3D Voxel Based Online Human Pose Estimation via Robust and Efficient Hashing

Masamichi Shimosaka, Yuichi Sagawa, Taketoshi Mori and Tomomasa Sato

Abstract— **In this paper, we present a novel framework to recover human body pose on multi camera systems. Our framework leverages 3D voxel data, which are reconstructed from multi-camera systems. 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 environments. We treat the pose estimation problem as estimating *human pose label* from the voxel features and tackle this by example based approach. To ensure the real-time speed and to improve precision of pose estimation, a newly fast and robust near-neighbor search metric is installed prior to the evaluation process, what we call CSI-PSH. We demonstrate the effectiveness of our approach with experiments on both synthetic and real image sequences.**

I. Introduction

In recent years, researches on home environment with distributed sensors and affluent databases are active, and believed to provide practical applications. Aware Home [1] 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, suits or markers, we employ marker-less motion capture framework on multi-camera systems. Multi-cameras are able to lighten the ambiguity in human pose estimation compared to monocular camera approaches. Especially, 3D voxel reconstruction, which is possible through multiple silhouette image integration, leads to viewpoint-free estimation.

Inspired by the success of example-based monocular view pose estimation [2], we treat the pose estimation problem as retrieval of most likely *human pose label* from the voxel. Human pose codebook is constructed beforehand from a large motion capture database, and the most appropriate posture codebook is retrieved per frame by comparing the likelihoods between 3D voxel feature and posture candidates.

In general, precise pose estimation requires the increase of number of examples. The number of the examples is directly related to the total computational cost. This trade-off between precision and computational cost must be carefully considered. The main contribution of the paper is to improve both efficiency and quality of example-based pose estimation framework. We overcome this dilemma by introducing a newly proposed fast near-neighbor search metric called CSI-PSH. This metric highly reduces codebook candidates during the matching phase and then leads to reduction of computational cost. CSI-PSH is a natural extension of a previously proposed algorithm called PSH (*parameter sensitive hashing*) [2]. CSI-PSH has an advantage over basic PSH in memory consumption, level of narrowing, and stability. The system with CSI-PSH will be able to deal with more than 300,000 examples with 432 dimensional feature vector in real-time performance of 30 fps where PSH-based systems [3] cannot handle.

A. Related works

Image-based human pose estimation has been a hot trend in the computer vision domain. Estimation approaches can be classified to generative (or model based) and discriminative approaches. In generative approaches [4], [5], an explicit model which is similar to the target is usually designed and an error measure between model and observation is defined and minimized at each frame. Generative approaches are well known to be flexible and accurate towards complex and unknown motions, but also computationally expensive. They also require a good initialization (mostly manual) and few of them can recover from tracking failures. On the other hand, discriminative approaches model and predict state condition directly from observation. Difficulty has been suggested in assuring sufficiency in the mapping between observation and state condition, but easier model construction and speed-up of the estimation process are notable characteristics. Moreover, discriminative approaches can be further classified into regression and example based approaches. Regression based approaches [6], [7] directly estimate human body joint angles and are known to be precise, but are also unstable to large training data. In contrast, example based approaches [2], [8] output discrete human pose, and it's precision is inferior to regression based approaches in some occasions. However, this approach is able to deal with large training data, and speed-up can be easily achieved due to the simplicity of the framework.

II. Framework of example-based human pose estimation from 3D voxel

A. Basic concept

Our framework of human pose estimation is based on the example based approach. In the human pose estimation phase, 3D voxel data $\boldsymbol{v}(t)$ is designed to be the input data. On the other hand, instead of outputting human body joint angle data $\boldsymbol{\theta}(t)$ in continuous quantity, output data $\boldsymbol{\theta}_{y(t)}$ is designed to be discrete. $y(t)$ indicates the estimated label of time t, which represents one of the N_y posture codebook $\{\boldsymbol{\theta}_j\}_{j=1}^{N_y} \equiv$

$\mathcal{Y}$ that are calculated beforehand. This configuration possibly causes a sparse inference result, which is an arguable point, but increase in posture candidates would lead to denseness of human posture state. Thus, example based inference can be considered as a sufficient approximation of continuous inference methods [8].

