Fast Online Human Pose Estimation via 3D Voxel Data

Yuichi Sagawa, Masamichi Shimosaka, Taketoshi Mori and Tomomasa Sato
Graduate School of Information Science and Engineering, The University of Tokyo
{sagawa, simosaka, tmori}@ics.t.u-tokyo.ac.jp, tomomasasato@jcom.home.ne.jp

***Abstract*— In this paper, a novel approach is proposed to recover human body pose from 3D voxel data. The use of voxel data leads to viewpoint-free estimation, which benefits in that reconstruction of a training model is needless in different multi-camera arrangements. Other notable aspects of our approach are real-time ensuring speed (up to 30[FPS]), flexibility towards various complex motions, and robustness. These are provided by an example based approach, which constructs human posture candidates beforehand from a large motion capture database. The most appropriate posture is estimated per frame by comparing the likelihoods between posture candidates and 3D voxel data. The evaluation is formulated by introducing a histogram-based feature vector that represents the 3D shape context of human body. In addition, a fast near-neighbor search metric is installed prior to the evaluation process, which reduces computational cost and ensures real-time processing. Estimation stability is also improved by a graphical model of motion, which adds a smoothing effect to the motion sequence. We demonstrate the effectiveness of our approach with experiments on both synthetic and real image sequences.**

I. Introduction

Recently, expectation towards robotic systems which enables daily life assistance are rising. Among many, researches on home environment with distributed sensors and affluent databases are active, and believed to offer practical applications. Aware Home [1] and Sensing Room [2] are examples of such systems. These projects intend to intelligently understand and analyze human action for future use.

For efficient human action analysis, a rich representation of human condition is necessary. One idea is to represent it in motion capture data format, but we believe that devices prone to wear are unnecessary for practical home use. So instead of using wearable devices, digital cameras are selected as sensing devices. Therefore, our goal is set to estimate human posture and construct an image-based simple motion capture interface.

In this paper, at first, our approach is briefly introduced and compared with conventional approaches. Then, significant phases are extracted from the whole process, and provided with detailed information. In the end, experimental results are presented with evaluation.

II. Image-Based Human Pose Estimation

Image-based human posture analysis has been a hot trend in the computer vision domain, but difficulties still remain such as 1) **vagueness due to monocular image processing** 2) **viewpoint dependency** 3) **offline processing due to huge computational cost**. Past researches were usually based on a generative approach which typically modeled the human body with a joint distribution, using the observation likelihood or the cost function [3], [4]. These approaches had flexibility towards complex and unknown motions, but inference involved complex search over state space, which lead to iterative 3D human model sampling or use of non-linear optimization methods. These arguments motivated the study of discriminative approaches [5], [6], which model and predict state condition directly from observation. These approaches require large training data for modeling, and difficulty has been suggested in assuring sufficiency in a mapping between observation and state condition. Although these problems exist, easier model construction and speed-up of the inference process are notable characteristics. For an example, approaches based on fast nearest-neighbor retrieval [7], [8], [9] and multiple image integration [9], [10], [11] has been worked on, and suggested difficulties are partially being improved.

We propose a discriminative approach mainly based on the ideas listed below, which provides a complete solution to the suggested difficulties. The main contribution of this approach is speed, which ensures real-time processing up to 30[FPS]. Other notable aspects are flexibility towards various complex motions, robustness, and reusability of training models. The whole process flow is presented in Fig. 1. Offline and online process respectively corresponds to model construction and human pose inference.

- **Multiple Image Integration**
  Vagueness is associated with silhouette-based monocular image processing. Integration of information extracted from multiple silhouette images lead to reduction of vagueness.
- **3D Voxel Reconstruction**
  3D voxel reconstruction is possible through multiple silhouette image integration. This leads to viewpoint-free estimation, which benefits in that reconstruction of a training model is needless in different multi-camera arrangements.
- **Computational Cost Reduction**
  Human posture candidates are defined beforehand from a large motion capture database. This policy is so-called an example based approach [7], [8], [9], which reduces the computational cost compared to random sampling or non-linear optimization methods. Instead of searching entirely through the state space, inference is simplified to a comparison of likelihoods between human posture candidates and 3D voxel data. Furthermore, a fast near-

neighbor metric [7] is installed prior to the evaluation process, which additionally reduces the computational cost.

