Ma SG, Wang WQ. Effectively discriminating fighting shots in action movies. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 26(1): 187–194 Jan. 2011. DOI 10.1007/s11390-011-1121-z

Effectively Discriminating Fighting Shots in Action Movies

Shu-Gao Ma[1,2] (马述高) and Wei-Qiang Wang[1,3,*] (王伟强), *Member, ACM, IEEE*

[1]*School of Information Science and Engineering, Graduate University of Chinese Academy of Sciences Beijing 100049, China*

[2]*Computer Science Department, Boston University, Boston MA 02215, U.S.A.*

[3]*Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences Beijing 100190, China*

E-mail: shugaoma@bu.edu; wqwang@ict.ac.cn

Received September 30, 2009; revised November 23, 2010.

Abstract Fighting shots are the highlights of action movies and an effective approach to discriminating fighting shots is very useful for many applications, such as movie trailer construction, movie content filtering, and movie content retrieval. In this paper, we present a novel method for this task. Our approach first extracts the reliable motion information of local invariant features through a robust keypoint tracking computation; then foreground keypoints are distinguished from background keypoints by a sophisticated voting process; further, the parameters of the camera motion model is computed based on the motion information of background keypoints, and this model is then used as a reference to compute the actual motion of foreground keypoints; finally, the corresponding feature vectors are extracted to characterizing the motions of foreground keypoints, and a support vector machine (SVM) classifier is trained based on the extracted feature vectors to discriminate fighting shots. Experimental results on representative action movies show our approach is very effective.

Keywords video analysis, motion, concept learning

1 Introduction

Among thousands of movies per year produced by the movie industry, action movies are always a category of the most popular ones. Generally fighting shots are the highlights of action movies, and many interesting applications can be created if fighting shots in them can be identified. For example, fighting shots extracted can be used to construct a movie trailer of an action movie for advertisement or as a movie preview for quick browsing. Another potential and important application of discriminating fighting shots is movie content filtering, to prevent violent fighting scenes from bringing bad influence on children and adolescents. Moreover, the technique of fighting shot identification can also be applied in movie content indexing and retrieval.

To our knowledge, the research works on discriminating fighting shots are relatively few. Among the related works, the concept of movie "tempo" has been widely used to discover action scenes in movies by some of the works, in which the tempo is characterized by the rhythm of shot changes or the pace of motions within a shot. For instance, Adams *et al.*[1] used the tempo to detect dramatic story sections and events in a movie, where the tempo was measured by the shot length and the average magnitude of camera motion within a shot. Based on [1], Liu *et al.*[2] further introduced the motion intensity and the motion complexity to measuring tempo, and the motion intensity and the motion complexity were defined as the average of magnitudes of MPEG motion vectors and direction entropies in a shot respectively. Then they detected action scenes in movies by thresholding the tempo. Similarly, Chen *et al.*[3] detected high-tempo shots to produce the movie trailer and preview for action movies, where they measured the tempo through shot length, audio effects and motion intensity. More works such as [4-6] also exploited the movie tempo to detect action scenes in movies. All the approaches mentioned above do not distinguish foreground motions from background motions in characterizing motions in a shot, though foreground motion is generally a more important clue in discriminating fighting shots. Many of them utilized the rule-based way (using thresholds). Generally the

Short Paper

This work was supported in part by the National High Technology Research and Development 863 Program of China under Grant No. 2006BAH02A24-2 and by the National Natural Science Foundation of China under Grant No. 60873087.

*Corresponding Author

construction of effective rules and the choice of suitable thresholds are not an easy task for discriminating fight shots, due to various complex actions in fighting, as well as other various fast non-fighting motions, which are difficult to model them by fixed templates. More complex motion analysis techniques have been investigated and applied in discriminating fighting scenes. Datta *et al.*[7] detected human fighting in videos through analyzing the motion trajectory and orientation of person's limbs. Just as the authors themselves pointed out, their approach will malfunction when more than two people are fighting, since over-crowding foreground objects and frequent occlusions make the tracking of human bodies become quite difficult. Mecocci and Micheli[8] grouped foreground areas into several regions of interest (ROI) and then further divided each ROI into several color stains according to the color of foreground areas. The relative movements of the color stains within each ROI were estimated and an ROI was considered as containing violent activities if this relative movement exceeded a certain threshold during a specific time interval. The approach in [8] is basically applicable for analyzing surveillance videos where background keeps static, and it also assumed that foreground objects all lie on the same planar floor, which may not hold for movies. Wang *et al.*[9] extracted structure tensor histograms from video shots to train an Adaboost classifier for shot classification, and one of their predefined shot types was "fighting". Although the structure tensor histogram can reflect motion patterns in videos, foreground motion and background motion are not separated within this representation yet.

