Intensity and Contrast Fundamentals

Machine Vision

Dr. Mostapha Kalami Heris

m.k.heris@shu.ac.uk

School of Engineering and Built Environment

Sheffield Hallam University

Learning outcomes

By the end of this session, you will be able to:

- Define and interpret grayscale and color histograms, including common shapes and what they imply for contrast.
- Apply and compare global intensity transforms: linear stretch, gamma correction, and histogram equalization.
- Use CLAHE safely to address non-uniform illumination, and set NumTiles and ClipLimit with purpose.
- Diagnose artifacts introduced by enhancement and explain their impact on downstream tasks such as edge detection.
- Record parameters, ranges, and outcomes to support reproducibility in lab reports.

From Week 1 to Week 2

- Images are **data**: matrices, types, ranges
- RGB as **stacked channels** (M×N×3)
- Correct interpretation precedes any operation on pixels
- **Why contrast now:** visibility of structures drives edges, features, and later decisions

Guiding Question: **What needs to become more visible for the downstream task to work better?**

Today’s Roadmap

- **Histogram literacy**
- **Global intensity transforms**
  - Linear remapping: `imadjust`, `stretchlim`
  - Nonlinear remapping: **gamma**
  - Equalization: `histeq`
  - Distribution matching: `imhistmatch`
- **Local contrast enhancement**
  - CLAHE: `adapthisteq` with sensible defaults and safe tuning

