Practicum report 2  
Video Shot Detection, Annotation, and Retrieval

Group 4:  
Tim HERINCKX 2e Master Elektrotechniek: ICT  
Nikolas TAILLIEU 1e Master Computerwetenschappen: SE  
Jeroen VANKELECOM 2e Master Elektrotechniek: ICT

# Local histogram method for abrupt and gradual transitions

The local histogram method of exercise 5 can also be extended to detect gradual transitions. In this case, the method has to take more frames into account and not only the previous one. By looking at more frames in the past and comparing these frames with the current frame, increasing difference will be noticed when looking at more past frames. The method of exercise 6 (Extended Local Histogram) stores the previous frames in a buffer (with a specified *bufferlength*). The current frame is always compared using the local histogram method to every other frame in the buffer (with a specified *threshold* for the histogram difference). If the current frame does not differ from the frames in the buffer, the current frame is added to the buffer. If the current frame differs from a specified number of frames in the buffer, a shot is detected. A cut occurres when all histogram differences are quite the same. When there is a lot of fluctuation on these differences, a gradual transition occurred. After a shot detection, the buffer is flushed and filled again.

# Frame selection technique

## Show shots

When the user wants to play a certain shot, he has to select it from the shot list and press the ‘play shot’ button (figure 4, n°1). This button is only available when the user has derived a shot list from the video with the ‘Detect Shots’ button. The VideoManager, which holds the video as well as the shot list derived from this video, is then called and he in return calls the self-created method ‘PlayShot(int start, int end)’ of the DxPlay object, with the given start and end values.

In the ‘PlayShot’ method, the IMediaSeeking object is derived from the FilterGraph and the start position and end position of the video are set with the given parameters. After PlayShot is called, the play method also ensures through the ‘m\_play\_StopPlayShot’ method that the Reset method will be called when the shot stops playing. This method resets the positions in the IMediaSeeking object to 0 for the start position and the number of the last frame for the end position. If this would not be set, the video would only be able to play until the last frame of the previously played shot from now on.

After all this is done and only when the video is currently stopped or paused, the Play method of the DxPlay object is called and the selected shot starts playing.

## Export shots

When the ‘Export Shots’ button (figure 4, n°2) is called, a dialog is given where the user can select a directory for the images. Each shot has its own image, which he keeps with him in his own Shot class. This Bitmap is a deep copy of the frame taken while extracting the shots from the video sequence. Each of the Bitmaps is then stored as a PNG image in the selected directory.

# Parameter settings, obtained precision and recall values

* **Pixel Difference**

The range of parameter varies between 0 and 765 (3 times a difference of 255). This parameter decides how much the RGB-values of a pixel may differ with the RGB-values of the corresponding pixel in the previous frame. Parameter lies between 0 and 1 and denotes the allowed percentage of pixel differences between the current and previous frame. To evaluate the performance of this method, we performed a sweep on these parameters to define the best settings. We expect that the method performs well for around 50 (two pixels differ when the difference in RGB-values is larger than 50) and around 0.20 (if more than 20% of the pixels differ, a shot is detected).

A higher allows more difference between corresponding pixels before they are seen as different. This will lead to less detected shots. Also a higher will lead to less shots, because more difference between consecutive frames is allowed. This will both lead to a decrease of recall and an increase of precision.

* **Exhaustive motion estimation**

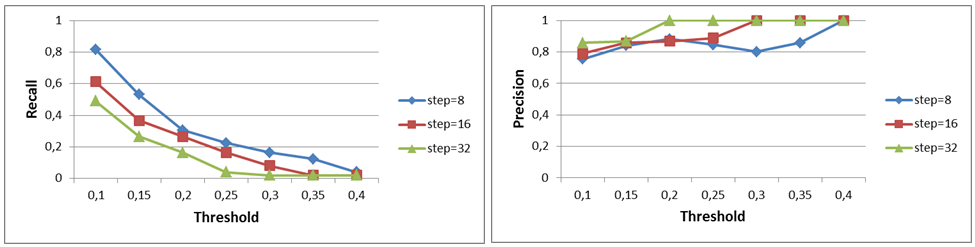
The threshold ranges from 0 to 1 and decides if the relative difference between the current and previous frame is large enough to detect a shot. For this method, the frame under consideration is divided in blocks with a specified size (2x2, 4x4, 8x8, 16x16). The third parameter, window size (in terms of pixels), determines the search window in which a perfect block match can be found.

