A Relational Representation for Generalized Knowledge in Robotic Tasks

Technical Report 04-101

Stephen Hart, Roderic Grupen, David Jensen
Computer Science Department
140 Governors Drive
University of Massachusetts
Amherst, MA 01003-4601
{shart, grupen, jensen}@cs.umass.edu

December 14, 2004

Abstract

In this paper, a novel representation is proposed in which experience is summarized by a wealth of control and perception primitives that can be *mined* to learn combinations of which features are most predictive of task success. Exploiting the inherent relational structure of these primitives and the dependencies between them presents a powerful and widely-applicable new approach in the robotics community. These dependencies are represented as links in a relational dependency network (RDN), and capture information about how a robot's actions and observations affect each other when used together in the *full context of the task*. For example, a RDN trained as an expert to "pick up" things will represent the best way to reach to an object, knowing that it plans on grasping that object later. Such experts provide information which might not be obvious to a programmer ahead of time, and can be consulted to allow the robot to achieve higher levels of task performance. Furthermore, it seems possible that new, more complex RDNs could be trained by learning the dependencies between existing RDNs. As a result, this paper proposes a hierarchical way of organizing complex behaviors in a principled way.

1 Introduction

In order to complete complex tasks in real-world environments, robots require a method of extracting relevant and domain-specific information from multi-modal sensorimotor data. At the same time, a robot must be able to generalize from past experience, using potentially limited current information to determine which actions will lead to the highest probability

of achieving goals. For example, a robot wishing to grasp an object can refer to knowledge learned from prior experience as well as available visual and tactile information. Furthermore, in the absence of good visual information, the robot should still be able to achieve good performance results. In order to behave in such a manner, a robot must have knowledge about how actions affect one another so that it can devise contingency plans if existing plans fail. Traditionally, robotic tasks have been addressed by decomposing them into simpler subtasks that are tackled separately. In general, however, each subtask, along with observations made during execution, may influence or affect other subtasks.

This work proposes a methodology for representing robotic tasks in such a way that, with sufficient training, a robot can develop knowledge about how to perform subtasks better in the context of the global task. Relational learning techniques are used to determine the statistical dependencies among features observed in training data, resulting in schematic task experts that can be used as a knowledge base. Mining the training data is useful for finding the most salient features of the task from the robot's perspective, which may not be obvious to a programmer ahead of time. This approach allows the robot to discover which information is the most semantically meaningful to itself. Exploiting this knowledge will lead to better task performance in the future.

Before moving on, let us discuss the requirements for generalized task experts. First, it is important that a robot should consider all of its experience at some task, successful or not. This experience can be gained by tele-operation, or guided execution by an operator. Second, expertise in a task should capture knowledge about how to perform that task in a variety of settings. For example, an expert should understand how to hammer in a nail by using a hammer or by using a shoe. Similarly, a "grasping" expert should know how to employ two fingers to grasp a small object as well as how to employ two hands to grasp a large object. Third, the experts should be able to recommend to the robot how to perform its task in the presence or absence of a varying amount perceptual information. For example, an expert for identifying individuals should be able make its best guess when using visual information, audio information, or both either together or alone. Fourth, task experts should be able to autonomously determine the statistical dependencies between observable features that can best indicate the success or failure of its task. For example, an expert for "picking up" objects should realize there is a dependency between the size of object to be picked up and the resources necessary to do so. If the object is large, the robot might need to use two hands. If it is small, one hand may suffice.

To meet the above criteria, this work defines generalized experts as *schemas* that can be represented in a probabilistic relational model (PRM[1]) [4, 6]. Each training example for a task is used as supporting evidence for learning the statistical correlations that can be used to address the above requirements. In the next section, schema theory and relational models are introduced. Section 3 will define precisely the how to represent robotic tasks with a relational model. Section 4 will show an experiment where a relational model (specifically, a relational dependency network [10]) is learned from actual experience of a robot picking up objects. Finally, Section 5 discusses conclusions and future work.

[1]Note that the acronym PRM in the robotics literature stands for "Probabilistic Road Map." In this paper, however, the acronym will represent "Probabilistic Relational Model."

