Learning the Spatial Semantics of Manipulation Actions through Preposition Grounding

Konstantinos Zampogiannis[1], Yezhou Yang[1], Cornelia Fermüller[2], and Yiannis Aloimonos[1]

Abstract— **In this paper, we introduce an abstract representation for manipulation actions that is based on the evolution of the spatial relations between involved objects. Object tracking in RGBD streams enables straightforward and intuitive ways to model spatial relations in 3D space. Reasoning in 3D overcomes many of the limitations of similar previous approaches, while providing significant flexibility in the desired level of abstraction. At each frame of a manipulation video, we evaluate a number of spatial predicates for all object pairs and treat the resulting set of sequences (Predicate Vector Sequences, PVS) as an action descriptor. As part of our representation, we introduce a symmetric, time-normalized pairwise distance measure that relies on finding an optimal object correspondence between two actions. We experimentally evaluate the method on the classification of various manipulation actions in video, performed at different speeds and timings and involving different objects. The results demonstrate that the proposed representation is remarkably descriptive of the high-level manipulation semantics.**

I. Introduction

Intelligent robots built for manipulation tasks need to learn how to manipulate. However, given that there is an infinite number of ways to perform a certain manipulation action, such as for example, making a peanut butter and jelly sandwich [1], the robot should be able to store and organize compact representations effectively, rather than simply a large set of examples that are hard to generalize. Various aspects and levels of representations of manipulation actions have been studied, such as objects and tools [2], [3], manipulator trajectories [4], [5], actions and sub-actions [6], object-wise touch relations [7], action consequences or goals [8], etc. In this work, we focus on another crucial aspect of manipulation actions, which is the object-wise spatial relations in 3D space and the correlation of their temporal evolution to the underlying manipulation semantics.

Why are spatial relations crucial for interpreting manipulations? First of all, while most primitive spatial relations can be inferred directly from the perceptual input, their exploitation allows for more complex types of reasoning about not only geometric, but also the underlying physical relations between objects. For example, by perceiving that an apple is “ABOVE” a plate as well as that these two “TOUCH”, we can infer that the plate is the supporting object. Secondly, at a higher level, by grounding the spatial relations for the most commonly used prepositions in natural language, we effectively establish a bridge from observation to language and then from language to execution.

Additionally, a correct understanding of object-wise spatial relations for a given action is essential for a robot to perform the action successfully. For example, when a robot is asked to “Pour a cup of coffee from the pitcher”, it not only needs to recognize “cup” and “pitcher” and generate a sequence of trajectories to perform the “pour” action, but also needs to perform geometric precondition checks on the 3D spatial relations between “pitcher” and “cup”: unless the “pitcher” tip is “ABOVE” the “cup”, “pour” should not be triggered! In other words, a correct understanding of spatial relations can provide the premise of an accurate execution.

Few work has touched upon object-wise spatial relations for manipulation actions, due to the difficulties inherited from object tracking (inevitable occlusions, etc.). A joint segmentation and tracking technique to reason about “touch” and “contain” relations from top-down 2D views is used in [7]. The limitations of their approach come from reasoning in 2D. For example, their system is not able to differentiate between spatial predicates “above” and “in”. Other approaches on spatial reasoning [9] require additional equipment, such as motion capture suits, which makes them impractical for more general purpose applications.

The goal of our work is to develop a system which equips the robot with a fundamental and reliable understanding of 3D spatial relations. During both observing and executing manipulation actions, a robot with our proposed capabilities will be able to understand and reproduce the evolution of the spatial relations between involved objects with high accuracy. We show that a 3D spatial relation based representation is highly descriptive of the underlying action semantics.

At the lower level of this work, we developed a system for RGBD object segmentation and tracking that does not require additional markers on the objects and allows us to overcome difficulties arising from 2D-only image reasoning. Given point cloud representations for all tracked objects, we adopted a straightforward yet intuitive way to model geometric relations for the most commonly used natural language spatial prepositions by partitioning the space around each object. For a given object pair, our spatial relation predicates effectively capture the *spatial distribution* of the first object with respect to the second. The temporal evolution of each such distribution is encoded in a *Predicate Vector Sequence* (PVS). At a higher level, we *a)* propose a novel action descriptor, which is merely a properly ordered set of PVSes for all involved object pairs, and *b)* introduce its associated distance measure, which relies on finding an optimal, in

[1]K. Zampogiannis, Y. Yang, and Y. Aloimonos are from the Department of Computer Science, and [2]C. Fermüller is from UMIACS, University of Maryland, College Park, MD 20742, USA. {kzampog, yzyang, yiannis} at cs.umd.edu and {fer} at umiacs.umd.edu

terms of PVS similarity, object correspondence between two given actions in this representation. Experiments on real manipulation videos for various actions, performed with significant amounts of variation (e.g., different subjects, execution speeds, initial/final object placements, etc.), indicate that our proposed abstract representation successfully captures the manipulation's high-level semantic information, while demonstrating high discriminative performance.

II. Related work

From the very beginning of robotics research, a great amount of work has been devoted to the study of manipulation, due to its direct applications in intelligent manufacturing. With the recent development of advanced robotic manipulators, work on robust and accurate manipulation techniques has followed quickly. For example, [10] developed a method for the PR2 to fold towels, which is based on multiple-view geometry and visual processes of grasp point detection. In [11], they proposed learning object affordance models in multi-object manipulation tasks. Robots searching for objects were investigated in [12], using reasoning about both perception and manipulation. [4] and [5] developed manipulation and perception capabilities for their humanoid robot, ARMAR-III, based on imitation learning for human environments. A good survey on humanoid dual arm manipulation can be found in [13]. These works reached promising results on robotic manipulation, but they focused on specific actions without allowing for generalization. Here, we suggest that the temporal evolution of object-wise 3D spatial relations is a highly descriptive feature for manipulation actions and thus can be potentially used for generalizing learned actions.