In this paper, we propose a novel approach to discriminating fighting shots in action movies. To robustly extracting motion information when both cameras and foreground objects move significantly, our approach first uses a robust keypoint tracking technique to reliably estimate the motion of local invariant features. Then, foreground keypoints are identified through a sophisticated voting process. Further, the parameters of the camera motion model is computed based on the motion information of background keypoints, and this model is then used as a reference to compute the actual movements of foreground keypoints. Finally, the velocity, moving direction, acceleration and angular velocity of foreground keypoints are evaluated to form foreground motion feature vectors and an SVM classifier is trained and used to discriminate fighting shots. Compared with the previous approaches, some features of our approach are summarized as follows: 1) foreground motions are separated from background motions, which makes our approach can robustly deal with videos with significant camera or foreground motions; 2) more complex features, acceleration and angular velocity are introduced to better characterize motion patterns of foreground objects, and our experiments have shown the usefulness of such information; 3) a machine learning way is exploited to avoid the difficulty of constructing effective rules for various movie contents.

The rest of this paper is organized as follows. Section 2 presents our approach in details. The experimental results are reported in Section 3. Section 4 concludes the paper.

2 Our Approach

We describe our approach in details in this section and organize the related information as follows. First, Subsection 2.1 introduces how to effectively and efficiently matching keypoints in consecutive frame samples to extract the motion information of keypoints between them; then, we introduce the sophisticated voting process to identify camera motions by construct the distribution of motion types, and the estimation of the camera motion model in Subsection 2.2. Further, Subsection 2.3 describes the related technique to robustly compute foreground motion feature vectors for every consecutively sampled frame pair, as well as how to characterize the motions of foreground objects. Finally, Subsection 2.4 briefly introduce the procedure of training an SVM classifier based on foreground motion feature vectors in a shot to discriminate fighting shots.

2.1 Tracking Keypoints

Our approach first extracts keypoints from sampled frames and uses the SIFT descriptors to represent them as [10], and then matches them between consecutive frame samples to track their movements. In Section 2, we use p to denote a keypoint on the current frame f, and use p^*, p' to denote its correspondences on the previous frame f^* and the next frame f' respectively, if its correspondences have successfully been found on frame f^* or/and frame f' , as shown in Fig.1. λ denotes the frame sampling interval.

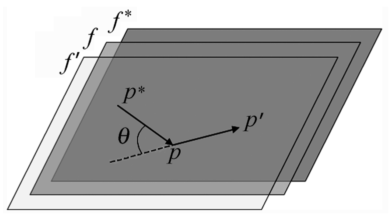

Fig.1. Corresponding keypoints on consecutive frames.

To promote the matching accuracy and speed, we assume 1) the spatial distance between a key-point p and its correspondence p' is always within a specified range r, if p' exists, and 2) the motions of keypoints (speed and direction) always change smoothly. So, for a keypoint p, our algorithm searches a circular region with radius r around p for its correspondence p' on frame f', as shown in Fig.2. Compared with [11] which searches correspondences in the whole frame, our approach has two advantages. First, it promotes computation efficiency by avoiding searching unrelated areas. Second, it can enhance matching accuracy by filtering out similar but distant keypoints, which are common in natural scenes (e.g., repeated textures) and often result in false matches.

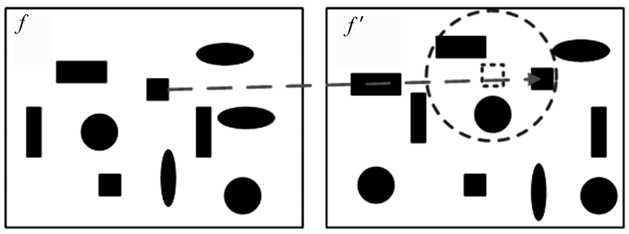

Fig.2. Searching correspondence in a circular region around keypoints.