2 Related Work

Previous work has shown that finding the statistical dependencies between various forms of data can lead to better task performance. Most notably, Piater [12] and Coelho *et al.* [3] describe how to learn the precise visual features of an object that can predict pre-grasp hand postures. Using these pre-grasp postures, the robot can increase its efficiency by minimizing the amount of pure haptic probing that is necessary to ensure a grasp. This paper expands upon this work in two ways. First, it provides a non-task specific method for which an *arbitrary number* of multi-modal dependencies can be used to recommend behavior. Second, these dependencies are *learned* from a sampled subset of all available features, not specified explicitly as in [3]. Before moving on, we will ground our approach in developmental psychology, as well as explain the tools we will use to learn this structure.

2.1 Schema Theory

Acquiring complex behaviors through the composition of simpler behaviors has been studied in both developmental psychology [5, 11] and computation neuroscience [1]. The notion of a *schema* has been used to represent knowledge, and is analogous to Minsky's society of agents [9]. Fundamentally, schemata represent skills that can be organized (with some tuning) into more complex skills. Initially, infants start with simple skills (such as reflexes or other innate behavior), but soon develop more complex and learned behavior through a method of composition. Piaget describes a process by which this development occurs based on a definition of adaptation in which there are two complementary components. *Accomodation* represents the individual's tendency to change in response to environmental demands, creating new schema from old when necessary. *Assimilation* represents the individual's tendency to deal with an environmental event in terms of existing structures, thereby reusing existing schema. Piaget has proposed a hierarchical theory of skill acquisition to organize existing skills into more complex skills through experience with the environment.

In the computational neuroscience literature, Arbib [1] characterizes schema as abstract, functional units of activity which generalize similar experiences while mapping those experiences to suitable goal-directed actions. For example, the schema for hammering in a nail allows a hammer, a shoe, or a rock to be used to do the hammering. A schema representing a seat considers the functional use of that seat, whether the actual support structure is a chair, a rock, or couch. Arbib has categorized schema into *perceptual schema* and *motor schema*. Perceptual schema represent concepts such as objects and their associated characteristics. For example, a perceptual schema representing a "ball" will have associated attributes such as "color," "size," and "velocity." Motor schema provide means of achieving goals, and can be characterized by the "degree of readiness" or "activity level". For example, there might be a motor schema for "throw" which has attributes that are temporally situated and related to the state of throwing. Motor and perceptual schema can be composed into coordinated control programs, such as "throwing a ball," that are robust, fault-tolerant, experts for performing a particular task in a variety of settings.

Although the idea that there are such cognitive entities as perceptual schema is a matter of philosophical debate, the notion that there are observable features in the world that are

necessary to attend to in order to achieve a task successfully is less subjective and is central to this paper. Also, although schema theory provides a means of describing the organization of human behavior, it makes no claim as to any conscious knowledge of these schema in the individual.

2.2 Relational Datasets

Most standard graphical models used today, such as Bayesian or Markov networks, assume that the relevant training data are "flat," or non-relational, and can be represented by a standard two-dimensional database. In other words, every training sample has the exact same set of observable variables. In general, this assumption is false, and transforming data in an attempt to make it true has forced complex relational structure to be lost in many systems. For example, research papers may have any number of authors, but how to correctly represent this fact in a "flat" database is not obvious. Robots can use various different sensors to perform the same task. *Probabilistic Relational Models* [4, 6], on the other hand, are more expressive models which exploit the relational structure of data. Exploiting this relational structure may lead to a deeper understanding of the domain.