In this framework, inference is simplified to a comparison of likelihoods between 3D voxel data $\boldsymbol{v}(t)$ and human posture candidates $\mathcal{Y}$. For likelihood calculation, a feature vector $\boldsymbol{q}(t)$ is extracted from $\boldsymbol{v}(t)$ via function $Q(\boldsymbol{v})$.

$$\boldsymbol{q}(t) = Q(\boldsymbol{v}(t)) \tag{1}$$

On the other hand, posture candidate $\boldsymbol{\theta}_j \in \mathcal{Y}$ is preprocessed into artificial voxel data $\boldsymbol{v}_j$ via function $V(\theta)$, which enables feature vector $\boldsymbol{q}_j$ to be extracted.

$$\boldsymbol{v}_j = V(\boldsymbol{\theta}_j) \tag{2}$$

$$\boldsymbol{q}_j = Q(\boldsymbol{v}_j) = Q(V(\boldsymbol{\theta}_j)) \tag{3}$$

In this way, $\boldsymbol{q}(t)$ (query feature vector) and $\{\boldsymbol{q}_j\}_{j=1}^{N_y} \equiv \mathcal{Q}$ (candidate feature vectors) are extracted. And then, likelihoods $\{\phi_j(t)\}_{j=1}^{N_y}$ between $\boldsymbol{v}(t)$ and $\mathcal{Y}$ is evaluated through function $S(\boldsymbol{q}, \tilde{\boldsymbol{q}})$.

$$\phi_j(t) = S(\boldsymbol{q}(t), \boldsymbol{q}_j) \tag{4}$$

Inference per frame is possible by selecting the label $y(t)$ that outputs the maximum value among likelihoods $\{\phi_j(t)\}_{j=1}^{N_y}$.

$$y(t) = \underset{j}{\operatorname{argmax}}\, \phi_j(t) \tag{5}$$

However, not making consideration of Markov property leads to improper transition and motion oscillation. Thus, a graphical model of motion is embedded to provide a smoothing effect. First order Markov property is represented by a directed graphical model of motion. Motion sequences are scanned, and each frame is referenced with a posture label. During the scanning process, the transition frequency between previous label i and current label j is accumulated, and used to eventually derive the transition probability $T_{i,j}$. In order to prevent getting caught into a local solution, $T_{i,j}$ is binarized as $0,\ 1$. The graphical model of motion is preprocessed after constructing posture candidates, and used in the online inference process like Viterbi decoding [9]. The maximization of posterior probability, label $y(t)$ that outputs the maximum value among accumulated likelihoods $\{p_j(t)\}_{j=1}^{N_y}$ is calculated as

$$p_j(t) = \begin{cases} \phi_j(t) & (t = 0) \\ \max\limits_i (p_i(t-1)T_{i,j}) + \phi_j(t) & (t \neq 0) \end{cases} \tag{6}$$

$$y(t) = \underset{j}{\operatorname{argmax}}\, p_j(t). \tag{7}$$

The estimation scheme mentioned above is summarized as Fig. 1. The key to success for real-time estimation is to reduce matching calculation $\phi_j(t)$. This will be described in next section III. The creation process of 3D voxel data, posture codebook, feature extraction, and similarity calculation will be presented in the following.

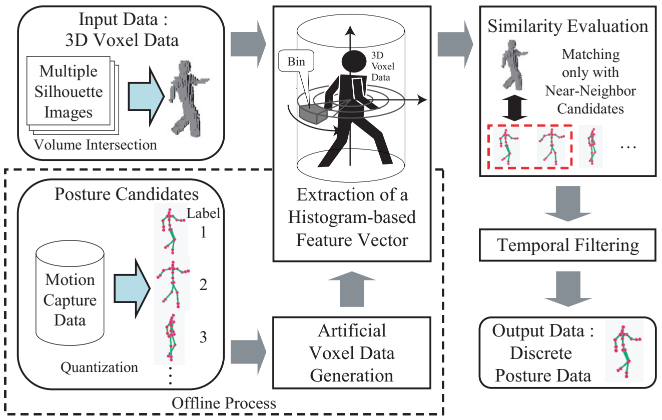

Fig. 1. Outline of the Human Pose Estimation Process

B. Preliminaries of the framework

1) Making voxels via volume intersection: The volume intersection method [10], [11] reconstructs 3D object shape from multiple silhouette images. 3D shape will consist of an assembly of voxels, which represent cubes of the divided 3D space. 8 cameras are used to capture multiple VGA images. Voxel size is set to 35mm, and 3D space is divided into a $120\times120\times68$ resolution.