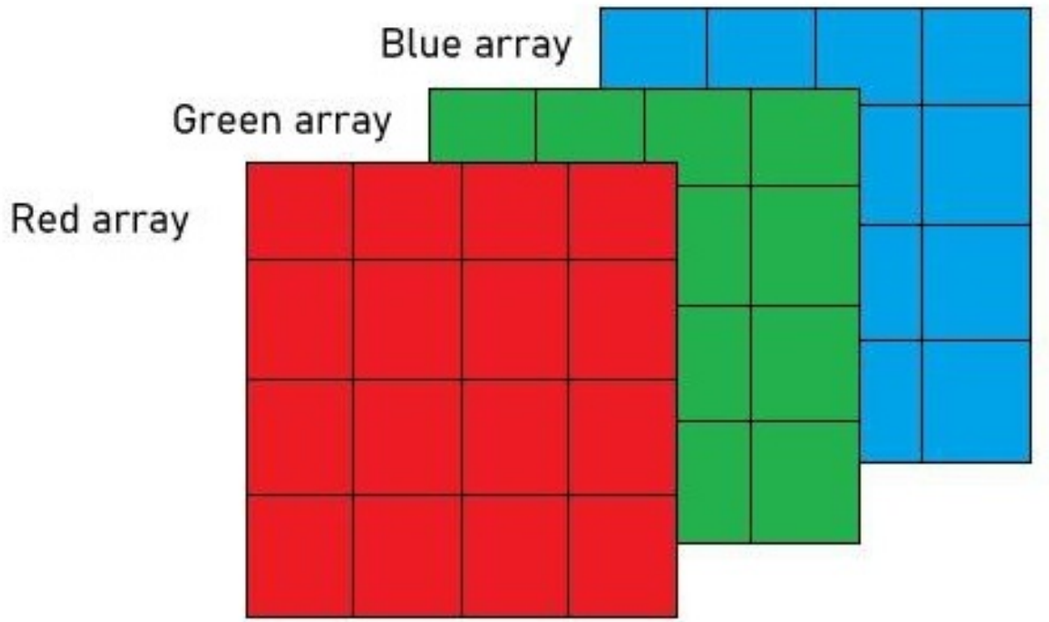
Image Representation Refresher

Data Types and Ranges

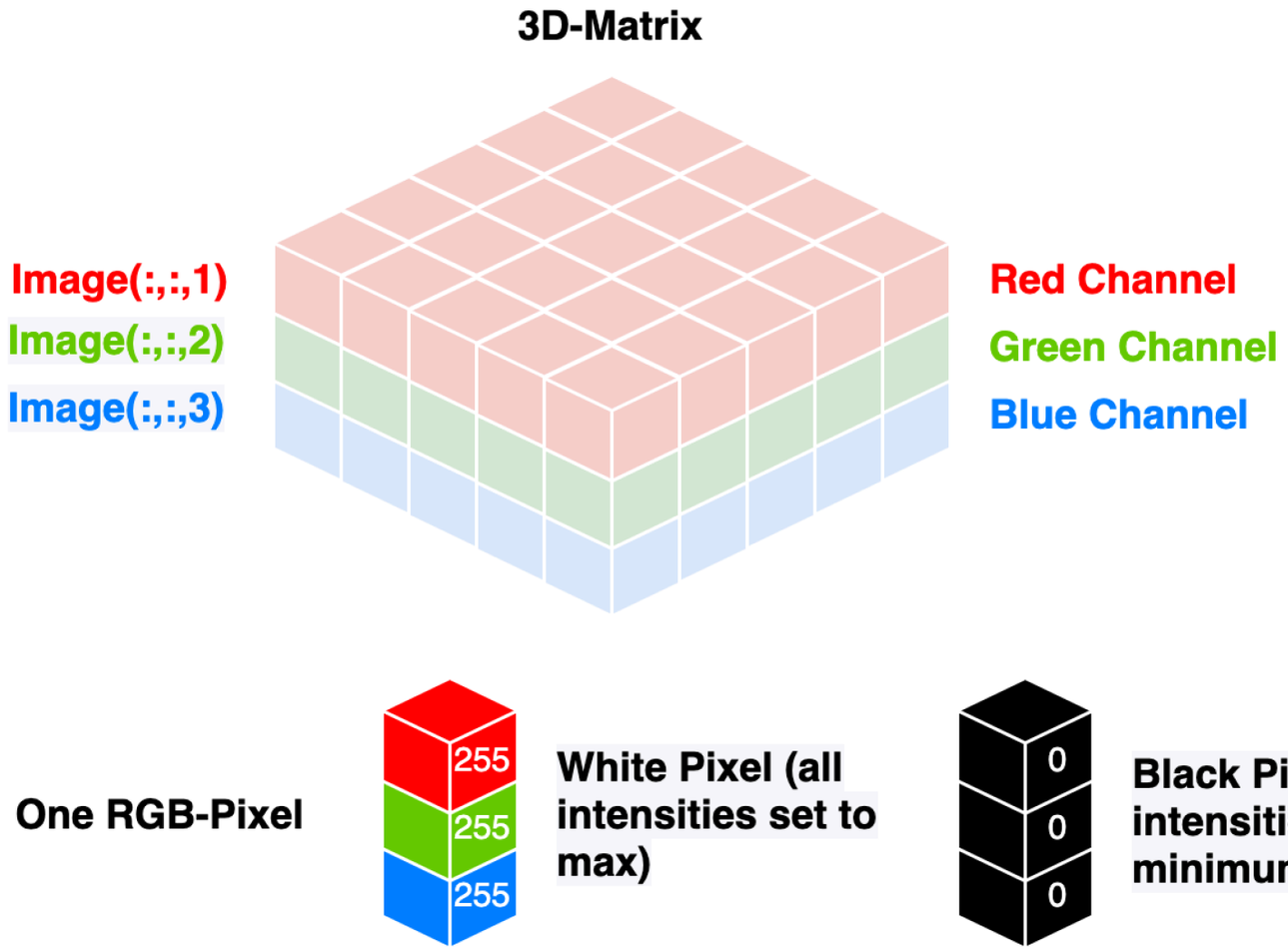
- Common classes in MATLAB
  - `uint8` in range 0 to 255
  - `uint16` in range 0 to 65535
  - `double` in range 0 to 1 for image functions
- Pitfall
  - Arithmetic on `uint8` can saturate
  - Accidental scaling mistakes change results
- Rule of thumb
  - Be explicit about type and expected range before applying transforms

Coordinate and Indexing Conventions

- Image as matrix
  - Rows → y, Columns → x
- Grayscale: `M × N`
- RGB color: `M × N × 3`
- Channel indexing
  - Red: `I(:,:,1)`
  - Green: `I(:,:,2)`
  - Blue: `I(:,:,3)`
- Consistent notation for the module



