VIVID TENNIS PLAYER RENDERING SYSTEM USING BROADCASTING GAME VIDEOS

Chieh-Li Chen, Jui-Hsin Lai, and Shao-Yi Chien

Media IC and System Lab
Graduate Institute of Electronics Engineering and Department of Electrical Engineering
National Taiwan University
BL-421, 1, Sec. 4, Roosevelt Rd., Taipei 106, Taiwan
chiehli.chen@gmail.com, juihsin.lai@gmail.com, sychien@cc.ee.ntu.edu.tw

ABSTRACT

Image-based rendering has been highly developed for its wide applications such as view synthesis and special effects in movies. In this paper, we proposed a tennis player rendering system synthesizing diverse player action/motion based on extracted database from broadcasting game videos. The system gathers database by retrieving the player from videos and synthesizes various kinds of player action/motion according to the user's instructions. The results show that the proposed rendering system can render smooth action/motion transition with satisfactory visual effect. For further applications, the proposed system can be used in interactive tennis games with image textures.

Keywords— motion rendering, tennis video, image-based rendering, morphing

1. INTRODUCTION

In recent tennis games, such as Wii Sports and PS3's TOP-SPIN3, characters of these games can react to users' instructions, move to the desired place and hit the ball. The interaction between characters and users make a giant leap in sports games which let users feel like they are the character who runs on the tennis court and performs all the rallys and volleys. However, textures of these games are created by computer graphics, which result in discontinuous motion transition and degrade the visual effect of tennis games. To provide more real tennis players and overcome artificial visual effect, our system creates vivid tennis players using broadcasting game videos and constructs various but also smooth player action/motion based on video synthesis techniques after receiving instructions from users.

For video synthesis, many previous works can be found in literatures. In previous works, Schödl *et al.* [1], presented a novel medium named Video Textures, which extracted texture of a video and synthesized new video clips by rearranging frames in the original video sequences. After that, Colqui *et al.* [2] extended the work to flocking behavior. By modifying the cost function in [1] and adding new constraints such as separation, alignment or cohesion, the flocking behaviors are simulated. Both of works presented a marvelous scheme to create new motions by displaying the original video clips in a different order. However, most of the character in their new videos are nature scenes, such as swinging grass and waterfalls, or small animals, like hamsters, flies or fish. No human video clips are used as the character of video clips because human motions are more complicated. In [3], Efros *et al.* introduced a template sequence and performed two forms of data-based action synthesis "*Do as I Do*" and "*Do as I Say*". The results are impressed, however, it can only synthesize actions based on the template of existing motion sequences. In [4], a photo-realistic animations of human movement was presented. The work overcomes many challenges encountered in synthesizing human actions; nevertheless, to gather the data of human video textures, lots of sensor need to be pasted on a person's body for motion detection, which makes data gathering complicated and increases costs.

Inspired by [1] and trying to render player action/motion without a template, we proposed a tennis player rendering system, which consists of two parts: database generator and player motion rendering system, as shown in Fig. 1. The database generator analyzes broadcasting game videos for foreground objects extraction, normalizes size of each player and classifies player actions into hit and motions. After data are gathered completely, the system receives instructions from users and renders corresponding action/motion based on the proposed player action model and smooth motion transition by morphing. For experiments, different game videos are used. To verify the results, subjective evaluation are performed, 32 subjects were asked to evaluate the visual effect of the rendering result. Among different rendering methods, the proposed method get the highest rank than other approaches and the result shows that most of the subjects thought the rendering results were smooth and satisfactory. Furthermore, we apply our proposed system to build a tennis game. The proposed rendering system can render player action/motion using image textures and present smooth motion transition, which provides different visual effect from general tennis games rendering with graphic textures.

This paper is organized as follows. The database generator is introduced in Sec. 2, including foreground object extraction, player size normalization, player action analysis and classifi-

