Joanneum Research at TRECVID 2005 – Camera Motion Detection

Werner Bailer, Peter Schallauer, Georg Thallinger

Institute of Information Systems and Information Management
JOANNEUM RESEARCH
Steyrergasse 17, 8010 Graz, Austria

{firstname.lastname}@joanneum.at

Abstract

Low-level feature extraction (camera motion)

Ground truth annotation

Manual camera motion annotation has been performed by three groups using a tool provided by Joanneum Research. Some types of content made it difficult or impossible for human annotators to describe the camera motion. The comparison of annotations of the same content done by two groups shows significant differences for some features. We discuss the questions that arise from the results of the manual annotation.

Submitted runs

Our approach is based on feature tracking, clustering the feature trajectories and selecting the cluster representing dominant motion. The parameters for the decision about the presence of certain types of camera motion from the estimated motion parameter sequence differ between the runs.

JRS1: same parameters for pan and tilt, stricter parameters for zoom

JRS2: slightly reduced parameters for tilt, reduced parameters for pan and zoom

Modification of pan and tilt parameters increased recall by slight loss of precision, the tradeoff between precision and recall was higher for zoom. The decision about the presence of a certain type of camera motion from the estimated motion parameter sequence is most critical point. Cross-evaluation between parameters trained on the development or test data set on the respective other set should be done.

1 Common Annotation of Camera Motion Ground Truth

1.1 Introduction

The TRECVID common feature annotation task does not include the annotation of camera motion ground truth. As the camera motion detection task is new in TRECVID 2005, also no ground truth data from previous years has been available. However, for testing camera motion detection algorithms and tuning of parameters, the availability of ground truth is necessary.

Joanneum Research has thus organized the common annotation of camera motion ground truth among the participants of the low-level feature extraction task. The ground truth annotation has been based on the TRECVID master shot boundary reference provided by Fraunhofer HHI [9]. We have provided a tool for manual annotation of camera motion (described in detail in Section 1.2), which produces MPEG-7 output in the same format as the master shot reference plus the camera motion annotation. The annotation has been done on a shot basis, labeling a shot as containing one or more of the motion types pan, tilt and zoom or labeling a shot as undefined, if it was not easily possible for the human annotator to describe the camera motion of the shot.

The annotation has been performed on a part of the TRECVID 2005 development data set. The following three groups have participated in the common annotation of camera motion ground truth: KDDI R&D Laboratories, University of Marburg (Dept. of Mathematics and Computer Science) and Joanneum Research (Institute of Information Systems and Information Management).