RGB Channels as 3D Model



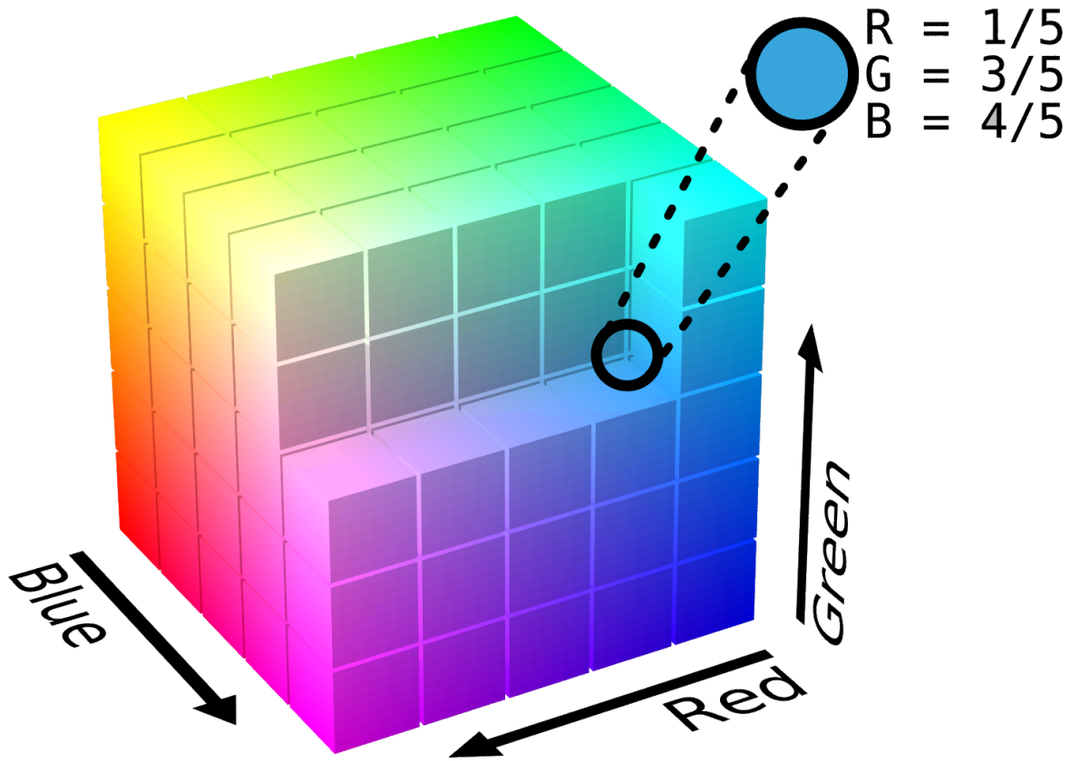
Sheffield Hallam University

Channel Literacy in MATLAB

```MATLAB
R = I(:,:,1);    % Red channel
G = I(:,:,2);    % Green channel
B = I(:,:,3);    % Blue channel

I_redonly = cat(3, R, 0*G, 0*B); % Create a red-only image

I_swapRG  = cat(3, G, R, B); % Swap red and green channels

% Zeroing a channel to study contributions
I_noBlue = I;
I_noBlue(:,:,3) = 0;
```

Color Spaces in Practice: RGB

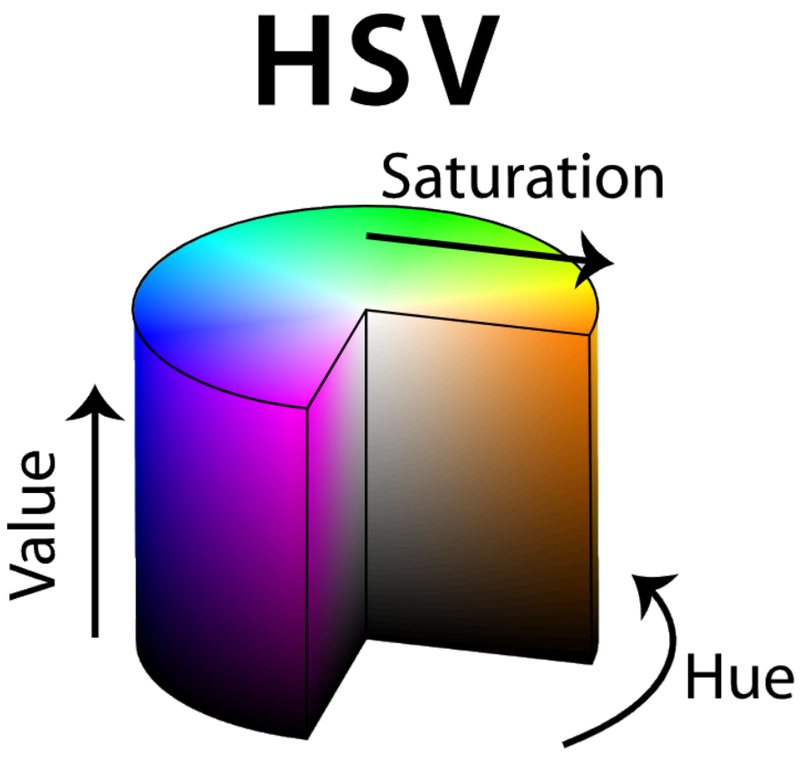
- **RGB** stands for Red, Green, Blue
- Device oriented and native to displays
- Simple indexing
- Per channel edits affect hue and saturation