Our system computes the matching score $S(p,p')$ between two keypoints p and p' based on both the appearance similarity of SIFT features $S_a(p,p')$ and the motion smoothness $S_m(p^*,p,p')$ by,

$$S(p,p') = w_a S_a(p,p') + w_m S_m(p^*,p,p'), \tag{1}$$

where w_a and w_m are the corresponding weights, and $w_a = 0.7$, $w_m = 0.3$ in our system. The measure of motion smoothness $S_m(p^*,p,p')$ is defined by

$$S_m(p^*,p,p') = 1 - \frac{\theta(\boldsymbol{p^*p}, \boldsymbol{pp'})}{\pi}, \tag{2}$$

where $\theta(\boldsymbol{p^*p}, \boldsymbol{pp'})$ denotes the included angle between motion directions $\boldsymbol{p^*p}$ and $\boldsymbol{pp'}$. For example, as shown in Fig.1, θ is the included angle between motion directions of keypoints p^* and p. If the correspondence p^* of p on frame f^* is not available, our system computes the matching score $S(p,p')$ just based on the appearance similarity of SIFT features, i.e., $S(p,p') = S_a(p,p')$. Although $S_m(p^*,p,p')$ only measures the change of motion direction and neglects the change of velocity, we discover that it has worked well enough. As the correspondence of p, only if keypoint p' has the highest matching score with p among all the keypoints in the search circular region of p, and at the same time this score exceeds a predefined threshold, the corresponding motion vector $\boldsymbol{pp'}$ is established.

If a keypoint moves fast, we need to check a relatively large region to search for its correspondence on the frame; if it is static or moves slowly, we only need to check a small region to avoid unnecessary search. Thus, for a keypoint p on the current frame, our algorithm dynamically adjusts the search range r of its correspondence p' on the next frame based on the velocity v^* of its correspondence p^* on the previous frame. If v^* is not available, since p has no correspondence on the previous frame f^*, or the current frame is an initial frame, a default value $0.2 \times l$ is chosen for r, where l denotes the length of diagonal of a video frame. Otherwise,

$$r = \max(0.1 \times l, 1.5 \times v^*). \tag{3}$$

Our system also adjusts the frame sampling interval dynamically. When the motion is insignificant, a larger λ is used to promote processing speed, otherwise a smaller λ is used to guarantee matching accuracy. Specifically, the adjustment is based on the average velocity $\bar{v}$ of keypoints on a frame, i.e.,

$$\lambda = \begin{cases} \min\left(\left\lceil \frac{\tau}{3} \right\rceil, \left\lceil \frac{\lambda_0 \Theta_l}{\bar{v}} \right\rceil\right), & \bar{v} \leqslant \Theta_l, \\ \lambda_0, & \Theta_l < \bar{v} < \Theta_h, \\ \max\left(1, \left\lceil \frac{\lambda_0 \Theta_h}{\bar{v}} \right\rceil\right), & \bar{v} \geqslant \Theta_h, \end{cases} \tag{4}$$

where λ_0 denotes the initial frame sampling interval ($\lambda_0 = 3$ in our system), τ denotes the frame rate of video, and the two speed thresholds are chosen as $\Theta_l = 0.02 \times l$, $\Theta_h = 0.152 \times l$ in our system.

2.2 Analyzing Camera Motion

Once we have found the correspondences of keypoints on consecutive frames, we can further compute and classify camera motions into different categories. In our work, camera motions include four main categories, i.e., static, translation, rotation, and zoom, and the category "translation" further includes eight subcategories based on different quantized translation directions with boundaries $\frac{(2k+1)\pi}{8}$, $k = 0, 1, \ldots, 7$, as shown in Fig.3. The camera motion category from the

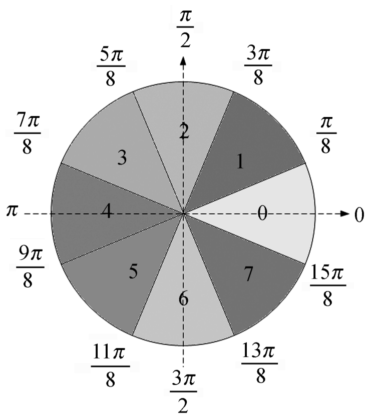

Fig.3. Different quantized translation directions.

current frame f to its next frame f' is determined by the result of a voting process. Once the camera motion category is identified, the keypoints which have consist motion with cameras form a set Ω_B of background keypoints, and otherwise a set Ω_F of foreground keypoints.

