21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Real-Time Human Object Motion Parameters Estimation from Depth Images

I-Chung Tsao[1] and Chung-Lin Huang[1,2]
1.Department of Electrical Engineering, National Tsing Hua University, HsinChu, Taiwan, ROC
2. Department of Applied Informatics and Multimedia, Asian University, Taichung, Taiwan, ROC

Abstract.

This paper introduces a vision-based motion capture system. Motion capturing technology consists of two categories: model-based tracking and example-based indexing. The motion capturing systems face two challenges: parameter estimation in high-dimensional space and self-occlusion. Our algorithm extends the locality sensitive hashing (*LSH*) method to find the approximate examples and then estimates the pose parameters in high search space. The contributions of this method are proposing the modified LSH function, applying Hough voting to estimate the pose parameters, and adding the temporal/prediction constraints to increase the prediction accuracy.

1. Introduction

Vision-based human body tracking and pose estimation has been simplified by the introduction of real-time depth camera [1~3]. However, until the launch of Kinect, none ran at interactive rates on consumer hardware while handling human body of different shapes undergoing general articulated motions. Most of vision-based approaches face two challenges: the parameter estimation in high-dimensional space and self-occlusion.

The vision-based human motion capturing can be divided into two categories: *model-based tracking* and *example-based pose estimation*. Many model-based human tacking methods apply particle filtering (*PF*) [11,12]. Example-based method exploits a set of labeled training examples. For human pose estimation, high-dimensional search space and large data sets make this method complicate. In [4, 5], human pose estimation can be solved by using similarity measure for shape matching. In [6], they overcome the high-dimensional space problem by using Local-Sensitive Hashing (*LSH*) [10] for fast approximate neighbor search. In [7], a patch-based approach combined with *LSH* is used to retrieve example patches and estimate the pose parameters. Shotton *et al.* [8] predict 3D positions of body joints from a single depth image. By using lots of training data, they train a random decision forest classifier. Wang *et al.*[9] propose an upper body motion capturing system using one or more cameras and a color shirt. They classify the color regions to estimate the pose and use the estimated pose to refine the color classification iteratively.

We estimate the pose parameter by assembling the local example patches pre-stored in the database. In recognition stage, we use a set of local patches and *modified LSH* to extract the similar example patches, and then apply the temporal and prediction constraints to the similar example patches and then use Hough voting to estimate the pose parameter..

2. Patch Database Construction

Our approach estimates the pose parameter by assembling the retrieved example patches indexed by the input local patches. We need a database containing these example patches generated by 3D human model.

2.1 3D Human Model

Human pose can be described by 10 pose parameters including the 3D positions of torso, neck, left/right shoulders, left/right elbows, left/right hips, and left/right knees. Then, we divide the pose parameter $\Theta=(\theta_1,\ldots\theta_{10})$ into six local pose parameters, $\Theta=\{\Theta_1,\ldots\Theta_6\}$ where $\Theta_1=(\theta_1, \theta_2)$ for the two joints of the right hand, $\Theta_2=(\theta_3, \theta_4)$ for the two joints of the left hand, $\Theta_3=(\theta_5, \theta_6)$ for the two joints of right leg, $\Theta_4=(\theta_7, \theta_8)$ for the two joints of the left leg, and $\Theta_5=\theta_9$ for the neck joint and $\Theta_6=\theta_{10}$ for the position of the torso.

2.2 Local Patch and Shape Context Extraction

We use Kinect to capture the images of human motion. After extracting the human silhouette, we trace along the boundary contour of the silhouette to find the sample points and then extract the local patches as shown in Figure 1. There are 60 sample points and 60 local patches with size 100×100.

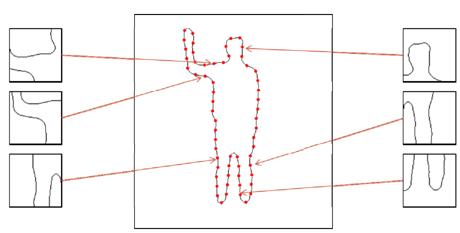

Figure 1. Path extraction along the boundary.

Based on the depth difference, we differentiate the boundary of silhouette as the boundary contour and the frontal contour. The frontal contour overlapped with the boundary contour is called the *augmented contour* as shown in Figure 2(d).

