A Relational Representation for Procedural Task Knowledge

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

Abstract

This paper proposes a methodology for learning joint probability estimates regarding the effect of sensorimotor features on the predicated quality of desired behavior. These relationships can then be used to choose actions that will most likely produce success. *relational dependency networks* are used to learn statistical models of procedural task knowledge. An example task expert for picking up objects is learned through actual experience with a humanoid robot. We believe that this approach is widely applicable and has great potential to allow a robot to autonomously determine which features in the world are salient and should be used to recommend policy for action.

Keywords: robotics, knowledge representation, machine perception

1. Introduction

Generalized task knowledge can be decomposed into *declarative* and *procedural* components (Mandler 2004). The declarative structure captures abstract knowledge about the task; e.g., to pick up an object, we must first find the object, reach to it, and then grasp it. The procedural structure captures knowledge about how to instantiate the abstract policy in a particular setting; e.g., we must use our left hand to pick up the object and use an enveloping grasp. With such a decomposition, it is possible to represent task knowledge in a general, robust, and fault-tolerant way. The declarative structure of a task defines an abstract *schema* that can guide an agent's behavior in the world, while the procedural substrate decorates this abstract schema with resources based on environmental context. This paper examines how to learn procedural knowledge for a task when the declarative structure is known or assumed.

The notion of a schema as a fundamental unit of behavior has been studied in both developmental psychology (Ginsburg & Opper 1988) and computational neuroscience (Arbib 1995). Arbib characterizes schemata as abstract units of activity that generalize over similar experiences and map those experiences to goal-directed actions (Arbib 1995). Arbib designates two types of schemata: *motor* and *perceptual*. Perceptual schemata 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 schemata provide means of representing actions, such as *throw*, and can be characterized by state variables such as "degree of readiness" or "activity level." Motor and perceptual schemata can be composed into coordinated control programs, such as THROWOBJECT.

We believe that such distinctions of perception and action are misleading. Recent findings in neuropsychology (Gallese *et al.* 1996) suggest that perceptual information is more closely tied to the dynamics of interaction with the environment. This notion is also consistent with the concept of Gibsonian *affordances* (Gibson 1977); a perceptual feature is meaningful only if it facilitates an action. This work does not suggest separate motor and perceptual entities in the way described by Arbib, nor does it suggest having explicit (and distinct) perceptual schemata for separate objects in the world.

We propose using probabilistic models of relational data (Friedman *et al.* 1999) to learn the procedural structure of a task. In particular, we will use a relational model to find the statistical dependencies between sensorimotor variables and task success. Relational models are useful because they provide a framework for learning from experience in a variety of settings. Once the declarative structure of a task recommends an action, the model can use the observed sensorimotor variables to determine which resources should be employed. For example, a robot employing such a model can learn how to pick up many objects, configuring a policy on features it has observed—such as picking up a large heavy ball with two hands rather than one. In the remainder of this paper we provide a more complete presentation of how to design procedural knowledge experts using a particular class of relational models—*relational dependency networks* (RDNs) (Neville & Jensen 2003; 2004). Using this framework, we are able to learn policies to successfully pick up a class of objects from real robot data.

2. Related Work

There can be a large amount of environmental variation between separate executions of the same task. To react accordingly to such variations, it is important to provide a framework in which actions suppress disturbances in a reactive

manner. Huber (Huber 2000) provides such a framework in which parametric closed-loop controllers define the primitive organizational structure to behavior. In such a manner, continuous real-time controllers represent discrete *objectives*, which capture the declarative structure of a policy, while the execution of the policy (the procedural structure) is defined by a controller's resource (i.e., sensor and effector) bindings.

Previous work has shown that finding the statistical dependencies between various forms of data can lead to better task performance. Most notably, Piater (Piater 2001) and Coelho *et al*. (Coelho, Piater, & Grupen 2001) describe how to learn visual features that can predict pre-grasp hand postures. Using these pre-grasp postures, the robot can increase its efficiency by minimizing the amount of pure tactile probing that is necessary to ensure a grasp. Furthermore, Coelho (Coelho 2001) shows how the dynamic response of a grasp controller can potentially provide characteristic perceptual information about the type of object being grasped.

