Department of Computer Science
Technical University of Cluj-Napoca

# Video Shot Boundary Detection

*Image Processing, Laboratory activity, 2022*

Name : Beáta Keresztes
Group: 30432
Email: keresztesbeata00@gmail.com

# Contents

# Chapter 1

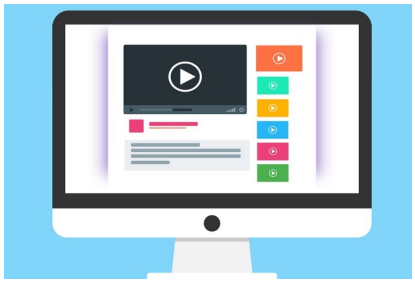
# Abstract

Video shot boundary detection has an important role in domains such as video indexing, video compression, video browsing and fast retrieval of videos from the storage space. In this project, multiple approaches are studied and tested on a sample data, then the results are analyzed and the performance of each algorithm is evaluated.

# Chapter 2

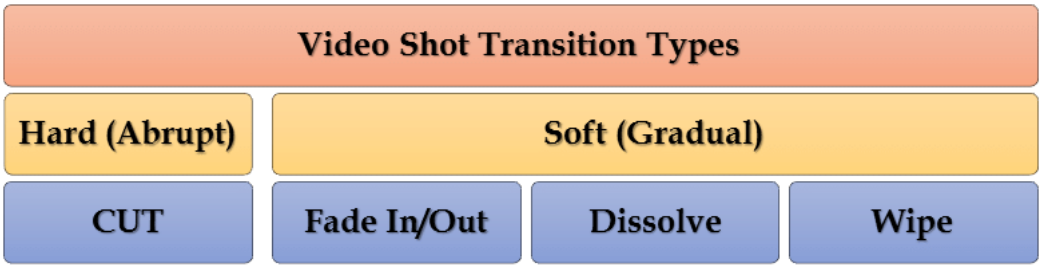
# Introduction

Videos are nowadays the most popular data type, and they currently dominate the internet as the main aid for communication. The human brain is more attracted to the visual form of data and it can interpret it much faster than the written text. Due to the increase in computer performance and media storage availability, the number of uploads and video accesses has also increased. Therefore, the demand for efficient and fast ways to store and access these files has become even higher.



Shot boundary detection or temporal video segmentation, is the main step of the content-based video retrieval, which helps with the efficient management and indexing of video files in an autonomous way. Shot boundary algorithms are based on detecting transitions (shot boundaries) in a video sequence, thus identifying and selecting the key frames from the video. Therefore, their performance can be measured by their ability to correctly detect these transitions.

The video shot transitions can be of classified into multiple categories based on the length of the transition and the pattern of changing the visual content.




The two main categories are:

1. HT (hard/abrupt transition): when two successive shots follow each other immediately without any special effects. Hard transitions represent the cuts which denote the start of a new shot in the video sequence.

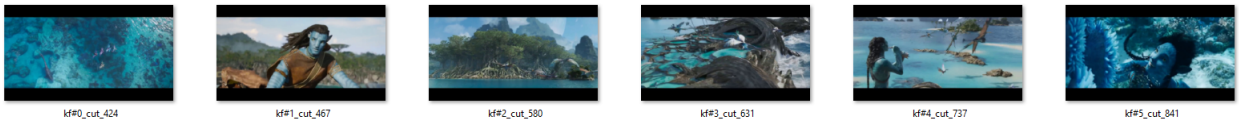



Figure 2.1: Sequence of cuts (hard transitions between shots)

