Image Processing - Exercise 1

Omri Melcer, omri.melcer, 208880211

Image Processing - Exercise 1

Name: Omri Melcer **Username:** omri.melcer **ID:** 208880211

1. Introduction

(a) Goal and Technique

The goal of this exercise is to detect a single scene cut within a video that contains exactly two different scenes. The primary technique used to solve this problem is histogram analysis. Specifically, we compare the **Cumulative Distribution Function (CDF)** of the grayscale intensity histograms of consecutive video frames. The frame pair with the largest difference in their CDFs is identified as the location of the scene cut.

(b) Video Categories

The exercise provides two categories of videos. The most simplest way of defining the categories is via explaining category 2: which is made out of two scenes but in the middle of one of them there is a filter or manipulation on the pixels of the scene, while the first category only has two scenes with such changes in the middle. From looking at the histograms of the two frames before and after the effect it looks like some sort of histogram equalization algorithm.

Our chosen algorithm (detailed below) was designed to be robust enough to handle both cases. By using the L1 norm (Sum of Absolute Differences) on the frames' CDFs, the algorithm effectively captures the *total* change in the global intensity distribution. This approach is sensitive enough to detect the massive, sudden change of a hard cut (Category 1) while also being robust enough to accumulate the significant, and to rule out the subtle changes.

2. Algorithm

(a) Conceptual Steps

The scene cut detection algorithm follows these conceptual steps:

1. **Read Video:** The entire video file is read, and all frames are stored in memory as a list.
2. greyscale convertion.
3. normalized histogram.
4. building cdf for each frame.
5. calculating distance with l1 norm between every to sequential frames.
6. **returning the cut as the two scenes that have the maximum cdf distance.**

(b) Modifications for Categories

No modifications where made between the two scene types.

3. Implementation Details

(a) Algorithm Implementation

The algorithm is implemented in Python with the following key functions:

- **`main(video_path, video_type)`:** The main entry point required by the exercise API. It calls `detect_scene_change()` to get the cut frame index and returns the required tuple `(scene_change_frame, scene_change_frame + 1)`.
- **`detect_scene_change(video_path, video_type)`:** This function orchestrates the entire process. It reads the video using `media.read_video`, then maps the `compute_frame_cdf` function to every frame to get a list of CDFs. Finally, it iterates from `i = 1` to `num_frames - 1`, computes the L1 distance `np.sum(np.abs(cdfs[i] - cdfs[i-1]))`, and uses `np.argmax` to find the index of the maximum distance.
- **`compute_frame_cdf(frame)`:** This helper function performs the per-frame processing. It uses `skimage.color.rgb2gray` for grayscale conversion, `np.histogram` with `bins=256` and `range=(0, 1)` to get the normalized histogram, and `np.cumsum` to compute the CDF.

(b) Libraries Used

- **`mediapy`:** Used for reading video frames (`media.read_video`). This library was chosen as a lightweight and efficient alternative to OpenCV for the simple task of reading frames.

- **`numpy`:** This is the core library for all numerical computations. It was used for array manipulation, `np.histogram`, `np.cumsum`, `np.sum`, `np.abs`, and `np.argmax`. It was chosen for its high performance and robust array-processing capabilities.
- **`skimage.color.rgb2gray`:** Used specifically for its reliable and accurate RGB-to-grayscale conversion.
- **`matplotlib.pyplot`:** Used exclusively in the non-submission functions (`visualize_scene_change` and `test_on_exercise_inputs`) to generate plots of the distance curve and display frames from before and after the detected cut, which was essential for verifying the results.

(c) Hyper-parameters

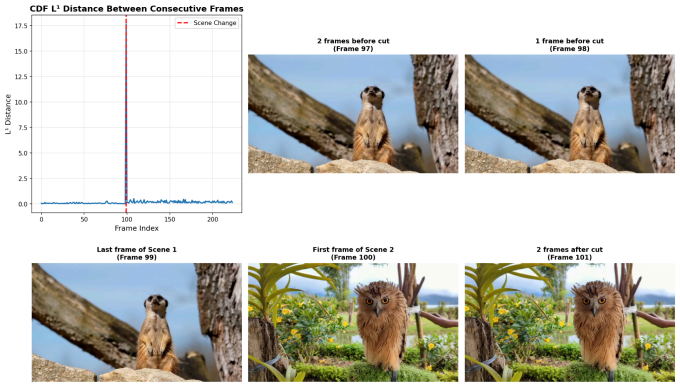
The algorithm is largely adaptive and avoids fixed thresholds. The primary design choices that function like hyper-parameters are:

