

# Effective Camera Motion Analysis Approach

Shugao Ma and Weiqiang Wang

**Abstract**—Camera motion analysis (CMA) is very useful for many video content analysis tasks, but few works competently handle the videos with significant camera or object motion. In this paper, we present an effective CMA approach for such challenging cases. The effectiveness of our approach comes from two aspects: (1) reliably matching keypoints on consecutively sampled frames, where both the appearance similarity and motion smoothness of keypoints are considered; (2) effectively distinguishing background keypoints from foreground ones by a novel and advanced voting process, where the voting weights of keypoints are dynamically adjusted to guarantee that the influence of foreground motion can be largely reduced in CMA. Thus, a parametric camera motion model can be naturally derived by the accurate estimation of the motion of background keypoints. The experimental results on the TRECVID2005 dataset and 500 shots from seven classic action movies demonstrate the effectiveness of our approach.

## I. INTRODUCTION

CAMERA Motion Analysis (CMA) aims at finding out the information of camera movements during shooting based on analyzing video data, and the output can be a pattern of the camera motion (e.g., static, pan, tilt or zoom, etc.) or a parametric model that reflects the image coordinate correspondences between consecutive frames. Since camera motion information is important low level video feature, CMA is quite important for video retrieval, and it is also quite useful for video content analysis tasks such as sub-shot detection, keyframe extraction, foreground object motion analysis and event detection.

Many CMA approaches have been presented. The methods [1]-[3] are based on analyzing optical flows. However, when the camera moves fast, there will be significant displacement between consecutive frames, which may lead to inaccurate optical flow estimation. So these approaches generally fail in the case of fast camera motion between consecutive frames. Some efficient approaches [4]-[8] are investigated, which exploit motion vectors in MPEG compressed domain. Wang et al. [4] iteratively estimate the camera motion model on the set of motion vectors. An initial estimation of the camera motion model is computed by linear least square, and in each

of the following iterations, the motion vectors that have large bias from the model estimated in the previous iteration are removed before re-estimating the camera motion model. Kim et al. [5] applied median filter on the motion vectors' vertical and horizontal components, and estimated the camera motion model using the output of the filter. Lee et al. [6] divided the frame into several regions and defined some of the regions as "background region". The statistical patterns of the motion vectors in the background regions are compared to a set of predefined templates to analyze the camera motion pattern. They also used median filter on the motion vectors as a preprocessing stage. Ewerth et al. [7] used two rules named "smoothness" and "neighborhood" to filter out irrelevant motion vectors and then estimated an 8-parameter camera motion model using the Nelder-Mead algorithm. Tiburzi et al. [8] divided the video frame into several regions and used a set of rules to filter the motion vectors in each region. They compute the largest set of motion vectors that differ at most  $\pi/2$  in phase in each region. If the set contains less than 60% of the total vectors in the region, all the motion vectors in this region are filtered, otherwise the motion vectors that are not in the set are filtered. Then they used the remaining motion vectors to estimate the camera motion model. In sum, MPEG motion vectors estimated by video encoders are not always consistent with the actual movement of macro-blocks, and many of them correspond to the movements of foreground objects, so the effectiveness of these methods relies on their preprocessing stages to reduce the influence of irrelevant motion vectors. However, when the video contains significant camera or object motion, such irrelevant motion vectors may be prevailing, making those preprocessing stages ineffective. Apparently, they can only work for video with special encoding formats or techniques. Ngo et al. [9] proposed a video motion analysis approach based upon the pattern analysis of spatio-temporal slices. Their work included CMA but their approach can not generate a parametric camera motion model which is important for analyzing foreground object motion. Lertrudachakul et al. [10] analyzes camera motion by analyzing the trajectories of Harris interest points that are tracked for a long time. However, when the camera moves fast and the background content changes rapidly, interest points on background may not be tracked for long. Also, the Harris interest point detector is not invariant to scale and affine transform [11] which may be significant between consecutive frames when the camera moves fast. Battiato et al. [12] used motion vectors of SIFT features to estimate the camera motion in a video, but inaccurate results is more prone to be generated in their approach, since foreground and background features are

Manuscript received December 15, 2009. This work was supported in part by the National Key Technologies R&D Program under Grant 2006BAH02A24-2 and by National Natural Science Foundation of China under Grant 60873087.