2) Construction of pose codebook: Posture codebook is constructed using a large motion capture database. Representative candidates are extracted through a clustering process. We use a kind of *agglomerative hierarchical clustering* (distance between clusters is defined based on the *group average method*) [9]. This makes it possible to acquire a more uniform and dense distribution in human posture space. Motion capture data downloaded from www.mocapdata.com are used (motion data include large body rotations, complex motions, and self occlusions), and resulted in a total of 322,992 posture codebook.

3) Feature extraction from voxel and pose codebook: To calculate the similarity between voxel and pose codebook, functions $Q(\boldsymbol{v})$ and $V(\boldsymbol{\theta})$ must be clarified. For $Q(\boldsymbol{v})$, we use simple histogram-based feature called *cylindrical histogram feature* [3] (see Fig. 1). In this feature, 3D space is divided into multiple bins based on a central axis set to the center of gravity of voxel data (3D space is divided in angle, height, and radius directions). Voxels are voted to the corresponding bins, and the resulting histogram will be normalized. Resolution of angle, height, and radius directions are set to 18, 8, and 3. Thus the dimension of the feature vector became 432.

For $V(\boldsymbol{\theta})$, 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.

4) Likelihood Evaluation Based on Feature Vectors: Likelihood evaluation is based on the Bhattacharyya coefficient [12]. Likelihood between a query feature vector $\boldsymbol{q}(t)$ and a candidate feature vector $\boldsymbol{q}_j \in \mathcal{Q}$ of label j is calculated from the equation below ($\boldsymbol{q} \in \mathbb{R}^{r_q}$),

$$\phi_t(j) = \sum_{r=1}^{r_q} \sqrt{\{\boldsymbol{q}(t)\}_r \{\boldsymbol{q}_j\}_r}. \tag{8}$$

III. Schemes for speeding up estimation: Cascaded Sparse Incremental PSH

Since the number of pose codebook is extremely large, it is impossible to maintain the real-time performance with the same estimation scheme. In this section, schemes for speeding up the estimation process will be presented in detail.

PSH (*parameter sensitive hashing*) [2] is well known and used technique as a near-neighbor search metric. In this paper, we newly propose an algorithm called CSI-PSH (*cascaded sparse incremental PSH*), which solves some disadvantages in PSH. In particular, CSI-PSH has advantages in memory consumption, level of narrowing, and stability. At first, prerequisites of fast near-neighbor search metrics are described. This is followed by a brief review of the PSH algorithm. And then, the mechanism for improving the performance of PSH in terms of memory consumption, level of narrowing, and stability will be explained.

A. Prerequisites of fast near-neighbor search

In the following discussions, a fast near-neighbor search metric will indicate an algorithm that searches for posture label sets that are similar to the query feature vector $\boldsymbol{q}$ in high probability.

Level of narrowing is evaluated by query ratio, which is the ratio of candidates that are retrieved in search. Lower query ratio corresponds to higher level of narrowing. In other words, higher query ratio increases redundancy of the evaluation. Another measure of search is the recall ratio, which is the ratio of successfully retrieved candidates that are similar to the query. The higher the recall ratio, the lower the probability of leaks will be in search.

Let $\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}$ be human posture data in joint angle format. Through forward kinematics, human posture data in joint position format can be calculated, which is notated as $\boldsymbol{x}, \tilde{\boldsymbol{x}}$. We evaluate the similarity between two postures by a distance metric $d_x(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ which is formulated as a mean distance of joint positions. Whether two postures are similar or not is defined by parameter R, and postures that satisfy $d_x(\boldsymbol{x}, \tilde{\boldsymbol{x}}) < R$ will be assumed to be similar. In addition, feature vectors $\boldsymbol{q}, \tilde{\boldsymbol{q}}$ are extracted through artificial voxel data generation.

B. Parameter Sensitive Hashing (PSH)

PSH is an extended version of a similar metric named LSH (*locality sensitive hashing*) [13]. LSH is known to offer an explicit connection between error expectation and cost. The difference between these two algorithms is the space where similarity of search is considered. LSH defines similarity in input space (feature space), while PSH considers it in parameter space (human posture space). In a human pose estimation framework, similarity is desired to be considered in parameter space, so PSH is preferred. When PSH is combined with the actual matching process (in input space), similarity in both input and parameter space will be considered, and the quality of the overall matching process will be enhanced.

