2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Synthesizing Object Receiving Motions of Humanoid Robots with Human Motion Database

Katsu Yamane[1], Marcel Revfi[2], and Tamim Asfour[2]

Abstract— **This paper presents a method for synthesizing motions of a humanoid robot that receives an object from a human, with focus on a natural object passing scenario where the human initiates the passing motion by moving an object towards the robot, which continuously adapts its motion to the observed human motion in real time. In this scenario, the robot not only has to recognize and adapt to the human action but also has to synthesize its motion quickly so that the human does not have to wait holding an object. We solve these issues by using a human motion database obtained from two persons performing the object passing task. The rationale behind this approach is that human performance of such a simple task is repeatable, and therefore the receiver (robot) motion can be synthesized by looking up the passer motion in a database. We demonstrate in simulation that the robot can start extending the arm at an appropriate timing and take hand configurations suitable for the object being passed. We also perform hardware experiments of object handing from a human to a robot.**

I. Introduction

Cooperative manipulation between a robot and a human is an essential skill for household robots. Among a wide range of tasks that can be categorized as cooperative manipulation, we are interested in human-to-robot object passing with a style similar to human-to-human passing (Fig. 1) where the human first starts moving the object towards the robot, which then reaches out its hand to receive the object. We assume that, for anthropomorphic robots, human-like motions are more user-friendly because of the familiarity even though they may not be optimal with respect to physical measures.

While humans can easily receive an object in this manner, it would be very challenging for a robot because of a few reasons. First, because the action is initiated by the human, the robot has to correctly recognize the beginning of the passing action. Second, the goal location of the hand is not given in advance and has to be determined by observing the human motion. Lastly, the planning has to be performed quickly so that the human does not have to wait for the robot's hand to come close enough for hand-off.

We solve these problems by applying a human motion database, assuming that human-to-human object passing is repeatable when the relative position and orientation of the two humans are similar, and therefore we can infer an appropriate receiver motion by observing the passer motion. If this is the case and we have a library of human-to-human object passing motions, we can use a "table lookup" approach to pull up the typical receiver motion that corresponds to an observed passer motion. Unfortunately, the table is usually too large to look up in realtime and therefore requires a sophisticated data structure and search algorithm.

Fig. 1. Natural human-to-human object passing motion.

In this paper, we employ a hierarchical database structure developed in [1] to solve the database size issue. The structure incorporates the features of database structures developed in information theory, animation, and robotics: 1) the observed poses are organized into the binary-tree structure for efficient search, 2) the nodes at the same depth are connected as a directional graph structure similar to motion graphs [2], [3], and 3) each node and edge are associated with pose distribution and transition probability respectively, similarly to Hidden Markov Models (HMMs) [4]–[6].

The original database structure only addressed the problem of storing and searching for motions of a single subject. We therefore make a few modifications to make it suitable for our application of synthesizing human-to-robot object passing motions as described in Section IV. Firstly, the database is now constructed from two-subject motion capture data, where a first subject passes an object to a second subject. Secondly, the binary tree is generated based on the passer's example motions instead of both subjects so that we can efficiently search the database only using the passer motion as the key. By building the database in this way, we can synthesize the robot's motion by 1) observing the passer human motion by, for example, a vision system, 2) finding a node sequence that matches the observation from the motion database with an improved search algorithm (Section V-B), and 3) computing the robot's motion based on the node sequence found in the previous step (Section V-C).

II. Related Work

Human-robot cooperative manipulation has been studied in a few different forms. One form is carrying a large object together [7]–[9], where the research focus has been on estimating the human intention and controlling the robot to follow the human while compensating for unexpected noise. Another form is passing an object to a human. Sisbot et al. [10], for example, developed a motion planning algorithm

[1]K. Yamane is with Disney Research, Pittsburgh. kyamane@disneyresearch.com
[2]M. Revfi and T. Asfour are with Karlsruhe Institute of Technology. marcel@revfi.de, asfour@kit.edu

978-1-4673-5642-8/13/$31.00 ©2013 IEEE

that allows a robot to hand an object in such a way that the human does not feel uncomfortable or threatening.

This paper focuses on yet another task where a robot receives an object from a human. Edsinger et al. [11] implemented two scenarios for this task, but they required the user to follow specific protocols. Micelli et al. [12] addressed the problem of realizing more natural human-to-robot object passing. We also have the same goal, but our approach takes advantage of prior knowledge on human-to-human object passing behavior, which may relax some of the sensing and perception challenges mentioned in [12].

Part of the human motion database developed in [1] has a structure often referred to as motion graphs [2]. Furthermore, the motion synthesis algorithm, which is our new contribution, interpolates the states of multiple nodes to obtain better results, similarly to the work by Safonova et al. [3] for motion synthesis of a single character.

While our work primarily focus on human-to-robot object passing, it has close relationship with existing methods for modeling the correlation between motions of human and robot or different body parts. Takano et al. [4] presented an HMM-based method for learning interactive behaviors from motion capture data of two humans. Lee et al. [5] developed a learning algorithm where a robot can imitate human motions by observing only a part of the human body based on the prior knowledge of human whole-body motions. It was later extended to active learning of physical interaction between a human and a robot [6]. Shukla et al. [13] proposed a method for learning correlation between different parts of the body using coupled dynamical systems.

