Which Object Fits Best?
Solving Matrix Completion Tasks with a Humanoid Robot

Connor Schenck, Jivko Sinapov, David Johnston, and Alexander Stoytchev
Developmental Robotics Laboratory
Iowa State University
{cschenck, jsinapov, dwtj, alexs}@iastate.edu

***Abstract*—Matrix completion tasks commonly appear on intelligence tests. Each task consists of a grid of objects, with one missing, and a set of candidate objects. The job of the test taker is to pick the candidate object that best fits in the empty square in the matrix. In this paper we explore methods for a robot to solve matrix completion tasks that are posed using real objects instead of pictures of objects. Using several different ways to measure distances between objects, the robot detected patterns in each task and used them to select the best candidate object. When using all the information gathered from all sensory modalities and behaviors, and when using the best method for measuring the perceptual distances between objects, the robot was able to achieve 99.4% accuracy over the posed tasks. This shows that the general framework described in this paper is useful for solving matrix completion tasks.**

I. Introduction

Intelligence tests have long been used to measure the Intelligence Quotient (IQ) of humans. Matrix completion tasks often appear on many intelligence tests. These problems consist of a grid of objects, where one entry in the grid is missing. The job of the test taker is to select an object from a set of given candidates that best fits in the empty slot in the grid. The most well-known intelligence test that employs matrix completion tasks, the Raven's Progressive Matrices (RPM) test [1], has been shown to predict performance on a large set of reasoning tasks [2], [3]. Other common intelligence tests, such as the Wechsler Abbreviated Scale of Intelligence (WASI) [4], also have sections that consist of matrix completions tasks.

Matrix completion tasks emphasize the ability to reason about the relationships between the objects in the matrix, rather than merely recalling stored knowledge about the objects. In fact, John Raven developed the RPM test specifically to remove biases that he saw in previous tests that made it difficult to accurately compare the scores of participants with and without extensive knowledge of concepts such as language [5]. Currently there are no robotic systems that are capable of building longitudinal knowledge bases or understanding language to the same extent that humans can. However, because matrix completion tasks do not require extensive background knowledge to solve, it is feasible to solve them with the current state of the art in robotics.

Matrix completion tasks appear not only on intelligence tests, but also on many other tasks outside of these tests. For example, the grid layout of the periodic table of the elements makes it easy to see the analogous relationships between elements in the same relative positions (e.g., cadmium can replace zinc in many vital enzymes, a relationship that is made apparent by the arrangement of the periodic table) [6]. In fact, when Mendeleev created the periodic table, due to the underlying properties of its layout, he was able to successfully predict the existence and properties of many yet undiscovered elements [6]. This suggests that the concepts underlying matrix completion tasks are very useful for solving other, related tasks in real-world environments.

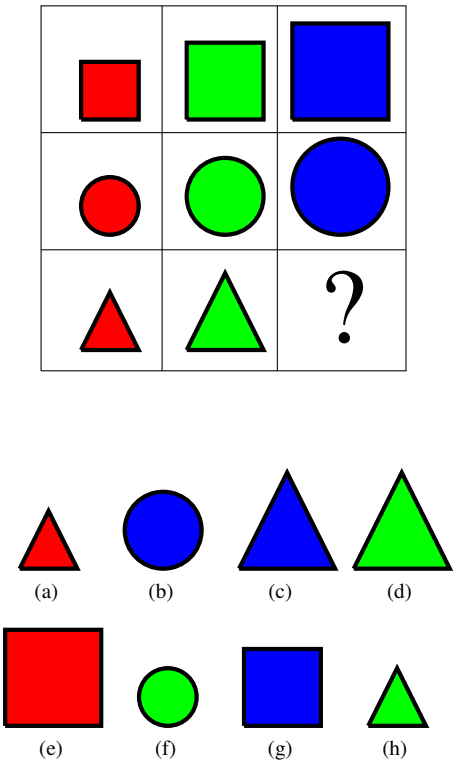


Fig. 1: An example matrix completion task. The correct answer is (c) because the large blue triangle is the same shape as the other objects in the bottom row, it is increasing in size over the previous object in its row, and it completes the permutation over the colors in the bottom row. Looking down the columns, it is also the same size and color as the other objects in its column and it completes the permutation over the shapes in the last column.

This paper formulates a framework for solving matrix

completion tasks by a robot. The robot first interacted with a set of objects while recording from its auditory, visual, and proprioceptive sensory modalities. Then we randomly generated 500 matrix completion tasks using objects from the set and posed them to the robot. The robot generated a set of distance functions using four different methods: raw context distances (unsupervised and supervised) and category-based distances (unsupervised and supervised), which are described in section V-E. It then used them to attempt to deduce the patterns in the given matrix in order to select the best candidate object for each task. Using the best of these four distance methods, the robot was able to achieve 99.4% accuracy on the tasks. To the best of our knowledge, this is the first attempt at using a robot to solve matrix completion tasks.

II. Related Work

A. Psychology

The most well-known matrix completion test, the Raven's Progressive Matrices (RPM) test [1], is known to predict performance on a wide range of reasoning tasks [3]. The RPM is composed exclusively of matrix completion tasks and more recent intelligence tests have begun to incorporate such tasks into sections of their tests. For example, the third version of the Wechsler Abbreviated Scale of Intelligence (WASI) added a section on matrix reasoning [4]. A study by Fuchs *et al.* [7] found that performance on the WASI's matrix reasoning section was predictive of third-graders' problem solving abilities. Another study found a correlation between performance on the WASI matrix reasoning section and the Halstead Category Test, which measures "complex spatial abstract reasoning and the use of conceptual rules in reasoning with ratios and proportions" [8]. Both of these suggest that performance on matrix completion tasks is indicative of performance on other complex reasoning tasks.

Dugbartey *et al.* [8] found that "elementary visuoperceptual abilities may be a necessary, but not sufficient, requirement for success" on the matrix reasoning section of the WASI test. This suggests that an ability to understand the objects in the matrix is fundamental to being able to solve the task. This was reinforced by a study by Richardson [9], which found that children performed significantly better on the RPM when problems were converted to use more meaningful symbols while the underlying structure of the problems remained the same. In particular, Richardson found that children were better able to perceive patterns present in the matrices after the problems were converted to use contextually meaningful symbols (e.g., cars and people) instead of abstract symbols (e.g., lines and circles) that were difficult for the children to understand.

Our previous work has dealt extensively with improving our robot's ability to understand the physical properties of objects [10], [11], [12], [13] and solving various tasks related to intelligence testing [14], [15], [16]. In this paper, similar to [9], we will utilize physical objects to pose matrix completion tasks to the robot while maintaining the general structure that underlies matrix reasoning problems.

TABLE I: The taxonomy of rules governing the variations in the rows of the matrices on the RPM test as identified by Carpenter *et al.* [17] and restated by Little *et al.* [18]. Similar rules can be stated for the columns of the matrix as well.

Rule	Description
constant	the same value occurs throughout a row, but changes down a column
increment	a quantitative increment between adjacent entries in an attribute such as size, position, or number
decrement	a quantitative decrement between adjacent entries in an attribute
permutation	a permutation of different values across a row for a categorical attribute such as figure type
logical AND	the third object in a row is the logical AND of the first two
logical OR	the third object in a row is the logical OR of the first two
logical XOR	the third object in a row is the logical XOR of the first two
distribution of 2	two objects in a row have an identical value and the third object has a different (usually null) value

TABLE II: Rules for generating matrix completion tasks identified by previous work and used in this paper.

