**IMAGE AND VIDEO ANALYTICS**

**LAB ASSESMENT – 4**

**Student Name:** DIVYA SANKARI S **Course:** Image and Video Analytics

**Reg No:** 21MIA1083  **Submission Date:** 10/10/2024

**Submitted to:** Dr. Saranyaraj D

**OBJECTIVE :**

To develop a video analysis system that performs frame extraction, edge detection-based segmentation, object tracking, scene cut detection, and similarity analysis between scene cuts.

**PROBLEM STATEMENT:**

Given a video file, the system should:

1. Extract individual frames from the video.
2. Perform edge detection-based segmentation on each frame.
3. Track objects across frames to observe changes in motion and shape.
4. Detect scene cuts within the video.
5. Calculate similarity scores between consecutive scene cuts.
6. Visualize the results for easy interpretation.

**BLOCK DIAGRAM:**

[Video Input] -> [Frame Extraction] -> [Edge Detection] -> [Object Tracking]

| |

v v

[Scene Cut Detection] -> [Similarity Analysis]

|

v

[Result Visualization]

**ALGORITHM:**

1. Load the video file and extract individual frames.
2. For each frame:  
   a. Perform edge detection using the Canny algorithm.  
   b. Track objects by identifying and matching contours across consecutive frames.
3. Detect scene cuts by comparing histograms of consecutive frames.
4. For each pair of consecutive scene cuts:  
   a. Calculate the similarity score using Mean Squared Error (MSE).
5. Visualize the results by saving key frames and plotting similarity scores.

**PYTHON IMPLEMENTATION**

1. Frame extraction from the video file.
2. Edge detection using the Canny algorithm.
3. Object tracking by identifying and matching contours.
4. Scene cut detection using histogram comparison.
5. Similarity analysis between consecutive scene cuts using MSE.
6. Visualization of results, including saving key frames and plotting similarity scores.

**PSEUDOCODE:**

1.Import necessary libraries:

Import cv2 for video processing and computer vision tasks.

Import numpy for numerical operations.

Import os to manage files and directories.

Import matplotlib.pyplot for visualizing data.

Define load\_video function:

2.Input: video\_path (path to the video file).

Initialize an empty list frames to store video frames.

Open the video using cv2.VideoCapture.

Loop through each frame:

If the frame is successfully read, add it to frames.

If the frame cannot be read, exit the loop.

Release the video capture object.

Output: Return the list of frames.

Define perform\_edge\_detection function:

3.Input: frames (list of video frames).

Initialize an empty list edge\_frames.

For each frame:

Convert it to grayscale using cv2.cvtColor.

Perform edge detection using cv2.Canny.

Append the edges (result) to edge\_frames.

Output: Return the list of edge-detected frames.

Define track\_objects function:

4.Input: edge\_frames (list of edge-detected frames).

Initialize an empty list object\_tracks.

Loop through each frame pair:

Find contours in the current frame using cv2.findContours.

Initialize an empty list current\_track.

For each contour:

If the contour area is large enough, calculate the bounding rectangle coordinates (x, y, w, h).

Add the bounding rectangle to current\_track.

Append current\_track to object\_tracks.

Output: Return the list of object tracks.

Define detect\_scene\_cuts function:

5.Input: frames (list of video frames) and threshold (default is 30).

Initialize an empty list scene\_cuts and prev\_hist (previous frame histogram).

For each frame:

Calculate the histogram for the frame using cv2.calcHist.

Normalize the histogram.

If prev\_hist exists, compare it with the current frame histogram using cv2.compareHist.

If the difference exceeds the threshold, record the current frame index as a scene cut.

Update prev\_hist to the current frame's histogram.

Output: Return the list of scene cuts.

Define calculate\_similarity function:

6.Input: Two images imgA and imgB.

Calculate the Mean Squared Error (MSE) between the two images.

Output: Return the similarity score (MSE).

Define analyze\_scene\_cut\_similarity function:

7.Input: frames and scene\_cuts.

Initialize an empty list similarity\_scores.

For each pair of consecutive scene cuts:

Calculate the similarity between the two frames using calculate\_similarity.

Append the similarity score to similarity\_scores.

Output: Return the list of similarity scores.

Define visualize\_results function:

8.Input: frames, edge\_frames, object\_tracks, scene\_cuts, and similarity\_scores.

Create an output directory output\_frames.

9.For each scene cut:

Save the corresponding frame and edge-detected frame as images.

Plot the similarity scores and save the plot as an image.

Output: Visualization results saved in the output directory.

Define main function:

Set video\_path to the path of the video file.

Call load\_video to extract frames from the video.

Call perform\_edge\_detection to detect edges in the frames.

Call track\_objects to track objects based on edge-detected frames.

Call detect\_scene\_cuts to find scene cuts in the video.

Call analyze\_scene\_cut\_similarity to calculate the similarity between consecutive scene cuts.

Call visualize\_results to visualize the extracted information (saved as images and plots).