III. Overview of the Original Database

This section reviews the database structure and search algorithm originally presented in [1].

A. Database Structure (Fig. 2)

Assume that we have one or more sample motion clips from motion capture or keyframe animation, and they are represented as state-space trajectories (the curves in the top figure). By sampling the trajectories at a fixed time step, we obtain a collection of sample states (the dots in the top figure). Starting from the top layer with a single cluster including all sample states, we iteratively generate a new layer by dividing each cluster into two as shown in the bottom left figure, where each cluster is represented by a gray oval depicting the mean and covariance matrix of the sample states in the cluster. This clustering method naturally generates a binary tree (bottom right figure) whose nodes correspond to the clusters. This structure allows efficient database search as described in Section III-B.

Once we know which samples belong to which node at each layer, we can connect the nodes with directed edges that represent the possible transitions among the nodes (black arrows in the bottom left figure). The nodes and edges in a single layer form a motion graph. We can also compute the transition probability of each edge by dividing the number of samples whose successors are in the destination node by the total number of samples in the origin node.

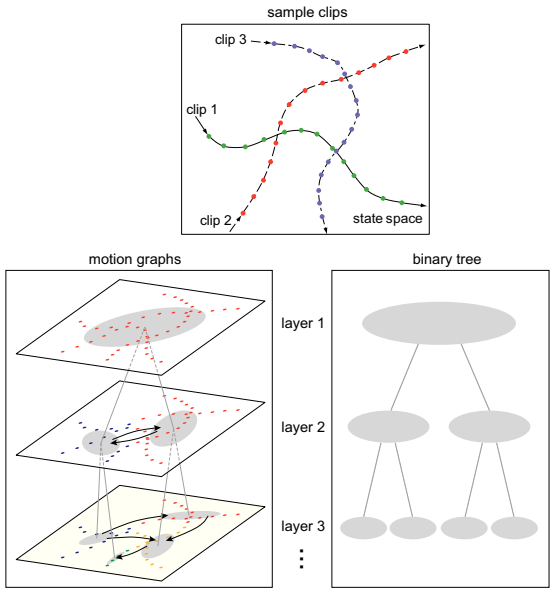

Fig. 2. Database structure. Top: sample motion trajectories represented in the state space. Bottom left: hierarchy of layers and motion graphs. Bottom right: binary tree representing the node hiearchy.

In summary, a database contains the following data:

- The state distribution of each node, usually the mean and covariance matrix assuming a Gaussian distribution.
- Parent-child relationship of the nodes in the binary tree.
- Transition probability of each edge.

The state space can be chosen to fit the properties of the motions and/or the objective of the database. Following [1], this work uses the Cartesian positions and velocities of feature points (markers) as the state space. This state space fits our objective because the hand position of the receiver, rather than the joint angles, is critical to the task. To make the database invariant to the horizontal location and orientation of the motion, the feature point positions and velocities are first normalized by translating and rotating each sample pose so that the root link of the human model is at the same location in the horizontal plane, facing the same direction.

B. Search Algorithm

For a given trajectory in the state space with r frames, the search algorithm finds the node transition $\{N_1, N_2, \ldots, N_r\}$ in a specific layer that minimizes the cost function

$$Z = -\sum_{i=1}^{r} \log P_{N_i}(\boldsymbol{x}_i) - \sum_{i=1}^{r-1} \log T(N_i, N_{i+1}) \qquad (1)$$

where $\boldsymbol{x}_i$ is the observed state at the i-th frame, $P_k(\boldsymbol{x})$ represents the likelihood that state $\boldsymbol{x}$ is generated from node k, and $T(i, j)$ is the transition probability from node i to j. Minimizing Eq.(1) results in the node transition that has the maximum likelihood of generating the given trajectory.

We illustrate the search algorithm using the example in Fig. 3, where the goal is to find the optimal node transition

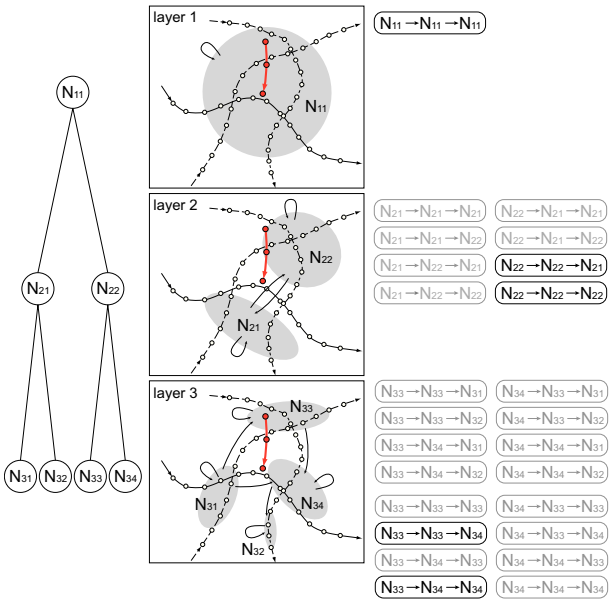

Fig. 3. Database search. Left: the binary tree, center: motion graphs, and right: possible node transitions.

in layer 3 for the trajectory with three frames shown as the red dots in each of the three figures of the middle column. The nodes in each layer are represented as N_{ij}, where i is the layer number and j is the node ID in that layer. The binary tree structure of the nodes is shown in the left figure of Fig. 3.