PSH is an algorithm that is based on hash functions and hash tables. L hash functions and hash tables are prepared, and each hash function $h(\boldsymbol{q})$ translates a query into a hash value. The hash value denotes an address in the hash table where data is stored. During the preliminary phase, posture candidate data (posture labels) are registered to every hash table via hash function. During search, registered data over all hash tables are collected to form the search result. Quality of hash functions will influence the search result, so the goal of training in PSH is to construct hash functions that are sensitive in parameter space.

A hash function $h(\boldsymbol{q})$ is defined as shown in Eq. (9). $b(\boldsymbol{q})$ is a binary function called a *decision stump*, and K of them are concatenated to form a hash function. Moreover, a decision stump $b(\boldsymbol{q})$ is defined by an index m of vector $\boldsymbol{q}$ and threshold T_m (see Eq. (10) [1]). Decision stumps are calculated through an optimal calculation [2], which uses pair data as training data. Training pairs $\{(\boldsymbol{q}^{(n)}, \tilde{\boldsymbol{q}}^{(n)}, z^{(n)})\}_{n=1}^{N_t}$ are associated with a label z that indicate whether the pairs are similar or not in parameter space (see Eq. (11)). ϵ is a parameter to assure that $d_x(\boldsymbol{x}, \tilde{\boldsymbol{x}}) < R$ will be satisfied between query and search result in high probability (this property is called *locality-sensitive* [2], [13]). After the optimal calculation, 432 decision stumps (same number to feature dimension) are obtained. The obtained decision stumps are associated with scores that describe sensitivity in parameter space. So top M decision stumps are chosen (to form a sensitive decision stump set $\boldsymbol{\mathcal{B}}$), and randomly allocated to L K-bit hash functions (allows duplication).

L and K are important parameters that significantly affect the performance of near-neighbor search. However K is manually set based on preliminary experiments, L is automatically determined during an iterative construction of hash function until the recall ratio of training pairs surpasses parameter T_R. The recall ratio is derived by considering the ratio of similar pairs that satisfy hash uniqueness (see Eq. (12)).

$$h(\boldsymbol{q}) = [b^{(1)}(\boldsymbol{q}), b^{(2)}(\boldsymbol{q}), \cdots, b^{(K)}(\boldsymbol{q})]^{\mathsf{T}} \tag{9}$$

$$b_{m,T_m}(\boldsymbol{q}) = \begin{cases} 1 & \text{if } \{\boldsymbol{q}\}_m \geq T_m, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

$$z = \begin{cases} +1 & \text{if } d_x(\boldsymbol{x}, \tilde{\boldsymbol{x}}) < R/(1+\epsilon), \\ -1 & \text{if } d_x(\boldsymbol{x}, \tilde{\boldsymbol{x}}) > R, \\ \text{not defined otherwise.} \end{cases} \tag{11}$$

$$h(\boldsymbol{q}^{(n)}) = h(\tilde{\boldsymbol{q}}^{(n)}) \quad \text{s.t.} \quad z = +1 \tag{12}$$

C. Incremental update of training pairs: I-PSH

Query ratio tends to be higher when L is larger. In addition, hash tables consume huge memory so L is required to be as low as possible. In PSH, every hash function

[1] $\{\boldsymbol{q}\}_m$ denotes the m-th element of vector $\boldsymbol{q}$

is constructed randomly and independently from the same decision stump set $\boldsymbol{\mathcal{B}}$. Therefore, it is probable that similar types of hash functions are wastefully defined. Such redundancy leads to larger values of L. At each step (iteration), training pairs are narrowed down based on evaluation of hash uniqueness (dissimilar pairs will not be updated) (see Fig. 2). This is followed by the optimal calculation process which outputs a newly updated sensitive decision stump set $\boldsymbol{\mathcal{B}}$. In this way, every hash function will be constructed from different decision stump sets, and the redundancy will be lightened, resulting in smaller number of hash functions. We introduce this approach to PSH and call the algorithm I-PSH (*incremental PSH*).

