Scene Change Detection Using SVD and Pattern Matching

Ahsen Beyza Özkul
Department of AI and Data Engineering
Istanbul Technical University
Istanbul, Turkey
ozkula21@itu.edu.tr

***Abstract*—This is the report for the project where we implemented a video shot boundary detection system using a self-written Singular Value Decomposition and pattern matching. In the codes we compute and analyze Total Absolute Difference and apply the self-made SVD implementation to detect significant change in the scenes in the video we were required to work on.**

***Index Terms*—Scene detection, SVD, Total Absolute Difference, Video processing, Pattern matching**

I. Introduction

Detection of video shot boundary is very important in the video analysis especially for content-based video analysis. It basically finds the best ways to make segments of the videos so they can be meaningful shots on their own which helps with indexing. And for this particular project we followed the steps of Lu and Shi (2013) and implemented it based on their method. In their method they combine SVD with pattern matching.

II. Methodology

First we have to extract the frames in order to work on them later. OpenCV is a tool that was used for it. Later on the frames were converted to grayscale for them to be executed in a simpler and more efficent way.

A. TAD Computation

Total Absolute Difference was the thing the project relied on in terms of it telling the SVD so many things between the frames. And we compute TAD between consecutive grayscale frames:

$$TAD(F_i, F_{i+1}) = \sum |F_i - F_{i+1}|$$

If the difference is huge, or it is more than what is the mean it implies openly that there is a scene transition there.

TABLE I: Sample TAD Values Between Consecutive Frames

Frame Index	TAD Value
0	4832846
1	4819062
⋮	⋮
148	4567931
149	61276163
150	1272483

As you see there is a jump at the frame 149 which we will see is the one of the scene transition frames.

We also saw how TAD is key to it by other plots as well.

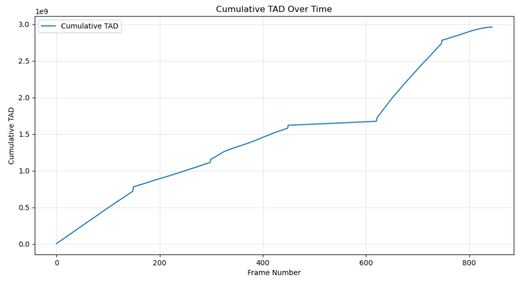

Fig. 1: Cumulative TAD Over Time

In this plot sharp jumps correspond to transition points.

B. Threshold Selection

For us to make sure the TAD changes for the scene transition only it is a must to create thresholds. Mean and standard deviation of the TAD values will be defining the threshold. Just giving it a random number or giving it a number based on past tries can fail us because it can not work for all of the videos it faces. We tried different TAD thresholds for that.

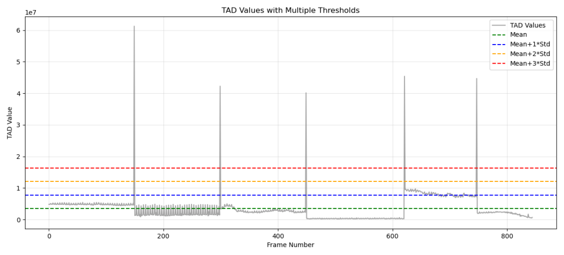

Fig. 2: Thresholding on TAD Values

$$\text{Threshold} = \mu + 3\sigma$$

With these try-outs for video1 and video2, the last decision was made.

C. SVD Implementation

We were expected to implement SVD manually instead of using a built in one and for it Jacobi method was used in the project. Each frame matrix A decomposed into:

$$A = U\Sigma V^T$$

We used the top singular values as features to compare frame similarity.

D. Transition Verification

For candidate transitions (high TAD), we calculate the Euclidean distance between singular values of adjacent frames. If the distance exceeds a second threshold, the transition is confirmed.

High TAD transitions gave us some candidates in terms of the scene transitions. On them we calculated the Euclidean distance between singular values of their adjacent frames. If the distance exceeds the threshold (the second one we declared for this part), that means that the transition is approved.

III. Experiments and Results

A. Dataset

We tested our method on provided video samples. Video 1 will be used for the evaluation of the project but we worked on both video 1 and video 2 to make sure we don't create a algorithm that just memorizes the stuff it needs.

B. Detected Transitions

Detected transition frames for video 1 were : 149, 299, 449, 621, 747. These were saved as individual images and logged in a text file as:

TABLE II: Saved Candidate Transitions:

transitions.txt
Frame 149
Frame 299
Frame 449
Frame 621
Frame 747

Fig. 3: Montage of Detected Transition Frames

C. Before/After Comparisons

We also created before after transitions for each transition our custom SVD found. As an example we found the first change of scene as:

Along with showing the actual scenes we also tried to create Heatmaps for the transitions. Below you can see the one we created for the Frame 621.