The search algorithm starts from the top layer and descends the binary tree until it reaches the desired layer.

The optimal node transition at the top layer is obviously $\{N_{11} \rightarrow N_{11} \rightarrow N_{11}\}$ because this layer conists of a single node. In layer 2, each of the N_{11} nodes in the node transition at layer 1 can evolve into either N_{21} or N_{22}, which yields $2^3 = 8$ possible node transitions shown in the middle row of the left column. However, most of these transitions can be eliminated by considering the probability of generating each frame at the two nodes. In this particular case, we can safely assume that the first and second frames are not generated by node N_{21} because the probabilities computed from the distribution of N_{21} are sufficiently small. We can therefore remove six node transitions shown in gray from the list, leaving only two possible node transitions.

We then proceed to layer 3. Because N_{21} has two descendants N_{31} and N_{32} while the descendants of N_{22} are N_{33} and N_{34}, each of the two node transitions at layer 2 yield eight transitions shown in the bottom figure of the right column. Again based on the probability distributions of the nodes, we can conclude that the first frame can only be generated by N_{33}, the second frame by N_{33} or N_{34}, and the third frame by N_{31} or N_{34}. We can further reduce the number of possible transitions by eliminating those with invalid transitions, such as that from N_{33} to N_{31}. Having only two remaining transitions, we can efficiently choose the node transition that minimizes Eq.(1).

IV. Database Structure for Object Passing Motion Synthesis

This section describes the new database structure we developed to adapt the original structure to our target task.

Previous work on motion graphs involved an optimization process that maps intuitive user inputs, such as walk paths [2] and start and goal configurations [3], to the character motion that best matches the input. Therefore the input and output of the optimization were both related to the character's motion space. In our work, on the other hand, they are in difference spaces: the input is in the passer's motion space and the output is in the receiver's motion space. We therefore have to modify the way we construct the database.

We build a database for human-to-robot object passing as follows (Fig. 4). We first record a number of examples of human-to-human object passing. The difference from [1] is that we do not use the entire two-person motion space for clustering because the receiver motion is not available while synthesizing the robot motion. Instead, we perform the clustering based only on the passer's sample states. The mean and covariance matrix of the passer's states included in each node are stored as in the original database.

In order to synthesize the receiver's motion, however, we also have to maintain the receiver's states corresponding to the passer's. After creating the binary tree and motion graphs using the passer's sample states, we compute and store the mean and covariance matrix of the receiver's states in the frames included in each node. We will use this information for extracting the receiver's pose that corresponds to the passer's pose.

The information included in the new database is thus summarized as follows:

- The distribution (means and covariance matrices) of the passer and receiver sample states in each node.
- Parent-child relationship of the nodes in the passer's binary tree.
- Transition probability of each edge.

The last modification concerns the normalization of the sample states. We normalize the sample states so that the receiver's root segment stays at the same horizontal location and direction, although clustering is performed based on the passer's sample states. This normalization makes more sense for our application because the receiver's motion should depend on the relative position and orientation of the passer with respect to the receiver, which will be eliminated if we normalize the samples using the passer's root segment. It also allows the use of on-board vision sensors that only gives the human pose with respect to the robot body.

V. Motion Synthesis

We now describe the algorithm for human-to-robot object passing motion synthesis based on the new database structure presented in the previous section.

A. Overview

Figure 5 summarizes the motion synthesis algorithm. We first observe the human passer's motion data, represented as

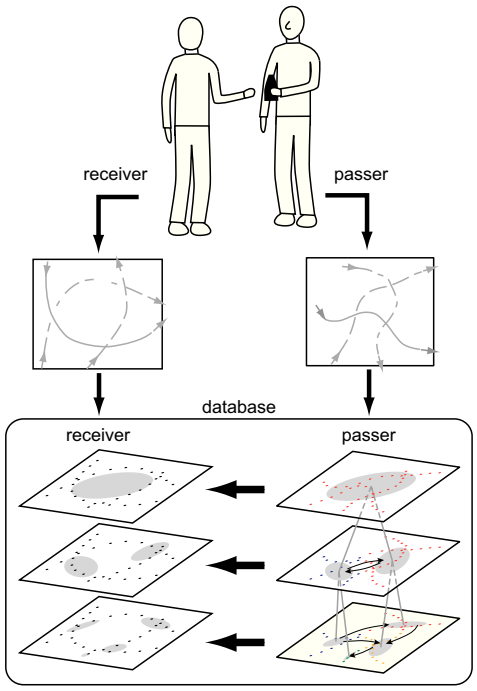

Fig. 4. Constructing a database for human-to-robot object passing.