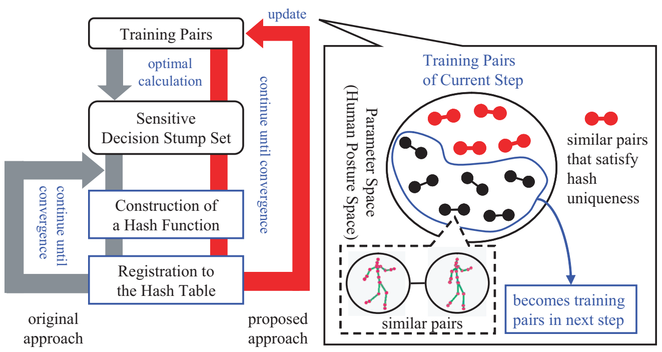
Training Pairs
optimal calculation
Sensitive Decision Stump Set
Construction of a Hash Function
Registration to the Hash Table
continue until convergence
original approach
update
continue until convergence
proposed approach
Training Pairs of Current Step
Parameter Space (Human Posture Space)
similar pairs that satisfy hash uniqueness
similar pairs
becomes training pairs in next step

Fig. 2. Incremental Update of Training Pairs

D. Sparse registration to Hash tables: SI-PSH

Query ratio is required to be kept as low as possible, because it directly affects the computational cost of matching. High query ratio is believed to attribute to conflicts in hash values. At first, all posture labels are registered to the hash table. Then, based on a feedback in human posture space using an evaluation dataset $\{\boldsymbol{x}^{(n)}, \boldsymbol{q}^{(n)}\}_{n=1}^{N_e}$, only partial posture labels will be left for registration. Sparseness of registration is decided by parameter T_S (posture labels are registered until the query ratio for the evaluation dataset reaches T_S) (see Fig. 3). In this way, we achieve a sparse registration of posture labels, which leads to fewer conflicts in hash values and lower query ratio. We introduce this approach to I-PSH and call the algorithm SI-PSH (*sparse incremental PSH*).

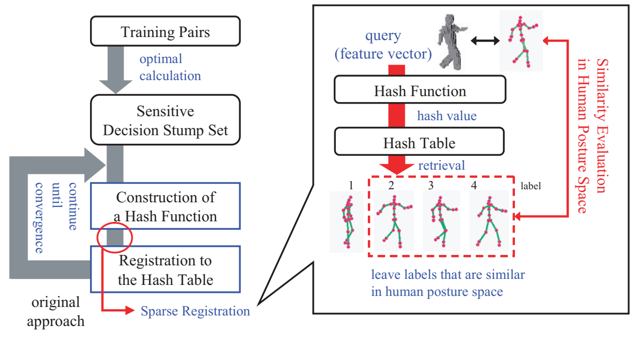
Training Pairs
optimal calculation
Sensitive Decision Stump Set
Construction of a Hash Function
Registration to the Hash Table
continue until convergence
original approach
Sparse Registration
query (feature vector)
Hash Function
hash value
Hash Table
retrieval
1
2
3
4
label
leave labels that are similar in human posture space
Similarity Evaluation in Human Posture Space

Fig. 3. Sparse Registration to Hash Tables

E. Cascaded search for stability: CSI-PSH

In PSH, query ratio depends on the property of the query, so it is not possible to consistently maintain a fixed query ratio for various queries. However, in a real-time system, it is unsuitable to allow such a fluctuation in query ratio because it directly influences the processing time of matching and the overall estimation process. So an approach that is capable of maintaining the query ratio to a certain fixed value T_Q is desired. We present an cascaded search approach which consists of two tactics: hash table with confidence scores (see Fig. 4), and redundant hash values (see Fig. 5). This cascaded search approach diverges in two ways depending on the initial search conducted in a normal way. If the query ratio of the initial search surpasses T_Q, the search result will be cut down based on confidence scores associated with hash values. If it does not surpass T_Q, range of search will be expanded based on cascaded search with redundant hash values. We introduce this approach to SI-PSH and call the algorithm CSI-PSH (*cascaded sparse incremental PSH*).

The idea is to rank search result elements based on a confidence score associated to hash values. Confidence is calculated by the similarity between two postures. When search results are ranked, it becomes possible to cut down the search result based on the ranking. The confidence score indicates a higher ranking when it's value is lower, and is calculated based on a feedback using the evaluation dataset $\{\boldsymbol{x}^{(n)}, \boldsymbol{q}^{(n)}\}_{n=1}^{N_e}$.

