Pose Estimation for Evaluating Standing Long Jumps via Dynamic Bayesian Networks

Hui-Huang Hsu[1], Yao-Bao Yen[1], Chu-Song Chen[2], and Chun-Ta Ho[3]
[1]*Department of Computer Science and Information Engineering, Tamkang University*
[2]*Institute of Information Science, Academia Sinica*
[3]*Department of Multimedia and Animation Arts, National Taiwan University of Arts*
Taipei, Taiwan, R.O.C.
E-mail: h_hsu@mail.tku.edu.tw

Abstract

A system is developed for analyzing poses in a standing long jump automatically. In the system, silhouette of the jumper is segmented from the background first. A thinning algorithm is then used to find a rough skeleton from the silhouette. Some image processing techniques are applied to make the resulted skeleton smoother and simpler. Key points are extracted from the skeleton. Finally, the dynamic Bayesian network (DBN) is used to determine the corresponding pose from the key points. The experimental result shows that pose estimation accuracy is quite good. According to the standing long jump standards, incorrect movements at different stages of the jump can thus be identified.

Keywords: *Motion analysis, Pose estimation, Thinning, Dynamic Bayesian networks, Standing long jumps*

1. Introduction

In physical education, doing the standing long jump in the right way is one important development of sports skills for primary school students. A video clip with a complete standing long jump contains about 40 frames. It is costly to have the teacher watch a lot of such video clips to determine bad movements of certain students so that advices can be provided to the students. There is a lot of work related to motion analysis of sports video in the literature. But, none of previous research worked on this problem. It is desired to have a system that can estimate the poses during the jump from a video clip automatically. With the determined poses in all the frames, bad movements can thus be identified. Such a system can further be used as a tutor for the student to do self-training. This problem should be easier than other motion analysis problems, like golf swinging and ball throwing since it mostly needs only 2D information from the side. Standing long jump video clips can be taken from the left-hand side of the jumper. This 2D information is sufficient to judge the movement of a jump. On the other hand, both golf swinging and ball throwing need 3D information. Either the 3D information is extracted from scenes taken by one video camera, or two video cameras are used to capture the movement from different angles at the same time.

In our previous work, the genetic algorithm was used to construct a skeleton from the extracted silhouette of the jumper [1]. The skeleton is defined by a stick model with the major joints of the human body. And it represents the pose of the jumper in each frame very well. However, the size of each stick needs to be given by the user beforehand. Also, the search process of the genetic algorithm is very time-consuming. Therefore, the thinning algorithm is utilized instead in this paper [6]. The thinning algorithm is much simpler. Although the generated skeleton is somewhat rough and not as precise as the predefined stick model, the result still can provide meaningful information about the pose. This information is then further processed by the DBN [2], which was not done in our previous research.

There are three parts of such a system: (1) human detection, (2) pose estimation, and (3) scoring [1]. In this paper, we focus on the second part – pose estimation. It is divided into two stages. In the first stage, the Z-S thinning algorithm is used to find a rough skeleton. Key points of the skeleton are extracted. Coded information of the relative positions of the key points is then used as the input to the DBN to do pose classification.

The rest of the paper is organized as follows. How the human object is extracted from the background is described in Section 2. Skeleton determination by the thinning algorithm is presented in Section 3. Pose classification by the DBN is delineated in Section 4. Experimental results are shown in Section 5. And finally a brief conclusion is drawn in Section 6.

1545-0678/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDCS.Workshops.2008.72

2. Object extraction

A simple algorithm modified from the one in [5] is used here to extract the human object. The algorithm was designed to do object tracking. The advantage of this method is that it is simple and fast. Suppose that the input image is N×N. The algorithm is described in the following.

i. Produce an average background matrix B_{ave} over a moving window of n×n.

$$B_{ave}(i,j,k)=\left(\frac{1}{n^2}\right)\sum_{i=i-(n-1)/2}^{i+(n-1)/2}\sum_{j=j-(n-1)/2}^{j+(n-1)/2}B(i,j,k)$$

where i and j denote the coordinates of each pixel and k = 1, 2, and 3 correspond to R, G, and B intensities, respectively.