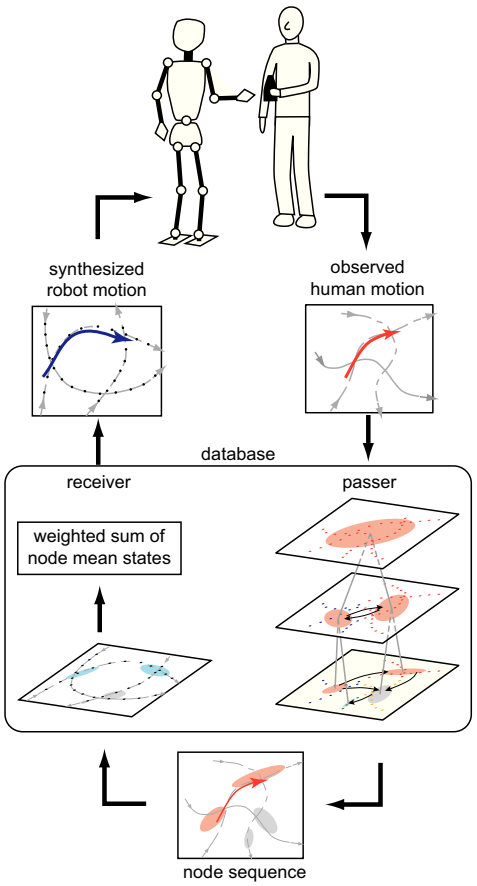

Fig. 5. Motion synthesis algorithm overview. The robot first observes the human motion, which is then sent to the database constructed from the passer's sample states. The search algorithm finds the node sequence that matches the observed human motion. The node sequence is sent to the receiver's database, where the robot motion is synthesized by computing a weighted average of the node mean states of the receiver's database.

a trajectory in the passer's motion space (the thick red line). We use this trajectory to search for the optimal node transition in the passer's database (red ovals in the bottom figure, Section V-B). We can then determine the corresponding node transition in the receiver's database (blue ovals). Once we know the best-matching nodes in the receiver's database, we can interpolate the means of the relevant nodes to obtain synthesized robot motion (thick blue line, Section V-C).

B. Sliding-Window Search

Although using longer sequence of observation generally results in better search precision because we have more information about the observed motion, it can be problematic for applications that require online search because it takes more computation time and the search result will not be updated until the entire observation sequence becomes available. In our application, this fact means that the robot cannot respond to the human action quickly.

A naïve solution for this issue is the sliding window approach, where a fixed length of latest observation is used for search at each frame. We can then consider the last node of the optimal transition at each frame as the current state of the observed motion. Unfortunately, the node transition obtained by appending the last node from every window may not be possible in the node transition graph because the search at each window is performed independently of other windows. This issue often happens when the observed human motion is not close to any of the samples in the database.

Let us illustrate this issue by a simple example shown in Fig. 6 with a database with four nodes N_1 to N_4. Suppose we are searching the database with a window size of 3, and we have observed a trajectory represented by the thick curve.

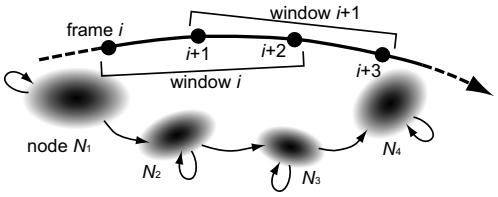

Fig. 6. A case where inconsistency in node transitions may happen.

The four black dots represent the frames we focus on in this example. Because the window size is 3, this example concerns the two windows, indexed as i and $i+1$, whose frames are included in this four-frame period.

At window i, the optimal node transition is likely to be $\{N_1 \rightarrow N_2 \rightarrow N_2\}$ because, although frame $i+2$ itself may be closer to N_3, transition to the same node ($N_2 \rightarrow N_2$) usually has higher transition probability than to a different node ($N_2 \rightarrow N_3$) because most of the neighboring frames have similar states and therefore included in the same node. In addition, the generation likelihood from either N_2 or N_3 is not particularly large and the difference is not enough

for making up the lower transition probability. Similarly, the search result at window $i+1$ would be $\{N_3 \rightarrow N_3 \rightarrow N_4\}$. Given these results, the resulting node transition from frame $i+2$ to $i+3$ would be N_2 to N_4, which is not a valid node transition in the graph.

A workaround would be to use the last two frames of window i, $\{N_2 \rightarrow N_2\}$, as the node transition of the first two frames of window $i+1$, and only search for the last node. Unfortunately, this method also has a problem that a wrong search result at one window cannot be corrected at later windows. In this example, the search result at window $i+1$ would be $\{N_2 \rightarrow N_2 \rightarrow N_3\}$, which is clearly not a good solution given the proximity of frame $i+3$ to N_4.

To solve these issues, we develop a new search algorithm that incorporates the result from the previous window while maintaining the ability to correct the previously found erroneous node transition if necessary. In contrast to the naïve approach, the new algorithm 1) stores a predefined number of best node transitions at each window and 2) attempts to find node transitions that are consistent with the result in the previous window if possible. In Fig. 6, for example, assume that the second best transition was $\{N_1 \rightarrow N_2 \rightarrow N_3\}$. If the search at window $i+1$ considers this transition, it would be possible to obtain the most intuitively correct transition, $\{N_2 \rightarrow N_3 \rightarrow N_4\}$.

The new search algorithm is realized by adding a new term to the cost function (1). At each window, we store the $m(>1)$ best candidate transitions from the search result. Let $Z_{i,k}$ $(k = 1, 2, \ldots, m)$ denote the values of Eq.(1) of the m candidate transitions at window i, and define $\hat{Z}_i = \sum Z_{i,k}$.