A redundant hash value is a hash value where partial bits have been reversed. In this way, addresses to search in a hash table will increase and lead to expansion of search. Number of bits that are reversed correspond to the stage of cascade in search. Each bit in L hash values is given a priority based on the proximity score p_m in decision stump $b_{m,T_m}(\boldsymbol{q})$ (see Eq. (13)), and bits that have higher priority will be reversed preferentially (when two or more bits are going to be reversed, priority will be determined based on the average proximity score). Proximity score p_m indicates whether $\{\boldsymbol{q}\}_m$ and T_m are close, and lies between 0 and 1. Higher proximity scores signify that the two values are close, and when the values are closer, the bit will have higher priority (reversing the bit becomes more reasonable). Moreover, not all the bits are allowed to be reversed since the combination of bits will become massive if it does. Based on parameter κ, only bits that satisfy $p_m > \kappa$ will be allowed to be reversed.

$$p_m = \begin{cases} 0 & \text{if } \{\boldsymbol{q}\}_m = 0, \\ \frac{\min(\{\boldsymbol{q}\}_m, T_m)}{\max(\{\boldsymbol{q}\}_m, T_m)} & \text{otherwise.} \end{cases} \tag{13}$$

F. Evaluation of fast near-neighbor search metrics

Parameters are set to R=50mm, ϵ=0.25, M=60, T_R=0.95, T_S=0.04, T_Q=0.01, κ=0.33, while 579,731 similar pairs with 174,937 dissimilar pairs were prepared (total of 754,668 pairs) for training pairs. Additionally, 233,062 elements of evaluation data were prepared for feedback training and hash functions were trained under these conditions. Table. II show

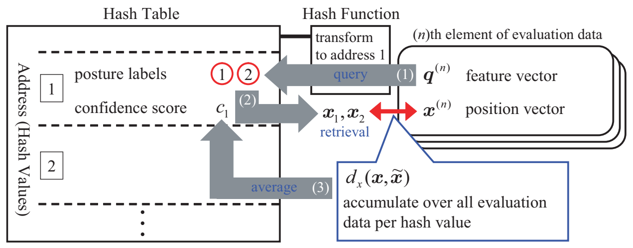

Fig. 4. Hash Table with Confidence Scores

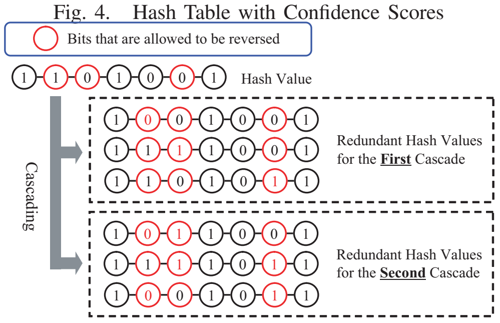
Bits that are allowed to be reversed
Hash Value
Cascading
Redundant Hash Values for the First Cascade
Redundant Hash Values for the Second Cascade

Fig. 5. Example of Redundant Hash Values

numbers of hash functions it took for each metric until convergence. CSI-PSH had a smaller number of hash functions (hash tables) than PSH, which lead to massive reduction in memory consumption (also note that each hash table of CSI-PSH consumes smaller amount of memory than that of PSH, due to the sparsity of hash tables). Additionally, effect of CSI-PSH can be seen in Fig. 6 where transitions of query ratio on a real voxel data sequence are presented. Metrics except CSI-PSH show very unstable results. Especially on PSH, there are two very huge peaks in around frame 250 and 350. Such peaks (high query ratio) will lead to extremely high computational cost of matching, and cannot be tolerated in a real-time system. In contrast, query ratio of CSI-PSH is consistently lower than that of PSH and is maintained near T_Q. Therefore, computational cost of matching will be lower than PSH and more stable as well. Moreover, query ratio of SI-PSH is mostly lower than that of CSI-PSH, but it does not directly indicate that SI-PSH has a better search performance. Since, too low query ratio lead to higher possibility of search leaks, it is important to assure that query ratio will not be too low. Hence, CSI-PSH not only provides stability in computational cost but also in performance (possibility of search leaks is suppressed).

TABLE I
NUMBER OF HASH FUNCTIONS

PSH	I-PSH	SI-PSH	CSI-PSH
342	144	198	198

IV. EXPERIMENTAL RESULT

A. Evaluation on artificial data

A human pose estimation experiment was conducted using artificial voxel data generated from motion capture data. The test motion set consists of 5484 frames of unknown motions which contain basic and dynamic motions such as walk, jump, sit, and skip. Results are shown in Table. II. Mean distance of joint positions (*Pos. Error*) and mean RMS error of joint angles (*Ang. Error*) [6] were used as error metrics, while processing time of the human pose estimation phase was measured.

