University of Marburg at TRECVID 2005: Shot Boundary Detection and Camera Motion Estimation Results

Ralph Ewerth[1,2], Christian Beringer[1,2], Tobias Kopp[2], Michael Niebergall[1,2], Thilo Stadelmann[1,2], and Bernd Freisleben [1,2]

[1] SFB/FK615, University of Siegen
D-57068 Siegen, Germany

[2] Department of Mathematics and Computer Science, University of Marburg,
Hans-Meerwein-Str., D-35032 Marburg, Germany

{ewerth, beringec, kopp, nieberga, stadelmann, freisleb}@informatik.uni-marburg.de

Abstract. In this paper, we summarize our results in the shot boundary task and the low-level feature task at TRECVID 2005. The low-level feature task was to retrieve the shots in which one of the following camera motion events was present: pan, tilt and zoom. An unsupervised approach to detect shot boundaries, aimed at minimizing the impact of parameter settings, is presented [4]. Frame dissimilarities are measured by motion compensated pixel differences of subsequent DC-frames and histogram intersection of DC-frames for several frame distances. A feature vector consists of the dissimilarity value and its ratio to the maximum neighbor value within a sliding window. K-means clustering is used for both cut detection and gradual transition detection. For cut detection, the best sliding window size is estimated by evaluating the clustering quality of the "cuts" cluster for several window sizes. Furthermore, we investigate whether an ensemble of classifiers improves the cut detection performance. For this purpose, the unsupervised learning approach is extended by two classifiers: an Adaboost-based classifier and a Support Vector Machine (SVM). These classifiers were trained on the TRECVID 2004 shot boundary test set. To retrieve shots with camera motion events, we have modified a previously presented approach to camera motion [5]. MPEG motion vectors are utilized to estimate the rotation and zoom parameters in a 3D-camera model. However, since motion vectors are optimal with respect to compression, many of them often do not model "real" motion adequately and can thus be considered as "outliers". Furthermore, we exclude motion vectors at the frame border and motion vectors in the middle of a frame since often (moving) objects of interest are captured in this frame area. Finally, the motion parameters must exceed a threshold for several frames to be considered as camera motion.

1 Structured Abstract

The paper is structured as follows. In this section, the results of our participation in both tasks are presented in form of the requested structured abstract. Algorithmic details are presented in section 2. A more detailed description of the experimental results is given in section 3. Section 4 concludes the paper.

The following definitions are used in this paper:

$$recall = \frac{\#correctDetectedItems}{\#Items}$$

$$precision = \frac{\#correctDetectedItems}{\#correctDetectedItems + \#falseAlarms} \qquad (1\text{-}3)$$

$$f1 = \frac{2*recall*precision}{recall + precision}$$

Shot Boundary Detection: "What approach or combination of approaches did you test in each of your submitted runs?"

An unsupervised approach which is aimed at minimizing the impact of parameter settings has been investigated [4]. K-means clustering is used for both cut detection and gradual transition detection.
Two different frame dissimilarity measures are applied to detect cuts: Motion-compensated pixel differences of subsequent DC-frames [9, 13] and the histogram dissimilarity of two frames within a pre-defined temporal distance of e.g. 2. A sliding window technique similar to [13] is applied to measure the relative local height of a peak value. For cut detection, the best sliding window size is estimated by evaluating the clustering quality of "cut clusters" for several window sizes. Thus, the maximum sliding window size serves as a parameter for both dissimilarity metrics. For cut detection, the unsupervised approach is optionally extended by two classifiers in order to build an ensemble of classifiers. An Adaboost and an SVM classifier is incorporated in that ensemble of classifiers.
Several frame dissimilarity measures are applied to detect gradual transitions for different frame distances. Feature vectors are created similar to the cut detection approach using a sliding window technique. K-means is applied to these feature vectors. This approach is extended by a fade detector following the proposal in [11]. Finally, false alarms are removed if the start and end frame of a transition are too similar.

Shot Boundary Detection: "What, if any significant differences (in terms of what measures) did you find among the runs?"