Several recent works have studied spatial relations in the context of robot manipulation, either explicitly or implicitly. From a purely recognition point of view, the prepositions proposed in [14] can be directly used for training visual classifiers. Recently, [15] built a joint model of prepositions and objects in order to parse natural language commands, following the method introduced in [16]. In [17] and [18], supervised learning techniques were employed to symbolically ground a specific set of spatial relations, using displacements of likely object contact points and histograms that encode the relative position of elementary surface patches as features respectively. However, in general contexts, most common spatial prepositions do not have ambiguous geometric meanings and can, in principle, be directly modeled. Instead of grounding them via learning from large sets of annotated data, we directly model spatial relations according to their clear geometric interpretation.

In [7], they developed a 2D video segmentation and tracking system, through which they proposed a compact model for manipulation actions based on the evolution of simple, 2D geometric relations between tracked segments. Reasoning with only 2D segments enforces an unwanted and unnecessary type of abstraction in the representation due to the loss of geometric information. In this paper, instead, we infer spatial relations between segmented point clouds in 3D space, which enables our system to generate representations of controlled levels of abstraction and discriminative power.

III. Our approach

A. Overview of our method

A very brief description of the processing steps involved in our method is provided here and details are discussed in the following subsections.

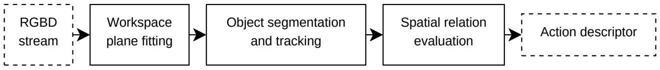


Fig. 1. Processing steps for our action descriptor extraction.

A schematic of our action descriptor extraction pipeline is given in Fig. 1. The input to our algorithm is an RGBD video of a manipulation action. The objects of interest are segmented in the first frame and tracked in 3D space throughout the rest of the sequence (Section III-B). At each time instant, we evaluate a predetermined set of pairwise spatial predicates for each pair of tracked objects (Section III-C), thus obtaining a sequence of spatial relation descriptors for every object pair (*Predicate Vector Sequence*, PVS). This set of sequences, arranged in a predetermined order, constitutes our proposed action descriptor (Section III-D). Subsequently, we define an appropriate pairwise distance function for this representation, which relies on standard dynamic programming techniques for time series similarity evaluation. Distance computation between two actions is reduced to finding an optimal, in a sense to be defined, object correspondence between the actions (Section III-E).

Assumptions. To simplify certain subtasks, we assume that the action takes place, at least at the beginning, on a planar surface (e.g., on a table) on which all involved objects initially lie. Furthermore, we assume that the depth sensor used to record the manipulation remains still throughout the action duration (no ego-motion). Since our method mostly pertains to applications of robots learning by watching humans or other robots and as long as objects remain visible in order to be reliably tracked, we do not consider the latter to be too restrictive.

B. Point cloud tracking

The initialization of the tracking procedure involves fitting a plane to the workspace surface in the first RGBD frame of the input sequence. This is done reliably using standard methods, i.e. least squares fitting under RANSAC. The points that lie a certain threshold distance above the estimated plane are clustered based on which connected component of their k-NN graph they belong to. Assuming that the maximum distance between points that belong to the same object is smaller than the minimum distance between points of different objects, this simple procedure yields an effective segmentation of all the objects in the initial frame.

Treating these initial segments (point clouds) as object models, we initiate one tracker for each object and perform rigid object tracking using the KLD-sampling adaptive Particle Filtering algorithm [19] that is implemented in [20]. This results in a point cloud for each object, for each video frame. We denote this set of point clouds at time index t by $\{X_1^t, \ldots, X_{N_o}^t\}$, where N_o is the number of objects in the action. In the following, we will use the terms X_i^t and "object i (at time t)", for some $i = 1, \ldots, N_o$, interchangeably. A sample output of our point cloud tracker for a two object manipulation recording is depicted in Fig. 2.

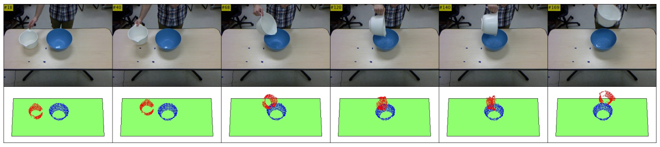


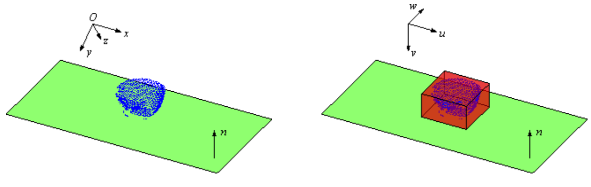
Fig. 2. Point cloud tracking for a `Pour` action. First row: RGB frames from input video. Second row: workspace plane and point clouds for the two tracked objects.

We must note that object segmentation and tracking are not the primary objectives of this study. The approach described here is the one we used in our experiments (Section IV) and primarily serves as a showcase for the feasibility of subsequent processing steps using readily available tools. We believe that any reasonably performing point cloud tracking algorithm is adequate for our purposes, as spatial relations can, in general, be quite insensitive to tracking accuracy.

C. Spatial relations

Here, we describe a straightforward yet intuitive approach to modeling pairwise spatial relations between objects, given their point clouds in 3D space. We begin by defining an auxiliary coordinate frame, whose axes will directly determine the `left`/`right`, `above`/`below` and `front`/`behind` directions. As mentioned previously, we assume a still camera (robot's eyes) looking at the planar surface where the manipulation takes place.

The auxiliary axes directions are calculated using general prior knowledge about the axis orientations of our RGBD sensor world coordinate frame, in which the object point clouds are in known positions, and the workspace normal vector that was estimated during the plane fitting step of segmentation/tracking. Naturally, since the planar workspace (initially) supports all the objects involved in the manipulation, we would want the `above`/`below` direction to be directly defined by its normal vector n.



(a) (b)

Fig. 3. Coordinate frames.