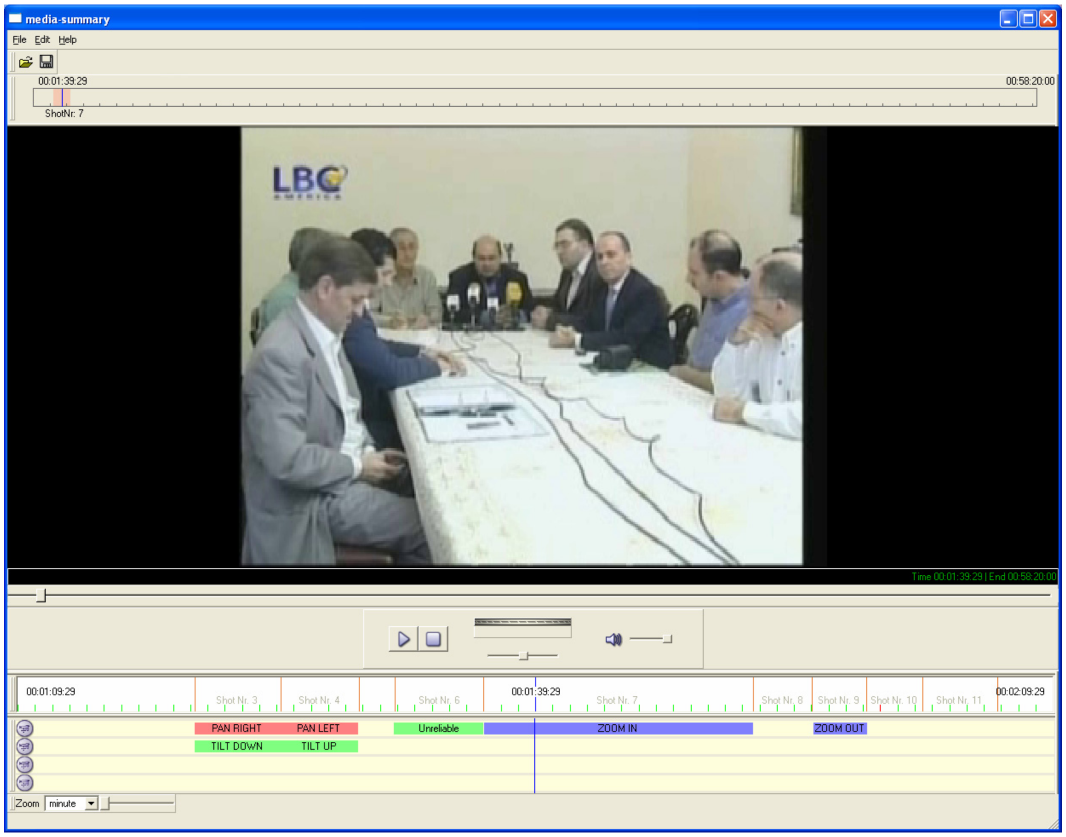

Figure 1: Screenshot of the tool for manual annotation of camera motion.

1.2 Tool for Manual Annotation of Camera Motion

Joanneum Research has provided a tool for the efficient manual annotation of camera motion. The tool allows a user to annotate the camera motion using the keyboard while the video is playing, i.e. annotation can be done in about real-time. It is based on components of the Joanneum Research Multimedia Mining Toolbox [7] and runs on the Win32 platform. It is implemented in C++, using Qt for the graphical user interface.

Figure 1 shows a screenshot of the annotation tool. Above the player, an overview timeline visualizes the duration of the whole video and highlights the time range that is displayed in the controls at the bottom. The component below the player and its controls is a visualization of the shot boundaries that are read from the input file. The camera motion view shows the camera motion annotation that have been read from the input file or manually annotated. Each camera motion annotation of a shot will appear on a new line, with the color indicating the type of motion. All visualizations are synchronized with the video player. The timeline, shot boundary and camera motion visualizations can also be used to navigate through the video.

The numeric block of the keyboard is used to perform the annotations. Those keys of the numeric block that can also be used as cursor keys are used for the annotation of pan left/right and tilt up/down. The central key (5) is used to annotate zoom in, the remaining outer keys are used to annotate zoom out. Additionally, the label defined/undefined can be toggled. Setting a shot to undefined removes all other camera motion annotation. According to the definition of the TRECVID task, camera motion will always be annotated on a whole shot. The camera motion view can also be used to edit annotation at a later time by deleting previous annotations or inserting new ones.

The tool uses the MPEG-7 files of the TRECVID master shot reference as input. The native MPEG-7 format of the annotation tool is the Detailed Audiovisual Profile [4], which is slightly different than the TRECVID MPEG-7 format, so that a XSL transform is performed before reading the input and after saving the output. The annotated camera motion is described using the MPEG-7 camera motion descriptor (cf. MPEG-7 part 3 [4]) with *MixtureCameraMotionSegment* elements. The camera motion descriptors are attached to the shots of the MPEG-7 master shot reference file.

The annotation tool is available at ftp://iis.joanneum.at/trecvid.

1.3 Results

During the common annotation, 40 videos of the TRECVID 2005 development data set (27.5 hours of video containing in total 11,918 shots) have been annotated. The annotators labeled 4,929 of the shots (41.4%) as containing camera motion. Of these shots containing camera motion, 927 (18.8%) were labeled to contain undefined motion. The shots labeled undefined are unevenly distributed across the data set, with the percentages of undefined shots in a video ranging from 0% to 29%.

The 4,002 shots with camera motion that could be recognized and described by the annotators, contained in total 4,436 pans, 1,492 tilts and 3,930 zooms. It is an interesting fact that each of the shots with defined camera motion has been found to contain on average 2.46 different camera motions. This means that if there is camera motion in a shot, mostly two or all three types of camera motion are present.