We expect optimal results for a block size of 8x8 (large enough) and a window size of 8 pixels (not too large to reduce the calculation time). Again, a higher threshold will lead to less shots detected, a decrease in recall and an increase in precision.

* **Optimized motion estimation**

In the optimized motion estimation method, the window size parameter is replaced by step size (in terms of pixels). The function of the parameter is the same: defining a search window. But every iteration, the search window is moved to the best match.

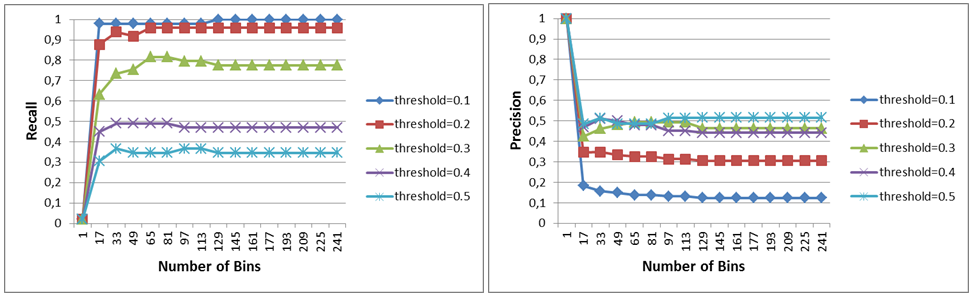
Again, we expect good results for a large block size and an acceptable step size. Figure 3 shows a sweep on the threshold between 0.1 and 0.4, for different values of step size. A larger step size will leads to a higher precision, but also to a lower recall (less shots are detected). Again, a higher threshold results in less detected shots, a lower recall and a higher precision.



*Figure 3. Recall and precision values in function of threshold for optimized motion estimation.*

* **Global histogram**

For the global histogram method, two parameters can be chosen. The threshold varies between 0 and 1 and indicates the relative difference between the global histograms of the previous frame and the current frame. A bin collects the pixels with the same specified range for the R-, G- or B-value. For this reason, the number of bins varies between 1 and 255.

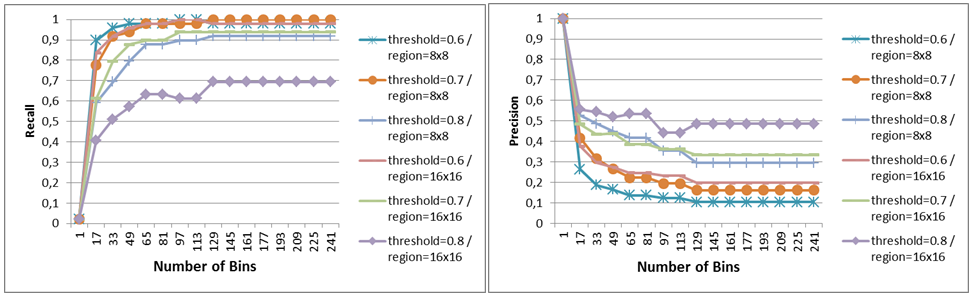


*Figure 2. Recall and precision values in function of bins for the global histogram method.*

We expect an optimal number of bins around 50. However, we performed a sweep over all possible numbers of bins. Figure 2 shows the recall and precision values for different settings of both parameters. To obtain acceptable values for recall and precision, a threshold of 0.3 and around 70 bins will satisfy. The optimal parameters are given in Table 1.

* **Local histogram**

To create local histograms, a third parameter is introduced. This parameter determines the region size (in pixels) for each local histogram. For small region sizes, a histogram will not be useful. For this reason, we expect good results for larger region sizes (8x8 and 16x16). In figure 3 we see again that a larger threshold will lead to a lower recall and a higher precision. A larger region size (16x16) results into a higher precision, but a lower recall.