ii. Form the average matrix A_{ave} with the same window size using the image with moving object A.

$$A_{ave}(i,j,k)=\left(\frac{1}{n^2}\right)\sum_{i=i-(n-1)/2}^{i+(n-1)/2}\sum_{j=j-(n-1)/2}^{j+(n-1)/2}A(i,j,k)$$

iii. Subtract background image B_{ave} from the image with moving object A_{ave} to get matrix C.

iv. Add absolute differences of R, G, and B intensities in each pixel in the resulted matrix C to form the foreground matrix D.

$$D(i,j)=|C(i,j,1)|+|C(i,j,2)|+|C(i,j,3)|$$

v. Find the maximum value in matrix D.

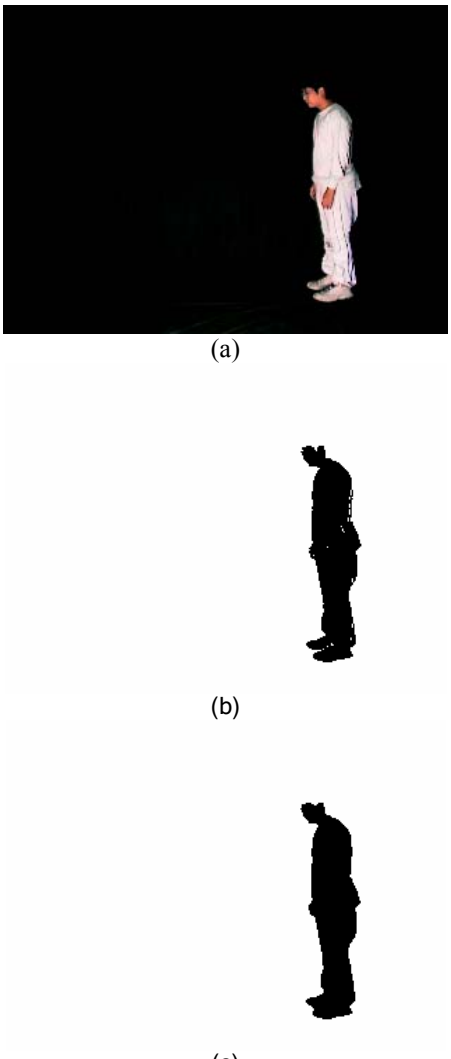
vi. Subtract a constant value from each pixel so that the maximum value of D becomes 255.

vii. If there is a negative number in the resultant matrix in the previous step, make it zero and form a new matrix R.

viii. If any number in R is greater than the threshold – ***Th_Object***, set matrix ***Obj*** to 1. Otherwise, set it to 0. The value of ***Th_Object*** is 20 here.

$$Obj(i,j)=\begin{cases}1, if\ R(i,j)>Th\_Object\\0, else\end{cases}$$

In this research, the video clips of standing long jumps were taken in a studio with a black background. By doing so, the light sources can be controlled and are more stable. An example of extracted object by the above-mentioned algorithm is shown in Figure 1(b). Some small holes and ridged edges exist in the extracted object. So the result is further smoothed by a median filter and the smoothed silhouette is shown in Figure 1(c).

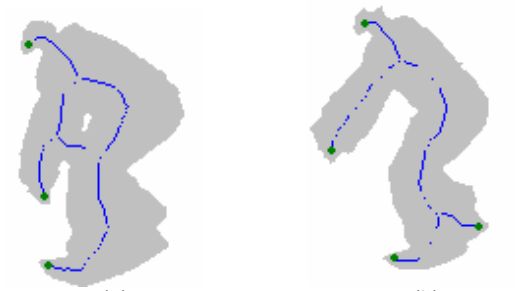


(a) (b) (c)

Figure 1. (a) Input video frame (b) Extracted silhouette (c) Smoothed silhouette

3. Skeleton extraction

To obtain pose information from the silhouette, a representative skeleton with key points is desired. In order to have a more efficient system, a thinning algorithm is chosen for producing such a skeleton. Here, the Z-S algorithm is used. The Z-S algorithm uses a peeling approach. It is fast and it can avoid the break-line problem. Interested readers are referred to [6] for details of the algorithm. However, there are still common problems for the thinning algorithm. First of all, it can result in loops, corners, and redundant line segments (Figure 2). Also, it is sensitive to noise.