978-1-4244-7493-6/10/$26.00

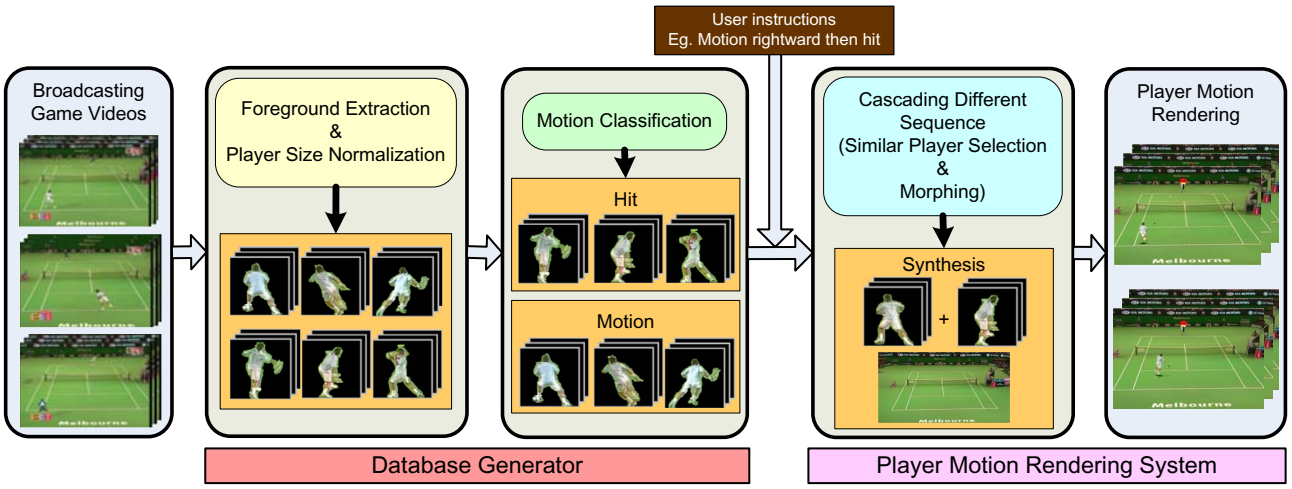


Fig. 1. Flow of the database generator and player motion rendering system.

cation, and player motion path selection. Next, player motion rendering is described in Sec. 3, involving a smart cascading cost function and a modified morphing technique. In Sec. 4, the experimental results are presented. Finally, an conclusion is given in Sec. 5.

2. DATABASE GATHERING

In this section, framework of the database generator is going to be described. Database generator is consisted of foreground object extraction, player size normalization, motion classification and minimal path selection. Details of each part of database generator will be described below.

2.1. Foreground Object Extraction

The first step of the player rendering system is to collect player database from game videos. In order to gather every action/motion of the player, we not only need to segment the player from video clips but also need to record the position of the player on the court in each frame. As the process begins, a fiducial plane generation [5] is adopted for foreground object extraction. Fiducial plane is the tennis field in a fiducial coordinate system in bird's eye view. By warping the fiducial plane, a clear field without foreground objects is constructed, and players can be extracted by simply computing the difference between the video frame and the clear field. Next, the point (x_v, y_v) where the player is currently standing on is recorded as the position of the player. We let x_v equal to the one of center of mass and y_v be the lowest point within the player. For a global position (x_f, y_f), we transform this position onto the fiducial plane by (1), where (x, y) stands for the point in video frame, (x', y') is the point in fiducial plane and m_0 to m_7 is the parameter of perspective projection matrix.

$$x' = G_x(x, y) = \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1},$$

$$y' = G_y(x, y) = \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1}. \quad (1)$$

2.2. Player Size Normalization

Since players are extracted from video clips, the size of the segmented players will be zoomed by cameras and influenced by the position of players on the court. Various player size makes the rendering process complex and difficult. Thus player size normalization is proposed. To normalize the size of the player, two factors are considered: the zooming parameter of the camera in each frame and the varying player size due to different positions on the court. The parameter of zooming rate of the camera between current video frame and the fiducial plane can be estimated by calculating the deviation of $G_x(x, y)$ and $G_y(x, y)$ respect to x and y respectively. By calculating the deviation of these two functions, a camera zooming function on x coordinate and y coordinate are obtained. Since we need to normalize player's size, $\frac{\partial G_x(x,y)}{\partial x}$ and $\frac{\partial G_y(x,y)}{\partial y}$ are multiplied together to get the final normalization function. Then, considering the player position on the video frame, we put (x_v, y_v) into (2) to get the final zooming value, z_{vf}, as shown in Fig. 2.

$$z_{vf} = \frac{\partial G_x(x, y)}{\partial x} \cdot \frac{\partial G_y(x, y)}{\partial y} \mid_{(x,y)=(x_v, y_v)} . \quad (2)$$