Our sensor coordinate frame is drawn in Fig. 3(a). As we can see, the z-axis corresponds to perceived depth, in roughly the front/behind direction, while the x and y axes point approximately to the right and downwards, respectively. This frame can be rotated so that the new axes, u, v and w, are aligned with the workspace and the aforementioned directions with respect to it (Fig. 3(b)). Axis v can be set to be parallel to n (e.g., pointing downwards). Axis w can then be defined as the projection of the original z-axis to the orthogonal complement of v. Axis u is then uniquely determined as being orthogonal to both v and w (e.g., so that the resulting frame is right-handed). Let $\hat{x}$, $\hat{y}$ and $\hat{z}$ be the unit length vectors that are codirectional with the sensor frame axes, $\hat{n}$ be a unit length workspace normal and $\hat{u}$, $\hat{v}$, $\hat{w}$ be the unit length vectors codirectional with the "workspace-aligned" auxiliary frame axes. The above construction, with the axis directions chosen as in Fig. 3(b), is captured by the following equations:

$$\begin{aligned}\hat{v} &= \mathrm{sgn}\left(\hat{y}\cdot\hat{n}\right)\hat{n},\\ \hat{w} &= \left(\hat{z} - \left(\hat{v}\cdot\hat{z}\right)\hat{v}\right)/\|\hat{z} - \left(\hat{v}\cdot\hat{z}\right)\hat{v}\|,\\ \hat{u} &= \hat{v}\times\hat{w},\end{aligned}$$

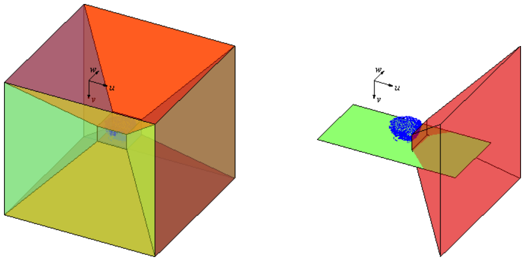
where $a \cdot b$ is the dot product of vectors a and b, $a \times b$ is their cross product, sgn is the signum function and $\|x\|$ is the Euclidean norm of x. Table I defines the 6 spatial relation defining directions in terms of the auxiliary frame axes.

TABLE I

SPATIAL RELATION DEFINING DIRECTIONS

Direction	`left`	`right`	`front`	`behind`	`above`	`below`
Reference vector	$-\hat{u}$	$+\hat{u}$	$-\hat{w}$	$+\hat{w}$	$-\hat{v}$	$+\hat{v}$

To infer spatial relations between objects, we will first build models for the regions of 3D space *relative* to an object. For example, to reason about some object being on the `right` of object X at some given time, we first explicitly model the space region on the `right` of X. We will consider 7 space regions with respect to a given object: one that will represent the object interior (for the `in` relation) and 6 around the object, aligned to the directions of Table I. We will represent all these relative spaces of X as convex polyhedra and denote them as $S_r(X)$, for $r \in \{\mathtt{in}, \mathtt{left}, \mathtt{right}, \mathtt{front}, \mathtt{behind}, \mathtt{below}, \mathtt{above}\}$, so that, for example, $S_{\mathtt{right}}(X)$ is the space region on the right of X.



(a) (b)

Fig. 4. Directional relative spaces.

We model $S_{\mathtt{in}}(X)$ simply as the smallest bounding box (cuboid) of X that is aligned with the u, v, w axes (Fig. 3(b) for the blue point cloud). The 6 "directional" relative spaces are built upon this one, as indicated in Fig. 4(a). Consider a uvw-aligned cube, concentric to $S_{\mathtt{in}}(X)$, of edge length significantly larger than the maximum workspace dimension. Clearly, each face of $S_{\mathtt{in}}(X)$ with the closest face of the surrounding cube that is parallel to it can uniquely define an hexahedron that also has these two as faces. We will use these 6 hexahedra as models for our directional relative spaces. In Fig. 4(b), we draw $S_{\mathtt{right}}(X)$, where X is represented by the blue point cloud.

A few remarks are in order. First, all relative spaces are represented as sets (conjunctions) of linear constraints, so checking if a point lies in them is very easy. Second, one would expect the directional relative spaces to be unbounded (e.g., polyhedral cones). They can be modeled this way by simply dropping one of their defining linear constraints. Finally, while more elaborate models can be considered for $S_{\mathtt{in}}(X)$, like the convex hull of X, the construction of Fig. 4(a) has the nice property of *partitioning* the space around X. This will enable us to easily reason about object relations in a probabilistic manner, in the sense that our (real-valued) relation predicates will define the *spatial distribution* of an object relative to another. Clearly, this provides a more flexible framework than modeling using binary-valued predicates, which can be simply inferred anyway.

We are now ready to define our models for a set of basic pairwise spatial object relations. We will model 7 of them (the inclusion relation and the 6 "directional" ones) directly based on their respective relative space, and an additional one that is indicative of whether two objects are in physical contact (touch). Let $\mathcal{R}^f = \{\mathtt{in}, \mathtt{left}, \mathtt{right}, \mathtt{front}, \mathtt{behind}, \mathtt{below}, \mathtt{above}, \mathtt{touch}\}$ be our full set of spatial relations. We note that these primitive relations correspond to the spatial prepositions most commonly used in natural language and can be used to model more complex relations (e.g., we can reason about on based on above and touch). For each $r \in \mathcal{R}^f$, we will define a pairwise predicate $R_r(X_i^t, X_j^t)$ that quantifies exactly whether X_i^t is positioned relatively to X_j^t according to relation r. For example, $R_{\mathtt{left}}(X_i^t, X_j^t)$ indicates to what degree object i is on the left of object j at time t.

