Image Processing - Exercise 1

Nitay Shoshana, nitay, 315315408

Introduction

(a) The goal of this exercise is to detect cuts in a video. The main technique I used was cumulative histogram comparisons between consecutive frames, returning the frame with the highest distance.

(b) The difference between the two categories is that in the first category, the scenes appear raw, unedited, while in the second category, some of the scenes have parts with increased contrast of colors, and also they have camera movements.

Algorithm

(a) Steps in Scene Cut Detection Algorithm

- (i) Initilize max_dist as -1.
- (ii) Read video into memory
- (iii) For each frame
  - (1) Convert frame to grayscale
  - (2) Calculate histogram for current frame
  - (3) Calculate Cumulative Sum of Histogram
  - (4) Compare Histograms - If a previous histogram exists, calculate the euclidean distance between the cumulative histogram of the current frame and the previous frame.
  - (5) If the calculated distance exceeds the max_dist, assign the “cuts” variable to (frame_index-1, frame_index).
  - (6) Update previous histogram - Set the histogram of the current frame as the previous histogram for the next iteration, ensuring each frame is compared only to its immediate predecessor.
- (iv) Return “cuts”

(b) Modifications Between Video Categories

No modification was done between categories.

Implementation Details

(a) Implementation of the Algorithm

1. variables initialization - The max_dist variable to -1.
2. Frame Processing - Frames are loaded with `media.read_video` and converted to grayscale using `PIL`.
3. Histogram Calculation - A 256-bin normalized histogram is generated with `np.histogram`.
4. Cumulative Sum and Comparison - The cumulative sum of histograms is calculated with `np.cumsum`, and the Euclidean distance between frames is found using `scipy.spatial.distance.euclidean`.
5. Cut Detection - If the distance exceeds max_dist, the frame is marked as a cut and added to the return tuple.

(b) Libraries Used

- From scratch - The main algorithm logic, cut detection logic.
- Libraries - `mediapy` (frame loading), `PIL` (grayscale conversion, saving frames to disk for debugging), `numpy` (histogram and cumulative sum), `scipy` (distance calculation). These libraries were chosen because we have learned them in the code bootcamps which made implementation much easier and clean.

(c) Hyper-parameters - None

(d) Challenges

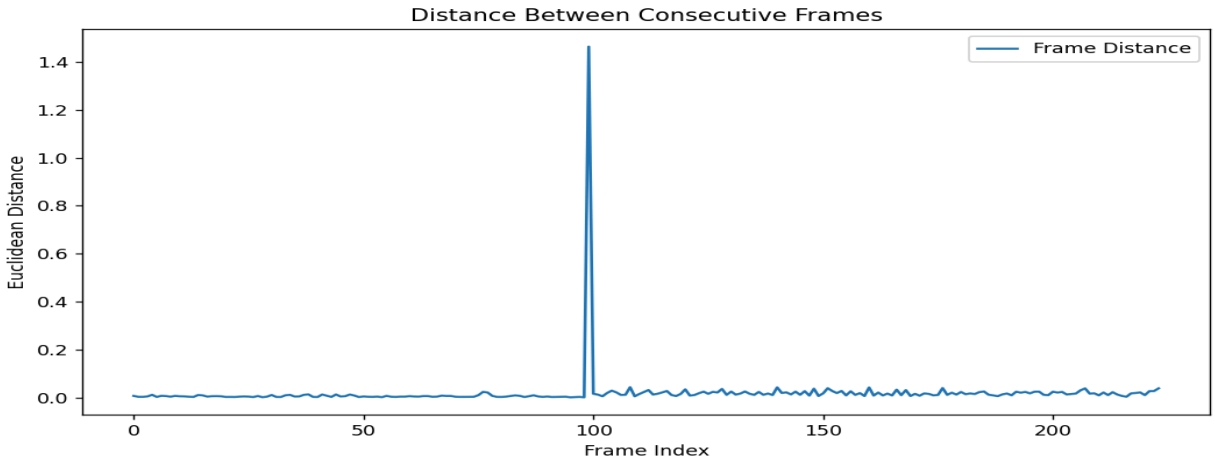
- Threshold Calibration - Adjusted values to avoid false positives in scene cuts. Required a little bit of trial and error.
- Making a generic solution - I know in the forum it was written that we can make an easing assumption - that we only have one cut. But I wanted to come up with a more general solution that will allow me to detect cuts in videos with multiple scene changes. I did it with threshholding over the distance of histograms. I ended up not using it in the final submission but it was fun making it.