Carpenter et al. 1990	*Little et al.* 2012	This paper
Constant in a row	Constant	**Constant**
Quantitative pairwise progression	Increment	**Increment**
	Decrement	**Decrement**
Figure addition or subtraction	Logical AND	N/A
	Logical OR	
Distribution of 3	Permutation	**Permutation**
Distribution of 2	Logical XOR	N/A
	Distribution of 2	

Carpenter *et al.* [17] proposed a taxonomy of rules that governs the patterns in the matrices of the RPM. They claimed that all problems on the Advanced Progressive Matrices test (one variant of the RPM), with the exception of two problems, could be stated as a combination of some subset of these rules. The rules they proposed (as restated by Little *et al.* [18]) are shown in Table I.

In the experiments presented in this paper, we will use the rules *constant*, *increment*, *decrement*, and *permutation* to generate matrix completion tasks for the robot to solve. Table II shows the mapping of the rules from the taxonomy that Carpenter *et al.* [17] derived, to the rules stated by Little *et al.* [18], and to the rules that we are using in this paper. We chose not to use any of the rules based on logical operators and the *distribution of 2* rule due to the constraints imposed by utilizing physical objects rather than images as the entries in each matrix completion task.

An example task that uses these rules is shown in Figure 1. The matrix in this figure exhibits multiple instances of the rules: *constant* in *shape* across the rows; *increment* in *size* across the rows; *permutation* for *color* across the rows; *permutation* for *shape* across the columns; *constant* in *size* across the columns; and *constant* in *color* across the columns. Given these constraints, the only object that can be placed in the empty square, while maintaining consistency with these

rules in the matrix, is object (c).

While intelligence tests have been shown to measure useful criteria of intelligence, the use of exploratory behaviors has been shown to be a fundamentally important part of intelligence in animals and humans. The use of exploratory behaviors has been reported in the literature since the late 1800s [19], [20], [21], [22], [23]. Both animals [24], [25], [26], [27], [28], [29], [30], [31] and humans [32], [33], [34] have been shown to make extensive use of exploratory behaviors. In fact, many correlations have been found between use of exploratory behaviors and performance on various other tasks [30], [35], [36], [37], [38], [39]. Even robots have been shown to benefit from the use of exploratory behaviors [40], [41], [42], [43], [44], [45], [46], [47], [48], [49]. This indicates that exploratory behaviors offer an effective way of learning about objects and the environment. In this paper, as in our previous work [15], [16], the robot will use multiple, stereotyped exploratory behaviors in order to learn about the objects.

B. AI and Robotics

In the AI literature there have been several attempts to formulate computational solutions for the Raven's Progressive Matrices (RPM) test [17], [18], [50], [51], [52], [53]. Most of these, however, utilize hand-coded features, rather than real data from sensors. The algorithm developed by Kunda *et al.* [52] was the only one that did not utilize hand-coded features. Instead, they relied on complex mathematical transformations between images of the objects in the RPM to solve the tasks. None of these computational solutions, though, used a robot. To the best of our knowledge, this paper is the first attempt at solving matrix completion tasks using an embodied approach.

Although there has not been any previous work done in robotics on matrix reasoning, there has been a significant amount of work that utilizes the similarity of objects to solve tasks. There have been numerous experiments that demonstrate a robot's ability to solve tasks using perceptual and functional similarity (e.g., [54], [44], [55], [56], [57], [58], [59], [60], [14]). The ability to measure the similarity between objects is fundamental to being able to solve many different tasks [14], [15], [16]. This paper tests the hypothesis that this ability is also fundamentally important to solving matrix completion tasks.

Our own previous work has used perceptual similarity extensively to solve multi-object tasks. In [14], Sinapov and Stoytchev used the similarity between pairs of objects to solve the odd-one-out task. In [16], Schenck and Stoytchev used the perceptual distance between objects to solve the object pairing task. In [15], Schenck *et al.* used the perceptual distance between objects to solve the order completion task. In this paper we extend our previous work and methodologies to solve matrix completion tasks.

Figure 3 shows the framework for solving multi-object tasks that has been developed over the course of our previous work [14], [15], [16]. The general structure of the framework is as follows. First, the robot interacts with all objects by performing a set of stereotyped exploratory behaviors. Second, the robot extracts features from the raw sensory data that it recorded and then computes multiple similarity scores between every pair of objects. Third, the robot combines the similarity scores from multiple sensorimotor contexts. Fourth, the robot uses these combined scores to compute the value of a task-specific objective function. Finally, the robot uses the result to pick the best candidate object that maximizes the task-specific objective function. This framework will be used in this paper to solve matrix completion tasks.

Fig. 2: The robot used in these experiments. It is shown here with only its right arm as the left arm was temporarily removed for maintenance when these experiments were performed. The Microsoft Kinect camera is mounted on the lower part of the robot's torso.

III. Experimental Platform

A. Robot and Sensors

All experiments described in this paper were performed using the robot shown in Figure 2. The robot is equipped with two 7-DOF Barrett Whole Arm Manipulators (WAMs), each with an attached Barrett Hand. Each WAM can measure its own joint angles and torques at a rate of 500 Hz. The robot used only its right arm to perform the behaviors in these experiments as its left arm was temporarily removed for maintenance. The robot also has an Audio-Technica U853AW cardioid microphone mounted in its head in order to capture auditory feedback at the standard 16-bit/44.1kHz over a single channel. During the experiments, the robot was also equipped with a Microsoft Kinect camera, which can capture both RGB video and depth information. The Kinect camera was attached to the lower part of the robot's torso, slightly above the table and pointed down at it.

B. Objects

The objects used in this paper were designed specifically to maintain the general structure of matrix completion tasks

Fig. 3: The framework used by the robot to solve perceptual reasoning tasks that involve multiple objects. The steps are as follows: 1) The robot explores the objects; 2) It extracts features from the raw sensory data it collected; 3) The robot combines the information from each of the sensorimotor contexts; 4) It computes the value of a task-specific objective function for each candidate solution; and 5) The robot selects the candidate object that maximizes the objective function.

(a) Color: green, red, and blue.

(b) Contents: glass, rice, beans, and screws.

(c) Weight: light, medium, and heavy.

Fig. 4: The properties by which the objects varied. Each object is a jar that is one of three colors, filled with one of four different types of contents, and weighing one of three different weights for a total of 36 objects (see Figure 5).

Fig. 5: The 36 objects used in this paper, grouped by color. Within each group, all objects of the same weight are in the same row and all objects with the same type of contents are in the same column.

as described in [17] while moving to the domain of physical objects (as opposed to images on a piece of paper). Figure 4 shows the three properties that the objects varied by. Each object is a cylindrical plastic jar that is 8.6 centimeters tall and 9.4 centimeters in diameter. The jars are semi-transparent, each being one of three colors: *blue*, *green*, or *red* (see Figure 4a). Each jar is filled with one of four different types of contents: *glass beads*, *rice*, *beans*, or *screws* (shown in Figure 4b). Each jar was filled until it weighed either 166g, 250g, or 337g (shown in Figure 4c). In all, there are 3 colors $\times$ 4 contents types $\times$ 3 weights $=$ 36 total jars (one for each permutation of the values). Figure 5 shows all 36 objects.