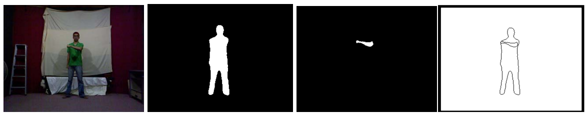

Figure 2. (a) Input image, (b) Human silhouette, (c) Frontal image, and (d) Augmented contour.

With *augmented contour*, we sample the boundary contour sparsely to extract the local patches which are described by the shape context. The shape context is described with constant radius R_{db}, vector $\boldsymbol{v}$ from the patch's position to the reference point of the model, and the contour points observed within the subarea of the patch. As shown in Figure 3, the patch is a circular shape which is divided into r radius in radial direction and θ angles in angular direction with $r\theta$ subareas. Its shape context is converted into 2-D histogram of which each beam represents the number of contour points inside the subarea. A local patch is divided into 24 subareas and described by the shape context which is a 24-D feature vector.

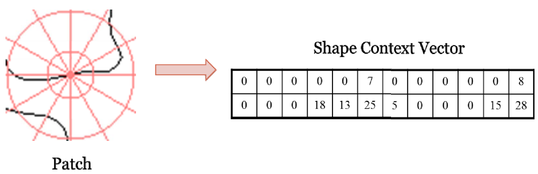

Figure 3. Shape Context of the local patch of Figure 1.

Based on the position to the centroid of the human silhouette, the local patch may be classified into six different categories. The local patches in the specific category can only vote for the corresponding local parameter Θ_i, i=1~6. The advantages of local patch categorization are (a) less collision of similar local patches, and (b) more effective Hough voting for the correct local pose parameter.

To make our algorithm invariant to the size variation, we rescale the size of each extracted local patch. Each sampled contour point is the center of the local patch. We compute the average distance of every pair of points as R_{db}. For each input silhouette, we also compute the mean distance between two sample point as R_{input}. Then, the radius of the input local patch is computed as $r_{input}=(R_{input}/R_{db})r_{db}$, where r_{db} is the radius of the local patch in the database.

3. Nearest Neighbor Search

Example-based pose estimation can be formulated as a nearest neighbor searching problem between the input patch and the example patches in the database which can be solved by the local sensitive hashing.

3.1 Local Sensitive Hashing

The local sensitive functions hash (*LSH*) function h is defined as

$$if\ d(\mathrm{u},\mathrm{v}) \le r\ then\ Pr\big(h(\mathrm{u}) = h(\mathrm{v})\big) \ge p_1 \qquad (1)$$

$$if\ d(\mathrm{u},\mathrm{v}) > (1+\epsilon)r\ then\ Pr\big(h(\mathrm{u}) = h(\mathrm{v})\big) \le p_2 (2)$$

where $\mathbf{u}=(u_1, \ldots, u_m)$ and $\mathbf{v}=(v_1, \ldots, v_m)$ are two samples, $d(\cdot)$ is a distance measure, h is a hash function that convert a sample point to a binary hash value. The set of hash functions satisfying the two conditions are called *locality-sensitive* hash functions. An effective *LSH* function must satisfy two conditions: $p_1 > p_2$, and $p_1 > 1/2$. A k-bit LSH function is $g(\boldsymbol{x})=[h_1(\boldsymbol{x}), \ldots h_k(\boldsymbol{x})]$.

The samples with the same hash are assigned to the same bucket called *collision*. The probability of collision for similar sample points is at least $1-(1-p_1)^k$, while the probability of collision for dissimilar sample is at most p_2^k. Different examples assigned to the same bucket create a collision.