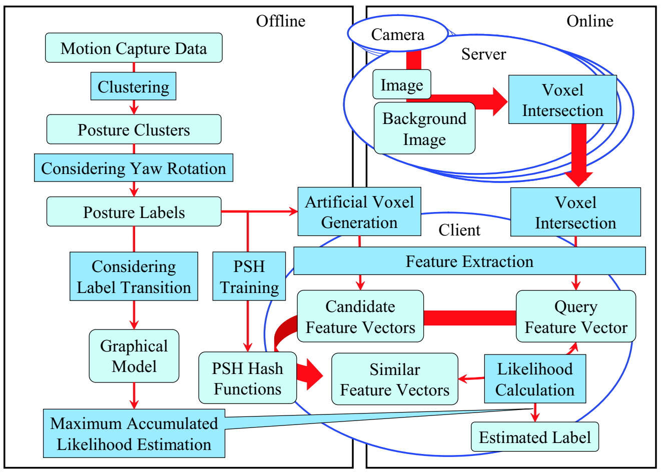

Fig. 1. Flow of Human Pose Estimation

III. Construction of Posture Candidates

Generally, human postures are treated as a continuous quantity, but we treat it as a discrete one, which enables defining posture candidates preliminarily. This possibly causes a sparse inference result, which is an arguable point, but increase in posture candidate numbers would lead to denseness of human posture state. Thus, example-based inference can be considered as a sufficient approximation of continuous inference methods [8]. Motion capture data downloaded from www.mocapdata.com (Walk, Jump, Sit Down, Throw, and other complex motions are included), and a human model with 19 joints and 5 end-points (60 DOF configuration, consisted of root coordinates and Euler angles of joints) are used in later experiments.

In a 3D human pose estimation task, the yaw rotation of a human body must be considered appropriately (see Fig. 5). For example, clustering using motion capture data containing postures with different yaw rotation will induce unstable results. Therefore, we construct posture candidates based on a two-stage clustering process, using posture clusters, which are defined as clusters consisted of posture candidates corresponding to a basis yaw rotation.

In the initial stage, about a few thousand posture clusters are extracted from a motion capture database. For efficient motion data use, each frame is rotated and adjusted to the basis yaw rotation before being passed to the K-means method [12]. Likelihood in the K-means method is formulated as the Euclidean distance $d_{\boldsymbol{X}}$ between 72-dimensional position vectors $\boldsymbol{X}$, $\tilde{\boldsymbol{X}} \in \mathbb{R}^{72}$ (coordinates of joints and end-points are concatenated, $\{\boldsymbol{X}\}_i$: i th element of vector $\boldsymbol{X}$).

$$d_{\boldsymbol{X}}(\boldsymbol{X}, \tilde{\boldsymbol{X}}) = \sqrt{\sum_{i=0}^{72} \{\boldsymbol{X}\}_i \{\tilde{\boldsymbol{X}}\}_i} \qquad (1)$$

In the second stage, posture candidates are constructed by rotating posture clusters at intervals of a step degree Θ. Posture candidates are managed with corresponding index values, which are defined as posture labels. The number of posture labels N_l is a multiple of posture cluster counts N_c and rotation resolution r_θ, which will result in about a few ten thousands.

$$\Theta = 360[deg]/r_\theta \qquad (2)$$

$$N_l = N_c r_\theta \qquad (3)$$

Additionally, a directed graphical model of motion, which represents first order Markov property is constructed (see Fig. 2). Motion sequences are scanned, and each frame is referenced with a posture label $y(t)$. During the scanning process, transition probability $T_{i,j}$ between the previous label i and current label j is accumulated to eventually form a whole graphical model. In order to prevent getting caught into a local solution, $T_{i,j}$ is binarized as $0,\ 1$.

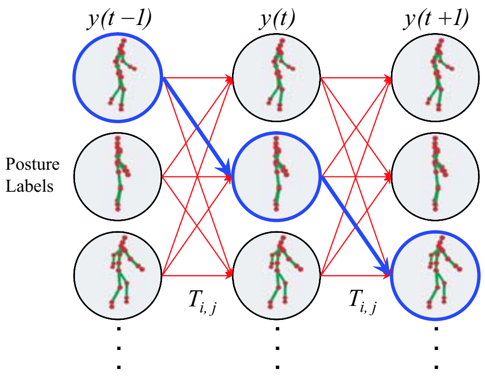

Fig. 2. Constructing a Graphical Model

IV. Volume Intersection Based on Fast Parallel Calculation

The volume intersection method is a method for reconstructing 3D object shape from multiple silhouette images (see Fig. 3). The output is an assembly of a voxel, which represents a bin of the divided 3D space. At first, silhouette images are obtained from a background subtraction process. Specifically, background subtraction is applied individually to 3 channels of HSV color space, and then combined as a single image through logical addition. Thresholds are determined from the Otsu's Algorithm [13], but the values are decreased for the purpose of suppressing the voxel loss to the minimum. Nonetheless, final voxel outputs will not receive large influence, because the voxel intersection method is robust to image noise. Then, silhouette pixels are back-projected per image to form multiple visual hull regions (intrinsic / extrinsic camera parameters are calculated beforehand). In the end, 3D voxels are integrated by extracting voxels that intersect in all visual hull regions. Implementation is done based on the policies listed below for speed-up.

