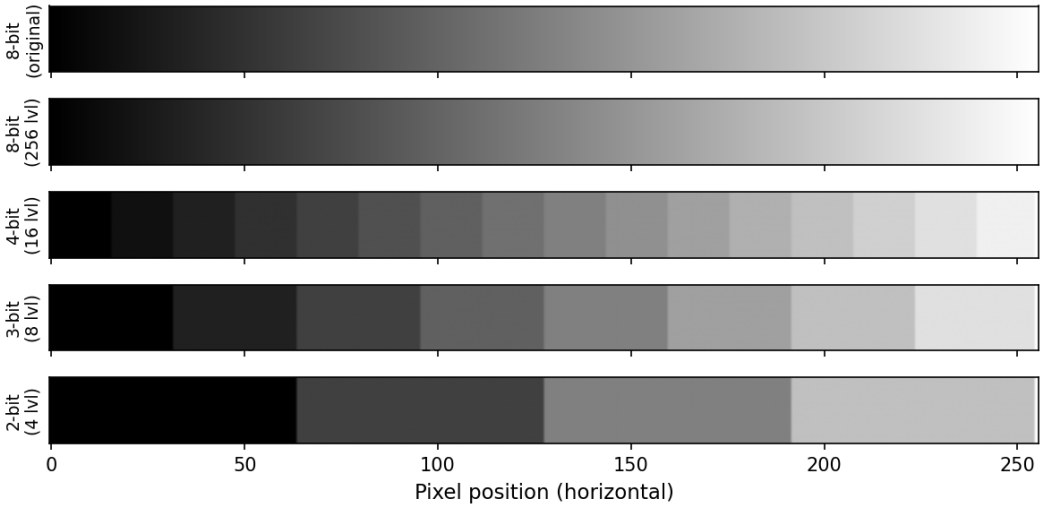
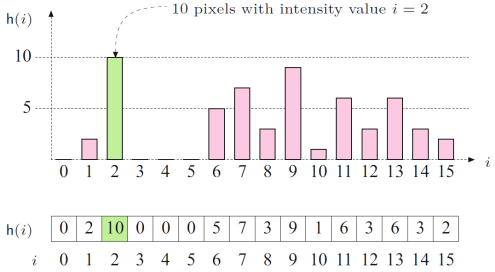


Image histogram

The histogram of an image describe the frequency of the intensity values that occur in it.
Note: the histogram does not encode the spatial arrangement of the pixels.



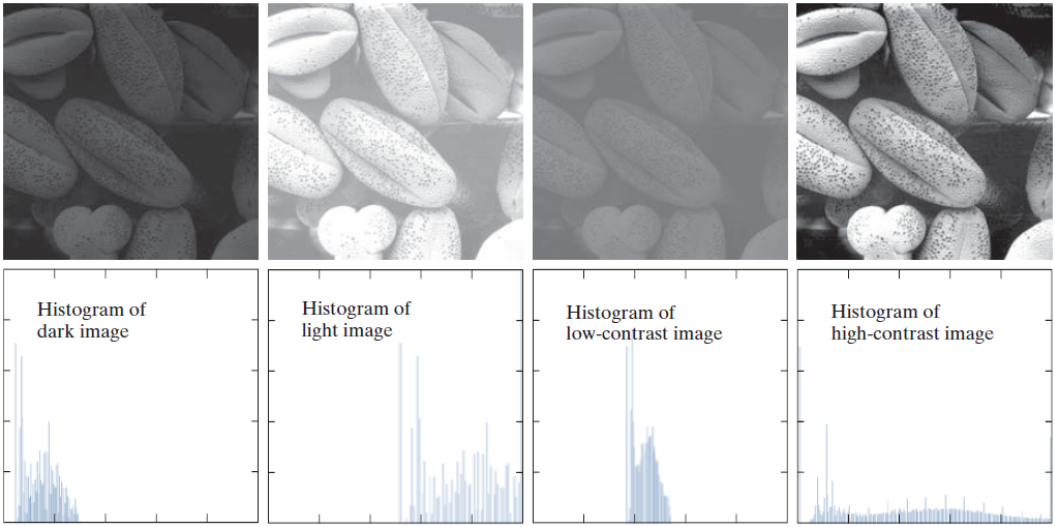
Let r_k, for $k = 0, 1, 2, \ldots, L-1$, denote the intensities of an L-level digital image, $f(x, y)$. The *unnormalized histogram* of f is defined as