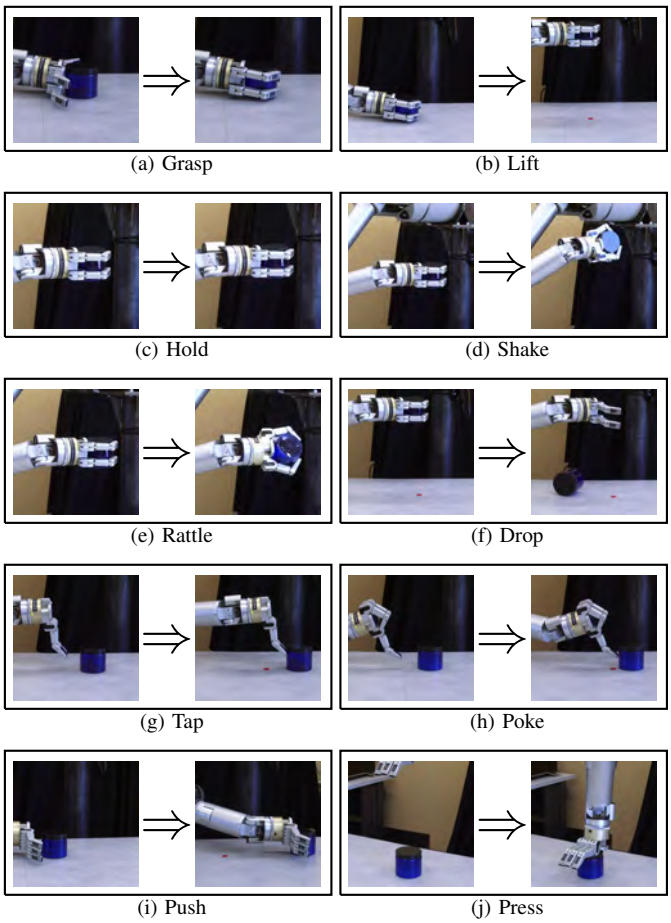
C. Exploratory Behaviors

The robot performed ten stereotyped behaviors to explore the objects: *grasp*, *lift*, *hold*, *shake*, *rattle*, *drop*, *tap*, *poke*, *push*, and *press*. All of these behaviors are shown in Figure 6. In addition to these behaviors, the robot also performed the *look* behavior (not shown in Figure 6), during which it took a visual snapshot of the object on the table in front of it with the Kinect camera before performing the other behaviors on it. These behaviors were used in our previous work [16] and were not designed specifically for the objects in this paper. In fact, this set of behaviors has been developed for use across many different object sets for different tasks (see [61], [10], [13], [16] for a progression of the behaviors as they evolved over time). All behaviors in this paper were performed with the robot's right arm and encoded using Barrett's API. The trajectory of the joint positions for each of the behaviors was executed using the default PID controller of the WAM. All behaviors were performed identically on each object, with only minor variations due to the initial placement of the object.

D. Sensorimotor Contexts

The robot in this paper used 21 sensorimotor contexts. A sensorimotor context is defined as a behavior combined with a sensory modality, e.g., *drop-audio*. We will use the notation *behavior-modality* to denote a context and the letter $\mathcal{C}$ to denote the set of all contexts. Table III shows all combinations of behaviors and modalities that the robot used. The robot used all the behaviors except *look* in combination with the modalities *audio* and *proprioception*. In addition to this, the robot also used the context *look-color*. This resulted in a total of $|\mathcal{C}| = 10 \times 2 + 1 = 21$ sensorimotor contexts.

For each object O_i and each context $c \in \mathcal{C}$, a set of feature vectors $\mathcal{X}_i^c$ was computed as described below in



(a) Grasp (b) Lift (c) Hold (d) Shake (e) Rattle (f) Drop (g) Tap (h) Poke (i) Push (j) Press

Fig. 6: Before and after images for the ten exploratory behaviors that the robot performed on all objects. From left to right and top to bottom: *grasp*, *lift*, *hold*, *shake*, *rattle*, *drop*, *tap*, *poke*, *push*, and *press*. The object was placed back in the initial position by the experimenter after some of the behaviors (e.g., drop).

TABLE III: The set of sensorimotor contexts used by the robot. The X's denote behavior-modality combinations that the robot used to solve matrix completion tasks.

Behavior	**Modality**		
	proprioception	*audio*	*color*
look			X
grasp	X	X	
lift	X	X	
hold	X	X	
shake	X	X	
rattle	X	X	
drop	X	X	
tap	X	X	
poke	X	X	
push	X	X	
press	X	X	

section IV. Each $\mathbf{x} \in \mathcal{X}_i^c$ is a feature vector computed from one interaction in context c. Because each behavior was performed 10 times on each object, there were 10 feature vectors in each $\mathcal{X}_i^c$, i.e. $|\mathcal{X}_i^c| = 10$.

In our previous work ([15], [16]) we used only 2 sensory modalities (*audio* and *proprioception*) and 10 behaviors for a total of 20 contexts. In those papers, vision was not required to solve the tasks. In this paper, however, color is an important property of the objects, so the robot was required to use vision to solve the tasks. Thus, we added the *look-color* context for a total of 21 contexts (the same 20 from our previous work plus *look-color*).

E. Software

The software used to control the robot was written in C++ using the Barrett API. This software also used the Barrett API to record the robot's joint torques during each behavior and the OpenAL library to record from the robot's microphones. RGBD images were captured from the robot's Kinect sensor using the Point Cloud Library [62]. The data was analyzed offline by software written in Java. The SPHINX4 toolbox [63] was used to process the audio data and the publicly available code for the spectral clustering algorithm [64] was used to perform clustering as described in Section V-E2.

F. Data Collection

The robot interacted with the objects by performing each of the behaviors on each object ten times. At the start of this process the experimenter placed the first object at a specified location on the table in front of the robot, and then the robot performed one of its behaviors on the object. The experimenter then placed the second object on the table in the same spot, and the robot performed the same behavior on it. This was repeated for all objects. The experimenter then placed the first object in the same spot on the table and the robot performed the next behavior on it, repeating this again for all objects. This was done until the robot had performed each behavior once on each object. This entire process was then repeated nine more times (for a total of ten repetitions) such that the robot had performed each behavior ten times on each object. There were 36 objects, 10 behaviors, and 10 repetitions, resulting in a total of $36 \times 10 \times 10 = 3600$ interactions with the objects.

IV. Feature Extraction

A. Proprioceptive Feature Extraction

During each interaction, the robot recorded the joint torques applied to its right arm. The robot sampled from all 7 joints at 500 Hz. This resulted in $7 \times m$ real numbers, where m is the number of temporal samples during each interaction. In other words, each interaction resulted in a matrix, where each column contains the joint torque readings at one point in time and each row contains the torque values applied to one joint over the course of the interaction. This matrix was too high-dimensional to be effective for the tasks in this paper, so

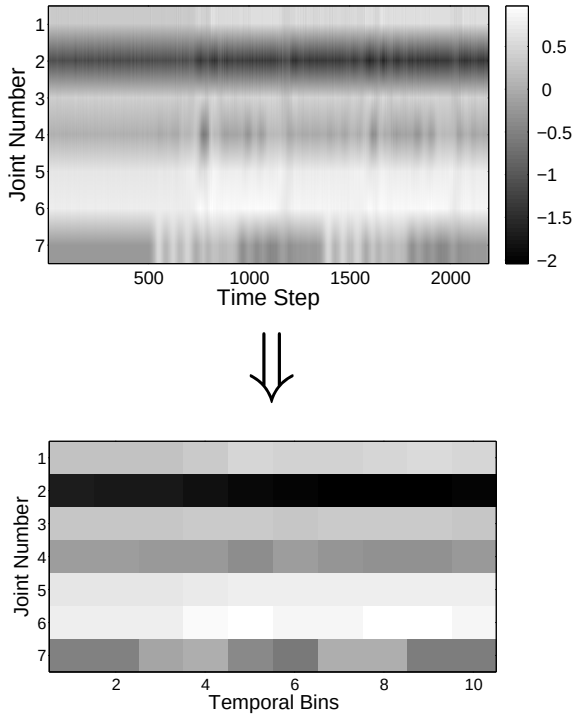


Fig. 7: An example sensory record of proprioceptive values. The top image depicts the raw joint torques recorded from the robot's arm during the interaction where light values denote positive torque values and dark values indicate negative torque values. The bottom image depicts the features extracted from that by binning the values for each of the 7 joints into 10 temporal bins.