Sheffield Hallam University

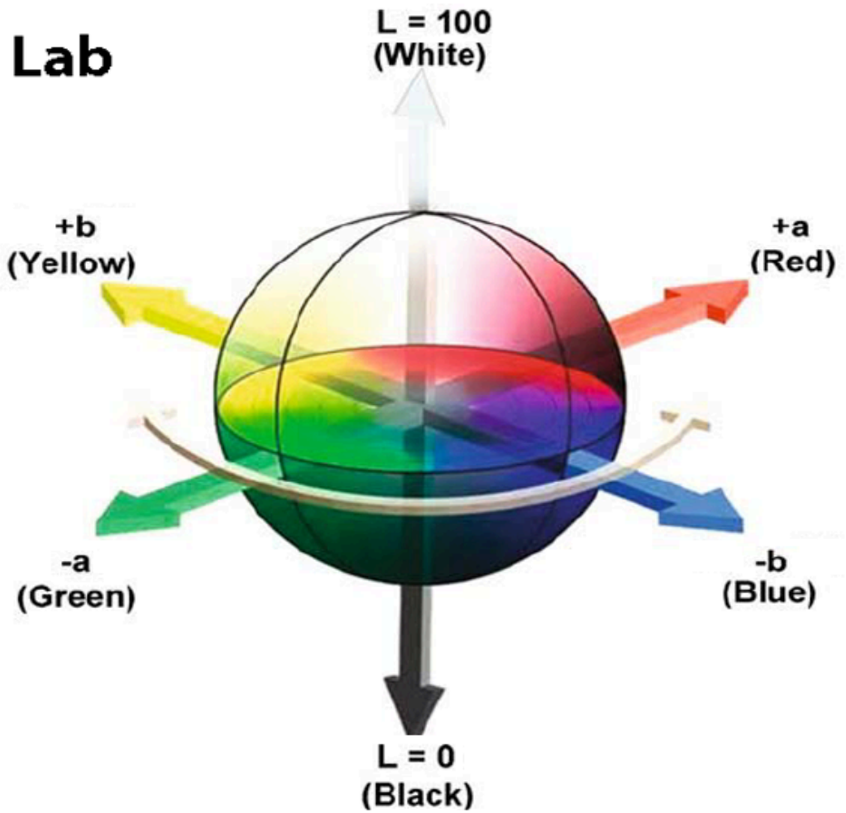
Color Spaces in Practice: HSV

- **HSV** stands for Hue, Saturation, Value
- Separates Hue, Saturation, Value
- Useful for color selection and thresholding
- Value is the maximum of RGB, not perceptual lightness



Color Spaces in Practice: L*a*b*

- **L*** is lightness, **a*** is green to red, **b*** is blue to yellow
- Approximately perceptually uniform
- Preferred for contrast work that should not change hue
- More complex conversion, but worth it for quality results

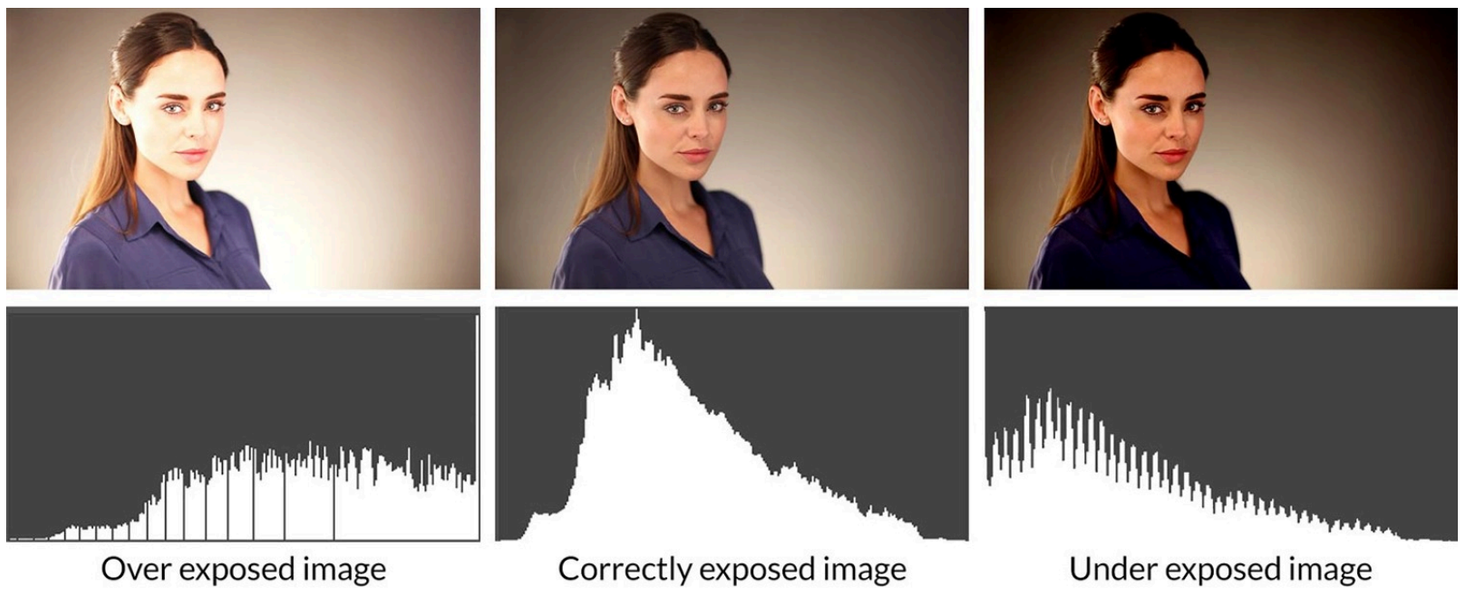


Sheffield Hallam University

Histograms & Contrast Foundations

What Is an Image Histogram?