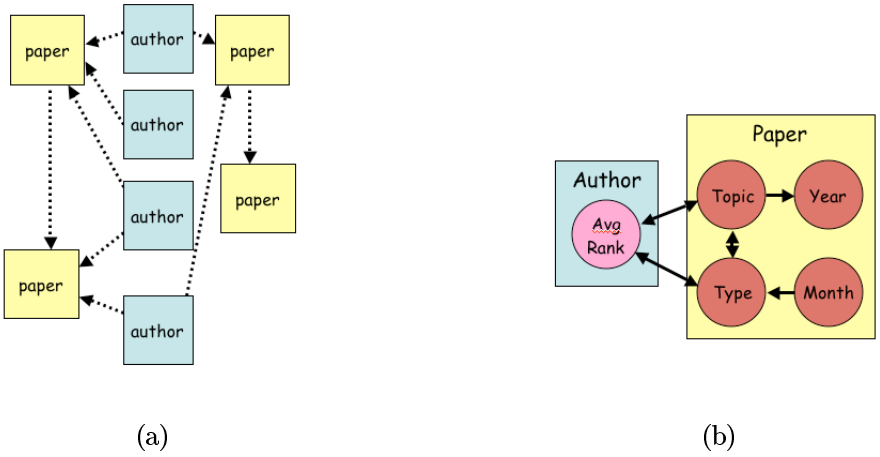
Non-relational data assumes that data instances are independently and identically distributed (i.i.d.). In other words, all instances have the same structure and knowing something about one instance tells you nothing about another. In general, of course, this is not the case. In the robotics domain, there might be many different ways to perform a task (such as picking up objects, sitting down, assembling a structure, or following a person), and each instance might influence a different set of random variables. These instances would not be identically distributed. Also, two concurrent instantiations of the same subtask might influence each other (consider an assembly task that requires picking up multiple objects, transferring objects between hands, temporarily putting objects down, etc.), and are therefore not independently distributed. In these situations, it is difficult to design (or learn) a single model that captures all contingencies, as well as to collect enough general experience that could be used to train one model. One recent approach for overcoming these issues has been the development and use of relational dependency networks (RDNs) [10, 8]. RDNs provide an intuitive graphical representation that facilitates knowledge discovery, encoding properties of the data in the structure of a graph. Quantitative properties of the data are specified by the parameters of the conditional probability distribution of each variable.

2.3 RDN Representation

Relational Dependency Networks (RDNs) [10] are a type of PRM and thus represent joint probability distributions over relational datasets. There are three types of graphs associated with models of the relational data: the *data graph* G_D, the *model graph* G_M, and the *inference graph* G_I.

The data graph G_D, Figure 1, represents objects in the data as nodes (e.g. *authors*, *papers*, *controllers*, $features$), and the directed (dashed) edges between the nodes represent explicit relations (e.g. *author-of*, *cites*, *uses*). Each object has a particular type, and a number of attributes and links associated with that type (papers have topics and citations,

controllers have state information and resource bindings. The dependencies between the attributes of nodes (either within or across particular object types) are modeled in the model graph G_M, and are represented by solid-line arrows. While the data graph explicitly characterizes the dataset, the model graph is a *learned* feature of the dataset, revealing information that is not necessarily apparent ahead of time. Figure 1 shows a fragment of the data graph for the research paper domain, and the corresponding learned model graph [10]. In Figure 1a, we can see that there are a varying amount of authors for each paper, and thus the data are not identically distributed. Also, different papers share authors, and so the data are not independently distributed either. In Figure 1b, we see the learned dependencies. For example, the "topic" of a paper is related to the "year" that the paper was published.



(a) (b)

Figure 1: **Research Paper (a) Data Graph (b) Model Graph**

The inference graph is the *rolled out* instantiation of the model and data graphs, and can be used to make statistical assertions and inferences about the data. Figure 2 shows the inference graph for the research paper domain.

Procedures for *collective inference* in RDNs make simultaneous statistical judgments about the same variables for a set of related data instances [8]. For example, collective inference could be used to simultaneously classify a set of hyper-linked documents as having the same topic. In the execution of robotic tasks, collective inference can be used to make better assessments of how that task can be performed. For example, one could ask for the probability of success of all subtasks, based on an assignment of a particular set of resource bindings.