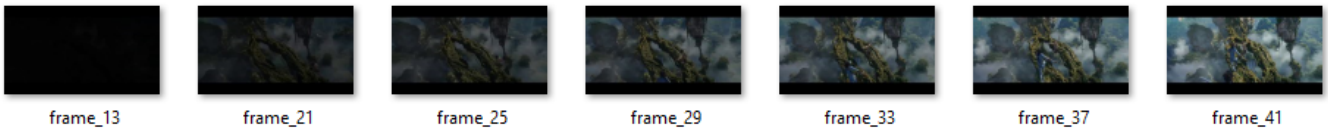



Figure 2.2: Fade in transition

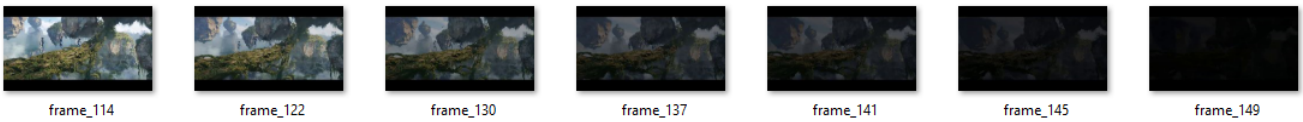



Figure 2.3: Fade out transition

2. ST (soft/gradual transition): when two shots are combined using special effects, and between them there can be multiple frames. Soft transitions can be of several types, such as fade in/out, dissolve or wipe.

# Chapter 3

# Algorithms

Multiple approaches are available so far for shot boundary detection, each with several variations and tuning attempts in order to increase their efficiency and reduce their complexity. These algorithms can be grouped into a few categories based on the approaches they use:

- Pixel-Based Approach (PBA)
- Histogram-Based Approach (HBA)
- Edge-Based Approach (EBA)
- Motion-Based Approach (MBA)

## 3.1 Pixel-Based Approach

The pixel-based approach (PBA) or pixel-wise comparison uses the pixel intensities of video frames to analyze the dissimilarity between consecutive frames.

PBA involves calculating the difference between two corresponding pixels (at location x and y) of two consecutive video frames. Then, the total sum of pixel differences is determined and compared with a threshold. A transition is declared if the sum exceeds the selected threshold.

The main drawback of Pixel based approaches (i.e. intensity pixels), whatever the metric used, is that they are unable to differentiate between a large change in a small area and a smaller change in a large area. Also, pixel-based techniques are the most sensitive to surrounding disturbances (i.e. noises, illumination changes) that may interfere with a given scene.

1. **PBA with multiple thresholds and noise filtering**

   Enhanced PBA to reduce the disturbance in the dissimilarity signal using partial differences and multiple thresholds. Replace each pixel in the original image with the average of their neighbours in a 3x3 kernel filter.

   Main steps of the algorithm:

(a) Use an averaging filter (3x3) to replace the pixels in a frame with the average of its neighbours. This is used to reduce the noise and camera motion effects.

(b) Compare the corresponding pixels of consecutive video frames and get the absolute difference between their intensity levels.

(c) If the partial difference exceeded threshold1, then that pixel is considered a change.

(d) Sum up all the partial differences of the pixels and divide by the number of pixels to obtain the average.

(e) If the result exceeded threshold2, then a cut is detected.

2. **PBA with adaptive thresholds** In this method, the fixed thresholds were replaced with a sliding window to compute the thresholds as the algorithm progresses (adaptive threshold). A mean value of the frame differences is computed based on the previously analyzed pairs of frames in the sliding window and the threshold is updated relative to this mean value. Use 2 sliding windows:

   - Bigger sliding window (M frames): determine the average value of the frame difference within this window
   - Smaller sliding window (N frames): detect shot segmentation point

## 3.2 Histogram-Based Approach

Color histograms or histograms reflect the distribution of colors in an image. Histograms are considered substitutes for PBAs because they do not consider the spatial information of all color spaces. Hence, histograms, to some extent, are regarded as invariant to local motion or small global motion compared with PBAs. For picture-in-picture transitions, change in small region (CSR), the histograms of two consecutive frames are expected to show similarities because of the minimal change in the frames.

1. **Detect Cut transitions with HBA:**

   - **HBA with Bin-to-bin difference**

$$HD(h_i, h_j) = \frac{1}{2N} \sum_{t=0}^{63} \left| h_i[t] - h_j[t] \right|$$

     For this method, colour histograms were computed, with 64 bins for each colour channel (R, G, B).

     Main steps of the algorithm:

     (a) Sum up the absolute differences between corresponding bins of consecutive frames and compute the average difference value.

     (b) Compare the result against a threshold value, and if it is above the threshold, a shot transition is detected.

   - **HBA with Histogram intersection**

$$HD(h_i, h_j) = 1 - \frac{1}{N} \sum_{t=0}^{63} min(h_i[t], h_j[t])$$

- Compare the frequency value of a gray level in the histograms of 2 consecutive frames and select each time the minimum.
- Calculate the average of these minimum values and subtract it from 1.