Shugao Ma is with the Graduate University of Chinese Academy of Sciences, Beijing, China (phone: +86-10-62542135, e-mail: sgma@jdl.ac.cn).

Weiqiang Wang is with the Graduate University of Chinese Academy of Sciences, as well as the the Key Lab of Intelligent Information Processing, the Institute of Computer Technology, the Chinese Academy of Sciences. (phone /fax: +86-10-62645933, e-mail: wqwang@jdl.ac.cn or wqwang@ict.ac.cn).

treated without discrimination.

For videos with significant camera and object motions, large viewpoint changes and significant object motion between consecutive frames bring significant appearance variance. In this paper, we present a new CMA approach which works effectively for videos with significant camera and object motions. Firstly, a robust method is devised to match keypoints between consecutively sampled frames based on both the appearance similarity and motion smoothness of the keypoints. The keypoints are detected and described as local invariant features using the DoG detector and SIFT descriptor [13] because such local features have good invariance to viewpoint changes and occlusions, so they can be better matched between consecutive frames. Secondly, the camera motion pattern is qualitatively analyzed through a novel voting process in which background keypoints are distinguished from foreground ones and their voting weights are dynamically adjusted so that the influence of foreground motion can be largely reduced. At last, a parametric camera motion model is computed using only background motion vectors found in the voting process.

This paper is organized as follows. The three stages of our method are described in order in section II. The results of evaluation experiments are reported in section III. Section IV concludes the paper and describes our future work.

## II. OUR CMA APPROACH

This section first introduces the method of estimating the motion information of keypoints (section II.A), then the identification of camera motion patterns by a sophisticated voting process (section II.B), and finally the computation of camera motion model (section II.C).

### A. Tracking Keypoints

Our approach first extracts keypoints using DoG detector and represent them using the SIFT descriptor [13], and then match them between consecutive frames to track their movements.

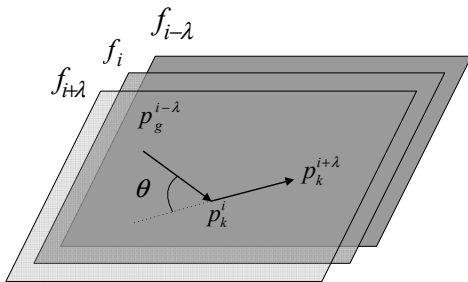


Fig. 1. Corresponding keypoints on three consecutive frames.  $p_g^{i-\lambda}$ ,  $p_k^i$ ,  $p_h^{i+\lambda}$  are on the three consecutively sampled frames  $f_{i-\lambda}$ ,  $f_i$ ,  $f_{i+\lambda}$  and they are matched through the keypoint tracking procedure.

Let  $p_a^b$  denote the  $a$ th keypoint on the  $b$ th frame  $f_b$  and  $\lambda$  represent the frame sampling interval. Fig.1 shows three

corresponding keypoints (i.e.  $p_g^{i-\lambda}$ ,  $p_k^i$ ,  $p_h^{i+\lambda}$ ) on three consecutively sampled frames (i.e.  $f_{i-\lambda}$ ,  $f_i$ ,  $f_{i+\lambda}$ ).

To promote the matching accuracy and speed, we assume:

1. The spatial distance between keypoint  $p_k^i$  and its correspondence  $p_h^{i+\lambda}$  is always within a specified range  $r_k^i$ .
- 2.

The motions of keypoints (speed and direction) always change smoothly.