The results of the common annotation are available at ftp://iis.joanneum.at/trecvid.

1.4 Discussion

The common annotation of camera motion has not only resulted in the availability of ground truth, but has also provided insights in the properties of this video feature and the way it is perceived by humans.

1.4.1 Problems experienced by annotators

There were a number of cases in which it turned out to be difficult or impossible for the human annotator to describe the type of camera motion in a video. These shots have been labeled to contain undefined motion. The most common problems experienced by the annotators were the following:

Small amount of motion or short duration of motion. In some cases it has been difficult to draw the line between instabilities of the camera and real camera motion. Special cases of this problem are hand-held camera sequences and sequences shot with a camera mounted on a moving vehicle. In both cases there can be intended camera motion, which is often difficult to separate from the instabilities that are always present.

Computer generated sequences, animations. It is often not clear to decide whether there is a simulated camera motion or objects are moving around in front of a static camera. This issue has been also raised by Alan Pan and discussed on the mailing list [2].

Editing effects such as split screen, overlaid images or at transitions. In these cases there are often several regions present, and each of them may have a different camera motion. A correct annotation of camera motion would thus require the decomposition of the scene into regions and the annotation of the camera motion of each of those.

Sequences with very short shots, time-lapse sequences. In sequences with a high cut frequency, it may not possible for a human to determine the camera motion of each of the short shots when the video is viewed in real-time. The same is true for some time-lapse sequences.

Combinations of camera motion. As can be seen from the results of the manual annotation reported in Section 1.3, very often several types of camera motion are present in the same shot, sometimes simultaneously. During the annotation it has turned out that it may be difficult for the human annotator to recognize multiple camera motions correctly, especially if they differ in strength. For example, the annotator may not recognize a slight pan, if it occurs during a strong zoom.

From these problems the following questions arise, which should be considered for the definition of camera motion feature and the annotation guidelines in future:

- What is the minimum extent (duration and/or strength) of perceivable camera motion? When is a motion occurrence sufficiently distinct to be included in a retrieval result?
- What is the temporal granularity of the camera motion annotation? The results have shown that several camera motions occur in the same shot. Some of those are in fact simultaneous, but many of them are consecutive. Of course, the annotation of camera motion on a smaller granularity than a shot raises questions about the definition of the boundaries of a camera motion occurrence.
- What is the spatial granularity of the camera motion annotation? In cases of transition effects or split screen it would be necessary to annotate several camera motions, which of course enormously increases the annotation work.

1.4.2 Subjectivity of ground truth annotation

The annotation of ground truth by humans also raises the question of how subjective these annotations are. We have performed a small experiment by assigning one of the videos[1] to two groups for annotation. Then we have evaluated the two manual annotations against one another.

Table 1 shows the results of the evaluation. It can be noted that one group has annotated much more pans and tilts than the other one, while the number of zooms is about the same. The correlation between the annotations depends on the type of camera motion. While the annotations for zoom are most similar, only one third of the tilt annotations match. The most surprising result concerns the shots that were labeled to contain undefined motion. Not only is the number of unreliable shots very different, but there is no overlap between the sets of undefined shots in each of the annotations.

It has to be noted, that the policy for annotating these files was slightly different than that used by NIST for creating the ground truth [10]. While the intention of NIST has been to select only a subset of shots which do unambiguously contain camera motion or not, the goal for the ground truth annotation was to annotate everything that could be annotated. Although the annotators could make use of the undefined label, they did this to a very different extent.

It would be very interesting to repeat this experiment on a larger scale, i.e. using a larger set of video which are annotated in parallel by several annotators. Also the question arises if the same observations about subjectivity have been made with the annotation of high level features, which were also double annotated in TRECVID 2005. If the level of subjectivity is found to be different in these cases, it should be investigated if this is due to the different nature of the features or due to differences in the clarity of the definitions of the annotation tasks.

2 Low-level Feature Extraction Task (Camera Motion)

This section describes the approach of our camera motion detection algorithm and presents and discusses the results of the evaluation of the official runs.