$$h(r_k) = n_k \quad \text{for } k = 0, 1, 2, \ldots, L-1 \qquad (1)$$

where n_k is the number of pixels in f with intensity r_k, and the subdivisions of the intensity scale are called *histogram bins*.
The *normalized histogram* of f is defined as

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN} \qquad (3\text{-}7)$$

where M and N are the number of image rows and columns, respectively. The sum of $p(r_k)$ for all values of k is always 1.


Histogram of
dark image
Histogram of
light image
Histogram of
low-contrast image
Histogram of
high-contrast image

Interpreting Histograms: Diagnosing Image Problems

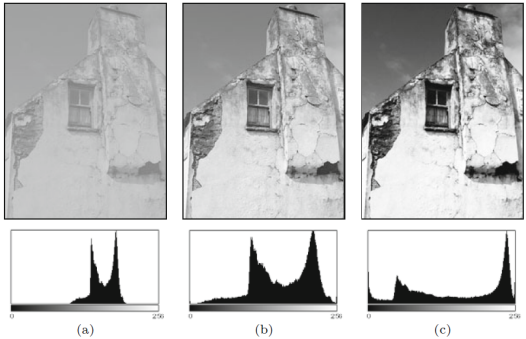
Key Properties:

- **Contrast:** The range of intensity values *effectively used*.
- **Dynamic Range:** The number of *distinct* pixel values present.

Identifying Defects:

- **Saturation:** Spikes at histogram tails (under/overexposure).
- **Spikes and Gaps:** Contrast changes create gaps/merges in the histogram.
- **Compression:** GIF reduces dynamic range; JPEG on simple graphics introduces fuzzy values.

Check Burger's book Section 3.2 for examples



a) Low contrast, b) normal contrast, c) high contrast.

Motivation for Lloyd–Max Quantization

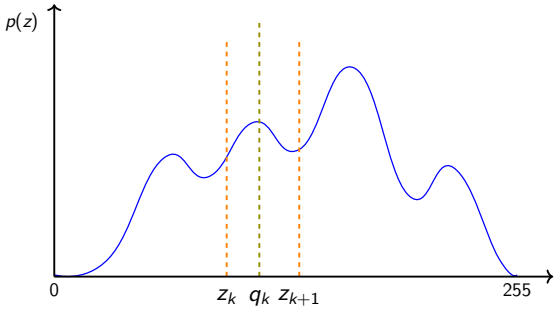
Uniform bins are optimal **only for uniform distributions**.

Natural images have a roughly **Gaussian/Laplacian** histogram (most pixels cluster near mid-grey.)

