Intelligent Robots and Systems, October 2012

Linguistic Transfer of Human Assembly Tasks to Robots

Neil Dantam Irfan Essa Mike Stilman

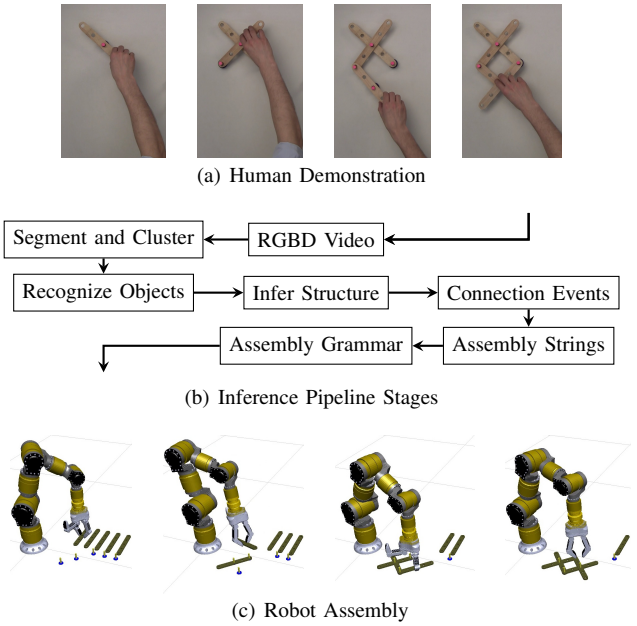
***Abstract*— We demonstrate the automatic transfer of an assembly task from human to robot. This work extends efforts showing the utility of linguistic models in verifiable robot control policies by now performing real visual analysis of human demonstrations to automatically extract a policy for the task. This method tokenizes each human demonstration into a sequence of object connection symbols, then transforms the set of sequences from all demonstrations into an automaton, which represents the task-language for assembling a desired object. Finally, we combine this assembly automaton with a kinematic model of a robot arm to reproduce the demonstrated task.**

I. Introduction

Hybrid system models combining formal language and differential equations are an effective and powerful approach to verify and control robotic systems. However, the development of these formal models is usually a manual process requiring both mathematical and domain-specific expertise. It would be more desirable to automate the development of formal tools for robotic systems. Using a linguistic representation of task decomposition and robot action, we demonstrate the automatic transfer of an assembly task from human to robot through a grammar suitable for both formal verification and efficient execution.

In previous work, we developed the Motion Grammar (MG) to represent complex robot tasks such as interactive Human-Robot Chess [1] as hybrid controllers based entirely on parsing a Context-Free Grammar. We also controlled a robot arm with MG for the game Yamakuzushi and formally explored correctness and completeness as they relate to a grammar-based controller [2]. Furthermore, we introduced a basic calculus of transformation rules that operate on both the syntax (discrete productions) and semantics (continuous control laws) of a Motion Grammar [3]. Our prior work builds on grammar-based methods widely employed to control robots and other dynamical systems [4], [5], [6], [7], [8]. The primary challenge for all linguistic approaches to control is the development of suitable models and specifications for the system, particularly for complex and varied robot tasks.

The contribution of this paper is a demonstration of *automatic transfer of an assembly task from human to robot using a verifiable model that represents the control policy for the task.* This demonstration system is a novel integration of techniques from computer vision, optimization, and language theory to go from physical human activity to syntactic task specification, Fig. 1. We operate in the application domain of assembly using a construction set of small wooden objects. In this domain, we convert a human demonstration to symbolic form, infer a syntactic model from a set of demonstrations, and simulate a robot reproducing the demonstrated task.



(a) Human Demonstration

(b) Inference Pipeline Stages

(c) Robot Assembly

Fig. 1. Automatic Transfer of Human Demonstration to Robot Assembly.

II. Related Work

We briefly summarize work in the areas of hybrid controls, grammatical inference, learning from demonstration, and robotic assembly, as it relates here. In addition, we give a consolidated explanation of the existing algorithms applied for this approach in [9].

Hybrid Control is an active research area, exploring systems with both discrete, event-driven, dynamics and continuous, time-driven, dynamics. [10] first applied Language and Automata Theory [11] to Discrete Event Systems. Hybrid Automata typically combine a Finite State Machine (FSM) with differential equations associated with each FSM state. This is a widely studied and utilized model [4], [5], [6], [7], [8]. In this paper, we model hybrid systems using the Motion Grammar which represents continuous dynamics with differential equations and discrete dynamics using a Context-Free Grammar (CFG).