2.1 Introduction

The detection of camera motion in an image sequence involves two basic problems. The first is the relation between the dominant motion in the sequence and the camera motion. It is possible to estimate the dominant motion to which the whole image is subject to. However, this is often but not necessarily the camera motion. For example, if a large object moves in front of the camera, the dominant motion will be estimated as the motion of this object, even if the camera is static. The second problem is that the motion of the camera during creation of the image sequence cannot be fully reconstructed, as it is not possible to discriminate between different types of camera motion that cause the same visual effect, for example, it is often not possible to distinguish between pan left and track left. If the target that has been shot is distant and the amount of motion is small, the translation of the camera cannot be distinguished from a rotation around its vertical axis.

Many approaches to camera motion estimation ignore the fact that camera motion can only be determined reliably over a larger time range and accept the most dominant motion between a frame pair as the camera motion. The alternative is to estimate a number of dominant motions and then decide over a longer time range. The selection of a dominant motion is based on the assumption that the camera motion is the most dominant one (e.g. the one with the largest region of support), and that it is consistent and smooth over the time range.

Our own previous work was based on this approach. We tracked points between frames of pairs and then clustered them by motion similarity. Then we would decide for a time window, which cluster represents the dominant motion. However, it turned out that the clusters were not stable over time and thus the one representing the camera motion could not be determined reliably.

Pan					Tilt				
True +	False +	False -	Prec.	Rec.	True +	False +	False -	Prec.	Rec.
52	0	27	1.00	0.66	15	1	29	0.94	0.34
Zoom					**undefined**				
True +	False +	False -	Prec.	Rec.	True +	False +	False -	Prec.	Rec.
45	5	3	0.90	0.94	0	2	12	0.00	0.00

Table 1: Evaluation of two manual camera motion annotations of the same video against one another.

[1] Video 160 of the development set (20041103_120000_NTDTV_NTDNEWS12_CHN.mpg).

The estimated camera motion can be described using a motion model with sufficient complexity. However, the description is more intuitive if the motion is described with common terms for camera operations, such as pan left/right, tilt up/down, zoom in/out and the amount of motion, even if not all of them can be determined analytically from the image sequence. This kind of annotation is especially useful for retrieval applications, where it can be used by the user to define the query. The problem is that at this stage the mapping from a sequence of motion parameters to a high level description of camera motion has to be done. This requires decisions about the minimum length and strength of a motion event to be perceived as a camera motion by a human viewer. The perception may depend on the content, for example human viewers will perceive a certain amount of motion differently whether it is the only motion or occurs in combination with a stronger motion, and viewers will accept different levels of camera instability depending on the environment.

In the TRECVID task the camera motion is annotated on shot granularity, so that in this case it is not necessary to identify segments containing a certain camera motion.

2.2 Approach

2.2.1 Overview

Our approach is based on feature tracking. The reason is that feature tracking is a compromise between spatially detailed motion description and performance. It still allows estimating the exact motion of objects and background separately, but is less expensive than estimating a dense motion vector field. The result of the tracking step is a set of feature trajectories. All trajectories existing throughout a time window (we use about 1/3 to 1/2 second) are clustered by similarity in terms of a motion model. In the current implementation, a four parameter model including horizontal and vertical translation, scaling and rotation is used. Clusters of trajectories from previous time windows are used to initialize clustering in the next one. From the resulting clusters, the dominant one in terms of temporal and spatial extent is selected as the cluster representing the camera motion. In the final step, a description of the camera motion is extracted from the motion parameter sequence associated with the dominant cluster. Figure 2 shows a diagram of the algorithm and the following sections explain the steps in more detail.

2.2.2 Feature tracking

Feature tracking is done on the input image sequence using the Lucas-Kanade tracker. The implementation we are using is an extension of that described in [1], which is included in OpenCV [8]. Our extensions concern the addition of new points for appearing objects or background, and the removal of points which cannot be tracked reliably any longer. The result of the tracking step for a time window is a set of point trajectories.

2.2.3 Clustering of trajectories

The clustering of the trajectories resulting from the tracking step is the key step of the algorithm. The reason for clustering trajectories instead of motion parameters in single frames is to achieve a more stable cluster structure over time. There are three main difficulties in clustering feature trajectories:

Temporal extent of trajectories. Not all trajectories may exist throughout the whole time window used for clustering. Trajectories may end when points exit the scene, cannot be reliably tracked any more or are occluded. New trajectories start when new reliable points are detected.