- **Speed-Up of Back-Projection Calculation**
  Under the assumption that extrinsic camera parameters are invariant, projection from 3D voxels to 2D pixels are calculated beforehand. The reference relation from pixel to voxel are stored in look-up table format [11]. By doing so, voxels are acquired by just scanning through silhouette pixels and referring the look-up table.
- **Cut Down of Network Traffic**
  A voxel condition is possibly represented in binary format (exist / non-exist). Therefore, data of 1[byte] can represent 8 voxel conditions. By applying this rule, network traffic can be suppressed to the minimum. Moreover, voxels are grouped into proximity regions, limiting data transfer to regions only where voxels exist, and eventually reducing more network traffic.

A volume intersected result using 8 cameras are shown in Fig. 4. Two images in the left show reconstructed voxels, while 16 images in the right show original and silhouette images of 8 cameras. For speeding up the rendering process, only surface voxels are rendered (approximately 1500 voxels are rendered).

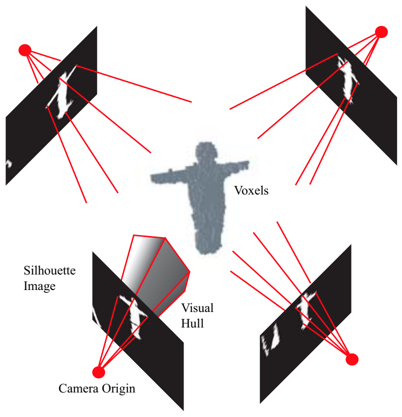

Fig. 3. Voxel Intersection Method

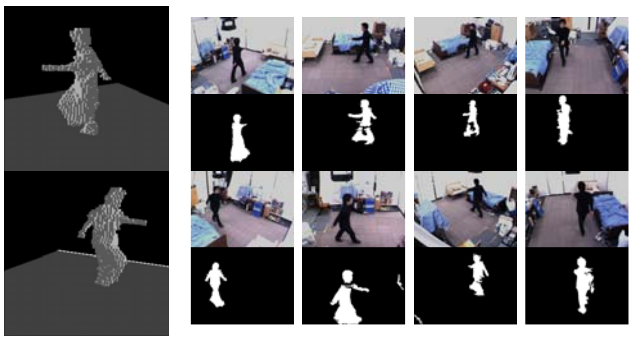

Fig. 4. Reconstructed 3D Voxels

V. Likelihood Evaluation Based on Feature Extraction

A. Histogram-Based Feature Extraction

In our approach, a feature with a position invariant property is desired. Kortgen et al. proposed a position and rotation invariant feature for 3D objects named 3D Shape Contexts, and applied the metric to 3D object matching [14]. This feature is easily extracted from voxel data, but in our approach the rotation invariant property is redundant and leads to higher computational cost. Huang et al. used a simplified 3D Shape Context feature for a gesture recognition application [15]. We extended their approach to extract a histogram-based feature from voxel data. Extracting procedures are listed below and an outline illustration is presented in Fig. 5.

1) 3D Voxel data is reconstructed from the voxel intersection method, and a central axis is set to the center of gravity.
2) Based on the central axis, 3D space is divided into r_z height divisions, r_r radius divisions, and r_θ angle divisions. The total resolution will be $r_z r_r r_\theta$, and a corresponding 1D array of the same size is prepared. When the index values of height, radius, and angle divisions are respectively defined as i_z, i_r, i_θ, index I of the array is calculated from (4).

$$I = i_\theta + i_r r_\theta + i_z r_\theta r_r \tag{4}$$

3) When a voxel v is included in the region of bin b_I, the distance from the central axis d_v is added to the corresponding array element. After this operation is applied to every voxel in the effective range, each array element is normalized by dividing it with a d_v total. The result array is defined as a query feature vector $\boldsymbol{q}(x_t)$ (see (5)). This weighting operation, has an effect to weaken the influence of voxels near the central axis, eventually increasing robustness.

$$\{\boldsymbol{q}(x_t)\}_I = \frac{\sum_{v \in b_I} d_v}{\sum_v d_v} \tag{5}$$

The effective range of this feature extraction process is provided by the subject's height h, which means a scaling problem will not occur. Furthermore, as a result of normalization, the final inference result will not depend on the resolution of voxel space or the subject's position. These properties offer high reusability of training models.

