J. Vis. Commun. Image R. 23 (2012) 932–947

Contents lists available at SciVerse ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Recognizing tactic patterns in broadcast basketball video using player trajectory

Hua-Tsung Chen [a,*], Chien-Li Chou [b], Tsung-Sheng Fu [b], Suh-Yin Lee [b], Bao-Shuh P. Lin [a]

[a] Information and Communications Technology Lab, National Chiao Tung University, Hsinchu 300, Taiwan
[b] Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan

ARTICLE INFO

Article history:
Received 2 February 2012
Accepted 6 June 2012
Available online 15 June 2012

Keywords:
Sports video analysis
Object tracking
Camera calibration
Multimedia system
Tactic analysis
Kalman filter
Pattern recognition
Broadcast basketball video

ABSTRACT

The explosive growth of the sports fandom inspires much research on manifold sports video analyses and applications. The audience, sports fans, and even professionals require more than traditional highlight extraction or semantic summarization. Computer-assisted sports tactic analysis is inevitably in urging demand. Recognizing tactic patterns in broadcast basketball video is a challenging task due to its complicated scenes, varied camera motion, frequently occlusions between players, etc. In basketball games, the action *screen* means that an offensive player perform a blocking move via standing beside or behind a defender for freeing a teammate to shoot, to receive a pass, or to drive in for scoring. In this paper, we propose a screen-strategy recognition system capable of detecting and classifying screen patterns in basketball video. The proposed system automatically detects the court lines for camera calibration, tracks players, and discriminates the offensive/defensive team. Player trajectories are calibrated to the real-world court model for screen pattern recognition. Our experiments on broadcast basketball videos show promising results. Furthermore, the extracted player trajectories and the recognized screen patterns visualized on a court model indeed assist the coach/players or the fans in comprehending the tactics executed in basketball games informatively and efficiently.

1. Introduction

With the rapid advance in video production technology and the consumer demand, the proliferation of digital content necessitates the development of automatic systems and tools for semantic multimedia information analysis, understanding, and retrieval. As important multimedia content, sports video has been attracting considerable research efforts due to the commercial benefits, entertaining functionality, and the audience requirements. Generally, the breaks or commercials make the broadcast sports videos somewhat tedious. Most of the audiences would rather retrieve the events, scenes, and players of interest than watch a whole game in a sequential way. Therefore, a great amount of research focuses on shot classification, highlight extraction, event detection and semantic annotation. Traditional video content analysis for quick browsing, indexing, and summarization has become a basic requirement. More keenly than ever, the sports fans and professionals desire to watch a sports game not only with efficiency but also with variety, profundity, and professionalism. Especially, the coach and players prefer better understanding of the tactic patterns taken in a game. Hence, the research on computer-assisted sports tactic analysis starts rising and flourishing.

Research on sports video analysis pours out in the past decade, mainly focusing on feature extraction, structure analysis, and bridging the semantic gap between low-level features and high-level events. Duan et al. [1] employ a supervised learning scheme to perform a top-down shot classification based on mid-level representations, including motion vector field model, color tracking model and shot pace model. Tien et al. [2] present an automatic system capable of segmenting a basketball video into shots based on scene change detection. Then, shots are classified into three types: close-up view, medium view, and full court view, based on the ratio of the dominant color pixels in the frames. The shots of full court view are more informative than the other two and are chosen for content analysis. Hung and Hsieh [3] categorize shots into pitcher-catcher, infield, outfield, and non-field shots. Then, they combine the detected scoreboard information with the obtained shot types as mid-level cues to classify highlights using Bayesian Belief Network (BBN) structure. Fleischman et al. [4] use complex temporal features, such as object, field type, speech, camera motion start time and end time, etc. Temporal data mining techniques are exploited to discover a codebook of frequent temporal patterns for baseball highlight classification. Cheng and Hsu [5] fuse visual motion information with audio features, including zero crossing rate, pitch period, and Mel-frequency cepstral coefficients (MFCC), to extract baseball highlight based on hidden Markov model (HMM). Assfalg et al. [6] present a system for automatic annotation of highlights in soccer videos. Domain knowledge is encoded into a set of finite state machines, each of which models a specific highlight. The visual cues used for highlight detection are ball motion, playfield zone, players' positions, and colors of uniforms. Chu and Wu [7]

* Corresponding author. Fax: +886 3 573 6491.
E-mail address: huatsung@cs.nctu.edu.tw (H.-T. Chen).

1047-3203/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jvcir.2012.06.003

consider most of the possible conditions in a baseball game based on the game-specific rules and extract the scoreboard information for event detection. Zhu et al. [8] address two problems: affective feature extraction and ranking model construction to rank highlights in broadcast tennis video. The affective features are extracted via recognizing the player actions and analyzing the audience response. The highlight ranking approach combines the player actions with the real-world trajectories and audio keywords to establish the mid-level representation of video content, and support vector regression is employed to construct the nonlinear highlight ranking mode.

Recently, the emerging trend of sports video analysis goes from semantics to tactics [9]. Since significant events are mainly caused by ball-player and player-player interactions, ball/player trajectories reveal important information for tactic analysis and strategy inference. Zhu et al. [10–12] analyze the temporal-spatial interaction among the ball and players to construct a tactic representation called *aggregate trajectory* based on multiple trajectories. The interactive relationship with play region information and hypothesis testing for trajectory temporal-spatial distribution are exploited to analyze the tactic patterns. Wang and Parameswaran [13] use ball trajectory and landing position as features to classify tennis games into 58 winning patterns. Yu et al. [14–16] present a trajectory-based algorithm for ball detection and tracking in soccer video. The ball size is first estimated from feature objects (the goalmouth and ellipse) to detect ball candidates. Potential trajectories are generated from ball candidates by a Kalman filter based verification procedure. Camera motion recovery helps in obtaining better candidates and forming longer ball trajectories. The true ball trajectories are finally selected from the potential trajectories according to a confidence index, which indicates the likelihood that a potential trajectory is a ball trajectory. Our previous works [17–19] track the ball motion in different kinds of broadcast sports videos and provide manifold trajectory-based applications to meet the practical requirement and professionals' needs, such as pitching evaluation in baseball, shooting location estimation in basketball, virtual replay, and set type recognition in volleyball.

Increasing research works are devoted to basketball video analysis due to a large audience base and a variety of user-specific requirements. Liu et al. [20] propose a multiple-modality approach, which first extracts the visual, motion and audio information from basketball videos for low-level video segmentation and classification. Then, domain knowledge is utilized for detecting the events such as "foul" and "shot at the basket." Also with a multiple-modality method, Zhang et al. [21] detect more event types in broadcast basketball video via aligning the web-casting text with the video. Chang et al. [22] presents a basketball wide-open warning system, which tracks players in broadcast basketball videos and issue a warning when there is an offensive player not well defended by his/her opponents. To meet the sports-professional's requirement, Hu et al. [23] propose a robust camera calibration method for broadcast basketball video, extract payer trajectories by a CamShift-based tracking method, and mapping player trajectories to the real-world court model. The player position/trajectory information in the real world coordinates is further utilized for professional-oriented applications, including wide-open event detection, trajectory-based target clips retrieval, and tactic inference.

These previous works on basketball tactic analysis are admirable since they greatly assist the professionals in inferring the offensive/defensive strategy of the opponent team and inspecting their weaknesses. However, there are still a variety of tasks on tactics analysis and statistics collection requiring assistance with computer technology, such as offensive strategy analysis and *screen* pattern recognition. In basketball, *scoring* is the most important event that audiences are interested in and the coach/players are aspiring to. Since basketball is a five-man sport, it is hard for an individual player to break the defense of the opponent team and score by himself. Hence, the offensive team normally executes some tactical strategies to get rid of the defenders and seek an open chance to shoot the ball. Playing a fundamental and essential role in offensive tactics, *screen* is a blocking move performed by an offensive player standing beside or behind a defender, in order to free a teammate to shoot, to receive a pass, or to drive in for scoring. Fig. 1 exemplifies a screen event, where the triangle represents a defender, the circle an offender and the square the screener, who is a teammate of the offender. In Fig. 1(a), no screen is set, and the offender has little chance to shoot since he keeps guarded by the defender. In Fig. 1(b), a screen is setting. The screener interferes with the defender in guarding the offender and creates a good chance for the offender to shoot.

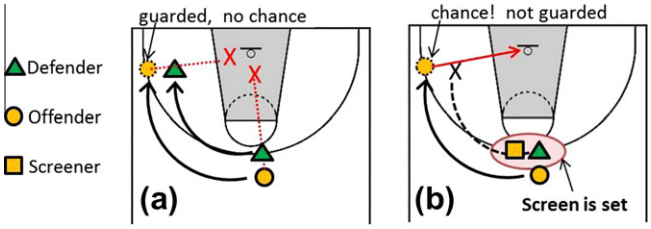

Fig. 1. Illustration of a screen event.

In general, there are plenty of various offensive tactics in basketball and most of the tactics commence with a screen. Screen is the fundamental essence of the offensive tactics. Thus, the coach/players highly desire for inferring the screen patterns executed by the opponents so as to discover the weaknesses of the opponents and then better adapt the operational policy of their own team during a game. Of course, professional coach/players should expertize on the recognition of screen tactics. However, it is a tedious, exhausting, and time-consuming work for a professional to collect and review videos of several matches for game annotation, tactic analysis, and statistics compiling. Hence, computer-aided technology for screen pattern recognition is compelling. With the foregoing motivation, in this paper we are inspired to design a camera calibration and player tracking system capable of recognizing *screen* patterns in basketball videos. The tactical information and statistics can be obtained automatically. The recognized screen patterns are indexed, allowing the coach/players to quickly retrieve the video clips of designated tactics for inspection, analysis, comparison, and so forth, needless of watching the whole videos of several matches. Furthermore, the extracted player trajectories and the recognized screen patterns visualized on a court model greatly assist the coach/players and even the audience in comprehending the tactics executed in basketball games informatively.

The rest of this paper is organized as follows. Section 2 introduces the overview of the proposed system. Section 3 describes the processing steps of camera calibration. Section 4 explains the details of player extraction and tracking for trajectory computation. Section 5 presents screen pattern recognition based on the player trajectory. Experimental results are reported and discussed in Section 6. Finally, we conclude this paper in Section 7.