There are numerous methods and domain-specific languages for the specification of robot tasks. The Subsumption Architecture specifies a task policy with a set of parallelly-executing Finite State Machines [12]. [13] presents a domain-specific language for robot manipulation tasks. [14] describes a type-safe, Turing-complete robot programming language. [15] uses an English-like syntax for Linear Temporal Logic in mobile robot motion planning. Unlike these efforts, we

The authors are with the Center for Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, GA 30332, USA. ntd@gatech.edu, irfan@cc.gatech.edu, mstilman@cc.gatech.edu. This work is supported by NSF grants CNS1146352 and CNS1059362.

focus on the automatic inference of robot task specifications from human demonstration.

Language models have previously been applied to activity recognition. Syntactic pattern recognition, described by [16], uses a grammar to parse and classify a tokenized system, as opposed to classification techniques such as SVMs that directly divide a continuous feature space. The syntactic approach is applied to human activity recognition by [17], [18], [19]. Our goal here is not to just recognize or classify an activity based on an existing grammar, but to observe an activity and produce the grammar describing it, and then transfer that grammar to a robot for execution.

Grammatical inference is an ongoing field of research focused on developing language models from example strings and learner queries [20]. While there are a number of positive results in the field, trivial grammatical inference problems are often undecidable. For example, the class of regular languages cannot be learned solely from positive examples. To develop a workable system given these challenges, we initially focus in inferring grammars for finite languages. However, our overall approach is also amenable to more powerful forms of inference such as informed learning.

There are numerous other approaches to learning from demonstration for robotic systems [21], [22]. Many approaches focus on learning continuous trajectories [23], while in this work, we focus on a symbolic abstraction of a specific task. Other symbolic learning approaches include [24] which learns goal configurations for sets of objects and [25] which learns a logical model for a STRIPS planner from multiple human demonstrations. Our work differs from these other methods by producing a syntactic task model which, combined with the semantics for a robot, represents a hybrid dynamical control policy that is formally verifiable and efficiently executable.

There are numerous other approaches for robot assembly tasks [26], [27]. Rather than treat the assembly task in isolation, our goal is to infer from human demonstration a verifiable and executable policy in the form of a grammar to control a robot performing this assembly task.

III. Language for Robot Assembly

To explore the automatic generation of grammars for robots, we focus on the domain of object assembly. Our system observes human demonstrations of assembly for some desired object, and from those demonstrations automatically constructs a formal grammar that represents the same task. We first review modeling robots in formal language, then describe our language for assembly tasks.

A. Robotic Systems as Language: The Motion Grammar

We can view both humans and robots as *hybrid systems* containing both continuous and discrete elements. We can model continuous system dynamics with differential equations and discrete dynamics with *formal language*. A *formal language* is a set of strings. Strings are sequences of atomic symbols, which we use to describe discrete events, predicates, and actions within our system. *Grammars* define formal languages. This combined hybrid model defines the full system dynamics and represents a *policy* to control the system.

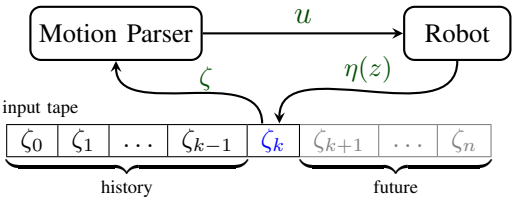


Fig. 2. Operation of the Motion Grammar.

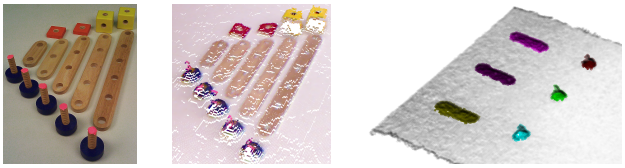
The Motion Grammar (MG) [1], [2], [3] is a model for hybrid systems. While our approach to grammar inference is widely applicable to any linguistic hybrid control method, we focus on the MG to provide a clear illustration. The MG combines differential equations for continuous dynamics with a Context-Free grammar for discrete dynamics. This model lets us formally describe complicated robotic systems such as the physical human-robot chess match in [1] and also gives us guarantees on both efficiency and verifiability of our control policy [2]. We now investigate automating the generation of the MG itself.