Unknown number of clusters. As the number of moving objects in the scene is not known, the number of clusters is also unknown. The number of resulting clusters of the previous time window serves as a hint, but object may appear or disappear.

Clustering by hidden feature. The feature trajectories just contain the *x*- and *y*-displacement of the moving objects. If clustering is performed using a more complex model than a simple translatory one, the feature used for clustering is hidden. A function must be defined which expresses how well a trajectory matches a set of motion parameters.

To deal with these issues, the clustering problem is separated into the following steps:

- Estimate a motion parameter sequence for a set of trajectories.
- Assign the trajectories to the motion parameter sequence.
- Cluster motion parameter sequences.

After initialization, these steps are performed iteratively until the cluster structure stabilizes.

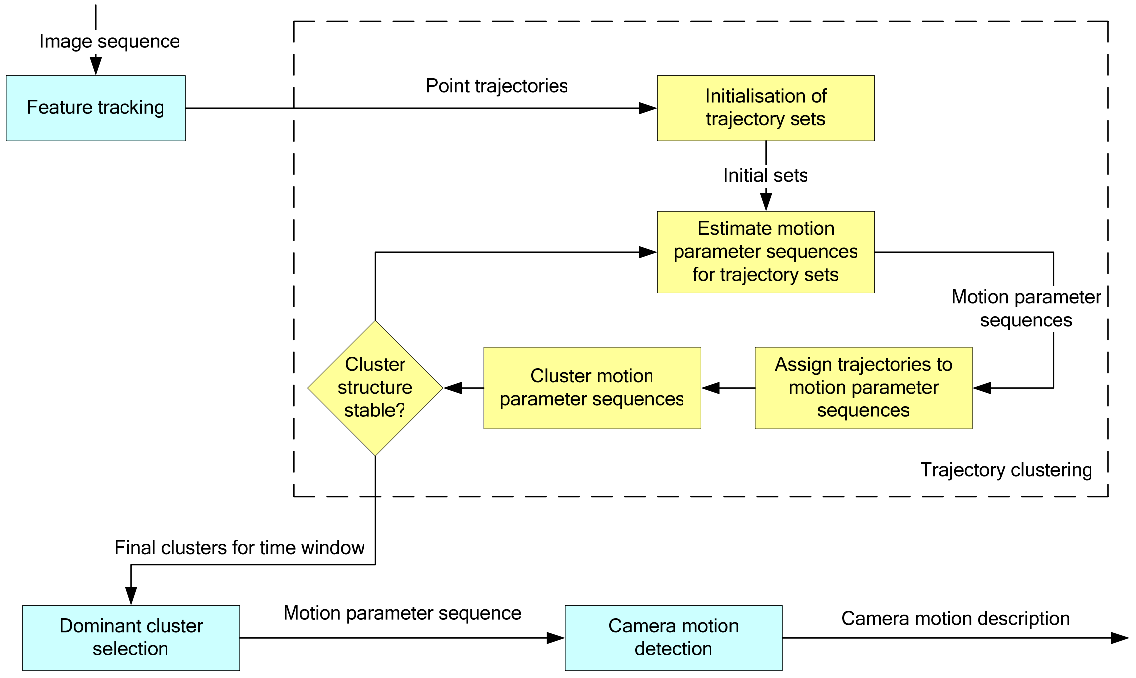

Image sequence
Feature tracking
Point trajectories
Initialisation of trajectory sets
Initial sets
Estimate motion parameter sequences for trajectory sets
Motion parameter sequences
Cluster structure stable?
Cluster motion parameter sequences
Assign trajectories to motion parameter sequences
Trajectory clustering
Final clusters for time window
Dominant cluster selection
Motion parameter sequence
Camera motion detection
Camera motion description

Figure 2: Diagram of the camera motion detection algorithm.

2.2.3.1 Initialization

The clustering step for a time window is initialized from the results of the previous one. The resulting motion parameter sequences are kept and those trajectories, that still exist, are assigned to parameter sets according to the previous assignment. All unassigned trajectories (i.e. those newly appearing in this time window) will be used to create additional sequences of motion parameters. Initial sets of unassigned trajectories are formed by grouping between 3 and 5 spatially adjacent trajectories, as it is likely that they belong to the same moving object. Motion parameter sequences are estimated for the trajectory sets as described in Section 2.2.3.2. and those with a low total matching error are selected as initial trajectory sets.