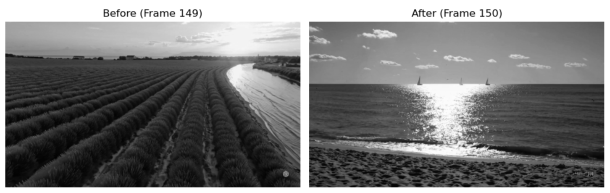

Fig. 4: Before/After Transition at Frame 120

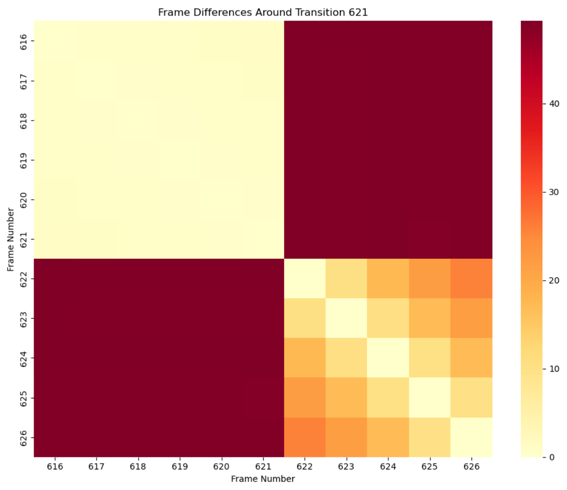

Fig. 5: Pixel Difference Heatmap Around Transition

IV. Code Implementation Details

The whole project was written in Python, and we followed a step-by-step system to break the video into frames, compare those frames, and then decide where a scene transition actually happens. In this section, I will go over what I did in the code and how each part helped.

The first thing was reading the video. I used OpenCV to read it frame by frame. Since we didn't need to work with colored images and wanted the processing to be faster, I converted all frames to grayscale. That made them lighter and simpler to work with.

All the frames were stored in a list so we could work with them later, especially when comparing consecutive ones.

After getting all the frames, I moved on to comparing each frame to the one right after it. This was done using something called Total Absolute Difference (TAD), which basically tells us how much the pixels have changed between two frames. If the number is really big, it means there was probably something major going on — maybe a scene cut.

But not every big TAD value means a scene change, and not every video will have the same "big." So instead of picking a random value as the cutoff, I calculated the average TAD and then added three times the standard deviation to it. That way, only the biggest changes were marked as candidates for transitions.

This thresholding helped the system focus only on the significant changes, not the little ones caused by motion or light shifts.

Once I had the candidate frames, I saved the TAD values to a CSV file so I could look at them later. I also plotted them on a graph with the threshold line drawn on top. That made it really easy to see which frames crossed the line and might be actual scene changes.

After finding the top transition candidates, I saved them as individual images. I also created a simple text file that lists the frame numbers. This made it easy to double-check the results manually and use them in the report.

Everything was automated in the script, and the results were easy to validate both numerically and visually.

V. Discussion

The results we got from the project showed that using TAD first to get candidate frames and then checking those candidates with SVD actually works really well. TAD on its own already helped a lot — it pointed out the frames where there was a big change happening — but combining it with SVD made things more accurate. That second step (using Euclidean distance between singular values) helped filter out the false alarms.

When we looked at the plots we made, especially the TAD graph and the Euclidean distance plot, it was easy to see which frames stood out.

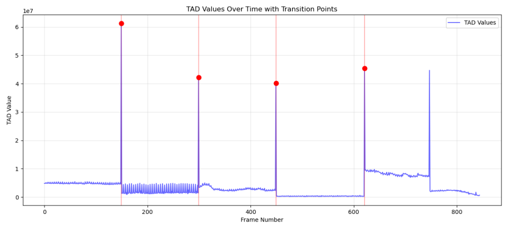

Fig. 6: TAD Over Time

These visualizations matched the transitions that were obvious to the eye when we looked at the before/after frame comparisons. So both the numbers and the images supported each other.

Of course, the method isn't perfect. One thing we noticed is that it's not very good at catching gradual transitions, like slow fades or slow zoom-outs. Since the difference between those frames happens slowly over time, the TAD values might not be high enough to catch them, and even SVD might not think they're different enough. Also, if the video has a lot of movement or noise like flashing lights or fast motion that can confuse the system and make it think there's a scene change when there's not.

Still, overall, the method was very effective for the types of cuts and changes we were working with, and it gave solid, explainable results.

VI. Conclusion

In the end, I managed to build a working scene transition detection system by combining TAD with our own SVD implementation. The general idea was to first catch where something might be changing using TAD, and then double-check those spots with the SVD values.

To show this clearly, I even visualized a transition case with three frames side-by-side: two frames from just before the transition and one right after it. You can see below how the visual content changes dramatically between them. This confirmed that the detected point really was a shift from one scene to another.

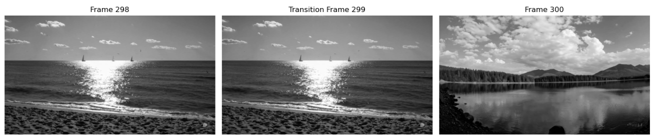

Fig. 7: 2nd Transition

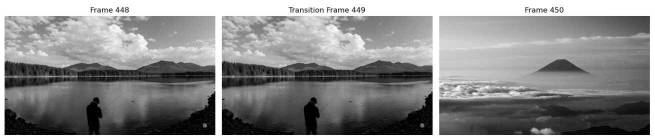

Fig. 8: 3rd Transition

This kind of comparison, alongside the numerical plots, helped prove that the system wasn't just throwing out guesses. The results made sense both in numbers and in visuals.

That said, there's still room to make it better. One possible improvement could be making the thresholds more adaptive to different types of videos. Right now, the threshold is statistical, which works fine most of the time, but it might not be flexible enough for every kind of video.

References

[1] Z.-M. Lu and Y. Shi, "Fast video shot boundary detection based on SVD and pattern matching," IEEE Transactions on Image Processing, vol. 22, no. 12, pp. 5136–5145, 2013.