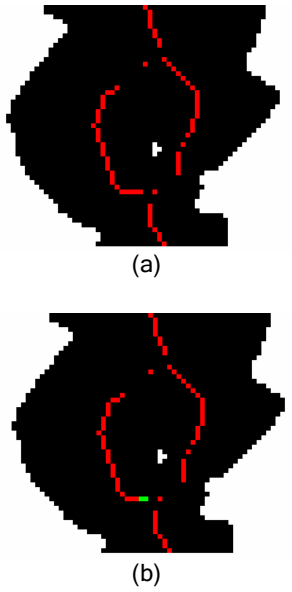
(a) (b)

Figure 2. (a) Loop (b) Corner and redundant line

To conquer the above-mentioned problems, methods in [7] are used to produce a better skeleton. First, the thinning result is converted into a graph. Then, the adjacent junction vertices are removed from the skeleton graph. An *adjacent junction vertex* is the vertex that has more than one junction vertices in its eight neighbors. This procedure can remove redundant vertices to make sure the degree of each node in the graph is less than or equal to 4 to simplify the graph structure. The result is a simplified graph version of the original thinning result. Figure 3(a) presents one example. Broken lines can be seen because adjacent junction vertices are removed.

Next, to avoid loops in the skeleton graph, a "maximum" spanning tree is built. It is similar to the minimum spanning tree, but maximum length instead of minimum length is chosen while the tree grows. Loops can be cut with this procedure. Because some adjacent junction vertices are removed in the previous step, maximum length instead of minimum length should be found here to make sure the new junction vertex can connect to all of its neighbors. There exists a loop in Figure 3(a). The result from the aforementioned loop-cut procedure is shown in Figure 3(b). A green dot is used to separate the red lines at the lower-left corner of the loop. In most case, junction vertices are the intersection points between body parts, like "head and hand" and "hand and foot."

Finally, we need to prune the branches caused by noise on the resulted silhouette in the thinning step. A branch is defined as a simple path from an end vertex to a junction vertex. If the branch consists of less than 10 vertices, it might be a noisy (redundant) branch and needs to be deleted. (See the lower-right corner branch of Figure 2(b).) Only one branch can be deleted at a time. Otherwise, both the noisy branch and the correct branch could be removed at the same time. Figure 4 illustrates such pruning results. Figure 4(c) is the desired result. Two more examples for thinning results are also given in Figure 5. These results show that the postures of the silhouettes can be properly represented.



(a)

(b)

Figure 3. (a) A loop (b) Loop cut

(a)

(b)

(c)

Figure 4. (a) Before pruning (b) Delete both branches (c) Delete only the noisy branch