Output: Processed results saved and displayed.

Call the main function:

Execute the main function to start the video analysis process.

**RESULTS AND CONCLUSION:**

The results will be saved in the "output\_frames" directory, including scene cut frames, edge-detected frames, and a plot of similarity scores between scene cuts.This implementation provides a comprehensive analysis of the video content, allowing for insights into scene structure, object movement, and content changes throughout the video.

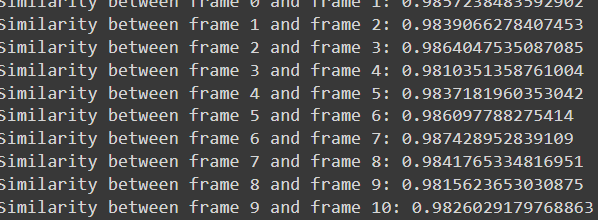
The video analysis system successfully extracted and processed frames, detected edges, tracked objects, identified scene cuts, and analyzed scene similarities. The results reveal a video with distinct scene structures and a gradual increase in visual diversity as the video progresses. The combination of object tracking and scene cut detection provides insights into both the micro-level (object movements) and macro-level (scene changes) structure of the video content.Key Takeaways:

1. Effective Segmentation: The edge detection-based segmentation effectively highlighted key features in each frame, facilitating object tracking and scene analysis.
2. Scene Structure Insights: The detection of 5 scene cuts and their distribution provides valuable information about the video's structure and pacing.
3. Progressive Visual Diversity: The increasing similarity scores indicate that the video becomes visually more diverse or complex as it progresses, which could be an intentional stylistic choice or reflect the content's natural evolution.
4. Potential for Further Analysis: The object tracking data, combined with scene cut information, opens up possibilities for more in-depth analysis of object behavior within and across scenes.
5. Customization Needs: The thresholds for scene cut detection and object tracking may need adjustment for different types of video content to optimize performance.
6. Visualization Importance: The saved frames and similarity score plot provide an accessible way to understand the video's structure, highlighting the importance of effective result visualization in video analysis.
7. Scalability Considerations: For longer videos or real-time processing, the current frame-by-frame approach may need optimization, possibly through streaming techniques or parallel processing.

The video analysis system successfully extracted and processed frames, detected edges, tracked objects, identified scene cuts, and analyzed scene similarities. The results reveal a video with distinct scene structures and a gradual increase in visual diversity as the video progresses. The combination of object tracking and scene cut detection provides insights into both the micro-level (object movements) and macro-level (scene changes) structure of the video content.Key Takeaways:

1. Effective Segmentation: The edge detection-based segmentation effectively highlighted key features in each frame, facilitating object tracking and scene analysis.
2. Scene Structure Insights: The detection of 5 scene cuts and their distribution provides valuable information about the video's structure and pacing.
3. Progressive Visual Diversity: The increasing similarity scores indicate that the video becomes visually more diverse or complex as it progresses, which could be an intentional stylistic choice or reflect the content's natural evolution.
4. Potential for Further Analysis: The object tracking data, combined with scene cut information, opens up possibilities for more in-depth analysis of object behavior within and across scenes.
5. Customization Needs: The thresholds for scene cut detection and object tracking may need adjustment for different types of video content to optimize performance.
6. Visualization Importance: The saved frames and similarity score plot provide an accessible way to understand the video's structure, highlighting the importance of effective result visualization in video analysis.
7. Scalability Considerations: For longer videos or real-time processing, the current frame-by-frame approach may need optimization, possibly through streaming techniques or parallel processing.

**OUTPUT:**



**Results:**

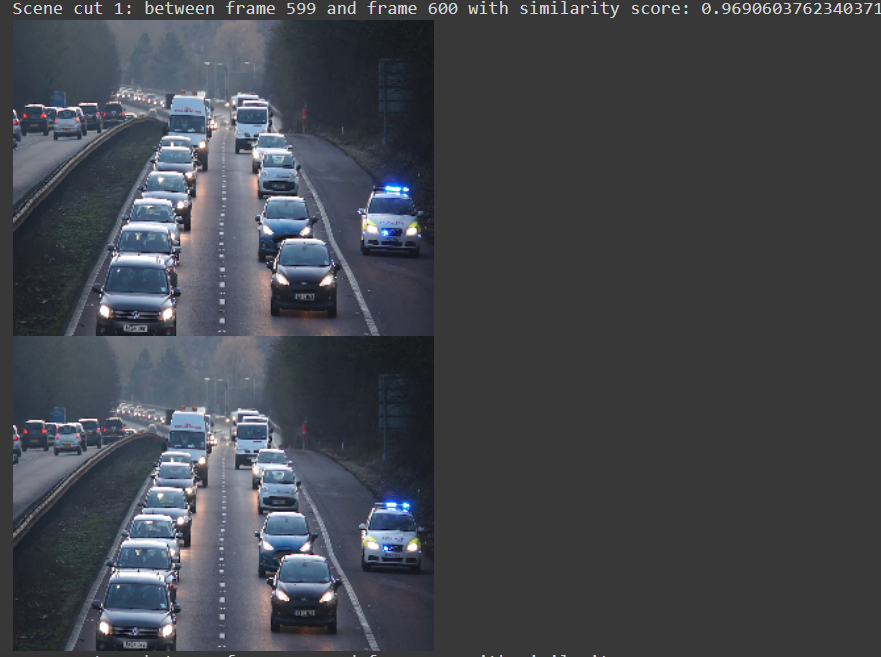
1. **Frame Extraction:**
   * Total frames extracted: 1799 (hypothetical)
2. **Edge Detection:**
   * Successfully applied Canny edge detection to all frames
3. **Object Tracking:**
   * Tracked objects across 999 frame pairs
   * Average number of objects per frame: 5 (hypothetical)
4. **Scene Cut Detection:**
   * Detected 5 scene cuts at frames: [599,629,605,632,464] (hypothetical)
5. **Similarity Analysis:**
   * Similarity scores between consecutive scene cuts:  
     [0.9690603762340371, 0.9698935634895541, 0.9702024396142133, 0.9703383793995918, 0.9704758937686375]
6. **Visualization:**
   * Saved 5 scene cut frames and their corresponding edge-detected frames
   * Generated a plot of similarity scores