2. Overview of the proposed system architecture

Object tracking is typically the medium converting the low-level features into high-level events in video processing. Especially, the ball/player trajectory brings significant information for tactic analysis. In basketball games, *screen* is the fundamental essence that most tactics are executed with. Enhanced from our previous work [24], a screen-strategy recognition system capable of detecting and classifying screen patterns in broadcast basketball video based on player trajectory is proposed in this paper. Fig. 2 illus-

trates the system flowchart. Broadcast basketball video contains several prototypical shots, including close-up view shots, median view shots, and court view shot, as exemplified in the top part of Fig. 2. Among the different kinds of shots, court view shots best present the spatial relationships between players and the court. Thus, our analysis is focused on the court view shots. For court view shot retrieval, please refer to [18].

The proposed system starts with camera calibration. To extract court lines, we detect white pixels and then eliminate the white pixels in white or textured regions based on the width test and line-structure constraint. Court lines are extracted using Hough transform. The court line intersections are used as *corresponding points* to compute the camera calibration matrix, which maps real world coordinates to image points or vice versa. With the court line information obtained, we compute the dominant colors and extract players within the court region. For tactic analysis, we have to classify the players and discriminate the offensive/defensive team. Player trajectories are extracted using Kalman filter and are mapped to the court model for screen pattern recognition. The visualized presentation of the extracted player trajectories and recognized screen patterns on the court model gives the coach/players or the fans a further insight into the game and guides them through an efficient way to tactic understanding.

3. Camera calibration

Camera calibration is an essential task to provide geometric transformation mapping the positions of the players in the video frames to the real-world coordinates or vice versa. Since the basketball court can be assumed to be planar, the mapping from a position $\mathbf{p} = (x, y, 1)^T$ in the court model coordinate system to the image coordinates $\mathbf{p}' = (u, v, 1)^T$ can be described by a plane-to-plane mapping (a homography) $\mathbf{p}' = \mathbf{H}\mathbf{p}$, where $\mathbf{H}$ is a 3×3 homography transformation matrix [25]. Homogeneous coordinates are scaling invariant, so we can reduce the degrees of freedom for the matrix $\mathbf{H}$ to only eight. The transformation $\mathbf{p}' = \mathbf{H}\mathbf{p}$ can be written in the following form:

$$\begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \begin{pmatrix} v \\ v \\ 1 \end{pmatrix} \text{where} \quad u = u'w' \quad v = v'c/w' \tag{1}$$

To compute the eight independent parameters, we need at least four pairs of *corresponding points*—the points whose real world coordinates and image coordinates are both known. Since line detection is more robust than locating the accurate positions of specific points, we utilize line intersections to establish point correspondence. In the process, we make use of ideas in general camera calibration, such as white line pixel detection and Hough Transform-based line extraction [25]. The correspondence between the video frame and the basketball court model are illustrated in Fig. 3, which also shows the court lines to be extracted: $L_1 \sim L_5$.

3.1. Court line extraction

In the literature, Farin et al. [25] have proposed an algorithm of extracting court lines and finding line-correspondences, which performs well in several kinds of sport videos such as tennis, volleyball, and soccer. Hough transform is applied to the detected white line pixels. The parameter space (θ, d) is used to represent the line: θ is the angle between the line normal and the horizontal axis, and d is the distance of the line to the origin. An *accumulator matrix* for all (θ, d) is constructed sampling at a resolution of one degree for $\theta$and one pixel for d. Since a line in (x, y) space corresponds to a point in (θ, d) space, line candidates can be determined by extracting the local maxima above a threshold σin the accumulator matrix. The line candidates are then classified into the horizontal lines and the vertical lines. Next, the line candidates are sorted according to their distances to the image boundary for finding line-correspondence.

Nevertheless, when applying the method [25] to basketball videos, we find that the performance is not as good as expected. One major problem is the determination of the threshold value σ. Fig. 4 shows sample results of court line extraction with different σ values. The original frame and the detected white line pixels are shown in Fig. 4(a) and Fig. 4(b), respectively. Fig. 4(c)~(e) show the extracted court lines with different σ values. Small σ value leads to many false lines, whereas large σ value causes insufficient court lines to compute camera calibration parameters. Moreover, the *free-throw line* (see Fig. 3) is very easy to be discarded because it is short in frames. To overcome the obstacles, we propose a step-by-step method to find line-correspondence in basketball video. We search each specific line within a specific range in order to

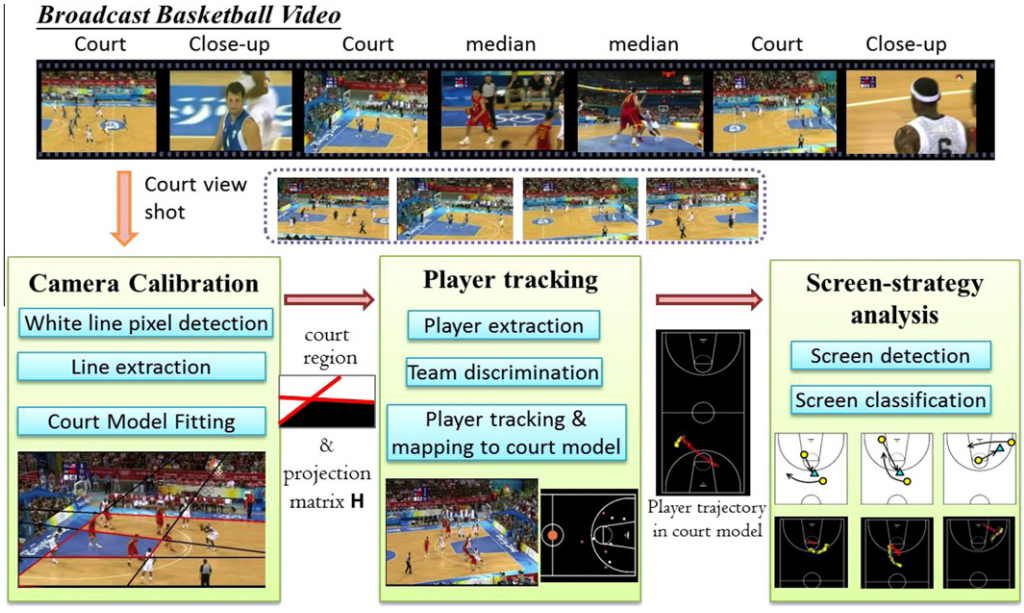

Fig. 2. Flowchart of the proposed system for screen pattern recognition in broadcast basketball video.

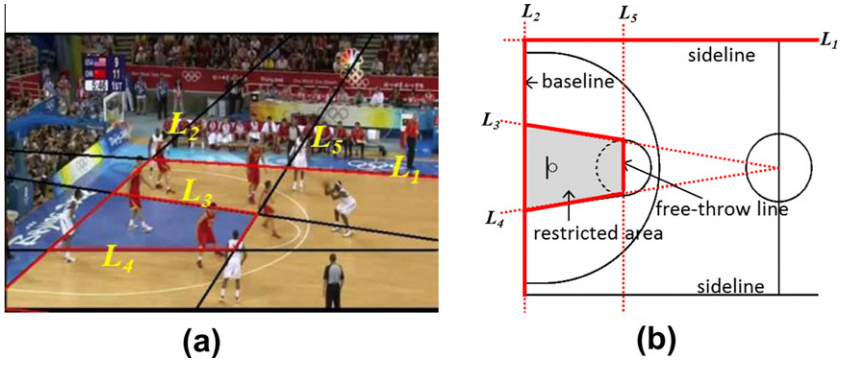

Fig. 3. Correspondence between (a) video frame and (b) basketball court model.

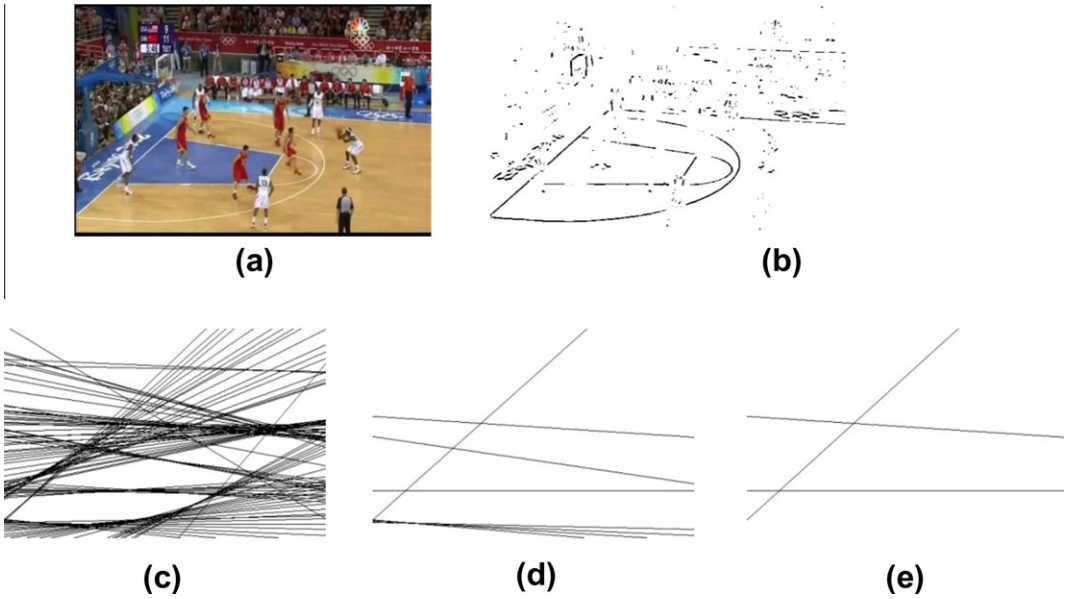

Fig. 4. Sample results of court line extraction. (a) Original frame. (b) Detected white line pixels. (c)-(e) Extracted court lines with different threshold σ.

gather all the necessary lines $L_1 \sim L_5$ (see Fig. 3). The detailed process is described in the following, and Fig. 5 presents the illustration.

(1) *Sideline*L_1 and *baseline*L_2: As shown in Fig. 5(a), the court region is mainly determined by the sideline L_1 and baseline L_2, which are the longest (near-)horizontal and (near-)vertical line in the frame, respectively. (We say "near" because the baseline and sideline are not exactly vertical and horizontal in the frame.) Thus, we extract L_1 via searching the local maximum in the range $|\theta - \pi/2| \leqslant \rho$of the accumulator matrix produced by Hough transform ($\theta = \pi/2$ represents a horizontal line) and extract L_2 via searching the local maximum in the range $|\theta - \pi/2| > \rho$, where $\rho$is a statistically determined threshold. To prevent extracting the sideline close to the frame bottom, we add the constraint that the parameter d of the sideline L_1 should be greater than quarter the frame height, as shown in Fig. 6. Furthermore, the slope of the detected baseline also helps to identify whether the frame presents the left court or the right.