For any keypoint p on the current frame f, a voting weight $w(p)$ is associated with it, and the value of $w(p)$ depends on whether its correspondence p^* on previous frame f^* exists, as well as whether p^* is a background keypoint or not. If p^* does not exist or frame f is an initial frame, we choose a default value for $w(p)$ ($w(p) = 0.5$ in our experiments). Otherwise, we compute the weight $w(p)$ by

$$w(p) = \begin{cases} w(p^*) + \Delta, & p^* \in \Omega_B \\ \max(0, w(p^*) - \Delta), & p^* \in \Omega_F \end{cases} \tag{5}$$

where $\Delta = 0.5 \times \omega$, and ω is another parameter called voting confidence, which indirectly reflects the confidence of claiming p as a background keypoint, and the related computation of ω is defined in the later description. Apparently, the evaluation in (5) can increase the voting weights of background keypoints, and reduce the voting weights of foreground ones, since background motion is correlated more strongly with camera motions, and we expect that background keypoints should play more significant role in estimating real camera motion.

To estimate more accurately the global motion of a camera, we expect that the voting weights distribute evenly on a frame. But sometimes many keypoints aggregating in small regions may have large enough voting weights to dominate the voting result. To avoid the case, we divide frame f into a set of M by N pixel blocks. For the i-th pixel block, only the keypoint p_s in it which has the largest voting weight and its correspondence p'_s exists in frame f' is chosen as the representative to participate the vote. As shown in Fig.4, a collection of motion vectors is reduced into a few representative motion vectors.

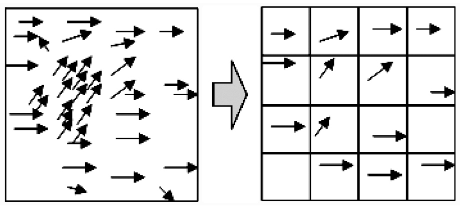

Fig.4. Generation of representative vectors.

Let $\boldsymbol{m}_s$, $s = 1, 2, \ldots, MN$, denote the motion vector associated with representative keypoint p_s and its correspondence p'_s on frame f', i.e., $\boldsymbol{m}_s = \boldsymbol{p}_s\boldsymbol{p}'_s$, O denote the center of frame f, θ_s denote the included angle from $\boldsymbol{O}_s\boldsymbol{p}_s$ to $\boldsymbol{m}_s$ (in Fig.5), and h_t denote the number of votes for the camera motion type t. The corresponding voting algorithm for determining camera motion categories is summarized in Algorithm 1. Algorithm 1 assumes that background regions cover most part of a frame, so most representative vectors will vote for the bin corresponding to the motion type of camera. If the motion magnitude of a representative keypoint is very small, i.e., $|\boldsymbol{m}_s| < \delta$, the keypoint only votes for the bin h_{static} denoting static. Otherwise, the keypoint always votes for one of the bins h^j_{tran}, $j = 0, 1, \ldots, 7$ denoting translation, and the specific bin depends on the direction of $\boldsymbol{m}_s$. At the same time, the keypoint is also permitted to vote for the bin h_{rotation} denoting rotation motion or the bin h_{zoom} denoting zoom, if the corresponding criterion is satisfied. At the end of the algorithm, the bin indexed by $\hat{t}$ which gains the most votes is identified and it indicates the camera motion type. Additionally, the confidence $\omega = h_{\hat{t}}/\sum_t h_t$ used in (5) is computed here, and that means that the more representatives vote for it, the more possible the final resolution is correct.

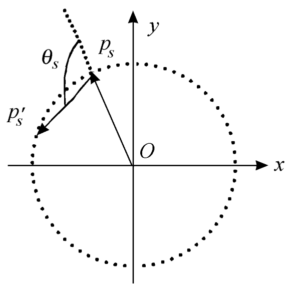

Fig.5. Definition of θ_s.