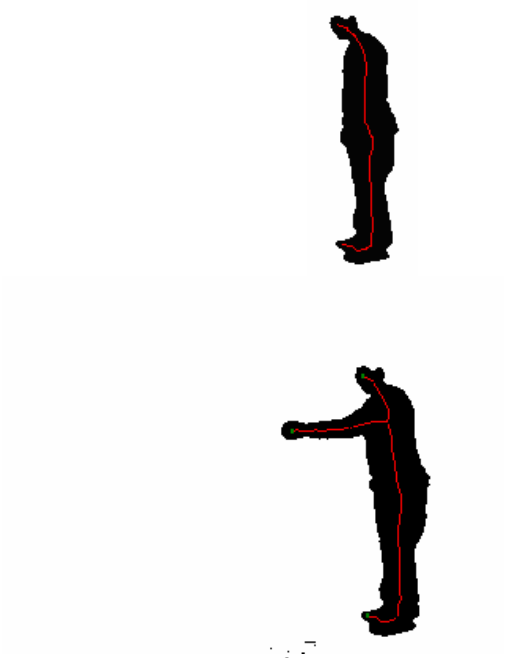


Figure 5. Examples of thinning results

4. Pose estimation

Poses are to be determined by the DBN in this research. The Bayesian network (BN) is a graph that obeys Bayes' theorem. It is also called the graphical model. The BNs are directed acyclic graphs. Each node represents a random variable and each edge indicates that the connected nodes are causally related. Each node has a discrete variable. Inference of a BN is the procedure to derive the states of one part of the nodes when the states of the other part of nodes are known. On the other hand, learning is the procedure to learn the random variable of the nodes when the behavior of part (or all) of the node is observed. DBNs are the kind of BNs that model sequences of variables. For a detailed discussion of DBNs, please refer to [9].

To use BNs to recognize poses, it needs to be trained first. There are two categories of training: qualitative training and quantitative training [8]. Qualitative training concerns the network structure of the model and quantitative training determines the specific conditional probabilities. In our case, the input to the BN is the skeleton information. This information is encoded with the locations of the key points of the skeleton on the eight areas of the plane. Two examples are shown in Figure 6. The output is a predefined pose.

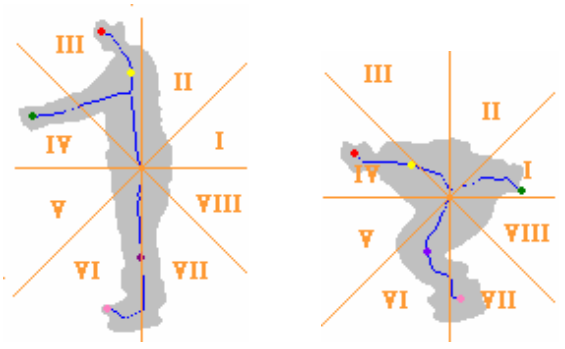


Figure 6. Examples of feature encoding of the key points on a skeleton

Each BN consists of eight observed nodes (Areas I ~ VIII), five hidden nodes (Head, Chest, Hand, Knee, Foot), and one root node (Pose name). But this is not enough. More information like previous pose is needed. As in [4] and [8], several BNs are used to decide if a certain event happens. Pose information of previous frame is input into the DBN because it is crucial to the pose of the current frame. Furthermore, there are four stages in a jump. They are *before jumping*, *jumping*, *in the air*, and *landing*. They can be used to help determine the poses. For example, poses belonging to "before jumping" and poses belonging to "landing" cannot occur consecutively because it does not exist in real cases. So a jumping stage flag is necessary. The structures of one BN and its DBN are shown in Figure 7. Part (a) of Figure 7 presents a BN for the "standing & hand swung forward" pose. There are totally 22 defined poses in our work. Part (b) of Figure 7 adds the influence of the current jumping stage and the previous pose on the determination of the current pose.

4.1. Training phase

When the first frame enters, we reset the jumping stage to "before jumping" and the current pose to "standing & hand overlap with body." In the later frames, the previous pose is the predicted pose in the previous frame. Similar poses from the "before jumping" stage and the "landing" stage can be distinguished by the jumping stage flag. At every training phase, we input the locations of Head, Hand and Foot, respectively. Next, the path from Head to Foot is used as the torso, and the waist location can be estimated. The waist location is set to be in the middle of the torso. Then the waist is put as the origin and the locations of all key points (Head, Chest, Hand, Knee or Foot) on the eight areas of the plane can be determined (Figure 6). These locations are

combined as the feature vector. Once the feature vector is received, the DBN can update the relation strength between the current pose and the previous pose.

