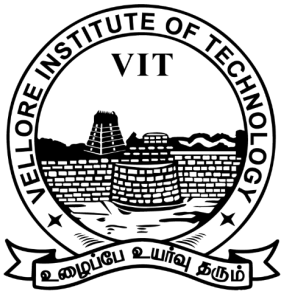


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

Image and Video Analytics

CSE4076

Spatio Temporal segmentation

Date : 08-10-2024

Done by: Dwarakanath.M 21MIA1160

Submitted to: Dr. Saranyaraj D

Objective:

• Extract individual frames and apply image processing techniques.

• Use spatio-temporal segmentation and color thresholding for object identification.

• Detect abrupt and gradual scene cuts.

• Compare pixel values and histograms between consecutive frames.

• Deliver final deliverables: binary segmentation images, marked frames for detected cuts, and a visual summary.

Problem Statement:

• Extracting object motion and scene transitions.

• Accurately segmenting video frames.

• Tracking objects of interest.

• Identifying significant changes between scenes.

• Detecting abrupt and gradual transitions.

• Understanding video's structure and content.

Expected Output:

• Binary segmentation images highlighting segmented objects.

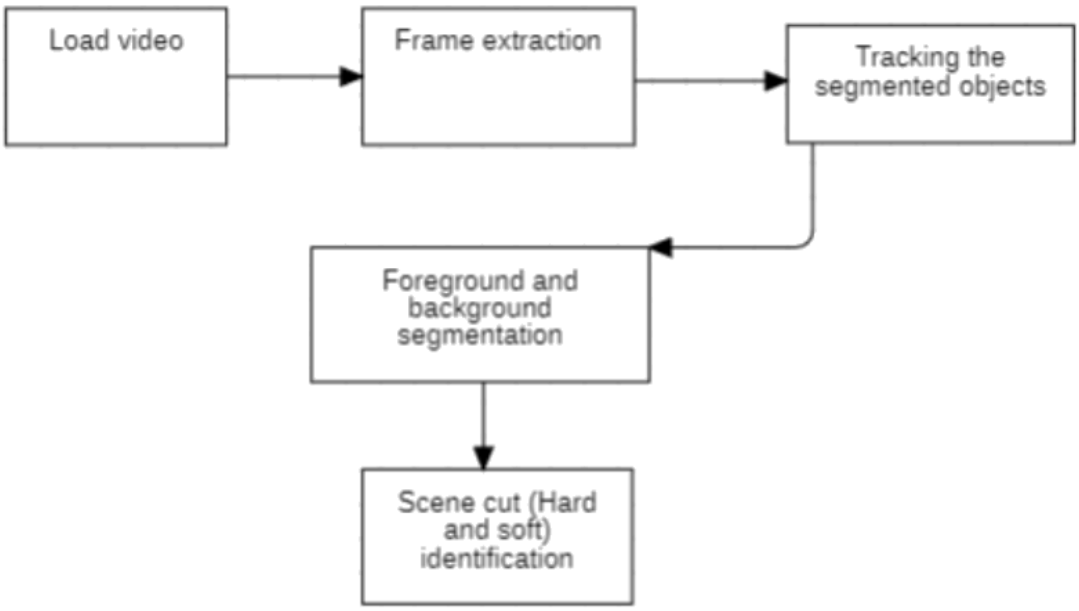
• Marked frames indicating detected scene cuts.

• Visual summary outlining cut boundaries.

• Comprehensive display of selected frames showcasing segmentation results and scene transitions.

• Summarizes video analysis for clear insights into object motion and scene structure.

Methodology:



Python Implementation:

Images Extraction

```python
import cv2
import os

video_path = 'DV1.mp4'
cap = cv2.VideoCapture(video_path)
output_dir = "frames"
os.makedirs(output_dir, exist_ok=True)

frame_num = 0
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    frame_path = os.path.join(output_dir, f"frame_{frame_num:04d}.png")
    cv2.imwrite(frame_path, frame)
    frame_num += 1

cap.release()
print(f"Total frames extracted: {frame_num}")
```

```
Total frames extracted: 1048
```

Segmentation

```python
import numpy as np
import cv2
import os
from google.colab.patches import cv2_imshow

def color_threshold_segmentation(frame, lower_color, upper_color):
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, lower_color, upper_color)
    segmented = cv2.bitwise_and(frame, frame, mask=mask)
    return segmented

frame_path = os.path.join(output_dir, 'frame_0000.png')
frame = cv2.imread(frame_path)

lower_green = np.array([40, 40, 40])
upper_green = np.array([80, 255, 255])

segmented_frame = color_threshold_segmentation(frame, lower_green, upper_green)

segmented_frame_path = os.path.join(output_dir, 'segmented_frame_0000.png')
cv2.imwrite(segmented_frame_path, segmented_frame)

cv2_imshow(segmented_frame)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Scene cuts detection

