Feature Descriptor

1. Extract HOG feature of a given image
2. Extract SIFT: Scale Invariant Feature Transform feature.
3. Extract SURF: feature Speeded-Up Robust Feature

```
from google.colab.patches import cv2_imshow
import cv2
import numpy as np
from skimage.feature import hog
from matplotlib import pyplot as plt
```

Extract HOG (Histogram of Oriented Gradients) Features

Histogram of Oriented Gradients (HOG) is a feature descriptor widely used in computer vision and image processing tasks like object detection and recognition. It captures the local gradient information of an image by dividing it into small regions and computing histograms of gradient orientations within each region.

Theory:

The main intuition behind HOG is that local object appearance and shape within an image can be characterized by the distribution of intensity gradients or edge directions. By computing histograms of gradient orientations over local regions of an image, we can capture the shape and texture information effectively.

The formula to compute the Histogram of Oriented Gradients (HOG) features involves the following steps:

1. Compute the gradient magnitude $G(x, y)$ and orientation $\theta(x, y)$ for each pixel in the image.
2. Divide the image into small cells of a fixed size.
3. For each cell, create a histogram of gradient orientations by accumulating the gradient magnitudes in the corresponding orientation bins.
4. Optionally, perform normalization within each block of cells to improve invariance to changes in illumination and contrast.
5. Concatenate the histograms of all cells to form the final feature vector H.

The specific formula for computing the histogram of oriented gradients within a cell can be represented as follows:

Let's denote:

- $G(x, y)$ as the gradient magnitude at pixel (x, y)
- $\theta(x, y)$ as the gradient orientation at pixel (x, y)
- N as the number of orientation bins
- H_i as the histogram bin corresponding to orientation bin i

The formula to accumulate the gradient magnitudes into orientation bins can be expressed as:

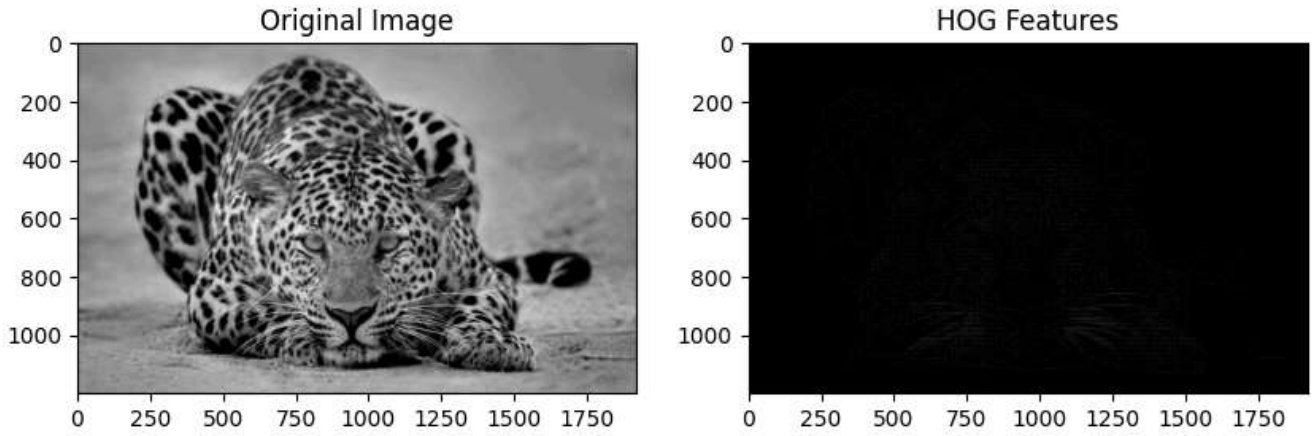
$$H_i = \sum_{(x,y) \in \text{cell}} G(x, y) \quad \text{if} \quad \theta(x, y) \in \text{bin}_i$$

where $(x, y) \in \text{cell}$ represents the pixels within the cell, and bin_i represents the orientation bin i.

After computing the histograms for all cells, the final feature vector H is obtained by concatenating the histograms of all cells.

This formula captures the distribution of gradient orientations within each local region of the image, allowing for the extraction of discriminative features for various computer vision tasks.

```
def extract_hog_features(image_path):
    # Load the image
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Check if image is loaded successfully
    if image is None:
        print("Error: Image not found.")
        return

    # Calculate HOG features
    hog_features, hog_image = hog(image, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2), visuali

    # Display the original image and HOG image
    plt.figure(figsize=(10, 6))
    plt.subplot(1, 2, 1)
    plt.imshow(image, cmap='gray')
    plt.title('Original Image')

    plt.subplot(1, 2, 2)
    plt.imshow(hog_image, cmap='gray')
    plt.title('HOG Features')

    plt.show()

# Specify the path to your image
image_path = '/content/drive/MyDrive/Cvip_Lab/images/cheetah-with-dazzling-eyes-4i7oz6tugoqokj1a.jpg'

# Extract HOG features
extract_hog_features(image_path)
```



Extract SIFT: Scale Invariant Feature Transform feature

SIFT (Scale Invariant Feature Transform) is a feature extraction algorithm used in computer vision to detect and describe local features in images. These features are invariant to image scale, rotation, and illumination changes, making SIFT a robust tool for various computer vision tasks such as object recognition, image stitching, and 3D reconstruction.