These two assumptions usually hold for most videos: firstly, the time interval between two consecutively sampled frames is small, so the spatial distance between corresponding keypoints is limited; secondly, abrupt camera and object motion changes are usually not frequent, so the keypoints' motion trajectories are usually smooth.

So, for keypoint  $p_k^i$ , our algorithm searches for its correspondence  $p_h^{i+\lambda}$  within a circular region on  $f_{i+\lambda}$  whose origin and radius are  $p_k^i$ 's coordinate and  $r_k^i$  respectively (Fig. 2(b)). Compared with the approach in [12] which searches correspondences in the whole frame (Fig. 2(a)), our approach has two advantages: firstly, it promotes computation efficiency by avoiding unnecessary keypoint comparison; secondly, it enhances matching accuracy by filtering out those similar but far away keypoints which are common in natural scenes (e.g. repeated textures) and a mean source for false matches. Since the motion speed of different keypoints may vary, it's not appropriate to set  $r_k^i$  to a constant value because fast moving keypoint needs a larger search region. So we set  $r_k^i$  according to the keypoint's motion velocity, which is described later in this section.

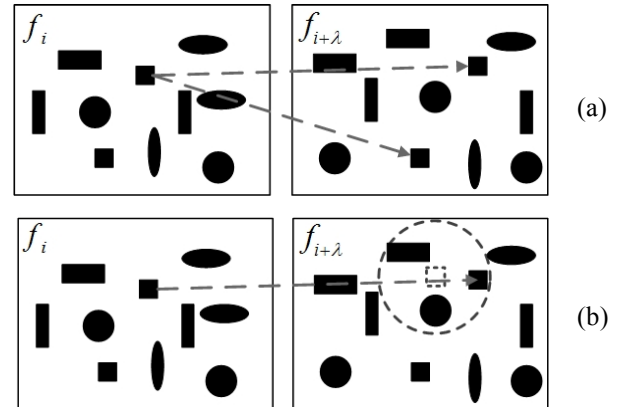


Fig. 2. Illustration and comparison of keypoint tracking strategies. The black spots denote the keypoints, in which the shape similarity of the spots represents the keypoint similarity. (a) is the straightforward way adopted by [12]. (b) illustrates our approach.

The matching score  $S(p_k^i, p_h^{i+\lambda})$  between keypoints  $p_k^i$  and  $p_h^{i+\lambda}$  is computed based on both SIFT feature similarity and motion smoothness using the following equation:

$$S(p, p') = w_a S_a(p, p') + w_m S_m(p, p'), \quad (1)$$

where  $S_a(p, p')$  denotes the SIFT feature similarity of  $p$  and  $p'$ ,  $S_m(p, p')$  measures the motion smoothness, and  $w_a$  and  $w_m$  are the corresponding weights (set to empirical values 0.7 and 0.3 in our system).  $S_a(p, p')$  is measured by the Euclidean distance between the two keypoints' SIFT feature descriptors, and  $S_m(p, p')$  is defined by:

$$S_m(p, p') = 1 - \theta_{(p, p')} / \pi, \quad (2)$$

where  $\theta_{(p, p')}$  is the included angle between the motion directions of keypoints  $p$  and  $p'$  (see  $\theta$  in Fig. 1). If  $p_k^i$  has no correspondence on  $f_{i-\lambda}$  or  $f_i$  is the initial frame, the motion direction of  $p_k^i$  is not available. In this case, we just let  $S(p_k^i, p_h^{i+\lambda}) = S_a(p_k^i, p_h^{i+\lambda})$ . Although  $S_m(p, p')$  only measures the motion direction change and neglects the velocity change, we found it works well enough. As the correspondence of  $p_k^i$ ,  $p_h^{i+\lambda}$  should have the highest matching score with  $p_k^i$  and this score should also be higher than a threshold. Then the corresponding motion vector  $\overrightarrow{p_k^i p_h^{i+\lambda}}$  is established.