During the search at window $i+1$, we add another term in the cost function for the optimization. For each node transition being considered in the optimization process, we compare the first $m-1$ frames of the transition with the last $m-1$ frames of each of the previous candidates. If they do not match the k-th candidate, we add $\hat{Z}_i - Z_{i,k}$ to the cost function. This term takes a larger value if the transition does not match with candidates with smaller cost function value, and therefore biases the search result towards those consistent with the better results in the previous window.

The new cost function for searching the optimal node transition at window $i+1$, Z'_{i+1}, is

$$Z'_{i+1} = Z_{i+1} + w_c \sum_{k=1}^{m} Z_{c,k} \tag{2}$$

where Z_{i+1} is the value of Eq.(1) computed from the node transition under consideration, w_c is a user-defined weight, and $Z_{c,k}$ denotes the cost for considering the node transition consistency with the previous window and takes one of the following values:

$$Z_{c,k} = \begin{cases} 0 & \text{if consistent} \\ \hat{Z}_i - Z_{i,k} & \text{if not consistent} \end{cases} \tag{3}$$

depending on whether the first $m-1$ nodes matches the last $m-1$ nodes in the k-th candidate from window i.

Let us illustrate the new cost function term using the previous example. In the "window i" column of Table I are three possible transitions for window i and their values of the original cost function (1). The "window $i+1$" column lists three possible transitions at window $i+1$ and corresponding values of the additional cost function term $\sum Z_{c,k}$, in the ascending order of the original cost function values (not shown in the table). Transition $\{N_3 \rightarrow N_3 \rightarrow N_4\}$ has the largest cost function value 36 because the first two nodes $(N_3 \rightarrow N_3)$ do not match any of the transitions from window i, and therefore may fall behind the other two transitions that have smaller cost function values (23 and 13). However, it is still possible that it continues to be the optimal transition even though it is not consistent with the results at window i, depending on the cost function values of other transitions.

TABLE I

AN EXAMPLE OF NEW COST FUNCTION VALUE.

window i		window $i+1$	
transition	Z_i	transition	$\sum Z_{c,k}$
$N_1 \rightarrow N_2 \rightarrow N_2$	3	$N_3 \rightarrow N_3 \rightarrow N_4$	$15+13+8=36$
$N_1 \rightarrow N_2 \rightarrow N_3$	5	$N_2 \rightarrow N_3 \rightarrow N_4$	$15+0+8=23$
$N_2 \rightarrow N_2 \rightarrow N_2$	10	$N_2 \rightarrow N_2 \rightarrow N_3$	$0+13+0=13$

C. Robot Motion Synthesis

The database search algorithm described in the previous section returns multiple node transition sequences that match the latest few frames of the observed human motion. The last nodes in these sequences therefore should be close to the state which the passer is currently at. Furthermore, the database now includes the receiver's state distribution corresponding to each node in the passer's database as described in Section IV. These extensions allow us to compute the receiver's pose by a simple interpolation as described below.

Let $\mathcal{N}$ denote the set of the last nodes of the m best node transitions returned by the search algorithm, excluding duplicates. We then use $\boldsymbol{x}_i$ $(i \in \mathcal{N})$ to represent the mean states of the nodes in $\mathcal{N}$. Also let $\hat{\boldsymbol{x}}$ denote the current observed state of the passer (human). We determine the weight w_i for each of the mean state such that the weighted sum of mean states becomes as close as $\hat{\boldsymbol{x}}$ by solving an optimization problem with a quadratic cost function

$$Z_W = \frac{1}{2} || \hat{\boldsymbol{x}} - \sum_{i \in \mathcal{N}} w_i \boldsymbol{x}_i ||^2 \tag{4}$$

and inequality constraints $w_i \geq 0 \, (\forall i \in \mathcal{N})$ using numerical optimization library SNOPT [14]. Finally, we use these weights to compute the weighted sum of the mean states at corresponding nodes in the receiver's database.

VI. RESULTS

We first present detailed analysis of the database properties using pre-recorded data in simulation, followed by results of preliminary hardware implementation of the method. Please also refer to the accompanying video.

TABLE II

STATISTICS OF THE DATABASES USED IN THE EXPERIMENT.

configuration	face-to-face		side-by-side	
# of clips	15		10	
marker set	all	arm	all	arm
# of frames	1686	1686	863	863
# of nodes	255	245	119	109
# of layers	10	10	9	8

A. Database Setup

It would be interesting to investigate whether the database can distinguish subtle differences in the observed motions. We are particularly interested in whether the synthesized robot motions adapt to the type of the object being transported. For this purpose, we prepared three object types: a tape that will be held from the top in handing and from the bottom in receiving, a bottle that will be held from the side in both handing and receiving, and a bag whose handle will be hung to the receiver's hand.

We also tested two different standing configurations: face-to-face and side-by-side. We captured examples with all three objects for face-to-face configuration and with a bottle and a bag for side-by-side. Examples from these configurations are built into two separate databases assuming that the configuration is known in advance.

We captured six instances of human-to-human passing behavior for each object and configuration while the subjects were standing at a distance of their choice. The subjects were instructed to perform the passing motion as naturally as possible. They were also told when the recording started, but were not given any specific timing to start the action.