Let $\mathcal{R}^s = \mathcal{R}^f \diagdown \{\mathtt{touch}\}$ be the set of relations that can be defined by an explicit space region with respect to X_j^t (reference object). Instead of making hard decisions, i.e. using binary values, we let the predicates $R_r(X_i^t, X_j^t)$, for $r \in \mathcal{R}^s$, assume real values in $[0, 1]$ that represent the fraction of X_i^t that is positioned according to r with respect to X_j^t. A reasonable way to evaluate $R_r(X_i^t, X_j^t)$, for $r \in \mathcal{R}^s$, is then simply as the fraction of points of X_i^t that lie in the relative space $S_r(X_j^t)$:

$$R_r(X_i^t, X_j^t) = \left|X_i^t \cap S_r(X_j^t)\right| / \left|X_i^t\right|, \qquad (1)$$

where by $|A|$ we denote the cardinality of set A. Since $S_r(X_j^t)$, for $r \in \mathcal{R}^s$, are mutually disjoint by construction, the predicates of (1) have the property:

$$\sum_{r \in \mathcal{R}^s} R_r(X_i^t, X_j^t) = 1,$$

i.e. they define a distribution of the points of X_i^t to the relative spaces of X_j^t.

The touch relation can be useful in capturing other, non-primitive spatial relations. For example, by having models for both above and touch, one can express the on relation as their "conjunction". More expressive power is desirable, as it can translate to better discriminative performance for our representation. We let our contactual predicate $R_{\mathtt{touch}}(X_i^t, X_j^t)$ assume binary values in $\{0, 1\}$. Whenever either of $R_{\mathtt{in}}(X_i^t, X_j^t)$ or $R_{\mathtt{in}}(X_j^t, X_i^t)$ is greater than zero, we can assume that X_i^t is touching X_j^t and, therefore, $R_{\mathtt{touch}}(X_i^t, X_j^t) = 1$. If this were the defining condition for $R_{\mathtt{touch}}$, incorporating the contactual predicate to our set of spatial relations would be redundant, in the sense that we already model $R_{\mathtt{in}}$. However, there are cases where both $R_{\mathtt{in}}(X_i^t, X_j^t)$ and $R_{\mathtt{in}}(X_j^t, X_i^t)$ are equal to zero, while X_i^t is touching X_j^t. We may fail to detect this situation using the above intersection test for various reasons. For example, the two objects could simply be extremely close to each other (touching), without part of one lying inside the other. The precision of our sensor and the accuracy of our tracker can also cause the above condition to falsely fail. To compensate for these situations, whenever both $R_{\mathtt{in}}(X_i^t, X_j^t) = 0$ and $R_{\mathtt{in}}(X_j^t, X_i^t) = 0$, in which case X_i^t and X_j^t are guaranteed to be linearly separable point sets, we perform an additional "proximity" test. We train a linear binary SVM on $X_i^t \cup X_j^t$, with the class labels given by the object ownership for each data point, and use the classifier margin, d_m, as a measure of distance between X_i^t and X_j^t. If d_m falls below a preset threshold d_T, we set $R_{\mathtt{touch}}(X_i^t, X_j^t) = 1$. Our complete model of the contactual predicate is then given by:

$$R_{\mathtt{touch}}(X_i^t, X_j^t) = \begin{cases} 1 & \text{if } R_{\mathtt{in}}(X_i^t, X_j^t) > 0 \\ & \text{or } R_{\mathtt{in}}(X_j^t, X_i^t) > 0 \\ & \text{or } d_m < d_T, \\ 0 & \text{otherwise.} \end{cases}$$

The value of d_T depends, among other things, on point cloud precision related parameters (e.g., sensor and tracking errors) and was set to a few millimeters in our trials. We note that, of all relations in $\mathcal{R}^f$, only touch is symmetric.

Spatial abstraction. At this point, we have defined our models for all relations $r \in \mathcal{R}^f$. Our goal is to define an action representation using the temporal evolution of spatial object relations. However, tracking all relations in $\mathcal{R}^f$ can yield a representation that is viewpoint-specific or unnecessarily execution-specific. For example, disambiguating between left and right or front and behind or, actually, any two of these is clearly dependent on the sensor viewpoint and might not be informative about the actual manipulation semantics. As an example, consider a "stir the coffee" action that involves a cup and a spoon. Picking up the spoon from the left of the cup, then stirring the coffee and finally leaving the spoon on the right of the cup is expected to have the same high-level semantics as picking up the spoon from the right of the cup, stirring and then leaving it in front of the cup. For this reason, we combine the relations $\{\mathtt{left}, \mathtt{right}, \mathtt{front}, \mathtt{behind}\}$ into one, which we can simply name around, to obtain a desirable kind of spatial abstraction that, in most cases we can think of, does not leave out information that is actually manipulation-specific. The new relation can be viewed as the disjunction of the 4 ones it replaces and its predicate is given by:

$$\begin{aligned} R_{\mathtt{around}}(X_i^t, X_j^t) =& R_{\mathtt{left}}(X_i^t, X_j^t) + R_{\mathtt{right}}(X_i^t, X_j^t) + \\ & R_{\mathtt{front}}(X_i^t, X_j^t) + R_{\mathtt{behind}}(X_i^t, X_j^t). \end{aligned}$$

This makes $\mathcal{R}^a = \{\mathtt{in}, \mathtt{around}, \mathtt{below}, \mathtt{above}, \mathtt{touch}\}$ the set of relations upon which we will build our action descriptors.

