Image Processing - Exercise 1

Lenny Agizim, lennehberg, 206661621

Introduction

This exercise was about detecting cuts (scene changes) in videos, using concepts like histogram distancing and cumulative histograms.

The videos to process were split into 2 types:

- Type 1 videos had 2 scenes, the difference between the scenes were significant, but the differences within each scene were somewhat miniscule.
- On the other hand, type 2 videos had differences in brightness and intensity within the scenes themselves, causing false-positives with basic approaches.

Algorithm

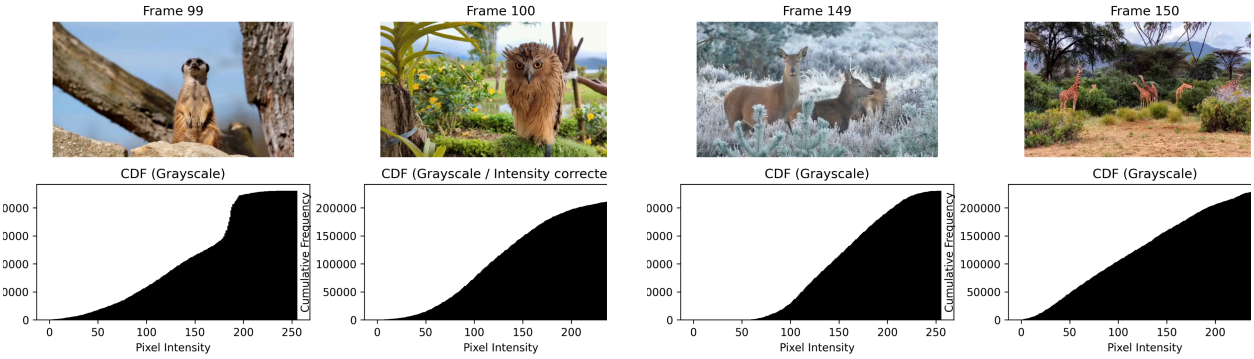
1. Convert the video to grayscale
    a. If the video is of type 2, power transform each frame (decreasing brightness changes within scenes)
2. Loop from second frame to last frame:
    a. If video is type == 1
        i. Calculate the differences between the raw histograms of the frames[i-1] and the frames[i].
    b. If video is type == 2
        i. Calculate the differences between the **cumulative** histograms of frames[i-1] and the frames[i].

    c. Check if the calculate distance is bigger than the max distance recorded
    d. If c, set max cut index to (i - 1, i)
4. Return max cut index.

Implementation Details

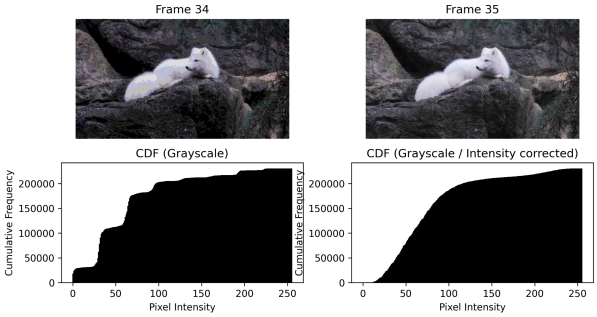
The main function takes a video path and video type, reads the videousing mediapy.read_vide, and calls convert_vid_to_grayscale.
The convert_vid_to_graysclae takes a numpy array representation of the video (vid_arr), iterates over the frames of the video, and converting the frame to an image using PIL.Image.fromarray, and then convert the image to grayscale using .convert("L"). The frame is converted back to an numpy array and is appended to a grayscale_frames list. The function then returns the list of grayscale_frames.
The main function then calls detect_cuts with the grayscale_frames returned from the conversion, and video type.
Detect_cuts initializes the max_differences and max_cut_index variable to 0, and iterates from the second frame to the last. If the type of the video is 1, the hist_diff is calculated by the differences between the raw histograms. Histograms are calculated using numpy.histogram with 256 bins. The difference is then calculated by doing simple vector subtraction using numpy.sum(numpy.abs(hist1 - hist2)).
If the video_type == 2, hist_diff is calculated by the differences between the cumulative histograms. Cumulative histograms are calculated by numpy.cumsum on the histograms that are calculated as described above. If the difference is bigger than the max difference detected so far, it is replaced by the calculated difference, and the cut index is set to (i, i + 1). The function then returns the max cut index.

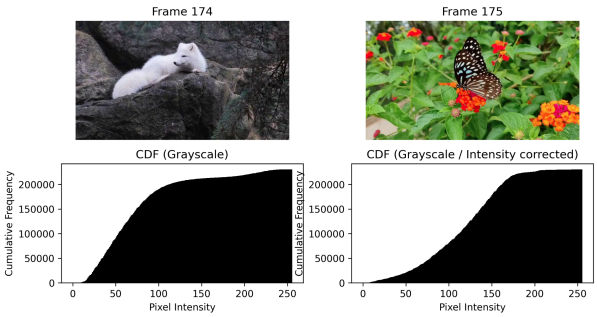
Category 1 Results



Due to the stark differences between the scenes (and the monotonous scenes themselves), a simple raw histogram reduction was quite enough to accurately detect a cut in the videos in category 1.
However, for the most accurate results, distance between frames should be counted between cumulative histograms.

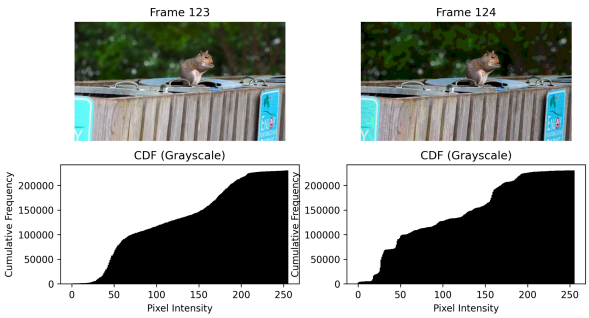
Category 2 Results

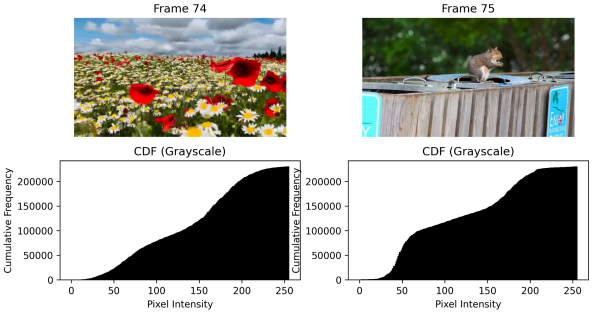




For category 2, a more aggressive approach was necessary due to sharp changes in intensity within the scenes, resulting in false positives when the brightness changes.
To combat this, the cumulative histograms' distance was compared instead of the raw histograms, leading to more accurate results.

(As shown in the figures, the figures on the left portray a raw histogram comparison result, and the figures on the right portray a cumulative histogram comparison)





Conclusion

In Conclusion, detecting cuts in videos can be a very dynamic problem. Whilst these methods are good for simple edge-cases, more robust algorithms can produce better results in complicated situations.
However, for such a simple implementation (using the right libraries) is surprisingly powerful, and can have very good performance.