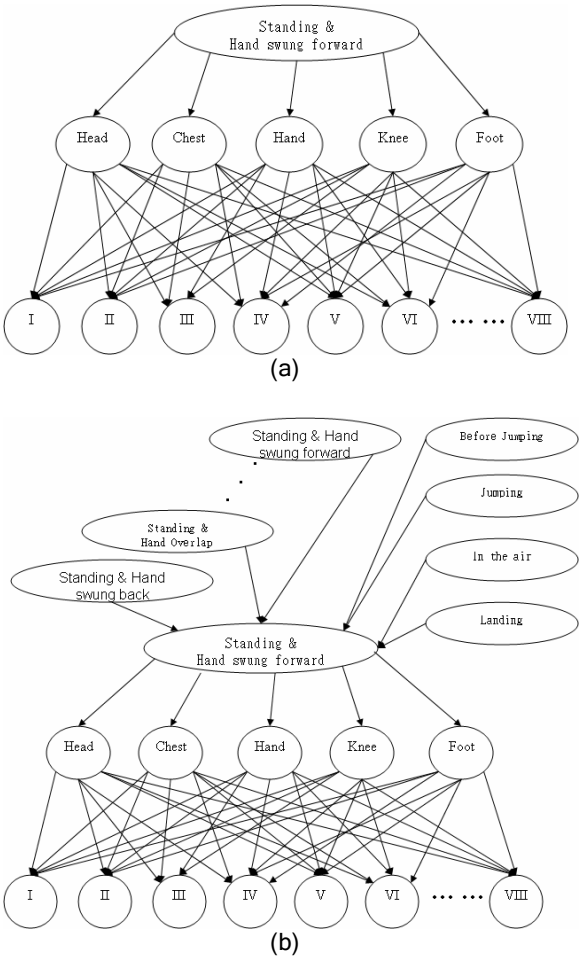


Figure 7. (a) Bayesian network structure
(b) Dynamic Bayesian network structure

4.2. Testing phase

When the first frame enters, we do the same thing as when the first frame is entering the training phase. And we set the lowest point to be Foot because no matter what pose it is Foot is always the lowest point. Then we try to assign body parts to other key points and to combine them as the feature vector to infer the most probable pose. Once the feature vector gets the maximum probability, we say that the pose represented by this feature vector is the most fitted pose.

One thing needs to be discussed next is that different poses in the training samples do not appear equally. For example, "Standing & hand swung forward" appears most of the time. But "Knee and foot extended & Hand raised forward" and "Waist bended & Hand raised forward" may appear much less frequently than "Standing & hand swung forward." Thus a threshold called Th_Pose is set for each pose other than "Standing & hand swung forward" to emphasize more on these poses. Otherwise, "Standing & hand swung forward" would dominate the decision making. If the probability of certain pose is greater than Th_Pose, we can say that the pose appears. When moving to the next frame, the current pose will be input to the next frame as the previous pose.

With a proper training on the DBN, the trained networks can find the most probable pose for each frame. The poses in consecutive frames can then be used to identify incorrect movements.

5. Experimental results

Twelve video clips are used as the training set and three others are used as the test set in our experiments. Totally, there are 522 frames in the training set and 135 frames in the test set. The representative frames of one of the test video clips are presented in Figure 8 to demonstrate the thinning results. The extracted skeletons represent their respective poses pretty well. Feature vectors from key points in these extracted skeletons are then used as input to the DBN to determine poses. Poses in most frames of the three test video clips can be correctly classified by the DBN. The overall accuracy is from 81% to 87% for the three test video clips.

This resulted accuracy is good. But there is still room for improvement. One reason for such a not-so-satisfied result is that the number of training samples is small. The probabilities of these poses are not large enough to be accepted. Moreover, when an "Unknown" or a misclassification appears, it will affect the inference of the subsequent frame. So the previous pose for the next frame should be set to the pose that is recognized most recently instead of "Unknown" for each DBN. From our experience, this is really useful in dealing with the problem with unknown poses. But a misclassified frame will still affect the classification of its subsequent frames. Most errors in our experiments occurred in consecutive frames.

6. Conclusion

We have implemented a pose estimation system for evaluating standing long jumps. The primary

results show that identification of pose classes is promising. However, some refinement on the DBN is still necessary. It would mainly be on the training data set. More training data with better definitions of poses are needed. Also, more partitions instead of just eight as shown in Figure 6 can be used for feature encoding. More information would further improve the classification results. When the poses can be accurately determined, incorrect movements, i.e. the ones different from the standing long jump standards, can be identified and advices to the jumper can be given.