(2) *Lines*L_3 and L_4: With the baseline and sideline extracted, we can determine the *court region* and filter out the white line pixels outside the court region, as shown in Fig. 5(b) and (c). We apply Hough transform to the remaining white line pixels and extract the longest two (near-) horizontal lines as L_3 and L_4, as shown in Fig. 5(d). The top edge L_3 and bottom edge L_4 can be distinguished by the slopes.

(3) *Free* − *throwline*L_5: Again, we filter out the white line pixels outside the region bounded by L_3 and L_4, as shown in Fig. 5(e), and apply Hough transform to the remaining white line pixels. Because the main camera is located at the center of the court, the angle between the free-throw line L_5 and the horizontal axis in the frame is greater than the angle between the baseline L_5 and the horizontal axis no matter which side of court is on screen, as show in Fig. 3(a). Hence, we extract the free-throw line L_5 via searching the local maximum in the ranges $0 < \theta < \theta_b$ and $\theta_b < \theta < \pi/2$ for left court frames and right court frames, respectively, where θ_b is the angle between the baseline normal and the horizontal axis. Fig. 5(f) shows the detected free-throw line L_5.

3.2. Computation of camera calibration parameters

With the court lines $L_1 \sim L_5$ extracted, now we are ready to compute the camera calibration parameters. Multiplying out the linear system in Eq. (1), we obtain two equations, Eqs. (2) and (3), for each corresponding point—the point whose real world coordinate (x,y) and image coordinate (u,v) are both known.

$$h_{00}x + h_{01}y + h_{02} = u(h_{20}x + h_{21}y + 1) \tag{2}$$

$$h_{10}x + h_{11}y + h_{12} = v(h_{20}x + h_{21}y + 1) \tag{3}$$

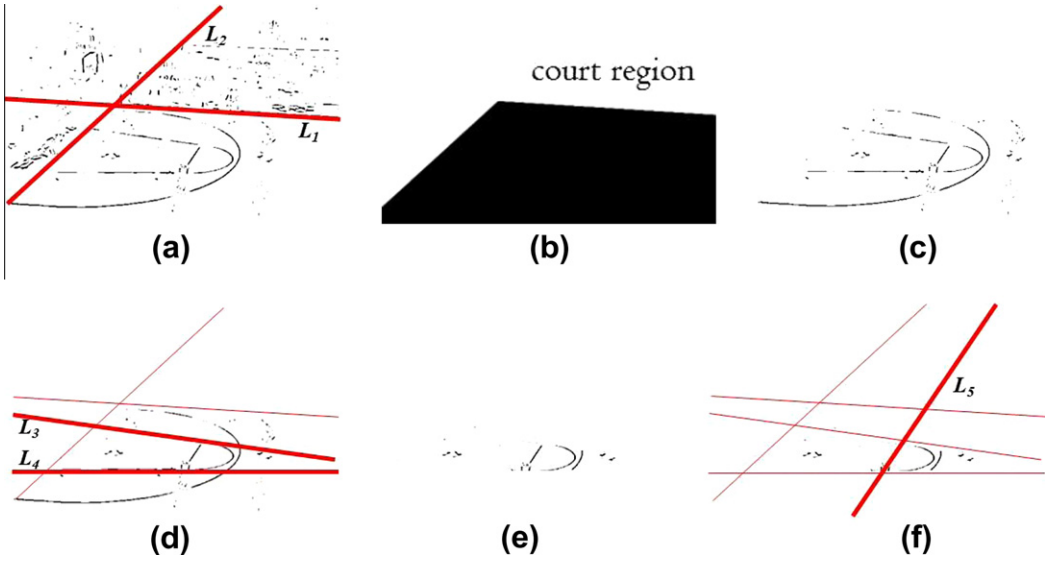

Fig. 5. Processing steps of extracting the court lines $L_1 \sim L_5$. (a) Detected sideline L_1 and baseline L_2. (b) Court region. (c) White line pixels within the court region. (d) Detected L_3 and L_4. (e) White line pixels within the region bounded by L_3 and L_4. (f) Detected free-throw line L_5.

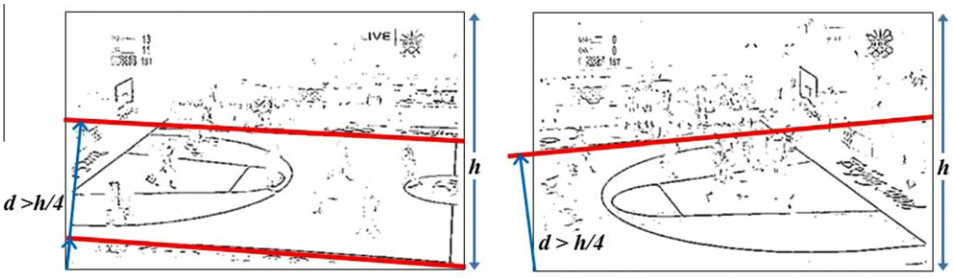

Fig. 6. Constraint to prevent ambiguity in sideline extraction.

From Eqs. (2) and (3), we set up a linear system **AB** = **C** as Eq. (4), where n is the number of corresponding points.

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1x_1 & -u_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -v_2x_2 & -v_2y_2 \\ & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -u_nx_n & -u_ny_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -v_nx_n & -v_ny_n \end{pmatrix} \begin{pmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{pmatrix} = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_n \\ v_n \end{pmatrix} \tag{4}$$

In our system, the extracted court lines $L_1 \sim L_5$ form six intersection points, namely $n = 6$. To solve **B**, we can over-determine **A** and find a least squares fitting for **B** with a pseudo-inverse solution:

$$\mathbf{AB} = \mathbf{C}, \quad \mathbf{A}^T\mathbf{AB} = \mathbf{A}^T\mathbf{C}, \quad \mathbf{B} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B} \tag{5}$$

Thus, we derive the parameters of camera calibration to form the matrix transforming the real world coordinate to the image coordinate. Since **H** homography matrix performs a plane- to-plane mapping, the image position $\mathbf{p}\prime$ can also be transformed to a position **p** on the real world court model by

$$\mathbf{p} = \mathbf{H}^{-1}\mathbf{p}' \tag{6}$$

4. Player extraction and tracking

Player tracking in broadcast basketball video is intrinsically challenge task due to the frequent occlusions of the players, especially the players of the same team wearing uniforms of the same color. In addition to the playing field, a court view frame also contains some areas of the audience or stadium, as shown in Fig. 3(a). Objects and noises in the audience region will cause false alarms and degrade the performance in player segmentation and tracking. To improve tracking accuracy and computational efficiency, we utilize the extracted baseline and sideline to determine the *court region*, as shown in Fig. 5(b), and direct the following processes at the court region, as illustrated in Fig. 7.

4.1. Foreground object segmentation

As shown in Fig. 3, the court region contains two parts: (1) the *restricted area* and (2) the playing field excluding the restricted area (say the *unrestricted area*). The color of the restricted area is almost uniform, as is also the case for the unrestricted area. We can construct the background model via detecting the dominant colors of the restricted area and the unrestricted area individually on the basis of Gaussian mixture model (GMM) [26,27]. Then, region growing is applied to construct the dominant color map, as shown in Fig. 7(b), where the non- dominant color pixels in black are deemed foreground pixels. The background subtraction method is used to segment the foreground objects consisting of the difference between the video frames and the background

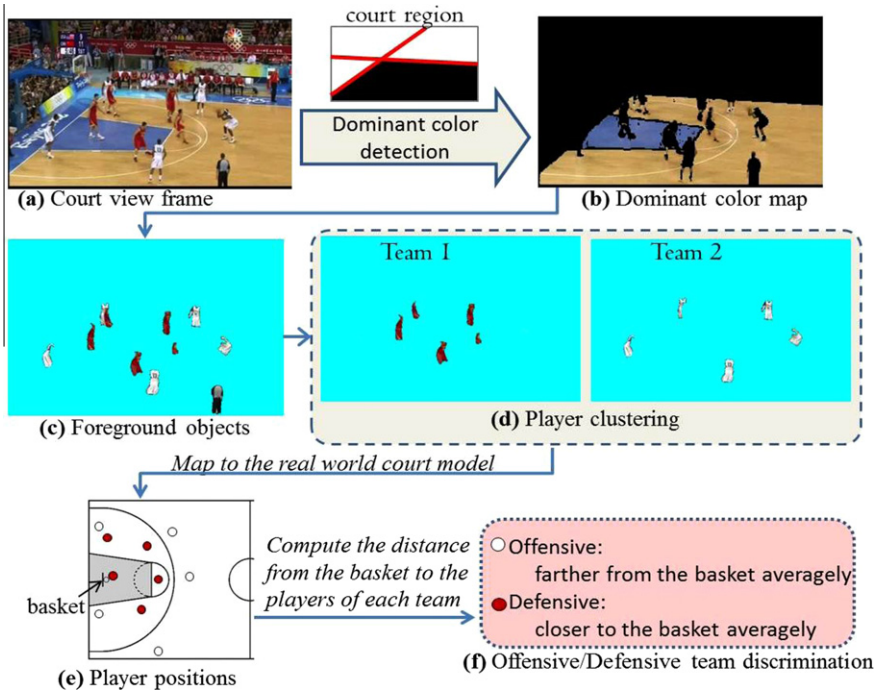

Fig. 7. Illustration of player segmentation and team discrimination

model, and morphological operations are performed on the foreground pixels to remove small objects, noises and gaps, as shown in Fig. 7(c).

4.2. Player clustering and offensive/defensive team discrimination

The foreground objects contain players of the two teams, referees and some noises. Yet, only the player regions are contributive to tactic analysis. Based on color information, we attempt to use k-means clustering to separate the foreground regions into two clusters, each of which represents the uniform color of a team. However, we cannot have just two clusters ($k = 2$) due to the non-player regions, such as referees or other noises To determine the cluster number, we experiments on different number of clusters, as shown in Fig. 8, where the x-axis indicates the cluster number k and the y-axis gives the clustering error. Generally, the more the clusters, the smaller the total distance between all data points and their corresponding cluster centroid, which is regarded as *clustering error*. Based on the experiment, we choose $k = 6$ since the clustering error almost converges when there are more than six clusters, and we select YCbCr color space in our system since YCbCr leads to smaller clustering error. Thus, we separate the foreground regions into six clusters and take the largest two clusters as the uniform colors of the two teams. Fig. 7(d) shows the clustered players of two teams.