- **HBA with Quick Shot Search**

  This shot search algorithm compares the first and the last representative frame of a specified part of the video. If they look the same, the algorithm skips searching inside that part as there is a low chance of finding a shot change, this way reducing the computation time. Because the constant change of frame details, in case of non-consecutive frames, the comparison should be based on the public features such as background and color histogram.

  **Global Dissimilarity Function (GDF)** It is used to compare frames which are far from each other. The GDF returns the distance between two color histograms of representative frames which is computed based on color histogram euclidean distance.

$$d^2(h_i, h_j) = \sum_R \sum_G \sum_B (h_i(r,g,b) - h_j(r,g,b))^2$$

  **Local Disimilarity Function (LDF)** LDF looks for a shot change in a local region and it returns frame index if finds it and zero otherwise. The Histogram intersection metric is used for the LDF.

  **Representative frame** In order to reduce noise effect and achieve a better estimation, this algorithm uses representative frame from a set of successive frames. This frame is obtained by applying a time domain median filter on a set of successive frames. The Representative frame is used a background extraction filter: it sorts pixels with the same location in all frames, selects a pixel with the median value from the list and place it in the exact position of the representative frame.

  **Shot detector algorithm** ShotDetector function does a local search in the frame set by calling LDF function. If LDF function finds any shot change it will return its location. Afterwards, ShotDetector function will be repeated to compute left representative frame in remained frames in the left part of shot change location and right representative frame in remained frames in the right part of shot change location.

2. **Detect Fade in/out transitions with HBA:**

   In case of fade-out transitions, the image intensity level gradually decreases, while in case of fade-in transition, the image intensity level slowly increases, so does the standard deviation. The difference between the pixel intensity values can be measured using histograms.
   Accumulating histogram difference (AHD) between consecutive frames n and (n+1):

$$AHD_n(x) = \sum_{t=0}^{255} H_{n+1}(t) - \sum_{t=0}^{255} H_n(t)$$

   Main steps of the algorithm:

   (a) Identify all the monochromatic frame images (with min standard deviation), using a predefined threshold.

   (b) Start with these monochromatic frames and verify if they can be the start of a fade-in or the end of a fade-out transition.

**Algorithm 1** Quick Shot Search

```
0: procedure QUICK_SHOT_SEARCH(LeftRep, LeftIndex, RightRep, RightIndex, MinPartLen)
0:    PartLen ← LeftIndex − RightIndex
0:    MidIndex ← LeftIndex + PartLen/2
0:    [ShotLoc, LeftMidRep, RightMidRep] ← SHOT_DETECTOR(MidIndex, MinPartLen)
0:    if ShotLoc > 0 then
shots.add(ShotLoc)
0:    end if
0:    if Partlen > MinPartLen then
0:       if GDF(LeftRep, LeftMidRep) < T then
QUICK_SHOT_SEARCH(LeftRep, LeftIndex,RightMidRep, MidIndex,MinPartlen)
0:       end if
0:       if GDF(RightMidRep, RightRep) < T then
QUICK_SHOT_SEARCH(RightMidRep, MidIndex,RightRep, RightIndex,MinPartlen)
0:       end if
0:    end if
0: end procedure=0
```

(c) Count the frames on the left of the monochromatic frame until the condition for fade-out holds, that is the intensity level of the image (illumination) monotonically increases the further we go to the left and so does the accumulating histogram difference between consecutive frames.$g_{min}(t)$ and $g_{max}(t)$ denote the min and max intensity values of the histogram associated to the $t^{th}$ frame.
Condition for fade-out (towards a solid black colour):

$$g_{min}(t) <= f_{min}(t-1)$$

and

$$g_{max}(t) <= g_{max}(t-1)$$

and

$$AHD(t) <= 0$$

(d) In the same way, count the frames on the right of the monochromatic frame until the condition for fade-in holds, that is the intensity level of the image (illumination) monotonically increases the further we go to the right and so does the accumulating histogram difference between consecutive frames. Condition for fade-out (towards a solid black colour):

$$g_{min}(t) <= f_{min}(t+1)$$

and

$$g_{max}(t) <= g_{max}(t+1)$$

and

$$AHD(t) => 0$$

(e) In order to eliminate false positives, the length of the fade-in/out transition is tested against a min value. If it is greater than it is accepted as a fade transition.