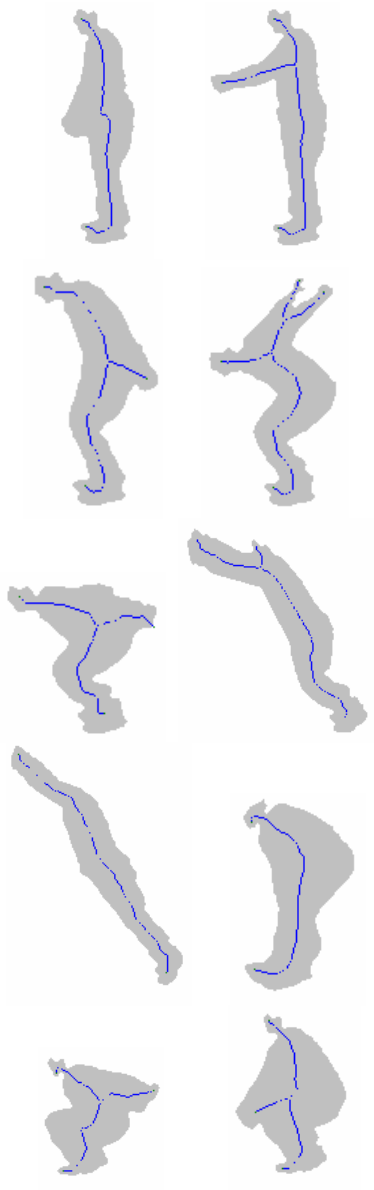


Figure 8. Skeleton extraction by thinning

Acknowledgements

This research was supported by National Science Council of the Republic of China (Taiwan) under grant number 95-2221-E-032-041.

References

[1] Hui-Huang Hsu, Sheng-Wen Hsieh, Wu-Chou Chen, Chun-Jung Chen, and Che-Yu Yang, "Motion Analysis for the Standing Long Jump," in *Proc. 26th IEEE International Conference Distributed Computing Systems Workshops*, Lisboa, Portugal, July 4-7, 2006.

[2] Ying Luo, Tzong-Der Wu, and Jenq-Neng Hwang, "Object-based Analysis and Interpretation of Human Motion in Sports Video Sequences by Dynamic Bayesian Networks," *Computer Vision and Image Understanding*, Special Issue on Video Retrieval and Summarization, Vol. 92, Issues 2-3, pp. 196-216, Nov.-Dec. 2003.

[3] Ibrahim Karliga and Jenq-Neng Hwang, "Analyzing Human Body 3-D Motion Of Golf Swing From Single-Camera Video Sequences," in *Proc. 2006 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 5, 2006.

[4] Sangho Park and J. K. Aggarwal, "Recognition of Two-person Interactions Using a Hierarchical Bayesian Network," in *Proc. First ACM SOGMM International Workshop on Video Surveillance*, Recognition section, pp. 65-76, Berkeley, CA, USA, Nov. 7, 2003.

[5] Sunil S. Polmottawegedara, Ranjith Munasinghe, and Asad Davari, "Tracking Moving Targets," in *Proc. the 38th Southeastern Symposium on System Theory*, Cookeville, TN, USA, March 5-7, 2006.

[6] S. Zhang and K. S. Fu, "A Thinning Algorithm for Discrete Binary Images," *Proc. the Int'l Conf. on Computers and Application*, pp. 879-886, Beijing, China, 1984.

[7] B. Kegl and A. Krzyzak, "Piecewise Linear Skeletonization Using Principal Curves," *IEEE Trans on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 1, pp. 59-73, Jan. 2002.

[8] C.-L. Huang, H.-C. Shih, C.-Y. Chao, "Semantic Analysis of Soccer Video Using Dynamic Bayesian Network," *IEEE Transactions on Multimedia*, Vol. 8, No. 1, 749-760, Feb. 2006.

[9] David Edwards, *Introduction to Graphical Modeling*, 2nd edition, Springer, 2007.