Other work has examined correlating sensorimotor information in humanoid robots. Natale *et al.* and Fitzpatrick *et al.* (Natale, Metta, & Sandini 2004; Fitzpatrick *et al.* 2003) learn to classify objects through interaction—both by pushing them and by grasping them. Kupiers *et al.* (Kupiers *et al.* 2005) uses a methodology that bootstraps knowledge from low-level sensorimotor primitives, but shows that performing even simple skills requires a large number of distinct learning algorithms. Work by Roy (Roy *et al.* 2002) allows a robot to develop a visually grounded grammar yet rely on predefined spatial relationships and objects. These approaches are task dependent and do not easily support generalization through a unified learning approach. This related work does, however, emphasize that useful categorization of sensorimotor features can lead to an appropriate course of action.

3. Relational Data

Traditional machine learning algorithms assume that all relevant training data contain the same set of observable (or unobservable) variables. These types of non-relational models require 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 the robotics domain, this is not the case. There are many different ways to perform a task, and each case can require a different set of actions. These instantiations of the task would not be identically distributed. Also, two instances of the same action employed during a task might influence each other (consider a task that requires grasping multiple objects that need to be assembled). Such instances are therefore not independently distributed. It seems natural, therefore, to use relational data representations to overcome these issues.

Relational learning techniques (Friedman *et al.* 1999) are prime candidates for learning procedural task knowledge because (1) different training episodes might exhibit varying structure, and (2) we are interested in determining statistical correlations between sensorimotor features and success in a task. By framing the problem in an appropriate relational way it is possible to find the most salient features of the task from the robot's perspective, which may not be obvious to a programmer ahead of time.

Relational Dependency Networks

Dependency networks approximate the joint distribution of the domain of variables with a set of conditional probability distributions (CPDs), which are learned independently. RDNs extend the concept of a dependency network to relational settings. Such a graphical model is advantageous in learning task expertise because we are most interested in how sensorimotor features influence each other and the success of intermediate task objectives. For example, with such a model we could hope to learn that the scale of an object will influence how a robot should reach to it, or how shape features will influence a grasp. RDNs have been used to learn models in diverse domains such citation analysis, the Internet Movie Data Base (IMDb), and genomics.

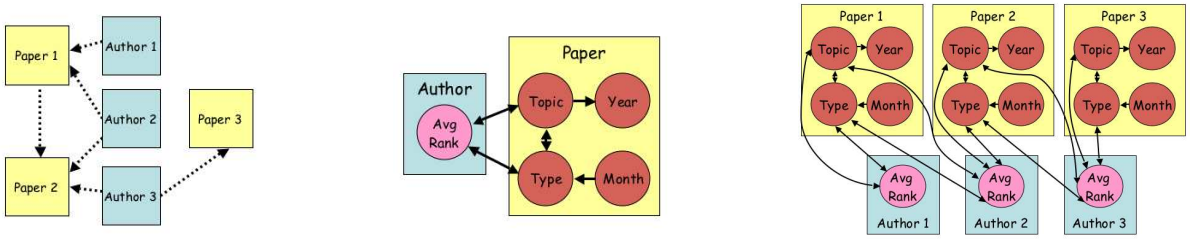
There are three types of graphs associated with models of relational data: the *data graph*, the *model graph*, and the *inference graph*. The data graph represents objects in the data as nodes (e.g., in standard database examples: $authors$, $papers$, $students$, $classes$). Edges between these nodes represent explicit relations (e.g., *author-of*, *cites*, *enrolled-in*). Each object has a particular type and a number of attributes and links associated with that type (*papers* have "topics" and "citations", *students* have "age" or "major"). The dependencies between the attributes of nodes (either within or across particular object types) are modeled in the model graph. 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. The inference graph is the *rolled out* instantiation of the model and data graphs; it can be used to make statistical assertions and inferences about the data. The inference graph overlays the dependencies from the model graph onto the structural data graph, capturing correlations among specific instances of the data. Figure 1 shows the three graphs for the citation analysis domain as presented in (Neville & Jensen 2004).