If a keypoint moves fast, we need to check a relatively large region on the frame to search for its correspondence; if it is static or moves slowly, we only need to check a small region to avoid unnecessary key point comparison. So our algorithm dynamically set the search range  $r_k^i$  of keypoint according to the velocity  $v_g^{i-\lambda}$  of  $p_g^{i-\lambda}$  which is the correspondence of  $p_k^i$  on  $f_{i-\lambda}$ . If  $v_g^{i-\lambda}$  is not available (e.g.  $p_k^i$  has no correspondence on  $f_{i-\lambda}$  or  $f_i$  is the initial frame),  $r_k^i$  is set to a default value  $l_d * 0.2$ . Otherwise,

$$r_k^i = \max(l_d * 0.1, v_g^{i-\lambda} * 1.5), \quad (3)$$

in which  $l_d$  is the length of the frame's diagonal. The lower bound  $l_d * 0.1$  is set to prevent missing matching points.

Frame sampling interval is adjusted dynamically according to the extent of motion in the video. When the motion is insignificant, large  $\lambda$  is used to promote processing speed, otherwise small  $\lambda$  is used to guarantee matching accuracy. Specifically,  $\lambda$  is adjusted based on the average velocity  $\bar{v}_i$  of keypoints on a frame as follows:

$$\lambda = \begin{cases} \min(\lceil R_f / 3 \rceil, \lceil \hat{\lambda} * (\delta_s / \bar{v}_i) \rceil), & \bar{v}_i < \delta_s \\ \hat{\lambda}, & \delta_s < \bar{v}_i < \delta_f \\ \max(1, \lceil \hat{\lambda} * (\delta_f / \bar{v}_i) \rceil), & \bar{v}_i > \delta_f \end{cases}, \quad (4)$$

where  $\hat{\lambda}$  is the initial frame sampling interval (empirically set  $\hat{\lambda}=3$  in our system),  $R_f$  is the frame rate of the video,  $\delta_s = l_d * 0.02$  and  $\delta_f = l_d * 0.15$ . The upper bound value  $\lceil R_f / 3 \rceil$  is set to guarantee matching accuracy.

The process of keypoint tracking between  $f_i$  and  $f_{i+\lambda}$  is then summarized as follows:

### **Tracking keypoints**

#### **Input:**

$P_i$  and  $P_{i+\lambda}$  - Keypoint sets detected from  $f_i$  and  $f_{i+\lambda}$ .

#### **Output:**

$MV_{i,i+\lambda}$  - Keypoint motion vector set between  $f_i$  and  $f_{i+\lambda}$ .

**Step 1.** For each  $p_k^i \in P_i$ , do the following steps:

- a. Find the subset  $P_{i+\lambda, k}$  of  $P_{i+\lambda}$  whose keypoints' spatial distance with  $p_k^i$  are less than  $r_k^i$ .
- b. Compute the similarity scores between  $p_k^i$  and every keypoint in  $P_{i+\lambda, k}$  using equation (1).
- c. Suppose  $p_h^{i+\lambda}$  has the largest similarity score with  $p_k^i$  in  $P_{i+\lambda, k}$ . If  $S(p_k^i, p_h^{i+\lambda}) > \delta_a$  where  $\delta_a$  is the minimum similarity threshold, then  $p_h^{i+\lambda}$  is the correspondence for  $p_k^i$  and add motion vector  $\overrightarrow{p_k^i p_h^{i+\lambda}}$  to  $MV_{i,i+\lambda}$ .
- d. Suppose the time interval between  $f_i$  and  $f_{i+\lambda}$  is  $t_i$ , then  $v_k^i = \left\| \overrightarrow{p_k^i p_h^{i+\lambda}} \right\| / t_i$ . Compute  $r_h^{i+\lambda}$  using equation (3) based on  $v_k^i$ .