3.2 Hash Function Determination

Given a sample set $P=\{\boldsymbol{p}\}$ with $\boldsymbol{p}=(x_1, \ldots x_d)$, we have $C= \mathcal{M}ax\{x_1, \ldots x_d \mid$ for all $\boldsymbol{p} \in \mathrm{P}\}$, and convert $\boldsymbol{p}$ to a C-bit binary vector as $v(\boldsymbol{p})= Unary_C(x_1), \ldots Unary_C(x_d)$. $Unary_C(x)$ indicates that a scalar x is represented by a sequence of *C-bits* bit stream of which there are x number of "1" followed by $C-x$ number of "0". Two closed enough samples are called the positive sample pair, whereas two distant samples are called the negative sample pair. After *LSH* function, if the two binary hash values of two positive sample pair are the same, then they are *True Positive*(*TP*), and if the two binary hash values of two negative sample pair are the same, then they are *False Positive*(*FP*). Here, we select the outcome of certain bit of $Unary_C(x)$. The selected bit will make the *TP* rate$\ge p_1$, and *FP* rate $< p_2$. The hash function of component x is a binary value, $h(x)$=0/1, which can be treated as a categorization process. The pseudo codes for *TP* and *FP* rates are

For *True Positive*:	For *False Positive*:
If $d(\mathbf{u}, \mathbf{v}) \le \mathrm{r}$	**If** $d(\mathbf{u}, \mathbf{v}) > (1+\epsilon)\ \mathrm{r}$
TP_{Count} +1	FP_{Count} +1
For every bit	**For** every bit
If k**u**(b) == k**v**(b)	**If** k**u**(b) == k**v**(b)
TP_{b} +1	FP_{b} +1

TP_{Count} is the total number of *TP* of the sample pirs, whereas FP_{Count} is the total number of *FP* of the sample pairs; If the b^{th} bit is selected, then the total number of *TP* generated is TP_b, and the total number of *FP* generated is FP_b. k$\boldsymbol{u}(b)$ is the outcome of the b^{th} bit of example $\boldsymbol{u}$.

3.3 Hash Table Construction

After hash function training process, we select k hash functions to generate the *k-bit* hash key to generate the hash table as the sample patch database. However, the shape context of two different local patches may be converted to the same hash key called *collision*. The example patches with the corresponding pose parameters are stored in the buckets in the hash table. The buckets with the same hash key are connected as a linked list. In each bucket, we store an example patch with the corresponding pose parameter. To reduce the invalid hash keys, we reorder the index of the hash key to reduce the empty bucket.

3.4 Modified LSH

The *Unary* operation with large C is very time consuming. *Unary* operation may add many "0" for the component corresponding to the subarea of small radius so that the length variation of the bit stream for each component will be huge. So, we propose a normalization process before Unary operation to reduce C by converting the shape context of local patch $\boldsymbol{p}=(x_1,\ldots,x_d)$ to $\boldsymbol{p}'=(y_1,\ldots,y_d)$, with $y_i=$ Int$[8\times x_i/C_i]$ and $0\leq y_i\leq 8$. In Figure 4, an 88-bit (8+60+20) stream is reduced to 24-bit (8+8+8).

Max vector

8	60	20

Vector 1	2	15	20
Vector 2	0	60	15
Vector 3	8	30	0

Normalize

Vector 1	2	2	8
Vector 2	0	8	6
Vector 3	8	4	0

Binary Vector 1	11000000	11000000	11111111
Binary Vector 2	00000000	11111111	11111100
Binary Vector 3	11111111	11110000	00000000

Figure 4. Normalization process.

Then, we propose a simplified Unary operation. Let x_d be the d^{th} component of $\boldsymbol{x}$ and converted to a bit stream of which the i^{th} bit can also be determined by comparing x_d with i. If $x_d\geq i$, then the i^{th} bit will be "1" otherwise it will be "0". After *Unary* operation, we find the hash function $h_d(\boldsymbol{x})$ for component d. The hash function generates a binary output by selecting the i^{th} bit of the bit stream generated by *Unary* operation. The i is determined by selecting the i^{th} bits of $Unary_C(x_d)$ which generate the best trade-off between higher *TP* rate and lower *FP* rate. A k-bit *LSH* function can be rewritten as $g(\boldsymbol{x})=[h_1(\boldsymbol{x}),\ldots h_k(\boldsymbol{x})]$ of which $h_d(\boldsymbol{x})=1$ if $x_d\geq i_d$, else $h_d(\boldsymbol{x})=0$ for $d=1,\ldots k$.

4. Parameter Estimation

For each input local patch, we find its category and use *LSH* indexing to extract the similar patches, and then apply the Hough voting to find the most probable pose parameter. However, the local pose parameter receiving the maximum votes may not be the right solution. So, we apply the temporal and prediction constraints before and after the Hough voting process.

