How-Models of Human Reaching Movements in the Context of Everyday Manipulation Activities

Daniel Nyga, Moritz Tenorth and Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{nyga, tenorth, beetz}@cs.tum.edu

***Abstract*— We present a system for learning models of human reaching trajectories in the context of everyday manipulation activities. Different kinds of trajectories are automatically discovered, and each of them is described by its semantic context. In a first step, the system clusters trajectories in observations of human everyday activities based on their shapes, and then learns the relation between these trajectories and the contexts in which they are used. The resulting models can be used for robots to select a trajectory to use in a given context. They can also serve as powerful prediction models for human motions to improve human-robot interaction. Experiments on the TUM kitchen data set show that the method is capable of discovering meaningful clusters in real-world observations of everyday activities like setting a table.**

I. Introduction

Humans performing everyday manipulation activities such as setting the table or cleaning up exhibit motion – in particular reaching behavior – that is both stereotypical [1] and optimized [2], [3]. Using stereotypical movement patterns rather than planning each individual manipulation task anew has a number of advantages, many of them are implied by the reduction of motion nondeterminism. Thus, stereotypical reaching patterns improve the predictability of movements both by the robot itself and for external observers. In addition, the monitoring, diagnosis, planning and learning of stereotypical movements is faster and can have better results.

Today's manipulation robots do not yet exploit the power of stereotypicality as a resource for better motion planning and control. In our research, we aim at building and investigating models of human manipulation activities and transfer these models in order to realize better autonomous robot control systems. In this respect, we distinguish between *how*- and *why*-models of human manipulation behavior. The *how*-models enable us to compactly describe and predict the manipulation behavior, while the *why*-models look into optimization and other criteria that might cause the humans to select the respective behavior.

In this paper we investigate *how*-models for reaching actions in the context of unconstrained human everyday activities such as setting the table. Figure 1 depicts the trajectories of the left and right hand of four subjects and 20 table-setting episodes. The image on the lower left shows the sub-sequences of the trajectories which correspond to the reaching movements in these scenarios. The picture on the lower right depicts the clustered trajectories for which we compute mean trajectories as their representatives.

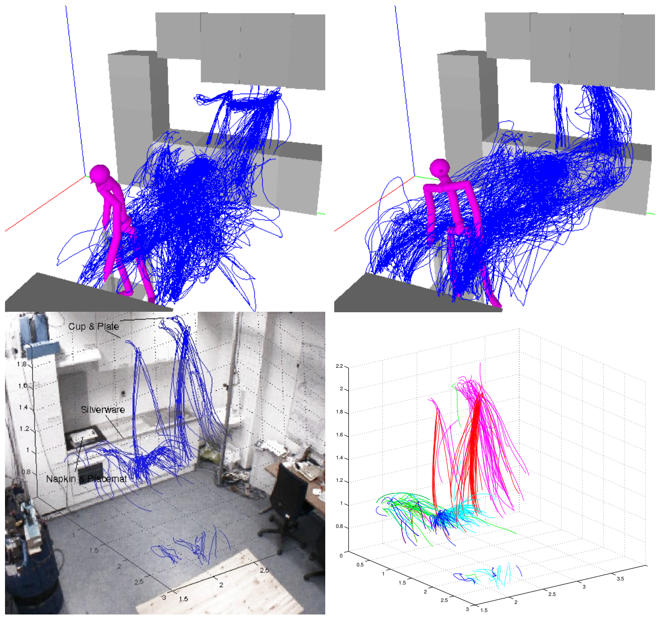


Fig. 1. *Top:* Trajectories of the left and right hand in 20 table-setting episodes. *Bottom left:* Reaching trajectories, drawn on top of an image of the experiment kitchen, and original positions of the manipulated objects. The image is only an approximate visualization without perfect alignment with the trajectories. *Bottom right:* Clusters found in this set of reaching trajectories.

The main contributions of this paper are (1) a clustering algorithm that reliably identifies clusters of trajectories in real-world observations of human activities, and (2) methods for learning semantic descriptions of the clusters which allow to select the appropriate trajectory in a given context. In the remainder of the paper, we first give an overview over the system components, introduce the trajectory representation, the clustering algorithm, and the techniques for semantically characterizing trajectory clusters. We present results of our evaluation on motion tracking data and finish with a discussion of related work and our conclusions.