- A histogram counts how many pixels fall into each intensity level
- For grayscale images, the x-axis is intensity, the y-axis is frequency
- It visualizes **dynamic range** and **contrast** at a glance
- **Mental model:** spreading the mass across the x-axis usually increases perceived contrast



Sheffield Hallam University

Building and Reading Histograms in MATLAB

```MATLAB
I = imread('pout.tif');     % example grayscale image
figure;
imhist(I);
title('Histogram of I');
```

- Peaks and valleys indicate dominant intensities and gaps
- Narrow cluster near the center → low contrast mid-tones
- Heavy mass near 0 or 255 → risk of clipping in dark or bright regions

Low vs High Contrast: Visual Patterns

- **Low contrast**
  - Histogram mass tightly clustered
  - Image looks flat or hazy
- **High contrast**
  - Histogram mass more spread across the range
  - Image appears crisper, edges more visible

⚠ **Caution:** “Flatter” is not always “better”. The best distribution is task dependent.

Dynamic Range, Clipping, and Headroom

- **Dynamic range**: the span of intensities used by the image
- **Clipping**: intensities pushed to 0 or 255 lose detail permanently
- **Headroom**: unused range at dark or bright ends that can be exploited by remapping

```MATLAB
minI = double(min(I(:))); % Minimum intensity
maxI = double(max(I(:))); % Maximum intensity

Range = maxI - minI;      % Dynamic range
```

Concept: Remapping Intensities

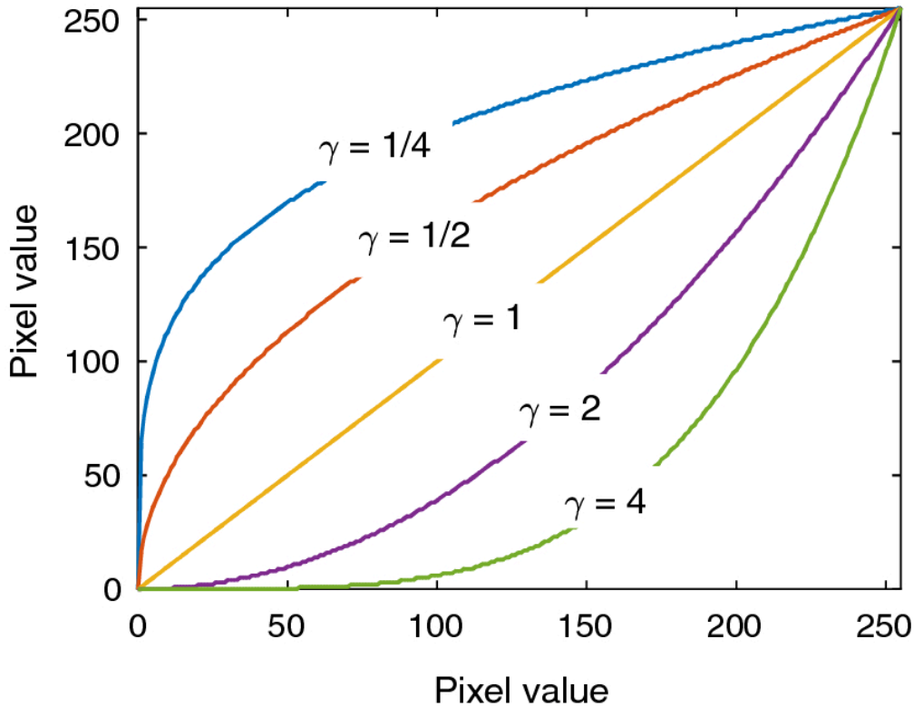
- We define a function f that maps input intensity x to output intensity $y = f(x)$
- Types of remapping
  - **Linear stretch**: expand used range into the full range
  - **Nonlinear**: emphasize specific regions (for example shadows or highlights)
- The histogram after remapping reflects how f redistributes pixels

Gamma Correction Intuition

- **Gamma** changes mid-tone emphasis while keeping black and white fixed
  - $\gamma < 1$ brightens mid-tones
  - $\gamma > 1$ darkens mid-tones
- Practical ranges to try: $0.6 \leq \gamma \leq 1.4$

```matlab
Iu = im2double(I);
gamma = 0.8;          % also try 0.6, 1.2, 1.4
J = Iu .^ gamma;
```

- Expect the histogram to bend toward the emphasized side



Transition

- You have the tools to **read** histograms and reason about contrast
- Next, we apply specific MATLAB operators for global and local enhancement:
  - Linear stretch and automatic limit selection
  - Histogram equalization and matching
  - CLAHE for local detail

Intensity Transformations: Global

Linear Stretch with imadjust

Idea

- Map the used intensity range to the full output range.
- Preserve ordering of pixel values.