4.1 Temporal Constrains

Human articulated motion is smooth and continuous so that the pose difference between two instances satisfies a so-called the *temporal constrain*. In training, we model the motion parameters of the limbs (*i.e.*, left shoulder and right elbow). For each joint, we collect the difference of pose parameter at two continuous time instance. The distribution of the difference can be described as a Gaussian distribution $N(\mu, \sigma)$.

4.2 Hough Voting

For each input local patch, with specific category q (q=1~6), we may compute the hash key to retrieve the corresponding similar example patches I_k^q in database Q^q. To simplify the estimation process, we assume that the probabilities of local pose parameters in different categories are statistically independent. So we have

$$p(\Theta|E)=\sum_{q=1}^{6}p(\Theta_i|E^q) \quad (4)$$

where E^q is a set of extracted local patches $E^q=\{e_i^q\}$ in the q^{th} category. Based on the local patches, we use *LSH* to retrieve the example patches $\{I_k^q, k=1,\ldots K\}$ in Q^q. Each example patch contributes votes to estimate the local pose parameter $\boldsymbol{\Theta_i}$. The local pose parameter estimation is described as

$$p(\Theta_i|E^q)=\sum_{i=1}^{|E^q|}\sum_{k=1}^{K}p(\Theta_i|I_k^q)p(I_k^q|e_i^q)p(e_i^q) \quad (5)$$

where $p(\Theta_i|I_k^q)$ is the likelihood of estimating the local pose parameter Θ_i based on the retrieved example patches $\{I_k^q, k=1,..K\}$, $p(I_k^q|e_i^q)$ represents the likelihood of finding the example patches, and $p(e_i^q)$ denotes the likelihood of observing the input patch.

For an input local patch, we may find multiple example patches with the similar shape context. To apply the temporal constraint, we assign a weight w_k to each retrieved example patch I_k^q. It indicates a different contribution to estimate the local pose parameter Θ_i by casting different weighted vote. The weight $w_k=p(I_k^q|e_i^q)=1/N_{I_k\in Q}$, if $|\Theta_i(t)-\Theta_i(t+1)|<\sigma$, otherwise w_k=0. $N_{I_k\in Q}$ is the number of retrieved sample patches in Q^q with the same Θ_i. Each input patch is assigned an equal priori.

4.3 Prediction Constraint

Due to self-occlusion, we cannot find the correct pose parameter based on the voting only. We propose a prediction constrains which relates the candidate pose parameter Θ_j with the predicted pose parameter $\Theta_{predict}$ and determines the weight for the candidate pose parameter Θ_j as w_{Θ_j}. We modify (5) as follows

$$p(\Theta_j|E^q)=\sum_{i=1}^{|E^q|}\sum_{k=1}^{K}w_{\Theta_j}p(\Theta_j|I_k^q)p(I_k^q|e_i^q)p(e_i^q)$$

The weight w_{Θ_j} is defined as

$$w_{\Theta_j}=\left(\frac{|\Theta_j-\Theta_{pred}|}{\sum_j|\Theta_j-\Theta_{pred}|}\right)^{-1}$$

Θ_j is the candidate pose parameter receiving enough votes and satisfying the temporal constraints and Θ_{pred} is the predicted pose parameter defined as

$$\Theta_{pred}=\frac{1}{N}\sum_{j=1}^{N}\Theta_j$$

The voting distributions for the example patches in different categories are different which is used for final pose parameter estimation.

5. Experimental Results

We use the commercial motion capture system to capture the positions and orientations of 10 joints as the ground truth. To create the real silhouette images of real human figure, we use the depth images from Kinect. We generate 12000 dataset images captured from two different human figures performing 4000 various poses. The training set contains 8000 images, whereas the testing set contains 4000 images.

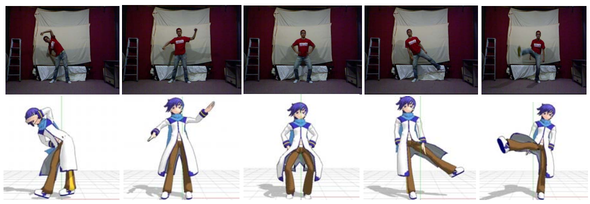

Figure 5. Input images and 3D avatar.