**Step 2.** Adjust frame sampling interval  $\lambda$  using equation (4)

## B. Analyzing Camera Motion Pattern

When the set of motion vectors  $MV_{i,i+\lambda}$  of keypoints between  $f_i$  and  $f_{i+\lambda}$  are established, we classify the camera motion type between the two frames by a voting process into four categories: static, translation, zoom and rotation, and the category “translation” includes 8 sub-categories according to translation directions with boundaries  $\pi/8, 3\pi/8, 5\pi/8, 7\pi/8, 9\pi/8, 11\pi/8, 13\pi/8, 15\pi/8$ . Let  $M(i)$  denote the voting process between  $f_i$  and  $f_{i+\lambda}$ , and  $M(i)$  contains two main steps which are detailed below:

**Step 1.** Motion vectors vote for the camera motion type according to their directions.

Each motion vector is given a voting weight that is set to its starting point’s weight, i.e.

$$w(\overline{p_k^i p_h^{i+\lambda}}) = w(p_k^i) \quad (5)$$

$w(p_k^i)$  is set to a default value (0.5 in our experiments) when  $f_i$  is the initial frame or  $p_k^i$  has no correspondence on  $f_{i-\lambda}$ , otherwise it is set at the end of the voting process  $M(i-\lambda)$  which is described later in this section.

To more accurately estimate the global motion of the camera, we expect that the voting weights distribute evenly on a frame. But sometimes many keypoints aggregate in some regions of the frame, so that these regions may have large enough voting weights to dominate the voting result if every vector in  $MV_{i,i+\lambda}$  participates the voting. To avoid the case,

we divide a frame  $f_i$  into a set of 16 by 16 pixel blocks, and in each block only the motion vector with the largest voting weight is qualified to vote (Fig. 3). The motion vector with the largest voting weight is chosen to vote because it is more likely to be on background: at the end of the  $M(i-\lambda)$ , the voting weights for keypoints are adjusted so that in  $M(i)$ , the background motion vectors will have larger weights while the foreground ones will have small weights.

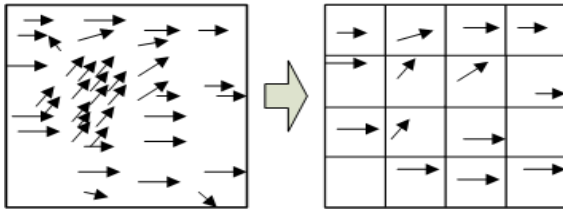


Fig. 3. Choosing representative motion vectors to vote. The rectangle represents the video frame and the arrows represent the motion vectors whose starting points are in this frame. The frame is divided evenly into blocks and only the motion vector with the largest voting weight in each block is qualified to vote.

Let  $V_t$  denote the total number of votes received by a camera motion type  $t$ . The representative motion vector  $v_m$

for block  $m$  votes for the camera motion type through the following procedure:

1. If  $|v_m| \leq \delta$  where  $\delta$  is a magnitude threshold (2 in our experiments), the weight  $w(v_m)$  is voted to  $V_{static}$ .

The reason for this is that the DoG local feature detector [13] may introduce in errors when computing the coordinates of the keypoints, which will attribute small motion vectors to static keypoints. So if the magnitude of a keypoint’s motion vector is very small, we consider it as static.

2. If  $|v_m| > \delta$ , let  $o^i$  denote the center of  $f_i$  and  $\theta$  denote the angle between  $v_m = \overline{p_k^i p_h^{i+\lambda}}$  and  $\overline{o^i p_k^i}$  (see Fig.4) and do the following two steps:

Firstly, vote for the translation type according to the direction of  $v_m$ .  $w(v_m)$  is accumulated into  $V_{translation}^i$ , where  $i$  is the direction index.