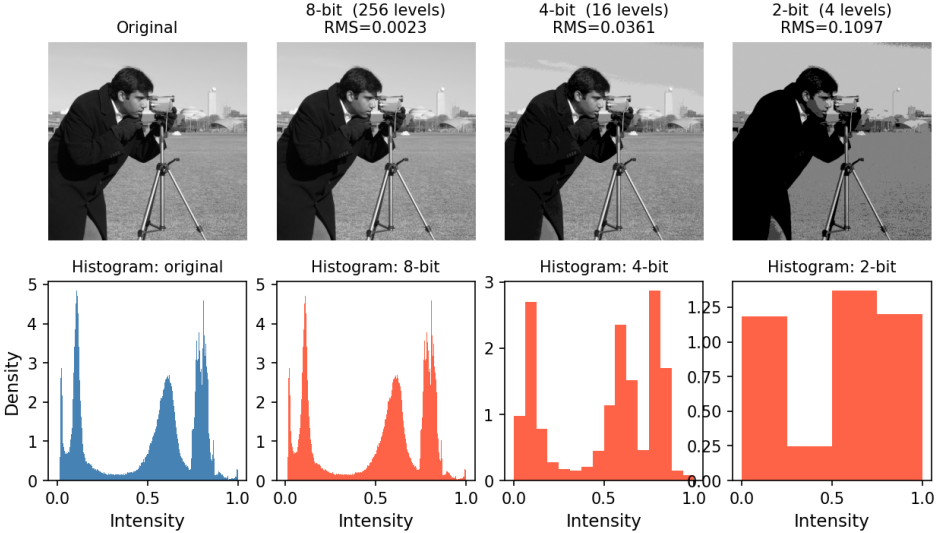
Goal

Adaptively design the size of the bins so the reconstruction error is minimum.

- Let $[z_k, z_{k+1}]$ define the boundaries of the k-th bin.
- Let q_k be the *representative* pixel value used for all pixels falling in that bin.
- $p(z)$ is the probability density function (histogram) of the image.



Example with Uniform Quantization



Minimizing Quantization Error

We want to minimize the **average quantization error** (MSE weighted by histogram $p(z)$):

$$\varepsilon = \sum_{k=1}^{K} \int_{z_k}^{z_{k+1}} (z - q_k)^2 p(z) dz$$

Setting $\partial\varepsilon/\partial z_k = 0$ and $\partial\varepsilon/\partial q_k = 0$ yields two intuitive rules:

Rule 1: Optimal Boundaries (z_k)
Place boundaries midway between representatives:

$$z_k = \frac{1}{2}(q_{k-1} + q_k)$$

Rule 2: Optimal Representatives (q_k)
Use the centroid of probability mass in each bin:

$$q_k = \frac{\int_{z_k}^{z_{k+1}} z\, p(z)\, dz}{\int_{z_k}^{z_{k+1}} p(z)\, dz}$$

Because z_k depends on q_k and vice versa, we solve **iteratively**: guess initial bins $\rightarrow$ compute q_k $\rightarrow$ update z_k $\rightarrow$ repeat until convergence.

Lloyd–Max: Iterative Algorithm

Algorithm

1. **Initialise** boundaries $\{z_k\}$ uniformly over $[0, 1]$.
2. **Centroid step:** compute q_k as the centroid of each bin using the image histogram as $p(z)$.

$$q_k = \frac{\sum_{z \in \text{bin}_k} z\, p(z)}{\sum_{z \in \text{bin}_k} p(z)}$$

3. **Midpoint step:** update boundaries:

$$z_k = \tfrac{1}{2}(q_{k-1} + q_k)$$

4. **Repeat** until $\max_k |z_k^{\text{new}} - z_k^{\text{old}}| < \epsilon$.

Uniform vs Lloyd–Max

Original image

Uniform 3-bit (8 lvl)
RMS=0.07311

Lloyd–Max (8 lvl)
RMS=0.04258

Exercise 2
Lloyd–Max on a Bimodal Image (~15 min)

Interpolation

Overview: Why Interpolation?

Interpolation estimates pixel values at **non-integer coordinates**:

- **Resizing** (zoom in / zoom out)
  - Given a scale S, an image of dimensions $M \times N$ results in another one of size $\lfloor SM \rfloor \times \lfloor SN \rfloor$.
  - $S > 1$: upsampling (enlarge); $0 < S < 1$: downsampling (reduce).
- **Rotation** and affine warping
- **Image registration** (sub-pixel alignment)
- **Pyramid upsampling** (Laplacian pyramid reconstruction)

a) Original, b) traslation, c) scaling in x and y directions, d) rotation about the center, e) projective transformation, and f) nonlinear distortion.