Figures 5 illustrates our experimental results. To measure the accuracy of the estimation, we compute the average error between the estimated joint positions of the ground truth as

$$Avg\ error=\Sigma^{10}_{j=1} ||\mathbf{g}_j - \mathbf{e}_j||/10 \qquad (9)$$

where j is the index of the joint, $\mathbf{g}_j$ indicates the joint 3D position parameter of the ground truth, $\mathbf{e}_j$ indicates the estimated joint 3D position parameter, and $||\cdot||$ is the Euclidian distance measure. As the number of input patches increases, the average error decreases but the computation time increases. However, the improvement is saturated after certain number of patches is selected.

The computation complexity is linearly increased with the number of patches. Here, we let the number of patches N_{input}=60, and fix the number of postures N_{pose} in the database. We compare the *LSH* searching and conventional full search method as shown in Table 1. Full search will find the pose with the least error, but the computation complexity is not acceptable.

Table 1 Error and complexity comparison.

	Linear Search	LSH
Avg Error(mm)	48.57	54.32
Computing Time(ms)	12780	37.01

From the estimation error of the local pose parameters, we find that upper-joints (such as shoulder and hip) is smaller than the lower-joints (such as elbow, wrist). It is because of hierarchical structure of human parts and the movement of the upper-joint is smaller than the lower-joint. However, the error is not reduced significantly with the increment of the number of selected patches. Using temporal constraint does not reduce much error, but reduce the computation dramatically. We compare the estimation errors and computing time of these two methods as the number of poses increases.

We compare the performance of convention *LSH* [7] and our modified *LSH* method in training time, estimation time and average error as shown in Table 2. We extract 60 patches from the test image for *LSH* method. Table 2 shows that the improvement of precision is limited, however, the improvement of training time and estimation time is enormous. Because current joint estimation is based on the previous location of the joint, the estimated pose is not correct if there is a sudden moving direction change,

Table 2. Compare the error and complexity of LSH[7] and our modified LSH.

	Training time	Computing Time/frame	Avg Error (mm)
LSH[8]	485 min	58.71 ms	57.54
Our LSH	273 min	36.02 ms	57.54

6. Conclusion

This paper presents a human motion parameter estimation method. First, we generate 2D posture image and the corresponding 3D position of the joints stored in the database. Then, we extract the local patch which is described by shape context and then use the modified LSH to find the example patches in the database. Finally, we use Hough voting to find the best matched pose and estimate the 3D joint locations.

Reference

[1] V. Ganapathi, *et al*. Real Time motion capturing using single time-of-flight camera. Proc. CVPR, 2011.
[2] M. Siddiqui *et al*. Human Pose Estimation from a Single View Point, Real-Time Range Sensor. Proc. Of CVCG at CVPR 2010.
[3] Y. Zhu *et al*. Constrained Optimization For Human Pose Estimation from Depth Sequences. ACCV, 2007.
[4] G. Morio *et al.* Estimating Human Body Configuration using Shape Context Matching. ECCV 2002.
[5] V. Athitsos *et al.* Estimating 3D Hand Pose from Cluttered Image. IEEE CVPR, 2003.
[6] G. Shakhnarovich, *et al.* Fast Pose Estimation with Parameter Sensitive Hashing. Proc. of ICCV 2003.
[7] K. Hara. Real-time Inference of 3D human Poses by Assembling Local Patches. IEEE Workshop of Applications of Computer Vision 2009.
[8] J. Shotton, *et al.* Real-time Human Pose Recognition in Parts from Single Depth Images. CVPR 2011.
[9] R. Wang, *et al.* Practical Color-based Motion Capture. 21th ACM SIGGRAPH/ Eurographics Sym. on Computer Animation, 2011.
[10] A. Gionis, *et al.* Similarity Search in High Dimensions via Hashing. 25th Int. Conf. on Very Large Data Base, 1999.
[11] J. Schmidt, *et al.* Kernel Particle Filter for Real-Time 3D Body Tracking in Monocular Color Images. 7th Int. Conf. on Automatic Face and Gesture Recognition, 2006.
[12] J. Deutscher, *et al.* Articulated Body Motion Capture by Annealed Particle Filtering. *CVPR 2000.*