Machine Vision

Practical Contrast Enhancement with MATLAB

Lab Activity Sheet 2

Author: Dr. Mostapha Kalami Heris

1 Introduction

1.1 Why this lab matters

This lab builds on Week 1 (images as data, types and ranges, safe display and saving) and prepares you for later tasks where contrast directly affects feature extraction, edge quality, and measurement accuracy. You will read and interpret histograms, apply global intensity transforms, and use Contrast Limited Adaptive Histogram Equalization (CLAHE) for non-uniform illumination. You will also log parameters and justify choices so that your results are reproducible.

1.2 What you will learn today

By the end of this lab you should be able to:

1. interpret grayscale and color histograms;
2. apply and compare linear stretch, gamma correction, and histogram equalization;
3. perform histogram matching against a reference image;
4. enhance local contrast with CLAHE using safe defaults;
5. record parameters, identify artifacts, and justify your choices.

1.3 Quick refresher from Week 1

- **Images are arrays.** Grayscale images are $M \times N$, RGB images are $M \times N \times 3$. Indexed and binary images also appear in practice.

- **Data types and ranges.** `uint8` uses 0–255, `uint16` uses 0–$65{,}535$, and `double` is typically scaled to $[0, 1]$ in MATLAB. Arithmetic on `uint8` can saturate. Be explicit about class and expected range.
- **Display hygiene.** `imshow` respects the native range, while `imagesc` rescales to the axes limits. Choose deliberately and document the choice when you compare results.
- **File formats.** PNG and TIFF are lossless, JPEG is lossy. Use `imfinfo` to verify bit depth and important metadata when saving.

1.4 Histogram literacy

A histogram counts pixels per intensity and reveals dynamic range, clipping risk, and overall contrast at a glance. Spreading mass across the x-axis typically increases perceived contrast, but a flatter histogram is not always better. Context matters.
Starter (MATLAB).

```
I  = imread('pout.tif');
figure; imshow(I); title('Original');
figure; imhist(I); title('Histogram');
minI = double(min(I(:)));
maxI = double(max(I(:)));
```

1.5 Global intensity transforms

Use these when illumination is roughly uniform.

Linear stretch (`imadjust` with optional `stretchlim`). Maps the used range to the full output range and preserves ordering.

- Use when the image occupies a narrow intensity band.
- Watch for crushed shadows or highlights if limits are too aggressive.

Examples.

```
Iu = im2double(imread('pout.tif'));
J1 = imadjust(Iu);                              % default limits
J2 = imadjust(Iu, stretchlim(Iu), [0 1]);       % auto limits
J3 = imadjust(Iu, [0.2 0.8], [0 1]);            % manual limits
```

Gamma correction (`imadjust(...,[],[],gamma)`). Re-weights mid-tones while keeping black and white fixed.

- Practical range: $\gamma \in [0.6, 1.4]$.
- Risk: $\gamma < 1$ can amplify shadow noise; $\gamma > 1$ can mute fine details.

Example.

```
Jg = imadjust(Iu, [], [], 0.8);  % try 0.6, 1.2, 1.4
```

Histogram equalization (`histeq`). Flattens the histogram to use the available range more evenly.

- Good for evenly underexposed scenes.
- Limitations: can amplify noise and produce unnatural tones.

Example.

```
Iu8 = im2uint8(Iu);
Jeq = histeq(Iu8, 256);
```

Histogram matching (`imhistmatch`). Normalizes a source image to the appearance of a reference image.

- Choose a reference with care; a poor reference can degrade results.

Example.

```
Src = imread('pout.tif');
Ref = imread('cameraman.tif');
Jm  = imhistmatch(Src, Ref);
```

1.6 Local contrast with CLAHE (`adapthisteq`)

Use when illumination is non-uniform and global transforms are insufficient.
Key parameters.

- `NumTiles`: degree of locality, for example `[8 8]`.
- `ClipLimit`: limits amplification, typical values `0.008--0.03`.
- `NBins`: histogram resolution per tile.
- `Distribution`: output CDF shape, for example `'uniform'`’rayleigh’|’exponential’|.

Rule of thumb. Start with `NumTiles=[8 8]` and `ClipLimit=0.01`, then adjust cautiously.
Example.

```
Iu = im2double(imread('rice.png'));
Jc = adapthisteq(Iu, 'NumTiles',[8 8], ...
                    'ClipLimit',0.01, ...
                    'Distribution','uniform');
```

Diagnose and mitigate.