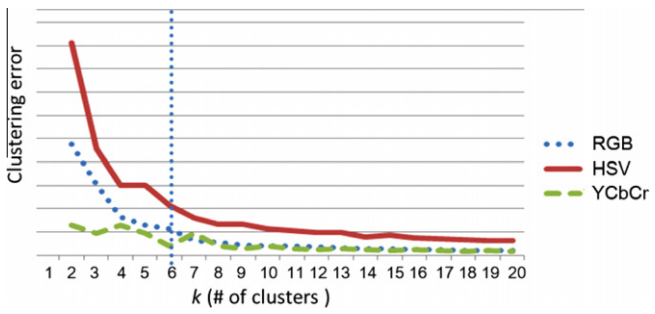

Fig. 8. Experimental data with different color spaces and number of clusters.

We then discriminate the offensive/defensive team to realize which team possesses the ball. In a basketball game, the players on defense try their best to avoid the offensive players from putting the ball into the basket, so each defender is expected to stand closer to the basket than the offender he is guarding, which helps to identify the team on offense. As show in Fig. 9, we map the in-frame positions of players to the real-world court model by Eq. (6). For each team, we compute the average distance from the basket to players. The team with players averagely farther away from the basket than the other team is recognized as the offensive team.

4.3. Kalman filter-based player tracking

Being widely used in moving object tracking [14,15], the Kalman filter provides optimal estimates of the system parameters, such as position and velocity, given measurements and knowledge of a system's behavior [28]. In this work, we apply a Kalman filter-based approach to track players. In general, the Kalman filter describes a system as:

$$\mathbf{x}_n = \mathbf{A}_n \mathbf{x}_{n-1} + \mathbf{w}_n \tag{7}$$

$$\mathbf{z}_n = \mathbf{H}_n \mathbf{x}_n + \mathbf{v}_n \tag{8}$$

where $\mathbf{x}_n$ is the state vector (representing estimated player position at the nth frame), $\mathbf{A}_n$ is the system evolution matrix, and $\mathbf{w}_n$ is the system noise vector. $\mathbf{z}_n$ is the vector of measurements (positions of player candidates), $\mathbf{H}_n$ is the unit array, and $\mathbf{v}_n$ is the measure noise vector.

Initially, we create a new tracker for each player candidate. Then, the Kalman filter algorithm is used for predicting each player's position in the next frame. The player candidate closest to the predicted position is regarded as measurement and the parameters of Kalman filter are update. If there is no candidate close to the predicted position, we regard the predicted state as measurement directly. Besides, since all players are expected to stay in the court, once a tracker state is out of the court, we mark it as *missing*. Every time a tracker misses, we double the search range, since the object may be occluded and the tracker misses

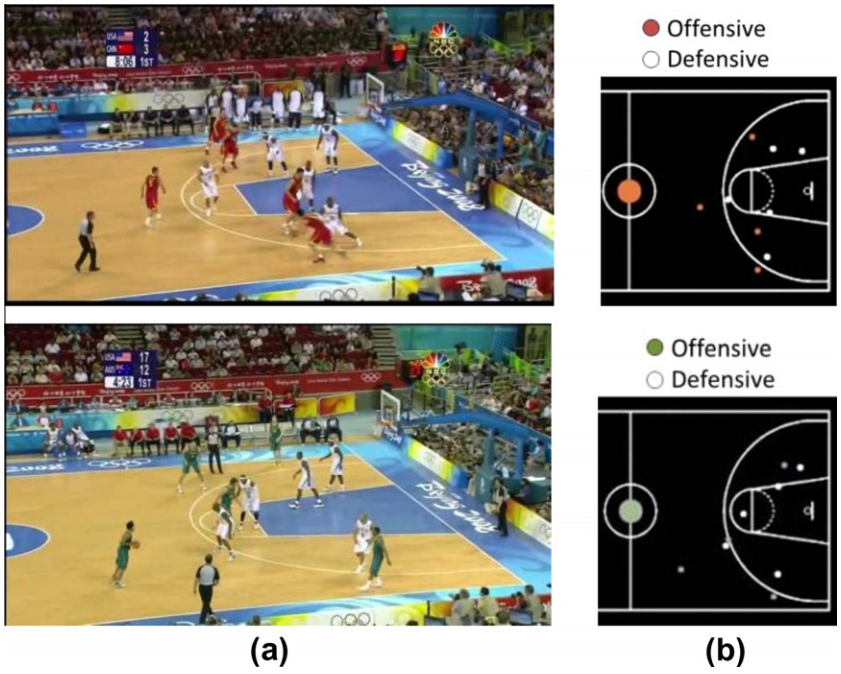

Fig. 9. Offensive/defensive team discrimination. (a) Original frame (b) Player positions mapped to the court model and the discrimination result.

temporarily. We expect that the tracker can keep tracking on the object when it shows again. If a tracker misses for consecutive ε_f frames, that is, the tracker is outside the court for too many frames, we terminate the tracker. After updating all trackers, we add new trackers for the untracked candidates. The Kalman filter-based player tracking is described in Algorithm 1.

Algorithm 1: Player Tracking

Input: tracker list trackers and candidate list candidates
Output: none
|| – *Step 1: update trackers* –
for each *tracker* **in** *trackers* **do**
  local *prediction*:= *predict (tracker)*
  local *measurement*
if ∃ *candidate* ∈ *candidates: dist (prediction, candidate)* < δ_t
  then
    measurement := *candidate*
set_tracked (candidate)
else
  measurement := *prediction*
end if

correct (tracker, measurement)
if *is_out_of court_bound (measurement)* **then**
increase_missing_count (tracker) **else**
reset_missing_count (tracker)
end if
if *missing_count (tracker)* > ε_f **then**
  terminate (tracker)
end if
end for
|| – *Step 2: create new trackers* –
for each *candidate* **in** *candidates* **do**
if *not_tracked (candidate)* **then**
add_tracker (trackers, candidate)
end if
end for

5. Screen pattern recognition

Typically, basketball tactics consist of a series of *screen* strategies, attempting to make an offensive teammate not guarded by a defender and have a good chance to score. Hence, screen is the fundamental essence of offensive basketball tactics. We introduce three most frequently used screen strategies: *front-screen*, *back-screen*, and *down-screen*, as illustrated in Figs. 10–12, respectively.

Front-screen: The screener sets the screen for the offender, who uses the screen to drive past the defender in the direction perpendicular to the direction from the defender to the basket, as shown in Fig. 10, where the triangle, circle, and rectangle indicate the defender, offender, and screener, respectively. Fig. 10(a) illustrates the trajectories of the offensive players involved in the screen. The moments before the screen, on the screen and after the screen are shown in Fig. 10(b)–(d), respectively. Front screen is used to create open shots inside and outside. Screening for the ball handler often creates mismatches in height or speed when defenders switch.

Back-screen: The screener sets the screen behind the defender so that the offender can cut toward the basket, as shown in Fig. 11. The screener must allow a step between himself and the defender here, as the defender will step back while trying to catch the offender.

Down-screen: As shown in Fig. 12, the screener sets the screen for the offender, usually down low on the court, who uses the screen to cut up toward the ball to receive the pass. The screener is usually facing the defender.

5.1. Screen detection

In order to prevent the defensive players from helping their teammates, the offensive players tend to make the space wider and seldom stay close to each other. Hence, the case of an offensive player standing next to another offensive player implies that the offensive team is probably executing a tactic. Therefore, we can initially determine whether there is a screen event based on the dis-

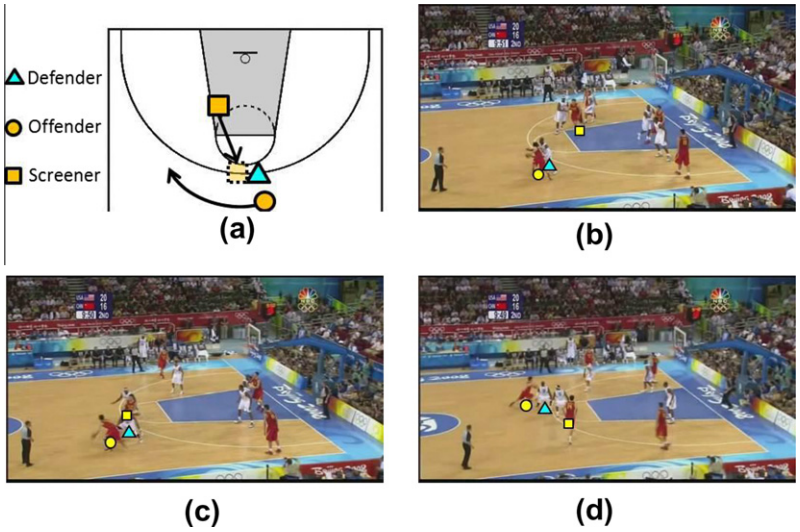

Fig. 10. Front-screen. (a) Player trajectories in the screen strategy. (b) Before screen. (c) On screen. (d) After screen.

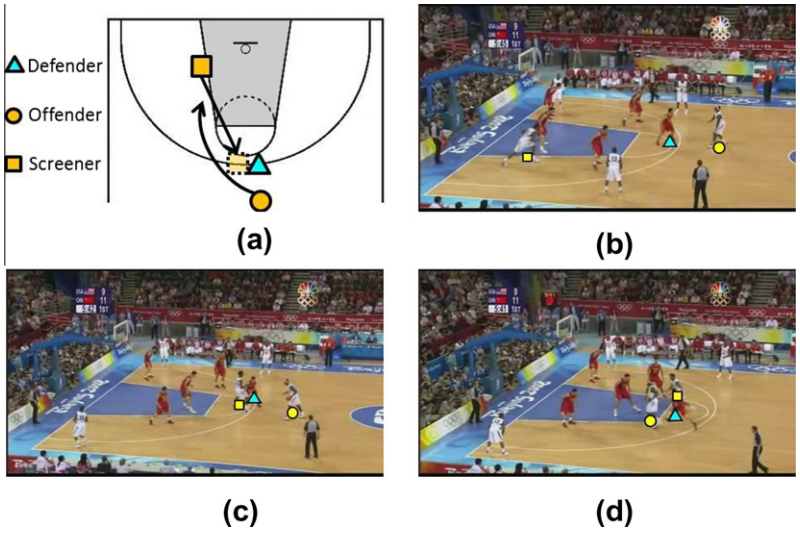

Fig. 11. Back-screen. (a) Player trajectories in the screen strategy. (b) Before screen. (c) On screen. (d) After screen.