II. System Overview

The proposed system consists of two main components: A trajectory clustering module, and models that describe the semantics of those clusters. For learning the semantic models, we employ the GrAM (Grounded Action Models) framework described in [4]. GrAMs have been applied to the analysis of soccer games and, in [5], for learning places used in manipulation actions; here, we learn trajectories instead. The methods described here are embedded in the

AM-EvA system [6] that combines data mining with knowledge representation for modeling and reasoning on human activities on multiple levels, ranging from the high-level activity context to intermediate levels like actions down to single motions. AM-EvA provides pre-segmented trajectory data for motions like *Reaching* or *LoweringAnObject* which are used as the input to this system. The trajectory clustering algorithm (Section IV) is implemented in MATLAB and called from AM-EvA's Prolog-based infrastructure via the Prolog-MATLAB interface *PLML* [7].

III. Trajectory representation

We define a set of trajectories, e.g. the set of all observed reaching motions, as $T = \{\tau_i\}$, and describe each trajectory $\tau_i \in T$ by an ordered sequence of m points π_i^k

$$\tau_i = (\pi_i^1, \dots, \pi_i^m), \;\; \pi_i^k \in \mathbb{R}^3 \tag{1}$$

The choice of a suitable coordinate representation is important in any information extraction application: In a poor representation, the aspects of interest may be hidden by less important, though strong, variations in the data. For example, the naive choice of an environment-global coordinate system has the drawback that the strongest variation in the data is usually the position of the subject in the environment, while more interesting aspects like the hand motions are hidden. We chose the following trajectory representation that abstracts away from the global position in space while maintaining important information like the concepts of up-/downward motions and the shape of the trajectories. For clustering, trajectories are described by the component-wise stepwise relative progression

$$\delta_i = (\delta_i^1 = \pi_i^2 - \pi_i^1, \; \dots \;, \delta_i^{m-1} = \pi_i^m - \pi_i^{m-1}) \tag{2}$$

which can be interpreted as a sequence of vectors, each pointing from a point π_i^{k-1} to the subsequent point π_i^k $(2 \leq k \leq m)$. Figure 2 illustrates these two representations. The diagram on the left shows a set of trajectories T in global coordinates as in (1). The right diagram visualizes the local coordinates of the corresponding δ_i in form of a "fir branch". The bold black line (the "branch") is the cluster centroid in global coordinates, whereas the "needles" are the vectors δ_i^k. This kind of visualization reveals interesting properties like the variance in different parts of the trajectory, the existence of clusters at different positions, or the cluster cohesion.

A. Distance metric

An appropriate distance metric is crucial for extracting meaningful clusters. For comparing trajectories, we first resample them to a fixed length, compute the component-wise distances, and combine the single distances to a common value. The less frayed the fir branch visualization looks (see Figure 2), the more cohesive the set of trajectories, that is, the more similar the orientations of the vectors δ_i^k. Formally, this can be described by the angles between the δ_i^k and δ_j^k vectors of two trajectories τ_i and τ_j at a particular component

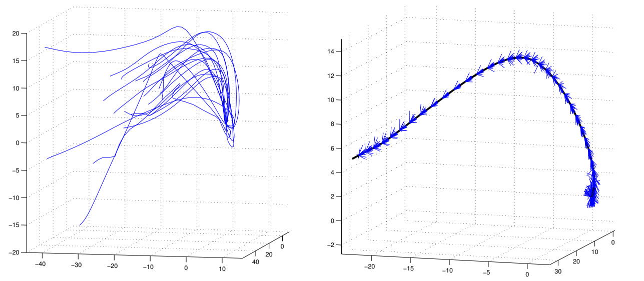


Fig. 2. *Left:* Trajectories in global coordinates given by π_i^k. *Right:* A "fir branch" visualization of the cluster centroid of the same set of trajectories in global coordinates, with the local coordinate vectors δ_i^k as "needles" with lengths normalized to 1. The high variance in the beginning and in the end, and the low variance in between, is clearly visible.

k which can be computed as:

$$\cos\varphi = \frac{\mathbf{x}^T\mathbf{y}}{|\mathbf{x}| \cdot |\mathbf{y}|} \Leftrightarrow \varphi = \cos^{-1}\frac{\mathbf{x}^T\mathbf{y}}{|\mathbf{x}| \cdot |\mathbf{y}|} \in [0, \pi] \tag{3}$$