Fig. 2 illustrates the operation of the Motion Grammar. The sensor output z is discretized into a stream of tokens ζ for the parser to read. The token type ζ is used to pick the correct grammatical production to expand at that particular step, and the semantic rule for that production uses the continuous value z to generate the input u. Thus, the Motion Grammar represents the language of the robotic system.

We represent language syntax in this paper using the conventional Backus-Naur Form (BNF) for Context-Free Grammars and also with state machines and Regular Expressions. A BNF grammar is written as a set of recursive productions $A \rightarrow X_1 \dots X_n$, where A is some nonterminal symbol that expands to the sequence of terminals and nonterminals $X_1 \dots X_n$. Regular expressions define languages based on the operators *concatenation*, *union*, and *kleene-closure*. A thorough coverage of these language representations is given in [11].

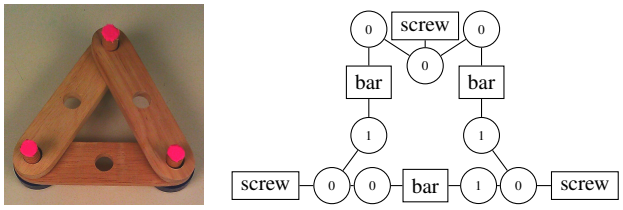
B. Hierarchical Decomposition of Tasks and Symbols

While formal language defines tokens as atomic symbols, these tokens are in fact abstractions of underlying phenomena. Consider the tokens of natural language: words may exist as vibrations in air, ink on paper, or magnetic transitions on a metal disk. Yet, all these representations define the same symbol. In formal grammars, this hierarchy is explicit through the relationship between nonterminal and terminal symbols. Terminal symbols are atomic. Nonterminals represent a set of strings of symbols, in essence a language of their own. Whenever it is necessary to deepen the abstraction for some terminal symbol, α, we can convert α to a nonterminal and define a new set of strings that α may expand to. We have used this approach for the manual construction of MG since it facilitates hierarchical task decomposition. For automatic grammar generation, we can again use this hierarchy of symbols to translate the task-appropriate symbols from humans to robots even though at the atomic level, human and robot actions are quite different.



(a) Constr. Kit (b) Point Cloud (c) Segmentation/Clustering

Fig. 3. Experimental Setup and Kinect Data



(a) Assembly (b) Connection Graph

Fig. 4. Connection Graph for an object assembly.

C. Experimental Setup

Our experimental setup consists of an assembly kit of wooden pieces, a Kinect RGBD camera, and a simulated Schunk LWA3 7-DOF robot arm with Schunk SDH 7-DOF dexterous hand. From a physical human demonstration, we infer the control policy for the task, and then implement that policy on the simulated Schunk robot. To capture the demonstration, the Kinect sensor is mounted above a table to observe a human performing the assembly task. The assembly pieces, Fig. 3(a), come from a Melissa & Doug brand wooden construction set. The only modification we make to the pieces is to attach a brightly colored dot to the end of screws. This simplifies distinguishing the screw from an attached bar in the Kinect image which has a limited resolution of 640 $\times$ 480 pixels. To illustrate our inference pipeline, we will show each of the steps required to build the simple assembly in Fig. 4(a). After inferring the policy from human demonstration, we simulate this policy with a kinematic model of the Schunk robot and then display the results, Fig. 1(c), with the Peekabot visualization tool.

D. Assembly Language

In an object assembly, the connections between objects form a graph. In the simple case, objects are the graph nodes and connections between objects are the edges. However, we can make this model more precise by accounting for the multiple connection points on objects. To do this, we introduce additional nodes for these connection points. Each object node has edges to each of its connection point nodes. Each connection in the assembly is represented by an edge between the two graph nodes for the connection. This type of graph is shown in Fig. 4.