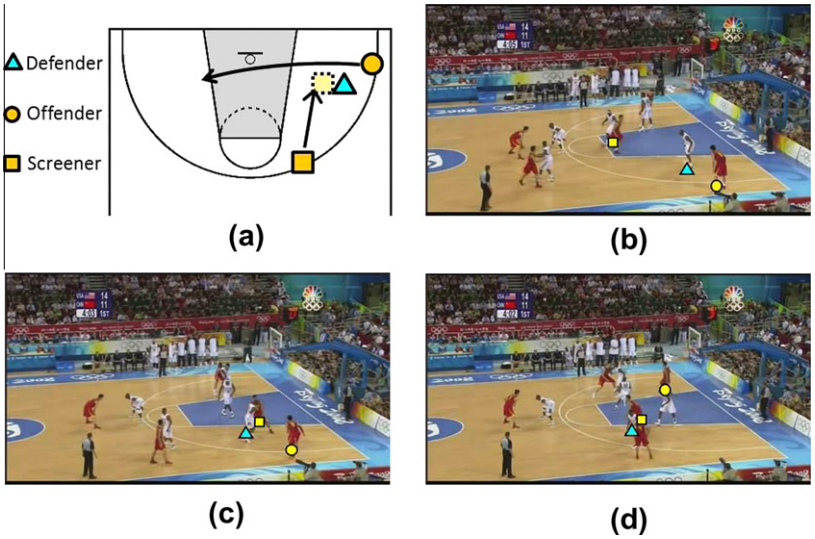

Fig. 12. Down-screen. (a) Player trajectories in the screen strategy. (b) Before screen. (c) On screen. (d) After screen.

tances between offensive players. Besides, since the screen is set to block the defender(s), there must be at least one defensive player between the screener and his teammate. Thus, the screener can also be recognized. Generally, the defender stays close to his target but not side by side since the defender has to prevent his target from driving to the basket. On the other hand, the screener must be in contact with the defender in order to block the defender effectively. Accordingly, if we find that there are two offensive players close to each other, and there is at least one defensive player between them, we can infer that there is a screen and the screener is the offensive player closer to the defensive player. Algorithm 2 depicts the proposed screen detection method. In the following, we use *screenee* to mean the offensive player whom the screener sets screen for.

Algorithm 2: Screen Detection

Input: $players_{off}$ and $players_{def}$ //*offensive and defensive players*
Output: *screener and screenee* //*the screener sets screen for the screenee*
screener:= *nil*
screenee:= *nil*
if $\exists$ $player_i \in players_{off}$, $player_j \in players_{off}$: $\delta_s < distance(player_i, player_j) < \Delta_s$ **and** $i \neq j$ **then**
  if $\exists$ $defender \in players_{def}$: $distance(defender, player_i) < \delta_s$ **or** $distance(defender, player_j) < \delta_s$ **then**
    if $distance(defender, player_i) < distance(defender, player_j)$ **then**
      $screener := player_i$
      $screenee := player_j$
    else
      $screener := player_j$
      $screenee := player_i$
    end if
  end if
end if
return *screener, screenee*

5.2. Screen classification

The foregoing introduction describes that each screen type follows a specific pattern. The *front-screen* is usually set around the three-point line and the screenee moves to an open space instead of driving to the basket. A screener sets a *back-screen* by moving from the low post to the high post, and the screenee drives to the basket after the screen. The *down-screen* is set by a screener moving from the high post to the low post. Therefore, we have to utilize the trajectories of the screener and screenee to recognize the screen pattern. Here we denote the initial position of the screener as p_{init}, the last position of the screener as p_{last}, the position of the screener when the screen is set as p_{screen}, and those of the screenee as $p\prime_{init}$, p'_{last}, and p'_{screen}. Also, the position of the basket is represented by p_{basket}. Take a look again at Figs. 10, 11. The down-screen is the only type that the screener moves down to the low post. Hence, we first check the screener trajectory and confirm whether the screener moves to the baseline by $|p_{basket} - p_{init}|$ and $|p_{basket} - p_{screen}|$. If the screener moves to the baseline, our proposed system recognizes it as a down-screen, as formulated in Eq. (9). The other two screen patterns (front-screen and back-screen) are hard to be distinguished by the screener trajectory due to the similar movement of the screeners. Once we find that the screener does not move to the baseline (not a down-screen), we analyze the screenee trajectory. In a back-screen the screenee tends to move to the basket, while in a front-screen the screenee is likely to move around outside the restricted area. Accordingly, we discriminate back-screen and front-screen by the angle between the screenee's moving direction ($d_{moving} = p'_{last} - p'_{screen}$) and the direction from the screenee to the basket ($d_{basket} = p'_{basket} - p'_{screen}$). If the angle between the two directions is small, that is, the screenee tends to move to the basket, our proposed system recognizes it as a back-screen; otherwise, it is recognized as a front-screen.

$$screenType = \begin{cases} Down, if\ |p_{basket} - p_{init}| > |p_{basket} - p_{screen}| \\ Back, if\ |p_{basket} - p_{init}| \leqslant |p_{basket} - p_{screen}| \text{ and } \cos^{-1}\left(\frac{d_{moving} \cdot d_{basket}}{d_{moving}||d_{basket}|}\right) < \theta_s \\ Front, if\ |p_{basket} - p_{init}| \leqslant |p_{basket} - p_{screen}| \text{ and } \cos^{-1}\left(\frac{d_{moving} \cdot d_{basket}}{d_{moving}||d_{basket}|}\right) \geqslant \theta_s \\ undefined, else \end{cases} \quad (9)$$

6. Experimental results and discussion

To evaluate the effectiveness of our proposed framework, we conduct the experiments on the video data of the Beijing 2008 Olympic Games. The testing videos include four men's basketball matches: USA vs. AUS, ARG vs. USA, USA vs. CHN, and ESP vs. USA recorded from live broadcast television programs with frame resolution of 640×352 (29.97 fps). We manually select 20 clips with obvious tactic execution from each of the four matches. Totally, we have 80 video clips (40 for training and the other 40 for testing).

6.1. Performance of court line detection and camera calibration

In Section 3, we first detect white pixels using color information. To improve the accuracy and efficiency of the subsequent line detection and camera calibration processes, we eliminate the white pixels in white regions or in textured regions based on the width test and the line-structure constraint. Fig. 13 demonstrates sample results of white line pixel detection. The original frames are presented in Fig. 13(a). In Fig. 13(b), although most of the white pixels in white regions such as white uniforms are discarded, there are still false detected white line pixels occurring in the textured areas. With line-structure constraint, Fig. 13(c) shows that the number of false detections reduced, and white line pixel candidates are retained only if the pixel neighbor shows a linear structure. Those discarded pixels mostly come from spectators, the score board overlay, the channel mark, and advertisement logos. Table 1 presents the numeric statistics of white pixel numbers with/without the line-structure constraint. It is obvious the line-structure constraint effectively filters out a high percentage of the non-line white pixels.

In Section 3.1, we have mentioned that Farin's court line extraction/camera calibration method [25] cannot achieve good performance for basketball videos, although it works well on tennis, volleyball, and soccer videos. The major reason is that the *free-throw line* is much shorter than other court lines and Farin's method cannot effectively extract the free-throw line as well as other court lines for finding line correspondences. Hence, we propose a step-by-step approach to find the court lines for camera calibration. The camera calibration results are inspected manually that a frame is regarded as "well calibrated" when all the five court lines ($L_1 \sim L_5$ in Fig. 3) are extracted correctly. The accuracy of camera calibration is then defined by

$$Accuracy = \frac{\#wellcalibratedframe}{\#totalframes} \quad (10)$$

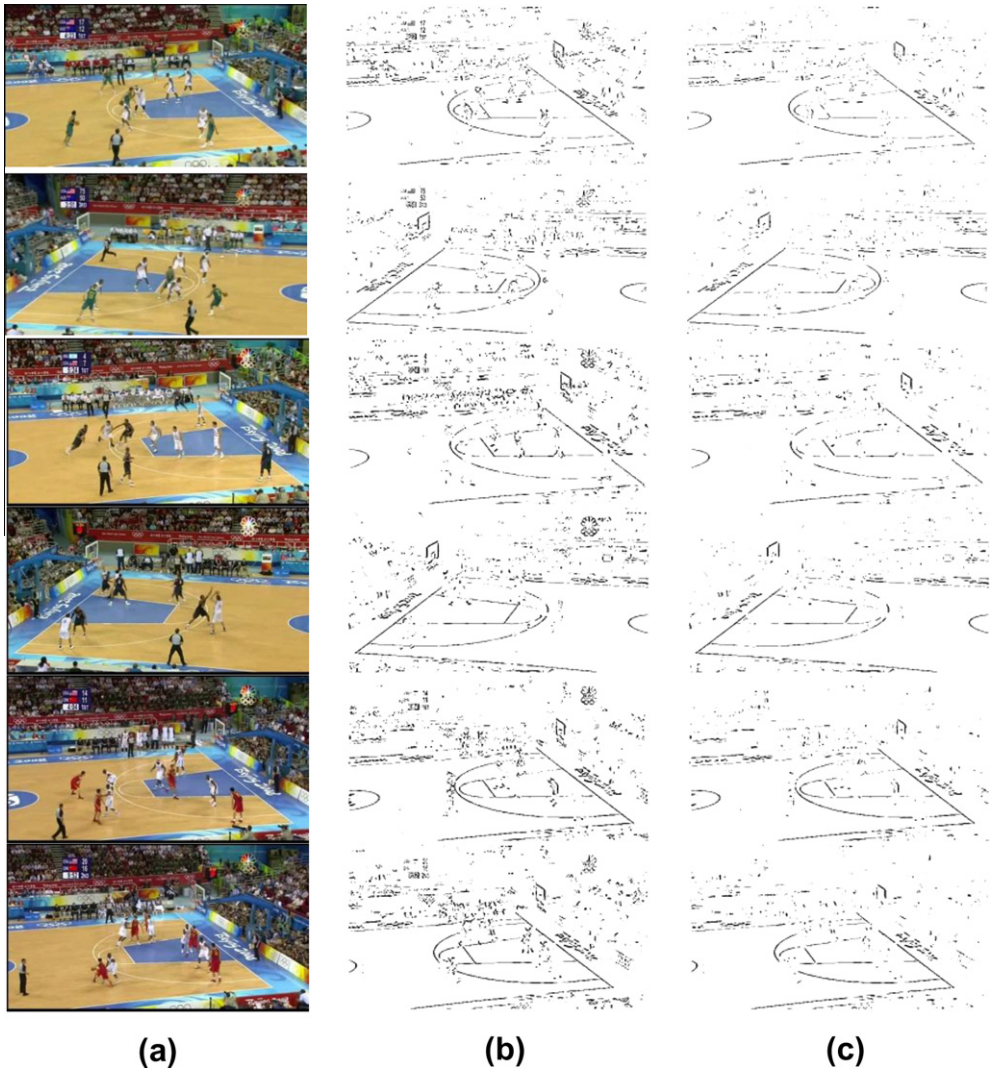

Fig. 13. Results of white pixel detection. (a) Original frame. (b) Without line-structure constraint. (c) With line-structure constraint.