D. Action descriptors

Let $\Phi^t(i,j)$ be the $|\mathcal{R}^a|$-dimensional vector of all relation predicates $R_r(X_i^t, X_j^t)$, $r \in \mathcal{R}^a$, for object i relative to object j at time t, arranged in a fixed relation order, e.g., (`in`, `around`, `below`, `above`, `touch`), so that:

$$\Phi^t(i,j) \equiv \left(R_{\mathtt{in}}(X_i^t, X_j^t), \ldots, R_{\mathtt{touch}}(X_i^t, X_j^t)\right), \qquad (2)$$

where $i, j = 1, \ldots, N_o$ and $i \neq j$. Let $\Phi(i,j)$ denote the sequence of the predicate vectors (2), for $t = 1, \ldots, T$:

$$\Phi(i,j) \equiv \left(\Phi^1(i,j), \ldots, \Phi^T(i,j)\right).$$

The latter captures the temporal evolution of all spatial relations in $\mathcal{R}^a$ of object i with respect to object j throughout the duration of the manipulation execution. We will call $\Phi(i,j)$ a *Predicate Vector Sequence* (PVS). PVSes constitute the building block of our action descriptors and can be represented as $|\mathcal{R}^a| \times T$ matrices.

Our proposed action descriptors will contain the PVSes for all object pairs in the manipulation. As will become clear in the next subsection, comparing two action descriptors reduces to finding an optimal correspondence between their involved objects, e.g., infer that object i_1 in the first action corresponds to object i_2 in the second. To facilitate this matching task, we require our proposed descriptors to possess two properties.

The first has to do with the fact that the spatial relations we consider are not symmetric. A simple solution to fully capture the temporally evolving spatial relations between objects i and j, where none of the objects acts as a reference point, is to include both $\Phi(i,j)$ and $\Phi(j,i)$ in our descriptor, for $i, j = 1, \ldots, N_o$ and $i \neq j$. This might seem redundant, but, given our predicate models, there is generally no way to exactly infer $\Phi^t(j,i)$ from $\Phi^t(i,j)$ (e.g., due to different object dimensions). Our descriptor will then consist of $N_r = N_o(N_o - 1)$ PVSes, as many as the *ordered* object pairs.

Finally, we need to be able to identify which (ordered) object pair a PVS refers to, i.e. associate every PVS in our representation with a tuple (i,j) of object indices. We opted to do this implicitly, by introducing a reverse indexing function and encoding the mapping information in the *order* in which the PVSes are stored. Any bijective function I_{N_o} from $\{1, \ldots, N_r\}$, the set of PVS indices, onto $\{(i,j) \mid i,j = 1, \ldots, N_o \wedge i \neq j\}$, the set of ordered object index pairs, is a valid choice as an indexing function. Our proposed action descriptors are then ordered sets of the form:

$$A \equiv (\Phi_1, \ldots, \Phi_{N_r}),$$

where, for $k = 1, \ldots, N_r$, $\Phi_k = \Phi(i,j)$ and $(i,j) = I_{N_o}(k)$. Function I_{N_o} can be embedded in the descriptor extraction process. Utilizing the knowledge of the PVS ordering within an action descriptor will simplify the formulation of establishing an object correspondence between two actions in the following subsection.

E. Distance measure

To complete our proposed representation, we now introduce an appropriate distance function d on the descriptors we defined above. If A^1 and A^2 are two action descriptors, we design $d(A^1, A^2)$ to be a symmetric function that gives a *time-normalized* distance between the two actions. Additionally, we allow comparisons between manipulations that involve different numbers of objects. This will enable us to reason about the similarity between an action and a *subset* (in terms of the objects it involves) of another action. In the following, for $k = 1, 2$, let N_r^k be the number of PVSes in A^k and N_o^k be the number of objects in manipulation A^k, so that $N_o^k(N_o^k - 1) = N_r^k$.

At the core of our action distance evaluation lies the comparison between PVSes. Each PVS is a time series of $|\mathcal{R}^a|$-dimensional feature vectors that captures the temporal evolution of all spatial relations between an ordered object pair. Different action executions have different durations and may also differ significantly in speed during the course of manipulation: e.g., certain subtasks may be performed at different speeds in different executions of semantically identical manipulations. To compensate for timing differences, we use the Dynamic Time Warping (DTW) [21] algorithm to calculate time-normalized, pairwise PVS distances. We consider the symmetric form of the algorithm in [21], with no slope constraint or adjustment window restriction.

Let $\Phi^1_{r^1}$ and $\Phi^2_{r^2}$ be PVSes in A^1 and A^2, respectively, where $r^1 = 1, \ldots, N_r^1$ and $r^2 = 1, \ldots, N_r^2$. We form the $N_r^1 \times N_r^2$ matrix $C = (c_{r^1 r^2})$ of all time-normalized distances between some PVS in A^1 and some PVS in A^2, where:

$$c_{r^1 r^2} = \text{DTW}(\Phi^1_{r^1}, \Phi^2_{r^2}).$$

In the following, we will calculate $d(A^1, A^2)$ as the total cost of an optimal correspondence of PVSes between A^1 and A^2, where the cost of assigning $\Phi^1_{r^1}$ to $\Phi^2_{r^2}$ is given by $c_{r^1 r^2}$.

One could simply seek a minimum cost matching of PVSes between two action descriptors by solving the linear assignment combinatorial problem, with the assignment costs given in C (e.g., by means of the Hungarian algorithm [22]). This would yield an optimal cost *PVS correspondence* between the two actions. However, it is clear that the latter does not necessarily translate to a valid *object correspondence*. Instead, we directly seek an object correspondence between A^1 and A^2 that induces a minimum cost PVS correspondence. The object correspondence between A^1 and A^2 can be represented as an $N_o^1 \times N_o^2$ binary-valued assignment matrix $X = (x_{ij})$, where $x_{ij} = 1$ if and only if object i in A^1, $i = 1, \ldots, N_o^1$, is matched to object j in A^2, $j = 1, \ldots, N_o^2$. We require that every row and every column of X has at most one nonzero entry and that the sum of all entries in X is equal to $\min(N_o^1, N_o^2)$. This ensures that X defines an one-to-one mapping from the objects in the action involving the fewest objects to the objects in the other action. An object assignment X is then evaluated in terms of the PVS correspondence it defines. We denote the latter by $Y_X = (y_{r^1 r^2})$, where $y_{r^1 r^2} = 1$ if and only if PVS r^1 in A^1 is mapped to r^2 in A^2. The $N_r^1 \times N_r^2$ matrix Y_X has the same structure as X, with $\min(N_r^1, N_r^2)$ nonzero entries. The cost of assignment X, in terms of its induced PVS assignment Y_X, is then given by the sum of all individual