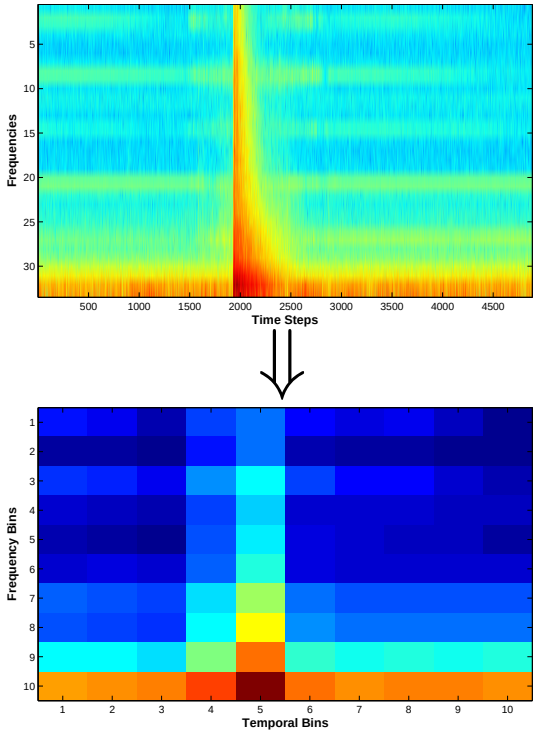


Fig. 8: An example sensory record of auditory values. The top image depicts the audio spectrogram, which was computed using a series of DFTs on the raw data that was recorded during the interaction (red denotes higher activation, green and blue denote lower activation). The bottom image depicts the features extracted from the spectrogram by binning the values into 10 temporal bins and 10 frequency bins.

features were extracted from it by binning the real values for each joint into 10 temporal bins. That is, the first $\frac{m}{10}$ columns were summed together into one column, then the second $\frac{m}{10}$ were summed together, and so on. This resulted in a feature vector $\mathbf{x} \in \mathbb{R}^{7\times 10}$. Figure 7 illustrates this process.

B. Auditory Feature Extraction

During each interaction, auditory data was recorded by the microphones in the robot's head in the form of a .wav file. Each .wav file was then converted into a spectrogram using the log-normalized Discrete Fourier Transform (DFT) with $2^5 + 1 = 33$ frequency bins. The SPHINX4 natural language processing library was used to compute the DFT for each audio file [63]. The spectrogram for each audio file was calculated by computing the DFT over the length of the interaction. This resulted in a $33 \times m$ dimensional matrix, where m is the number of time samples. Like in the proprioception case, this matrix was too high-dimensional to be useful. To lower the dimensionality, a 10×10 spectro-temporal histogram was computed for each of the audio spectrograms. That is, each spectrogram was divided up into $10 \times 10 = 100$ bins and then all the values in each bin were summed. This resulted in a feature vector $\mathbf{x} \in \mathbb{R}^{10\times 10}$. An example of this process is shown in Figure 8. In addition to the temporal binning done in the previous method, this method also performed frequency binning.

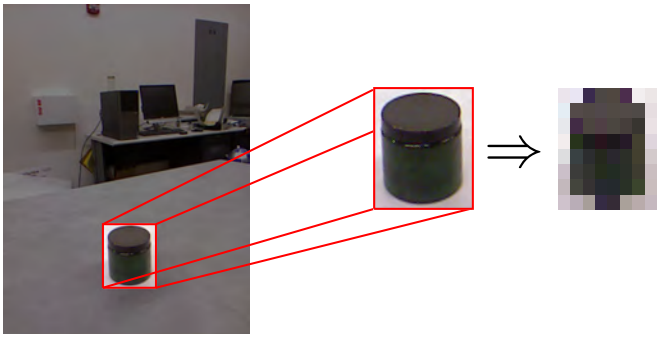


Fig. 9: An example image recorded during the *look* behavior. The left image shows the raw RGB data that the robot's camera recorded. The middle image is the segment of the robot's field of view where the object was always placed on the table. The right image is the 8×8 grid that this segment was binned into, where each location in the grid is the average of the pixels that fall into that cell.

C. Visual Feature Extraction

Visual features were computed based on the color information from the robot's Kinect camera. For simplicity, color features were only extracted during the behavior *look* (based on the methodology described in [13]). During each *look* behavior the robot recorded a short series of images of the

object sitting on the table in front of it. The object was always placed in approximately the same spot on the table, so it was segmented out of each recorded image using a pre-set region of interest. For each image, the robot then divided this region into $r \times r$ bins (where $r = 8$) and averaged the *HSV* values in each bin, resulting in a vector $\mathbf{x}_i \in \mathbb{R}^{r \times r \times 3}$. This process is shown in Figure 9.

For each image in a series of images from one *look* behavior, the robot simply averaged the vectors together as follows:

$$\mathbf{x} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{x}_i$$

where k is the number of images captured during the *look* behavior. This resulted in a feature vector $\mathbf{x} \in \mathbb{R}^{r \times r \times 3}$ for each *look* behavior.

V. Experimental Methodology

A. Problem Formulation

Let $\mathcal{M}$ denote a square matrix of objects with n rows and n columns[1] where $\mathcal{M}_{ij}$ is the object in the i-th row and the j-th column. Let R_i be the i-th row of $\mathcal{M}$ and C_j be the j-th column of $\mathcal{M}$. Let $\mathcal{P}^r$ and $\mathcal{P}^c$ be sets of patterns defined over the rows and columns, respectively. For all $p \in \mathcal{P}^r$, let $f_p^{\rightarrow}$ denote a binary function that takes as input a row of $\mathcal{M}$ and evaluates to true if pattern p is present in that row and false otherwise. The binary function $f_p^{\downarrow}$ is similarly defined for all $p \in \mathcal{P}^c$.

Let $present^{\rightarrow}$ be a binary function defined over matrices and row patterns. Given a pattern $p \in \mathcal{P}^r$ and a matrix $\mathcal{M}$, it is evaluated as follows:

$$present^{\rightarrow}(p, \mathcal{M}) = \begin{cases} true, & \forall_{(\mathcal{R}_i \in \mathcal{M} - \{\mathcal{R}_n\})} \ f_p^{\rightarrow}(\mathcal{R}_i) = true \\ false, & \text{otherwise.} \end{cases}$$

In other words, this function evaluates to true if and only if p is present in all but the last row of $\mathcal{M}$. The function $present^{\downarrow}$ is similarly defined for all $p \in \mathcal{P}^c$.

A matrix $\mathcal{M}$ is considered valid with respect to $\mathcal{P}^r$ and $\mathcal{P}^c$ if and only if it satisfies the following conditions:

$$\forall_{p \in \mathcal{P}^r} \ present^{\rightarrow}(p, \mathcal{M}) \Rightarrow f_p^{\rightarrow}(\mathcal{R}_n) = true,$$
$$\forall_{p \in \mathcal{P}^c} \ present^{\downarrow}(p, \mathcal{M}) \Rightarrow f_p^{\downarrow}(\mathcal{C}_n) = true.$$

The first condition enforces that if the pattern $p \in \mathcal{P}^r$ exists in the first $n-1$ rows of the matrix, then it must also exist in the last row. The second condition enforces the same constraint, but on the columns. The idea behind this is that the patterns detected in the first $n-1$ rows and columns can be used to determine the candidate object that best fits in the last spot in the matrix.

A matrix completion task is defined as the ordered pair $(\mathcal{M}, \mathcal{G})$ where $\mathcal{M}$ is a matrix that is missing the object $\mathcal{M}_{n,n}$ (i.e., the object in the lower-right corner), and $\mathcal{G}$ denotes a set of candidate objects such that exactly one object may be placed in the empty space and cause $\mathcal{M}$ to be a valid matrix as defined above. In other words, the task is to select the object from $\mathcal{G}$ that creates a valid matrix with respect to the patterns in $\mathcal{P}^r$ and $\mathcal{P}^c$.

[1]In this paper we used only square matrices, but it is easy to extend this methodology to non-square matrices.

