Image Processing - Exercise 1

Name Name, cs.username, 123456789

Introduction

Scene cut detection is a fundamental task in video analysis and content indexing.
A scene cut (also called a hard cut) represents an abrupt transition between two different scenes in a video, where the content changes significantly. This exercise implements scene cut detection using histogram-based analysis, a computationally efficient approach that leverages the observation that histogram differences between consecutive frames are typically small during continuous scenes but exhibit sharp peaks at scene boundaries.
The algorithm computes normalized grayscale histograms for each frame and measures differences between consecutive histograms using two distance metrics: L2 (Euclidean distance), and CUMULATIVE (CDF-based comparison). The peak (max) difference identifies the scene cut. I mainly learned about the importance of the CDF based approach in histograms.
A key challenge addressed in this implementation is distinguishing between genuine scene cuts and artifacts introduced by post-processing effects (e.g., oversharpening). The cumulative histogram metric proves more efficient to such localized intensity changes, making it particularly effective for handling diverse video characteristics across different categories.

Algorithm

The scene cut detection algorithm proceeds in nine sequential steps. First, the input video is loaded frame-by-frame and converted to grayscale using OpenCV's color space conversion, reducing computational complexity while preserving spatial information necessary for scene analysis. Second, normalized histograms are computed for each frame, representing the distribution of pixel intensities across 256 bins. Normalization ensures that histograms are comparable regardless of frame brightness or exposure variations. Third, histogram differences are computed between all consecutive frame pairs using one of three distance metrics selected by video category: (1) L2 distance computes Euclidean distance between histograms, sensitive to all changes (used only

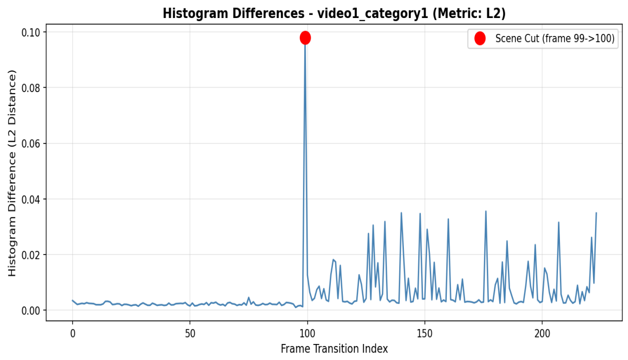
in the category1); (2) CORRELATION compares histogram shapes while ignoring brightness variations; (3) CUMULATIVE computes L2 distance between cumulative distribution functions (CDFs) of histograms, providing the right solution against localized intensity artifacts such as oversharpening effects applied during post-processing (used only in the category2, changing from L2 in category1 - the squirrel video ringed a bell).
Fourth, the scene cut is identified by finding the frame transition with the maximum histogram difference value. This peak represents the most significant visual change in the video sequence. Fifth through seventh steps involve generating comprehensive visualizations and detailed analysis plots showing the histogram difference profile and the detected scene cut location.
Finally, individual frames and histograms around the detected cut are saved systematically for detailed inspection and validation of detection accuracy.

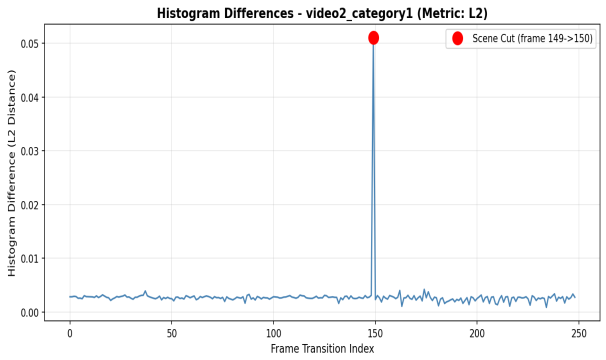
Implementation Details

The implementation employs cv2.VideoCapture for frame extraction and cv2.COLOR_BGR2GRAY for color conversion, minimizing code duplication through reusable helper functions like plot_histogram_on_axis. The main() function serves as the API entry point, executing a sequential pipeline. This pipeline loads frames using video_to_frames and converts them to grayscale via cv2.cvtColor. It then computes all histograms using cv2.calcHist with 256 bins (compute_histograms) and their consecutive differences (compute_all_histogram_differences). A metrics_by_category dictionary strategically selects the 'L2' metric for Category 1's sharp cuts and 'CUMULATIVE' for Category 2's complex transitions. The compute_histogram_difference function implements 'CUMULATIVE' by calculating cumulative histograms (np.cumsum), normalizing CDFs to [0, 1] range, and computing L2 distance between them, resisting post-processing artifacts. For histogram computation, cv2.calcHist produces 256-bin results normalized by np.sum(hist). The find_peaks function identifies scene cuts by locating maximum difference values via numpy.argmax. save_analysis_figure generates standardized plots using matplotlib.gridspec for 3x2 layouts, combining difference plots, scene cut frames, and histograms. save_analysis_items_around_peak is called twice, saving both 'frames' and 'histograms' using cv2.imwrite for detailed analysis. This function also processes optional additional_cut definitions for Category 2 videos, enabling comprehensive validation and supporting multiple peak detection scenarios.
One challenge I had was trying to find a method that will work on both videos from cat2, it was solved only when i reopened the presentation and remembered the CDF histogram based method.