From the representation in Fig. 4, we can produce an appropriate set of event symbols. The meaningful events of the assembly domain are when the connection graph is modified by connecting a new object to the assembly or when creating an additional connection between objects already part of the assembly. This event is represented as the tuple $o_i \times c_j \times o_k \times c_\ell$, where o_α is some object and c_β is the connection point on that object. In our figures, we write these symbols as "p.q-r→x.y-z" where p is the type of object o_i, q is the object number of o_i, $p.q$ is then o_i, r is c_j, x is the object type of o_k, y is the object number of o_k, $x.y$ is then o_k, and z is c_ℓ. A sequence of these connection symbols represents the full construction of the assembly.

The language over these assembly symbols is a *syntactic* model of the assembly task policy. Each string in the language is a plan to assemble the desired object. In this language, the task is abstracted to the level where we can transfer it from human to robot. By then combining this assembly language with the continuous *semantics* and lower level abstractions for our robot, we produce a Motion Grammar representing the hybrid dynamical control policy for the robot assembly task.

IV. Human Activity to Event String

The first step in our system for automatically generating motion grammars is converting a human demonstration of the desired task, assembling an object, into a string of the connection events that the human performs. Given multiple example strings, we can then infer the Motion Grammar for the robot.

A. Image Segmentation and Clustering

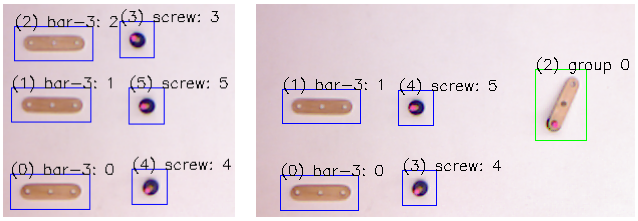
First, we segment the RGBD image to identify the clusters representing objects and partial assemblies. Since the table is the largest feature in the image, we can robustly fit a plane to the table using RANSAC. For large objects, the height of each point above the table segments the object from the table. However, because some of our objects are within the depth sensing error of the Kinect, we cannot use the depth information alone. Instead, we combine the depth and color information to perform the segmentation.

We perform segmentation by computing the Mahalanobis distance D_m of each point in the space of height above table z, hue h, and saturation s. This approach assumes a uniformly colored table, which is appropriate in our setup. To approximate mean and variance of h and s, we iteratively compute these values for points on the table according to z, then reject outliers. Then, with the resulting mean and variance for the space, we compute the Mahalanobis distance for each point in the image using $D_m = \sqrt{(x-\mu)E^{-1}(x-\mu)}$, where $x = [z\ h\ s]^T$, μ is the mean of x, and E is a weight matrix.

All points with both distance D_m above a threshold and with z above the table are taken as part of objects or assemblies. Then, these points are clustered according to Euclidean distance (Fig. 3(c)).

B. Object Recognition and Tracking

The next step is to recognize the objects that form each cluster and to track the objects across subsequent images. Since we have a small, fixed set of objects, we can recognize these objects by template matching in the RGB (sans D) image, Fig. 5(a). However, the tracking problem is complicated by two factors. First, many of our objects look identical so we cannot independently track them across subsequent frames. Second, because a human is moving the objects with his



(a) First Image (b) Second Image

Fig. 5. Recognized and labeled objects. Specific objects are tracked across subsequent images and objects combined into one cluster are grouped.

(a) Lines (b) Structure

Fig. 6. Inferred structure of the object assembly. Segmented points are iteratively clustered and line fit.

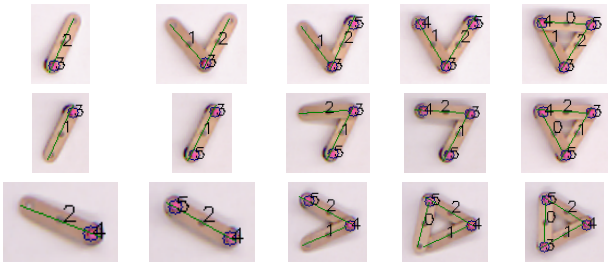


Fig. 7. Each row is a demonstration sequence for the example object.

or her hands, tracking is only relevant when objects are occluded. We handle these issues by assuming that most of the objects in frame are stationary, which is appropriate given that the human has only two hands to move objects. From this assumption, we convert object tracking to the Assignment Problem.

The Assignment Problem is an optimization problem that consists of finding the minimum cost matching between two sets, A and B, where the distances between members of A and B are known. Several subtasks in our inference pipeline are instances of this problem.