4. Relational Dependency Networks in Robotic Tasks

In this section we present a way of learning procedural task knowledge for robotic tasks using RDNs.

Gathering the Data

To gather useful information from experience, the robot must observe visual, tactile, and proprioceptive signals as well as events generated by actions. Each trial implementation is a procedural instantiation of the declarative structure of the task. In the experiments discussed in this paper, the declarative structure is given *a priori*. If some subset of the signals and events are recorded in a history, then learning can proceed asynchronously (i.e., as the robot behaves in the world) while different features from the captured data are correlated. Because relational data need not be i.i.d.,



(a) (b) (c)

Figure 1: **Research Paper (a) Data Graph, (b) Model Graph, and (c) Inference Graph**

each training trial can follow a different declarative structure, sometimes employing a controller multiple times with different resource bindings. For example, a robot may try to pick up an object by reaching to it with its left hand after it first tries and fails reaching with its right hand. Similarly a robot may try a number of different grasps (two-finger, three-finger, enveloping, etc.) until it succeeds.

Representing Procedural Task Knowledge

In the context of robotic tasks, a proper RDN representation of objects and attributes needs to be determined before learning algorithms can be applied. The fundamental unit of activity that we have employed is controller based. It seems natural, therefore, to make all RDN objects represent controllers. Relations between them may be causal or conjunctive. For example, relations might represent that REACH comes before GRASP, or that REACH should be performed in the null-space (Nakamura 1991) of OBSTACLEAVOIDANCE.

Attributes of controller objects can contain a wide variety of information that may be different for different controllers. Following the representation used in the control basis work of Huber *et al.* (Huber & Grupen 1996), two important attributes of controllers are "resource-binding" and "state." For a REACH controller, the "resource" may be a humanoid's left or right arm. For a TRACK controller, it may be a particular set of cameras. A controller's "state" attribute represents the binary convergence property at any given time. Attributes can be real-valued and multi-dimensional. They should contain any information gathered through the execution of the controllers whose objects they are attached to.

Figure 2 shows example objects for LOCALIZE, REACH, and GRASP controllers. LOCALIZE contains information gained through the processing of viewed images. The attributes "locale" and "bounding box dimensions" represent the xyz-positions of viewed objects and the dimensions of their image-plane, respectively. The "orientation" attribute of the REACH controller represents the resulting end-effector orientation upon reach completion, which in general is useful for performing grasp pre-shaping. The "fingers" attribute of the GRASP controller designates the number of fingers that should be used to grasp. The "lift-able" attribute indicates whether the object was lifted successfully upon grasp completion. This is necessary because achieving a grasp does not always mean that an object can be picked-up. For example, consider grasping a long heavy box from the end: the weight of the object might not allow the hand to successfully lift the object despite grasp convergence.

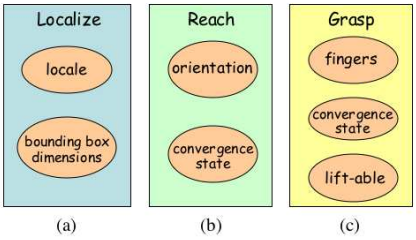


Figure 2: **Objects and Attributes for (a)** LOCALIZE**, (b)** REACH**, and (c)** GRASP

In general, the number of attributes available for a controller may be infinite. A LOCALIZE controller, for example, may gather information from a pair of stereo images. Because there are an infinite number of possible features available from an image (e.g., all Gaussian filters at all scales and orientations), it may be necessary to sample those used to build a model. This issue is beyond the scope of this paper, but is currently being pursued by the authors.

5. PICKUP Experiments

In this section we present a schema for picking up objects learned through actual experience with the UMass humanoid robot, Dexter (Figure 3). Dexter has two 7-DOF Barrett Whole Arm Manipulators (WAMs), each with a three-fingered Barrett hand mounted at its wrist. A stereo camera pair is also mounted on Dexter's shoulders. For the experiments presented in this paper, only one of Dexter's arms was used.