Algorithm 1. Classify Camera Motions by Voting

```
Input: $\boldsymbol{m}_s$, $s = 1, 2, \ldots, MN$; $\delta$, a threshold parameter; $\epsilon$, a bias tolerance parameter;
Output: camera motion type $\hat{t}$; corresponding decision confidence $\omega$;
For ($s = 1, 2, \ldots, MN$)
{
  if ($|\boldsymbol{m}_s| < \delta$)
  {
    the weight $w(p_s)$ is voted to $h_{\text{static}}$;
    continue;
  };
  Compute the direction of $\boldsymbol{m}_s$, quantize it and then the weight $w(p_s)$ is voted to $h^j_{\text{tran}}$, where $j$ denotes the corresponding translation direction;
  if ($\theta_s \in [\frac{\pi}{2} - \epsilon, \frac{\pi}{2} - \epsilon]$)
  {
    the weight $w(p_s)$ is voted to $h_{\text{rotation}}$;
```

```
            **continue**;
        };
        **if** ($\theta_s \in [-\epsilon, \epsilon]$ or $\theta_s \in [\pi - \epsilon, \pi + \epsilon]$)
        {
            the weight $w(p_s)$ is voted to $h_{\text{zoom}}$;
        };
    }
    $\hat{t} = \arg\max_t \{h_t\}$;
    $\omega = \frac{h_{\hat{t}}}{\sum_t h_t}$;
```

In some special scenario where abrupt video content changes occur, such as flashing or occlusion, ω may be too small to declare a valid voting result. Thus, if $\omega < \eta$, we do not give the judgement of camera motion categories from frame f to frame f', and just take f' as an initial frame in the next round of computation. In our system, $M = N = 16$, $\delta = 2$, $\epsilon = 0.6$, and $\eta = 0.25$. Once the camera motion type from frame f to frame f' is determined, each existing motion vector from frame f to frame f' is classified as background if it is in accord with the camera motion type, or foreground otherwise. Further, we can use (5) to compute the weights of endpoint of each motion vector on frame f'. Finally, frame f' becomes a current frame, and the system starts to analyze camera motion from frame f' to its next sampled frame.

For the tradeoff between simplicity and accuracy, we approximate the camera motion from frame f to frame f' as an affine transform of coordinates of keypoints, and represent it as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix} \tag{6}$$

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.

2.3 Extracting Foreground Motion Feature

In our application, fighting as a kind of motion of foreground objects is always associated with the motion of foreground keypoints. So it is an significant part to robustly identify foreground keypoints and define an effective feature to characterize motions of foreground keypoints.

Through the camera motion analysis process in Subsection 2.2, keypoints p on frame f are classified into two categories, i.e., background and foreground, and it is the same for the corresponding motion vectors, if their correspondences p' are found on frame f'. In the above computation, two wrong cases may occur: (a) the wrong correspondence is found on frame f' for p, which results in a wrong estimation of motion vectors; (b) some foreground keypoints are wrongly classified as background due to they have similar motion direction as cameras. To robustly identify foreground keypoints and more accurately estimate the foreground motion vectors, the corresponding computation is performed to deal with the problems. First, for each background motion vector $\boldsymbol{pp'}$, the bias is computed through comparing p' against the estimated location from the camera motion model defined by (6). If this bias is larger than a threshold, the keypoint p is revised as a foreground keypoint. Additionally, let $\tilde{p}'$ denote the keypoint which has the second highest matching score with p on frame f', and p has the highest matching score. The motion vector $\boldsymbol{pp'}$ will not be used in extracting foreground motion features, if the following condition is not satisfied, i.e.,

$$\frac{S(p, p')}{S(p, \tilde{p}')} > \zeta \tag{7}$$

where ζ is a threshold. Lowe[10] has shown that (8) can effectively filter out wrongly matched keypoint pairs.

After the above processing, we assume that an accurate correspondence p' has been found on frame f' for a foreground keypoint p. In fact, the motion $\boldsymbol{pp'}$ of foreground keypoints consists of two parts, camera motion and their actual motion relative to camera. Only actual motion vectors of foreground keypoints contain useful information in representing the motion of foreground objects. Thus, we first compute the location of point p due to camera motion using (5), and let $\dot{p}'$ denote it. Then the actual motion of a foreground keypoint p is represented by vector $\tilde{\boldsymbol{m}}(p)$, and $\tilde{\boldsymbol{m}}(p)$ is computed by

$$\tilde{\boldsymbol{m}}(p) = \boldsymbol{pp'} - \boldsymbol{p\dot{p}'}. \tag{8}$$