The different settings of the *maximum* possible sliding window size had very little impact on the detection results, which indicates that the unsupervised approach has reached a high degree of parameter independence. Increasing the maximum sliding window size leads to a slightly higher precision rate while recall is slightly lower, as it can be expected. The application of an ensemble of classifiers improved the cut detection performance.
The parameter defining the sliding window size had also little impact on the detection of gradual transitions which was quite stable over all runs. The f1-measure ranged between 0.681 and 0.706. The frame-based recall and precision measures were clearly more balanced when the option "resize" was enabled.

Shot Boundary Detection: "Based on the results, can you estimate the relative contribution of each component of your system/approach to its effectiveness?"

The ensemble approach increased the cut detection performance in all cases. Comparing the f1-measure for cut detection, the ensemble's f1-results are between 0.903 and 0.912 while the results of the basic unsupervised approach are between 0.893 and 0.899. In all cases, the ensemble approach led to a better result in terms of both recall and precision.
Resizing the gradual transition intervals after clustering improved the f1-measure for the frame-based measures from about 0.52 up to 0.625 respectively 0.672.

Low-level Feature Task: "What approach or combination of approaches did you test in each of your submitted runs?"

Instead of computing the optical flow, our approach utilizes motion vector information embedded in compressed MPEG videos. Two steps are involved to generate a reliable motion vector field. The motion vectors at the frame border as well as the area in the middle of a frame are ignored. An adequate outlier removal algorithm is applied to the remaining field to remove vectors which are probably not related to motion. An optimization approach estimates the rotation and zoom parameters of a 3D-camera model for each P-frame in a video. A certain type of camera motion is assumed to be present within a shot if its related camera model parameter exceeds a pre-defined threshold in this shot for several frames. We have submitted one low-level feature run.

Shot Boundary Detection and Low-level Feature Task: "Overall, what did you learn about runs/approaches and the research question(s) that motivated them?"

The unsupervised approach to shot boundary detection has reached a mature level of robustness and detection quality, in particular for the task of cut detection. The cut detection performance could be improved via an ensemble of classifiers. One third of the cut detection false alarms has been caused by cuts which were edited in a small local frame area (picture-in-picture), mainly in one of the test videos. It seems that the subjective task of ground truth data creation affects detection measures by about 2-3% for our approach in terms of precision, mainly for cut detection.
The camera motion estimation approach which makes use of motion vector information embedded in compressed videos worked very well. The detection of zoom achieved a precision of 0.931 and a recall of 0.894. In terms of the f1-measure, this was the best submission for this task this year. Detecting tilt and pan worked also well with a precision of 0.962 and a recall and 0.724, and 0.924 (precision) and 0.761 (recall), respectively. For tilts, only the runs of one other institute achieved a better f1-value.
Thus, it seems to be possible to approximate the optical flow based on motion vector information. Clearly, the filtering of the motion vector field is essential to achieve good results. We suppose that the very good zoom estimation benefited from the consideration of motion vectors in an outer frame area (but not at the border) in combination with the use of a 3D-camera model. Future experiments should investigate this subject.

2 Algorithmic Details

2.1 Shot Boundary Detection: Algorithmic Details

The shot boundary detection approach is split up in two parts in order to detect cuts and gradual transitions appropriately.

2.1.1 Cut Detection

Unsupervised learning is utilized in the cut detection approach which is optionally extended to an ensemble of three classifiers. For this purpose, two additional classifiers are trained on an appropriate training set. The unsupervised approach works as follows.
Two frame dissimilarities are used for the unsupervised cut detection task. Motion compensated pixel differences of subsequent frames (i.e. their frame distance is 1) and frame histogram differences are computed. The histograms have 512 bins where each bin represents a combination of the Y, Cb and Cr color channel each with 8 quantization levels. A frame distance larger than 1 (e.g. 2 or 3) is used for the

histogram differencing in order to detect very short gradual transitions. Time series that are based on a frame distance *n>1* are subsampled by a factor of *n*.
For the dissimilarity values with a frame distance of 1, frame difference normalization is applied [2, 3] in order to remove noise and compression artifacts in the dissimilarity time series. Two features are then extracted for both metrics for those frame positions where the dissimilarity is the maximum within a sliding window of size *2*m+1*:

1.) the ratio of the dissimilarity value divided by the maximum dissimilarity value in this video, and,

2.) the ratio of the second largest value divided by the maximum of the sliding window.