## 3.3 Edge-Based Approach

Edge-based approaches consider a low-level feature of a frame. These approaches are implemented to detect HT and ST. In EBAs, transition is declared when the locations of the edges of the current frame show a large difference with the edges of the previous frame that have disappeared. Edge detection (including new and previous edges) and edge change ratio (ECR) are the required metrics for computing edge changes.

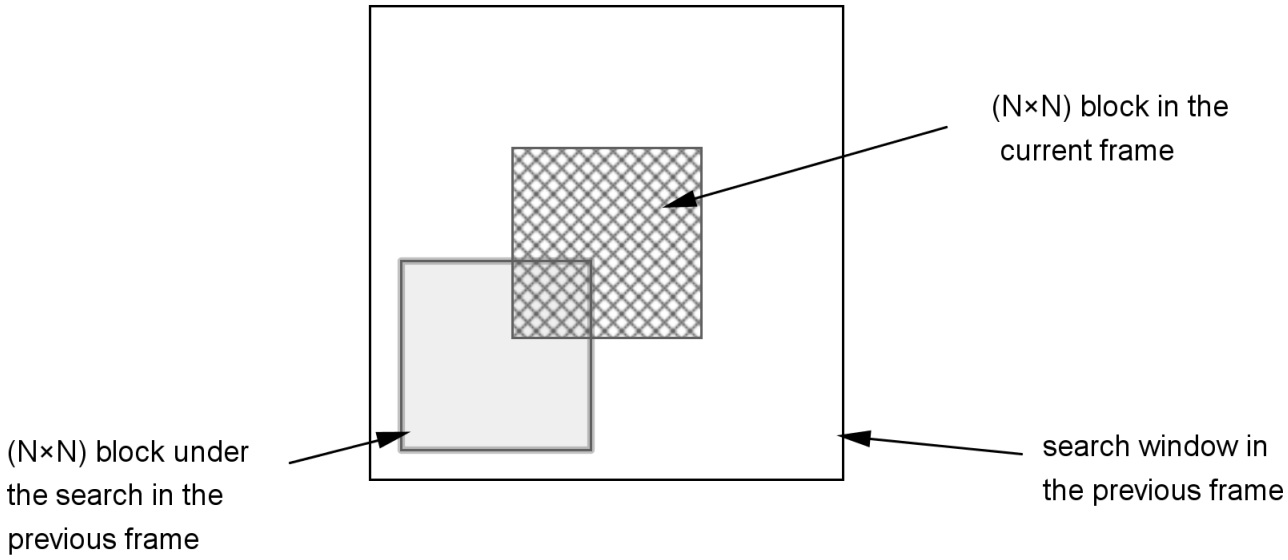
EBAs in terms of performance are less efficient then HBAs, and usually they are more expensive to compute.

- **Detect Cuts:**
  Main steps:
  1. Apply a Gaussian noise filter to smooth the edges, then apply the Canny edge detection algorithm to extract the edges from the current frame.
  2. Based on the previously obtained edge pixels, compute the ratio of the entering and exiting edge pixels: $\rho_{in}$ and $\rho_{out}$.
     $\rho_{in}$ should be a large value during cut and fade-in transitions or at the end of a dissolve transition.
     $\rho_{out}$ should be large during fade out and cut transitions or at the start of a dissolve transition.
  3. Finally, compute the edge change ratio (similarity measure) as: $\rho = max(\rho_{in}, \rho_{out})$
- **Detect fade in/fade out transitions:**
  The edges appear gradually in fade-in transition and disappear gradually in fade-out transition. Main steps:
  1. Apply a Gaussian noise filter to smooth the edges, then apply the Canny edge detection algorithm to extract the edges from the current frame.
  2. Based on the previously obtained edge pixels, compute the ratio of the entering and exiting edge pixels: $\rho_{in}$ and $\rho_{out}$.
     $\rho_{in}$ should increase during fade-in transitions.
     $\rho_{out}$ should increase during fade out transitions.
  3. Identify the monochromatic frames, those which have a uniform colour, thus very small standard variation, and store them in an array.
  4. Consider each monochromatic frame that was found and verify if it represents the first frame of a fade-in transition or the last frame of a fade-out transition.
     In order to eliminate the false positives, identifying a fade transition, where in fact there is only a cut, the length of the fade transition should be tested. If it is greater than a predefined minimum length, then it is accepted.