- Tile seams: reduce locality (fewer tiles) or increase blending.
- Noise in flat areas: lower `ClipLimit`.
- Over-processed look: reduce both `NumTiles` and `ClipLimit`.

1.7 Color-safe contrast edits

Editing RGB channels directly can change hue. When you only intend to change contrast, convert to CIELAB, edit L^*, then convert back to RGB.

1.8 Dataset and setup

Use MATLAB built-in images so that results are comparable across students, for example `cameraman.tif`, `pout.tif`, `peppers.png`, `rice.png`, and `coloredChips.png`.

Comparison helper function. Store the following code inside the file `showpair.m` in your project directory, so you can use it in your codes.

```
function showpair(A,B,t1,t2)
    figure; tiledlayout(2,2);
    nexttile; imshow(A); title(['A: ' t1]);
    nexttile; imhist(im2uint8(A)); title('A histogram');
    nexttile; imshow(B); title(['B: ' t2]);
    nexttile; imhist(im2uint8(B)); title('B histogram');
end
```

1.9 Safety checklist before activities

- Convert to `double` before mathematical operations; track whether values are in $[0, 1]$ or $[0, 255]$.

- Compare linear stretch against equalization before adopting the latter.
- With CLAHE, start conservatively; watch for tile boundaries, haloing, and noise.
- For color images, prefer L^* edits when you only want contrast changes.
- Log parameters and ranges; support comparisons with histograms and a short written rationale.

2 Histogram Literacy & Global Transforms

2.1 Aim

Build practical fluency with reading histograms and applying global contrast methods in MATLAB. You will follow structured steps, make small parameter choices, and document outcomes in a reproducible way for your own learning.

2.2 Outcomes

By the end of Activity 1 you can:

1. read a histogram to infer dynamic range, clipping risk, and contrast;
2. apply linear stretch, gamma correction, and histogram equalization correctly;
3. justify chosen parameters with short notes and side-by-side comparisons.

2.3 Setup

- Use MATLAB built-in images: `pout.tif`, `cameraman.tif`. You may add one more image of your choice if time permits.
- Create a new script with a clear section for each method. Keep a parameter table as you work for your own reference.
- Adopt the helper function `showpair` from the Introduction for consistent comparisons.

2.4 Instructions

A. Load and inspect.

```
I  = imread('pout.tif');          % grayscale
Iu = im2double(I);                % work in [0,1]
figure; imshow(I);       title('Original');
figure; imhist(I);       title('Original histogram');
minI = min(Iu(:)); maxI = max(Iu(:));
rngWidth = maxI - minI;
```

Record **min**, **max**, and **range**. State in one sentence what the histogram suggests (for example, dark-biased, narrow range, near-clipping).

B. Linear stretch (`imadjust` with and without `stretchlim`).

```
J1 = imadjust(Iu);                              % default in/out
L  = stretchlim(Iu);                            % auto input limits
J2 = imadjust(Iu, L, [0 1]);                    % auto stretch
J3 = imadjust(Iu, [0.20 0.80], [0 1]);          % manual limits
showpair(Iu, J1, 'Original', 'imadjust default');
showpair(Iu, J2, 'Original', 'imadjust + stretchlim');
showpair(Iu, J3, 'Original', 'imadjust [0.2 0.8] -> [0 1]');
```

Note: Aggressive limits can crush shadows or highlights. If you see clipping, relax the limits.

C. Gamma correction (γ in $[0.6, 1.4]$).

```
Jg1 = imadjust(Iu, [], [], 0.6);
Jg2 = imadjust(Iu, [], [], 0.8);
Jg3 = imadjust(Iu, [], [], 1.2);
showpair(Iu, Jg2, 'Original', 'Gamma = 0.8');
```

Note: $\gamma < 1$ brightens midtones and can amplify shadow noise. $\gamma > 1$ darkens midtones and may hide detail.

D. Histogram equalization (`histeq`).

```
I8  = im2uint8(Iu);               % histeq expects integer types
Jeq = histeq(I8, 256);
showpair(Iu, im2double(Jeq), 'Original', 'histeq 256 bins');
```

Note: Equalization can over-flatten and produce an unnatural look on some images. Compare with linear stretch before deciding.

2.5 Try these as you go (minor contributions)

1. For linear stretch, try two manual input ranges, for example $[0.15, 0.85]$ and $[0.05, 0.95]$. Briefly note which is better and why.
2. For gamma, test one darker image and one brighter image. Note the gamma that best improves midtone visibility for each.
3. For equalization, switch the source image to `cameraman.tif` and repeat once. Describe any visible noise amplification.