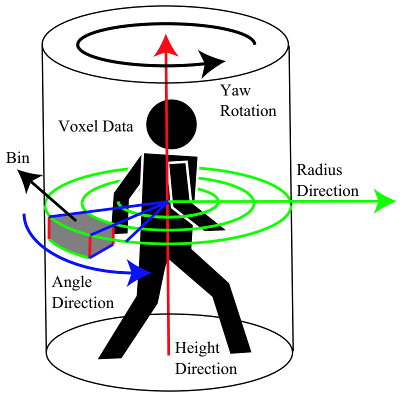

Fig. 5. Extracting a Histogram-based Feature from Voxel Data

B. *Artificial Voxel Data Generation*

Likelihoods between posture candidates and 3D voxel data are calculated with histogram-based feature vectors. Because posture candidates are represented in human body joint angle format, some kind of transformation is necessary for direct evaluation. Histogram-based feature vectors of posture candidates are extracted from joint angle format data through artificial voxel data generation. Artificial voxel data is generated by approximating body links as a cylinder or an ellipsoid (approximated radii are configured manually). At first, joint angle format data is reflected to the human model. Then, voxels included in the approximated region of each body link is extracted to configure a voxel data (see Fig. 6, artificial voxel data in the right are generated from the human model in the left). In this way, candidate feature vectors $\boldsymbol{\mathcal{Q}}$ are generated from posture candidates.

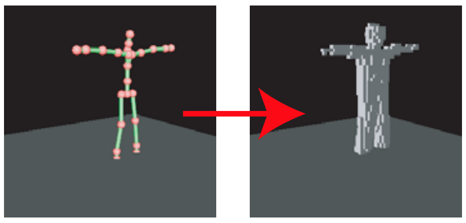

Fig. 6. Artificial Voxel Data Generation

C. *Likelihood Evaluation*

Likelihood evaluation is based on the Bhattacharyya Coefficient [16]. Likelihood between a candidate feature vector $\boldsymbol{q}_j \in \boldsymbol{\mathcal{Q}}$ of label j and a query feature vector $\boldsymbol{q}(x_t)$ is calculated from the equation below.

$$\phi_t(j) = BC(\boldsymbol{q}_j,\ \boldsymbol{q}(x_t)) = \sum_{r=0}^{r_z r_r r_\theta} \sqrt{\{\boldsymbol{q}_j\}_r \{\boldsymbol{q}(x_t)\}_r} \qquad (6)$$

VI. Near-Neighbor Search for Computational Cost Reduction

A. *Parameter Sensitive Hashing (PSH)*

When a naive nearest-neighbor search algorithm is combined with the likelihood evaluation process, the computational cost will be proportional to the number of posture labels. In order to reduce this computational cost, a fast near-neighbor search metric named **Parameter Sensitive Hashing (PSH)** [7] is introduced. PSH is an extended version of a similar metric named **Locality Sensitive Hashing (LSH)** [17], and it's introduction drastically decreases the likelihood calculation frequency to a few percent, which enables real-time processing.

In PSH, a query feature vector $\boldsymbol{q}$ is given to trained hash functions $\boldsymbol{\mathcal{G}}$ for input, and similar labels $\boldsymbol{\mathcal{S}}$ are obtained as output. Specifically, L hash functions $\boldsymbol{\mathcal{G}}$ are constructed through training, and each hash function $g_l(\boldsymbol{q}) \in \boldsymbol{\mathcal{G}}$ outputs an index value for both $\boldsymbol{q}$ and a candidate feature vector $\boldsymbol{q}_j$ of label j. When the index values are the same value, j will be an element of $\boldsymbol{\mathcal{S}}$ (allows duplication). The actual implementation uses a look-up table which maps posture labels based on index values, where the index value of $\boldsymbol{q}$ is used as a query. Therefore, there is no need to calculate index values for $\boldsymbol{\mathcal{Q}}$ during the search process. Hash function $g(\boldsymbol{q})$ consists of concatenated K decision stumps $h_k(\boldsymbol{q})$ (see (7)), and $h_k(\boldsymbol{q})$ is a binary function which is defined by function $\psi_o(\boldsymbol{q})$ and threshold T_d (see (8)). We define $\psi_o(\boldsymbol{q})$ as a function that returns the o th element of $\boldsymbol{q}$, which suggests that the purpose of PSH training is to extract sensitive functions and optimal T_d values.