Secondly, if  $\theta \in (\pi/2 - \varepsilon, \pi/2 + \varepsilon)$ ,  $w(v_m)$  is voted to  $V_{rotation}$ ; if  $\theta \in (-\varepsilon, \varepsilon)$  or  $\theta \in (\pi - \varepsilon, \pi + \varepsilon)$ ,  $w(v_m)$  is voted to  $V_{zoom}$ . The reason is: if the camera is in rotation and the focus is at  $o^i$ , then  $v_m$  should be perpendicular to  $\overline{o^i p_k^i}$ ; if the camera is zooming in (or out) and the focus is at  $o^i$ ,  $v_m$  should have the same (or opposite) direction with  $\overline{o^i p_k^i}$ . Sometimes the focus of camera rotation or zooming is biased from  $o^i$ , so we used  $\varepsilon$  as a bias tolerance value (0.6 in our experiments) to deal with such cases.

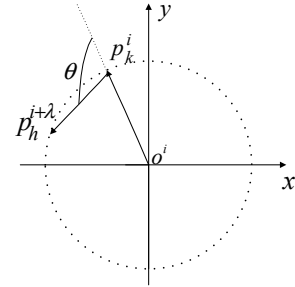


Fig. 4. Angle between  $v_m = \overline{p_k^i p_h^{i+\lambda}}$  and  $\overline{o^i p_k^i}$ .  $o^i$  is the center of the frame  $f_i$ . If  $\theta$  is  $\pi/2$ ,  $v_m$  is tangent to the circle around  $o^i$  so it should vote for rotation. If  $\theta$  is 0 (or  $\pi$ ),  $v_m$  is in accord with (or against with)  $\overline{o^i p_k^i}$  so it should vote for zooming.

**Step 2.** Decide the camera motion type according to the voting result, and then separate the foreground motion vectors from the background ones.

After all blocks have voted, the camera motion type with the most votes, denoted by  $\hat{V}$ , is chosen as the camera motion type. Let  $V$  denote the total votes, so the voting confidence  $\omega_i$  for this voting result is defined as  $\omega_i = \hat{V} / V$ . When abrupt

video content changes occur, such as flashing or occlusion,  $\omega_i$  may be too small to declare a valid voting result. Thus, If  $\omega_i < \delta_{conf}$  ( $\delta_{conf}$  is empirically set to 0.25 in our experiments), we discard this voting result and do  $M(i)$  again with  $MV_{i,i+\lambda+1}$  between  $f_i$  and  $f_{i+\lambda+1}$ .

Once the camera motion type is determined, each motion vector in  $MV_{i,i+\lambda}$  is classified as background if it is in accord with the camera motion type, or foreground otherwise. Denote the subset of background motion vectors in  $MV_{i,i+\lambda}$  as  $MV_{i,i+\lambda}^b$  and the foreground subset as  $MV_{i,i+\lambda}^f$ , and we can set the weight for the endpoint of each motion vector by

$$w(p_h^{i+\lambda}) = \begin{cases} w(p_k^i) + \Delta, & \overline{p_k^i p_h^{i+\lambda}} \in MV_{i,i+\lambda}^b \\ \max(0, w(p_k^i) - \Delta), & \overline{p_k^i p_h^{i+\lambda}} \in MV_{i,i+\lambda}^f \end{cases}, \quad (6)$$

where the increment (or decrement) amount  $\Delta$  is set according to the voting confidence:

$$\Delta = \omega_i * 0.5. \quad (7)$$

The evaluation in equation (6) increases the voting weights of background keypoints on frame  $f_{i+\lambda}$ , and reduces the voting weights of foreground ones. Thus the influence of background motion is promoted while that of foreground motion is reduced for the next round  $M(i+\lambda)$ , so that the voting result in  $M(i+\lambda)$  can more accurately reflect the real camera motion type.

### C. Computing Parametric Model

In many applications such as foreground object motion analysis and event detection, we need to analyze the real motion of foreground objects. Since the motion features of foreground objects can be reflected by the keypoints distributed on them, we can extract foreground object motion information by analyzing foreground keypoints' motions. To estimate the real motion of foreground keypoints, it is necessary to first compute their motion components caused by the camera motion, which need a parametric camera motion model. So apart from qualitatively classify the camera motion type, we also compute a parametric camera motion model. Furthermore, since we have separated the background motion vectors from the foreground ones through the voting process, we can use only the background motion vectors to estimate camera motion model, which can improve the model accuracy.