Interpolations

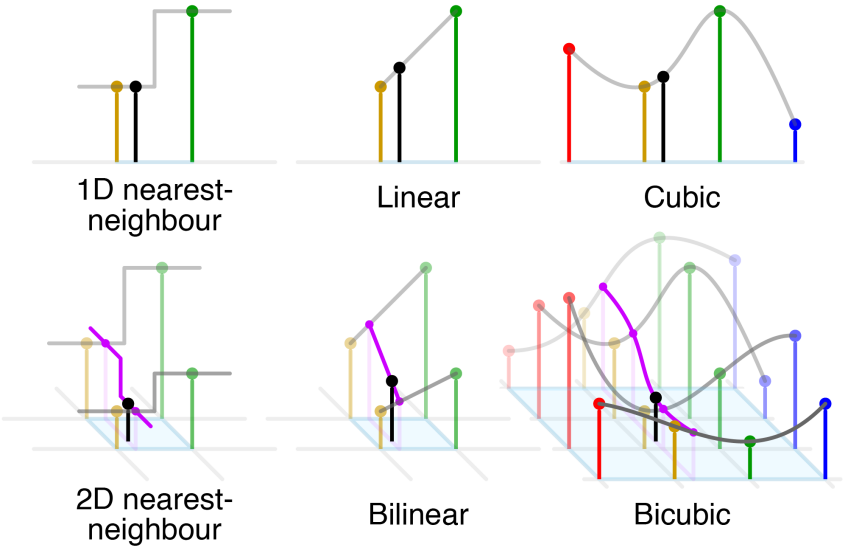


Image source: wikipedia

Nearest-Neighbor Interpolation

The simplest scheme: map each query point to the **closest grid point**.

$$I'(x, y) = I(\lfloor x + 0.5 \rfloor,\ \lfloor y + 0.5 \rfloor)$$

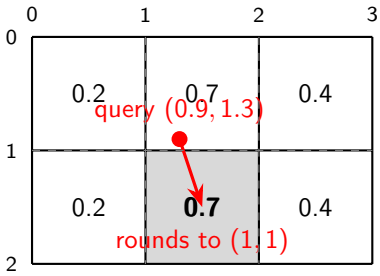
NN with scaling

To generate an output image I' scaled by a factor S from an input image $I(x, y)$, we map each output pixel coordinate (u, v) back to the source:

$$I'(u, v) = I\left(\left\lfloor \frac{u}{S} + 0.5 \right\rfloor, \left\lfloor \frac{v}{S} + 0.5 \right\rfloor\right)$$

(the dimension of I' is $\lfloor SM \rfloor \times \lfloor SN \rfloor$).

Example: locate NN of pixel at (0.9,1.3)

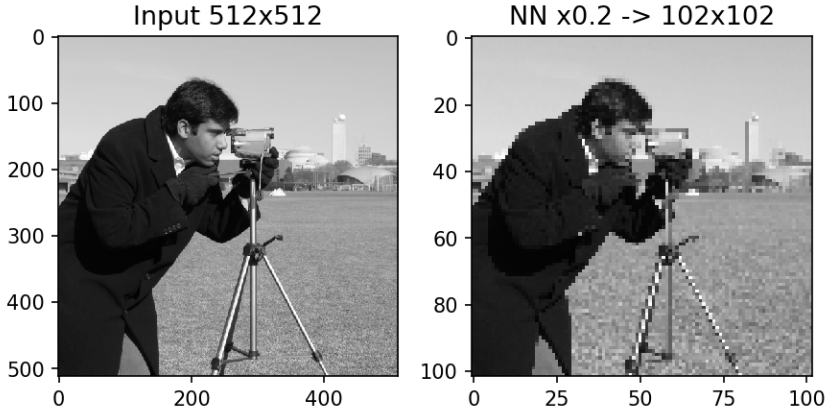


Advantages:

- Time complexity: $O(1)$ per pixel.
- Preserves *exact* original values.
- Correct for **label maps** and segmentation masks (no blending between class indices).

Disadvantages:

- Blocky / staircase artefacts (step discontinuities).
- Not appropriate when the output must be smooth.



Linear Interpolation

Given: two samples $Q_0 = f(x_0)$, $Q_1 = f(x_1)$. **Estimate:** $f(x)$ for $x_0 \leq x \leq x_1$.