$$g(\boldsymbol{q}) = [h_1(\boldsymbol{q}), h_2(\boldsymbol{q}), \cdots, h_k(\boldsymbol{q})]^\mathsf{T} \qquad (7)$$

$$h_k(\boldsymbol{q}) = \begin{cases} 1 & \text{if } \psi_o(\boldsymbol{q}) \geq T_d, \\ 0 & \text{otherwise.} \end{cases} \qquad (8)$$

B. *PSH Training*

The sensitiveness of $\psi(\boldsymbol{q})$ is defined as the likelihood of outputting same values for similar data pairs compared to dissimilar data pairs. Therefore, sample pairs labeled with similar or dissimilar labels are used for PSH training. Label z for a sample pair $(\boldsymbol{q},\ \tilde{\boldsymbol{q}})$ will be resolved, based on the Euclidean distance $d_{\boldsymbol{X}}$ in parameter space (position vector space $\in \mathbb{R}^{72}$) and not in input space (feature vector space $\in \mathbb{R}^{r_z r_r r_\theta}$). Threshold R is an application-dependent value, and parameter ϵ is set to 0.5.

$$z = \begin{cases} +1 \text{ if } d_{\boldsymbol{X}}(\boldsymbol{X}, \tilde{\boldsymbol{X}}) < R/(1+\epsilon), \\ -1 \text{ if } d_{\boldsymbol{X}}(\boldsymbol{X}, \tilde{\boldsymbol{X}}) > R, \\ \text{not defined otherwise.} \end{cases} \qquad (9)$$

Sample pairs are constructed through scanning of motion capture data (a position vector $\boldsymbol{X}$ is extracted per frame). Similar sample pairs are configured with pairs of $\boldsymbol{X}$ and a most similar posture label that fulfills $z_{m,n} = +1$, while dissimilar sample pairs are configured with pairs of $\boldsymbol{X}$ and a randomly selected posture label that fulfills $z_{m,n} = -1$. By adding noise to sample pairs in this phase, an improvement in inference robustness can be expected. In (9), a gray zone exists where labels are not defined, but pairs that fall under this zone would not be used in the training process. Hash functions constructed using these sample pairs will assure that (10) would be fulfilled between query and output data in high probability (This property is called locality-sensitive [7], [17]). LSH assures a locality-sensitive property in input space, while PSH assures it in parameter space, which is why PSH is named this way.

$$d_{\boldsymbol{X}}(\boldsymbol{X}, \tilde{\boldsymbol{X}}) < R \qquad (10)$$

In the initial stage of training, sensitive decision stumps $\boldsymbol{\mathcal{H}}$ are obtained. Sensitiveness is evaluated from score s_o (the lower, the higher sensitivity is), which is defined as a ratio of false positive counts N_{fp} and false negative counts N_{fn} to total sample counts N (see (11)). N_{fp}, N_{fn} and T_d that minimizes s_o will be derivable through two loop calculations [7]. After constructing a decision stump $h_k(\boldsymbol{q})$ for each feature vector element, decision stumps that fulfill (11) (T_s : threshold) will be registered to $\boldsymbol{\mathcal{H}}$.

$$s_o = \frac{(N_{fp} + N_{fn})}{N} > T_s \qquad (11)$$

In the second stage of training, KL decision stumps are randomly selected from $\boldsymbol{\mathcal{H}}$ (allows duplication), to form L hash functions of K bit size. Generally, a large K increases precision and decreases query ratio, while a large L increases query ratio and decreases precision. Thus, K and L are significant parameters that drastically effect the overall performance.

C. Extension of PSH

The fundamental idea of PSH is to combine weak discriminators to configure a precise discriminator. Simply applying PSH to a human pose estimation task will result in failure, because no matter how you combine weak discriminators, it is impossible to sufficiently cover the whole posture parameter space. Therefore, we propose an approach that divides parameter space, and constructs discriminators individually in each partial parameter space. We specifically propose a divisional strategy based on human yaw rotation, which divides parameter space into r_θ partial parameter spaces. Sample pairs are individually prepared for each yaw rotation, which will be used to construct corresponding hash functions. This approach not only has an advantage in precision, but also in memory consumption in that look-up table size of hash functions will decrease due to registering posture label number reduction.

Moreover, a same value is set to human yaw rotation resolution and histogram radius resolution. This enables simple mapping between a yaw rotation index value and a decision stump index value o (there is correspondence between a yaw rotation index value and i_θ). Posture candidates are constructed by simply rotating posture clusters, which means that hash functions for the basis yaw rotation can be appropriately operated to form hash functions in other yaw rotation steps. As a result, only sample pairs for the basis yaw rotation will be needed, and training time will drastically be reduced.