Figure 3: **Dexter - the UMass bi-manual humanoid**

The RDN uses the LOCALIZE, REACH, and GRASP objects shown in Figure 2. In addition to performing any visual processing, the LOCALIZE controller finds the centroid of an object from both image planes and performs a triangulation to discern its spatial location. The REACH controller moves the wrist to a specified location and is paramaterized by the desired resulting orientation (e.g., top or side). The GRASP controller descends a gradient defined by the squared wrench residual (Coelho, Piater, & Grupen 2001; Platt Jr., Fagg, & Grupen 2002). It is parameterized by the number of virtual fingers to be used (either two or three). Local force and contact normal information is taken from individual 6-DOF load cell sensors mounted on each of Dexter's fingertips.

One hundred trials were performed in which two objects—a cylindrical coffee can and a rectangular box—were placed in front of Dexter. The cylindrical object was always placed right side up, but the box was placed either standing up or laying on its side. The objects were placed in locations uniformly distributed in front of Dexter. Most locations were within the robot's workspace, but some were not. Furthermore, some locations were only reachable using certain reach orientations; Dexter can reach further from a side approach than a top approach.

In the course of the experimental trials, Dexter first localized the object, performing background subtraction to discern the object's 3D location as well as the bounding box dimensions on the image plane. For simplicity, only one object was presented at a time. Dexter then reached to the object, randomly choosing to approach from either the top or the side. If the reach failed, Dexter chose to reach again using another approach. If any of the reach actions completed, then Dexter began grasping the object using either two or three fingers. If a grasp failed, then Dexter could chose to grasp again using another combination of fingers. If any grasp succeeded, then Dexter applied force to the object and attempted to lift it. In most cases, the object was lift-able if it was grasped. However, Dexter could not lift the box if it grasped it from the side while it was laying down.

Because each trial may require up to two reaches and two grasps, each trial takes on one of four distinct structural forms (Figure 4). By using a relational data representation, we are able to learn a single model despite multiple instances of GRASP or REACH—which may depend on each other—in our training set. Note that this type of data would have to be forced into a "flat" representation if non-relational learning techniques were to be employed. Instead, all of the training data is collected into a data graph.

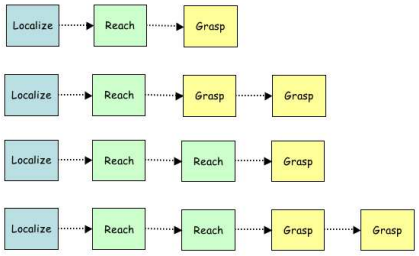


Figure 4: **Four data graph instantiations of the data**

Learning the Procedural Knowledge

We have assigned the PICKUP task the declarative structure LOCALIZE-REACH-GRASP using the intuition we, as programmers, have about how to pick up objects. In general, the declarative structure for a task may not be known and may need to be learned in conjunction with the procedural knowledge.

The declarative structure provides a template from which we can build our RDN and learn procedural knowledge expertise. The experimental data gathered with Dexter were used to train the model, resulting in the RDN shown in Figure 5. The model was learned using the PROXIMITY system for relational knowledge discovery[1]. The model captures how resource attributes can be instantiated given other observed variables.

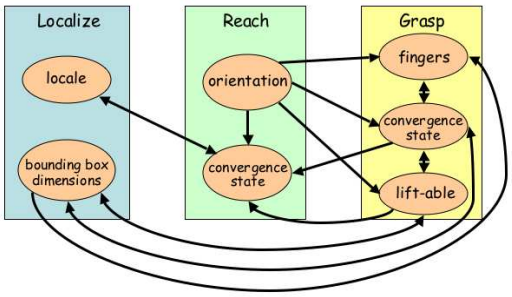


Figure 5: PICKUP **RDN built from experimental data**

Attribute Trees

The RDN algorithm estimates a conditional probability distribution of each attribute. For each attribute of an object, a locally consistent relational probability tree (or RPT) (Neville *et al.* 2003) is learned showing how attributes of the

[1] PROXIMITY was developed by the Knowledge Discovery Laboratory at the University of Massachusetts Amherst. It can be downloaded from http://kdl.cs.umass.edu/proximity/

same object or attributes of neighboring objects influence its value. Figure 6 shows the RPT for the "lift-able" attribute of the GRASP object learned by PROXIMITY. Interpreting this tree, we can see that if no grasp succeeds, then there is virtually no chance of the object being liftable. At the next level of the tree, there is a split pertaining to how wide the object is (in these experiments 159 pixels was deemed "wide" by the learning system). If it is not wide, then there is a an 83% chance of the object being liftable. If the object is wide, and the robot reaches from the side, then there is only a 7% chance of being able to lift it. If a top reach is made, then there is a good chance the object will be liftable. This tree is consistent with the observations made in the course of the experiments and explained in the previous section. Note that there is no indication of the number of fingers used in the grasp in this tree. We can see from Figure 5 that the finger count is conditionally independent of the value of "lift-able," given the success of the grasp.