Description: SIFT works by identifying key points in an image, called keypoints, and describing their local appearance using histograms of gradient orientations. These keypoints are chosen based on their stability under different transformations and their distinctiveness compared to neighboring keypoints. The keypoint descriptors are then used to match keypoints between different images or to recognize objects.

Formula: The key formula used in SIFT is the creation of a histogram of gradient orientations for each keypoint. This histogram captures the distribution of gradient orientations in a local neighborhood around the keypoint. The formula for computing the histogram is given by:

$$H(x, y, \theta) = \sum_{p \in N(x,y)} w(p) \cdot \text{Bin}(I_p, \theta)$$

where:

- $H(x, y, \theta)$ is the histogram at location (x, y) with orientation θ,
- $N(x, y)$ is the local neighborhood around (x, y),
- $w(p)$ is a weight function for pixel p,
- I_p is the intensity of pixel p,
- $\text{Bin}(I_p, \theta)$ is the bin in the histogram corresponding to the gradient orientation of pixel p.

Theory: SIFT operates in several stages, including:

1. **Scale-space extrema detection:** Identify potential keypoints at different scales by searching for local extrema in the difference-of-Gaussian (DoG) pyramid.
2. **Keypoint localization:** Refine the detected keypoints by fitting a 3D quadratic function to the DoG values around each keypoint.
3. **Orientation assignment:** Assign a dominant orientation to each keypoint based on gradient orientations in the local neighborhood.
4. **Descriptor generation:** Compute a descriptor for each keypoint based on gradient histograms in a local window around the keypoint.

By combining these stages, SIFT is able to extract robust and distinctive features from images that can be used for various computer vision tasks.

In conclusion, SIFT is a powerful feature extraction algorithm that provides scale-invariant and robust keypoints for image analysis and understanding.

```
def extract_sift_features(image_path):
    # Load the image
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Check if image is loaded successfully
    if image is None:
        print("Error: Image not found.")
        return

    # Initialize SIFT detector
    sift = cv2.SIFT_create()

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

    # Draw keypoints on the image
    image_with_keypoints = cv2.drawKeypoints(image, keypoints, None)

    # Display the image with keypoints
    cv2_imshow(image_with_keypoints)
```

```
# Specify the path to your image
image_path = '/content/drive/MyDrive/Cvip_Lab/images/cheetah-with-dazzling-eyes-4i7oz6tugoqokj1a.jpg'

# Extract SIFT features
extract_sift_features(image_path)
```

Extract SURF: feature Speeded-Up Robust Feature

Description: SIFT (Scale Invariant Feature Transform) is a feature extraction algorithm widely used in computer vision for detecting and describing local features in images. It is designed to be robust to changes in scale, rotation, and illumination, making it suitable for various applications such as object recognition, image matching, and 3D reconstruction. SIFT identifies keypoints in an image and generates descriptors that capture the local appearance of these keypoints.

Formula:

1. **Scale-space extrema detection:**
    - Compute the Difference of Gaussians (DoG) pyramid at multiple scales.
    - Identify local extrema in the DoG pyramid to find potential keypoints.
2. **Keypoint localization:**

- Refine the location and scale of keypoints by fitting a 3D quadratic function to the DoG values.

3. **Orientation assignment:**
   - Assign an orientation to each keypoint based on gradient orientations in the local neighborhood.
4. **Descriptor generation:**
   - Compute a descriptor for each keypoint based on histograms of gradient orientations in a local region around the keypoint.

Theory:

- SIFT starts by constructing a scale-space representation of the image using Gaussian blurring to create the DoG pyramid.
- Keypoints are detected as local extrema in scale-space, which ensures scale invariance.
- Keypoint localization refines the keypoint positions to sub-pixel accuracy.
- Orientation assignment computes the dominant gradient orientation at each keypoint to achieve rotation invariance.
- Descriptor generation creates a compact representation of the keypoint's local appearance using histograms of gradient orientations.

By combining these steps, SIFT generates robust and distinctive features that can be used for tasks such as image matching and object recognition. Its ability to handle variations in scale and orientation makes it a valuable tool in computer vision research and applications.

```
def extract_orb_features(image_path):
    # Load the image
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Check if image is loaded successfully
    if image is None:
        print("Error: Image not found.")
        return

    # Initialize ORB detector
    orb = cv2.ORB_create()

    # Detect keypoints and compute descriptors
    keypoints, descriptors = orb.detectAndCompute(image, None)

    # Draw keypoints on the image
    image_with_keypoints = cv2.drawKeypoints(image, keypoints, None)

    # Display the image with keypoints
    cv2_imshow(image_with_keypoints)

# Specify the path to your image
image_path = '/content/drive/MyDrive/Cvip_Lab/images/cheetah-with-dazzling-eyes-4i7oz6tugoqokj1a.jpg'

# Extract ORB features
extract_orb_features(image_path)
```