Table 1
Statistics of white line pixel detection with/without line-structure constraint. (N_1: average number of white line pixels without line-structure constraint per frame, N_2: average number of white line pixels with line-structure constraint, P_d: percentage of discarded non-line white pixels.)

Matches	N_1	N_2	P_d(%)
USA vs. AUS	10932.9	7200.1	34.14
ARG vs. USA	8516.0	4651	45.39
USA vs. CHN	9061.2	5379.0	40.64
ESP vs. USA	11509.7	6676.4	41.99

Table 2
Performance of the proposed court line extraction/camera calibration method.

Matches	# Frames	# Well calibrated frames	Accuracy
USA vs. AUS	1445	1358	93.98
ARG vs. USA	1242	1087	87.52
USA vs. CHN	1359	1201	88.37
ESP vs. USA	1368	1306	95.47
Total	5414	4952	91.47

The results in Table 2 show that up to 91.47% of frames can be well calibrated by our proposed approach, which facilitates the subsequent analysis. Example calibrated results are demonstrated in Fig. 14, where black lines are court lines extracted by our proposed method, and red lines give the real court model projected onto image coordinates. Since the camera motion in a shot is continuous, the corresponding points (the court line intersections) should not move dramatically in successive frames. Hence, though court lines may be occluded by players in some frames, the incorrect coordinates of the corresponding points can be recovered by interpolation.

Farin's court line extraction/camera calibration method [25] is implemented for comparison. The threshold of Hough transform is set $\sigma = 25$ experimentally. A larger threshold will lead to the misdetection of the free-throw line in most frames, resulting in a lower accuracy, while a smaller threshold will cause the explosively increase of false lines, requiring much higher computational cost. The results are presented in Table 3, where N_L gives the average numbers of extracted lines per frame. Since the number of court lines is fixed, a larger N_L value means more false lines are extracted. On average, the accuracy of Farin's method is 77.32% with about 22 lines extracted per frame, compared to the accuracy 91.47% of our proposed approach. The main reason causing errors is the misdetection of the free-throw line.

Fig. 14. Results of court line extraction/camera calibration.

Table 3
Performance of the court line extraction/camera calibration method in [25]. (N_L: average numbers of extracted lines per frame).

Matches	# frames	# well calibrated frames	Accuracy	N_L
USA vs. AUS	1445	1154	79.86	23.78
ARG vs. USA	1242	1002	80.68	20.99
USA vs. CHN	1359	973	71.60	20.83
ESP vs. USA	1368	1057	77.27	22.54
Total	5414	4186	77.32	22.08

6.2. Results of player extraction and offensive/defensive team discrimination

With the court line information obtained after camera calibration, we compute the dominant colors within the court region and extract the non-dominant colored regions as foreground objects. The extracted players are further clustered into the two teams. Fig. 15 demonstrates the player extraction and clustering results. Original frames are given in Fig. 15(a). The dominant colors of the restricted area and the unrestricted area are determined individually to form the dominant color maps, as shown in Fig. 15(b). Based on color information, we classify the foreground pixels into two groups to obtain the player mask for each team, as presented in Figs. 15(c) and (d).

Through player clustering, some non-player foreground objects, e.g. the referees (see the red ellipses in Fig. 15(a)), can be removed. However, noises may occur in the player mask in the case that the referee has similar color with the players, as in 3rd and 4th rows of Fig. 15. Sometimes player bodies may be divided into parts, such as arms and numbers on uniforms (see the blue squares in 1st and 2nd rows of Fig. 15), even though we have performed the dilation operation to remove gaps. On the other hand, when two objects are close to each other, it is hard to judge whether they are two players or they are just two parts belonging to one player (see the green triangles in 3rd, 4th, and 6th rows of Fig. 15). To resolve this problem, we regard two objects close to each other as different objects, and let the tracking procedure clarify if they belong to the same object through their trajectories. In some cases, the players standing near the court boundary may be extracted incorrectly since we only consider pixels within the court region. This phenomenon may affect the result of player tracking. However, based on basketball domain knowledge, we know that the players standing near the court boundary do not have enough space for screen setting, so we can simply ignore the players nearby the court boundary.

6.3. Performance of player tracking and screen pattern recognition

For tactic analysis, we have to discriminate the offensive and defensive teams first. For each of the two teams, the average distance from the basket to the players is computed. The team with players averagely farther away from the basket is recognized as the offensive team, and the other is the defensive team. In the 40 testing clips, all offensive/defensive teams can be correctly discriminated. Sample results of offensive/defensive team discrimination are shown Fig. 16, where the left part of each figure shows the video frame, and the right part presents the player positions mapped to the court model as well as indicates the offensive and defensive teams.

The proposed system tracks players using Kalman filter, and then detects and recognizes screen patterns based on the player trajectory information. Fig. 17 shows the results of player tracking and screen pattern recognition of some testing clips. The offender, screener, and defender are labeled with yellow, green and red ellipses under their feet, respectively. The left part of each image in Fig. 17 shows the frame when a screen pattern is detected, and the right part presents the recognized screen pattern along with the player trajectories of the offender and screener.

We manually inspect the player tracking results of each frame in the testing clips and evaluate the performance with *precision* and *recall* measures defined in Eq. (11) [29]:

$$\text{precision} = \frac{\#\text{correct correspondences}}{\#\text{established correspondences}}, \quad \text{recall} \frac{\#\text{correct correspondences}}{\#\text{actual correspondences}} \tag{11}$$

where "# actual correspondences" denotes the number of objects at the moment and "# established correspondences" represents the total number of created trackers. A *correct* correspondence means the tracker really corresponds to one player. Table 4 presents the precision and recall of player tracking results for each testing clip. Overall, the average precision and recall rates are 89.71% and 89.20%, respectively. Most failures result from the occlusion and merging problems. Occlusion is normally a major obstacle in object tracking, and it occurs frequently in basketball video. Furthermore, it is difficult to predict trajectories of basketball players, since they usually change directions rapidly. So far, the occlusion problem cannot be perfectly solved in basketball video. Besides, player bodies sometimes are divided into parts, so the number of the extracted players may be more than the number of the actual players. This phenomenon leads to redundant trackers for the same player and decreases the accuracy.

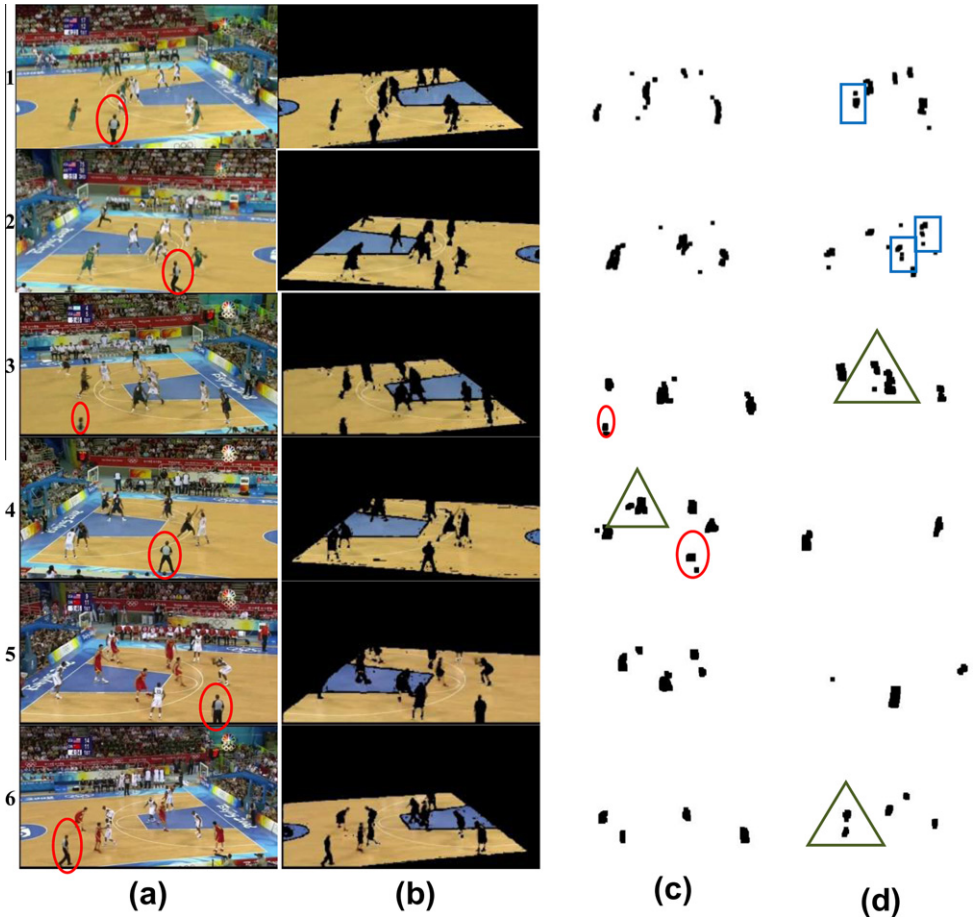

Fig. 15. Results of player extraction and clustering. (a) Original frame. (b) Dominant color map. (c) Player mask of team 1. (d) Player mask of team 2.

Fig. 16. Sample results of offensive/defensive team discrimination.