Then, k-means (k is known a-priori in case of cut detection: 2) is applied to all feature vectors belonging to the same sliding window size and the same metric. The cluster whose average feature vector is nearer to the feature space point (1, 1) is considered as the "cuts" cluster. Now, for each "cuts"-cluster and for each sliding window size and metric the silhouette coefficient is computed which describes the compactness of a cluster (more details are described in [5]). The cluster with the highest coefficient represents the best sliding window size for a given metric and is considered as the cut detection result. If a cut is detected using both metrics, only the shorter transition is included in the final result.

2.1.2 Video Cut Detection Using an Ensemble of Classifiers

Furthermore, the possibility to improve the cut detection performance by combining multiple cut detection "experts" was investigated. It has been shown that such an ensemble of classifiers can improve accuracy in recognition tasks [6]. Since most transitions in a video are abrupt (without any transitional frames between the different shots), the ensemble was added to the unsupervised approach for cut detection.
We have chosen Adaboost (e.g. described in [12]) as the first classifier to select the best features for a given training set (in our case the TRECVID 2004 shot boundary test set). The key idea of the Adaboost approach is to combine a number of *n* "weak classifiers" to build a strong classifier within *n* rounds of training. For each feature, a threshold is estimated which minimizes the classification error. The classification error is computed based on the weights of the training samples. Misclassified training samples are re-weighted such that they have more impact in the next training round for the next "weak classifier". Each "weak" classifier's weight depends on its error rate. The final strong classifier rule checks if the weighted sum of the weak classifiers' positive votes exceeds a threshold. For the task of cut detection, we have defined 42 features for a certain frame distance describing dissimilarity of DC-frames with respect to:

- motion compensated pixel differences,
- histogram differences,
- luminance mean and variance,
- edge histograms of Sobel-filtered (vertically and horizontally) DC-frames,
- local histogram differences, and
- ratio of the second largest dissimilarity value divided by the local maximum for several sliding window sizes.

In this study, we have further used Adaboost for feature selection where the best *m<=n* features are used to train a SVM on the same test set of 2004. Two frame distances (1 and 2) were investigated resulting in a total feature number of 84. Thus, we finally got three classifiers evaluating each frame (considering the unsupervised approach as a kind of classifier as well). A majority vote is implemented in our approach, i.e. a cut is detected if at least two "experts" vote that a frame belongs to a new shot.

2.1.3 Gradual Transition Detection

The main idea of the gradual transition detection approach is to view a gradual shot change as an abrupt shot change at a lower temporal resolution. It is also an unsupervised approach. This basic approach is extended by a fade detector following the approach in [11]. The approach works as follows:
First, frame dissimilarities are computed based on histograms of approximated DC-frames. Those dissimilarities are computed for a certain temporal resolution Δt_j (where #J is the number of temporal resolutions and $j \in \{0, 1, ..., J-1\}$). Thus, a set of frame dissimilarity values $\{d_{0, \Delta t0}, d_{1, \Delta t0}, ..., d_{i, \Delta tj}, ..., d_{z, \Delta t(J-1)}\}$ is obtained for all temporal resolutions Δt_j, where $d_{i, \Delta tj}$ is the dissimilarity value for the frames $i*\Delta t_j$ and $(i+1)*\Delta t_j$, and $d_{z, \Delta tj}$ is the last dissimilarity value of temporal resolution Δt_j with index *z*. The feature vectors are now created similar to the task of cut detection and consist of two components, too. The basic sliding window size of $2*m+1$ is computed separately for each temporal resolution Δt_j by: $max(m/\Delta t_j, c)$, where c is a constant, e.g. c=1. Then, these feature vectors are clustered using k-means (again with 2 clusters) in *one* clustering process. If two or more feature vectors are finally in the "gradual transition cluster" and have a frame overlap or a cut has been detected in this frame interval before, then, the longer transition(s) are removed. The transition start and end positions are optionally refined by comparing the dissimilarity between pairs of frames in the transition interval. Finally, false alarms are removed if the frame dissimilarity between start and end frame is below a threshold.