Based on the set of actual motion vectors of foreground keypoints p on frame f, i.e., $\{\tilde{\boldsymbol{m}}(p)\}$, as well as the set of actual motion vectors of foreground keypoints p^* on frame f^*, i.e., $\{\tilde{\boldsymbol{m}}(p^*)\}$, we can extract foreground motion feature, including velocity, motion direction, angular velocity and acceleration, etc. The foreground motion feature vector of current frame f consists of velocity-direction histogram, angular velocity histogram, acceleration histogram, static-to-motion ratio, motion-to-static ratio, continuity ratio. Let n_f denote the number of foreground keypoints on frame f whose correspondence can be found on frame f', and n_b denote the number of foreground keypoints on frame f whose correspondence can be found on both frame f' and frame f^*. The concrete definitions of different feature components are described as follows.

• *Velocity-Direction Histogram.* It is a two-dimensional histogram, and the velocity of foreground keypoint p is computed by $\tilde{\boldsymbol{m}}(p)/\lambda$, where the time interval λ use $\frac{1}{8}$ second as the unit time. The velocity is quantized into 8 bins. The first 7 bins have the same bin width $\frac{l}{35}$ and the 8th bin corresponds the velocity larger than $\frac{l}{5}$ (definition of l see (3)). The motion direction range $[-\frac{\pi}{8}, \frac{15\pi}{8})$ is evenly quantized into 8 bins, as shown in Fig.3. The velocity-direction histogram is finally normalized by n_f.

• *Angular Velocity Histogram.* Only when keypoint p has the correspondence p^* on frame f^*, it is used to compute the angular velocity histogram. It is also true for the remaining features. Since the change of motion directions depends on the included angle θ between $\tilde{\boldsymbol{m}}(p)$ and $\tilde{\boldsymbol{m}}(p^*)$, and $\theta = \arccos(\frac{\tilde{\boldsymbol{m}}(p)\tilde{\boldsymbol{m}}(p^*)}{|\tilde{\boldsymbol{m}}(p)||\tilde{\boldsymbol{m}}(p^*)|})$. The angular velocity is computed by θ/λ. The angular velocity is quantized into 9 bins, where the first bin corresponds to linear motion ($\theta \approx 0$) and the bin width of the remaining 8 bins is $\frac{\pi}{8}$). The angular velocity histogram is finally normalized by n_b.

• *Acceleration Histogram.* The acceleration of keypoint p is computed by $(|\tilde{\boldsymbol{m}}(p)| - |\tilde{\boldsymbol{m}}(p^*)|)/\lambda$. The acceleration is quantized into 17 bins. Constant motion is mapped to the first bin, positive accelerations are mapped to the 2nd$\sim$9th bins, and negative accelerations are mapped to the 10th$\sim$17th bins. The bin width of 2nd$\sim$8th bins and 10th$\sim$16th bins is $l/35$. The acceleration larger than $l/5$ is mapped to the 9th bin, if it is positive, or the 17th bin, if negative. The acceleration velocity histogram is finally normalized by n_b.

• *Static-to-Motion Ratio, and Motion-to-Static Ratio.* Through a small threshold μ, we can define two kinds of keypoints. Concretely, if $|\tilde{\boldsymbol{m}}(p)| > \mu$ and $|\tilde{\boldsymbol{m}}(p^*)| \leqslant \mu$, keypoint p is called a static-to-motion keypoint. On the contrary, if $|\tilde{\boldsymbol{m}}(p)| \leqslant \mu$ and $|\tilde{\boldsymbol{m}}(p^*)| > \mu$, keypoint p is called a motion-to-static keypoint. The static-to-motion ratio is defined as the ratio of the number of static-to-motion keypoints to n_b, and the motion-to-static ratio is defined as the ratio of the number of motion-to-static keypoints to n_b. In our system, we choose $\mu = 2$.

• *Continuity Ratio.* The continuity ratio is defined as n_b/n_f.