In spite of some errors in player tracking, the proposed system can still correctly recognize the screen patterns in 35 testing clips. Only five clips are misrecognized. Example results of correctly recognized screen patterns are demonstrated in Fig. 17. Take Fig. 17(a) for explanation. The screener, labeled with green ellipse under his feet, moves up from near the free-throw line to the top of the three-point line and sets the screen, and his offensive teammate (labeled with yellow ellipse) drives past the defender to the upper region. Thus, our algorithm regards it as a "front screen." Fig. 18 shows three failed cases. In Fig. 18(a), the screener moves from near the top sideline to the top of the three-point line and sets the screen, and the screenee moves from near the bottom sideline to the top sideline. It should be a "front-screen," but the screener is closer to the basket when he sets the screen than he starts to move. As a result, our algorithm regards it as a "down-screen." In Fig. 18(b), the screener moves from the free-throw line to the low post, and the screenee tries to move to the three-point line. It is a special case since the screener is setting the screen on the

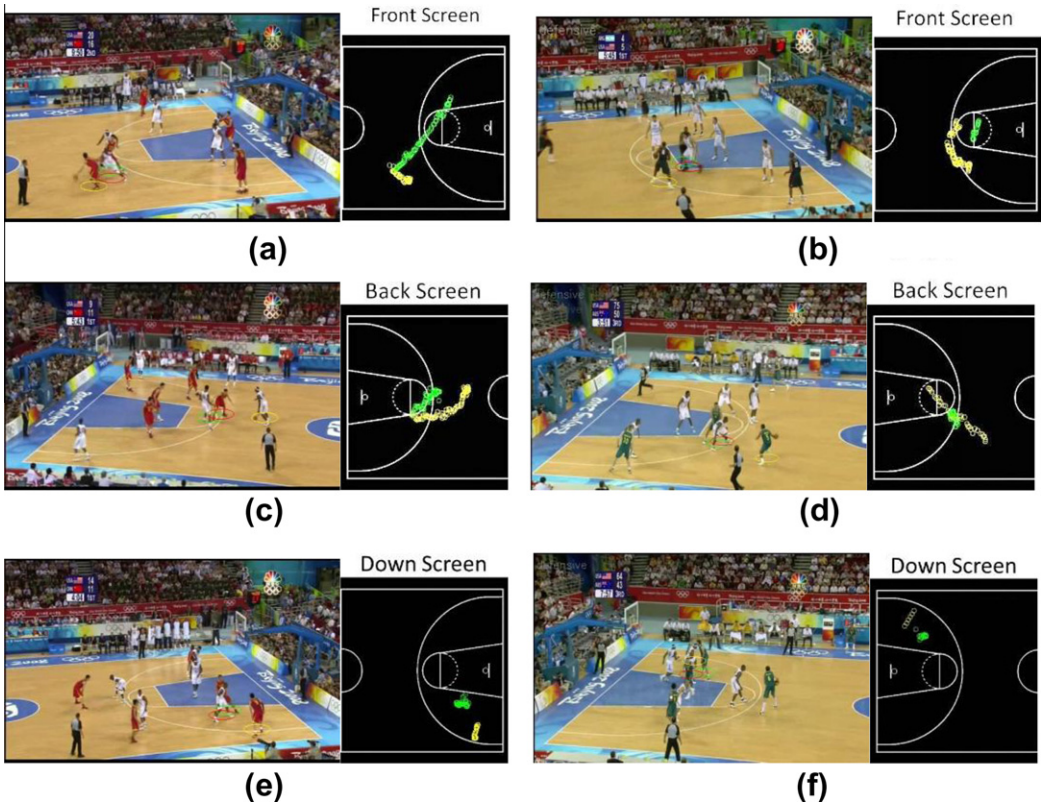

Fig. 17. Results of player tracking and screen pattern recognition. (a and b) Front screen. (c and d) Back screen. (e and f) Down screen.

Table 4
Precision and recall of player tracking results for each testing clip.

USA vs. AUS			ARG vs. USA			USA vs. CHN			ESP vs. USA		
Clip	Precision	Recall	Clip	Precision	Recall	Clip	Precision	Recall	Clip	Precision	Recall
1–1	89.27	88.78	2–1	89.68	89.30	3–1	94.26	94.46	4–1	95.65	75.86
1–2	76.39	79.94	2–2	87.26	86.94	3–2	93.28	94.43	4–2	88.33	72.60
1–3	80.46	82.97	2–3	92.83	93.60	3–3	91.40	90.56	4–3	89.83	81.54
1–4	81.53	82.08	2–4	86.80	88.80	3–4	92.01	92.84	4–4	91.94	91.94
1–5	88.16	89.51	2–5	88.64	88.18	3–5	99.19	97.27	4–5	90.91	76.92
1–6	84.83	85.36	2–6	88.15	86.65	3–6	92.56	94.02	4–6	87.67	79.01
1–7	82.91	84.29	2–7	92.70	94.03	3–7	92.50	93.98	4–7	93.94	83.78
1–8	86.02	85.18	2–8	90.36	91.49	3–8	93.24	94.17	4–8	93.62	83.02
1–9	84.74	85.21	2–9	86.88	87.53	3–9	92.34	92.79	4–9	93.75	93.75
1–10	89.18	90.34	2–10	93.48	92.27	3–10	97.02	96.32	4–10	94.74	78.26
All	84.74	85.77	**all**	89.63	89.72	all	94.07	94.32	all	91.67	81.77

moving path of the screenee instead of standing next to the defender, so that the screener is farther away from the defender than the screenee when the screen is detected, and our algorithm fails to recognize the screener correctly. In Fig. 18(c), the screen is set near the top edge of the restricted area, and the screenee moves from outside the three-point line the free-throw line. Although the screenee does not drive to the basket and it should be a "front-screen," the angle between his moving direction and the basket direction is small so that our algorithm mistakes the screen type as a "back-screen." Overall, the proposed system achieves a high accuracy of 87.5% (35/40) in screen pattern recognition.

6.4. Comparison and discussion

In the literature, many approaches of content analysis in basketball games have been developed. The player tracking scheme with applications to tactic analysis in [23] better meets the basketball professionals' requirements. Thus, we compare and discuss our proposed system with Hu's work [23], wherein player trajectories are extracted by a CamShift-based tracking method and mapped to the real world court model for professional-oriented applications, including wide-open event detection, trajectory-based target clips retrieval, and tactic inference.

We apply Hu's system [23] to our testing data, and Table 5 presents the player tracking results.[1] Fig. 19 illustrates the comparison on player tracking in terms of precision and recall. Compared with the average precision and recall rates (89.71% and 89.20%, respectively) of our player tracking method, Hu's system achieve a similar precision rate (85.71% on average) but a lower recall rate (68.35% averagely). One major factor causing the player tracking performance not to be as good as in [23] is that our testing data contain frequent occlusion of players of the same team since we focus on screen pattern recognition. Thus, players are easily missed due to

[1] Many thanks to Dr. Hu Min-Chun, who is the corresponding author of [23], for providing us with their system for performance comparison.

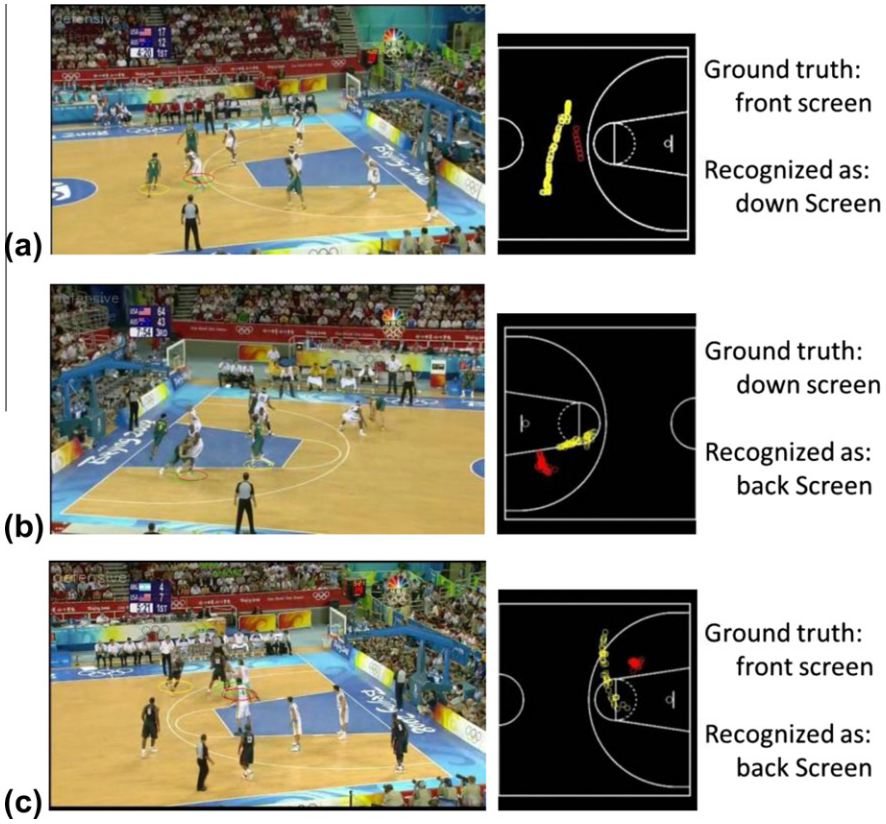

Fig. 18. Failed cases of screen pattern recognition. (a) Front screen is misrecognized as down screen. (b) Down screen is misrecognized as back screen. (c) Front screen is misrecognized as back screen.

Table 5
Player tracking results of [23].

USA vs. AUS			ARG vs. USA			USA vs. CHN			ESP vs. USA		
Clip	Precision	Recall	Clip	Precision	Recall	Clip	Precision	Recall	Clip	Precision	Recall
1–1	90.41	66.73	2–1	85.26	51.92	3–1	83.48	52.46	4–1	89.47	72.34
1–2	92.03	74.26	2–2	76.92	52.33	3–2	82.43	64.44	4–2	94.74	73.97
1–3	91.46	75.76	2–3	91.45	51.94	3–3	85.20	70.46	4–3	96.12	80.00
1–4	88.15	75.50	2–4	74.01	76.71	3–4	95.59	63.93	4–4	91.37	80.38
1–5	91.82	77.68	2–5	82.64	59.52	3–5	78.50	70.34	4–5	87.10	58.70
1–6	80.99	69.28	2–6	83.70	53.47	3–6	88.03	55.38	4–6	84.87	80.63
1–7	79.56	80.00	2–7	78.45	59.87	3–7	95.63	63.22	4–7	88.03	67.76
1–8	85.71	73.33	2–8	83.56	65.56	3–8	98.26	64.20	4–8	92.74	72.78
1–9	86.16	72.49	2–9	72.47	68.32	3–9	94.83	54.19	4–9	83.15	74.50
1–10	79.63	76.44	2–10	83.33	59.52	3–10	76.67	79.58	4–10	74.06	76.29
All	88.11	72.55	all	80.17	60.41	all	85.96	64.97	all	86.56	74.11

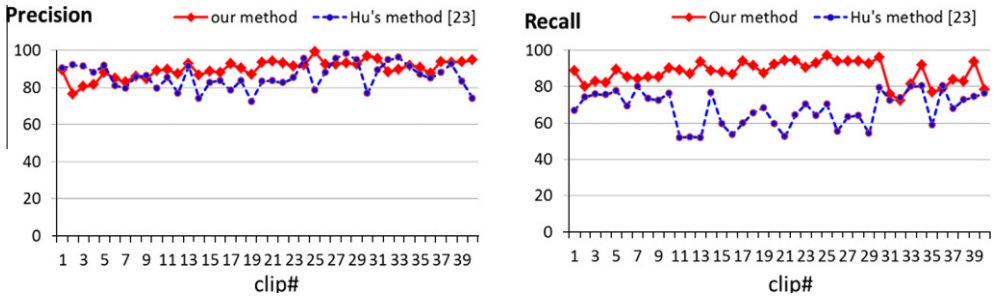

Fig. 19. The comparison on player tracking between Hu's method [23] and ours.

the well-known "error merge" problem, which means trackers lose their associated objects and falsely coalesce with other objects when two or more players of the same team occlude with each other. However, this comparison cannot assertively conclude which method outperforms the other, but shows that different methods have their advantage in different cases.