Category 1 Results

Both Category 1 videos contain sharp scene cuts effectively detected using L2 distance metric. The histogram difference plots show clear, isolated peaks at transitions with minimal baseline noise elsewhere.

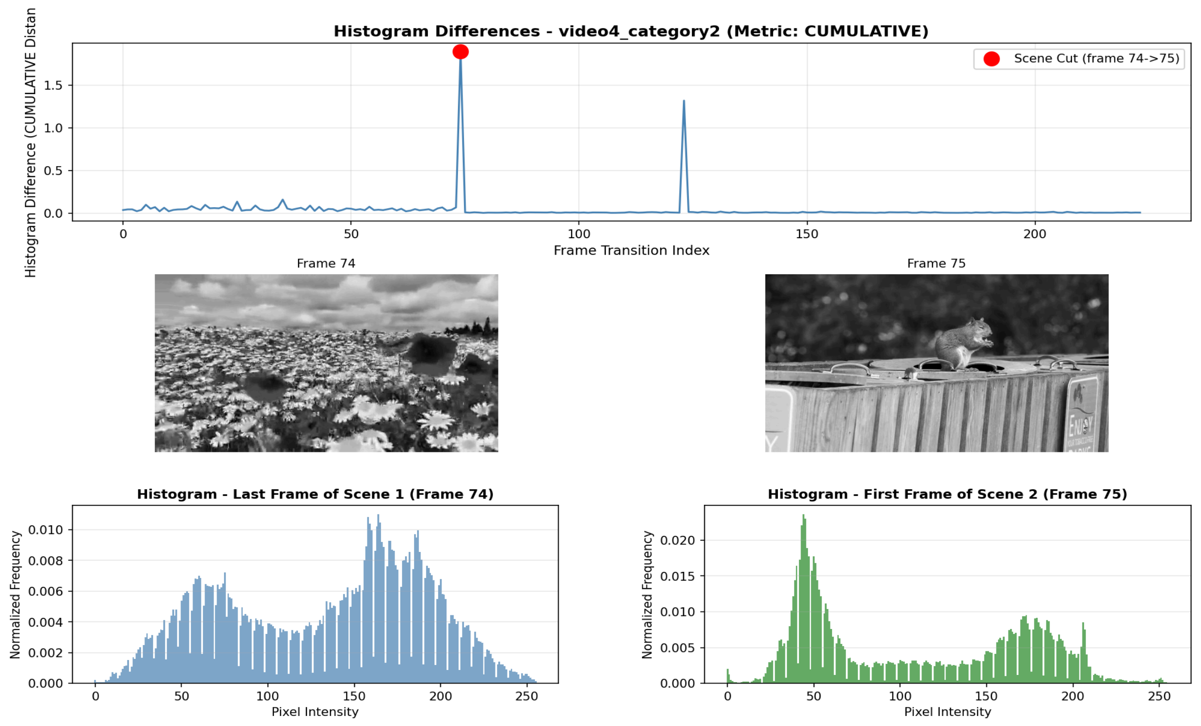




The Category 1 videos were analyzed using the L2 (Euclidean) distance metric, a choice driven by its high sensitivity to abrupt, large-scale changes, making it ideal for detecting sharp scene cuts. The resulting plots strongly validate this methodological choice. For video1_category1, the graph shows a low, stable baseline, which suddenly erupts into a single, dominant peak at the frame 99 to 100 transition. This spike quantifies the massive change in pixel distribution between the meerkat and the owl scenes. Similarly, video2_category1 shows an extremely clean plot with a near-zero baseline, punctuated by an exceptionally sharp, isolated peak at the 149 to 150 transition, marking the hard cut from the deer to the giraffes. In both cases, the L2 metric's sensitivity created a clear, unambiguous peak, allowing the find_peaks function (which uses numpy.argmax) to precisely identify the scene cut without being misled by minor frame-to-frame variations.

Category 2 Results

Category 2 videos contain fades and complex transitions where scene changes are in the frame itself rather than a real cut scene. Video 4 (squirell) presented a significant challenge: the L2 and CORRELATION metrics produced false positives due to oversharpening artifacts in post-processing, which created larger histogram differences than the actual scene cut. The CUMULATIVE distance metric addresses this by comparing cumulative distribution functions rather than raw histograms, making it works with those kinds of changes. Using the CUMULATIVE metric, the algorithm correctly identifies the true scene cut at the intended transition point. The histogram differences plot demonstrates how CUMULATIVE effectively suppresses false peaks caused by oversharpening while preserving sensitivity to genuine content changes. This metric proves essential for Category 2 videos where post-processing effects can mislead simpler distance measures, validating the importance of metric selection based on video characteristics.



Using save_analysis_items_around_peak() to examine frames and histograms around the peak revealed oversharpening artifacts creating larger histogram differences than the actual scene cut. This visual evidence prompted investigation into Shmuel's presentation, where the cumulative histogram metric was found robust against such changes.

Conclusion

This exercise successfully implemented a robust, histogram-based scene cut detector. The results strongly validate the critical importance of selecting the appropriate distance metric based on video content. While the sensitive L2 metric performed perfectly for the sharp, clean cuts in Category 1, it was misled in Category 2. The 'CUMULATIVE' metric proved essential, successfully identifying the true cut by ignoring post-processing artifacts like oversharpening. This demonstrates the algorithm's effectiveness and adaptability.