2.2 Camera Motion Estimation (Low-Level Feature Task)

In the experiments we have tested an approach to camera motion which is mainly based on our proposal in [5]. This approach allows the estimation of camera motion in MPEG videos and works directly on motion data available from the compressed video stream. The algorithm consists of three main steps:

1) *Extraction of motion vectors.*
2) *Computing a reliable motion vector field.*
3) *Estimation of camera motion parameters.*

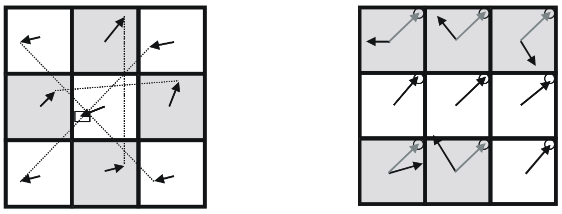

Figure 1: Two examples for outlier criteria showing a motion vector and its neighbors. Left: "smoothness"; Right: "neighborhood".

Compared to our previous proposal [5], some modifications have been made in step 2 and step 3. A large area in the middle of the motion vector field was not considered in the subsequent processing since moving objects are often present in this frame area. Motion vectors at the border of the frame were not considered as well. Originally, we have also used this approach to deal with the distinction of translational and rotational camera movement. Since this was not required in the TRECVID low-level feature task, we resigned to estimate the translational motion component and used only the rotation and zoom parameters from the 3D-camera model. The three main steps are now described in more detail.

1 Extraction of motion vectors. The motion vectors are extracted directly from the compressed MPEG stream. In MPEG, the encoding of a P-frame is based on a previous reference frame, while the encoding of a B-frame can be based on two reference frames, a previous as well as a subsequent reference frame. Only the motion vectors from P-frames are processed in our approach, for two reasons. First, usually each third to fifth frame in a MPEG video is a P-frame, and thus, the temporal resolution is sufficient for most applications. Second, both the prediction direction and the temporal distance of motion vectors are not unique in B-frames, resulting in additional computational complexity. For each macroblock, a motion vector is estimated which points to a similar block in a reference frame. Motion estimation algorithms try to find the best block match in terms of compression efficiency. This can lead to motion vectors that do not represent the camera motion or object motion at all, which e.g. is possibly the case for homogenous areas in images due to noise and low image quality.

2 Computing a reliable motion vector field. To achieve a motion vector field which approximates the "real" optical flow well, the motion vector field has to be processed adequately. First, the motion vectors at the frame border and in the middle of the frame are removed. The latter ones are removed to reduce the impact of moving objects which are usually captured in the middle of a frame. Additionally, we apply an outlier removal algorithm which was proposed in [1]. There are two main steps in the algorithm. A motion vector MV is declared as an outlier if both of the following criteria are not met (see the examples shown in Figure 1):

2.1. Smooth change. MV is compared to each average of four pairs of opposite neighbors – if the number of averages that are close to the central MV is below a threshold, then the criterion of smoothness is not met.

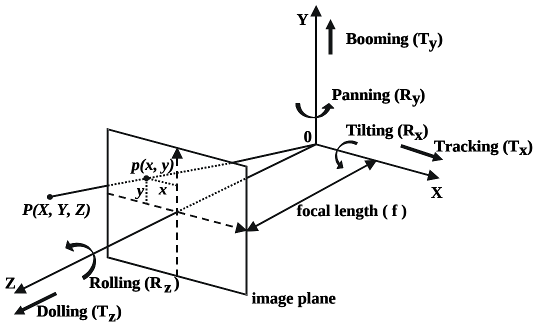

Figure 2: The 3D camera model that has been used in the proposed approach.

2.2. Neighborhood. A neighborhood motion vector supports the central MV if it lies within a tolerance circle. If the number of supporting vectors is below a threshold, then the criterion of neighborhood is not met.