We obtain the distance function d_v for two vectors $\mathbf{x}$ and $\mathbf{y}$ by normalizing the angle φ to the interval $[0, 1]$.

$$d_v(\mathbf{x}, \mathbf{y}) = \frac{\varphi}{\pi} = \frac{\cos^{-1}\frac{\mathbf{x}^T\mathbf{y}}{|\mathbf{x}|\cdot|\mathbf{y}|}}{\pi} \tag{4}$$

The overall distance between a pair of trajectories τ_1 and τ_2 is defined as the averaged sum of the vector distances for each component

$$d_\tau(\tau_1, \tau_2) := d_\delta(\delta_1, \delta_2) = \frac{1}{m-1}\sum_{k=1}^{m-1} d_v(\delta_1^k, \delta_2^k), \tag{5}$$

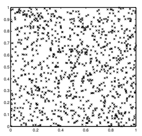
Note that, due to the averaging, the domain of values is still $d_\tau \in [0, 1]$. In some cases, the relative length difference of the vector pairs δ_1^k and δ_2^k can provide important information, which can be taken into account by extending $d_\delta(\delta_1, \delta_2)$ to

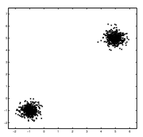
$$d_\delta(\delta_1, \delta_2) = \frac{1}{2(m-1)}\sum_{k=1}^{m-1}\left(d_v(\delta_1^k, \delta_2^k) + 1 - \frac{\min(|\delta_1^k|, |\delta_2^k|)}{\max(|\delta_1^k|, |\delta_2^k|)}\right) \tag{6}$$

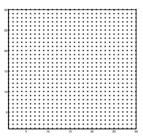
IV. Clustering procedure

The problem of clustering trajectories is characterized by high-dimensional data and an unknown number of clusters. This is challenging for common clustering algorithms: The k-Means algorithm [8] can handle an unknown number of clusters only via costly cross-validation. Other techniques like EM (Expectation/Maximization) or SAHN (Sequential, agglomerative, hierarchical, non-overlapping) clustering inherently handle an unknown number of clusters, but are computationally too expensive for high-dimensional data.

In this paper, we propose a multi-step hierarchical clustering algorithm motivated by *CART* (Classification and Regression Trees) [9]. CART is a classification method to create binary decision trees by iteratively partitioning a data set into two cuboid sub-regions, each time taking only a few dimensions into account that allow to distinguish the two sub-sets with high accuracy. This *splitting rule* is typically given by a simple threshold (also called *decision stump*, a decision tree with only one node). The process is repeated until a stopping criterion is reached.


(a) $h \approx 0.514$


(b) $h \approx 0.973$


(c) $h \approx 0.121$

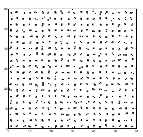

(d) $h \approx 0.998$

Fig. 3. Four exemplary data sets and the resulting Hopkins indices ($n = 1000$ and $m = 10$): (a) Randomly distributed data (b) Strong cluster structure, sampled from two Gaussian distributions (c) Periodically distributed data (d) "Small clusters problem" of the classical Hopkins index

CARTs are appealing due to their simplicity and the fact that the resulting models are intuitive to humans. Additionally, it is a non-parametric and non-linear method, such that only little a-priori knowledge about the data is required, which makes CART suitable for data mining tasks.

If there is a good method for selecting the components to use in the splitting rule, CARTs perform very well on high-dimensional data. For classification problems, this choice is usually made using entropy-driven methods, selecting that feature from which the largest information gain in distinguishing the classes in the data set can be expected. In our unsupervised setting, however, other techniques are needed to select the components with the strongest cluster structure.