*Figure 3. Recall and precision values in function of bins for the local histogram method.*

* **Extended local histogram**

The extended local histogram method makes use of two new parameters: bufferlength and variance. The bufferlength determines how many frames we will look back in the past. If there is a lot of fluctuation in the differences between the current frame and the previous frames in the buffer, the variance parameter can decide if a gradual transition, a cut or no transition appeared. Threshold, region size and number of bins have the same range like in local histogram.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Return of the Jedi | | | Youth without youth | |
| parameters | Precision | Recall | Time [ms] | Precision | Recall |
| Local pixeldif. | δ2=40 δ3=0.2 | 98% | 7% | 4570 | 74% | 21% |
| Exh. Motion | treshold=0.1 blocksize=8, windowsize=8 | 84% | 60% | 841490 | 63% | 100% |
| Opt. Motion | threshold=0.1,blocksize=8, stepsize=8 | 82% | 75% | 113296 | 63% | 100% |
| Glob. Hist. | treshold=0.3, bins=65 | 82% | 49% | 6714 | 37% | 41% |
| Local Hist. | threshold=0.7, bins=49, regionsize=16x16 | 88% | 44% | 13528 | 74% | 33% |
| Ext. Hist | Threshold=0.6, var=0.3, bins=41, regionsize=16x16, buffer=5 | 88% | 88% | 15503 | 68% | 65% |

# *Table 1. Optimal parameters for the different methods and obtained recall and precision.*

# Complexity of the different methods

The complexity of all methods is dependent on the resolution of the video sequences (. The discussion below only focuses on the other factors.

The pixel difference method only compares each pixel with the corresponding pixel in the previous frame, the complexity does not depend on the chosen parameter values. It only scales with the resolution of the video.

In the exhaustive motion estimation method, each block is compared multiple times with a block from the previous frame. Since displacements of the block between [- w, w] (w=window size) in both x and y direction are investigated, all pixels are compared times with pixels from the previous frame. This means that the total complexity is proportional to . The block size has no significant effect on the complexity since the total number of pixel comparison operations does not change with the block size. The optimized motion estimation method reduces the number of calculations by performing a logarithmic search procedure. In each step a block is compared with 9 other blocks. The step size (equivalent of window size of the exhaustive method) is divided by two in each iteration. This implies that the complexity is proportional 9(log2(s)+1). The block size again has no significant effect on the total number of operations. For the same search window size, the optimized version will always run faster. The speedup becomes larger with larger window sizes.

The complexity of the global histogram comparison method does not really depend on the chosen parameters. The memory footprint and computation time will slightly increase when more bins are used, but this will not have a significant impact on the overall complexity. In the local histogram method, the number of pixels that has to be checked also does not change with the parameter values. However, the number of histogram values that has to be compared increases with the number of bins and decreases with increasing region size. The memory usage will scale with the same dependencies. The sixth method is an extension of the local histogram comparison. For the complexity, the same relationships hold as for the local histogram comparison. In addition, the memory usage and calculation time will scale linearly with the length of the buffer.

# Differences between the methods

The Pixel Difference method compares each pixel with the corresponding pixel from the previous frame. It is the simplest and fastest method of the six, but the accuracy of the method is not great. Especially the recall values are low. The Motion Estimation methods both search for the best matching block in a region in the previous frame. They deliver better results than the Pixel Difference method, but they are also the slowest and most complex methods in the test. The Optimized version is faster than the Exhaustive, because it does not calculate all matches within the search window. It performs a logarithmic search. The Exhaustive method delivers better results, but the Optimized method is only slightly less accurate. The Histogram methods don’t calculate pixel differences. They calculate first one or more histograms and compare those. The advantage is that subtle differences in pixel values do not necessarily count as *mismatches*. They don’t look for matching blocks which makes them faster than the MotionEstimation methods. The Global Histogram method calculates one histogram for the whole image, while the other two calculate histograms for separate blocks. This makes them slower (more complex), but also perform better. The Global and Local Histogram methods perform on par with the MotionEstimation methods in terms of recall, but precision is noticeably lower. The Extended Histogram method (which uses more than one frame as history) is only a bit slower than the Local Histogram method, but it can vastly improve the precision values.

