National University
Of Computer and Emerging Sciences

Department of Computer Science

Name :Tazmeen Afroz **Date:** 4 Nov 2025
Semester: 7th **ID :22P-9252**

Computer Vision

Lab 6 &7: SIFT

Lab Task 1 – Scale-Space and Difference-of-Gaussians (DoG) Construction

Lab Task 2 – Keypoint Detection and Refinement

Objective: Detect and filter keypoints using DoG extrema and local gradient analysis.

Steps:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

# %%
img = cv2.imread('green.jpg', cv2.IMREAD_GRAYSCALE)

image_h, image_w = img.shape

sigma_base = 0.707
k = np.sqrt(2)
octaves = []

for octave_i in range(4):
```

```
    if octave_i == 0:
        current_img = img
    else:
        current_img = cv2.resize(current_img, (current_img.shape[1]//2,
current_img.shape[0]//2), interpolation=cv2.INTER_LINEAR)

    sigma_values = []
    gaussian_images = []

    # Keep 5 scales as original
    for scale_i in range(5):
        current_sigma = sigma_base * (k ** scale_i)
        sigma_values.append(current_sigma)

        blurred_img = cv2.GaussianBlur(current_img, (0, 0), sigmaX=current_sigma,
sigmaY=current_sigma, borderType=cv2.BORDER_DEFAULT)
        gaussian_images.append(blurred_img)

    octaves.append({
        'gaussian_images': gaussian_images,
        'sigma_values': sigma_values,
        'shape': current_img.shape
    })

# %%
# Visualize Gaussian Pyramid
fig, axes = plt.subplots(4, 5, figsize=(20, 16))
fig.suptitle('SIFT Gaussian Pyramid - 4 Octaves × 5 Scales', fontsize=16)

for octave_i in range(4):
    for scale_i in range(5):
        ax = axes[octave_i, scale_i]
        img_to_show = octaves[octave_i]['gaussian_images'][scale_i]
        sigma_val = octaves[octave_i]['sigma_values'][scale_i]
```

```
        ax.imshow(img_to_show, cmap='gray')
        ax.set_title(f'Octave {octave_i}, σ={sigma_val:.3f}')
        ax.axis('off')

plt.tight_layout()
plt.show()
```

1. Identify potential keypoints by checking local extrema in 3×3×3 neighborhoods across DoG images.
2. Eliminate low-contrast points (e.g., |DoG| < 0.04).
3. Remove edge-like points using the principal curvature ratio (Hessian test).
4. Visualize detected keypoints on the original image using cv2.circle() or plt.scatter().

Lab Task 3 – Orientation Assignment and Descriptor Generation

Objective: Compute orientation and generate 128-dimensional SIFT descriptors.

Steps:

1. For each keypoint, compute gradient magnitude and orientation within a circular window.
2. Construct an orientation histogram (36 bins, 10° each) weighted by gradient magnitude and Gaussian distance.
3. Assign one or more dominant orientations to each keypoint.
4. Divide the local patch into 4×4 subregions; compute an 8-bin orientation histogram for each region.
5. Flatten, normalize, and visualize the descriptor.
6. Compare the output with cv2.SIFT_create() for validation.

Expected Output:

- Keypoints with orientations drawn as arrows or lines.
- Descriptor arrays printed or visualized as histograms.
- Side-by-side comparison with OpenCV's SIFT results.