To convert object tracking to the assignment problem, we represent the point clusters in the initial frame as set A and in the subsequent frame as set B. The distance $d(a, b)$ is then the Euclidean distance between the centroids of the two clusters in each frame, $d(a, b) = \sqrt{x_a^T x_b}$. We can then solve this Assignment Problem using the Hungarian Algorithm, enabling us to track motion when multiple identical objects are moved without crossing.

By recognizing objects and tracking them across frames, we can determine when an object is added to an assembly, Fig. 5(b). In the event tuple, (o_i, c_j, o_k, c_ℓ), this gives us object o_i. The next step is to determine which other object o_i is connected to and how these two objects are connected.

C. Structure Recognition

To identify the precise connections between objects in an assembly, we first locate the individual objects within that assembly. Our system locates the bars and screws and then infers the connections between them.

We locate the screws in the assembly using template matching. The bars are located by iteratively fitting lines and clustering points to the closest lines. The resulting lines are shown in Fig. 6(a).

Now that we have the locations of a number of identical bars and screws, we track the specific object for each located element. By assuming that the elements of the assembly are mostly stationary between frames, this becomes another instance of the assignment problem. The first set A is the located objects from the previous frame and the second set B is the elements in the current frame. Distance between sets is Euclidean distance between object positions, $d = \sqrt{x^T x}$. Solving this assignment problem gives the specific object for each located element.

Having located the specific objects in the assembly, we now infer the connections between them. To do this, we assume that screws and bars can only connect at fixed locations on the bar, which is true for our construction set. Then we identify the connections with another assignment problem. The first set is the screws in the assembly. The second set is the connection points on the bars. Solving this gives us a connection for each screw and a single bar. To identify which screws connect multiple bars, we first identify the intersections between the lines for each bar. If that intersection goes through a screw, then that screw must connect the intersecting bars. Thus, we identify all connections between screws and bars in the assembly.

D. Symbol Generation

Given the connection graph at each frame, we can now abstract the demonstration to a sequence of symbols. Whenever the connection graph changes between subsequent frames, we add a new symbol representing that change to the sequence. Since our assemblies contain many identical objects, we also renumber the objects in the order they are added to the assembly. Some assembly strings are shown in Fig. 8. Thus, we abstract the human demonstration of assembly construction to a sequence of object connections which we use to infer a motion grammar for the robot to repeat the task.

V. Event Strings to Robot Grammar

Now that we have reduced the human demonstrations to an initial symbolic abstraction, we can transform this abstraction into a controller for the robot. First, we use the example strings to infer a syntactic model of the assembly task. Then, we combine the syntax of this assembly language with the

"1.0-0→0.0-0"	"1.0-0→0.1-0"	"1.1-0→0.0-1"	"1.2-0→0.1-1"	"1.1-0→0.2-0"	"1.2-0→0.2-1"
"1.0-0→0.0-0"	"1.1-0→0.0-1"	"1.2-0→0.1-0"	"1.0-0→0.1-1"	"1.1-0→0.2-0"	"1.2-0→0.2-1"
"1.0-0→0.0-0"	"1.1-0→0.0-1"	"1.0-0→0.1-0"	"1.1-0→0.2-0"	"1.2-0→0.1-1"	"1.2-0→0.2-1"

Fig. 8. Generated strings from demonstrations, indicating the sequence of object connections. A connection between screw i and bar k at bar location ℓ is "1.i-0→ 0.k-ℓ."

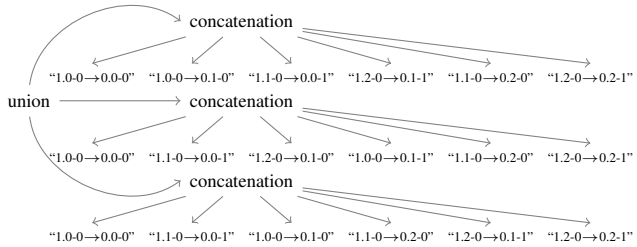


Fig. 9. Regular Expression parse tree for Assembly Task.

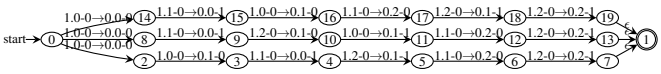


Fig. 10. Inferred NFA for Assembly Task. Language symbols are on edges; state labels are arbitrary.

semantic model of our robot to produce an MG for the demonstrated task.