A. *Extended Hopkins index*

The Hopkins index [10] indicates if clusters exist in a data set and is used here to select the dimensions to be used for splitting the data. It is computed by uniformly sampling two sets of $m << n$ points: a set $S = \{s_1, \ldots, s_m\} \subset X$ from the dataset $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$, and another set $R = \{r_1, \ldots, r_m\}$ from the convex hull around the data. The algorithm then computes, for each point in these data sets R and S, the distance to the nearest neighbor in X:

$$d_R = \{d_{r_i} \mid \min_j \|r_i - x_j\|, 1 \leq j \leq n\},\ \ 1 \leq i \leq m \quad (7)$$

$$d_S = \{d_{s_i} \mid \min_j \|s_i - x_j\|, 1 \leq j \leq n\},\ \ 1 \leq i \leq m \quad (8)$$

The ratio of these distances yields the Hopkins index $h \in [0, 1]$ that describes how the points inside the data set are distributed with respect to arbitrary points in the convex hull.

$$h(X) = \frac{\sum_{i=1}^m d_{r_i}^p}{\sum_{i=1}^m d_{r_i}^p + \sum_{i=1}^m d_{s_i}^p} \quad (9)$$

If there is no significant cluster structure, the distances inside the data set do not differ much from those between random other points in the convex hull (Figure 3(a)). The resulting Hopkins index is thus near to $h \approx 0.5$. A very small Hopkins index $h \approx 0$ results from data that exhibits a regular structure (Figure 3(c)). A Hopkins index that is $h \approx 1$ is a sign that significant clusters are present in the data. Intuitively, a large Hopkins index means that the distances from arbitrary points in the convex hull to the nearest data point are large, while most points in the data set have a neighbor nearby. Since the Hopkins index highly depends on the sets R and S, it is recommended to sample several different sets R and S and average over the results.

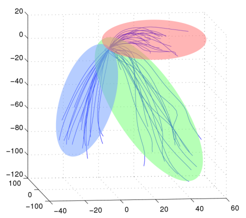
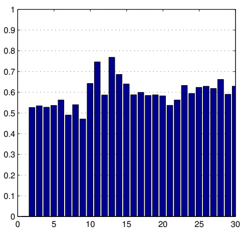



Fig. 4. *Left*: A set of trajectories showing three clusters. The trajectories are resampled using 30 samples, transformed into the local coordinate frame, and shifted to identiacl end points. *Right*: Hopkins indices for each of the 30 trajectory components. The clusters in this example can be extracted using only the components 10-15 with Hopkins-Indices significantly above 0.5.

Our experiments have shown that the classical Hopkins index yields misleading results when being applied to data sets with many small clusters, i.e. clusters containing only few data points. An example of such a data set, consisting of many clusters of only two points, is shown in Figure 3(d). Though the Hopkins index of $h \approx 0.998$ is very high, we would not describe the data as highly clustered.

The problem arises from the fact that that all points have one very close neighbor, while the overall distances d_R are rather large. We therefore extend the classical Hopkins index by taking not only the nearest, but the k nearest neighbors of points r_i and s_i into account. For $k = 1$, we obtain the classical Hopkins index. Experiments have shown that defining $k = m$ often yields good results.

B. *Clustering based on the Hopkins index*

The Hopkins index serves as the criterion to select the components used for clustering: The bigger its value, the stronger the cluster structure we expect to find. We thus compute the Hopkins index for each of the $k = 1, \ldots, m-1$ components of a set of trajectories $T = \{\tau_i\}$ as

$$h_T = (h(\{\delta_i^1\}), \ldots, h(\{\delta_i^{m-1}\})),\ \ 1 \leq i \leq n \quad (10)$$

where n is the number of trajectories in T, m is the number of points per trajectory in T, and δ_i^k is the stepwise relative progression of the i-th trajectory from point k to point $k+1$ as defined above.

Figure 4 shows an example set of trajectories with three clusters (left) and the distribution of the Hopkins indices for the 30 trajectory components (right). Most of the indices are close to 0.5, so no strong clusters are expected in these components. Only the components 10 to 15 have Hopkins indices significantly above 0.5 and are thus expected to show a significant cluster structure. In our experiments, using the five highest Hopkins indices proved sufficient to obtain good clustering results. At each node of our clustering tree, we perform k-means clustering with $k = 2$ and the metric (5) on these five components to obtain a dichotomization of the data. To initialize the cluster centroids for k-means, we sample 10% of the data and perform a pre-clustering with randomly initialized cluster centroids.