```python
def detect_scene_cuts(video_path, threshold=30.0):
    cap = cv2.VideoCapture(video_path)
    ret, prev_frame = cap.read()
    prev_frame = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)

    scene_cuts = []
    frame_num = 1

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        diff = cv2.absdiff(prev_frame, gray_frame)

        mean_diff = np.mean(diff)

        if mean_diff > threshold:
            scene_cuts.append(frame_num)

        prev_frame = gray_frame
        frame_num += 1

    cap.release()
    return scene_cuts


scene_cuts = detect_scene_cuts(video_path, threshold=30.0)
print(f"Scene cuts detected at frames: {scene_cuts}")
```

```
Scene cuts detected at frames: [283, 312, 313, 888, 980]
```

Scene Cut frame Display

```python
def display_scene_cut_frames(scene_cuts, num_frames=5):
    scene_cut_frames = scene_cuts[:num_frames]
    fig, axs = plt.subplots(1, len(scene_cut_frames), figsize=(20, 5))

    for i, (frame_num, frame) in enumerate(scene_cut_frames):
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        axs[i].imshow(frame_rgb)
        axs[i].set_title(f"Frame {frame_num}")
        axs[i].axis('off')

    plt.show()

scene_cuts = detect_scene_cuts(video_path, threshold=30.0)
print(f"Scene cuts detected at frames: {[frame_num for frame_num, _ in scene_cuts]}")

display_scene_cut_frames(scene_cuts, num_frames=5)
```

```
Scene cuts detected at frames: [283, 312, 313, 888, 980]
```

Detecting Hard & Soft Cut

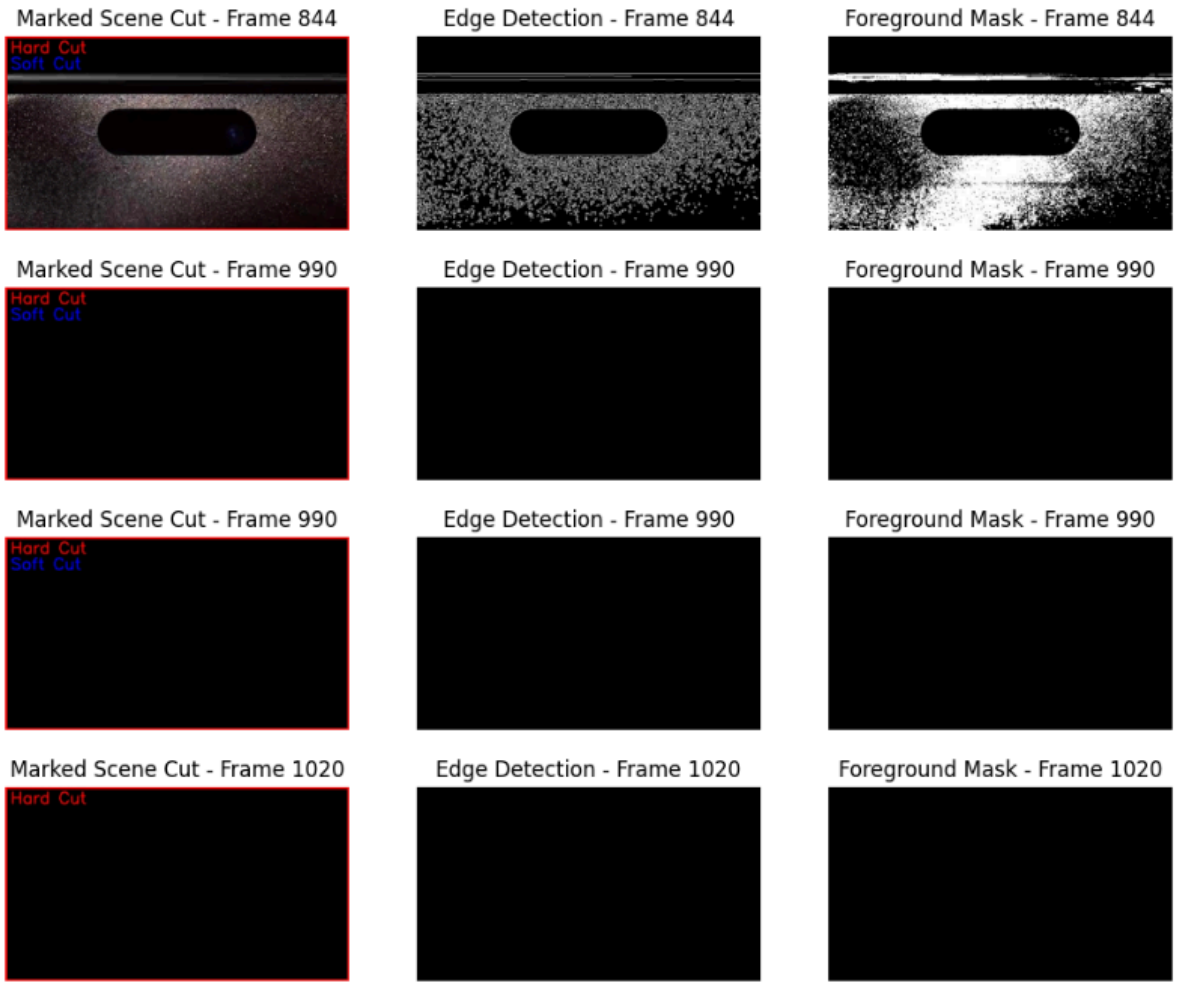
```
hard_cut_threshold = 0.7
soft_cut_threshold = 0.05

hard_cuts = []
soft_cuts = []
scene_boundaries = []