B. Task Generation

To generate a set of matrix completion tasks to test the robot on, we first had to generate two sets of patterns $\mathcal{P}^r$ and $\mathcal{P}^c$ using the generation rules described in section II-A. Once again, the rules used in this paper are: *constant*, *increment*, *decrement*, and *permutation*. A pattern is defined as an instantiated rule, i.e., a rule and a property that the rule applies to. For example, the rule *constant* applied to the property *color* would be denoted by *constant:color* and would mean that the color of the entries in the row or column is constant. In general, we will use the notation *rule:property* to denote patterns in the matrix.

The objects in this paper vary by three properties: *color*, *contents*, and *weight*. To determine the patterns to use, we applied the rule *constant* to all three properties, the rules *increment/decrement* to the ordered properties (*weight*), and the rule *permutation* to the unordered properties (*color* and *contents*). This resulted in seven patterns: *constant:contents*, *constant:color*, *constant:weight*, *permutation:contents*, *permutation:color*, *increment:weight*, and *decrement:weight*. The same set of patterns were used for both the rows and the columns.

The matrix completion tasks were generated as follows. Two patterns, p^r and p^c, were randomly selected from the set of patterns for rows, P^r, and columns, P^c, respectively. A matrix $\mathcal{M}$ was randomly selected from the set of all valid matrices such that p^r was present in all the rows of $\mathcal{M}$ and p^c was present in all the columns. Seven objects were then randomly selected from the set of objects not in $\mathcal{M}$ such that none of them could replace the last object in $\mathcal{M}$ and create a valid matrix. These objects comprised the set $\mathcal{G}$. The object in the lower-right corner of $\mathcal{M}$ (i.e., the correct solution) was then removed from $\mathcal{M}$ and placed in $\mathcal{G}$. The resulting matrix completion task was the ordered pair $(\mathcal{M}, \mathcal{G})$.

It is worthwhile to mention that, because the number of valid matrices is exponentially large, the above algorithm is intractable. Therefore, we used a slightly different algorithm that is functionally equivalent to that algorithm to generate matrix completion tasks. We divided the set of patterns into groups, one for each property. For example, all the patterns over *color* in one group (*constant:color*, *permutation:color*), all the patterns over *contents* in another group (*constant:contents*, *permutation:contents*), etc. For each group, we iterated over all possible permutations of the values of length $3 \times 3 = 9$ of the associated property (that is, one value for each position in the 3×3 matrix), keeping only the permutations that correspond to valid matrices with respect to the group of patterns (that is, matrices that are valid if only that one property is considered). While the number of possible permutations for even a single property is also exponentially large (e.g., because there are 3 values for *color*, there are $3^9 = 19,683$ possible permutations), it is tractable when the size of the matrix and the number of values for the properties are relatively small (which in this

case, they are). We were then able to randomly select one valid permutation from each of these sets and combine them together. Doing so resulted in a complete matrix. That is, since one permutation was selected for each of the properties, the property values for each object in the matrix were specified by combining these permutations together. This uniquely specified the object that belonged in each location in the matrix. This allowed us to randomly sample from the set of all possible valid matrices. Given this, we could then generate the tasks as described in the previous paragraph.

C. Selecting the Best Candidate to Complete a Matrix

Given a matrix reasoning task $(\mathcal{M}, \mathcal{G})$, the robot must select the best candidate object from the set $\mathcal{G}$ to complete the matrix $\mathcal{M}$. In order to do this, the robot first generates a set of distance functions $\mathcal{D}$ (described below in section V-E) such that the value of $D(O_i, O_j) \in [0, 1]$ is the distance between object O_i and object O_j as measured by $D \in \mathcal{D}$.

Next, the robot selects the best candidate object from $\mathcal{G}$ by finding the object $O_k \in \mathcal{G}$ that minimizes the following objective function:

$$q(O_k, \mathcal{M}, \mathcal{D}) = \sum_{D \in \mathcal{D}} A_D \left[\sum_{j=1}^{n-1} \left(D(\mathcal{M}_{n,j}, O_k) - \mathbf{E}[D(\mathcal{M}_{n,j}, \widehat{\mathcal{M}}_{n,n})] \right)^2 \right], \quad (1)$$

where $\mathcal{M}$ is an $n \times n$ matrix that is missing its lower-right element, A_D is the consistency of D across the rows of $\mathcal{M}$ (explained below), and $\mathbf{E}[D(\mathcal{M}_{n,j}, \widehat{\mathcal{M}}_{n,n})]$ is the expected distance between $\mathcal{M}_{n,j}$ and $\widehat{\mathcal{M}}_{n,n}$ with respect to D. In this formula $\widehat{\mathcal{M}}_{n,n}$ represents the robot's estimation of the missing object, so the expected distance is computed, rather than the actual distance, because the object is missing. The intuition behind this function is that the robot computes the difference between the object that *should* be in the missing space and O_k. It does this by computing the squared difference between what it expects D to evaluate to and what it evaluates to when placing O_k in the missing spot, for all $D \in \mathcal{D}$.

In equation (1), A_D denotes the consistency of the distance function D. A consistent distance function is one in which objects in the same relative positions, but in different rows, vary in the same manner (e.g., the first and the second object in each row are always the same distance apart for D, regardless of the row). It is assumed that the more consistent a distance function is (i.e., values closer to 1 for A_D), the more useful that function is for solving the task. Conversely, the more inconsistent a distance function is (i.e., values closer to 0 for A_D), the less useful that function is for solving the task. Thus, A_D acts as a task-specific weight, allowing the robot to identify the distance functions that vary in the most consistent way in the matrix. It should be noted, though, that the objective function q, as defined in equation (1), only evaluates patterns across the rows and not down the columns. In section V-D we will extend this function to also evaluate patterns down the columns of $\mathcal{M}$.

The expected value $\mathbf{E}[D(\mathcal{M}_{n,j}, \widehat{\mathcal{M}}_{n,n})]$ is computed as follows:

$$\mathbf{E}[D(\mathcal{M}_{n,j}, \widehat{\mathcal{M}}_{n,n})] = \frac{1}{n-1} \left[\sum_{i=1}^{n-1} D(\mathcal{M}_{i,j}, \mathcal{M}_{i,n}) \right]. \quad (2)$$

Intuitively, this is simply the average distance computed using D between pairs of objects in the same relative positions in every row except the last one.

The consistency, A_D, of a distance function D with respect to a matrix $\mathcal{M}$ is measured as:

$$A_D = \prod_{a=1}^{n-2} \prod_{b=a+1}^{n-1} A_D^{a,b}. \quad (3)$$

In equation (3) $A_D^{a,b}$ is the consistency between rows a and b, which is defined as:

$$A_D^{a,b} = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} h(\left| D(\mathcal{M}_{a,i}, \mathcal{M}_{a,j}) - D(\mathcal{M}_{b,i}, \mathcal{M}_{b,j}) \right|), \quad (4)$$

where h is the consistency function. Thus, A_D measures how often D agrees with itself for two pairs of objects in the same relative positions but in different rows. The consistency function h is defined as[2]:

$$h(x) = 1 - \log_2(x + 1).$$

To summarize, in order to solve the matrix completion task, the robot selects the object O_k in $\mathcal{G}$ that minimizes the squared difference between that object and the expectation based on the consistency of the distance functions in $\mathcal{D}$ with respect to the matrix $\mathcal{M}$.

The asymptotic running time to compute this objective function for a given matrix $\mathcal{M}$ and a given candidate object O_k is: $O\left(|\mathcal{D}| \times n^4\right)$, where $|\mathcal{D}|$ is the size of the set of distance functions and n is the size of the matrix (that is, a square matrix with n rows and n columns). The $|\mathcal{D}|$ term is due to the first summation in equation (1) over the set of distance functions. The term n^4 is due to the computation of A_D. Computing each sub-term $A_D^{a,b}$ takes $O(n^2)$ time and there are $O(n^2)$ of them, thus it takes $O(n^4)$ time to compute A_D. It takes $O(n^2)$ to compute the inner summation in equation (1), but the computation of A_D dominates the running time of the function. For relatively small n (in this paper $n = 3$), the running time is fairly short.