3 Relational Dependency Networks in Robotic Tasks

In this section, we present a way of representing robotic tasks in terms of RDNs that will facilitate the use of relational data mining techniques. Using this representation, the robot can

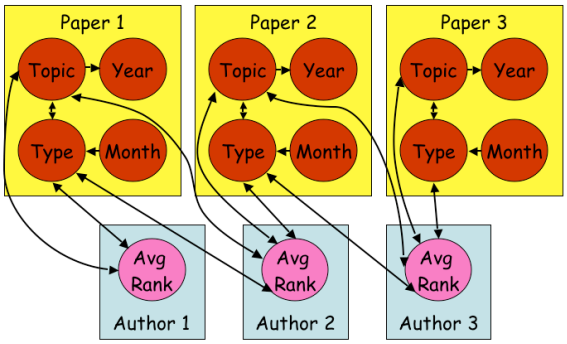


Figure 2: **Research Paper Inference Graph**

learn that, when executing a particular controller, there are statistically correlated features contained in other available sensory information. If such characteristics are not present in a future execution of the task, then the robot could make assertions about how probable (or improbable) it is to successfully complete that task. Ultimately, this could lead to the robot learning how to employ specific controllers designed to make such characteristics change in the desirable way, and thus maximizing the probability of successful achievement of its goals. This will be future research, beyond the scope of this work.

3.1 Gathering the Data

To gather useful information from task experience, the robot must make observations of sensory, kinematic, proprioceptive, and control information in the course of behavior derived from a nominal schema. One possible way to gather this information is to have a tele-operator perform a task one or more times, possibly multiple ways. Another possibility is to autonomously employ a specific sequence of controllers with a particular binding of resources (i.e. a visual servo controller for a peg-in-hole task). In both cases, the robot gathers all available multi-modal information, including controller activity. Learning can then be a post-processing phase in which different features from the captured data are correlated.

3.2 Representing the Data

In the context of robotic tasks, a proper RDN representation of objects and attributes needs to be determined before learning algorithms can be applied. Here we will use the notion of perceptual and motor schema from Section 2 to recommend a representation. We can define *Controller* objects (conceptually similar to a motor schema), that have both a "Convergence State" attribute and an attribute defining a parameterization of available "Resources" that are bound to it. This representation also allows for a compact and discrete representation of controller activity, which is a continuous time process. Employing this discrete-event dynamic system approach over controllers has also been used in the control basis work of Huber *et al.* [7].

We can also define *Perception* objects that have feature attributes associated with them.

These attributes may be coarse, such as "Scale," "Size," "Color," or "Velocity," or may be more fine grained and closer to the raw sensory information, such as multivariate filter responses. Note that if the coarse approach is taken, a separate *Perception* object must be defined for all physical objects relevant to the robot for the task. If the finer grained approach is taken, we only need one PERCEPTION object, but with as many filter attributes associated with it as we make available, despite how many physical objects might be present. Figure 3 shows a graphical description of one type of controller object (*Controller X*) and a PERCEPTION object with coarsely defined attributes.

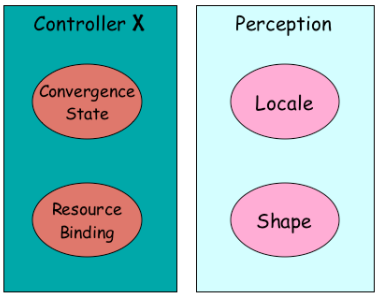


Figure 3: **Objects and Attributes**

There are obvious relations that can be made between these objects when we construct the data graph. A controller uses particular sensory information or may involve the manipulation of some physical object. Also, there are many tasks where certain controllers need to complete before others. We can add directed edges in our data graph reflecting both which physical objects or sensory streams controllers depend on, and also which controllers need to complete before others (do REACH before GRASP, for instance). By following this procedure only a minimal amount of domain knowledge need be employed.

There is, however, no way of representing the complex statistical dependencies that might be present between features in the context of the entire task. This is, in fact, the knowledge we would like our relational learner to discover. For example, which resources should be used during reach and grasp based on the visual features discerned during localize? Or, which resources should be employed during grasp given the parameterization and success of reach? Analyzing the available data will lead to a more meaningful understanding of the task to the robot, and can ultimately be exploited as a predictor of successful task completion.

The following section describes an example task that subscribes to the above representation, showing how such complex dependencies can be learned using training on a real robot both with and without supplemental synthetic data used to impose safety constraints.