A. Strings to Regular Expression

First, we convert the set of demonstration strings S to a regular expression R. This is directly accomplished by taking the union over all demonstration strings S. Thus, $R = \bigcup_{\sigma \in S} \sigma$. The language of this regular expression $L(R)$ is now a syntactic abstraction of all given demonstrations. For our example assembly, we transform the strings in Fig. 8 to the regular expression in Fig. 9.

B. Regular Expression to Nondeterministic Finite Automaton

Next, we convert the regular expression R to a Nondeterministic Finite Automaton (NFA) N using the McNaughton-Yamada-Thompson (MYT) algorithm [28, p159]. Because Regular Expressions and NFA are equivalent representations, this transformation is always possible. The MYT algorithm recursively walks the parse tree for the regular expression, producing an NFA which represents the same language. The resulting NFA for our example is shown in Fig. 10. Note that this NFA follows the conventional form of language symbols on edges and arbitrary state labels [11], [28].

C. Nondeterministic Finite Automaton to Minimum Deterministic Finite Automaton

We now convert the assembly NFA to a minimum-state DFA using Brzozowski's algorithm [29]. Because NFA and DFA are equivalent representations, this transformation is always possible. Brzozowski's algorithm produces a minimum state DFA by reversing all connections in the FA and converting the result to a DFA, then repeating that procedure once more. The resulting DFA for assembly is shown in Fig. 11.

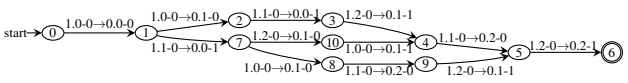


Fig. 11. Minimum State DFA for Assembly Task.

Production	Semantic Rules
$\langle \mathrm{T}_1 \rangle \rightarrow \lfloor \mathrm{t}_0 \rfloor \langle \mathrm{T}_2 \rangle$	$x_r = x_0 + \frac{1}{2}\ddot{x}_m t^2, \dot{x}_r = t\ddot{x}_m$
$\langle \mathrm{T}_2 \rangle \rightarrow \lfloor \mathrm{t}_1 \rfloor \langle \mathrm{T}_3 \rangle$	$x_r = x_0 + \frac{1}{2}\ddot{x}_m t_1^2 + \dot{x}_m(t - t_1), \dot{x}_r = \dot{x}_m$
$\langle \mathrm{T}_3 \rangle \rightarrow \lfloor \mathrm{t}_2 \rfloor \langle \mathrm{T}_4 \rangle$	$x_r = x_n - \frac{1}{2}\ddot{x}_m(t_n - t)^2, \dot{x}_r = \dot{x}_m + \ddot{x}_m(t_2 - t)$
$\langle \mathrm{T}_4 \rangle \rightarrow \lfloor \mathrm{t}_3 \rfloor$	$x_r = x_n, \dot{x}_r = 0$

Fig. 12. Syntax-Directed Definition for trapezoidal velocity profiles. For each trajectory stage T_i, the input $\dot{q}_r$ is computed according to (2).

VI. Simulation of Assembly

Finally, we combine the inferred syntactic abstraction of the assembly DFA with the semantic model of the robot to produce a control policy for object assembly. Then in simulation, we apply this control policy to reenact the task.

A. Kinematic Model

For the continuous dynamics of manipulation, we use a kinematic model of the robot. This model allows movement of the end-effector in Cartesian coordinates using the Jacobian pseudoinverse. To operate robustly near singularities, we used the Damped Least Squares method, [30]. The Schunk LWA3 has a redundant DOF, so to avoid both joint limits and numerical drift, we use a null-space projection towards the zero position of each joint. Our workspace control law is then,

$$J^\dagger = J^T(JJ^T + \lambda I)^{-1} \tag{1}$$

$$\dot{q}_r = J^\dagger\left(\dot{x}_r - K_x(x_r - x)\right) - \left(J^\dagger J - I\right)K_q q \tag{2}$$

where J is the Jacobian at the current configuration, λ is the damping parameter, q is the current configuration, x is the current Cartesian position, x_r is the reference Cartesian position, $\dot{x}_r$ is the reference Cartesian velocity, K_x and K_q are gain matrices, and $\dot{q}_r$ is the input velocity given to the robot. Orientation error between x and x_r is computed using difference of rotation vectors [31]. Thus, (2) allows us to track a Cartesian trajectory.