D. Extending the Methodology to Columns

The methodology described so far only considers relationships between objects in the same row. In most common matrix completion tests, however, the relationships between the objects down the columns are just as important for solving the tasks. One way to alter the methodology to work with column-wise relationships is to simply transpose the matrix and use the same objective function, i.e., evaluate $q(O_k, \mathcal{M}^T, \mathcal{D})$. Since the transpose operator flips the columns and the rows, the modified objective function

[2]The consistency function was empirically determined based on the condition that its output should be maximized when given a minimal disagreement value (i.e., $x = 0$) and its output should be minimized when given a maximal disagreement value (i.e., $x = 1$).

evaluates the relationships between the objects down the columns.

It is not enough, though, to just evaluate the relationships down the columns or across the rows of a matrix independently. Most matrix reasoning tests require that the test taker be able to combine these two together in order to pick the correct answer. In order to do this, we define the super objective function Q to be equal to:

$$Q(O_k, \mathcal{M}, \mathcal{D}) = q(O_k, \mathcal{M}, \mathcal{D}) + q(O_k, \mathcal{M}^T, \mathcal{D}).$$

In other words, Q simply sums the values of the objective function q evaluated for two different matrices, $\mathcal{M}$ and $\mathcal{M}^T$, using the same object O_k and the same set of distance functions $\mathcal{D}$. In this way, the best candidate object is defined as the one that best approximates the relationships between the objects in the matrix down the columns *and* across the rows.

E. Measuring Object Similarity

Four different methods for generating the set of distance functions were evaluated. The first was simply the Euclidean distance between the features for each object in each sensorimotor context. The second used spectral clustering to group the objects into labeled categories and then measured the distance between the objects based on their category memberships. The third was an extension of the first; it added supervision to the context distances in an attempt to improve the performance. The fourth method was similar to the third; it added supervision to the second method in an attempt to improve performance.

1) Context Distance Measurements: One distance function D_c was computed for each sensorimotor context $c \in \mathcal{C}$. Given two objects, O_i and O_j, the output of D_c is defined as:

$$D_c(O_i, O_j) = \mathbf{E}\Big[\|\mathbf{x}_a - \mathbf{x}_b\| \Big| \mathbf{x}_a \in \mathcal{X}_i^c, \mathbf{x}_b \in \mathcal{X}_j^c\Big],$$

where $\|\mathbf{x}_a - \mathbf{x}_b\|$ is the L2-norm distance between $\mathbf{x}_a$ and $\mathbf{x}_b$ and $\mathcal{X}_i^c$ and $\mathcal{X}_j^c$ are two sets of feature vectors for context c for O_i and O_j respectively (recall that the robot repeated the same behavior multiple times on each object). This expectation is estimated by:

$$D_c(O_i, O_j) = \frac{1}{|\mathcal{X}_i^c| \times |\mathcal{X}_j^c|} \sum_{\mathbf{x}_a \in \mathcal{X}_i^c} \sum_{\mathbf{x}_b \in \mathcal{X}_j^c} \|\mathbf{x}_a - \mathbf{x}_b\|.$$

To mitigate the effect of outliers, the output of each distance function D_c was normalized to be between 0 and 1 using the logistic function, $\frac{1}{1+e^{-x}}$, with the middle two quartiles of the comparisons falling in the range $[0.1, 0.9]$. The resulting set $\mathcal{D}$ contained exactly one distance function D_c for each context $c \in \mathcal{C}$.

This method for computing context distance measurements is identical to the one used in our previous work. For more details see [15].

2) Category Distance Measurements: As before, one distance function D_c was computed for each sensorimotor context $c \in \mathcal{C}$. For each context, the spectral clustering algorithm [64] was used to recursively cluster the objects[3] into a set of categories. Given a set of categories, the output of D_c was computed as

$$D_c(O_i, O_j) = 1 - I(label_c(O_i) \equiv label_c(O_j)),$$

where $label_c(O_i)$ is the category label for O_i in context c and I is the indicator function, which is 1 if its argument is true and 0 otherwise. Intuitively, the output of the distance function is 0 if the two objects belong to the same category in a specific sensorimotor context and 1 if they don't.

3) Context Distance Measurements with Supervision: In this and the next method, we added supervision by giving the robot a set of training examples. In our previous work ([15], [16]), the robot learned by weighting each context (analogous to the distance functions used here) based on the individual performance of each context on a training set of tasks. In both of those papers we found that certain individual contexts performed near perfectly on certain types of tasks on their own because the objects in them largely varied by a single property. In this paper, however, the objects vary by multiple properties, and as a result we empirically determined that similar weighting schemes would not work because no individual distance function performed even moderately well by itself on the training set of tasks. Thus, we developed a new algorithm as a solution to this problem.

This method builds on the context distances method. Given the set of distance functions $\mathcal{D}$ computed using that method and a set $\mathcal{L} = \{(\mathcal{M}_1, \mathcal{G}_1), ..., (\mathcal{M}_n, \mathcal{G}_n)\}$ of training matrix completion tasks, for which the correct answers are known, the robot attempted to find the set $\mathcal{D}' \subseteq \mathcal{D}$ that maximized performance on the training set. Since there are an exponential number of subsets, finding the best one in the general case is intractable. To approximate the correct solution, the robot, starting with $\mathcal{D}'_0 = \mathcal{D}$, attempted to prune $\mathcal{D}'_0$ down to only the most useful distance functions. To do this, the robot iteratively removed the worst performing distance function. That is, at iteration n, the robot computed $\mathcal{D}'_{n+1}$ as

$$\mathcal{D}'_{n+1} = \mathcal{D}'_n - \left\{ \underset{D \in \mathcal{D}'_n}{\operatorname{argmin}} \ \text{performance}(\mathcal{D}'_n - \{D\}, \mathcal{L}) \right\}$$

where the performance function measures how well the given set of distance functions perform on the given training tasks. This process was repeated until $|\mathcal{D}'_n| = 0$. The robot then selected the subset that performed the best on the training tasks, i.e., $\mathcal{D}' = \mathcal{D}'_k$ where $k = \max_n \text{performance}(\mathcal{D}'_n, \mathcal{L})$. This algorithm is shown in more detail in Figure 10.

This algorithm relies on the assumption that some of the computed distance functions are not useful for solving matrix

[3]The spectral clustering algorithm requires a distance function between the data points in order to cluster them. Since the robot created a separate clustering for each sensorimotor context, it computed the distance between each pair of objects in each context in the same way as described in section V-E1.

```
function PRUNE(𝒟, ℒ)
    𝒟'[ ] ← emptyArray
    count ← 0
    𝒟'[count] ← 𝒟
    while |𝒟'[count]| > 1 do
        bestPerformance ← 0
        bestSet ← null
        for all D ∈ 𝒟'[count] do
            set ← 𝒟'[count] − {D}
            p ← evaluatePerformance(set, ℒ)
            if p > bestPerformance then
                bestPerformance ← p
                bestSet ← set
            end if
        end for
        𝒟'[count + 1] ← bestSet
        count ← count + 1
    end while
    bestPerformance ← 0
    bestSet ← null
    for i ← 0 to length(𝒟') − 1 do
        p ← evaluatePerformance(𝒟'[i], ℒ)
        if p > bestPerformance then
            bestPerformance ← p
            bestSet ← 𝒟'[i]
        end if
    end for
    return bestSet
end function
```

Fig. 10: The algorithm that prunes the set of distance functions. It takes as input an initial set of distance functions $\mathcal{D}$ and a set of training tasks $\mathcal{L}$ for which the correct answers are known. The method $evaluatePerformance$ returns the accuracy of the given set of distance functions on the given training tasks.

completion tasks. Unlike the variable A_D in the objective function (equation (1)), which weights each distance function based on its consistency for an *individual* task, this algorithm attempts to globally prune distance functions. That is, it considers *all* the training tasks at once and removes distance functions that underperform across all of them, rather than on an individual task level.