Step 1: assume a *straight line* through the two knots. The line through (x_0, Q_0) and (x_1, Q_1) is:

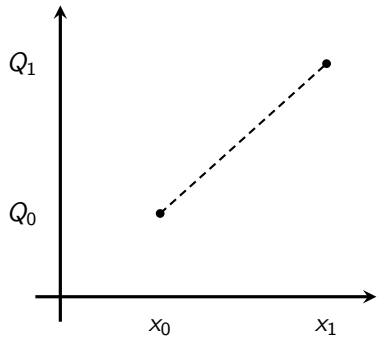
$$f(x) = Q_0 + \frac{Q_1 - Q_0}{x_1 - x_0}(x - x_0)$$

Step 2: let $t = \frac{x - x_0}{x_1 - x_0} \in [0, 1]$ (normalised position):

$$\boxed{f(x) = (1 - t)\, Q_0 + t\, Q_1}$$

or

$$\boxed{f(x) = \frac{x - x_0}{x_1 - x_0}\, Q_1 + (1 - \frac{x - x_0}{x_1 - x_0})\, Q_0}$$

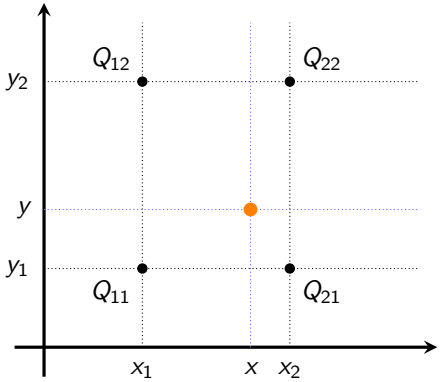


2D Case: Bilinear Interpolation

What if we want if want to interpolate an image? What value should be computed for a given 2D coordinate (x, y)?

Idea

Do 1D linear interpolation twice. Once for the x coordinate, and then for the y coordinate, using the outcome of the former.



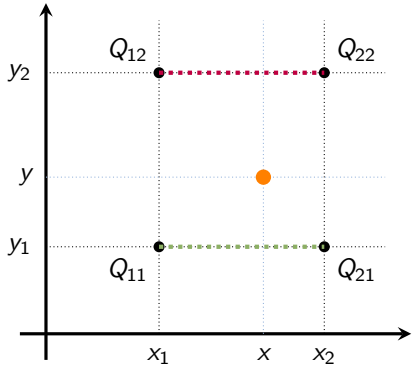
Bilinear Interpolation: Step by Step

Assume: (x, y) is within the bounds of $Q_{11}, Q_{21}, Q_{12}, Q_{22}$.

Step 1: Interpolate linearly along x on each row:

$$f(x, y_1) = \left(\frac{x - x_1}{x_2 - x_1}\right) Q_{21} + \left(1 - \left(\frac{x - x_1}{x_2 - x_1}\right)\right) Q_{11}$$

$$f(x, y_2) = \left(\frac{x - x_1}{x_2 - x_1}\right) Q_{22} + \left(1 - \left(\frac{x - x_1}{x_2 - x_1}\right)\right) Q_{12}$$



Assume: (x, y) are given and is within the bounds of $Q_{11}, Q_{21}, Q_{12}, Q_{22}$. Build

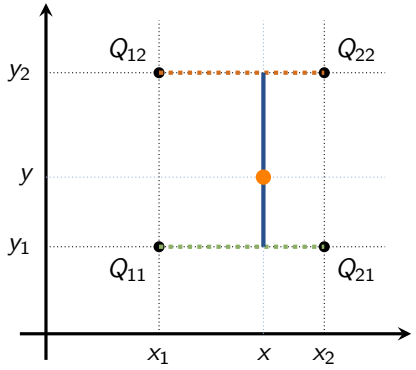
interpolation model on y:

$$f(x, y_1) = \left(\frac{x - x_1}{x_2 - x_1}\right) Q_{21} + \left(1 - \left(\frac{x - x_1}{x_2 - x_1}\right)\right) Q_{11}$$

$$f(x, y_2) = \left(\frac{x - x_1}{x_2 - x_1}\right) Q_{22} + \left(1 - \left(\frac{x - x_1}{x_2 - x_1}\right)\right) Q_{12}$$