2.3. Player Action Analysis and Classification

Gestures and activities of a player are various and continuous, it is hard and difficult to gather every kind of data for vivid player rendering. Observing that after the hitting event, the player will reconsider the strategy and decide the next step all over again, we can separate each motion by every hitting event even though transitions between actions and actions are subtle to discover. Moreover, player's action/motion are more complicated than action/motion of hamsters, subjects in [1], and thus difficult to render new action/motion without a organized model. To

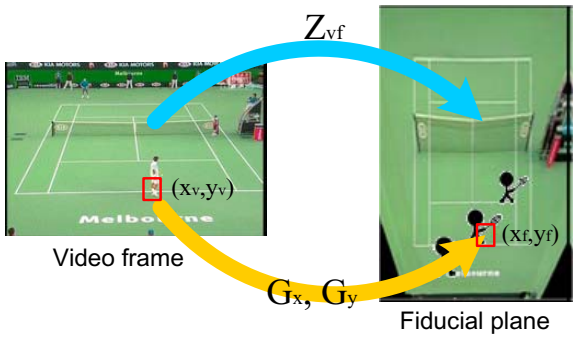


Fig. 2. Illustration of player position transformation and player size normalization.

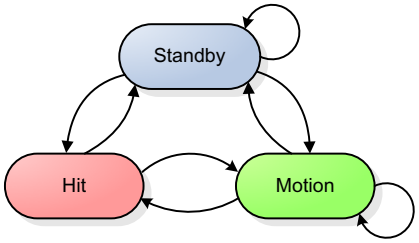


Fig. 3. Player action model.

solve this problem, we construct a player action model and separate action/motion of the player into three categories, which are *standby*, *motion* and *hit*, as shown in Fig. 3, where arrows stand for allowable state transitions. All actions/motions of the player can be composed by these states, for example, a player may move toward right, wait for the ball to come, then perform a forehand swing to hit the ball, which can be modeled as the composition of motion(rightward), standby, and finally, hit(forehand swing).

Under the conception, to classify different actions into correct categories, the first step is to find hit gestures and then actions between two hit gestures will be considered as moving motions. In order to detect hit gestures automatically and more accurately, the system takes both the shape of players and other information from the court such as the ball trajectory into account. In [5], the hitting gestures are detected when the distance between the ball and the player is less than the threshold. Here we only use two categories, forehand swing and backhand swing, to classify various hitting actions for two reasons. First, smashes seldom appear and data are difficult to gather. Second, rally and volley look similar and are hard to tell them apart. While the player performs either one of them, the action looks like a simple swing on the screen, and thus, we simplify the hitting action categories and merge these two hitting actions into forehand swing and backhand swing. After the hitting gesture is detected, an optical-flow algorithm [6] is applied. Calculating and summing the value of each optical flow on the right side and left side of the player. If flows on the right side of the player is larger then the left side, then the hitting gesture is regarded as forehand swing. On the contrary, the hitting gesture is regarded as backhand swing. Actions between two hitting gestures can then be classified as moving motion. Considering the relative position of player in the first frame and last frame of the motion sequence, the moving direction can be decided.

2.4. Player Motion Path Selection

Gathering various motions and hitting gestures of a player, the database is complete and large. However, too many data results in longer rendering time and lagged video display due to finding the most similar motion successive sequence. To reduce the rendering time, a well-organized database is needed. Here the most well-organized database stands for minimum database which can render the most abundant and diverse player actions.

To speed up the rendering process, a smaller moving database is to be selected. We focus on the moving motions since synthesizing different motions together to render new motions is the main part of the rendering system. To find the minimal spanning path, the dispersion and moving distance of combinations of existing paths are considered. The more various directions there are, the more new paths are able to be rendered. Obtaining player position on the fiducial plane from Sec. 2.1, a moving vector $\mathbf{v}_i(x_i, y_i)$ is used to record direction of the motion. Direction and distance of motions are derived as

$$\theta_i = \arctan \frac{y_i}{x_i}, \tag{3}$$

$$|\mathbf{v}_i| = \sqrt{x_i^2 + y_i^2}, \tag{4}$$

where θ and $|\mathbf{v}_i|$ denotes the argument and the magnitude of motion vector respectively. Knowing the direction and distance of each path, the dispersion of paths on the court can be estimated. Here, 360 degrees of moving direction are segmented into small bucket with 30 degree per bucket. Moving motions are classified into different buckets according to their arguments. Dispersion is calculated as (5):