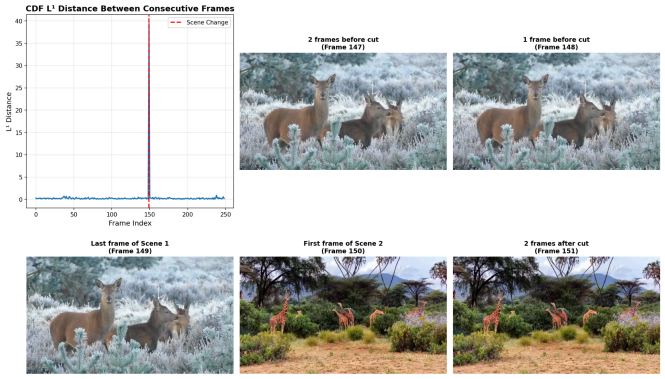
- **Histogram Bins:** Fixed at 256, corresponding to standard 8-bit grayscale intensity levels.
- **Distance Metric:** The L1 norm was chosen. This was a deliberate choice over other metrics (like the L-infinity norm) because it sums *all* changes across the distribution, making it robust to both hard cuts (where all indices change) and dissolves (where many cdf indices change gradually). Throughout the project I handled several types of norms, I choose this norm in the and for the simple reason that it just worked better.

(d) Implementation Challenges

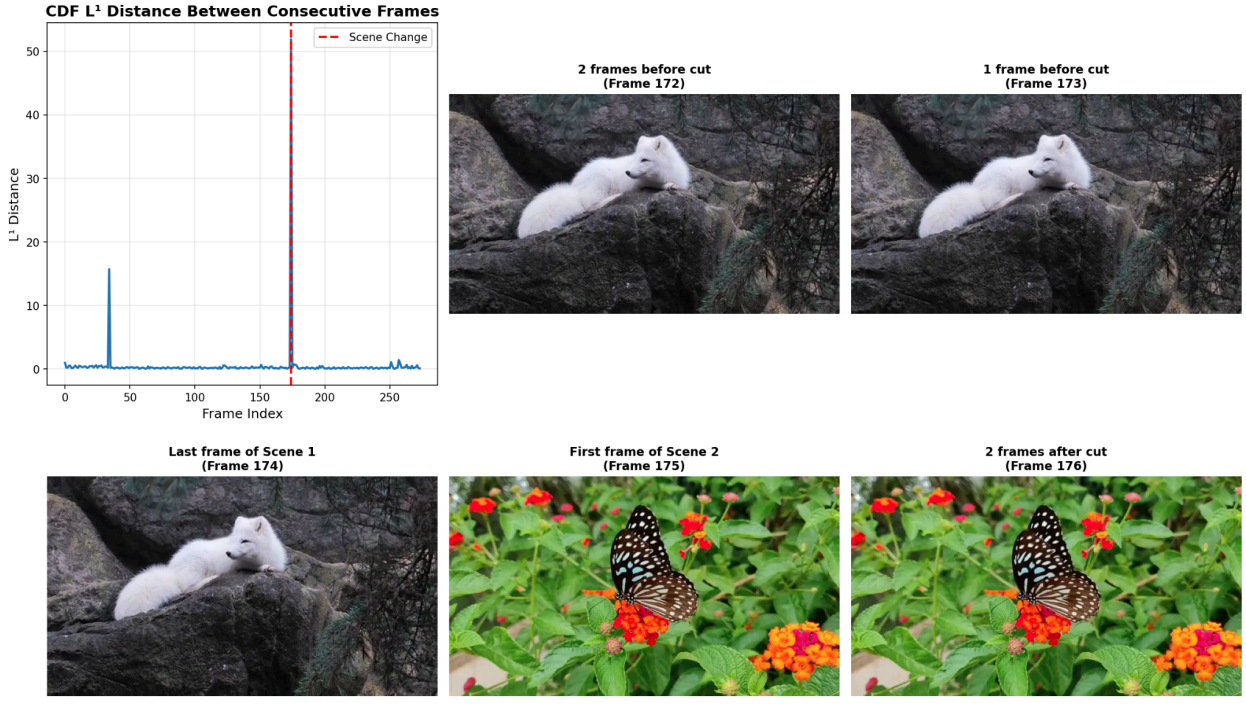
The main challenge was choosing the norm, which will be shown in the results below. Practically chosen after trial and error, though, after analyzing the histograms of the “affect” and understanding that it’s a histogram equalization algorithm - I could understand why L1 would work better.