Use to interpolate on x too:

$$f(x, y) = \left(\frac{y - y_1}{y_2 - y_1}\right) f(x, y_1) + \left(1 - \left(\frac{y - y_1}{y_2 - y_1}\right)\right) f(x, y_2)$$



Expanding the Expression

Let

$$\alpha = \frac{x - x_1}{x_2 - x_1}, \qquad \beta = \frac{y - y_1}{y_2 - y_1}$$

then,

$$f(x, y_1) = (1 - \alpha)\, Q_{11} + \alpha\, Q_{21} \qquad f(x, y_2) = (1 - \alpha)\, Q_{12} + \alpha\, Q_{22}$$

and

$$f(x, y) = (1 - \beta)\, f(x, y_1) + \beta\, f(x, y_2).$$

Expanding fully:

$$f(x, y) = (1 - \alpha)(1 - \beta)\, Q_{11} + \alpha(1 - \beta)\, Q_{21} + (1 - \alpha)\beta\, Q_{12} + \alpha\beta\, Q_{22}$$

Which is the compact **matrix formulation**:

$$f(x, y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x, x - x_1 \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}$$

Bilinear Interpolation Properties

Important: bilinear $\neq$ linear!

The cross-term $\alpha\beta$ makes bilinear interpolation a *degree-2 polynomial* in (x, y), **not** a linear function.
Linear in each variable separately (hence "bi-linear").

Consequences:

- Does not work great for extrapolation (query values beyond the values).
- Sufficient for most image resampling tasks.

Computation cost

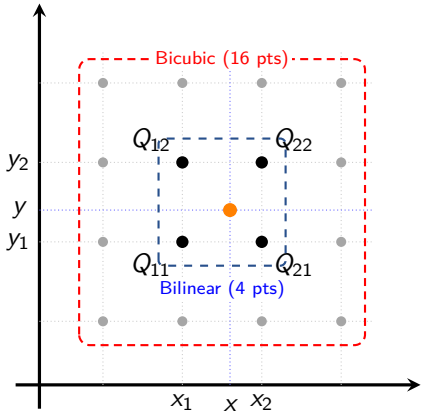
3 linear interpolations: 2 horizontal + 1 vertical
$\Rightarrow$ 4 multiplies, 4 adds per output pixel.

From Bilinear to Bicubic

Bilinear is fast and linear in each axis, but it often smooths fine detail and can look blurry when upscaling.

Bicubic applies cubic interpolation in each axis, which usually preserves detail better than bilinear.

Because a cubic model needs more neighboring samples, we expand the neighborhood from a 2×2 **patch** to a 4×4 **patch**.



Advantages over bilinear:

- Sharper detail in many images.
- Commonly used in Photoshop, MATLAB `imresize`, and OpenCV.

Disadvantages:

- Ringing / overshoot near sharp edges.
- Uses 16 neighboring samples (vs. 4 for bilinear).
- Can produce values *outside* $[0, 1]$.

Bicubic Interpolation

$$f(x,y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} Q_{i,j} \cdot W(u-i) \cdot W(v-j)$$

where $u = x - \lfloor x \rfloor$, $v = y - \lfloor y \rfloor$ are fractional distances, and $W(t)$ is the cubic weight function ($a = -0.5$):

$$W(t) = \begin{cases} (a+2)|t|^3 - (a+3)|t|^2 + 1 & |t| \leq 1 \\ a|t|^3 - 5a|t|^2 + 8a|t| - 4a & 1 < |t| < 2 \\ 0 & |t| \geq 2 \end{cases}$$

Distances $|t|$ for 4 neighbours:

- $i = -1$: $|t| = u + 1$
- $i = 0$: $|t| = u$
- $i = 1$: $|t| = 1 - u$
- $i = 2$: $|t| = 2 - u$

Introduction to Spline Interpolation

Fitting a single high-degree polynomial through many points causes wild oscillations (Runge's phenomenon). A **spline** uses **piecewise** low-degree polynomials linked smoothly at *knots*.