There are many parametric camera motion models (e.g. 4-parameter model, 6-parameter model and 9-parameter model) and they can all be estimated using the extracted background keypoint motion vectors. For the tradeoff between simplicity and accuracy, here we illustrate how to estimate a 4 parameter camera motion model using the background keypoint motion vectors. The camera motion

between  $f_i$  and  $f_{i+\lambda}$  is approximated as an affine transform of keypoints' coordinates and is represented as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix}, \quad (8)$$

where  $(x, y)$  and  $(x', y')$  denote the original and new coordinate respectively. The four model parameters  $a$ ,  $b$ ,  $c$  and  $d$  can be estimated using Least Square Minimization by the motion vectors of background keypoints. By subtracting the camera motion component from the motion vectors of foreground keypoints, we can estimate their actual motion vectors, and then the motion characteristics of foreground keypoints, such as velocity, direction, acceleration and angular velocity, can be computed to extract foreground object motion information.

## III. EVALUATION EXPERIMENTS

We performed two evaluation experiments to verify the effectiveness of our CMA approach on both ordinary videos and videos with significant camera and object motion.

To evaluate our CMA's effectiveness on ordinary videos, the first experiment is carried out on the public dataset provided by TRECVID 2005 for the low level feature (camera motion) extraction task. This dataset mostly consists of news videos and contains 2226 video shots manually labeled as containing different camera motion types including pan, tilt and zoom. We chose the same evaluation measures: precision and recall, as TRECVID 2005 does, with the following definitions:

$$precision = \frac{tp}{fp+tp}, \quad recall = \frac{tp}{fn+tp}, \quad (9)$$

where  $tp$  is the number of shots with camera motion types correctly declared,  $fp$  is the number of shots wrongly declared and  $fn$  is the number of shots mistaken for other types. The experimental results on the dataset are summarized in Table I. It shows that our results are comparable to the best results reported in TRECVID [14] and the recalls of pan and tilt generated by our algorithm are higher than the best reported results (88% and 78% respectively in [14]). So our approach can work effectively on ordinary videos.

The other experiment is carried out on a dataset consisting of 500 video shots chosen from seven action movies including "Kill Bill", "The Terminator", "The Rock", etc. The camera motion types in each shot are manually labeled by us as static, pan, tilt, or zoom. Most of them contain significant camera motions and/or foreground motions and some contain irregular light changes such as light flashing. The experimental results are summarized in Table II which proves that our approach is effective on videos with significant motion. Some of the results are even better than those on the TRECVID dataset. A possible explanation is that our approach is more sensitive to significant camera motion, since we observed that many shots with subtle camera motion in the TRECVID dataset are declared as static by our

approach, and that is why a large difference in recall can be seen from Table I and Table II.

TABLE I  
EXPERIMENTAL RESULTS ON TRECVID 2005 DATASET

Type	Precision	Recall
<i>Pan</i>	91.3%	90.4%
<i>Tilt</i>	88.2%	83.2%
<i>Zoom</i>	97.2%	65.5%

TABLE II  
EXPERIMENTAL RESULTS ON ACTION MOVIE VIDEO SHOTS

Type	Precision	Recall
<i>Pan</i>	93.6%	98.5%
<i>Tilt</i>	92.9%	97.3%
<i>Zoom</i>	92.3%	87.8%
<i>Static</i>	97.8%	88.1%

#### IV. CONCLUSION AND FUTURE WORK

In this paper, we present an effective camera motion analysis approach for video shots. The experimental results show that our algorithm is comparable to the state-of-the-art CMA methods and especially works effectively on shots from action movies which contain more significant camera motions and object motions. This work can provide a prerequisite for the complete motion analysis of video shots including both camera and object motion. Thus, our future work includes the analysis of the foreground object motions in video shots and further application of the motion information in understanding the semantics of video content, for instance, such as violent video detection, etc.