3 Estimation of camera motion parameters. A 3D model (see Figure 2) as described in [10] has been chosen to describe camera motion events. Formulas 4 and 5 describe the translational components u_x and u_y in the image plane depending on the focal length f, the translational movement t_x and t_y along the x-axis and y-axis, the rotational components $r_{x,}$ r_y and r_z around all three axes, and the zoom factor r_{zoom}. Consider an external point at (X, Y, Z) which is projected onto the image plane at point (x, y), where x = f*(X/Z) and y = f*(Y/Z). Motion estimation is done as follows.

$$u_x(x,y) = -\frac{f}{z} * t_x - \frac{x*y}{f} * r_x + f\left(1+\frac{x^2}{f^2}\right) * r_y - y * r_z + f\left[\tan^{-1}\left(\frac{x}{f}\right)\right]\left(1+\frac{x^2}{f^2}\right) * r_{zoom} \tag{4}$$

$$u_y(x,y) = -\frac{f}{z} * t_y - \frac{x*y}{f} * r_y + f\left(1+\frac{y^2}{f^2}\right) * r_x - x * r_z + f\left[\tan^{-1}\left(\frac{y}{f}\right)\right]\left(1+\frac{y^2}{f^2}\right) * r_{zoom} \tag{5}$$

Let ν_i be the difference vector between a motion vector at macroblock position (x, y) in the original and the estimated motion vector field, let V be the sum of all ν_i, and let θ_i be the absolute angle between a vector ν_i and V. As suggested in [10], the parameter values $r_{x,}$ r_y, r_z and r_{zoom} are estimated by minimizing the term $P = \Sigma\ \nu_i^2 * \theta_i$ using the Nelder-Meade algorithm. This results in an estimated motion vector field VF where the difference vectors ν_i are mostly parallel. In contrast to [10], r_z is not used and VF is not further considered to estimate translational camera motion parameters t_x and t_y since such a fine distinction was not requested in the low-level feature task. Finally, the motion parameters must exceed a threshold for several frames in a shot to be considered as camera motion.

3 Experimental Results

3.1 Experimental Settings

The shot detection approach has been tested with the following parameter settings for all runs. The frame distance for the second cut detection metric was set to 2. The frame distances for the gradual transition detection were set to: 6, 10, 20, 30, 40, 50. The MDC decoder has been used for MPEG decoding [7]. Feature selection using Adaboost was performed on the TRECVID 2004 shot boundary test set. Eleven features were selected from the whole feature set to build an Adaboost classifier. The best 7 features were used to train a SVM (using the library "LibSVM" [14]) on eight of the twelve videos from the last year's test set. Only a subset of features and videos was chosen, since training of the SVM is a very time-consuming task. For the camera motion estimation task, the thresholds were set to: $r_x = 0.001$, $r_y = 0.0015$, and $r_{zoom} = 0.00075$. They were estimated using the TRECVID 2005 training set.

3.2 Experimental Results

The experimental settings and the results for the different runs are shown in Table 1 and 2. The parameters for the sliding window sizes had very little impact for both cut detection and gradual transition detection. Increasing the maximum sliding window size led to slightly better precision values whereas recall decreased very slightly. Using a classifier ensemble improved cut detection performance slightly: Comparing the f1-measure for cut detection (see f1-measures for all submitted runs and all shot boundary tasks in Table 3), the f1-measures for the ensemble runs are between 0.903 and 0.912 while the results of the basic unsupervised approach are between 0.893 and 0.899. In all cases, the ensemble approach led to better results in terms of both recall and precision. The contribution of the classifiers to the ensemble's performance was analyzed as well. Table 4 shows the cut detection results (for a cut "length" of 0 and <= 5) for only the Adaboost classifier respectively the SVM classifier on the TRECVID 2005 test set. The SVM achieves a very high precision for the cut "transition" with a length <= 5, whereas the Adaboost classifier is superior in recall. The low precision of 57.7% for the Adaboost classifier is mainly caused by one video ("NASA-Connect-AO.mpg") whereas the precision for the other videos is about 83%.
The sliding window size parameter had also very little impact on the detection of gradual transitions which was quite stable over all runs. The f1-measure ranged between 0.681 and 0.706. The frame-based recall and precision measures were clearly more balanced when the option "resize" was enabled.
The camera motion results are displayed in Table 5 and Table 6. The detection of zoom worked very well with a precision of 0.931 and a recall of 0.894 (f1-measure: 0.912). In terms of the f1-measure, it was the best submission for this task this year. Detecting tilt and pan worked also well with a precision of 0.962 and a recall and 0.724 (f1: 0.826) respectively 0.924 and 0.761 (f1: 0.835). For tilts, only the runs of one institute achieved better f1-measures.