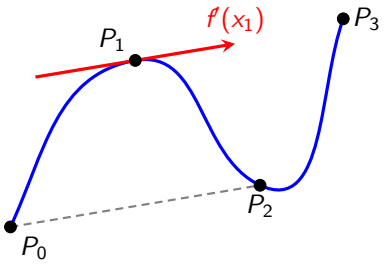
Key Properties (Cubic Splines):

- Passes exactly through each data point.
- Different cubic per interval.
- Continuous derivatives at knots.

Catmull-Rom Spline (used in image processing):
Tangent at P_i is parallel to the line $P_{i-1} \to P_{i+1}$:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}}$$

Local control, no global system to solve.

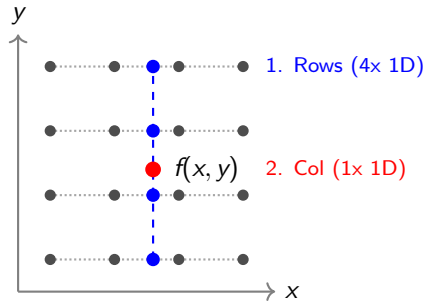


Spline Interpolation in 2D

The cubic filter is **separable**: compute via five 1D interpolations.

Algorithm:

1. **Fetch:** Identify the 4×4 grid surrounding the target.
2. **Horizontal pass:** 1D cubic spline on each of the 4 rows $\rightarrow$ **4 intermediate points**.
3. **Vertical pass:** 1D cubic spline on those 4 points $\rightarrow$ **final value**.



Practical choice

Bilinear: real-time rendering, thumbnailing.
Bicubic: high-quality resizing, medical display.
Spline: registration, when C^2 smoothness is required.

Image Pyramids & Multi-scale Analysis

Motivation: Why Multi-Scale?

The Problem with Fixed Scales:

- Objects in the real world appear at different sizes depending on distance.
- A fixed-size filter (e.g., a 3×3 kernel) only captures structures of that specific size.
- It might find the edges of a leaf, but completely miss the shape of the whole tree.

The Solution:

- Analyze the image at multiple resolutions simultaneously.
- **Coarse scales** capture global context and large structures.
- **Fine scales** capture local texture and sharp details.

The Core Idea

Instead of scaling our filters up (which is computationally expensive), we scale the image down.

Gaussian Image Pyramid

Definition

$$G_0 = \text{original}, \qquad G_{k+1} = \mathrm{DOWN}(G_k * h_\sigma)$$

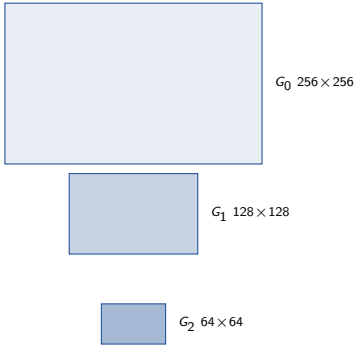
Each level: blur with Gaussian h_σ, then subsample $\downarrow 2$.

Why blur before subsampling?

- To prevent **aliasing**. High frequencies must be removed before reducing the sampling rate.

Properties:

- Level k has $1/4$ the pixels of level $k-1$.
- Total extra storage is cheap: $\sum_{k=1}^{\infty} 4^{-k} = 1/3$ of original.
- Foundation for SIFT, ORB, and optical flow.



Visualizing the Gaussian Pyramid

Progressive blurring and downsampling

G_0 (Original)	G_1	G_2	G_3	G_4 (Coarsest)
256×256	128×128	64×64	32×32	16×16

- **Loss of Detail:** As k increases, high-frequency details (sharp edges, fine textures) are permanently removed by the Gaussian filter h_σ.
- **Scale Reduction:** Each successive image has exactly one-quarter the number of pixels of the previous level (width and height are halved).
- This sequence of images serves as the base structure from which we compute the Laplacian pyramid and scale-space features.

Laplacian Pyramid

Definition (Burt & Adelson, 1983)

$$L_k = G_k - \mathrm{UP}(G_{k+1}), \qquad L_N = G_N \text{ (coarsest level)}$$

$\mathrm{UP}(G_{k+1})$ bilinearly upsamples G_{k+1} to the size of G_k.