#### ACKNOWLEDGMENT

This work was supported in part by National Key Technologies R&D Program under Grant 2006BAH02A24-2 and by National Natural Science Foundation of China under Grant 60873087.

#### REFERENCES

[1] P. Bouthemy, M. Gelgon, and F. Ganansia, "A unified approach to shot change detection and camera motion characterization", in *IEEE Transaction on Circuits System and Video Technology*, vol. 9, no. 7, pp. 1030-1044, Oct. 1999.

[2] M. V. Srinivasan, S. Venkatesh and R. Hosie, "Qualitative estimation of camera motion parameters from video sequences", in *Pattern Recognition*, vol. 30, no. 4, pp. 593-606, Apr. 1997.

[3] W. Xiong and J. C. M. Lee, "Efficient scene change detection and camera motion annotation for video classification", in *Computer Vision and Image Understanding*, vol. 71, no. 2, pp. 166-181, Aug. 1998.

[4] R. Wang and T. Huang, "Fast Camera Motion Analysis in MPEG Domain", in *Proceedings of IEEE International Conference on Image Processing*, Kobe, Japan, October 24-28, 1999, vol.3, pp. 691-694.

[5] Jae-Gon Kim, Hyun Sung Chang, Jinwoong Kim, Hyung-Myung Kim, "Efficient Camera Motion Characterization for MPEG Video Indexing", in *Proceedings of IEEE International Conference on Multimedia and Expo*, New York, USA, July 30-August 2, 2000, vol. 2, pp. 1171-1174.

[6] Sangkeun Lee, Monson H. Hayes, III, "Real-time camera motion classification for content-based indexing and retrieval using templates", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Orlando, Florida, May 13 - 17, 2002, vol 4, pp. 3664-3667.

[7] Ralph Ewerth, Martin Schwalb, Paul Tessmann and Bernd Freisleben, "Estimation of Arbitrary Camera Motion in MPEG Videos", in *Proceedings of the 17th International Conference on Pattern Recognition*, Cambridge, UK, August 23-26, 2004, vol. 2, pp. 512-515.

[8] Fabricio Tiburzi and Jesús Bescós, "Camera Motion Analysis in On-line MPEG Sequences", in *Proceedings of Eighth International Workshop on Image Analysis for Multimedia Interactive Services*, Santorini, Greece, June 6-8, pp. 42-42, 2007.

[9] Chong-Wah Ngo, Ting-Chuen Pong and Hong-Jiang Zhang, "Motion Analysis and Segmentation Through Spatio-Temporal Slices Processing", in *IEEE Transactions on Image Processing*, vol. 12, issue 3, pp. 341 - 355, March 2003.

[10] T. Lertrudachakul, T. Aoki and H. Yasuda, "Camera Motion Estimation by Image Feature Analysis", in *Proceedings of Third International Conference on Advances in Pattern Recognition*, Bath, UK, August 22-25, 2005, pp. 618-625.

[11] C. Schmid, R. Mohr and C. Bauckhage, "Evaluation of interestpoint detectors", in *International Journal of Computer Vision*, vol. 37, issue 2, pp. 151-172, 2000.

[12] S. Battiato, G. Gallo, G. Puglisi and S. Scellato, "SIFT Features Tracking for Video Stabilization", in *Proceedings of International Conference on Image Analysis and Processing*, Modena, Italy, Sep. 10-14, 2007, pp. 825-830.

[13] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", in *International Journal on Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[14] Wessel Kraaij, Tzveta Ianeva, "TRECVID-2005 Low-level (camera motion) feature task", at <http://www-nlpir.nist.gov/projects/tvpubs/tv5.papers/tv5.llf.slides.final.pdf>.