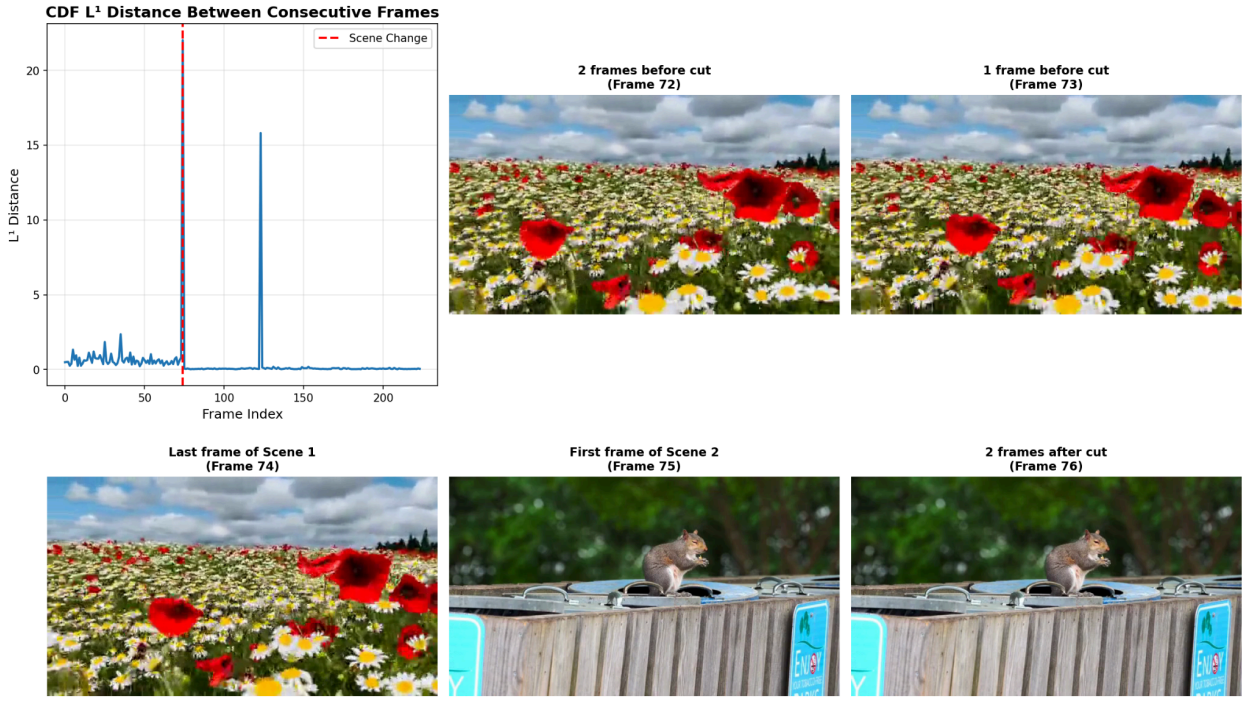
4. Category 1 Result:





5. Category 2 Result:

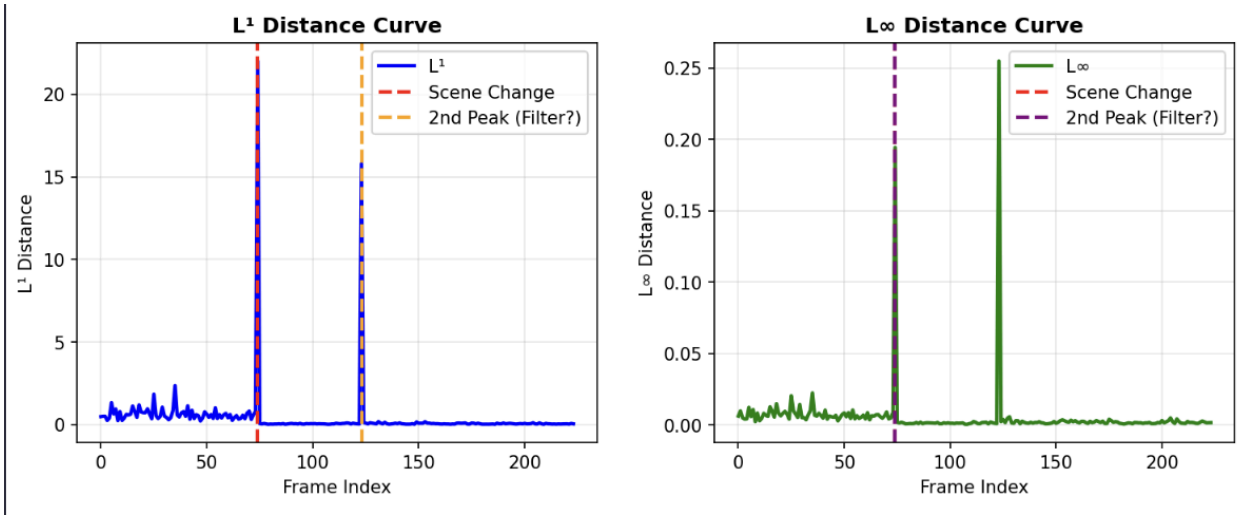




(b) Differences from Category 1

You can easily spot the second spike of the graph as the effect.

(c) & (d) Visualizations and Algorithm Effectiveness



This is the difference between the L inf distance and L1, in the 4th film. The L inf clearly identifies the affect as the scene change

6. Conclusion

This exercise successfully demonstrated a robust method for automatic scene cut detection using classical image processing techniques. The core finding is that comparing the **Cumulative Distribution Function (CDF)** of grayscale histograms using the **L1 distance** is a powerful and flexible approach.

Identifying the effect as Histogram equalization (or similar effect) helps understanding why L inf won’t always work, a very high peak in the original histogram can get to a very low value, the the L inf norm will make as a great cdf difference.