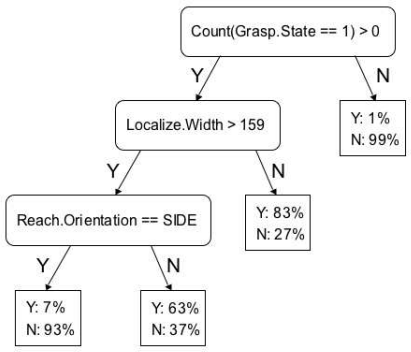


Figure 6: **Relational probability tree for the "lift-able" attribute. The** COUNT **function counts the number of times the specified object attribute has the indicated value (in this case, the** GRASP **"convergence state" is equal to true). This applies if there are one or more instances of that object in a trial.**

This RPT example makes apparent how the RDN architecture focuses a robot's attention on only a small set of available features to predict a given attribute value. Each RPT, therefore, learns the *affordance* of its attribute with respect to the task. At inference time, the RDN uses a Gibbs sampling technique to bring together the locally consistent RPTs for each attribute, only then determining a global probability estimate.

Validation

While performing the experiments, many dependencies were observable. For example, not all locations were reachable. The cylindrical object was only graspable with two fingers from the side, but either two or three fingers from the top. The box was best graspable with two fingers as opposed to three. We can see quite clearly in Figure 5 that many of these relationships are manifested in the learned model graph. The dimensions of the object on the background-subtracted image were the robot's only notion of shape. As a result, we would expect all of the GRASP attributes to be correlated with the "bounding box dimensions" attribute. Also note that the "lift-able" attribute is dependent on the "convergence state" of GRASP, as well as the "bounding box dimensions" of the object and the "orientation" attribute of REACH. This is to be expected because only when grasping the long box from the side (while it was horizontal) was an object graspable but not liftable.

Ultimately, the true test of the system is how well it recommends resources for performing a task again. In this section we show that the RDN model graph learned from the experimental data can be used to construct a policy for the robot to pick up objects of the classes discussed. More specifically, the best values of the resource attributes can be *inferred* from the model at decision points, given the observed variables and the desired state variables.

To test the model, a number of trials were performed in which the two objects were placed in front of Dexter and a resource allocation was inferred from the RDN. When the objects were placed in the robot's field of view, the LOCALIZE controller determined the "bounding box dimensions" and "locale" variables. These observed variables were then used, along with the desired state variable settings (REACH and GRASP "convergence state," and "lift-able" all equal to true), to infer the values of REACH "orientation" and GRASP number of "fingers." In general, the model recommended the policy in which it picked up the cylindrical coffee can either from the top (with equal probability of two or three fingers) or the side with two fingers. The policy to pick up the box was to use two fingers (it rarely was able to pick up the box with three fingers during training), always from the top if it was laying horizontally, and either from the side or top if was standing vertically. For a number of locations outside Dexter's reachable workspace, the model predicted a very low probability of success for any resource allocation. All of these predictions were consistent with the training data and yielded a policy that allowed Dexter to pick up the two objects correctly.

6. Conclusions and Future Work

In this paper we have proposed a novel architecture that will allow a robot to learn about the complex statistical relationships present when performing tasks in the world. The PICKUP task presented in Section 5 demonstrates the proposed architecture with a specific example, but the approach is general and can be applied to find the affordances of sensorimotor variables in any given task. Many dependencies between these features will not be obvious to a human, and it is important to have learning techniques that will allow a robot to discover these relationships from its own experience.

