Department of Computer Engineering
University of Peradeniya
CO543: Image Processing - Lab 1 (2025)

Seeing Images as Numbers & Understanding Basic Transformations

A computer sees the world as numbers. Before models, filters, segmentation, or recognition, you must understand how an image exists and how simple pixel-space operations change meaning. In this laboratory, you explore this by viewing an image the way algorithms see it, and by reconstructing handwritten digits from raw numeric CSV data.

Task 1 — View the Image as an Array of Numbers

Load the scenic image and display it. Then print a small 5×5 pixel patch to expose raw values. Observe how numerical patterns translate into smooth shapes and shading. In your report, explain how the pixel values relate to what you see, and why computers need to work at this level of representation.

```
#Code hint:
img = cv2.imread("image.jpg")
print(img[100:105, 100:105])
plt.imshow(img[:,:,::-1])
```

More to think:
If two very different images share similar numeric ranges, why do they still look different?

Task 2 — Convert to Grayscale and Inspect Structure

Convert the same image to grayscale and inspect another small pixel region. Describe in your report what information is lost when removing color, what is preserved, and why grayscale is the foundation for many early-stage image processing operations.

```
#Code hint:
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print(gray[100:105, 100:105])
```

More to think:
If grayscale simplifies computation, why don't we always use grayscale in modern vision systems?

Task 3 — Resize and Zoom to Study Resolution Effects

Resize the image down and then resize it up again. Zoom into a detailed edge region (mountain boundary). In your report, discuss what visual details disappear and how pixelation or blur emerge and how reducing resolution might affect image processing tasks.

```
#Code hint:
small = cv2.resize(gray, (100,100))
big = cv2.resize(small, (500,500))
```

More to think:
Can reduction of resolution ever *help* image processing tasks? (Hint: noise vs structure)

Task 4 — Crop to Focus on Meaningful Regions

Crop one central region and one edge region. Explain in your report how cropping isolates regions of interest and why spatial context sometimes matter more than isolated detail. Relate this to face crops vs whole-scene understanding.

```
#Code hint:
crop = gray[120:220, 120:220]
```

More to think:
Would cropping always help recognition? Or can cropping remove crucial context?

Task 5 — Flip and Rotate to Explore Viewpoint Changes

Flip the image horizontally, vertically, and rotate at different angles. In your report, comment on which transformations preserve interpretation and which distort meaning. Explain how viewpoint variation relates to data augmentation and real camera motion.

```
#Code hint:
flip_h = cv2.flip(gray,1)
M = cv2.getRotationMatrix2D(center=(cx,cy), angle=45, scale=1.0)
rot = cv2.warpAffine(gray, M, (w,h))
```

More to think:
Should a recognition model treat flipped objects as the same or different?

Task 6 — Invert Pixel Intensities (Negative Image)

Invert the image intensities and visualize the result. In your report, explain why inversion disrupts interpretation and when negative imaging might be beneficial (e.g., medical contrast, forensic enhancement).

```
#Code hint:
inv = cv2.bitwise_not(gray)
```

More to think:
Could inversion ever reveal patterns that the human eye misses?

Task 7 — Convert Digit CSV to Images

Load the digit CSV file, reshape each row into 28×28, and display samples. In your report, describe how a numeric table becomes a meaningful shape and why machine learning depends on correct reshaping.

```
#Code hint:
data = pd.read_csv("Digits_Lab_01.csv").to_numpy()
digit = data[0].reshape(28,28)
plt.imshow(digit, cmap='gray')
```

More to think:
If shuffled row order still shows recognisable digits when reshaped, what does that imply about spatial encoding?

Task 8 — Apply a Simple Affine Transform

Apply shear or shift to the scenic image. In your report, explain how these distortions affect object shapes and relate this to camera perspective effects.

```
#Code hint:
pts1 = np.float32([[0,0],[100,0],[0,100]])
pts2 = np.float32([[0,0],[80,20],[20,100]])
M = cv2.getAffineTransform(pts1, pts2)
aff = cv2.warpAffine(gray, M, (w,h))
```

More to think:
Would a classifier trained on undistorted images recognize sheared versions?

Final Reflection

Write one coherent paragraph answering:

> When you view images as numeric matrices and apply simple pixel operations, how does your perception of "what an image is" change? Which transformation taught you the most about how computers interpret structure instead of meaning?

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)*