VII. Inference Based on a Graphical Model of Motion

Likelihood $\phi_t(j)$ is calculated between a query feature vector $\boldsymbol{q}(x_t)$ and a candidate feature vector $\boldsymbol{q}_j$, which corresponds to similar labels $\boldsymbol{\mathcal{S}}$ ($j \in \boldsymbol{\mathcal{S}}$). Inference is possibly done per frame, by selecting label $y(t)$ that outputs the maximum value among likelihood $\phi_t(j)$.

$$y(t) = \underset{j}{\operatorname{argmax}}\, \phi_t(j) \tag{12}$$

However, not making consideration of first order Markov property leads to improper transition and inference oscillation. Thus, a graphical model of motion is embedded to provide a smoothing effect. Smoothing algorithms such as Viterbi decoding[12] are unsuitable because the optimal sequence cannot be computed until the entire input has been observed. Thus, based on maximization of posterior probability, label $y(t)$ that outputs the maximum value among accumulated likelihood $p_t(j)$ is selected per frame during the inference process.

$$p_t(j) = \begin{cases} \phi_t(j) & (t=0) \\ \max_i(p_{t-1}(i)T_{i,j}) + \phi_t(j) & (t \neq 0) \end{cases} \tag{13}$$

$$y(t) = \underset{j}{\operatorname{argmax}}\, p_t(j) \tag{14}$$

VIII. Experiment on Image-Based Human Pose Estimation

We performed experiments on image-based human pose estimation. The target motion was designed to walk around in the first half, and perform stretching exercises in the second half. Experiments were performed on 3 occasions, 1) **No PSH** : a nearest-neighbor approach was applied instead of PSH 2) **PSH + Graphical Model** : proposed approach 3) **No Graphical Model** : a graphical was not embedded in the inference process. The main idea of these experiments were to confirm the effectiveness of PSH and a graphical model of motion. The advantage of PSH, which is mainly speed, was expected to be confirmed between experiments 1) and 2). Although the final result of 1) would have possibly been more accurate than 2), the accuracy reduction rate was expected to fall in the acceptable range. The effect of embedding a graphical model is motion sequence stability (exclusion of unnatural transitions), which was expected to be confirmed between experiments 2) and 3). Specific experiment conditions are listed below and results are shown in Fig. 7.

- **Multi-Vision Environment / Volume Intersection**
  8 cameras (Point Grey Research IEEE 1394 Digital Camera Dragonfly2) and 4 server PCs (Dell Precision 690, Intel Xeon Processor 5060 3.20 GHz $\times$ 2, 4GB Memory) were prepared, and 2 cameras were connected to each server PC. Cameras were completely synchronized during capture, and intrinsic / extrinsic camera parameters were calculated beforehand. Furthermore, a client PC was prepared in the same specification, which was designed to handle the integration process in volume intersection and the main posture inference process. In addition, 3D space was divided into 120$\times$120$\times$60 resolution as a 35mm size cubic voxel.
- **Posture Labels**
  161520 frames of motion capture data (captured in 30 FPS) were used in clustering to extract 1106 posture clusters (clusters were estimated as Gaussian distributions of equal covariance). Yaw rotation resolution r_θ was set to 12 (Θ : 30[deg]), which resulted in 13272 posture labels (see (3)).
- **Histogram-Based Feature Vector**
  The effective range of feature extraction was defined by height and radius values. The height limit from the ground level was set to $1.3h$, while the radius limit from the central axis was set to $0.5h$ (h : subject's height). The dimension of the histogram-based feature vector was set to 384 (8$\times$4$\times$12).
- **PSH**
  Motion capture data was scanned and 239534 sample pairs for the basis yaw rotation were created (there were

119767 pairs for both similar and dissimilar pairs). T_s was set to 0.675 which resulted in 54 sensitive decision stumps $\boldsymbol{\mathcal{H}}$ out of 384. K and L were respectively set to 12 and 30, so that the query ratio would approximately satisfy $1 \sim 3\%$. Consequently, the total number of hash functions became 360.

The results show flexibility towards complex motions and robustness towards voxel noise. Comparison between results of 1) and 2) indicate the high performance of PSH in that precision reduction is limited to the minimum. The effect of a graphical model can also be confirmed by comparing results of frame 433 and 434. Moreover, inference in frame 476 has failed due to occlusion, but a comeback can be seen afterwards in frame 499, which means that the use of a graphical model does not lead to getting caught into a local solution. Inference failure in frame 1514 has been caused by posture candidates, which did not contain appropriate candidates. Therefore, posture parameter space has not been completely represented by posture candidates, which is a future task.