We compute the Cartesian trajectories using a trapezoidal velocity profile [31]. This is defined by the grammar in Fig. 12. This grammar shows how the reference positions are computed during each stage of the velocity profile. By combining (2) and Fig. 12, we have a semantic model permitting the robot to reach a desired Cartesian position for manipulation.

B. Manipulation Grammar

Next, we employ the grammar of Fig. 12 to hierarchically decompose the connection symbols from Fig. 11. This grammar, shown in Fig. 13, will expand the connection symbols $\langle \mathrm{o_i}, \mathrm{c_j}, \mathrm{o_k}, \mathrm{c}_\ell \rangle$ to sequences of robot trajectories necessary to perform the connection.

Note that to implement the assembly task, we must satisfy the geometric constraints in addition to the ordering constraints expressed by the DFA. One could naïvely handle these geometric constraints by initially placing objects arbitrarily and then later repositioning – or dragging – the object to satisfy the constraint. However, if we account for the geometry in our language, we can minimize this repositioning. Thus, we consider distances between pairwise connections in our language as follows. Given two symbols (o_i, c_j, o_k, c_ℓ) and (o_m, c_n, o_k, c_p), we observe that o_i and o_m are both connected to o_k and thus their positions are

$$
\begin{array}{rcl}
\langle \mathrm{o_i}, \mathrm{c_j}, \mathrm{o_k}, \mathrm{c_\ell} \rangle & \rightarrow & \lfloor \neg \mathrm{placed(o_i)} \rfloor \, \langle \mathrm{P(x_i, x_{ws})} \rangle \langle \mathrm{o_i}, \mathrm{c_j}, \mathrm{o_k}, \mathrm{c_\ell} \rangle \\
 & | & \lfloor \mathrm{placed(o_i)} \rfloor \, \langle \mathrm{P(x_k, x_i)} \rangle \\
\langle \mathrm{P} \rangle & \rightarrow & \langle \mathrm{pick} \rangle \langle \mathrm{place} \rangle \\
\langle \mathrm{pick} \rangle & \rightarrow & \langle \mathrm{move} \rangle \langle \mathrm{grasp} \rangle \\
\langle \mathrm{place} \rangle & \rightarrow & \langle \mathrm{move} \rangle \langle \mathrm{ungrasp} \rangle \\
\langle \mathrm{move} \rangle & \rightarrow & \langle \mathrm{T_1} \rangle \, \lfloor |\mathrm{x} - \mathrm{x_r}| < \epsilon \rfloor \\
\langle \mathrm{grasp} \rangle & \rightarrow & \{\mathrm{close}\} \, \lfloor |\mathrm{x} - \mathrm{x_r}| < \epsilon \rfloor \\
\langle \mathrm{ungrasp} \rangle & \rightarrow & \{\mathrm{open}\} \, \lfloor |\mathrm{x} - \mathrm{x_r}| < \epsilon \rfloor
\end{array}
$$

Fig. 13. Pick and Place Grammar for Schunk LWA3 and SDH.

constrained by the distance between c_ℓ and c_p, given as $x_j - x_n = x_\ell - x_p$. When o_i is already placed, we select an x_n to satisfy this constraint: $x_n = (x_p - x_\ell) - x_j$.

Now, we expand the grammar of Fig. 13 to make the connection. First, if object o_n has not been placed, we place it at position x_{ws} calculated according to the constraint. Then, we pick o_k and place it. The picking and placing follow the trajectories of Fig. 12, and the robots grasps by pinching the object between two fingers of the SDH. Through the combination of this manipulation grammar and the inferred assembly automaton of Fig. 11, the robot reenacts the human demonstration, shown in both Fig. 1(c) and the video accompanying this paper.

VII. Conclusion

We demonstrate the automatic transfer of an assembly task from a human to a robot. This linguistic inference pipeline discretizes the human demonstration into a sequence of semantically relevant object connection actions. Then, it infers a grammar from a set of those demonstrations which represents a syntactic abstraction of the task. Finally, we combine the syntax of the task language with the robot semantics and show in simulation the robot performing the demonstrated assembly task. This method simplifies the construction of formal models for robot verification and control, easing the way for building safer and more reliable robots.