We then built the database using five of the six clips for each object, and left one clip for the synthesis experiment. As a result, we used 15 clips for the face-to-face database and 10 for the side-by-side database. We manually divided each clip at the time when the hand-off took place and only included the first half in the database. The length of segmented clip ranged from 3.3 s to 4.5 s. The motion data were recorded at 120 fps but were downsampled to 30 fps.

One of the features of the database presented in [1] is that we can use different marker sets for motion capture and clustering. We took advantage of this feature by applying two different marker sets for clustering: the same marker set as motion capture and a marker set that only includes the arm markers. The former assumes that the passer's whole body motion is observable, while the latter considers a more realistic scenario where only the arm pose is observable.

Table II summarizes the statistics of the four databases. The parameters used in the following examples are a window width of 0.1 s, $w_c = 0.1$, and $m = 10$.

B. Search Time

We first demonstrate the scalability of the search algorithm by comparing the search time with different window widths. The computation time will also be affected by the number of layers and the size of the state space. As shown in Fig. 7, the search time increases mostly linearly with respect to the window width. It also remains under the sampling timestep (1/30 s) up to the window width of 6 frames.

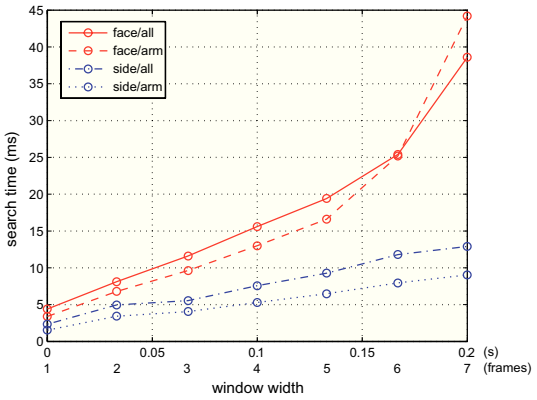

Fig. 7. Search time for the four databases.

TABLE III

MARKER POSITION ERRORS FROM GROUND TRUTH ($\times 10^{-3}$M^2).

object	tape		bottle		bag	
configuration	face-to-face					
database	all	arm	all	arm	all	arm
all markers	2.58	5.85	2.19	4.61	2.12	3.98
arm markers	9.04	9.46	10.3	9.78	5.99	5.37
configuration	side-by-side					
database	all	arm	all	arm	all	arm
all markers	—	—	1.48	1.75	0.252	1.10
arm markers	—	—	12.7	11.6	1.34	5.81

C. Evaluation of Synthesized Motions

We created the input data for the search algorithm by extracting the passer's motion from the clip that has been left out from the database. The receiver's motions in the clips were then used in the evaluation as the ground truth.

Figure 8 shows snapshots from synthesized motions at 0.5 s interval when the human and robot are facing each other. Snapshots for side-by-side configuration are shown in Fig. 9. The poses were computed by an inverse kinematics algorithm directly from the synthesized marker positions using a humanoid robot model whose kinematics parameters are different from the human subject. No clean up, such as smoothing, was applied.

The rightmost column in Figures 8 and 9 shows the closeup images at hand-off. Note that the hand configurations at the end of the sequence correctly reflect the way each object is passed. Also note that the robot (the figure on the left) starts moving before the human reaches the location where passing takes place.

We evaluate the synthesized motions by comparing with the ground truth motions included in the test inputs. Table III shows the error represented by the average squared distance between a synthesized marker of the receiver and the corresponding marker in the ground truth. The average was taken during the last 0.5 s of the synthesized motion because the poses closer to the hand-off are more critical for the task. As

shown in the table, the errors only with the arm pose input are larger than that when the whole body pose is used, but the errors in the arm markers are comparable. This result implies that reduced measurement focusing on the critical part of the body may be enough for specific applications.

D. Hardware Implementation

We implemented the method on a humanoid robot with upper body. The four joints in the left arm (three shoulder joints and the elbow joint) were used for the reaching motion. The hand was fixed at a half-gripped pose so that the user can hang the object with a handle. To measure the user motion, we used a real-time optical motion capture system [15] with eight cameras using a simplified marker set consisting 13 markers in total (marker set $\mathcal{H}$) attached to the trunk, upper arm, lower arm, and hand.

Because we did not have accurate kinematics parameters of the robot, we used a data-based inverse kinematics model. We first measured the positions of 9 markers (marker set $\mathcal{R}$) attached to the robot's trunk, upper arm, and lower arm at various joint angle samples using the same optical motion capture system. We then modeled the joint angles as a linear interpolation of the samples with weights computed by radial basis functions based on the desired marker positions [16].

The database was constructed using the face-to-face samples such that it uses marker set $\mathcal{H}$ for sample clustering and search, and outputs the positions of the markers of $\mathcal{R}$. We therefore do not have to perform inverse kinematics computation for the human. Furthermore, the robot's hand position roughly matches the receiver's despite the difference in the kinematics because we try to match the marker positions rather than joint angles.

Snapshots from hardware experiment are shown in Fig. 10, where the first frame corresponds to the time when the user started moving his arm. We can observe that the robot started moving its arm before the user reached his desired passing location. Figure 11 shows a more interesting interaction where the user teases the robot by extending and retracting his arm a few times before handing the object.