2.2.3.2 Estimation of motion parameters

The estimation of the sequence of motion parameters for a set of trajectories is formulated as an optimization problem. The function to be minimized is an error function containing the matching error between the measured trajectory and the prediction of the trajectory and a smoothness constraint on the motion parameter sequence. The matching error is calculated as the sum of the Euclidian distances between the measured point positions and those predicted using the current motion parameter sequence estimates over the frames of the time window. In the current implementation, a four parameter motion model with x and y translation, scaling and rotation is used. The smoothness constraint uses a penalty function for differences between subsequent motion parameter tuples (currently the difference of curvature is used), weighted to tolerate small deviations. This models that fact that in natural scenes neither camera nor object motion change abruptly but always gradually.

To solve the optimization problem, Newton's method with simple dogleg (using the implementation of [5]) is used. It has been found to perform better than other tested algorithms (e.g. BFGS in the implementation of [11]) on synthetic data, and at least as good as others on noisy data. The solution of the optimization problem is a motion parameter sequence that approximates the set of trajectories.

2.2.3.3 Assignment of trajectories to motion parameter sequences

After motion parameter sequences have been (re-)estimated, the trajectories are assigned based on the error between the measured trajectory and the predicted trajectory by using the estimated motion parameter sequence. The error for one trajectory is calculated using the matching error function described in 2.2.3.2. Each trajectory is assigned to the best matching motion parameter sequence in terms of this error function.

2.2.3.4 Clustering of motion parameter sequences

Clustering is performed on the motion parameter sequences, which has two main advantages over directly clustering trajectories: The cluster criterion is now a visible feature, i.e. the motion parameters itself, and the number of data items to be clustered is significantly less than the number of trajectories. Hierarchical clustering using the single linkage algorithm [3] is performed on the motion parameter sequences. As the goal in this application is the detection of the candidate clusters for dominant motion, the cutoff value for the clustering step has been selected so that usually a small number of clusters is returned, even if this might merge smaller moving objects into a similarly moving larger object or the background.

A distance function between motion parameter sequences has been used, which penalizes differences in scale and rotation higher than differences in translation. The distance values of the most similar clusters are also used as the terminating condition for the trajectory clustering step.

2.2.4 Dominant cluster selection

From the clusters resulting from the clustering step, the one representing the dominant motion of the sequence is selected. As only the cluster data has to be kept, this decision can be done over a long time range (up to several seconds). The dominant cluster is selected based on two criteria: the size of the cluster (i.e. the average fraction of points which is member of this cluster in every frame of the time range being evaluated) and the temporal stability of the cluster (i.e. the time for which it exists). As a result one cluster or a set of temporally only slightly overlapping clusters is selected. The dominant motion is described by the motion parameter sequences associated with these clusters.

2.2.5 Camera motion detection

The camera motion detection step analyzes the motion parameter sequence which has been found to represent the dominant motion and detects the presence of pan, zoom and tilt. The detection is done in a time window, for which the accumulated *x*- and *y* translation and the multiplied scale factor are calculated. In order to be robust against short time motion, the input is median filtered. The parameters of this step are the time window in which analysis is performed, the size of the median kernel, and the minimum thresholds for accumulated translation and multiplied scale factor.

2.3 Results

We have submitted two runs with different parameters for the camera motion detection step. All other settings were the same for both runs: the maximum number of points to be tracked was 300, the time window for trajectory clustering has been set to 10 frames, and subsequent windows have been overlapping by 3 frames.

For the camera motion detection step, the size of time window for motion analysis has been set to 25 frames, and a median kernel of 10 frames has been used. The thresholds for the accumulated translation and multiplied scale factors are shown in Table 2. The values for theses thresholds have been determined empirically from tests on the part of the development data set, for which ground truth annotation has been created.

Table 3 shows the evaluation results for the two submitted runs.