4 Example Task - Localize-Reach-Grasp

In this section, we look more closely at a particular robotic behavioral schema, LOCALIZE-REACH-GRASP, by suggesting a data graph representation and show what type of model we might expect to learn by applying relational learning techniques.

4.1 Experimental Procedure

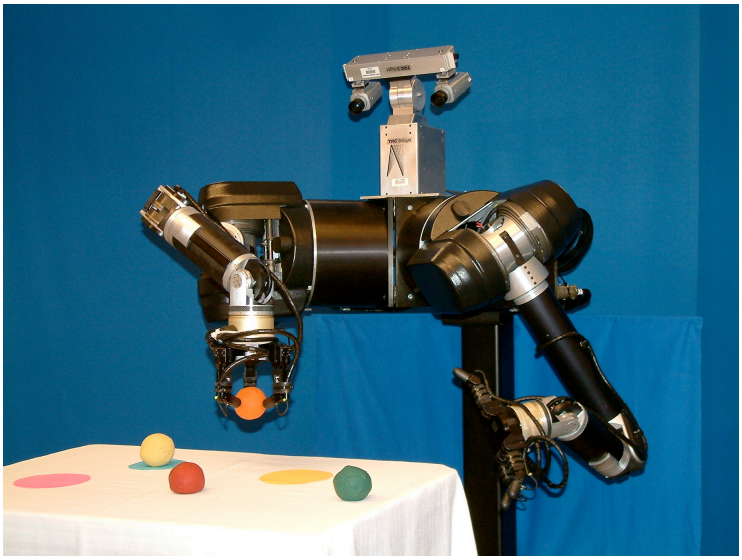


Figure 4: **Dexter - The UMass Bi-Manual Humanoid**

Fifty trials were performed in which one of two objects - either a tall, narrow box, or a small coffee can - were placed in front of the UMass humanoid robot Dexter (Figure 4). The objects were placed in one of three pre-defined locales: to Dexter's left, to Dexter's right, or directly in front of Dexter. These locales were defined by the intrinsic reachable workspace of each of Dexter's arms. Dexter is, of course, capable of reaching to the center locale with either hand. The robot was allowed to reach to either of these three locales either from the top or the side. The geometry of the objects were such that the coffee can is too short for Dexter to grasp from the side, and the tall box is too small to grasp from the top. While grasping, Dexter was allowed to use either 3 fingers or 2 fingers. In general, Dexter is more successful at grasping the box from the side with 2 fingers than 3, but capable of grasping the can from the top using either 2 or 3 fingers.

To represent this experiment in our RDN, we have chosen three controller objects: LOCALIZE, REACH, and GRASP, which are low level schema in their own right. Each object has an attribute pertaining to its "Convergence State," and also to its "Resource" binding. For LOCALIZE, the only available resource is Dexter's stereo head. In general, there could be many different camera pairs that could be used to localize an object. The available resources for the REACH object are the "Left Arm" and the "Right Arm." In addition to the resource attribute for the REACH object, there is also an "Orientation" attribute which specifies if either a "Top" or a "Side" reach is to be performed. The available resources for the GRASP object are the "Left Hand" and the "Right Hand," each with either "2 Fingers" or "3 Fingers." In this experiment, there is only one object at a time for the robot to grasp. We therefore can define a single PERCEPTION object with attributes for both "Locale," and "Shape."

This experiment has been designed such that we expect an RDN to notice that there are

statistical dependencies between the shape of the object, the locale of the object, and the reach and grasp resource bindings that are allocated to perform the task. For example, if the robot grasps the tall box with two fingers from the side it will be more likely to succeed than if it uses three fingers or grasps from the top. Similarly, if the can is located on the left side of the robot, it should use its left arm and left hand with either two or three fingers, reaching from the top. Conversely, we would not expect a high probability of success if the can is placed on the left side and the robot tries to reach with its right hand. Also, there should be a statistical dependency between the successful completion (convergence) of the grasp controller and the reach controller. If the reach fails, the grasp will not succeed.