There are many avenues to build upon this work. We will transfer the inferred grammar to a physical robot to demonstrate the practical utility of this method beyond prior applications of the Motion Grammar. In addition, we will explore further tools for grammatical inference including negative examples, inference of repetitions, and queries to the instructor, in order to infer more complicated task languages for larger assemblies. Probabilistic inference methods could also help this system tolerate errors in the recognition stages. Through this ongoing work, we hope to make it easy and efficient to develop safe and reliable robots.

Acknowledgments

Our thanks to Ana Huamán and John Turgeson for their prior development of the 3D models for the Schunk robot and wooden bar, respectively, used in our simulation.

References

[1] N. Dantam, P. Kolhe, and M. Stilman. The motion grammar for physical human-robot games. In *ICRA*. IEEE, 2011.

[2] N. Dantam and M. Stilman. The motion grammar: Linguistic planning and control. In *RSS*. IEEE, 2011.

[3] N. Dantam and M. Stilman. The motion grammar calculus for context-free hybrid systems. In *ACC*. IEEE, 2012.

[4] C.G. Cassandras and Stéphane Lafortune. *Introduction to Discrete-Event Systems*. Springer, 2nd edition, 2008.

[5] D. Hristu-Varsakelis and W.S. Levine, editors. *Handbook of Networked and Embedded Control Systems*. Birkhauser, 2005.

[6] J. Lygeros, K.H. Johansson, S.N. Simic, J. Zhang, and S.S. Sastry. Dynamical properties of hybrid automata. *Trans. on Automatic Control*, 48(1):2–17, 2003.

[7] T.A. Henzinger. The theory of hybrid automata. In *Logic in Computer Science*, pages 278–292. IEEE, 1996.

[8] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. *Hybrid systems*, pages 209–229, 1993.

[9] N. Dantam, M. Stilman, and I. Essa. Algorithms for linguistic robot policy inference from demonstration of assembly tasks. Technical Report GT-GOLEM-2012-002, College of Computing, Georgia Institute of Technology, 2012.

[10] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *Analysis and Optimization of Systems*, 25(1):206–230, January 1987.

[11] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, Reading, MA, 1979.

[12] R. Brooks. A robust layered control system for a mobile robot. *IEEE journal of robotics and automation*, 2(1):14–23, 1986.

[13] T. Lozano-Pérez and R.A. Brooks. An approach to automatic robot programming. Technical Report A.I. Memo 842, Massachusetts Intitute of Technology, 1985.

[14] E. Klavins. A language for modeling and programming cooperative control systems. In *ICRA*, volume 4, pages 3403–3410. IEEE; 1999, 2004.

[15] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. on Robotics*, 25(6):1370–1381, 2009.

[16] K. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, 1981.

[17] Y.A. Ivanov and A.F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[18] D. Minnen, I. Essa, and T. Starner. Expectation grammars: Leveraging high-level expectations for activity recognition. In *Computer Vision and Pattern Recognition*, volume 2, pages II–626. IEEE, 2003.

[19] D. Moore and I. Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *National Conf. on Artificial Intelligence*, pages 770–776. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[20] Colin de la Higuera. *Grammatical Inference*. Cambridge University Press, 2010.

[21] B.D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[22] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. *Handbook of Robotics Chapter 59: Robot Programming by Demonstration*. Springer, 2007.

[23] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2003.

[24] C. Chao, M. Cakmak, and A.L. Thomaz. Towards grounding concepts for transfer in goal learning from demonstration. In *Intl. Conf. on Development and Learning*, 2011.

[25] S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *ROMAN*, pages 358–363. IEEE, 2006.

[26] L.S.H. de Mello and S. Lee. *Computer-aided mechanical assembly planning*, volume 148. Springer, 1991.

[27] R.E. Jones and R.H. Wilson. A survey of constraints in automated assembly planning. In *ICRA*, volume 2, pages 1525–1532. IEEE, 1996.

[28] A. Aho, M. Lam, R. Sethi, and J. Ullman. *Compilers: Principles, Techniques, & Tools*. Pearson, 2nd edition, 2007.

[29] J.A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical theory of Automata*, 12:529–561, 1962.

[30] Y. Nakamura and H. Hanafusa. Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, pages 163–171, 1986.

[31] J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson, 3rd edition, 2005.