Use when

- Image occupies a narrow band of intensities.
- You want a predictable, monotonic expansion.

Watch for

- Crushed shadows or highlights if limits are too aggressive.

```MATLAB
% A grayscale example
I = imread('pout.tif');

% Convert to double in [0,1] for imadjust
Iu = im2double(I);

% Default limits based on data
J1 = imadjust(Iu);

% Manual limits to avoid clipping
J2 = imadjust(Iu, [0.2 0.8], [0 1]);
```

Choosing Limits Automatically with stretchlim

The goal is to pick robust lower and upper bounds by trimming outliers.

- `stretchlim(I, [pLow pHigh])` controls tail trimming, for example `[0.02 0.98]`.
- Useful default for quick enhancement without manual tuning.
- **Trade off:** May remove faint details if tails contain valid structure.

```MATLAB
I = im2double(imread('pout.tif'));

% Default 1% trim at each end
L = stretchlim(I);

% Compute new image
J = imadjust(I, L, [0 1]);
```

Nonlinear Remapping with Gamma

The goal is to re-weight mid-tones while keeping black and white fixed.

- Gamma (γ) less than 1 brightens mid-tones.
- Gamma (γ) greater than 1 darkens mid-tones.
- **Risk:** Over brightening amplifies noise in shadows.

```MATLAB
I = im2double(imread('pout.tif'));

% Also try gamma = 0.6, 1.2, 1.4
gamma = 0.8;

% Apply gamma correction
J = imadjust(I, [], [], gamma);
```

Histogram Equalization with histeq

- **Concept:** Flatten the histogram to use the dynamic range more evenly.
- **Good for:** Revealing global contrast when the scene is evenly underexposed.

Limitations:

- Can over-amplify noise and produce unnatural tones.
- Flat histogram is not a universal objective.

```MATLAB
I = imread('pout.tif');

% Basic grayscale equalization
J  = histeq(I);

% Control the number of histogram bins
Jk = histeq(I, 256);

figure;
tiledlayout(1,3);
nexttile; imshow(I);  title('Original');
nexttile; imshow(J);  title('histeq default');
nexttile; imshow(Jk); title('histeq bins=256');
```

Match One Image to Another with imhistmatch

Use case

- Normalize a dataset to a reference appearance.

When helpful

- Quantitative comparisons across images.
- Preprocessing before classical feature extraction.

Caveat

- If the reference is unsuitable, the match can harm task performance.

```MATLAB
I   = imread('pout.tif');        % source
Ref = imread('cameraman.tif');   % reference

% Align the source histogram to the reference
J = imhistmatch(I, Ref);

figure;
tiledlayout(1,3);
nexttile; imshow(I);   title('Source');
nexttile; imshow(Ref); title('Reference');
nexttile; imshow(J);   title('Matched to Ref.');
```

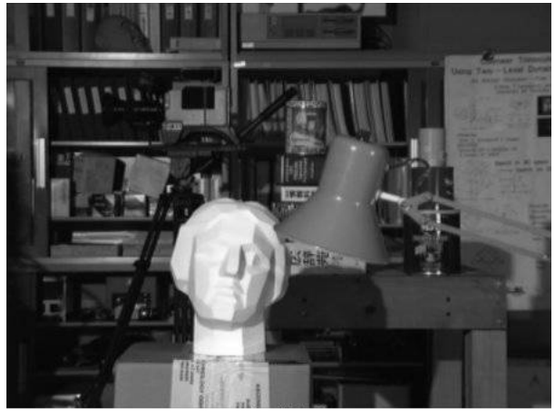
Putting It Together: Compare Global Methods

```MATLAB
I = im2double(imread('pout.tif'));

A = imadjust(I);                          % linear default
B = imadjust(I, stretchlim(I), [0 1]);    % auto stretch
C = imadjust(I, [], [], 0.8);             % gamma
D = histeq(im2uint8(I));                  % equalization needs uint types

tiledlayout(2,2);
nexttile; imshow(A); title('imadjust default');
nexttile; imshow(B); title('stretchlim + imadjust');
nexttile; imshow(C); title('gamma 0.8');
nexttile; imshow(D); title('histeq');
```