The fifty trials that were performed were meant to capture experience grasping objects many ways and many times (repeats of trials). Physical experiments were, in general, not performed if they were not likely to work (for example, reaching with the left arm, but grasping with the right hand). These experiments were not performed because of a negligible chance of success and also to maintain safety to the robot. This does not mean, of course, that all attempts to grasp the objects were successful. Both the grasping and reaching do not work all the time, and it may, for example, be better to grasp the can with two fingers than with three. In response to the limited number of actual experiments on the hardware (50), a number of simulated experiments (276) were generated to represent cases that might be damaging to the robot or extremely unlikely to work. Training the RDN both with and without this synthetic data was performed, and the results are described in the next section. It should be noted that generating synthetic data might be one means of imposing safety constraints on the learned schema. If the robot has no chance of success in some configuration, it will not chose that configuration. Exploiting the full potential of safety constraints in this manner has been left for future work.

4.2 Constructing the Data Graph

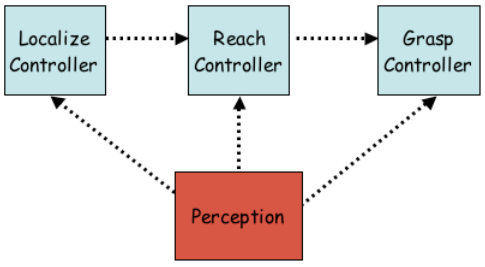


Figure 5: **Localize-Reach-Grasp Data Graph**

In Figure 5 we see the data graph using the object types described above. There are relations between the PERCEPTION object and the three controller objects. Each of the LOCALIZE, REACH, and GRASP controllers depend on the physical object. Also, in order for the task to complete successfully, LOCALIZE must come before REACH, and REACH must come before GRASP. From these obvious relations, we can add the dotted arrows in

Figure 5. In the following section we will discuss the model graphs that were learned using the experimental data with and without the synthetic data.

4.3 Learning the Model

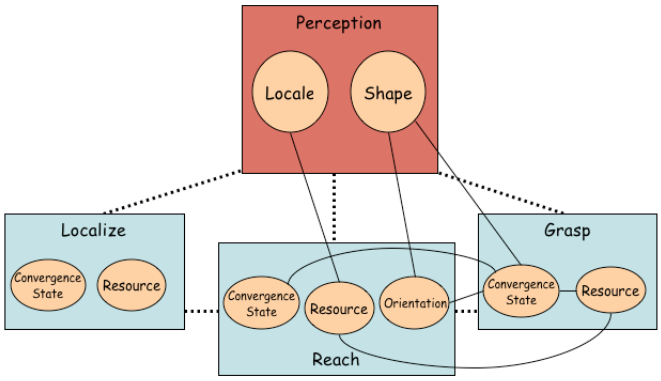


Figure 6: **Model Built with Experimental Data**

The experimental data gathered while performing 50 trials with Dexter was used to train the RDN shown in Figure 6 using the open-source system for relational knowledge discovery PROXIMITY [13] developed at the Knowledge Discovery Laboratory at the University of Massachusetts Amherst. We can see quite clearly that many of the main relationships we would expect, are manifested as statistical dependencies. For example, the "Convergence State" of REACH and the "Convergence State" of GRASP are linked. Also the "Shape" in the *Percerption* object influences both the "Orientation" of REACH and the success of GRASP (remember that the tall box could only be reached to from the side, and the coffee can could only be reached to from the top). The "Resource" attributes of both GRASP and REACH are linked because the robot should be grasping with the same arm it uses to reach. The "Resource" attribute is linked to the "Locale" attribute, because where the object is placed influences which arm the robot should reach with. Note that no dependencies between the LOCALIZE attributes were learned. This is because there was only one resource available to Dexter to localize, its stereo camera head, and all localize attempts in the 50 trials were successful. In general, we would expect the localize attributes to be affected by things such as the locale of the object (if an object is too far over to a side for the robot to see it, for instance, we would expect localize to fail).