2.4 Discriminating Fighting Shots

For each video shot, the foreground motion feature vector between every consecutively sampled frame pair can be extracted, and they form a set of feature vectors $\boldsymbol{\Phi}$. For different shots, the size of feature set $\boldsymbol{\Phi}$ may vary, so we compute the mean vector $\bar{\boldsymbol{\phi}}$ of all the feature vectors $\{\boldsymbol{\phi}|\boldsymbol{\phi} \in \boldsymbol{\Phi}\}$ as the foreground motion feature vector of the shot, i.e.,

$$\bar{\boldsymbol{\phi}} = \frac{1}{K}\sum_{i=1}^{K}\boldsymbol{\phi}_i, \quad \boldsymbol{\phi}_i \in \boldsymbol{\Phi}. \tag{9}$$

We label a collection of video shots as containing fighting event or not, and their foreground motion feature vectors are extracted to form the training set. We use the SVM to construct the classifier for discriminate fighting shots.

3 Evaluation Experiments

Our fighting shot discrimination approach is based on video motion analysis, so we first report the evaluation results on the effectiveness of our camera motion analysis method, and then give the evaluation results of fighting shot discrimination.

3.1 Camera Motion Analysis Evaluation

We perform two groups of evaluation experiments to verify the effectiveness of our camera motion analysis approach. To evaluate its 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 choose the same evaluation measures: precision and recall, as TRECVID 2005 does, with the following definitions:

$$\mathit{precision} = \frac{tp}{fp + tp}, \quad \mathit{recall} = \frac{tp}{fn + tp}, \tag{10}$$

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 1. It shows that our results are comparable to the best results reported in TRECVID[12] and the recalls of pan and tilt generated by our algorithm are higher than the best reported results (88% and 78% respectively in [12]). So our approach can work effectively on ordinary videos.

Table 1. Experimental Results on TRECVID 2005 Dataset

Motion Type	*Precision* (%)	*Recall* (%)
Pan	91.3	90.4
Tilt	88.2	83.2
Zoom	97.2	65.5

The other group of 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 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 2 which show the effectiveness of our approach 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 observe that many shots with subtle camera motion in the TRECVID dataset are declared as static by our approach, and that is why a difference in recall can be seen between Table 1 and Table 2. The sensitivity benefits from the adaptive adjustment of sampling interval λ by (4) which can make the subtle motion become more apparent and make significant motion become appropriate.

Table 2. Experimental Results on Action Movie Video Shots

Motion Type	*Precision* (%)	*Recall* (%)
Pan	93.6	98.5
Tilt	92.9	97.3
Zoom	92.3	87.8
Static	97.8	88.1

3.2 Fighting Shot Discrimination Evaluation

The evaluation dataset for fighting shot discrimination contains 523 video shots from four action movies which are manually labeled as containing fighting events or not. Table 3 details the number of shots in each category in the dataset.

Table 3. Evaluation Dataset

Motion Name	Fighting	Non-Fighting	Total
Flash Point	33	49	82
Dead and Alive	69	71	140
Fists of Fury	60	85	145
Kill Bill	76	80	156
Total	**238**	**285**	**523**

For every movie, we train an SVM classifier using the shots from the other three movies and then classify the shots of this movie. We use the LIBSVM[13] as the SVM training tool. The RBF kernel is chosen for the SVM classifier and the best parameters are estimated using cross validation. The result is evaluated by precision and recall for the classification of fighting shots, which is defined the same as in the previous subsection. For this evaluation, *tp* is the number of correctly classified fighting shots, *fp* is the number of shots wrongly classified as fighting shots and *fn* is the number of fighting shots that are not detected. We perform two groups of experiments: the first one uses only the velocity-direction histogram as the feature vector and the other uses all components of the feature vector described in Subsection 2.3. Table 4 shows the results from the first group of experiments and Table 5 shows the results from the second group of experiments.

Table 4 and Table 5 show that our method can effectively discriminate fighting shots in a movie. Through comparing Table 4 with Table 5, we can see that besides the velocity and direction, the angular velocity, the acceleration and the other extracted information are useful in helping promote the classification performance.