Average process times per frame and precision evaluation results on synthetic data sequences have been arranged in Table. I. The total process time of the proposed approach is shorter than the image capture interval time in 30[FPS] (indicates high speed-up performance of PSH), which assures real-time processing. Synthetic evaluation has been done using 6229 frames of motion capture data that were not used in training. Input data was created from Euler angle format data through an artificial voxel generation process. Precision was based on the Euclidean distance $d_{\boldsymbol{X}}$ between an estimated posture and the true posture (considered a success when (10) was satisfied). Judged from the results, effectiveness of PSH and a graphical model of motion can be confirmed in synthetic data sequences as well as in real image data sequences.

TABLE I
PROCESS TIME AND PRECISION EVALUATION

PSH	No	No	Yes	**Yes**
Graphical Model	No	Yes	No	**Yes**
Background Subtraction	20[ms]	20[ms]	20[ms]	**20[ms]**
Volume Intersection	1[ms]	1[ms]	1[ms]	**1[ms]**
Network Transfer	2[ms]	2[ms]	2[ms]	**2[ms]**
Server Side Total	23[ms]	23[ms]	23[ms]	**23[ms]**
Volume Intersection	1[ms]	1[ms]	1[ms]	**1[ms]**
Voxel Rendering	1[ms]	1[ms]	1[ms]	**1[ms]**
Main Inference Process	80[ms]	110 [ms]	1[ms]	**2[ms]**
Client Side Total	82[ms]	112 [ms]	3[ms]	**4[ms]**
Total Process Time	105[ms]	135[ms]	26[ms]	**27[ms]**
Image Capture(30[FPS])	33[ms]	33[ms]	33[ms]	**33[ms]**
Precision	0.856	0.869	0.832	**0.849**

IX. CONCLUSION

A novel approach to recover human pose from 3D voxel data has been proposed. Experimental results show that real-time processing up to 30[FPS] has been achieved by introducing an example-based approach and a fast near-neighbor search metric. In this paper, other notable aspects of the proposed approach such as motion stability, flexibility towards various complex motions, and robustness towards noise have also been presented. Future works are improvements in precision, representation of human posture space, and robustness towards occlusion.

REFERENCES

[1] Cory D. Kidd, Robert Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E. Starner, and Wendy Newstetter. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In *Proc. CoBuild*, pp. 191–198, 1999.

[2] Taketoshi Mori, Hiroshi Noguchi, Aritoki Takada, and Tomomasa Sato. Sensing Room: Distributed Sensor Environment for Measurement of Human Daily Behavior. In *Proc. INSS*, pp. 40–43, 2004.

[3] Masanobu Yamamoto, Akitsugu Sato, Satoshi Kawada, Takuya Kondo, and Yoshihiko Osaki. Incremental Tracking of Human Actions from Multiple Views. In *Proc. CVPR*, pp. 2–7, 1998.

[4] Kameda Yoshinari, Minou Michihiko, and Ikeda Katsuo. Dimensional Pose Estimation of an Articulated Object from its Silhouette Image. In *Proc. ACCV*, pp. 612–615, 1993.

[5] Greg Mori and Jitendra Malik. Estimating Human Body Configurations Using Shape Context Matching. In *Proc. ECCV*, pp. 666–680, 2002.

[6] Cristian Sminchisescu, Atul Kanaujia, Zhiguo Li, and Dimitris Metaxas. Discriminative Density Propagation for 3D Human Motion Estimation. In *Proc. CVPR*, Vol. 1, pp. 390–397, 2005.

[7] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast Pose Estimation with Parameter Sensitive Hashing. In *Proc. ICCV*, Vol. 2, pp. 750–757, 2003.

[8] Leonid Taycher, Gregory Shakhnarovich, David Demirdjian, and Trevor Darrell. Conditional Random People: Tracking Humans with CRFs and Grid Filters. In *Proc. CVPR*, Vol. 1, pp. 222–229, 2006.

[9] Liu Ren, Gregory Shakhnarovich, Jessica K. Hodgins, Hanspeter Pfister, and Paul Viola. Learning Silhouette Features for Control of Human Motion. *ACM Transactions on Graphics*, No. Vol. 24, No. 4, pp. 1303–1331, 2005.