VII. Conclusion

In this paper, we presented a method for synthesizing the receiver's motion in human-to-robot object passing tasks. We applied a data-driven approach under the assumption that human-to-human object passing behavior is repeatable in similar situations. For this purpose, we extended an existing human motion database to our target task that requires different motion spaces for search and synthesis. We also developed a new sliding-window search algorithm that realizes realtime motion search with continuous observation while ensuring that the resulting node transition is consistent with the database. The experiments using real human motion data demonstrated that the planner is capable of synthesizing reaching motion of the robot with appropriate hand position and orientation to receive objects of different grasp types. The reaching motion also starts while the passer's hand is moving towards the location where hand-off takes place.

Several directions remain as future work. We would like to perform tests on databases with wider variety of passing motions including different distances between the passer and receiver. Although the database will be significatly larger, its hierarchical structure allows the search to be performed at higher layers. Searching with coarser nodes would also be useful to prevent the robot from reacting too quickly especially at the earlier stage of the passing motion when the robot is not sure if the human has actually started the passing motion. Sensitivity of the algorithm to observation noise may be an issue if we are to obtain the data without instrumenting the human because the observed human motion is likely to be contaminated by significant noise especially. For hardware experiment, adding finger motions and other secondary behaviors would be important to facilitate the hand-off and make the motion more engaging.

Acknowledgement

M. Revfi was supported by the International Center for Advanced Communication Technologies (interACT).

References

[1] K. Yamane, Y. Yamaguchi, and Y. Nakamura, "Human motion database with a binary tree and node transition graphs," *Autonomous Robots*, vol. 30, no. 1, pp. 87–98, 2011.

[2] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 473–482, 2002.

[3] A. Safonova and J. Hodgins, "Construction and optimal search of interpolated motion graphs," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 106, 2007.

[4] W. Takano, K. Yamane, K. Sugihara, K. Yamamoto, and Y. Nakamura, "Primitive communication based on motion recognition and generation with hierarchical mimesis model," in *Proceedings of IEEE International Conference on Robotics and Automation*, Orlando, FL, May 2006, pp. 3602–3609.

[5] D. Lee and Y. Nakamura, "Mimesis model from partial observations for a humanoid robot," *The International Journal of Robotics Research*, vol. 29, no. 1, pp. 60–80, 2010.

[6] D. Lee, C. Ott, and Y. Nakamura, "Mimetic communication model with compliant physical contact in human-humanoid interaction," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1684–1704, 2010.

[7] K. Harada, S. Kajita, F. Kanehiro, K. Fujiwara, K. Kaneko, and K. Yokoi, "Real-time planning of humanoid robot's gait for force-controlled manipulation," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 1, pp. 53–62, 2007.

[8] M. Lawitzky, A. Mörtl, and S. Hirche, "Load sharing in human-robot cooperative manipulation," in *Proceedings of IEEE International Symposium on Robot and Human Interactive Communication*, 2010, pp. 185–191.

[9] A. Thobbi, Y. Gu, and W. Sheng, "Using human motion estimation for human-robot cooperative manipulation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2873–2878.

[10] E. Sisbot and R. Alami, "A human-aware manipulation planner," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1045–1057, 2012.

[11] A. Edsinger and C. Kemp, "Human-robot interaction for cooperative manipulation: handing objects to one another," in *Proceedings of IEEE International Symposium on Robot and Human Interactive Communication*, 2007, pp. 1167–1172.

[12] V. Micelli, K. Strabala, and S. Srinivasa, "Perception and control challenges for effective human-robot handoffs," in *RGB-D Workshop, Robotics: Science and Systems Conference*, 2011.

[13] A. Shukla and A. Billard, "Coupled dynamical system based hand-arm grasp planning under real-time perturbations," in *Proceedings of Robotics: Science and Systems*, 2011.

[14] P. Gill, W. Murray, and M. Saunders, *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*. http://www.cam.ucsd.edu/ peg/papers/sndoc7.pdf, 2006.