PVS assignment costs:

$$J(Y_X) = \sum_{r^1=1}^{N_r^1} \sum_{r^2=1}^{N_r^2} c_{r^1 r^2} y_{r^1 r^2}. \tag{3}$$

According to our descriptor definition in the previous subsection, the PVS with index r^1 in A^1 refers to the ordered object pair $(o_1^1, o_2^1) = I_{N_o^1}(r^1)$ and, similarly, r^2 in A^2 refers to $(o_1^2, o_2^2) = I_{N_o^2}(r^2)$ (superscripts indicate action). Clearly, $y_{r^1 r^2}$ is nonzero if and only if object o_1^1 in A^1 is assigned to o_1^2 in A^2 and o_2^1 in A^1 is assigned to o_2^2 in A^2:

$$y_{r^1 r^2} = x_{o_1^1 o_1^2} x_{o_2^1 o_2^2}.$$

Using the above, we can rewrite and optimize (3) in terms of the object assignment variables x_{ij}, for $i = 1, \ldots, N_o^1$ and $j = 1, \ldots, N_o^2$:

$$\boxed{\begin{array}{ll} \underset{X}{\text{Minimize}} & J(X) = \sum_{r^1=1}^{N_r^1} \sum_{r^2=1}^{N_r^2} c_{r^1 r^2} x_{o_1^1 o_1^2} x_{o_2^1 o_2^2} \\ \text{where} & (o_1^1, o_2^1) = I_{N_o^1}(r^1),\ (o_1^2, o_2^2) = I_{N_o^2}(r^2) \\ \text{subject to} & \sum_{j=1}^{N_o^2} x_{ij} \leq 1, \quad i = 1, \ldots, N_o^1 \\ & \sum_{i=1}^{N_o^1} x_{ij} \leq 1, \quad j = 1, \ldots, N_o^2 \\ & \sum_{i=1}^{N_o^1} \sum_{j=1}^{N_o^2} x_{ij} = \min(N_o^1, N_o^2) \\ & x_{ij} \in \{0, 1\}, \quad i \in \{1, \ldots, N_o^1\}, \\ & \qquad\qquad\qquad j \in \{1, \ldots, N_o^2\}. \end{array}}$$

The binary quadratic program above encodes an instance of the quadratic assignment problem (QAP), which is NP-hard. QAP instances of size (number of objects) $N > 30$ are considered intractable [23]. However, most manipulations of practical interest involve a number of objects well below that limit. In our implementation, we used the SCIP (constraint integer programming) solver [24]. To evaluate the correctness and running time behavior of our optimization scheme, we ran a small number of tests. For various numbers of objects N, we built an action descriptor A^1 by randomly generating $N(N-1)$ PVSes. We constructed A^2 from A^1 based on an arbitrary object permutation (assignment), by rearranging the PVSes of A^1 (according to I_N and the known object permutation) and adding Gaussian noise to them. Minimization of $J(X)$ gave back the correct object assignment, even for significant amounts of noise variance. Running time for $N = 10$ objects was in the order of a few (≈ 10) seconds on a laptop machine.

The minimum value of $J(X)$, over all possible object assignments, directly defines the distance between actions A^1 and A^2:

$$d(A^1, A^2) = \min_X (J(X)).$$

The function $d(A^1, A^2)$ is symmetric and, being a sum of DTW distances, gives a time-normalized measure of action dissimilarity. As noted before, $d(A^1, A^2)$ is also defined when A^1 and A^2 involve different numbers of objects ($N_o^1 \neq N_o^2$). For example, if $N_o^1 < N_o^2$, $d(A^1, A^2)$ is expected to be exactly the same as if A^2 only involved the N_o^1 of its objects to which the objects in A^1 are assigned. This flexibility can be useful in sub-action matching scenarios.

IV. Experiments

A. Data description

All our experiments were performed on a set of 21 RGBD sequences of manipulation executions. All actions involve 2 objects and are partitioned in 4 distinct semantic classes:

- `Pour`: water poured from a pitcher into a bowl (8 executions).
- `Transfer`: small object placed inside a bowl (6 executions).
- `Stack`: a book placed on top of another (2 executions).
- `Stir`: bowl content stirred using a ladle (5 executions, one of them performed by our robot).

Executions within each semantic class were performed by various individuals and with various initial and final positions of the manipulated objects. For example, in some instances of `Stir`, the ladle was initially picked from the left of the bowl and was finally placed on its left, in others, it was picked from the right and then placed on the left, etc. Naturally, there were significant timing differences across instances of the same semantic class (different overall durations and execution speeds at each action phase).

We also included a robot execution for an instance of `Stir` (Fig. 5, bottom rows) that took roughly 4 times the average human execution duration to complete and demonstrated disproportionately long “idle” phases throughout the manipulation. Our robot platform is a standard Baxter humanoid with parallel grippers. To generate trajectories, we used predefined dynamic movement primitives [25]. The trajectory start and end points were given from the point cloud segmentation and transferred onto the robot via a standard inverse kinematics procedure. We also used visual servoing to ensure a firm grasp of the tool.