It is also important that the available feature set from which the system can learn models is rich enough to ensure procedural task knowledge in a variety of contexts. Coelho showed that dynamic models characterizing the behavior of controllers are useful in determining the environmental context in which the controller is running (Coelho 2001). These

models were learned from the phase-space trajectory of a grasp controller's run-time execution and could then be used to distinguish the shape of the object being grasped. Membership in such parametric models may be useful attribute values for the RDN format presented in this paper because they allow the controllers to function as abstract sensors.

There may, however, be an uncountable set of possible features that are made available to the system. Learning a model from a large feature set is intractable and far too time consuming for a system that lives on a robot. In the future we plan to look at tasks that require the robot to learn models from only a *sampled set* of available features, resampling if poor policies are learned. In such experiments it will be necessary for the to robot mine those features that are useful predictors, as well as ignore those that are erroneous.

To develop true autonomous systems, the declarative task structure must be learned in addition to the procedural structure. There has been previous work in the *control basis* framework in which abstract policies were learned using reinforcement learning techniques (Huber 2000). However, as the available information streams increase, techniques that rely on an explicit state space representation (e.g., reinforcement learning) become impractical when learning at the most primitive level. Mining the sensorimotor streams to learn procedural structure first allows reinforcement learning techniques to deal only at the more abstract declarative level, which has a smaller dimensionality. We propose using relational techniques to perform this mining because they capture many properties of task experience, discussed in this paper, in a natural and intuitive way.

Acknowledgments

This research was supported in part by NASA and DARPA under contract numbers NAG9-01445 and HR0011-04-1-0013 and by NASA Graduate Student Research Program fellowship NNJ04JF85H. We would like to thank Emily Horrell, Robert Platt, and Jennifer Neville for their help and support. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of NASA, DARPA, or the U.S. Government.

References

Arbib, M. 1995. Schema theory. In *The Handbook of Brain Theory and Neural Computation*, 830–834. Cambridge, MA: MIT Press.

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

Coelho, J. A. 2001. *Multifingered Grasping: Grasp Reflexes and Control Context*. Ph.D. Dissertation, Department of Computer Science, University of Massachusetts Amherst.

Fitzpatrick, P.; Metta, G.; Natale, L.; Rao, S.; and Sandini, G. 2003. Learning about objects through action: Initial steps towards artificial cognition. In *IEEE International Conference on Robotics and Automation*.

Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference of Artificial Intelligence*.

Gallese, V.; Fadiga, L.; Fogassi, L.; and Rizzolatti, G. 1996. Action recognition in the premotor context. *Brain* 119:593–609.

Gibson, J. 1977. The theory of affordances. In *Perceiving, Acting and Knowing: Toward an Ecological Psychology*, 67–82. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

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

Huber, M. 2000. *A Hybrid Architecture for Adaptive Robot Control*. Ph.D. Dissertation, Department of Computer Science, University of Massachusetts Amherst.

Kupiers, B.; Beeson, P.; Modayil, J.; and Provost, J. 2005. Bootstrap learning of foundational representations. In *Developmental Robotics, AAAI Spring Symposium Series*.

Mandler, J. M. 2004. *The Foundations of Mind: Origins of Conceptual Thought*. New York, NY: Oxford University Press.

Nakamura, Y. 1991. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley.

Natale, L.; Metta, G.; and Sandini, G. 2004. Learning haptic representation of objects. In *International Conference on Intelligent Manipulation and Grasping*.

Neville, J., and Jensen, D. 2003. Collective classication with relational dependency networks. In *2nd Workshop on Multi-Relational Data Mining, KDD*.

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

Neville, J.; Jensen, D.; Friedland, L.; and Hay, M. 2003. Learning relational probability trees. In *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining*.

Piater, J. H. 2001. *Visual Feature Learning*. Ph.D. Dissertation, Department of Computer Science, University of Massachusetts Amherst.

Platt Jr., R.; Fagg, A. H.; and Grupen, R. A. 2002. Nullspace composition of control laws for grasping. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

Roy, D.; Gorniak, P.; Mukherjee, N.; and Juster, J. 2002. A trainable spoken language understanding system for visual object selection. In *International Conference for Spoken Language Processing*.