EXPT 7	Feature Detection

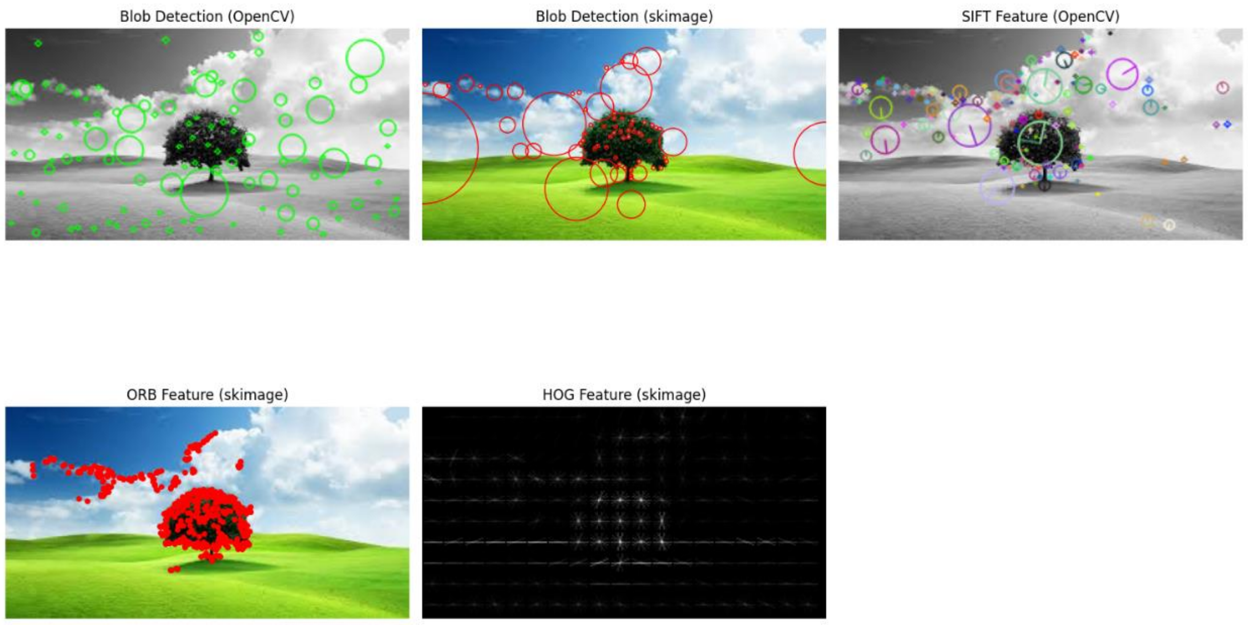
AIM:-

To implement advanced feature detection techniques including blob detection for identifying regions of interest, SIFT for scale-invariant keypoint detection, and Histogram of Oriented Gradients (HOG) for object descriptor generation.
CODE:-

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
from skimage import io, color, feature
from skimage.feature import hog
from skimage.util import img_as_ubyte

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    print("Upload 'input.jpg' to Colab.")
else:
    # Ensure image is 8-bit for OpenCV functions
    img_ubyte = img_as_ubyte(img)

    # Blob Detection (OpenCV)
    params = cv2.SimpleBlobDetector_Params()
    params.filterByArea = True
    params.minArea = 5
    params.maxArea = 5000
    params.filterByCircularity = False
    params.filterByInertia = False
    params.filterByConvexity = False
    params.filterByColor = False
    params.minThreshold = 0
    params.maxThreshold = 255
    detector = cv2.SimpleBlobDetector_create(params)
    keypoints_cv = detector.detect(img_ubyte)
    result_blob_cv = cv2.drawKeypoints(img_ubyte, keypoints_cv, np.array([]),
(0,255,0), cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

    # Blob Detection (skimage)
    img_sk = io.imread('input.jpg')
    if img_sk.ndim == 3:
        img_gray_sk = color.rgb2gray(img_sk)
    else:
        img_gray_sk = img_sk
    blobs_sk = feature.blob_log(img_gray_sk, max_sigma=30, num_sigma=10,
threshold=0.1)

    # SIFT Feature Detection (OpenCV)
    sift = cv2.SIFT_create()
    keypoints_sift_cv, descriptors_sift_cv = sift.detectAndCompute(img_ubyte,
None)
    result_sift_cv = cv2.drawKeypoints(img_ubyte, keypoints_sift_cv, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

    # SIFT Feature Detection (skimage - using ORB as alternative)
    orb = feature.ORB(n_keypoints=500)
    orb.detect_and_extract(img_gray_sk)
    keypoints_orb_sk = orb.keypoints


    # HOG Feature (skimage)
    fd_hog_sk, hog_image_sk = hog(img_gray_sk, pixels_per_cell=(16, 16),
cells_per_block=(2, 2), visualize=True, feature_vector=True)

    # Create subplots
    fig, axes = plt.subplots(2, 3, figsize=(15, 10))
```

```python
ax = axes.ravel()

# Display images
ax[0].imshow(cv2.cvtColor(result_blob_cv, cv2.COLOR_BGR2RGB))
ax[0].set_title("Blob Detection (OpenCV)")
ax[0].axis('off')

ax[1].imshow(img_sk)
for blob, radius in zip(blobs_sk, blobs_sk[:, 2] * np.sqrt(2)):
    y, x = blob[:2]
    circle = plt.Circle((x, y), radius, color='r', fill=False)
    ax[1].add_patch(circle)
ax[1].set_title("Blob Detection (skimage)")
ax[1].axis('off')

ax[2].imshow(cv2.cvtColor(result_sift_cv, cv2.COLOR_BGR2RGB))
ax[2].set_title("SIFT Feature (OpenCV)")
ax[2].axis('off')

ax[3].imshow(img_sk)
ax[3].scatter(keypoints_orb_sk[:, 1], keypoints_orb_sk[:, 0], c='r', s=15)
ax[3].set_title('ORB Feature (skimage)')
ax[3].axis('off')

ax[4].imshow(hog_image_sk, cmap='gray')
ax[4].set_title("HOG Feature (skimage)")
ax[4].axis('off')

# Hide the unused subplot
fig.delaxes(ax[5])


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

OUTPUT:-



RESULT:-

Successfully extracted distinctive features from images using various detection algorithms. The methods demonstrated robustness to scale, rotation, and illumination changes, making them suitable for object recognition and image matching applications.