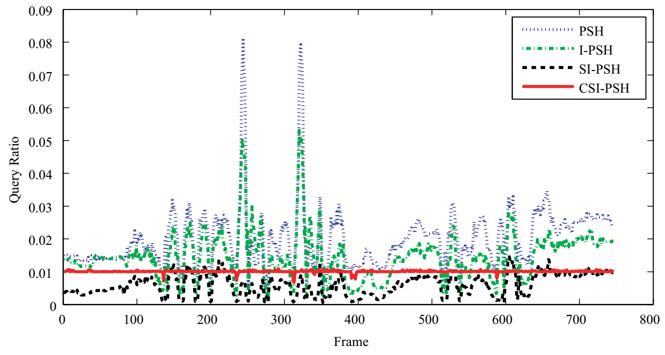

Fig. 6. Transition of Query Ratio

The **With CSI-PSH** setting corresponds to the proposed approach, and experiments on other settings are conducted for comparison. Precision of the **With SI-PSH** setting are lower than that of the **With CSI-PSH** setting. This attributes to search leaks in SI-PSH, and signifies that CSI-PSH can recover from such errors. The **With PSH** setting show similar results to the **With CSI-PSH** setting, but real-time processing will not be possible in the current configuration because processing time exceeded 33 ms. The **No FastNN Search** setting, was conducted by not applying fast near-neighbor search metrics. Precision of the **No FastNN Search** setting differ slightly to that of the **With CSI-PSH** setting, while processing time differ greatly. Hence, reasonableness of near-neighbor search can be confirmed.

Agarwal's approach [6] is a representative of regression based approaches which are known to be precise. In [6], an *Ang. Error* evaluation on a walking sequence was conducted and the error was reported to be 6.0 deg. This value is better than the proposed approach, but when limiting the test motion set to walking, the error was 6.9 deg. Thus, the proposed approach nearly reached precision of regression based approaches while offering better performance in speed. Also note that the model in [6] was trained only with walking sequences, which indicates that it was not a completely fair comparison (precision is believed to fall when the model becomes more complex). Taycher's approach [8] is a representative of example based approaches which are known to be relatively fast. In [8], a *Pos. Error* evaluation was conducted and the error was reported to be 100mm∼250mm. Therefore, the proposed approach has much better precision, and can be mentioned that it is provided with both precision and speed.

B. Experiment on real image sequences

An experiment was conducted on a daily life simulated space, using 8 cameras and capturing images in 30 fps. We leverage parallel pipe-line framework with 3 stages computation. The overall processing time is presented in Table. III. Each stage has completed within 33 msec, which