C. *Stopping Criterion*

The proposed clustering algorithm requires a criterion to decide when to stop adding nodes to the cluster tree. We

```
Algorithm: CLUSTERTRAJ

Input:   $T = \{\tau_i\},\ i = 1, \ldots, n$, a set of trajectories
         $thr$, a threshold for cluster cohesion
         $d$, the number of points to use for k-means
Output:  $t$, a clustering tree
Static:  $no\_clusters$, cluster counter (initialized to 0)
1) $T \leftarrow$ removeOutliers($T$)
2) $h_T \leftarrow$ computeHopkins($T$)
3) $c_T \leftarrow$ computeCohesion($T$)
4) $cent \leftarrow$ computeCentroid($T$)
5) if $c_T < thr \mid\mid \forall h_i \in h_T : h_i < 0.5$
       $t.dim = null$
       $t.left = null$
       $t.right = null$
       $t.centroid = cent$
       $t.cluster\_id = no\_clusters + 1$
       $no\_clusters = no\_clusters + 1$
       return $t$
6) $dim \leftarrow \{d$ dimensions with highest Hopkins index$\}$
7) $T_{tmp} \leftarrow \{\{\pi^i_k\} \mid i \in dim, k = 1...n\}$
8) $(T_1, T_2) \leftarrow$ k-means($T_{tmp}, 2, d_\tau$)
   $t.cent = cent$
   $t.dim = dim$
   $t.left =$ CLUSTERTRAJ($T_1, thr, d$)
   $t.right =$ CLUSTERTRAJ($T_2, thr, d$)
   $t.cluster\_id = -1$
   return $t$
```

Fig. 5. Algorithm for clustering a set of trajectories.

```
Algorithm: CLASSIFYTRAJ

Input:   $t$, a clustering tree
         $\tau = (\pi^1, ..., \pi^l)$, a trajectory of length $l$
Output:  $id$, the cluster membership of $\tau$
1) if $t.dim$ is not empty
       $\tau_{left} \leftarrow \{\{\pi^i\} \mid i \in t.dim, \pi^i \int .left.cent\}$
       $\tau_{right} \leftarrow \{\{\pi^i\} \mid i \in t.dim, \pi^i \int .right.cent\}$
       $\tau_{traj} \leftarrow \{\{\pi^i\} \mid i \in t.dim, \pi^i \in \tau\}$
       if $d_\tau(\tau_{left}, \tau_{traj}) < d_\tau(\tau_{right}, \tau_{traj})$
           $id \leftarrow$ CLASSIFYTRAJ($t.left, \tau$)
       else
           $id \leftarrow$ CLASSIFYTRAJ($t.right, \tau$)
2) else
       return $id = t.cluster\_id$
```

Fig. 6. Algorithm for assigning an unseen trajectory to a cluster given a clustering tree.

define the *cohesion* within a trajectory cluster as the mean value of the average distances of each member to each other member:

$$c(T) = \frac{1}{n^2} \sum_{(\tau_i,\tau_j) \in T \times T} d_\tau(\tau_i, \tau_j), \tag{11}$$

where $T = \{\tau_1, \ldots, \tau_n\}$ denotes the set of trajectories within the cluster. The clustering stops if the cohesion does not improve significantly any more. Since the distances between trajectories (Eq. 5), and thus also the cohesion values, are in the interval $[0, 1]$, a constant threshold can be used; we empirically chose $c_{thr}(T) \approx 0.15$.

There are other methods for assessing the similarity of data points within a cluster, most of which are based on variance or entropy calculations. However, a threshold based on the sum-of-cosines distance measure performed better in our experiments. Additionally, the matrix of distances $d_\tau(\tau_i, \tau_j)$ resulting from (11) can be reused for detecting outliers, which makes it computationally attractive.

D. Outlier Detection

Outliers in the set of trajectories can result from errors of the tracking system and should be recognized and filtered to prevent them from distorting the rest of the data. We regard a trajectory x_k as an outlier iff at least one component x_k^i of the vector deviates more than twice the standard deviation σ^i from the mean μ^i, i.e.

$$outlier(x_k) \Leftrightarrow \exists i \in \{1, \ldots, p\}. \left| \frac{x_k^i - \mu^i}{\sigma^i} \right| > 2 \tag{12}$$

Here, the mean distances within a cluster computed by (11) are used. If an outlier is being detected, the whole trajectory is removed from the data set without substitution.