Run	Cuts: MaxWin Size Metric1	Cuts: MaxWin Size Metric2	Ensemble	False Alarm Removal	Gradual: Max Win Size	Gradual: Resize Transition
marburg0	15	5	1	0	20	1
marburg1	15	5	1	1	20	0
marburg2	15	5	0	0	24	0
marburg3	15	10	1	1	24	1
marburg4	15	10	0	0	24	1
marburg5	15	15	1	0	20	0
marburg6	18	9	1	1	12	0
marburg7	18	9	0	0	12	0
marburg8	15	15	1	1	20	0

Table 1: The parameter settings for the different runs.

Run	Cuts		Gradual Transitions		Gradual T. Frame-based		All Transitions	
	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.
marburg0	0.928	0.880	0.694	0.672	0.588	0.784	0.868	0.828
marburg1	0.932	0.888	0.704	0.692	0.799	0.393	0.874	0.839
marburg2	0.936	0.864	0.715	0.684	0.798	0.396	0.880	0.820
marburg3	0.922	0.891	0.717	0.667	0.652	0.600	0.870	0.833
marburg4	0.920	0.867	0.731	0.656	0.649	0.602	0.871	0.811
marburg5	0.925	0.895	0.718	0.685	0.793	0.390	0.872	0.841
marburg6	0.926	0.892	0.719	0.646	0.819	0.358	0.874	0.826
marburg7	0.924	0.868	0.735	0.628	0.818	0.359	0.876	0.803
marburg8	0.924	0.900	0.722	0.691	0.849	0.387	0.873	0.846

Table 2: Recall and precision for the different runs, separated for cuts, gradual transitions, for gradual transitions on a frame basis, and for all transitions.

Run	Cuts	Gradual Transitions	Gradual Transitions Frame-based	All Transitions
marburg0	0.903	0.683	0.672	0.848
marburg1	0.909	0.698	0.527	0.856
marburg2	0.899	0.699	0.529	0.849
marburg3	0.906	0.691	0.625	0.851
marburg4	0.893	0.691	0.625	0.840
marburg5	0.910	0.701	0.523	0.856
marburg6	0.909	0.681	0.498	0.849
marburg7	0.895	0.677	0.499	0.838
marburg8	0.912	0.706	0.532	0.859

Table 3: F1-measures for all runs, separated for cuts, gradual transitions, frame-based detection performance of gradual transitions, and all transitions.

Cut Detection Performance	**Recall**	**Prec.**
SVM (cut transitions with length <=5)	**0.838**	**0.947**
SVM (cut transitions with length <=0)	0.939	0.871
Adaboost (cut transitions with length <=5)	**0.957**	**0.577**
Adaboost (cut transitions with length <=0)	0.978	0.484

Table 4: Experimental results for the classifiers used in the ensemble approach.

Run	**Pan**		**Tilt**		**Zoom**		**Total Average**	
	Recall	**Prec.**	**Recall**	**Prec.**	**Recall**	**Prec.**	**Recall**	**Prec.**
marburg0	0.761	0.924	0.724	0.962	0.894	0.931	0.793	0.939

Table 5: Recall and precision for the different camera motion types: pan, tilt, zoom, and the average for all types.

Run	**Pan**	**Tilt**	**Zoom**	**Total Average**
marburg0	0.835	0.826	0.912	0.860

Table 6: F1-measures for the different camera motion types: pan, tilt, zoom, and the average for all types.

4 Conclusions

Our unsupervised approach to shot boundary detection has reached a mature level of robustness and detection quality, in particular for the task of cut detection. The cut detection performance was improved using an ensemble of classifiers achieving a recall of 0.925 and a precision of 0.90 in the best case. One third of the cut detection false alarms was caused by cuts which were edited in a small local frame area (picture-in-picture) and have not been annotated as cuts, mainly in the test video "20041102_160001_cctv4_daily_news_chn.mpg". This affected the detection measures for our approach by about 2-3% in terms of precision, mainly for cut detection. The detection of gradual transitions was satisfactory with an f1-measure of about 0.7, and its improvement will be subject to future work, in particular with respect to frame accuracy.