$$\text{Dispersion}(\mathbf{R}') = \frac{\sum_{i=1}^{n} (\theta_i' - \bar{\theta})}{n}, \theta_i' \in \mathbf{R}', \mathbf{R}' \subseteq \mathbf{B}_p,$$

$$30(p-1)^\circ \leq deg(\text{bucket}_p) < 30p^\circ,\ 1 \leq p \leq 12, \tag{5}$$

where $\mathbf{B}_p$ is the set of paths whose argument are between $30(p-1)^\circ$ and $30p^\circ$, $\mathbf{R}'$ is the subset of $\mathbf{B}_p$, in which n paths are selected randomly, and $\bar{\theta}$ stands for the average degree of $\mathbf{R}'$. Then the dispersion of the combination paths can be estimated by their corresponding variance.

Besides the dispersion of each combination of moving motions, distance of each path is also taken into consideration. In addition, the tortuosity of each path is also estimated for we would like to keep the path where player directly move to destination without winding around. Finally, by combining the result with dispersion, the combination of motions with the largest degree variance and the longest moving distance was selected as the minimal spanning paths.

3. PLAYER MOTION RENDERING

After gathering intact data of player action/motion and receiving instructions from user, the rendering system can synthesize different clips to generate actions/motions which fit in with user's requirement. Since the action/motion is composed of different video clips, if the system cascades two arbitrary sequences together, the new sequence will look unreasonable. Thus, we introduce a cost function, which defines the similarity between two sequences and helps rendering system to find suitable candidate to cascade together. Even though finding the proper successor, directly cascades them will result in discontinuities between two clips. Therefore, we introduce morphing technique to smooth the transition process. The details of these two topics are described in following.

3.1. Cascading Cost Function

There are more than one sequence in each category, and thus, when it comes to state transitions, the system needs to find a proper successor to cascade behind the current sequence, such as finding an adequate hitting action to cascade after current moving motion. To find the suitable successor, computing the similarity between all pairs of frames in the database is indispensable. In our current implementation, the similarity D_{ij} between frame i and frame j is defined based on texture, shape and direction of each action sequence, written in

$$D_{ij} = c_t D_{ij,tex} + c_s D_{ij,shape} + c_v D_{ij,v}. \quad (6)$$

$$D_{ij,tex} = \sum_{y=1}^{H} \sum_{x=1}^{W} |I_i(x,y) - I_j(x,y)|^2. \quad (7)$$

$$D_{ij,v} = \mathbf{v}_i(x_i, y_i) \cdot \mathbf{v}_j(x_j, y_j). \quad (8)$$

$D_{ij,tex}$ is the texture similarity of players. The system compares the similarity of texture by computing two frames' color difference pixel-by-pixel. $D_{ij,shape}$ stands for distance of shape. Besides texture, we also compute the shape distance of each frames using Hausdorff distance [7], trying to find the successor with similar shape for better visual effect. $D_{ij,v}$ is the distance of moving direction. Since a motion clip may not be long enough for the player to move to desired position, moving direction is also under consideration to cascade different sequences and let the player move to destination. These features are combined together by coefficient c_t, c_s, and c_v to compute the similarity D_{ij}.

Receiving the request from user, such as moving from the bottom-left corner to top-right corner in the court, the system searches for a proper motion sequence in its database according to the direction. If the chosen sequence is long enough, the task is complete and the system is ready for the next request, otherwise the chosen sequence requires a successor. Rendering system finds the sequence as the successor whose first frame has the smallest D_{ij}.

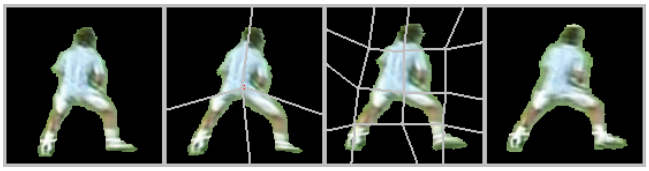


Fig. 4. Morphing process. (a) The last frame in first sequence, (d) the first frame in second sequence, (b),(c) hierarchical morphing process.