[10] Ivana Mikic, Mohan Trivedi, Edward Hunter, and Pamela Cosman. Human Body Model Acquisition and Tracking Using Voxel Data. *International Journal of Computer Vision*, No. Vol. 53, No. 3, pp. 199–223, 2003.

[11] Roland Kehl, Matthieu Bray, and Luc Van Gool. Full Body Tracking from Multiple Views Using Stochastic Sampling. In *Proc. CVPR*, Vol. 2, pp. 129–136, 2005.

[12] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2005.

[13] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man and Cybern*, Vol. SMC-9, No. 1, pp. 62–66, 1979.

[14] Marcel Kortgen, Gil-Joo Park, Marcin Novotni, and Reinhard Klein. 3D Shape Matching with 3D Shape Contexts. In *Proc. WSCG*, 2003.

[15] Kohsia S. Huang and Mohan M. Trivedi. 3D Shape Context Based Gesture Analysis Integrated with Tracking using Omni Video Array. In *Proc. CVPRW*, Vol. 3, p. 80, 2005.

[16] Thomas Kailath. The Divergence and Bhattacharyya Distance Measures in Signal Selection. *IEEE Trans. on Comm. Technology*, Vol. 15, pp. 52–60, 1967.

[17] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. In *Proc. VLDB*, pp. 518–529, 1999.

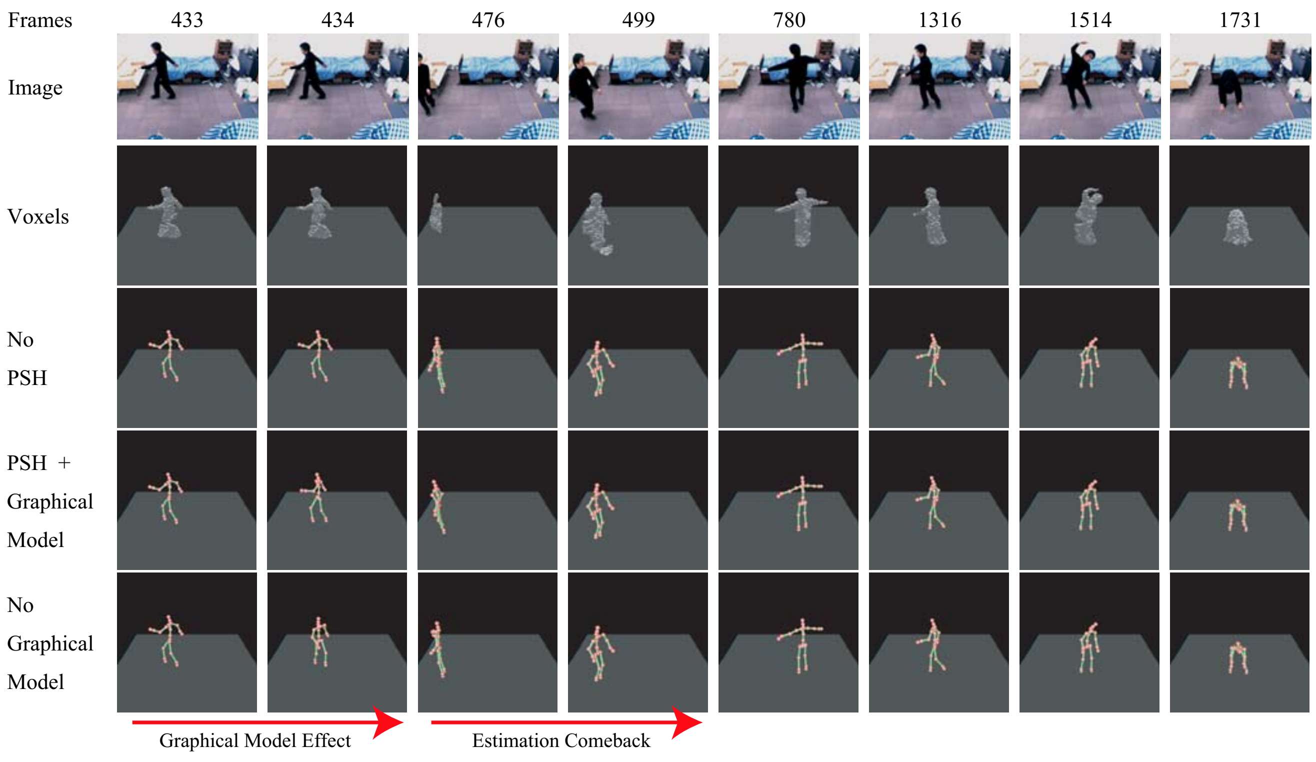

Fig. 7. Human Pose Estimation