2.6 Common pitfalls and safeguards

- Performing arithmetic on `uint8` and causing silent saturation. Convert to `double` first.
- Assuming a flatter histogram is always better. Judge by task and artifacts.
- Overusing equalization when a gentle linear stretch would suffice.

2.7 If you finish early

Apply your best settings to a second image and write one short paragraph that compares how image content influences the parameters you consider "best." Include at least one histogram comparison to support your observation.

3 Histogram Matching & Color-safe Enhancement

3.1 Aim

Explore appearance normalization via histogram matching and practice contrast enhancement on color images without altering hue. You will make design choices, consult MATLAB documentation as needed, and justify your reasoning with brief notes for yourself.

3.2 Outcomes

By the end of Activity 2 you can:

1. use `imhistmatch` to align the tonal appearance of one image to a reference;
2. choose a sensible reference image and explain why it is appropriate;

3. enhance contrast in color images by operating on the L^* channel in CIELAB while preserving hue.

3.3 Setup

- Select two grayscale images with different looks, for example `pout.tif` and `cameraman.tif`.
- Select one RGB image, for example `peppers.png`.
- Keep your Activity 1 `showpair` helper available for comparisons.

3.4 Instructions

A. Histogram matching (grayscale). Pick a source and a reference. Match the source histogram to the reference and compare with your best global method from Activity 1.

```
Src = imread('pout.tif');    % source
Ref = imread('cameraman.tif');% reference
M   = imhistmatch(Src, Ref);

showpair(im2double(Src), im2double(M), ...
         'Source', 'Histogram-matched to Ref');
figure; imhist(Src); title('Source histogram');
figure; imhist(Ref); title('Reference histogram');
figure; imhist(M);   title('Matched histogram');
```

Reflection prompt: When does matching help, and when does it introduce artifacts or reduce local contrast? Note one benefit and one drawback for your pair.

B. Color-safe enhancement in CIELAB. Convert to CIELAB, enhance only L^*, then convert back to RGB.

```
Irgb = im2double(imread('peppers.png'));   % work in [0,1]
Ilab = rgb2lab(Irgb);                      % L* in [0,100]
L    = Ilab(:,:,1) / 100;                  % normalize L* to [0,1]

% Option 1: gentle global adjustment on L*
L1 = imadjust(L, stretchlim(L), [0 1]);

% Option 2: conservative CLAHE on L*
```

```
L2 = adapthisteq(L, 'NumTiles',[8 8], 'ClipLimit',0.01, ...
                     'Distribution','uniform');

% Recompose (scale back to [0,100] for lab2rgb)
Ilab1 = Ilab; Ilab1(:,:,1) = L1 * 100;
Ilab2 = Ilab; Ilab2(:,:,1) = L2 * 100;

J1 = lab2rgb(Ilab1);   % global on L*
J2 = lab2rgb(Ilab2);   % CLAHE on L*

showpair(Irgb, J1, 'Original RGB', 'L* global adjusted');
showpair(Irgb, J2, 'Original RGB', 'L* CLAHE');
```

Reflection prompt: Which L* method better preserves natural color while improving detail, and why. If you see haloing or an over-processed look, reduce locality (fewer tiles) or lower `ClipLimit`.

3.5 Try these as you go

1. Swap the reference image in histogram matching to a very different scene. Note how the choice of reference changes the outcome.
2. Replace `stretchlim` with a manual input range in the L* global adjustment and record when clipping appears.
3. Test `adapthisteq` with `NumTiles=[6 6]` and `[10 10]` on L*. Observe tile boundary artifacts and adjust.

3.6 Common pitfalls and safeguards

- Using an unsuitable reference for `imhistmatch` that forces an unnatural look. Choose a reference with similar content and exposure intent.
- Editing RGB channels directly for contrast and unintentionally shifting hue. Prefer L* edits when the goal is contrast only.
- Over-aggressive CLAHE on L* causing noise amplification in flat regions. Lower `ClipLimit` or reduce the number of tiles.

3.7 If you finish early

Apply your preferred L* method to a second RGB image and write two sentences explaining how scene content affects the parameters you consider reasonable. Include one histogram of the L* channel before and after to support your observation.

4 Local Contrast with CLAHE & Failure-mode Diagnosis

4.1 Aim

Tackle non-uniform illumination and subtle detail loss using local methods. You will design and tune a CLAHE workflow, diagnose artifacts, and apply principled mitigations using MATLAB documentation and brief web searches as needed.

