CourseName: Computer Vision Lab **Course Code:** CSP-422

Experiment:1.3

Aim: Write a program to analyze the impact of refining feature detection for image segmentation.

Software Required: Any Python IDE

Description:

Refining feature detection for image segmentation involves enhancing the quality and accuracy of the detected features to improve the subsequent segmentation process. Some techniques commonly used for refining feature detection in the context of image segmentation are:

· Multi-scale Analysis: Image segmentation often requires detecting features at different scales to capture objects of various sizes. Multi-scale analysis involves applying feature detection algorithms (e.g., SIFT, SURF) at multiple scales by using image pyramids or scale-space representations. This helps in capturing features across different levels of detail and scale.

· Non-Maximum Suppression: Feature detection algorithms such as corner detectors or blob detectors may produce multiple responses for a single feature due to noise or image variations. Non-maximum suppression techniques help in selecting the most salient or distinct features by suppressing the less significant responses in their vicinity.

· Adaptive Thresholding: Feature detection algorithms often rely on setting a threshold to determine the presence of features. However, a fixed threshold may not be optimal for all images, especially in cases with varying lighting conditions or contrast. Adaptive thresholding techniques dynamically adjust the threshold based on local image statistics to improve feature detection accuracy.

· Edge Refinement: Edges are important cues for many segmentation tasks. After detecting edges using techniques such as the Canny edge detector or gradient-based methods, refining the edges can enhance feature detection. This can involve edge thinning, edge linking, or edge contour enhancement techniques to improve the accuracy and connectivity of detected edges.

· Feature Filtering and Selection: Feature detection algorithms may produce a large number of features, including false positives or irrelevant features. Filtering and selecting the most informative and relevant features can improve the subsequent segmentation process. This can be done based on feature quality measures (e.g., response strength, repeatability), spatial distribution, or contextual information.

CourseName: Computer Vision Lab **Course Code:** CSP-422

· Feature Fusion: Combining multiple types of features or feature descriptors can enhance the discriminative power of feature detection. Feature fusion techniques involve integrating information from different feature extraction methods or channels (e.g., color, texture, shape) to capture a more comprehensive representation of the image content. This can be achieved through concatenation, weighted averaging, or more advanced fusion strategies.

· Contextual Information: Utilizing contextual information can improve the accuracy of feature detection and segmentation. Context-aware feature detection methods take into account the relationships between neighboring pixels or features to refine the detection results. This can involve incorporating spatial constraints, semantic priors, or contextual cues from surrounding regions.

· Deep Learning-based Approaches: Deep learning models, such as convolutional neural networks (CNNs), can be trained to directly detect and refine features for segmentation tasks. These models learn feature representations from large amounts of data and can capture complex patterns and contextual information. Fine-tuning or incorporating pre-trained CNN models can improve feature detection performance.

Steps:

1. Import libraries (e.g., OpenCV).
2. Load image dataset.
3. Preprocess images if needed.
4. Select target object image.
5. Extract features from target object.
6. Loop through dataset images:
    a. Extract features from each image.
    b. Match features between target object and image.
    c. Calculate matching score.
    d. Store matching score and image.
7. Sort images based on matching scores.
8. Display top-ranked images and scores.

CourseName: Computer Vision Lab **Course Code:** CSP-422

Implementation/Output:

In [2]:
```
from skimage.io import imread, imshow
import matplotlib.pyplot as plt
import cv2
```

In [3]:
```
image = cv2.imread('C:/Users/Yash Gupta/Downloads/ked.jpg')
# Adjust the contrast and brightness for clarity enhancement
alpha = 1.5 # Contrast control (1.0 means no change)
beta = 30 # Brightness control (0 means no change)
enhanced_image = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
# Apply Canny edge detection for feature extraction
org_edges = cv2.Canny(image, 100, 200)
enh_edges=cv2.Canny(enhanced_image, 100, 200)
```

In [27]:
```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
axis[0,0].imshow(image)
axis[0,0].axis('off')
axis[0,0].set_title("Original")

axis[0,1].imshow(enhanced_image)
axis[0,1].axis('off')
axis[0,1].set_title("Enhanced")

axis[1,0].imshow(org_edges)
axis[1,0].axis('off')
axis[1,0].set_title("Edges")


axis[1,1].imshow(enh_edges)
axis[1,1].axis('off')
axis[1,1].set_title("Edges enhanced")
```

CourseName: Computer Vision Lab **Course Code:** CSP-422

```
Out[27]: Text(0.5, 1.0, 'Edges enhanced')
```