B. Spatial relations evaluation

We begin with a quick evaluation of the performance of our spatial relation models. To establish our ground truth, we sampled 3 time instances from each execution sequence. To evaluate a rich enough set of spatial relations, we sampled at roughly the beginning, the middle and the end of each manipulation. For each sample frame, we picked one of the objects to act as reference (say X_1) and picked the relation $r \in \mathcal{R}^s = \{$`in`, `left`, `right`, `front`, `behind`, `below`, `above`$\}$ that best described the position of the second object, X_2, relative to X_1. For testing, we calculated the spatial predicates $R_r(X_2, X_1)$, for all $r \in \mathcal{R}^s$ for all 63 annotated frames and labeled each according to the relation of maximum predicate value. This gave a classification error rate of $3/63 \approx 4.8\%$. However, 2 of the errors were due to tracking issues and not the spatial predicates per se: in both `Stack` executions, significant part of the book at the bottom overlapped the top one (the tracked point clouds were barely distinguishable), so `in` dominated `above`. The third error was from a `Stir` instance (during the stirring phase), where we decided `in` with a ground truth of `above`, which was the second largest predicate. Overall, we believe that, up to severe tracking inaccuracy, our spatial relation estimation can be considered reliable.

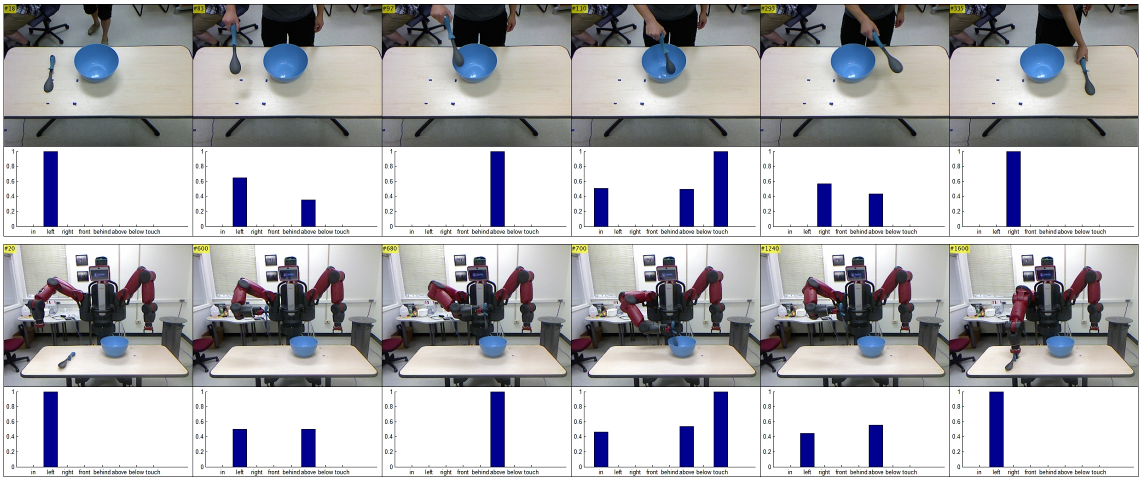


Fig. 5. Spatial relations evolution: ladle relative to bowl for two instances of `Stir`.

In Fig. 5, we depict part of the temporal evolution of the spatial predicate vector of the ladle relative to the bowl for two executions of `Stir` (samples from the corresponding PVS for all relations in $\mathcal{R}^f$): one performed by a human and one by our robot. From the figure, we can see that our system can reliably track the temporal evolution of spatial relations in both observation and execution scenarios.

C. Action classification

To evaluate the discriminative performance of our proposed representation, we begin by forming the matrix $D = (d_{ij})$ of all pairwise distances for our $N_a = 21$ manipulation executions, where $d_{ij} = d(A^i, A^j)$, for $i, j = 1, \ldots, N_a$. We depict the values of D in Fig. 6. As expected, D is symmetric with zero diagonal. Manipulation executions of the same semantic class were grouped together to consecutive indices (e.g., 1-8 for `Pour`, 9-14 for `Transfer`, 15-16 for `Stack` and 17-21 for `Stir`). Given this, the evident block-diagonal structure of low distance values in D (blue regions) suggests that the proposed representation can be quite useful in classification tasks.

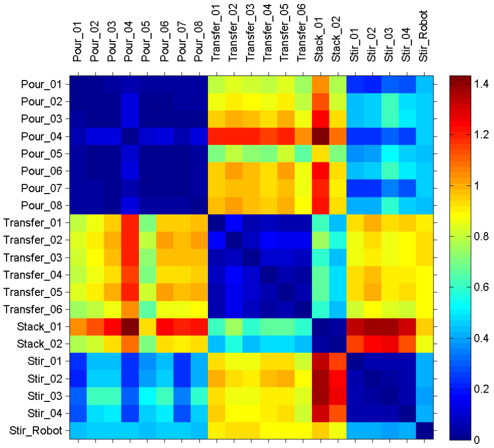


Fig. 6. Matrix $D = (d_{ij})$ of pairwise distances.

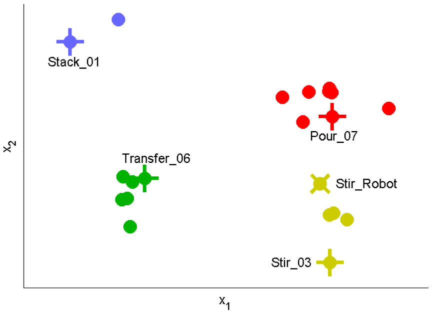


Fig. 7. Clustering and embedding of our action descriptors in 2 dimensions, based on our similarity/distance measure.

To confirm this intuitive observation, we considered a clustering scenario by applying the Affinity Propagation algorithm [26] on our data. Affinity Propagation is an unsupervised clustering algorithm that does not assume prior knowledge of the number of classes. Instead, the resulting number of clusters depends on a set of real-valued "preference" parameters, one for each data point, that express how likely the point is to be chosen as a class "exemplar" (cluster centroid). A common choice [26] is to use the same preference value for all points, equal to the median of all pairwise similarities. A similarity measure between actions A^i and A^j, for $i, j = 1, \ldots, N_a$, is directly given by $s_{ij} = -d_{ij}$. Clustering using this scheme resulted in 4 clusters that correctly corresponded to our 4 semantic classes and there were no classification errors. In Fig. 7, we plot a 2-dimensional embedding of the action descriptors for all executions, where we use the same color for all data points of the same cluster and mark the cluster centroids. The outcome of our simple clustering experiment confirms our intuition about matrix D (Fig. 6), suggesting that our proposed abstract representation is indeed descriptive of the actual high-level manipulation semantics.