Figure 7 shows the learned dependencies for the network trained in PROXIMITY using the 276 trials of synthetic data, in addition to the 50 trials of experimental data. We can see that more dependencies are present than in Figure 6, which is not surprising given that the set of data represents a more complete set of training instances for the task. For example, "Locale" is now linked to the "Convergence State" of REACH. In all of the experimental data, the robot only reached to the correct locale. The synthetic data, however, captures data instances where the robot reaches to the wrong side, and thus fails at reaching (and

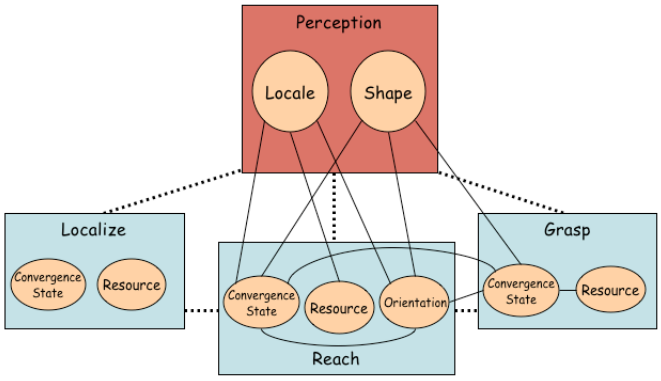


Figure 7: **Model Built with Synthetic and Experimental Data**

later at grasping). As a result, such a dependency shows up when both data sets are used to train the RDN.

4.4 Validation - Using the RDN

Ultimately, the true test of a model graph learned for a task, is how well it performs when used to recommend resources to perform that task again. In this section, we can show that the RDN learned from the combined set of experimental and synthetic data, can be used to construct a policy for the robot to pick up objects of the classes discussed.

Table 1 shows the probabilities for successfully grasping an object, examining just the different options the robot has for reach orientation and finger resource allocation. These probabilities were taken by examining the Relational Probability Trees (RPTs) learned by PROXIMITY for each attribute.

Shape	Orientation	Num Fingers	P(grasp success)
can	**top**	**2**	**62%**
can	top	3	46%
can	side	2	2%
can	side	3	2%
tall box	top	2	1%
tall box	top	3	1%
tall box	**side**	**2**	**90%**
tall box	side	3	55%

Table 1 - **Probabilities of Grasp Success for Possible Resource Allocations**

We can see the optimal policy in bold based on these probabilities. If the can is placed in front of the robot there is a slightly better (62% vs. 46%) chance of success if a two finger top grasp is used than a 3 finger top grasp. There is only a 2% chance of success if a side

grasp is used on the can. Similarly, there is very little chance, 1%, that the robot will be able to grasp the tall box from the top, but there is a a very high chance it will be able to grasp it from the side with 2 fingers.

The best recommend policy for which arm and which hand to used based on locale was the obvious for each case. For example, if the object was placed on the left of the robot, there was no chance of success if the robot tried to reach with its right arm, and vice versa. There was also a negligible distinction between which arm or hand to use if the object was placed in the center locale.

All of these results are consistent with the training data, and show that the learned RDN can, in fact, function as a domain expert for the localize-reach-grasp task.

5 Conclusions and Future Work

In this report a compact way of representing generalized task expertise has been proposed. The performed localize-reach-grasp experiment was used to show how the complex, but in this case intuitive dependencies can be learned using a Relational Dependency Network. For example, the networks shown in Figures 6 and 7 capture the knowledge that if a tall object is placed on the robot's left, the network recommends using the two fingers on its left hand to grasp that object, reaching to it from the side.

In general, more subtle dependencies between finer grained features in the data, such as multivariate Gaussian responses over a sensory stream, can influence how a task should be performed. The proposed framework can handle such complexity for an arbitrary number of observed features by including the parameterization of these Gaussian models in the set of attributes of the PERCEPTION object. For efficiency reasons, however, a robot may only wish to use a sampled subset of these features to decide on which actions it can take that will lead to task success. During task execution if this subset proves insufficient to guarantee a minimal probability of success, the robot can re-sample from the full set of features acquired during the training phase and learn a new RDN in the hope that it will discover more salient relations between other relevant features it did not initially consider. Both using finer grained features and sampling from them will be the subject of future experiments.

Another way to add to the wealth of information contained in the attributes of the PERCEPTION and *Controller* objects might be to include parametric models over the observed dynamics of both controller response. For example, it is known that certain classes of objects prescribe to certain controller dynamics [2]. Capturing this information in the object attributes might allow the robot to observe the behavior of its controllers, potentially modifying its actions if will lead to a better chance of success.