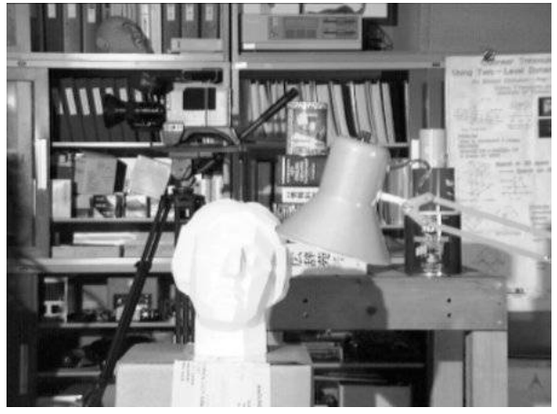
Local Contrast Enhancement (CLAHE)

Why Local Enhancement?

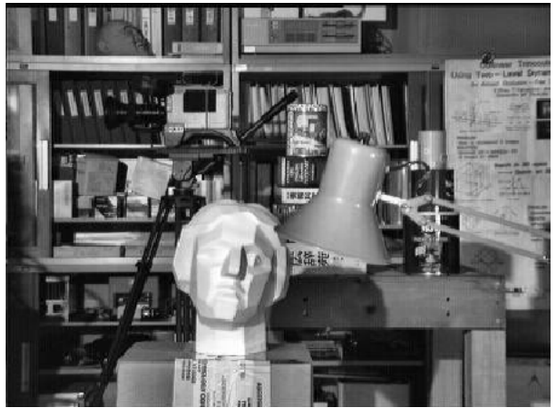
- Global methods struggle when illumination is **non-uniform**
- Important details can be **locally** low contrast even if the global histogram looks fine
- Solution: enhance contrast **per region** while controlling noise growth



Original Image



Global Stretch



Local Stretch (CLAHE)

Sheffield Hallam University

CLAHE Concept

- **CLAHE = Contrast Limited Adaptive Histogram Equalization**
- Partition the image into tiles, equalize each tile, then blend to avoid seams
- **Contrast limit** clips extreme histogram peaks to prevent noise amplification

> Intuition: gentle, local “spreading out” of intensities, with a safety valve

`adapthisteq` at a Glance

Key parameters

- `NumTiles`: Controls locality (e.g. `[8 8]`)
- `ClipLimit`: Caps amplification (e.g. `0.01` to `0.03`)
- `NBins`: Histogram resolution
- `Distribution`: Shape of output CDF (`'uniform' | 'rayleigh' | 'exponential'`)

Rules of thumb

- Start with `NumTiles=[8 8]`, `ClipLimit≈0.01`
- Increase `ClipLimit` cautiously if the result is too flat
- Decrease `NumTiles` if tile boundaries appear

```MATLAB
I  = imread('pout.tif');
Iu = im2double(I);

% Sensible defaults
J1 = adapthisteq(Iu);  % defaults

% Conservative tuning
J2 = adapthisteq(Iu, 'NumTiles', [8 8], ...
    'ClipLimit', 0.01);

% More aggressive local enhancement
J3 = adapthisteq(Iu, 'NumTiles', [16 16], ...
    'ClipLimit', 0.03, 'Distribution','rayleigh');
```

What Happens to the Histogram?

- Each tile is equalized, so local histograms become broader
- The **global** histogram may not be flat, and that's acceptable
- Expect better micro-contrast in textured areas and shadows

```matlab
J = adapthisteq(Iu,'ClipLimit',0.01);
figure;
tiledlayout(1,2);
nexttile; imhist(I);  title('Original (global)');
nexttile; imhist(im2uint8(J)); title('CLAHE result (global)');
```

Failure Modes and Mitigations

- **Tile boundaries** visible

  Reduce locality: larger tiles (for example `[8 8]` instead of `[16 16]`)

- **Noise amplification** in flat regions

  Lower `ClipLimit`, or apply light denoising after CLAHE

- **Over-processed appearance**

  Reduce both `NumTiles` and `ClipLimit`; prefer `'uniform'` distribution

Conservative vs Aggressive Settings

Conservative (safer)

```matlab
Jc = adapthisteq(Iu, ...
     'NumTiles', [8 8], ...
     'ClipLimit', 0.008, ...
     'Distribution','uniform');
```

- Minimal artifacts
- Gentle improvement in local detail
- Good first attempt for most images

Aggressive (use sparingly)

```matlab
Ja = adapthisteq(Iu, ...
     'NumTiles', [16 16], ...
     'ClipLimit', 0.03, ...
     'Distribution','rayleigh');
```

- Stronger local contrast
- Risk of haloing and noise boost
- Use when details are very faint

A Simple Decision Recipe

1. Try **global**: `stretchlim + imadjust`
2. If details remain hidden in regions, try **CLAHE**
3. Start with conservative parameters, inspect for artifacts
4. Adjust `ClipLimit` and `NumTiles` only as needed

```matlab
A = imadjust(Iu, stretchlim(Iu), [0 1]);
B = adapthisteq(Iu, 'NumTiles',[8 8], 'ClipLimit',0.01);
```

Common Pitfalls Checklist (1)

Mixing numeric classes and ranges

- Pitfall: Arithmetic on uint8 saturates or clips.
- Fix: Convert to double before math. Keep track of `[0, 1]` vs `[0, 255]`.

