# Department of Computer Engineering
# University of Peradeniya
# CO543: Image Processing - Lab 2 (2025)

---

## Illumination, Contrast & Thresholding Fundamentals

Real-world images rarely come perfectly lit. Cameras encounter shadows, glare, over-exposure, under-exposure, and uneven lighting. This laboratory teaches how to enhance color tone, normalize brightness, and separate objects from the background using pixel intensity analysis. You will use the provided images captured in low contrast, high contrast, bright, and dim lighting conditions to understand how intensity transformations shape perception.

---

### Task 1 — Visualize Intensity & Histograms

Load each lighting-variant image and display its histogram. Then compare visually where pixel values cluster and how tonal distribution affects scene interpretation. In your report, describe how the histogram shape relates to the lighting condition and why understanding distribution is essential for image processing.

```
#Code hint:
img = cv2.imread("dark.tif",0)
hist = cv2.calcHist([img],[0],None,[256],[0,256])
plt.plot(hist)
```

**More to think:**
If two images have similar histograms, can they still look visually different? Why?

---

## Task 2 — Manual Threshold & Observe Failure Cases

Apply a fixed threshold to each image and examine where it succeeds and fails at separating foreground from background. In your report, discuss why the same threshold cannot handle all illumination conditions and identify regions where misclassification occurs.

```
#Code hint:
_,th = cv2.threshold(img,120,255,cv2.THRESH_BINARY)
```

**More to think:**
Is thresholding really “deciding” what an object is, or just comparing brightness?

---

## Task 3 — Apply Otsu & Adaptive Thresholding

Use Otsu’s method and adaptive thresholding to address uneven lighting. Compare results to manual thresholding and explain why data-driven decisions often outperform fixed pixel cutoffs.

```
#Code hint:
_,otsu = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
adaptive = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                 cv2.THRESH_BINARY,11,2)
```

**More to think:**
Why can adaptive thresholding still struggle on textured backgrounds?

---

## Task 4 — Gamma & Log Transformations

Enhance dark images with gamma < 1 and reduce brightness in over-exposed images using gamma > 1. Apply log transform for subtle contrast expansion. In your report, describe how these transformations mimic camera exposure control and when each is appropriate.

```
#Code hint:
gamma = 0.5
corrected = np.power(img/255.0, gamma)
```

**More to think:**
Does gamma correction change object identity or only perception?

---

## Task 5 — Contrast Stretching & Histogram Equalization

Normalize pixel range with linear stretching, then apply histogram equalization. Compare the differences — one adjusts the linear scale, the other redistributes intensities.

```
#Code hint:
eq = cv2.equalizeHist(img)
```

**More to think:**
When can histogram equalization amplify noise instead of details?

---

## Task 6 — Bit-Plane Slicing

Extract different bit planes and visualize how each layer contributes to the image. In your report, discuss why higher bit-planes carry semantic structure while lower bits capture texture/noise.

```
#Code hint:
bit4 = (img & 16) * 16
```

**More to think:**
Could accessing lower bit-planes ever reveal hidden watermarks or metadata?

---

## Final Reflection

In a single paragraph, explain how illumination and histogram shape influence segmentation and perception. Which transformation felt most powerful, and which seemed risky without careful use?

---

## Submission

Submit **one Jupyter notebook (.ipynb)** with:

- *Code for each task (concise, no redundant cells)*
- *Output images shown in-notebook*
- *Short answers where required*
- *Final reflection at the end*
- ***E_21_xxx_Lab_yy.ipynb** (xxx = reg no., yy = lab number)*

---