5.1 Exploiting the Full Power or RDNs

Although the LOCALIZE-REACH-GRASP example presented in Section 4 is demonstrative of how RDNs can be used to learn complex dependencies in a task, this example does not exploit the full power of RDNs. In particular, the above training data meets the identically distributed criteria, and could have thus been represented by a simple Bayesian network. There are many robotic situations where representing the experience in such a "flat" way

is not sufficient, and in these cases RDNs are a logical alternative. Consider the task of tightening a lug-nut. Should a robot first pick up a drill, guarantee it can actuate the drill, and then tighten down the lug-nut? Should it just use its fingers to tighten it down, maybe stabilizing the supporting structure it is attached to with its other hand? Should it pick up a wrench using a power grasp and use that? There is no single flat network structure that can handle all of these cases, but all fall under the same category of "Tightening a Lug-nut." Unlike Bayesian or Markov networks, RDNs can use all of this experience to train a single graphical model. Furthermore, the complex dependencies common to all examples (for example how force and visual information are related) would still be captured in this framework.

Similarly, the LOCALIZE-REACH-GRASP experiments used to train the RDN were independently distributed (the other part of the i.i.d. assumption), because no two trials influenced each other. This too, is not generally the case. If a robot was to pick up a series of objects, maybe in the context of building a more complex assembly, certain sub-instances of localize-reach-grasp might be dependent on each other, specifically if they use the same PERCEPTION objects. Again, RDNs can handle this situation, learning the dependencies between attributes, using all sub-instances of localize-reach-grasp to train the network. It follows that RDNs can be used to learn dependencies in a more complete set of situations than can Bayesian or Markov networks, and thus provide a powerful new tool for the robotics community.

5.2 Implications

Even without fully exploiting the full power of RDNs, the above representation, as can be seen by the experiment in Section 4, presents a powerful framework for learning the complex statistical dependencies necessary when generalizing a schematic behavior to address a variety of cases. This paper lays the groundwork for a system which provides the fundamental building blocks for a hierarchical representation of complex behaviors in robotic tasks. A RDN can be learned for each task, functioning as a knowledge expert schema for that task. Multiple schema can be organized into complex patterns to produce new schema, each time learning a new RDN that uses as its objects the component RDNs, with their parameterizations as their attributes. In this way, the robot can learn increasingly general knowledge about how to perform tasks best in the context of other tasks, a requirement that is necessary to act intelligently in real world situations.

References

[1] M. Arbib. *Schema Theory*, volume The Handbook of Brain Theory and Neural Computation. MIT Press, first edition, 1995.

[2] J. A. Coelho. *Multifingered Grasping: Grasp Reflexes and Control Context.* PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 2001.

[3] J. A. Coelho, J. H. Piater, and R. A. Grupen. Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robotics and Autonomous Systems Journal*, 37(2-3):195–219, November 2001.

[4] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference of Artificial Intelligence*, Stockholm, Sweden, August 1999.

[5] H.P. Ginsburg and S. Opper. *Piaget's Theory of Intellectual Development.* Prentice Hall Inc., Englewood Cliffs, N.J., 1988.

[6] D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research, March 2004.

[7] M. Huber and R.A. Grupen. A hybrid discrete dynamic systems approach to robot control. Technical Report 96-43, Department of Computer Science, University of Massachusetts Amherst, October 1996.

[8] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *KDD '04*, Seatle, Washington, 2004.

[9] Marvin Minsky. *The Society of Mind.* Simon & Schuster Inc., New York, 1986.

[10] J. Neville and D. Jensen. Dependency networks for relational data. In *Proceedings of The Fourth IEEE International Conference on Data Mining*, 2004.

[11] J. Piaget. *Biology and Knowledge.* Edinburgh University Press, Edinburgh, 1971.

[12] J. H. Piater. *Visual Feature Learning.* PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 2001.

[13] PROXIMITY. http://kdl.cs.umass.edu/proximity/index.html, Knowledge Discovery Laboratory, University of Massachusetts Amherst.