3.2. Smooth Different Sequence Cascading

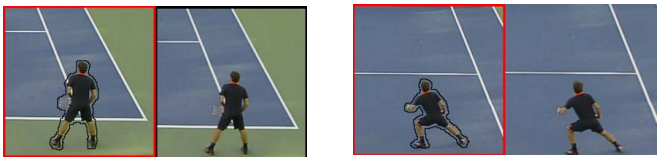
Although we have gathered an almost complete database, the successor still may not be a perfect match to the current sequence. If the system cascades them directly, the synthesized motion will look weird and jerky. To overcome this problem, the morphing technique is adopted to help cascade two sequences together and make a more pleasing and smoother transition process. Inspired by [8], a hierarchical and tile-based image warping scheme is used, as shown in Fig. 4. Each grid is labeled as different levels and modified in a hierarchical order. Different grid position changing order results in different morphing consequence, and thus, in order to find the proper changing sequence, we use gradient descent to detect more important area, choose it as the grid position changing candidate and determine its new position first. After the grid is moved to its new position, a transfer function is constructed. The reference pixel value can be found through backward mapping. We use bilinear interpolation to interpolate the new pixel value $\tilde{I}(i,j)$ and use color difference between morphing image and reference image as the cost function to judge the morphing results, written in the first term of (9).

In [8], in some cases, the morphing process needs user-specified anchor points for better morphing result. With the help of Scale Invariant Feature Transform [9], anchor points can be found automatically. The cost function is modified as (9), where (x_k, y_k) refers to the position of the kth feature point in the original image, $(\tilde{x_k}, \tilde{y_k})$ refers to the position of the kth feature point in the morphed image. By aligning n anchor points, and a better result is presented. Finally, we also smooth the moving direction and velocity by interpolating intermediate speed for smooth motion transition.

$$W = \sum_{j=1}^{H} \sum_{i=1}^{W} |I(i,j) - \tilde{I}(i,j)|^2 + \lambda \sum_{k=1}^{n} |(x_k, y_k) - (\tilde{x_k}, \tilde{y_k})|^2. \quad (9)$$

4. EXPERIMENTAL RESULTS

In this paper, we record the four Grand Slam and use broadcasting videos as database for the proposed rendering system. To simplify the rendering process, we normalize player size as mentioned in Sec. 2.2. The results of size normalization are satisfactory, as Fig. 5 shows. In Fig. 5(a), the original size of the



(a) Reduction (b) Magnification

Fig. 5. The results of player size normalization. The original size is marked by red rectangle, and right ones show the player size after normalization.

Table 1. Precision and recall rate of motion classification.

	Forehand swing	Backhand swing	Motion
Event Number	23	12	74
Precision	100%	100%	96.1%
Recall	96%	92%	98.6%

player is larger than the standard size (shown in the left image), as a result, the modified figure (shown in the right image) reduce the size of the player. On the other hand, in Fig. 5(b), the player is smaller than the standard size, and thus the player is magnified. In both cases, either to magnify the smaller player to a larger one, or to reduce the larger player to a smaller one, edges and details of the player are preserved and clear.

As mentioned in Sec. 2.3, after a hitting gesture is detected, an optical-flow algorithm is applied to judge forehand swing or backhand swing. The motion between two hitting gestures is regarded as moving motion. The precision and recall of motion classification are shown in Table 1, which are satisfactory by having a percentage over 92%.

In motion synthesizing, part of three synthesized video clips are shown in Fig. 6. Fig. 6(a) is synthesized by Video Textures [1], which provides a bad motion transition effect because it cascades different video clips without a player action model and thus the player changes his action/motion suddenly. Fig. 6(b) finds the suitable successor with our player action model and uses fade-in/out to cascade two sequences. To find a suitable successor, the coefficients in cascading cost function c_d, c_s, and c_v are chosen as the inverse of the largest value of $D_{ij,tex}$, $D_{ij,shape}$ and $D_{ij,v}$ with the same i to normalize $D_{ij,tex}$, $D_{ij,shape}$ and $D_{ij,v}$ to the same order. Moreover, c_v equals to 0 when state transits from motion to hit/standby, and equals to a nonzero constant when it transits from hit/standby to motion. The result shows a better visual effect because of reasonable motion transition but fade-in/out method let the transition looks weird. Fig. 6(c) finds successor by our player action model and cascades two sequences using our proposed method. With our method, player in the first image gradually change to the second image and provides a good state transition.