4.2 Outcomes

By the end of Activity 3 you can:

1. decide when a global method is insufficient and a local method is appropriate;
2. tune `adapthisteq` parameters to improve local detail while controlling artifacts;
3. identify and mitigate common failure modes such as tile seams, haloing, and noise amplification.

4.3 Setup

- Choose a low-contrast or unevenly illuminated image, for example `rice.png` or a photo you captured.
- Keep your Activity 1 global best result and code handy for side-by-side comparisons.
- Open MATLAB documentation for `adapthisteq` and `imhist` for quick reference.

4.4 Instructions

A. Baseline with a global method. Start with your best global approach from Activity 1 to establish a fair baseline.

```
I   = im2double(imread('rice.png'));
Jg  = imadjust(I, stretchlim(I), [0 1]);   % or your preferred global method
showpair(I, Jg, 'Original', 'Global baseline');
```

B. First-pass CLAHE with conservative settings. Begin gently, then iterate in small steps.

```
Jc = adapthisteq(I, 'NumTiles',[8 8], 'ClipLimit',0.01, ...
                    'Distribution','uniform');   % conservative start
showpair(I, Jc, 'Original', 'CLAHE [8 8], Clip=0.01');
```

C. Diagnose artifacts and iterate. Use the checklist below. Change one parameter at a time and observe histograms and details.

- **Tile seams or blockiness**: reduce locality `NumTiles=[6 6]` or `[4 4]`.
- **Haloing around edges**: lower `ClipLimit`, consider `Distribution='uniform'`.
- **Noise in flat regions**: lower `ClipLimit` or slightly increase `NBins`.
- **Over-processed look**: reduce both `NumTiles` and `ClipLimit`.

```
Jc2 = adapthisteq(I, 'NumTiles',[6 6], 'ClipLimit',0.008, ...
                     'Distribution','uniform', 'NBins',256);
showpair(Jc, Jc2, 'CLAHE v1', 'CLAHE v2 (tuned)');
```

D. Optional: apply CLAHE only where needed. For images that mix bright and dark areas, try a mask-driven approach to avoid over-enhancing already good regions.

```
% Simple luminance mask (threshold can be tuned)
M   = I < graythresh(I);                  % darker regions
Jcx = adapthisteq(I, 'NumTiles',[6 6], 'ClipLimit',0.008);
Jmm = I; Jmm(M) = Jcx(M);                 % merge selectively
showpair(I, Jmm, 'Original', 'Masked CLAHE on darker regions');
```

4.5 Try these as you go

1. Compare `Distribution='uniform'` vs `'rayleigh'`. Note which better preserves natural texture on your image.
2. Increase locality to `NumTiles=[12 12]` and observe when seams begin to appear. Back off to the largest value that remains artifact-free.
3. For a color image, repeat in CIELAB by applying CLAHE to L^* only, then convert back to RGB. Compare with your global L* result from Activity 2.

4.6 Common pitfalls and safeguards

- Pushing `ClipLimit` too high to force contrast, which amplifies noise and haloing. Prefer small, incremental changes.
- Using too many tiles on small images, which creates block artifacts. Match `NumTiles` to image size and content.
- Judging results only by the histogram. Always inspect fine structures and flat regions at 100% zoom.

4.7 If you finish early

Create a short recipe that lists your default starting values for CLAHE and how you branch based on observed artifacts. Apply your recipe to a second image to test its robustness.

5 Optional Extensions

5.1 Explorations if time remains

- **Adaptive locality**: Compare `NumTiles=[6 6]` vs `[10 10]` on images of different sizes.
- **Distribution variants**: Try `Distribution='rayleigh'` and `'exponential'` and note visual differences.
- **Masked enhancement**: Apply CLAHE only to dim regions using a luminance mask; compare to global CLAHE.
- **Color fidelity test**: Enhance an RGB image by per-channel `imadjust` vs L^*-only and inspect hue shifts.

5.2 Thinking prompts

- When is histogram matching preferable to gamma correction, and why.
- What failure modes signal that local methods are overused.
- How would you select parameters for images intended for quantitative measurement rather than aesthetics.

6 Key Takeaways

- Inspect histograms, but judge quality on image content and task requirements.
- Prefer gentle global methods first; move to local enhancement when illumination is non-uniform.
- Tune CLAHE conservatively. Reduce tiles or clip limit at the first sign of seams, haloing, or noise.
- For color images, operate on L^* in CIELAB to preserve hue when contrast is the only goal.
- Keep notes of parameters and observed effects to build reusable decision recipes.