Category 1 Results

Video 1

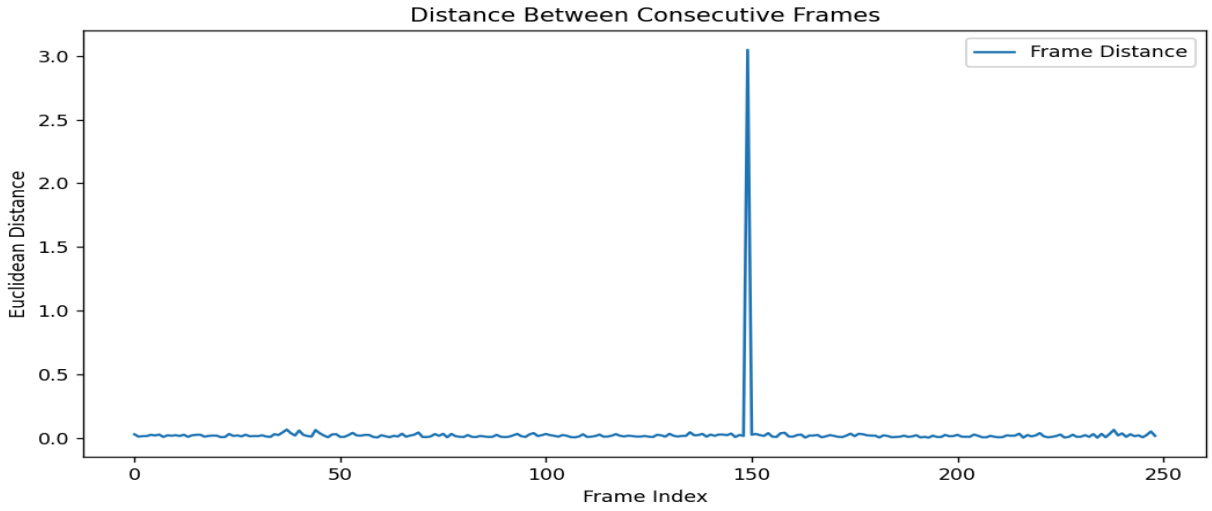
Cuts in frames (99,100)

The distance between cumsums in between the 99th and 100th frames was the greatest of all others.



