

**EXP NO: 6****FEATURE DETECTION USING SIFT AND HOG****AIM:**

To implement feature detection techniques using Scale Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) for image analysis and object recognition.

**THEORY:**

Feature detection and extraction are fundamental tasks in computer vision. They are used to identify key points and descriptors that represent the unique structures in an image.

- **SIFT (Scale Invariant Feature Transform):** Detects local features in an image that are invariant to scale, rotation, and illumination. It extracts key points and descriptors for matching between images.
- **HOG (Histogram of Oriented Gradients):** Captures edge and gradient structures in localized portions of the image. It is widely used for object detection (e.g., pedestrian detection).

**REQUIREMENTS:**

- Python 3.x
- OpenCV
- Matplotlib
- NumPy
- Scikit-image

**ALGORITHM FOR SIFT:**

Step 1: Start

Step 2: Read the input image using `cv2.imread()`.

Step 3: Convert the image to grayscale using `cv2.cvtColor()`.

Step 4: Create a SIFT detector using `cv2.SIFT_create()`.

Step 5: Detect keypoints and compute descriptors using `detectAndCompute()`.

Step 6: Draw the keypoints on the image using `cv2.drawKeypoints()`.

Step 7: Display the image with detected features.

Step 8: End

**ALGORITHM FOR HOG:**

Step 1: Start

Step 2: Read the input image and convert it to grayscale.

Step 3: Normalize the image size for consistency.

Step 4: Compute gradients (magnitude and orientation) of the image.

Step 5: Divide the image into small connected regions called cells.

Step 6: Compute histogram of gradients for each cell.

Step 7: Normalize histograms across blocks of cells for illumination invariance.

Step 8: Concatenate the histograms to form the HOG descriptor.

Step 9: Visualize the HOG features.

Step 10: End

**CODE:**

```
# Experiment 6: Feature Detection using SIFT and HOG
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.feature import hog
from skimage import color, exposure
from google.colab import files

print("Upload your image file ")
uploaded = files.upload()
# Path after upload
image_path = "/content/img.jpg"

def sift_feature_detection(image_path):
    img = cv2.imread(image_path)
    if img is None:
        print("Image not found!")
        return

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Create SIFT detector
```

```
sift = cv2.SIFT_create()

# Detect keypoints and descriptors
keypoints, descriptors = sift.detectAndCompute(gray, None)

# Draw keypoints
img_sift = cv2.drawKeypoints(img, keypoints, None,
                             flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

# Display
plt.figure(figsize=(8, 6))
plt.title("SIFT Feature Detection")
plt.imshow(cv2.cvtColor(img_sift, cv2.COLOR_BGR2RGB))
plt.axis("off")
plt.show()

# -----
# HOG Implementation
# -----
def hog_feature_detection(image_path):
    img = cv2.imread(image_path)
    if img is None:
        print("Image not found!")
        return

    gray = color.rgb2gray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

    # Normalize size
    resized_img = cv2.resize(gray, (128, 128))

    # Compute HOG
```

```
hog_features, hog_image = hog(resized_img, orientations=9, pixels_per_cell=(8, 8),
                              cells_per_block=(2, 2), block_norm='L2-Hys',
                              visualize=True, feature_vector=True)

hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

# Display
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6), sharex=True, sharey=True)

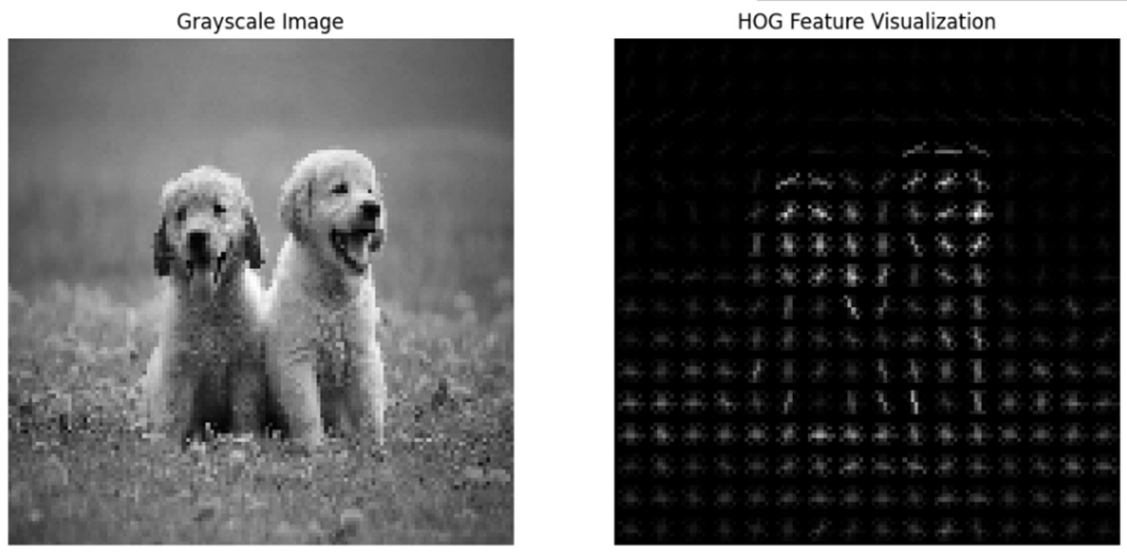
ax1.axis("off")
ax1.imshow(resized_img, cmap=plt.cm.gray)
ax1.set_title("Grayscale Image")

ax2.axis("off")
ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
ax2.set_title("HOG Feature Visualization")

plt.show()

# -----
# Run both methods
# -----
sift_feature_detection(image_path)
hog_feature_detection(image_path)
```

**OUTPUT:**

**RESULT:**

Key features in the image were successfully detected using SIFT, highlighting distinctive points and key points. HOG extracted gradient-based features, effectively capturing the shape and structure of objects. Both methods produced feature representations suitable for image analysis and recognition tasks.