Intuition: Storing only the differences

- L_k captures the band-pass residual (the details lost during blurring).
- L_0 = fine details (sharp edges).
- L_{N-1} = coarse structure.

Perfect reconstruction:

$$G_0 = \sum_{k=0}^{N} \mathrm{UP}^{(k)}(L_k)$$

RMSE $\approx 10^{-6}$ (machine precision)

Applications

- **Image blending** (seamless compositing without ghosting).
- **Compression** (mostly zeros, highly compressible).
- **Feature detection** (bands respond to characteristic scales).

Visualizing the Laplacian Pyramid

Band-pass details across scales

L_0 (Fine edges)

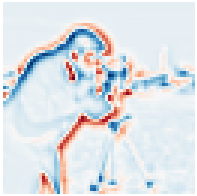
256×256

L_1

128×128

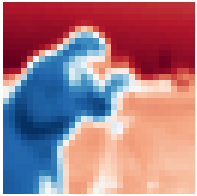
L_2

64×64



L_3

32×32



L_4 (Low-pass base)

16×16

- Red indicates positive differences, blue indicates negative.
- **Sparsity:** Notice how much of the L_0 through L_3 images are grey (zero). This visualizes exactly why the Laplacian pyramid is so highly compressible.
- **The Base (L_4):** The coarsest level isn't a difference image; it's the remaining low-frequency information needed to anchor the reconstruction.

Summary

Sampling

- A digital image is obtained by sampling a continuous signal onto a discrete grid and quantizing its values.
- The **Nyquist–Shannon theorem** dictates that $f_s \geq 2f_{\max}$; violating it causes **aliasing**.
- **Anti-aliasing**: low-pass filter (Gaussian blur) *before* subsampling.

Quantization

- Maps continuous intensities to discrete levels ($L = 2^B$).
- **Uniform** quantization is simple; **Lloyd–Max** minimizes MSE by adapting bins to the histogram.
- Histograms reveal contrast, dynamic range, and artefacts.

Interpolation

- Estimates pixel values at non-integer coordinates (resizing, rotation, registration).
- **Nearest-neighbour**: fast, exact values, blocky.
- **Bilinear**: smooth, 2×2 neighbourhood.
- **Bicubic**: sharper, 4×4 neighbourhood, possible ringing.
- **Splines**: C^2 smooth, best for registration.

Image Pyramids

- **Gaussian**: progressive blur + downsample.
- **Laplacian**: band-pass residuals, perfect reconstruction. Foundation for blending, compression, and feature detection.

Learning Goals — Self-Check

Can you now...

1. ✓ **Explain** the sampling → quantization pipeline that converts a continuous scene into a digital image?
2. ✓ **State** the Nyquist–Shannon theorem, identify aliasing conditions, and describe anti-aliasing filters?
3. ✓ **Distinguish** uniform from Lloyd–Max quantization and explain when each is appropriate?
4. ✓ **Read** an image histogram and diagnose common defects?
5. ✓ **Compare** interpolation methods (NN, bilinear, bicubic, spline) and select the right one for a given task?
6. ✓ **Describe** the Gaussian and Laplacian pyramids and their role in multi-scale analysis?

If not...

Review the corresponding slides, work through the examples, and check Gonzalez Ch.~2.3–2.4, 4.3 and Burger Ch.~1.4, 3.

Thank you for attending!

Questions?

hugo.guillenramirez@unibe.ch

Next week: Lecture 4: Point Operations & Linear Filtering

See you next week!

References

1. Gonzalez, R.C. & Woods, R.E. (2018). Digital Image Processing (4th ed.). Pearson.
2. Burger, W. & Burge, M. (2016). Digital Image Processing: An Algorithmic Introduction Using Java (2nd ed.). Springer.
3. Burt, P.J. & Adelson, E.H. (1983). The Laplacian pyramid as a compact image code. IEEE Trans. Commun., 3(4).
4. Lloyd, S.P. (1982). Least squares quantization in PCM. IEEE Trans. Inf. Theory, 28(2), 129–137.