E. Clustering Algorithm

Figure 5 outlines the complete algorithm for clustering trajectories. The algorithm takes a set of trajectories T (Eq. (1)), a threshold thr for the cohesion within a cluster (Eq. (11)), and the number of components d to use for clustering as parameters. After outlier removal, the cohesion and Hopkins indices as in (11) and (10) are computed, as well as the centroid of the set T, given by the arithmetic mean of all trajectory coordinates [steps 1) to 4)]. If the abort criteria (cohesion below the threshold or all Hopkins indices lower than 0.5, meaning that we cannot expect any clusters) evaluate to $true$, the algorithm returns a leaf node of the clustering tree with a cluster centroid and a cluster ID given by a numeric value and both children and dimensions set to $null$. Otherwise, the algorithm selects the d dimensions with the largest Hopkins indices from h_T and performs k-means clustering with $k = 2$ on these dimensions using the distance measure d_τ (5). The result is a partition in two clusters $T_1 \cup T_2 = T$, $T_1 \cap T_2 = \emptyset$ on which CLUSTERTRAJ is recursively applied, while making the resulting trees children of the current node.

Assigning new, unseen trajectories to a cluster can be done with the CLASSIFYTRAJ algorithm (Figure 6). It uses the tree obtained from CLUSTERTRAJ as a classification/decision tree, traverses it from the root to a leaf, and checks the trajectory distances based on the d Hopkins dimensions at each node.

V. SEMANTIC ACTION MODEL DEFINITION

Grounded Action Models (GrAMs) as introduced in [4] are intensional specifications of a learning problem. They list a set of features (called *observables*) which can serve for predicting a certain property (called *predictable*). Given a training set of observed action instances of a class like *Reaching*, the system learns the relation between their observable properties and the one to be predicted. In our case, the observable properties describe things like the manipulated object or the hand that is used, while the shape of the trajectory (i.e. the cluster ID) is to be inferred.

For example, the intensional model *reachTrajModel* is defined as an instance of an *ActionModel* that is to be learned from the training set *reachTraj* (all trajectories of *type Reaching*) with the observables *objectActedOn* and *bodyPartsUsed*:

```
owl_assert(reachTrajModel,   type,         'ActionModel')
owl_assert(reachTrajModel,   forAction,    reachTraj)

owl_assert(reachTraj,        type,         'Restriction')
owl_assert(reachTraj,        onProperty,   type)
owl_assert(reachTraj,        hasValue,     'Reaching')

owl_assert(reachTrajModel,   observable,   objectActedOn)
owl_assert(reachTrajModel,   observable,   bodyPartsUsed)

owl_assert(reachTrajModel,   predictable,  trajectory-Mean)
```

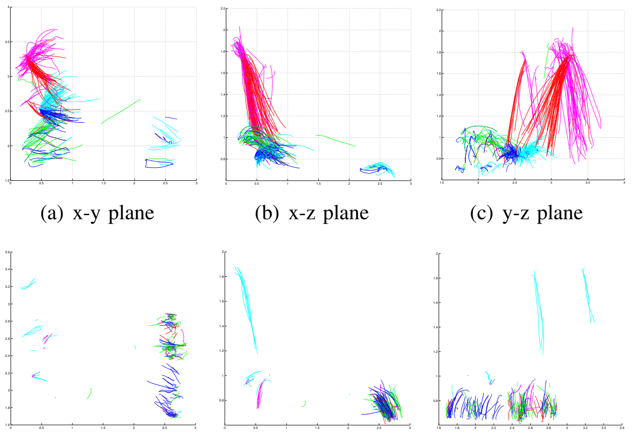
The action model is learned as a classifier on top of the cluster assignment that uses the values of the *observable* features to learn rules that explain the values of the *predictable* features. These rules form the *extensional* action model and semantically describe in which context a trajectory cluster is being used. They are equivalent to a sub-class definition in Description Logic, like for instance the class of "*Reaching* actions performed with the right hand towards a cup inside the cupboard". These sub-class definitions are learned autonomously from data as those rules that best explain the relations in the training set. Table I gives an example of a learned extensional model.

Action models can either be used for classification (inferring the context given an observed trajectory), or for selecting a trajectory given a certain context. An application in robotics is the selection of an *appropriate* reaching trajectory in the respective action context.

VI. Evaluation

We evaluate the system on the TUM Kitchen Data Set [11] which provides motion tracking and other sensory data of 20 table setting episodes performed by four human subjects. All sequences are labeled, and we use these labels to select trajectory segments (like *Reaching*) as input data for the clustering algorithm. We use the trajectories of the left and right hand as visualized in Figure 1.