## 3.4 Motion-Based Approach

A Block Matching Algorithm is a way of locating matching macroblocks in a sequence of digital video frames for the purposes of motion estimation.

A block matching algorithm involves dividing the current frame of a video into macroblocks and comparing each of the macroblocks with a corresponding block and its adjacent neighbors in a nearby frame of the video, thus detecting the movement of a macroblock from one location to another.
BMA can be also used to detect gradual transitions.




From the BMA algorithms, the one selected is the Three Step Search (TSS), which is also the earliest fast block matching algorithm.
A metric for matching a macroblock with another block is based on a cost function. The metric used in this algorithm is the MSE (Mean Squared Error). TSS algorithm steps:

1. Start with search location at center: $\mathrm{S}_x, S_ySetstepsizeS = 4$.
2. Search 8 neighbour locations +/- S pixels around location (0,0) and the center location (0,0), and pick among the 9 locations searched, the one with minimum cost function (smallest MSE).
3. Set the new search origin to the above picked location and set the new step size as S = S/2.
4. Repeat the search procedure until S = 1.
5. The resulting location for S=1 is the one with minimum cost function and the macro block at this location is the best match.

One advantage of this algorithm is that instead of a simple exhaustive search in which case it searches every possible location in a given search window, it only looks at the closest neighbours and decides the next step based on the cost function. Therefore it evaluates cost for a smaller number of macro-blocks than the exhaustive search, and it is better suited for large search windows. Shot detection with MBA steps:

1. Compare the block in current video frame to every block in the successive video frames (inside the window w).

2. A shot transition is detected if the block difference exceeds the given threshold value T.

# Chapter 4

# Performance evaluation (Experiments)

In order to measure the performance of the shot detection algorithms, the following metrics were used:

- **Accuracy**: is the number of positive predictions relative to all the predictions made, and it reflects the ability of the model to detect the majority of the positive cases, shot transitions.
$$Accuracy = \frac{TP + FP}{TP + FP + TN + FN}$$
- **Precision**: the number of correctly detected transitions (reported as true positives) relative to the number of detected transitions (reported as true positives and false positives)
$$Precision = \frac{TP}{TP + FP}$$
- **Recall**: the number of correctly detected transitions (reported as true positives) relative to the number of transitions in the ground truth (reported as true positives and true negatives).
$$Recall = \frac{TP}{TP + FN}$$
- **F1-score**: combines precision and recall to achieve one score.
$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

where

TP = True positives (correctly detected shots)
FP = False positives (falsely detections)
TP = True negatives (correctly predicted that a frame is not a shot)
FN = False negatives (missed shots)

Algorithms:

1. PBA-v1 = PBA with multiple thresholds and noise filtering

| Algorithm | Params | Transition type | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|---|---|
| PBA - v1[1] | T1 = 25, T2 = 0.6 | cut | 0.98 | 0.25 | 0.5 | 0.33 |
| | T1 = 25, T2 = 0.4 | cut | 0.96 | 1 | 0.29 | 0.45 |
| | T1 = 15, T2 = 0.5 | cut | 0.78 | 1 | 0.06 | 0.12 |
| PBA - v2[2] | M = 25, N = 20 | cut | 0.866 | 0.916 | 0.089 | 0.162 |
| | M = 20, N = 10 | cut | 0.917 | 0.75 | 0.118 | 0.204 |
| | M = 15, N = 15 | cut | 0.898 | 0.666 | 0.088 | 0.156 |
| | | gradual | 0.757 | 0.022 | 0.0168 | 0.0191 |
| HBA - v1[3] | T = 0.15, Bins = 256 | cut | 0.816 | 0.583 | 0.044 | 0.082 |
| | T = 0.15, Bins = 64 | cut | 0.869 | 0.5 | 0.054 | 0.098 |
| | T = 0.25, Bins = 256 | cut | 0.895 | 0.583 | 0.077 | 0.137 |
| | T = 0.25, Bins = 64 | cut | 0.933 | 0.5 | 0.107 | 0.176 |
| HBA - v2[4] | T = 0.15, Bins = 64 | cut | 0.923 | 0.5 | 0.092 | 0.155 |
| | T = 0.2, Bins = 64 | cut | 0.923 | 0.5 | 0.054 | 0.09 |
| | T = 0.25, Bins = 64 | cut | 0.933 | 0.5 | 0.107 | 0.098 |
| HBA - v3[5] | T = 40000 | cut | 0.907 | 0.166 | 0.02 | 0.04 |
| | T = 37500 | cut | 0.905 | 0.166 | 0.027 | 0.047 |
| | T = 35000 | cut | 0.902 | 0.166 | 0.027 | 0.046 |
| HBA - v4[6] | $\sigma = 20$, length = 15 | fade_in | 0.839 | 0.819 | 0.324 | 0.464 |
| | | fade_out | 0.946 | 0.493 | 0.886 | 0.634 |
| | $\sigma = 15$, length = 10 | fade_in | 0.839 | 0.819 | 0.324 | 0.464 |
| | | fade_out | 0.945 | 0.468 | 0.902 | 0.616 |
| | $\sigma = 5$, length = 10 | fade_in | 0.839 | 0.819 | 0.324 | 0.464 |
| | | fade_out | 0.951 | 0.531 | 0.913 | 0.672 |
| EBA - v1[7] | T = 0.3, M = 20, N = 15 | cut | 0.976 | 0.083 | 0.1 | 0.09 |
| | T = 0.25, M = 20, N = 15 | cut | 0.975 | 0.083 | 0.1 | 0.09 |
| | T = 0.2, M = 25, N = 20 | cut | 0.971 | 0.166 | 0.125 | 0.142 |
| EBA - v2[8] | $\sigma = 30$, length = 5 | fade_in | 0.918 | 0.305 | 0.536 | 0.389 |
| | | fade_out | 0.875 | 0.063 | 0.138 | 0.086 |
| | $\sigma = 35$, length = 5 | fade_in | 0.908 | 0.305 | 0.448 | 0.363 |
| | | fade_out | 0.863 | 0.063 | 0.108 | 0.08 |
| | $\sigma = 15$, length = 5 | fade_in | 0.913 | 0.041 | 0.428 | 0.075 |
| | | fade_out | 0.875 | 0 | 0 | 0 |
| MBA[9] | T=10,N=10,M=20,B=64 | cut | 0.945 | 0.416 | 0.113 | 0.178 |
| | | gradual | 0.3 | 0.666 | 0.321 | 0.433 |
| | T=0.5,N=10,M=20,B=16 | cut | 0.94 | 0.333 | 0.086 | 0.137 |
| | | gradual | 0.279 | 0.672 | 0.314 | 0.178 |
| | T=0.5,N=10,M=15,B=4 | cut | 0.928 | 0.333 | 0.071 | 0.117 |
| | | gradual | 0.279 | 0.672 | 0.314 | 0.428 |

Table 4.1: Comparison of algorithms for detecting Cuts and Gradual(Fade) transitions

2. PBA-v2 = PBA with adaptive thresholds using two sliding windows of sizes MxM and NxN, able to detect both cuts and gradual transitions
3. HBA-v1 = HBA with bin-to-bin difference
4. HBA-v2 = HBA with histogram intersection
5. HBA-v3 = HBA with quick shot search
6. HBA-v4 = HBA with AHD for detecting fade transitions
7. EBA-v1 = EBA with ECR for detecting cuts
8. EBA-v2 = EBA with ECR for detecting fade transitions
9. MBA = MBA with BMA for detecting both cuts and gradual transitions

**Conclusions**

- From the above results, we can see that the majority of algorithms had a very high precision, over 80%, except the MBA algorithm when detecting gradual transitions, its accuracy came out very low.
- The PBA approach has produced the highest precision score, therefore it can correctly identify the most cuts, due to the fact that it operates on the smallest level, the level of individual pixels, and it uses a very precise dissimilarity metric when comparing frames.
- The EBA approach had the highest recall score, which means it is able to identify all the frame transitions, even with a few false positive predictions.
- The F1-score had the highest values for the HBA approach which uses a modified Quick-Search algorithm to reduce the running time and thus improve the performance of the algorithm. A high F1-score indicates a good combination of optimal values for precision and recall.

### 4.0.1 Dataset

The data used for evaluating the algorithms is a video sequence from a movie trailer Avatar 2, and it was selected because of the various transition types it contains, both cuts and fade transitions.

# Chapter 5

# References


1. Video Shot Detection approaches
2. Shot Boundary Detection: Fundamental Concepts
3. Block Matching algorithms
4. Fades Detection algorithms
5. Quick-Search algorithm for Shot Detection