4) Category Distance Measurements with Supervision: This method is similar to the previous method except that it builds on the category distances method rather than the context distances method. The robot first uses the spectral clustering algorithm to cluster the set of objects into a set of categories, one set for each context. Next, given the set of distance functions $\mathcal{D}$ computed from these sets of categories (as described in Section V-E2) and a set of matrix completion tasks for training $\mathcal{L} = \{(\mathcal{M}_1, \mathcal{G}_1), ..., (\mathcal{M}_n, \mathcal{G}_n)\}$, for which the correct answers are known, the robot again attempted to prune the set of distance functions down to only the most useful. Similar to the last method, it did this by iteratively removing the worst performing distance function from the set until it found the best subset of functions. It used the same algorithm as before, which is described in Figure 10.

F. Evaluation

The robot was evaluated on the set of objects described in section III-B, which vary by *color*, *contents*, and *weight*. The values for *color* are *red*, *blue*, and *green*. The values for *contents* are *glass*, *screws*, *beans*, and *rice*. The values for *weight* are *light*, *medium*, and *heavy*. It should be noted that the robot was never given these values during the evaluation.

We randomly generated 500 matrix completion tasks using the methodology described in section V-B. The robot then generated each of the four sets of distance functions described in section V-E. Each set of distance functions was evaluated independently. Because some distance functions required training, we performed 10-fold cross-validation across the matrix completion tasks. That is, we split the 500 tasks into 10 equally sized groups, trained the robot on 9 of the 10 groups, and tested it on the remaining one. This process was repeated for each group. It is worthwhile to note that the robot was never given *a priori* knowledge of the patterns used to generate the matrix completion tasks. Rather, the only supervision it was given was in the form of example matrix reasoning tasks.

Performance is reported as accuracy or kappa. The accuracy is computed as

$$\%Accuracy = \frac{\#correct\ answers}{\#total\ tasks} \times 100.$$

We also wanted to know how the robot performs when varying the number of candidate objects to choose from. Since chance accuracy depends on the number of candidate objects in the task, the kappa score was computed to compensate for varying degrees of chance. Cohen's kappa statistic [65] was computed as follows:

$$kappa = \frac{P(a) - P(e)}{1 - P(e)},$$

where $P(a)$ is the performance of the robot's model and $P(e)$ is chance accuracy. This allows the direct comparison of results where chance accuracy may differ.

The evaluation was performed off-line after the robot interacted with all 36 objects.

VI. Results

A. Performance on a Single Task

Figure 11 illustrates one of the 500 matrix completion tasks that the robot solved. The objective function values for each of the 8 candidate objects are shown for both the context and category distance functions with and without supervision. The matrix in the figure exhibits the patterns *constant:color* and *decrement:weight* across its rows and *permutation:color* and *constant:weight* down its columns. Given this, it can be deduced that the missing object must be *light* and *red*. The only candidate object that has both of these property values is (g), which is the correct answer. In this case, the property *contents* was irrelevant to the task.

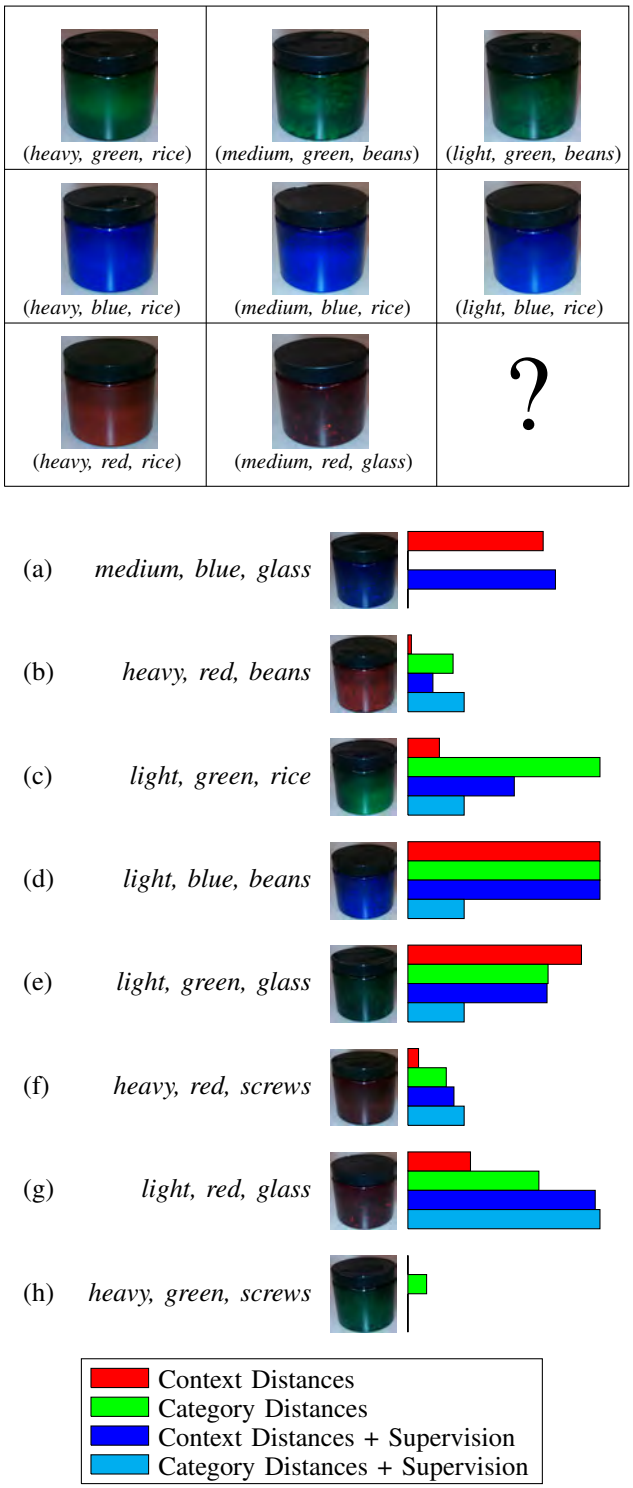


Fig. 11: An example matrix reasoning task solved by the robot as part of this experiment. The words below each object in the matrix represent the values for each of the three properties for that object. The bars next to each candidate object represent the normalized objective function values for each of the four distance methods.

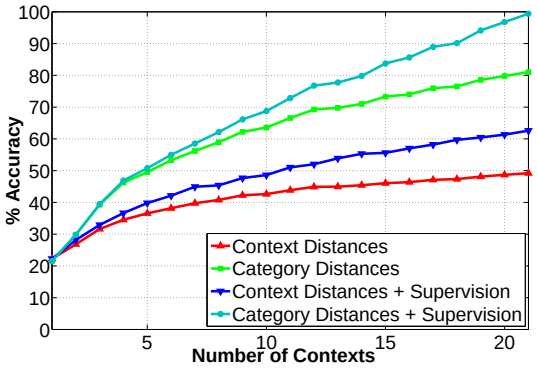


Fig. 12: Accuracy versus number of contexts used to solve the tasks. As expected, the accuracy improves as the robot is allowed to use information from more sensorimotor contexts. Each line represents a different distance function method. The two category distance methods perform better than the two context distance methods. As expected, the category method with supervision performs the best.

The objective function values were computed using all 21 contexts. The context distance method ranked three candidate objects ((d), (e), and (a)) higher than the correct answer (g). The category distance method performed about the same, ranking (c), (d), and (e) higher than the correct answer. Thus, both of the unsupervised methods failed to pick the correct answer.