The trajectories are first re-sampled to a length of 100 points using spline interpolation. Since the provided labels often do not exactly match the beginning and end of a trajectory segment, we cut off 25% of the frames at the beginning and end of the trajectories, namely the first 16% and the last 9%. The trajectories are then transformed into a person-intrinsic coordinate frame defined by the left and right shoulder *SBL* and *SBR*. This transformation only affects the x and y coordinates, the z coordinate stays unaffected. In the resulting representation, person-related spatial relations like "in front of", "left of" or "above" can be described more easily. Since the human body is a deformable system, there is no optimal person-intrinsic coordinate frame, but as Figure 4 (left) shows, the shoulder-centric coordinates yield reasonable results. Finally, the ending points of the trajectories, corresponding to the object position, are aligned.


(a) x-y plane (b) x-z plane (c) y-z plane

(d) x-y plane (e) x-z plane (f) y-z plane

Fig. 7. Cluster assignments, indicated by the trajectory color, for *Reaching* trajectories (upper row) and trajectories for *LoweringAnObject* (lower row).

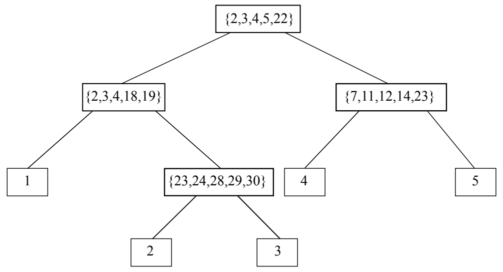


Fig. 8. Clustering tree for reaching trajectories. The sets of the internal nodes correspond to the Hopkins dimensions or *dim* sets used in the ClusterTraj algorithm. The numbers at the leaf nodes denote the resulting cluster IDs.

A. Cluster Assignments

Figure 1 (bottom right) and Figure 7 show the trajectory clusters identified by ClusterTraj for the motions *Reaching* and *LoweringAnObject*. These results show that the algorithm is able to distinguish different forms of trajectories even if they are in similar regions of the environment, like the upwards motions in clusters 1 and 4. The trajectories for putting down objects are not as well separated as those for reaching, but their shapes are also well distinguished. The blue cluster, for instance, is mainly directed to the left, while the green one points to the right.

The clustering tree for the *Reaching* trajectories with five leaf nodes – corresponding to the cluster IDs – and the four internal nodes is visualized in Figure 8. The components which have been identified by Hopkins analysis and which are used for clustering are listed in the internal nodes. Interestingly, all of these points $\{2, 3, 4, 5, 7, 11, 12, 14, 18, 19, 23, 24, 28, 29, 30\}$ are part of the first third of a trajectory, indicating that its shape is already planned and fixed at an early stage.

B. Learned Semantic Action Models

Table I shows the associations between observable properties and the cluster IDs that are learned based on the specification of the intensional model. The system is able to distinguish trajectories with different meaning, though they have similar shapes and are in similar regions of the

bodyPartsUsed	objectActedOn	Cluster Assignment
LeftHand	PlaceMat	3
	Cup	4
	DinnerPlate	4
	Napkin	3
	SilverwarePiece	2
	Drawer	3
	Cupboard	1
RightHand	PlaceMat	3
	Cup	4
	DinnerPlate	4
	Napkin	3
	SilverwarePiece	5
	Drawer	3
	Cupboard	1

TABLE I

EXTENSIONAL ACTION MODEL FOR *Reaching* MOTIONS

environment like reaching for a cupboard handle (Cluster 4) and taking an object out of that cupboard (Cluster 1). For some objects, our algorithm also found differences in the reaching behavior of the left and right hand, e.g. for *SilverwarePiece* (Cluster 2/5 resp.). Some objects at approximately the same position, such as *Cup* and *DinnerPlate*, or *Napkin* and *PlaceMat*, show reaching trajectories that are too similar for the algorithm to be discernible. Since most of the objects are always picked up with the same hand, there is little variation in clusters for the *bodyPartsUsed* property.