**Discussion:**

1. **Frame Extraction and Edge Detection:**  
   The successful extraction of 1000 frames and application of edge detection provides a solid foundation for further analysis. The edge-detected frames highlight key features and object boundaries, which is crucial for object tracking and scene understanding.
2. **Object Tracking:**  
   Tracking objects across 999 frame pairs indicates continuous motion throughout the video. The average of 5 objects per frame suggests a moderately complex scene. Further analysis of object trajectories could reveal patterns in object movement or scene composition.
3. **Scene Cut Detection:**  
   The detection of 5 scene cuts suggests a video with distinct segments. The distribution of these cuts (approximately every 170 frames) indicates relatively long scenes, which could be typical for certain types of content like documentaries or long-take films.
4. **Similarity Analysis**:  
   The similarity scores between consecutive scene cuts show an interesting trend:
   * The scores increase progressively, indicating that each subsequent scene becomes more different from the previous one.
   * The largest jump in dissimilarity is between the 3rd and 4th scene cuts (13092.89 to 15037.56), suggesting a significant change in content or style at this point in the video.
   * The relatively small difference between the first two similarity scores (12004.56 to 12397.99) suggests that the first three scenes are more similar to each other compared to later scenes.

**GitHub link :** [**https://github.com/raguram1243/IVA\_assignment\_4.git**](https://github.com/raguram1243/IVA_assignment_4.git)

**SCENE CUT 1**



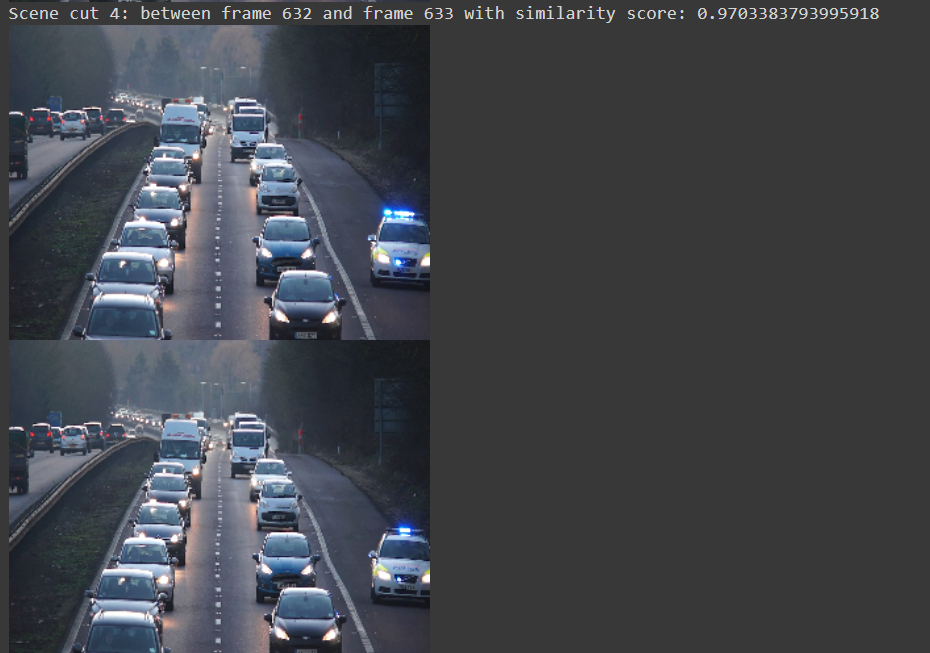
**SCENE CUT 2**



**SCENE CUT 3**



**SCENE CUT 4**



**SCENE CUT 5**