2.4 Discussion

The results are quite different for the different features. The pan threshold has been reduced between the two runs, resulting in 11% gain of recall with just a slight loss of precision. The decision not to reduce the threshold any further was due to the growing number of false positives on the development data. Further experiments on the test data should be done to determine the parameter that optimizes the tradeoff between precision and recall.

The threshold for tilt was higher than that for pan, as we have observed that the manual annotation of tilts on the development data has been more selective than that of pans. The results show—similar to the pan results—a minimal loss of precision for a much larger gain of recall, so that we should experiment with a further reduction of the tilt threshold. The zoom results show that the recall rates are the two best, with a rather low precision rate. The gain in precision between the two runs is about twice as much as the loss in recall.

	Pan	Tilt	Zoom
Run 1	0.064	0.064	0.017
Run 2	0.055	0.058	0.020

Table 2: Thresholds for accumulated translation values and multiplied scale factors. The unit of the translation values is the fraction of the image width/height.

	Pan		Tilt		Zoom		Mean	
	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Run 1	0.927	0.606	0.831	0.657	0.610	0.947	0.789	0.737
Run 2	0.919	0.676	0.828	0.686	0.640	0.933	0.796	0.765

Table 3: Results of the submitted runs.

It has not yet been analyzed how the results were influenced by our choices for the temporal parameters, i.e. the median filtering of the motion parameters and the time window used to determine the camera motion. For time reasons, only two runs have been done on the full test data set. In order to draw better conclusions on how the results develop with different parameters, more runs on the test data set will be performed. It would also be interesting to use in one test the manually annotated part of the development set and in another test the ground truth annotation of the test data set as training data to estimate the optimal parameters for our detection algorithm. Then the respective other data set could be used for evaluation. The similarity of the estimated parameters and of the evaluation results would show how data dependent the parameters of the detection algorithm are.

3 Acknowledgements

The authors would like to thank their colleagues at the Institute of Information Systems of JOANNEUM RESEARCH for their support and feedback, especially Peter Kovanda for doing camera motion annotation and evaluating preliminary results, Helmut Neuschmied and Franz Höller for adapting the camera motion annotation tool and Gregor Stefancic for helping with the high-level feature annotation. Many thanks also go to KDDI R&D Laboratories and University of Marburg for joining the common annotation of camera motion ground truth.

The work presented here has been funded partially under the 6th Framework Programme of the European Union within the IST projects “PrestoSpace” (IST FP6-507336, http://www.prestospace.org) and “NM2—new media for a new millennium” (IST-004124, http://www.ist-nm2.org).

4 References

[1] J.-Y. Bouguet, “Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm”, Technical Report, Intel Corporation, Microprocessor Research Labs 2000, OpenCV documentation.

[2] “Camera Motion Annotation for Animation?”, TRECVID 2005 mailing list archive. URL: http://cio.nist.gov/esd/emaildir/lists/trecvid2005/msg00154.html

[3] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*. 2nd ed., Wiley-Interscience, 2001.

[4] Information Technology—Multimedia Content Description Interface, Part 3: Visual. ISO/IEC 15938-3, 2001.

[5] C. T. Kelley, “Iterative Methods for Optimization”. URL: http://www4.ncsu.edu/eos/users/c/ctkelley/www/matlab_darts.html

[6] MPEG-7 Detailed Audiovisual Profile (DAVP). URL: http://mpeg-7.joanneum.at

[7] Multimedia Mining Toolbox. URL: http://www.joanneum.at/cms_img/img2287.pdf

[8] Open Source Computer Vision Library (OpenCV). URL: http://sourceforge.net/projects/opencvlibrary

[9] C. Petersohn, "Fraunhofer HHI at TRECVID 2004: Shot Boundary Detection System", *TREC Video Retrieval Evaluation Online Proceedings*, TRECVID, 2004. URL: http://www-nlpir.nist.gov/projects/tvpubs/tvpapers04/fraunhofer.pdf

[10] “Question about camera motion durations”, TRECVID 2005 mailing list archive, URL: http://cio.nist.gov/esd/emaildir/lists/trecvid2005/msg00164.html

[11] S. Ulbrich, “Optimieriung 3. MATLAB Routinen und Beispiele”. URL: http://www-m1.ma.tum.de/m1/personen/sulbrich/opt3/codes.html