# Conclusions

### Best method

The extended local histogram method is clearly the best method. It provides the most accurate result of all tested methods and performs the calculations in a reasonable amount of time. The only downside of this method is the memory consumption when long buffer lengths are used. In terms of parameter settings, we would recommend to chose the buffer length not too big, since this buffer also equals the minimum shot length (5 to 10 performs well in most cases). Larger block sizes also deliver better results (e.g. 16x16). Thresholds between 0.5 and 0.7 deliver the best results. Variance has to be set lower than threshold/2 in order to obtain useful results.

### Improvements

This method could be further improved in terms of precision by investigating other criteria to compare the histograms of the current frame with those in the buffer. A combination of motion estimation and local histograms (e.g. looking for matching histograms in a search window) might also deliver some improvements.

In terms of processing speed, the biggest improvements can be found in parallelising the calculation of the histograms and the comparison between the histograms in the buffer. Short circuiting comparisons might also be a solution (e.g. do not compare with the histograms of all previous frames if comparing with two frames already shows that it is a good match).

### Shortcomings

We discovered during the tests that the optimal choice of parameters is dependent on the content of the video. The parameters should be automatically adapted to the specific kind of content. We also noticed that the start point of a shot can be ambiguous when gradual transitions occur, which makes it hard to compare the results of different methods. We would propose to define an arbitrary rule to break ties in this kind of scenarios. A third shortcoming is the ability to handle large sudden changes in one scene (e.g. the explosions in the jedi video). We would solve this by checking if a frame did resemble another frame some frames ago, which can then be used to conclude that something happened, but it is still the same shot.

### Influence of the spatial resolution

The different methods can be expected to deliver better results with higher spatial resolutions, mainly because the frames contain more details, which makes comparison more reliable. The main disadvantages is the complexity. For all methods it scales with (w\*h). In the motion estimation methods, the window/step size will have to scale with the resolution as well to cover the same *search area*. This will further increase the computation time when a higher resolution is chosen, especially in the exhaustive motion search (quadratic dependency on the window size)

### Influence of the framerate

Increasing the framerate will probably make the detection more reliable in sequences with lots of motion. It will make the difference between subsequent frames from the same scene smaller. The computation time will linearly scale with the framerate because more frames have to be processed. It has no influence on the computation time for one frame.

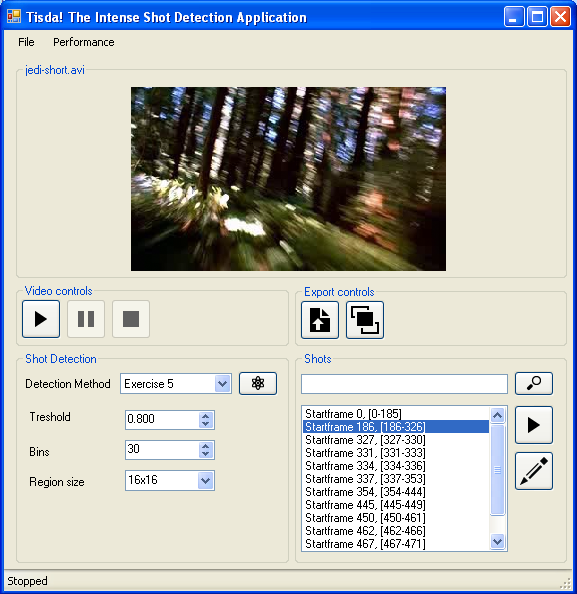
### Importance of metadata

Adding metadata to video content makes it easier to search for specific content. The metadata gives a high level interpretation of the content. This information can be directly used by software for high-level analysis or search. Furthermore, the *internet of things* requires a high-level description of the content in order to create relationships between content.

### Possible deployment problems

The fact that the users has to chose the parameter values requires that he/she has sufficient background knowledge about video processing. In this respect, adaptive algorithms can determine (some of) those parameters themselves. This makes it easier for the end user. We noticed that the parameter selection is very sensitive. Accuracy hugely depends on a good choice for these parameters. A second problem is that the accuracy of the algorithms is not yet good enough for reliable shot detection without user intervention. Although it is not a real problem, the required calculation time/effort can be an obstacle.

# Appendix



**2**

*Figure 4. The GUI of the ‘Tisda’ application*