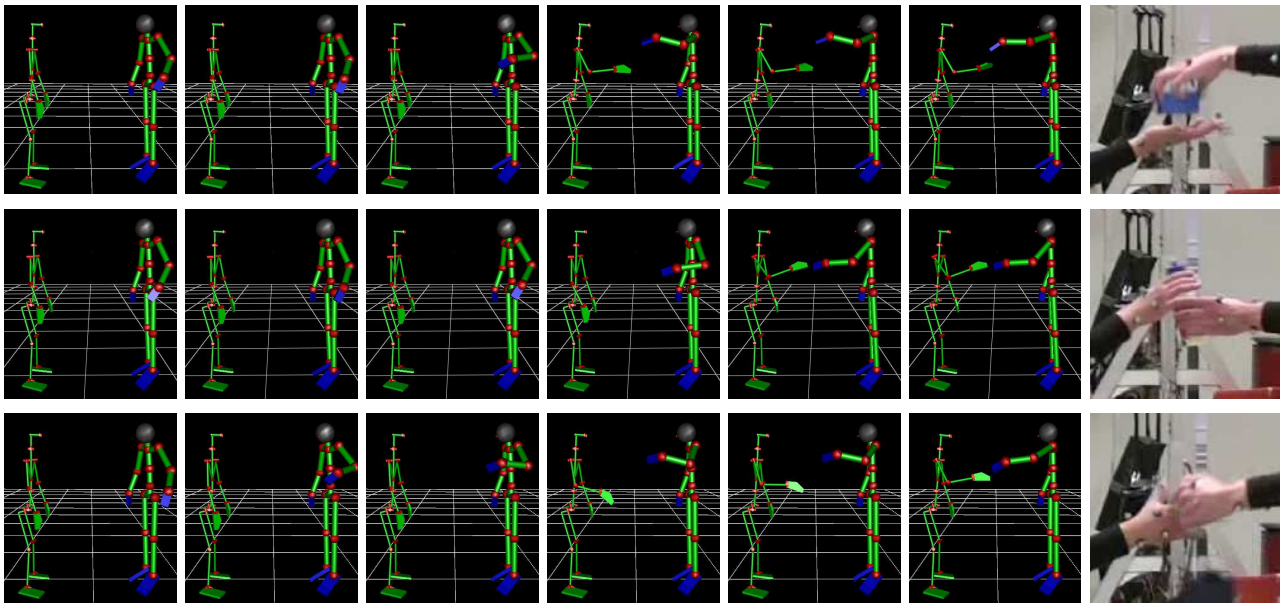

Fig. 8. Snapshots taken at 0.5 s interval from synthesized human-to-robot passing motion for various objects when the human and robot are facing each other. The object, not shown in the images, is passed from the right figure (human) to the left (robot). From the top row: tape, bottle, and bag.

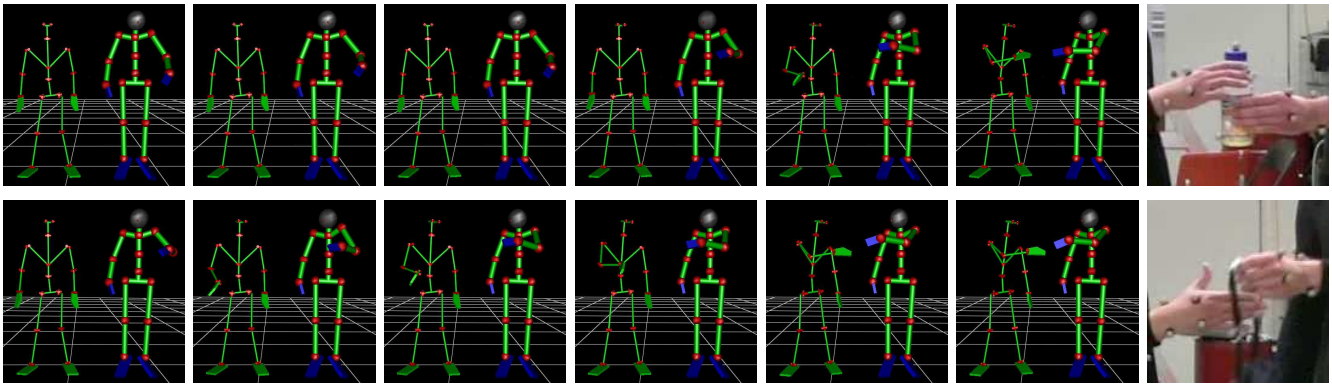

Fig. 9. Snapshots taken at 0.5 s interval from synthesized human-to-robot passing motion for various objects when the human and robot are standing side-by-side. The object, not shown in the images, is passed from the right figure (human) to the left (robot). From the top row: bottle and bag.

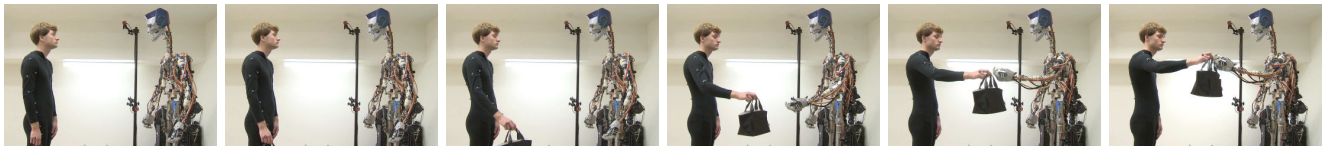

Fig. 10. Snapshots taken at 0.5 s interval from hardware experiment.

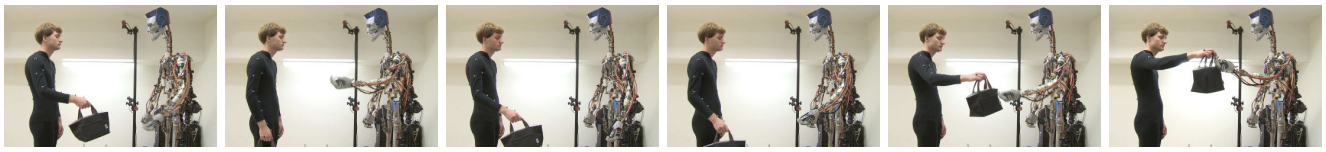

Fig. 11. Example of more interesting interaction where the user teases the robot.

[15] Natural Point, Inc., "OptiTrack," http://www.naturalpoint.com/optitrack/.

[16] C. Rose, P.-P. Sloan, and M. Cohen, "Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation," *Eurographics*, vol. 20, no. 3, 2001.