Table 4. Experimental Results Using Only Velocity-Direction Histogram

Motion Name	*Precision* (%)	*Recall* (%)
Flash Point	64.10	75.76
Dead and Alive	78.26	78.26
Fists of Fury	69.56	57.14
Kill Bill	71.43	75.00
Average	**71.85**	**71.85**

Table 5. Experimental Results Using All the Features

Motion Name	*Precision* (%)	*Recall* (%)
Flash Point	67.4410	87.88
Dead and Alive	78.82	97.10
Fists of Fury	70.00	70.00
Kill Bill	86.76	77.63
Average	**76.95**	**82.77**

4 Conclusion

An effective fighting shot discrimination approach is proposed in this paper. Our approach consists of two parts, robust camera motion analysis computation, and classification model based on motion of foreground objects. Our camera analysis computation can effectively deal with ordinary videos, and is especially good at analyzing the fast, significant camera motion in action movies, due to the trick of adaptive frame sampling in our approach. Through robustly identify foreground keypoints and characterizing motion of foreground objects by the velocity, moving direction, acceleration and angular velocity of foreground etc., we can effectively discriminating discriminate fighting shots. Since the audio signal can also provide useful information in discriminating fighting scenes, integrating audio cues in this task is one of our future work. Additionally, more efforts are required in exploring the proposed camera motion analysis algorithm to be applied in other computer vision tasks.

References

[1] Adams B, Dorai C, Venkatesh S. Novel approach to determining tempo and dramatic story sections in motion pictures. In *Proc. the 7th IEEE International Conference on Image Processing*, Vancouver, Canada, Sept. 10-13, 2000, pp.283-286.

[2] Liu A, Li J, Zhang Y, Tang S, Song Y, Yang Z. An innovative model of tempo and its application in action scene detection for movie analysis. In *Proc. the 9th IEEE Workshop on Applications of Computer Vision*, Copper Mountain, USA, Jan. 7-9, 2008, pp.1-6.

[3] Chen H W, Kuo J H, Chu W T, Wu J L. Action movies segmentation and summarization based on tempo analysis. In *Proc. the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval*, USA, Oct. 15-16, 2004, pp.251-258.

[4] Chen L, Ozsu M T. Rule-based scene extraction from video. In *Proc. the 9th IEEE International Conference on Image Processing*, Rochester, USA, Sept. 22-25, 2002, pp.737-740.

[5] Cheng W, Liu C, Xu D. An approach to action scene detection in martial arts movies. *Acta Electronica Sinica*, 2006, 34(5): 915-920.

[6] Gong Y, Wang W Q, Jiang S Q, Huang Q M, Gao W. Detecting violent scenes in movies by auditory and visual cues. In *Proc. the Pacific-Rim Conference on Multimedia*, Tainan, China, Dec. 9-13, 2008, pp.317-326.

[7] Datta A, Shah M, Lobo N D V. Person-on-person violence detection in video data. In *Proc. the 16th International Conference on Pattern Recognition*, Quebec, Canada, Aug. 11-15, 2002, pp.433-438.

[8] Mecocci A, Micheli F. Real-time automatic detection of violent-acts by low-level colour visual cues. In *Proc. the 14th IEEE International Conference on Image Processing*, San Antonio, USA, Sept. 16-19, 2007, pp.345-348.

[9] Wang S, Jiang S, Huang Q, Gao W. Shot classification for action movies based on motion characteristics. In *Proc. the 15th IEEE International Conference on Image Processing*, San Diego, USA, Oct. 12-15, 2008, pp.2508-2511.

[10] Lowe D G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004, 60(2): 91-110.

[11] Battiato S, Gallo G, Puglisi G, Scellato S. SIFT features tracking for video stabilization. In *Proc. the 14th International Conference on Image Analysis and Processing*, Modena, Italy, Sept. 10-14, 2007, pp.825-830.

[12] Kraaij W, Ianeva T. TRECVID-2005 low-level (camera motion) feature task. http://www.nlpir.nist.gov/projects/tvpubs/tv5.papers/tv5.llf.slides.final.pdf, 2005.

[13] Fan R, Chen P, Lin C. Working set selection using the second order information for training SVM. *Journal of Machine Learning Research*, 2005, 6(12): 1889-1918.

Shu-Gao Ma received his M.S. degree in computer science from Graduate University of Chinese Academy of Sciences in 2009. He is currently a Ph.D. candidate in Boston University, US. His current research interests include multimedia content analysis, computer vision.

Wei-Qiang Wang is a professor in School of Information Science and Engineering, Graduate University of Chinese Academy of Sciences, Beijing, China. He is a member of IEEE, ACM. He received his Ph.D. degree from Institute of Computing Technology, Chinese Academy of Sciences, in 2001. His current research interests include multimedia content analysis, computer vision and machine learning.