Video 2

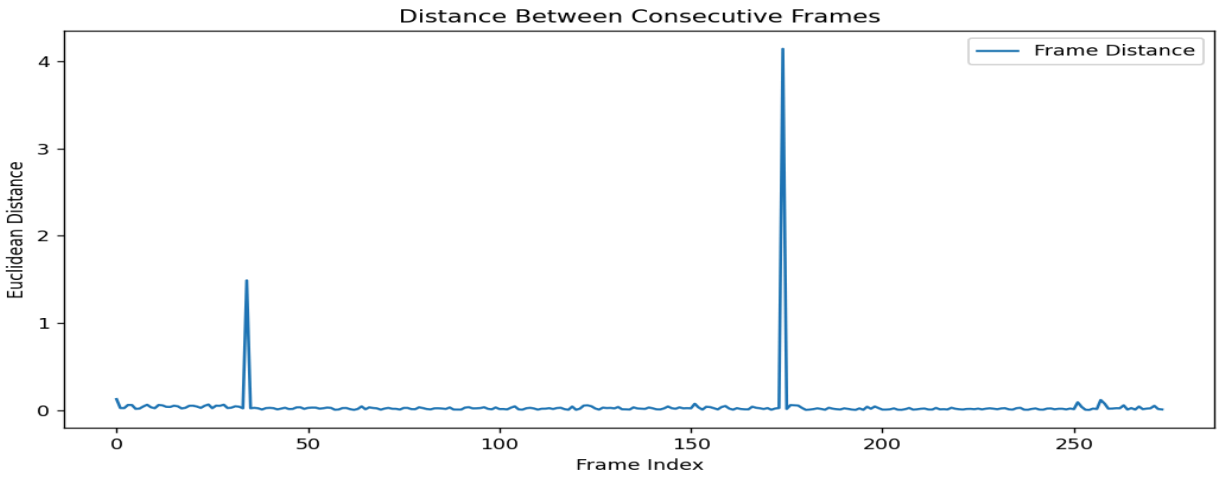
In the same fashion for video 2:
Cuts in frames: (149,150)



Category 2 Results

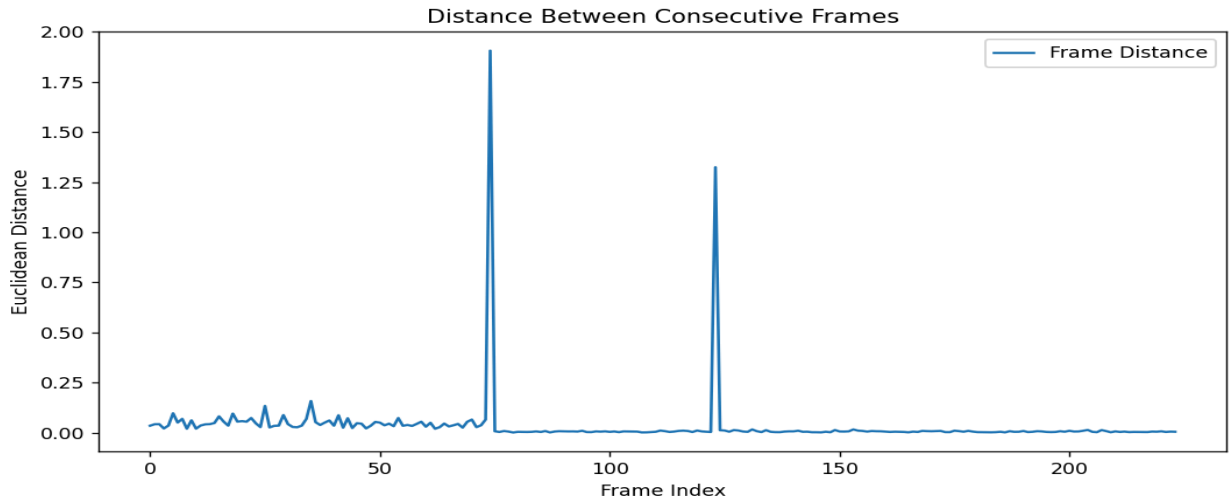
Video 1

Cuts in frames: (174,175)
In the second category, even though the code saw lower spikes in change of cumsum in other areas besides the chosen cut, due to the reduced sensitivity (higher threshold), it was not registered as a cut.



Video 2

In the same fashion:
Cuts in frames: (74,75)



(b) Category 2 in contrast to category 1 has camera movements, and contrast changes. I just watched the video and noticed them.

Conclusion

Our scene cut detection algorithm effectively identifies significant transitions between scenes by comparing cumulative histograms of consecutive frames. The Euclidean distance metric, coupled with an adjustable threshold, allows us to tailor sensitivity for different video types. Testing showed that a lower threshold captures subtle cuts in slower-paced videos, while a higher threshold minimizes false positives in fast-paced content. This approach provides a flexible way to detect scene cuts based on changes in histograms.