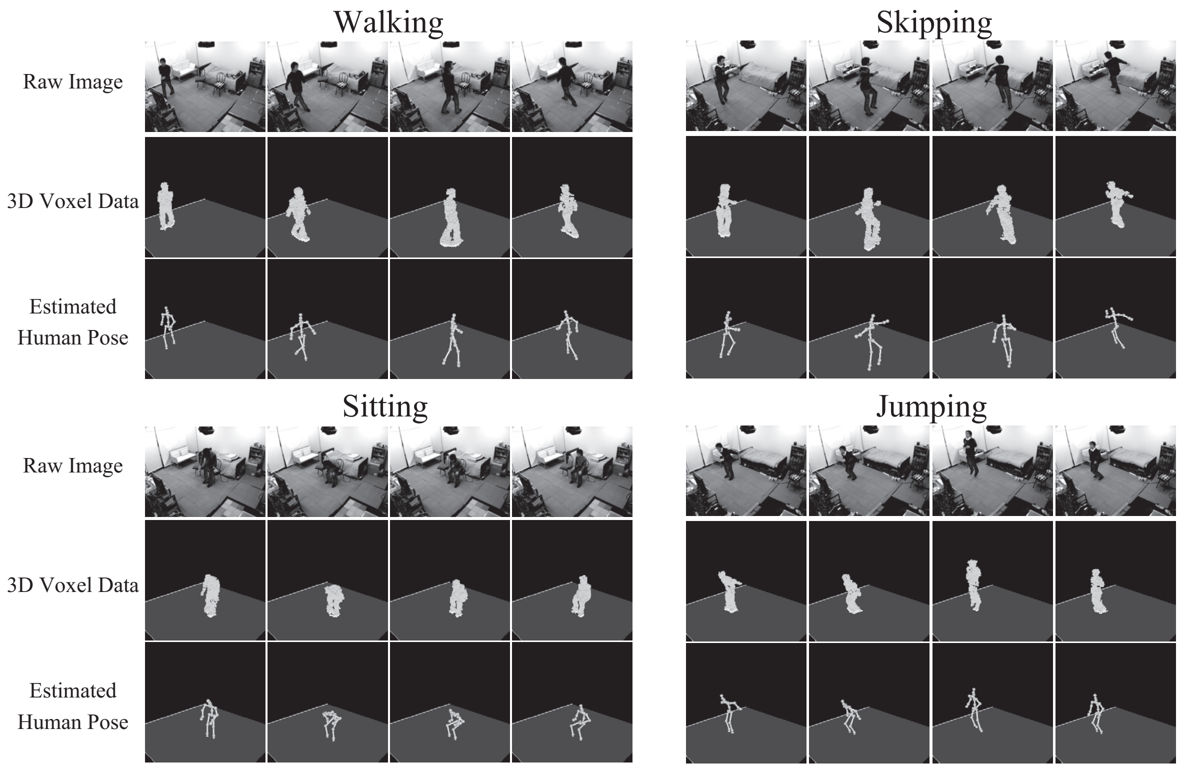

Fig. 7. Experimental Results on Human Pose Estimation

TABLE II
EVALUATION OF ESTIMATION ERROR AND PROCESSING TIME

	Pos. Error	Ang. Error	Processing Time
With CSI-PSH	**46.0 mm**	**8.53 deg**	**21.3 msec**
With SI-PSH	47.5 mm	8.60 deg	15.5 msec
With PSH	46.0 mm	8.43 deg	39.6 msec
No FastNN Search	45.6 mm	8.33 deg	1382 msec

signifies that online processing of 30 fps has been achieved (however, certain amount of delay will occur). Experimental Results are presented in Fig. 7. This implies that our system successfully capture various complex motion in real-time.

TABLE III
PROCESS TIME OF THE OVERALL PIPELINE PROCESS

	Process Time
Background Subtraction	16 msec
Visual Hull Extraction	10 msec
Stage 1 Total	**26 msec**
Visual Hull Integration	6 msec
Labeling (Noise Removal)	3 msec
Stage 2 Total	**9 msec**
Human Pose Estimation	21 msec
Render	5 msec
Stage 3 Total	**26 msec**
Delaying Frames	3

V. 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. Future tasks are to realize multiple-person tracking and to tackle the occlusion problem by furniture for practical daily house use.

REFERENCES

[1] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner, and W. Newstetter, "The Aware Home: A living laboratory for ubiquitous computing research," in *Proc. of CoBuild*, 1999, pp. 191–198.

[2] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter sensitive hashing," in *Proc. of ICCV*, vol. 2, 2003, pp. 750–757.

[3] Y. Sagawa, M. Shimosaka, T. Mori, and T. Sato, "Fast Online Human Pose Estimation via 3D Voxel Data," in *Proc. of IROS*, 2007, pp. 1034–1040.

[4] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman, "Human body model acquisition and tracking using voxel data," *International Journal of Computer Vision*, vol. 53, no. 3, pp. 199–223, 2003.

[5] R. Kehl, M. Bray, and L. V. Gool, "Full Body Tracking from Multiple Views Using Stochastic Sampling," in *Proc. of CVPR*, vol. 2, 2005, pp. 129–136.

[6] A. Agarwal and B. Triggs, "3D human pose from silhouettes by relevance vector regression."

[7] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas, "Discriminative density propagation for 3D human motion estimation," in *Proc. of CVPR*, vol. 1, 2005, pp. 390–397.

[8] L. Taycher, G. Shakhnarovich, D. Demirdjian, and T. Darrell, "Conditional random people: Tracking humans with CRFs and grid filters," in *Proc. of CVPR*, vol. 1, 2006, pp. 222–229.

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

[10] W. N. Martin and J. K. Aggarwal, "Volumetric description of objects from multiple views," *IEEE Trans. on PAMI*, vol. 5, no. 2, pp. 150–158, 1983.

[11] A. Laurentini, "How far 3D shapes can be understood from 2D silhouettes," *IEEE Trans. on PAMI*, vol. 17, no. 2, pp. 188–195, 1995.

[12] T. Kailath, "The divergence and Bhattacharyya distance measures in signal selection," *IEEE Trans. on Comm. Technology*, vol. 15, pp. 52–60, 1967.

[13] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. of VLDB*, 1999, pp. 518–529.