VII. RELATED WORK

Our work can be seen as a kind of imitation learning [12]: Robots observe human actions, analyze and abstract the observed data, and use them to imitate the actions. An overview of the research in this area can be found in [13]. Recent approaches learn motions for instance as differential equations [14] or Gaussian Mixture Models [15]. These methods assume that the motions are explicitly demonstrated, i.e. that everything that has been observed is to be imitated. In contrast, we approach the problems of deciding *what* to imitate and of learning in the task context: The robot observes complete activities like setting a table and extracts models of single motions from this data. This requires methods to identify distinct trajectory clusters and to select a suitable one based on its semantics.

An approach for clustering human reaching trajectories using point distribution models (PDM [16]) has been presented by Stulp [17]. PDMs are learned by first performing a Principal Component Analysis (PCA) and then clustering the data using the Mahalanobis distance. However, their experiments use very clean, pre-segmented trajectories. Further, both [16] and [17] assume that high variance implies the existence of clusters, though the subspace obtained using PCA does not necessarily coincide with the subspace spanned by the cluster centers [18], especially in real-world clustering problems without well-separated clusters.

VIII. CONCLUSIONS

We presented our system for clustering and semantically annotating trajectories observed in human manipulation activities. The system learns models of human motions in the context of complete activities and is able to robustly cluster noisy trajectory data obtained from real-world observations. Semantic characterizations of the clusters are learned to select an appropriate motion in a given situation.

Clustering similar motions is an important step before learning low-dimensional trajectory representations in order to eliminate ambiguities introduced by the different ways an action can be performed. A cluster that contains only trajectories of a kind is much better suited for learning e.g. parameterizable function representations. Applications of our models include robots imitating the trajectory a human would use in a given situation, but also the prediction of human motions, e.g. for human-robot interaction or motion tracking.

IX. ACKNOWLEDGMENTS

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*, see also `www.cotesys.org`.

REFERENCES

[1] C. Atkeson and J. Hollerbach, "Kinematic features of unrestrained vertical arm movements," *J. Neurosci.*, vol. 5, no. 9, pp. 2318–2330, 1985.

[2] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz., "Optimizing principles underlying the shape of trajectories in goal oriented locomotion for humans." in *Proceedings of the International Conference on Humanoid Robots*, 2006.

[3] E. Todorov and M. Jordan, "Optimal feedback control as a theory of motor coordination," *Nature neuroscience*, vol. 5, no. 11, pp. 1226–1235, 2002.

[4] N. v. Hoyningen-Huene, B. Kirchlechner, and M. Beetz, "GrAM: Reasoning with grounded action models by combining knowledge representation and data mining," in *Towards Affordance-based Robot Control*, 2007.

[5] M. Tenorth and M. Beetz, "KnowRob — Knowledge Processing for Autonomous Personal Robots," in *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2009.

[6] M. Beetz, M. Tenorth, D. Jain, and J. Bandouch, "Towards Automated Models of Activities of Daily Life," *Technology and Disability*, vol. 22, 2010.

[7] Samer Abdallah, "PLML – A Prolog-Matlab interface," 2006, http://www.swi-prolog.org/contrib/SamerAbdallah/index.html.

[8] J. Hartigan and M. Wong, "A k-means clustering algorithm," *JR Stat. Soc. Ser. C-Appl. Stat*, vol. 28, pp. 100–108, 1979.

[9] L. Breiman, *Classification and Regression Trees*. Boca Raton, Florida: Chapman & Hall/CRC, 1984.

[10] B. Hopkins and J. Skellam, "A new method for determining the type of distribution of plant individuals," *Annals of Botany*, vol. 18, no. 2, p. 213, 1954.

[11] M. Tenorth, J. Bandouch, and M. Beetz, "The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition," in *IEEE Int. Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS)*, 2009.

[12] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[13] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Springer Handbook of Robotics*. Springer, 2008, ch. 59. Robot programming by demonstration.

[14] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *International Conference on Robotics and Automation*, 2009.

[15] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[16] P. Roduit, A. Martinoli, and J. Jacot, "A quantitative method for comparing trajectories of mobile robots using point distribution models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 2441–2448.

[17] F. Stulp, E. Oztop, P. Pastor, M. Beetz, and S. Schaal, "Compact models of motor primitive variations for predictable reaching and obstacle avoidance," in *9th IEEE-RAS International Conference on Humanoid Robots*, 2009.

[18] C. Ding, X. He, H. Zha, and H. D. Simon, "Adaptive dimension reduction for clustering high dimensional data," *IEEE International Conference on Data Mining*, vol. 0, p. 147, 2002.