As for tactic analysis, three applications are introduced in [23]: (1) *Wide-open detection*: the relative/absolute player positions in the court model coordinates are used to detect *wide open* events. Wide-open means that some offensive player is not well defended by his/her opponents. (2) *Trajectory-based video retrieval*: the distance between a query trajectory (manually drawn) and each trajectory in the database is computed to retrieve similar video clips. (3) *Defensive strategy analysis*: based on player moving trajectories, the "stability" of defensive players is evaluated to classify the defensive strategy into *one-on-one defense* or *zone defense*. To realize the offensive strategy, we propose the screen pattern recognition method in this paper. Playing a fundamental and essential role in offensive tactics, *screen* is a blocking move performed by an offensive player, who stands beside or behind a defender, in order to free a teammate to shoot, to receive a pass, or to drive in for scoring. Hence, the visualized presentation of the player trajectories and recognized screen patterns on the court model assists the professionals or the audience in comprehending the offensive tactics executed in basketball games informatively and efficiently. It is hard to judge which tactic analysis application is more useful to the professionals or the general audience. Besides, increasing researchers have been devoted in this area. It can be anticipated that a wider variety of tactics analysis applications will be designed for basketball and even for more kinds of sports.

7. Conclusion

The more you know the opponents, the better chance of winning you stand. Thus, game study in advance of the play is an essential task for the coach and players. Tactic analysis provides informative and detailed insight of sports games but until now little work has been devoted to this topic. It is a growing trend to assist game study for intelligence collection in sports games with computer technology. To cater for this, in this paper we propose a screen-strategy recognition system capable of detecting and classifying screen patterns in broadcast basketball video. The proposed system automatically detects the court lines for camera calibration, determines the court region and extracts the players using color information. The extracted players are further classified and discriminated into the offensive and defensive teams. Then, the proposed system tracks players using Kalman filter and calibrates the players' positions to the real-world court coordinates by the homography matrix obtained from camera calibration. Analyzing the extracted player trajectories on the real-world court model, the system can detect and recognize the screen pattern on execution. Our experiments on broadcast basketball videos show promising results. Furthermore, the extracted player trajectories and the recognized screen patterns are visualized on a court model, which indeed facilitates the coach/ players or the fans understanding the tactics executed in basketball games.

In addition to the player trajectory, ball trajectory is also vital information for tactic analysis. Hence, we are currently working on integrating the framework of this paper with our previous work [18], which tracks the ball and reconstruct the 3D ball trajectory for shooting location estimation in basketball video, to build a powerful basketball video analysis system. It is our belief that the preliminary work presented in this paper will inspire extensive research and manifold applications of automatic tactic analysis and intelligence collection in various kinds of sports games.

Acknowledgments

This research is supported in part by projects ICTL-100-Q707, ATU-100-W958, NSC 98–2221-E-009–091-MY3, and NSC 101–2218-E-009–004. Many thanks to Dr. Hu Min-Chun for providing us with their system for performance comparison.

References

[1] L.-Y. Duan, M. Xu, Q. Tian, C.-S. Xu, J.-S. Jin, A unified framework for semantic shot classification in sports video, IEEE Transactions on Multimedia 7 (6) (2005) 1066–1083.
[2] M.-C. Tien, H.-T. Chen, Y.-W. Chen, M.-H. Hsiao, S.-Y. Lee, Shot classification of basketball videos and its applications in shooting position extraction, in: Proceedings IEEE International Conference on Acoustic Speech Signal Processing, 2007, pp. I- 1085–1088.
[3] M.-H. Hung, C.-H. Hsieh, Event detection of broadcast baseball videos, IEEE Transactions on Circuits and Systems for Video Technology 18 (12) (2008) 3829–3832.
[4] M. Fleischman, B. Roy, D. Roy, Temporal feature induction for baseball highlight classification, in: Proceedings ACM Multimedia Conference, 2007, pp. 333–336.
[5] C.-C. Cheng, C.-T. Hsu, Fusion of audio and motion information on HMM-based highlight extraction for baseball games, IEEE Transactions on Multimedia 8 (3) (2006) 585–599.
[6] J. Assfalg, M. Bertini, C. Colombo, A.D. Bimbo, W. Nunziati, Semantic annotation of soccer videos: automatic highlights identification, Computer Vision and Image Understanding 92 (2-3) (2003) 285–305.
[7] W. -T Chu, J.-L. Wu, Explicit semantic events detection and development of realistic applications for broadcast baseball videos, Multimedia Tools and Applications 38 (1) (2007) 27–50.
[8] G. Zhu, Q. Huang, C. Xu, L. Xing, W. Gao, H. Yao, Human behavior analysis for highlight ranking in broadcast racket sports video, IEEE Transactions on Multimedia 9 (6) (2007) 1167–1182.
[9] G. Zhu, C. Xu, Q. Huang, Sports Video Analysis: From Semantics to Tactics, Multimedia Content Analysis, Springer, 2009. pp.1–44.
[10] G. Zhu, C. Xu, Q. Huang, Y. Rui, S. Jiang, W. Gao, H. Yao, Event tactic analysis based on broadcast sports video, IEEE Transactions on Multimedia 11 (1) (2009) 49–67.
[11] G. Zhu, Q. Huang, C. Xu, Y. Rui, S. Jiang, W. Gao, H. Yao, Trajectory based event tactics analysis in broadcast sports video, in: Proceedings 15th ACM International Conference Multimedia, 2007, pp.58–67.
[12] G. Zhu, C. Xu, Y. Zhang, Q. Huang, H. Lu, Event tactic analysis based on player and ball trajectory in broadcast video, in: Proceedings of the 2008 international conference on Content-based image and video retrieval, 2008, pp. 515-524.
[13] J.-R. Wang, N. Parameswaran, Detecting tactics patterns for archiving tennis video clips, in: Proceedings IEEE 6th International Symposium on Multimedia Software Engineering, 2004, pp. 186–192.
[14] X. Yu, C. Xu, H.-W. Leong, Q. Tian, Q. Tang, K.-W. Wan, Trajectory-based ball detection and tracking with applications to semantic analysis of broadcast soccer video, in: Proceedings 11th ACM International Conference Multimedia, 2003, pp.11–20.
[15] X. Yu, H.-W. Leong, C. Xu, Q. Tian, Trajectory-based ball detection and tracking in broadcast soccer video, IEEE Transaction on Multimedia 8 (6) (2006) 1164–1178.
[16] X. Yu, X. Tu, E.-L. Ang, Trajectory-based ball detection and tracking in broadcast soccer video with the aid of camera motion recovery, in: Proceedings IEEE International Conference on Multimedia and Expo, 2007, pp.1543–1546.
[17] H.-T. Chen, H.-S. Chen, M.-H. Hsiao, W.-J. Tsai, S.-Y. Lee, A trajectory-based ball tracking framework with enrichment for broadcast baseball videos, Journal of Information and Science Engineering 24 (1) (2008) 143–157.
[18] H.-T. Chen, M.-C. Tien, Y.-W. Chen, W.-J. Tsai, S.-Y. Lee, Physics-based ball tracking and 3D trajectory reconstruction with applications to shooting location estimation in basketball video, Journal of Visual Communication and Image Representation 20 (3) (2009) 204–216.
[19] H.-T. Chen, W.-J. Tsai, S.-Y. Lee, J.-Y. Yu, Ball tracking and 3D trajectory approximation with applications to tactics analysis from single-camera volleyball sequences, Multimedia Tools and Applications (2011), http://dx.doi.org/10.1007/s11042-011-0833-y.
[20] S. Liu, M. Xu, H. Yi, L.-T. Chia, D. Rajan, Multimodal semantic analysis and annotation for basketball video, EURASIP Journal on Applied Signal Processing 2006 (2006) 1–13..
[21] Y. Zhang, C. Xu, Y. Rui, J. Wang, H. Lu, Semantic event extraction from basketball games using multi-modal analysis, in: Proceedings IEEE International Conference on Multimedia and Expo, 2007, pp. 2190–2193,.
[22] M.-H. Chang, M.-C. Tien, J.-L. Wu, WOW: wild-open warning for broadcast basketball video based on player trajectory, in: Proceedings ACM International Conference Multimedia, 2009, pp. 821–824.

[23] M.-C. Hu, M.-H. Chang, J.-L. Wu, L. Chi, Robust camera calibration and player tracking in broadcast basketball video, IEEE Transactions on Multimedia 13 (2) (2011) 266–279.
[24] T.-S. Fu, H.-T. Chen, C.-L. Chou, W.-J. Tsai, S. Y. Lee, Screen-strategy analysis in broadcast basketball video using player tracking, in: Proceedings IEEE International Conference Visual Communications and Image Processing, 2011, pp. 1–4.
[25] D. Farin, S. Krabbe, P. H. N. de With, W. Effelsberg, Robust camera calibration for sport videos using court models, in: Proceedings Storage and Retrieval Methods and Applications for Multimedia, 2004, pp. 80–91.
[26] J. Han, D. Farin, P.H.N. de With, Broadcast court-net sports video analysis using fast 3-D camera modeling, IEEE Trans. Circuits and System for video technology 18 (11) (2008).
[27] Y. Liu, S. Jiang, Q. Ye, W.Gao, Q. Huang, Playfield detection using adaptive GMM and its application, in: Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, vol.2, 2005, pp.421–424.
[28] G. Welch, G. Bishop, An Introduction to the Kalman Filter, Technical Report TR95-041, University of North Carolina at Chapel Hill, 1995.
[29] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Computing Surveys 38 (4) (2006).