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    hist = cv2.calcHist([gray_frame], [0], None, [256], [0, 256])
    hist = cv2.normalize(hist, hist).flatten()

    if prev_hist is not None:
        hist_diff = cv2.compareHist(prev_hist, hist, cv2.HISTCMP_CORREL)

        if hist_diff < hard_cut_threshold:
            hard_cuts.append(frame_number)
            scene_boundaries.append(frame_number)
            cv2.putText(frame, 'Hard Cut', (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
            cv2.rectangle(frame, (0, 0), (frame.shape[1], frame.shape[0]), (0, 0, 255), 5)

        if soft_cut_threshold < hist_diff < hard_cut_threshold:
            soft_cuts.append(frame_number)
            scene_boundaries.append(frame_number)
            cv2.putText(frame, 'Soft Cut', (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

    marked_filename = os.path.join(marked_output_folder, f'marked_frame_{frame_number:04d}.jpg')
    cv2.imwrite(marked_filename, frame)

    prev_hist = hist
    frame_number += 1

cap.release()
```

Segmentation using Canny edge detection and background subtraction

```
cap = cv2.VideoCapture(video_path)
frame_number = 0
back_sub = cv2.createBackgroundSubtractorMOG2()

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray_frame, 100, 200)

    fg_mask = back_sub.apply(frame)

    edge_filename = os.path.join(segmented_output_folder, f'edge_frame_{frame_number:04d}.jpg')
    fg_filename = os.path.join(segmented_output_folder, f'foreground_frame_{frame_number:04d}.jpg')

    cv2.imwrite(edge_filename, edges)
    cv2.imwrite(fg_filename, fg_mask)

    frame_number += 1

cap.release()

for cut_frame in scene_boundaries:
    marked_frame = cv2.imread(os.path.join(marked_output_folder, f'marked_frame_{cut_frame:04d}.jpg'))

    edge_frame = cv2.imread(os.path.join(segmented_output_folder, f'edge_frame_{cut_frame:04d}.jpg'))
    fg_frame = cv2.imread(os.path.join(segmented_output_folder, f'foreground_frame_{cut_frame:04d}.jpg'))

    plt.figure(figsize=(12, 8))

    plt.subplot(1, 3, 1)
    plt.title(f'Marked Scene Cut - Frame {cut_frame}')
    plt.imshow(cv2.cvtColor(marked_frame, cv2.COLOR_BGR2RGB))
    plt.axis('off')

    plt.subplot(1, 3, 2)
    plt.title(f'Edge Detection - Frame {cut_frame}')
    plt.imshow(cv2.cvtColor(edge_frame, cv2.COLOR_BGR2RGB))
    plt.axis('off')

    plt.subplot(1, 3, 3)
    plt.title(f'Foreground Mask - Frame {cut_frame}')
    plt.imshow(cv2.cvtColor(fg_frame, cv2.COLOR_BGR2RGB))
    plt.axis('off')

    plt.show()
```

Results:

```
Extracted 1048 frames to /content/extracted_frames
Detected Hard Cuts at Frames: [16, 218, 224, 313, 600, 631, 808, 828, 840, 843, 844, 990, 1020]
Detected Soft Cuts at Frames: [16, 218, 224, 313, 600, 631, 828, 840, 843, 844, 990]
Scene Boundaries: [16, 16, 218, 218, 224, 224, 313, 313, 600, 600, 631, 631, 808, 828, 828, 840, 840, 843, 843, 844, 844, 990, 990, 1020]
```



Conclusion:

• Segments and analyzes video content to track objects over time.

• Detects scene cuts.

• Utilizes techniques like frame extraction, preprocessing, segmentation, and optical flow.

• Enhances understanding of dynamic scenes.

• Provides insights into object movement and scene transitions.

• Useful for video analysis and computer vision applications.

Key Takeaway:

• Segmentation Techniques:

Use edge detection and color thresholding to identify objects.

• Object Tracking:

Optical flow methods track object movement across frames.

• Scene Cut Detection:

Uses pixel differences for accuracy in understanding scene changes.

• Visualization:

Highlights segmented objects and scene cuts for better understanding.

Github Link:

https://github.com/dwarakanth/IVA_A4_1160

Drive Link:

https://drive.google.com/file/d/1b7FB43iTrcY8WVbNzvzGaeVIbF1GpSxv/view?usp=drive_link