The camera motion estimation approach which makes use of motion vector information embedded in compressed videos worked very well. The detection of zoom achieved a precision of 0.931 and a recall of 0.894. In terms of the f1-measure, this was the best submission for this task this year. Detecting tilt and pan worked also well with a precision of 0.962 and a recall and 0.724 respectively 0.924 and 0.761. For tilts, only the runs of one other institute achieved better f1-values. Thus, it seems to be possible to approximate the optical flow based on motion vector information. Clearly, the filtering of the motion vector field is essential to achieve good results. We suppose that the very good zoom estimation benefited from the consideration of motion vectors in the outer frame area (but not at the frame border) in combination with a 3D-camera model. Future experiments will investigate this subject.

5 Acknowledgements

This work is financially supported by the Deutsche Forschungsgemeinschaft (SFB/FK 615, Teilprojekt MT).

6 References

1. Dante, A., and Brookes, M.: Precise Real-Time Outlier Removal from Motion Vector Fields for 3D Reconstruction. In Proc. of IEEE International Conference on Image Processing, Vol. 1, Barcelona, Spain, (2003) 393-396
2. Ewerth, R., and Freisleben, B.: Improving Cut Detection Algorithms for MPEG Videos by GOP-Oriented Frame Difference Normalization. In Proc. of the 17th International Conference on Pattern Recognition, Vol. 2. Cambridge, United Kingdom (2004) 807-810
3. Ewerth, R., and Freisleben, B.: Frame Difference Normalization: An Approach to Reduce the Error Rates of Video Cut Detection in MPEG Videos. In Proc. of the IEEE International Conference on Image Processing, Vol. 2. Barcelona, Spain (2003) 1009-1012
4. Ewerth, R., and Freisleben, B.: Video Cut Detection without Thresholds., Proc. of 11th Int'l Workshop on Systems, Signals and Image Processing, Poznan, Poland (2004) 227-230
5. Ewerth, R., Schwalb, M., Tessmann, P., and Freisleben, B.: Estimation of Arbitrary Camera Motion in MPEG Videos. In Proc. of the 17th International Conference on Pattern Recognition, Vol. 1. Cambridge, United Kingdom (2004), 512-515
6. Kuncheva, L. I., Whitaker, C. J., Shipp, C. A., and Duin, R. P. W.: Limits on the Majority Vote Accuracy in Classifier Fusion. In Pattern Analysis and Applications, Vol. 6, No. 1, , Springer-Verlag, London (2003) 22-31
7. Li, D., Sethi, I.: MPEG Developing Classes.
   http://www.cs.wayne.edu/~dil/research/mdc/docs.
8. Petersohn, C.: Fraunhofer HHI at TRECVID 2004: Shot Boundary Detection System.
   TREC Video Retrieval Evaluation Online Proceedings, TRECVID, 2004.
   URL: www-nlpir.nist.gov/projects/tvpubs/tvpapers04/fraunhofer.pdf
9. Shen, K., and Delp, E. J.: A Fast Algorithm for Video Parsing Using MPEG Compressed Sequences. In Proc. of IEEE ICIP 1995. Washington, DC. (1995) 252-255
10. Srinivasan, M. V., Venkatesh, S., and Hosie, R.: Qualitative Estimation of Camera Motion Parameters from Video Sequences. In Pattern Recognition, Vol. 30, No. 4, Elsevier Science Ltd. (1997) 593-606
11. Truong, B.-T., and Venkatesh, S.: New Enhancements to Cut, Fade and Dissolve Detection. In Proc. of ACM International Conference on Multimedia, Los Angeles, USA, (2000) 219-227
12. Viola, P., and Jones, M. J.: Robust Real-Time Face Detection. In International Journal of Computer Vision 57(2), Kluwer Academic Publishers, Netherlands, (2004) 137-154
13. Yeo, B., and Liu, B.: Rapid Scene Analysis on Compressed Video. In IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 6. (1995) 533-544
14. Chang, C.-C., and Lin, C.-J.: LIBSVM: A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.