Blind histogram equalization

- Pitfall: Over-flattening that amplifies noise or destroys texture cues.
- Fix: Prefer CLAHE with conservative ClipLimit. Compare to linear stretch.

Aggressive CLAHE settings

- Pitfall: Tile boundaries and haloing.
- Fix: Start with `NumTiles = [8 8]`, `ClipLimit ≈ 0.01`. Increase slowly and verify.

Common Pitfalls Checklist (2)

Color space misuse

- Pitfall: Editing RGB channels shifts hue.
- Fix: Convert to Lab. Operate on L* when the goal is contrast only.

Inconsistent display and saving

- Pitfall: Display looks correct but saved image is washed out.
- Fix: Confirm class before `imshow` and `imwrite`. Document gamma and scaling.

Unjustified parameter choices

- Pitfall: “Tweaked until it looked good.”
- Fix: Log parameters, show before and after histograms, and state a rationale.

Wrap-up / Key Takeaways

- **Pick the right tool**:
  - Linear stretch (global range)
  - Gamma (mid-tones)
  - Histeq (uniform scenes)
  - CLAHE (non-uniform light).
- **Safe starters**:
  - `imadjust(I, stretchlim(I), [0 1])`
  - `adapthisteq(I, 'NumTiles', [8 8], 'ClipLimit', 0.01)`
- **Evaluate**: Check before/after histograms and look for artifacts.

Exit Ticket

✍️ Before you leave, write down:

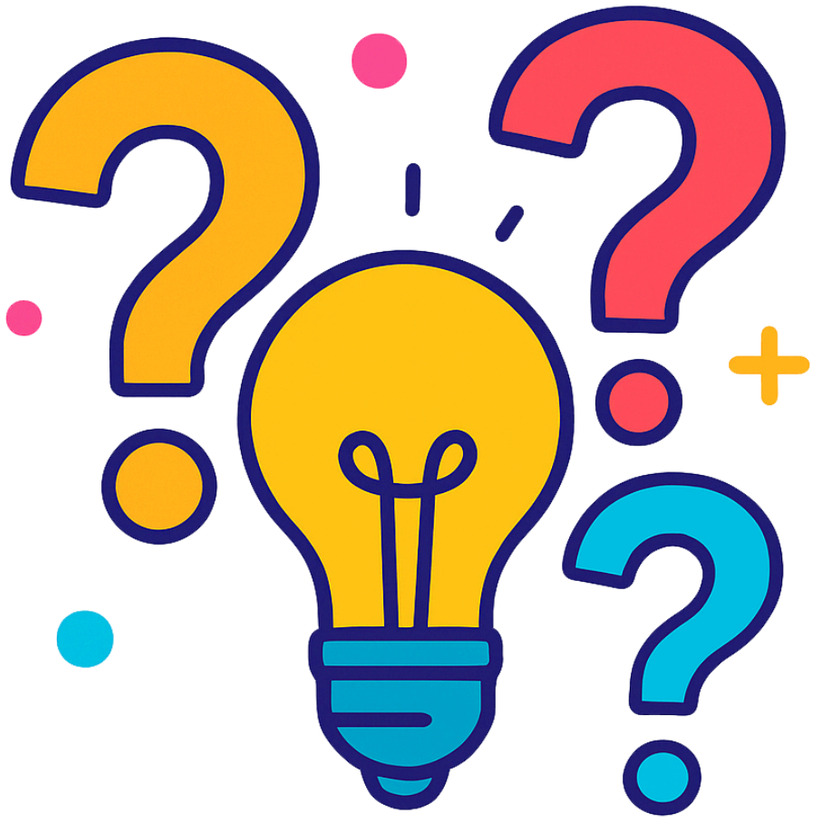
1. **One clear concept** from today
2. **One confusing concept**
3. **One thing to try in MATLAB** this week

Sheffield Hallam University

Questions & Support

- You are encouraged to **ask questions anytime** 💡
  - During the lecture
  - In lab sessions
  - By email → m.k.heris@shu.ac.uk
- **No question is too simple** — asking questions:
  - Helps you learn faster
  - Builds confidence
  - Improves understanding for the whole class

👉 If something is unclear, **just ask!**



Sheffield Hallam University