It is worth noting that the descriptor for the robot stirring scenario was correctly classified as a `Stir` instance (marked in Fig. 7). This empirically shows that, even when the specific movement trajectories are quite different, e.g., between human trials and robot executions, our PVS-based representations remain relatively invariant under the proposed distance function. Thus, our learned from observation manipulation representations could be used to provide additional constraints for robot control policies.

V. Conclusions and Future Work

In this paper, we introduced our direct take on grounding spatial relations, by properly partitioning the space around an object to a set of relative spaces, and then proposed a novel compact representation that captures the geometric object interactions during the course of a manipulation. Experiments conducted on both human and robot executions validate that *1)* our relative space models successfully capture the geometric interpretation of their respective relation and the corresponding predicates can be reliably evaluated; *2)* the temporal evolution of object-wise spatial relations, as encoded in our abstract representation, is indeed descriptive of the underlying manipulation semantics.

Knowledge of models for spatial relations for common natural language prepositions is a very strong capability by itself. First, it enables certain types of nontrivial spatial reasoning, which is a fundamental aspect of intelligence. Second, it narrows the gap between observation and execution, with language acting as the bridge. Particularly, from the human-robot interaction perspective, a robot equipped with these models will be able to answer a rich repertoire of spatial queries and understand commands such as: "pick up the object on the left of object X and in front of object Y". Another such task, closer in spirit to this work, is the automatic generation of natural language descriptions for observed actions.

In this work, the matching of our spatial-relation sequence representations is performed at once on whole sequences. Currently, we are investigating the possibility of modifying the matching algorithm into an online one. An online action matching algorithm is needed if we want a fast system that observes actions and is able to *predict* during their course (e.g., for monitoring the correctness of execution). Additionally, here, we only took into account one aspect of manipulation actions: object-wise spatial relations. A complete action model, that would attempt to bridge the gap between observation and execution, needs to be a multi-layer combination of spatial relations and many other aspects, such as movement trajectories, objects, goals, etc.

Acknowledgements. This research was funded by the support of the European Union under the Cognitive Systems program (project POETICON++), the National Science Foundation under INSPIRE grant SMA 1248056, and by DARPA through U.S. Army grant W911NF-14-1-0384 under the project: Shared Perception, Cognition and Reasoning for Autonomy.

References

[1] W. Yi and D. Ballard, "Recognizing behavior in hand-eye coordination patterns," *International Journal of Humanoid Robotics*, vol. 6, no. 03, pp. 337–359, 2009.

[2] H. Kjellström, J. Romero, D. Martínez, and D. Kragić, "Simultaneous visual recognition of manipulation actions and manipulated objects," *Proceedings of the 2008 IEEE European Conference on Computer Vision*, pp. 336–349, 2008.

[3] A. Gupta and L. Davis, "Objects in action: An approach for combining action understanding and object perception," in *Proceedings of the 2007 IEEE International Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[4] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 02, pp. 183–202, 2008.

[5] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schrder, and R. Dillmann, "Toward humanoid manipulation in human-centred environments," *Robotics and Autonomous Systems*, vol. 56, pp. 54–65, 2008.

[6] A. Guha, Y. Yang, C. Fermüller, and Y. Aloimonos, "Minimalist plans for interpreting manipulation actions," in *Proceedings of the 2013 International Conference on Intelligent Robots and Systems*. Tokyo: IEEE, 2013, pp. 5908–5914.

[7] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object-action relations by observation," *I. J. Robotic Res.*, vol. 30, no. 10, pp. 1229–1249, 2011.

[8] Y. Yang, C. Fermüller, and Y. Aloimonos, "Detection of manipulation action consequences (MAC)," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. Portland, OR: IEEE, 2013, pp. 2563–2570.

[9] M. Wächter, S. Schulz, T. Asfour, E. Aksoy, F. Wörgötter, and R. Dillmann, "Action sequence reproduction based on automatic segmentation and object-action complexes," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2013, pp. 0–0.

[10] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2308–2315.

[11] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 4373–4378.

[12] M. R. Dogar, M. C. Koval, A. Tallavajhula, and S. Srinivasa, "Object search by manipulation," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013.

[13] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation: A survey," *Robotics and Autonomous Systems*, 2012.

[14] A. Gupta and L. S. Davis, "Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers." in *ECCV (1)*, ser. Lecture Notes in Computer Science, D. A. Forsyth, P. H. S. Torr, and A. Zisserman, Eds., vol. 5302. Springer, 2008, pp. 16–29.

[15] S. Guadarrama, L. Riano, D. Golland, D. Gohring, Y. Jia, D. Klein, P. Abbeel, and T. Darrell, "Grounding spatial relations for human-robot interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2013.

[16] D. Golland, P. Liang, and D. Klein, "A game-theoretic approach to generating spatial descriptions," in *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2010, pp. 410–419.

[17] B. Rosman and S. Ramamoorthy, "Learning spatial relationships between objects," *I. J. Robotic Res.*, vol. 30, no. 11, pp. 1328–1342, 2011.

[18] S. Fichtl, A. McManus, W. Mustafa, D. Kraft, N. Kruger, and F. Guerin, "Learning spatial relationships from 3D vision using histograms," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 501–508.

[19] D. Fox, "KLD-Sampling: Adaptive Particle Filters," in *Advances in Neural Information Processing Systems 14 (NIPS 2001)*. MIT Press, 2001, pp. 713–720.

[20] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[21] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, Feb 1978.

[22] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[23] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European Journal of Operational Research*, vol. 176, no. 2, pp. 657–690, 2007.

[24] T. Achterberg, "SCIP: Solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, July 2009.

[25] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1398–1403.

[26] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.