To justify the visual effect of motion rendering, a survey is designed and 32 subjects are asked to rank these video clips together with the one produced by directly cascading according to

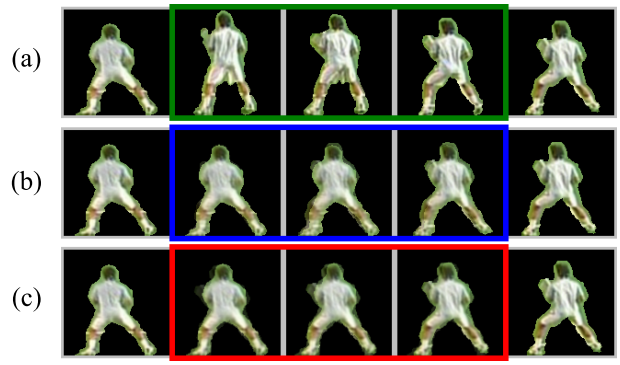


Fig. 6. The results of different cascading methods. In each row, left-most image stands for the last image in first sequence, right-most image stands for the first image in second sequence and between them are the transition results using different synthesizing methods. (a) Using Video Textures. (b) Proposed player action model without morphing. (c) Proposed player action model with morphing.

Table 2. Average ranking of motion sequence with different cascading method (the lower the better).

	Video Textures	Cascade Directly	Fade-in/out	Proposed method
Average Rank	4.00	2.53	1.53	1.43

their smoothness at cascading points, where 1 means the highest rank and 4 means the lowest rank. As Table 2 presents, motion rendering with player action model provides better visual effect, and cascading sequence with the proposed method shows the smoothest transition.

With the method mentioned in Sec. 2.4, we reduce the motion database from 56 motion paths to 30 motion paths(with $n = 3$). Another survey is performed to the same subjects. In the second survey, we render new motions using the original intact database and the selected database. Subjects are asked to judge whether which rendering result shows better rendering quality not only based on smooth transition but also on the moving path of the player. The result shows that rendering with smaller database gets rank 1.22 while with larger database gets rank 1.50, which reveals that rendering motion with smaller database performs the same as or better than with larger database. As containing less similar candidates, the smaller database keeps the motion structure of the original database but has less redundancy, and thus, provides accurate motion rendering. For more results of rendering with different size of database, please refer to our website.[1]

Finally, we apply the proposed system to build a tennis game. Users give instructions to control the player, and the system renders the corresponding action/motion immediately. As shown in Fig. 7. While rendering player action/motion, we find that tran-

[1] http://media.ee.ntu.edu.tw/~chiehli/tennis_player.html

Fig. 7. Illustration of our tennis game built based on the proposed player rendering system.

sitions between two sequences are rapid, and users can not actually tell the difference between each cascading methods. Thus, although the proposed cascading method gets the highest rank, it is only higher than Fade-in/out with 0.1, which indicates that they are similar to subjects. However, with the rapid motion transition, both of the cascading methods provides smooth motion transition and satisfactory visual effect. The results of the tennis game also can be seen on our website.

5. CONCLUSION

In this paper, we have presented a tennis player rendering system which is composed of the methods of database collection, player model construction, and player action/motion rendering. According to users instructions, the proposed rendering system synthesizes the corresponding player action/motion. As subjective evaluation shows, the proposed rendering results gets higher score than the existing approaches, which means that the system synthesizes smooth player action/motion and provides satisfactory visual effect. Furthermore, we also apply the proposed system to build a tennis game. Unlike general tennis games using graphics as their textures, our tennis game provides vivid player action/motion with image textures.

6. REFERENCES

[1] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa, "Video textures," in *Computer Graphics*, 2000, pp. 489–498.

[2] G. Colqui, M. Tomita, T. Hattori, and Y. Chigusa, "New video synthesis based on flocking behavior simulation," in *International Symposium on Communications, Control and Signal Processing*, March 2008, pp. 936–941.

[3] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *International Conference on Computer Vision*, 2003, pp. 726–733.

[4] M. Flagg, A. Nakazawa, Q. Zhang, S. B. Kang, Y. K. Ryu, I. Essa, and J. M. Rehg, "Human video textures," in *Interactive 3D Graphics and Grames*, 2009, pp. 199–206.

[5] J.-H. Lai and S.-Y. Chien, "Tennis video enrichment with content layer separation and real-time rendering in sprite plane," in *IEEE, Multimedia Signal Processing*, Oct. 2008, pp. 672–675.

[6] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, pp. 221–255, Feb. 2004.

[7] D.P. Huttenlocher, G.A. Klanderman, and W.A. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 850–863, 1993.

[8] P. Gao and T. W. Sederberg, "A work minimization approach to image morphing," *The Visual Computer*, vol. 14, no. 8, pp. 390–400, 1998.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov. 2004.