The two supervised[4] methods, however, performed better. The supervised context distances method ranked the correct object, (g), in second place after (d). The supervised category distance method picked the correct answer, ranking (g) higher than any other candidate. It selected the distance functions computed from the contexts *lift-audio*, *hold-audio*, *rattle-audio*, *push-audio*, *rattle-proprioception*, and *look-color*. This indicates, as expected, that not all contexts are useful for solving this type of matrix completion task. This suggests that methods that use supervision to prune the contexts to the most useful ones could perform better. The next section expands on this by looking at performance over all 500 tasks.

B. Performance Across All Tasks

Figure 12 compares the accuracy of all four methods of measuring distances. It shows that the robot's accuracy on matrix completion tasks improves when it is given access to more information in the form of sensorimotor contexts. It is interesting to note that both of the category distances methods (with and without supervision) perform the best. This suggests that features derived from category labels are

[4]For the example task, both of the supervised methods were trained on 450 randomly selected tasks from the set of 500 generated for this paper. In other words, they were trained on 9 of the 10 folds. The example task shown in Figure 11 was not included in that training set.

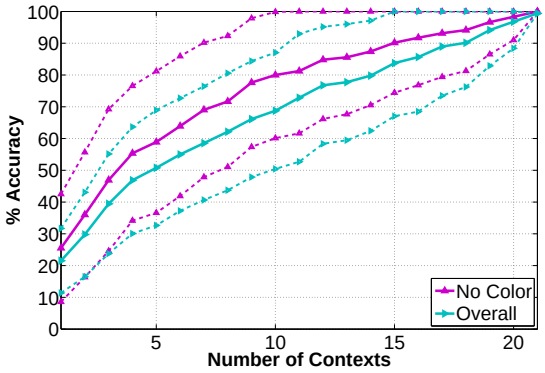


Fig. 13: Accuracy versus number of contexts for the subset of tasks that don't require color information and for all tasks. The line labeled "Overall" is the same as the line labeled "Category Distances + Supervision" in Figure 12. The other line was computed in the exact same way as the overall line, except the 500 tasks were reduced to just the 173 that did not require perception of color to solve. The standard deviation for each data point is also plotted using dashed lines.

more useful for this kind of task. Additionally, for both category and context distances, the method with supervision always outperforms its unsupervised counterpart. The best performing method was the category method with supervision. When given access to all 21 contexts, it was able to achieve 99.4% accuracy on the testing set of matrices. That is, using all available information and the best performing method, the robot was able to determine the correct answer to all but 3 of the 500 problems that were presented to it.

As described in section III-D, there was only one context, *look-color*, that had access to visual data collected from the robot's camera. Because color is so important to solving the tasks, we wanted to know how this affected the robot's performance. Figure 13 shows the robot's performance on only the matrix completion tasks that did not require the perception of color to solve (173 out of 500) as compared to the robot's performance on all 500 tasks. On average the robot performs better when the task does not involve color, especially in the middle part of the graph (5 to 15 contexts). It is also worth mentioning that the upper limit of the standard deviation converges to 100% accuracy sooner for tasks not involving color than for all tasks. Just as we expected, because there are no redundant contexts in which *color* can be perceived (as opposed to *weight* and *contents*), the robot has a harder time identifying color as a relevant property, and thus tasks that require it are harder to solve.

C. Performance Compared to Difficulty of the Task

Figure 14 shows six figures that compare the robot's performance for different types of task difficulty. Figure 14a shows the robot's performance as a function of the number of candidate objects that it can choose from to complete the matrix. It is interesting to note that, even though the scores are reported as the kappa value to compensate for different chance accuracies, the robot still performs better when given fewer options to pick from than when given more. Conversely, Figure 14b shows that when the number of patterns present in the matrix increases, the robot gets better at solving the task. Interestingly, even though Figure 14b was computed using the context distances method without supervision (as opposed to the category distances method with supervision as in all the other figures[5]), it was still possible to achieve 100% accuracy on matrices with 6 patterns. Intuitively this makes sense because the more patterns present in a matrix, the more constrained the possible candidate objects are, and thus the easier the task is to solve.

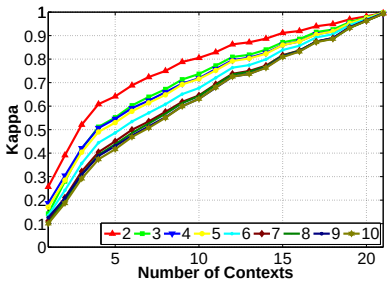
Figure 14c shows the robot's performance when the objective function was computed only across the rows of the matrix in each task, down the columns, or both. As expected, the robot is able to perform better when using an objective function that takes into account the information across the rows and down the columns. Also, as expected, the robot's performance when only using rows or only using columns is approximately the same because the task generation algorithm isn't biased towards rows or columns.

Figures 14d, 14e, and 14f show the robot's performance on different subsets of the matrix completion tasks. Figure 14d shows the performance on tasks that include at least one instance of each of the different rules; Figure 14e shows the robot's performance on tasks that contain each of the different patterns; and Figure 14f shows the robot's performance on tasks including at least one pattern over each of the three different properties of the objects.

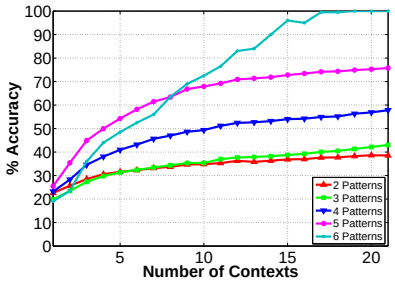
Figure 14e shows that the robot performed better on tasks in which the matrix had at least one pattern that was over the property *weight*. This is confirmed by Figure 14f, which shows the line for *weight* to be higher than the other two. Figure 14d also shows that the robot performs better when the rules *increment* and *decrement* are included, which, as stated in section V-B, are applied exclusively to *weight*. This indicates that, overall, the robot performed better on tasks that required it to perceive the weight of the objects. Intuitively, this makes sense because for many of the behaviors the robot was supporting or moving the full weight of the object, meaning the data collected from the proprioceptive modality often contained information about the weight of the object. Conversely, only a few of the behaviors caused the contents of the objects to shift and register a sound that the robot could detect with its microphones, and only one behavior (*look*) was used to extract color features. Thus, as the number of contexts available to the robot is increased, it is more likely that a context that can reliably perceive *weight* will be selected, which would improve the robot's performance on tasks that involve *weight*.

Additionally, the results shown in Figure 14 suggest that the robot tends to perform better when the task is more

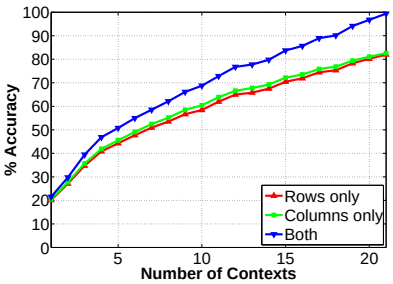
[5]This was done because in the version of this graph that used the category distances method with supervision there was no significant difference between the five lines, which reached near 100% accuracy when given access to all contexts.



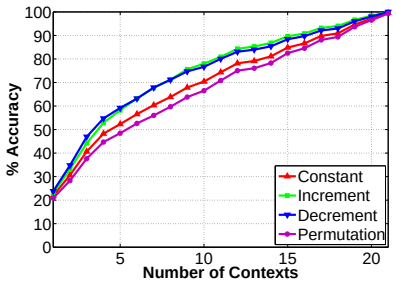
(a) Each line represents the performance when a different number of candidate objects were given to the robot to choose from. Each datapoint was computed using the category distances method with supervision. The vertical axis, unlike in the rest of the figures in this paper, represents the kappa value rather than accuracy in order to compensate for the change in chance accuracy for each line.



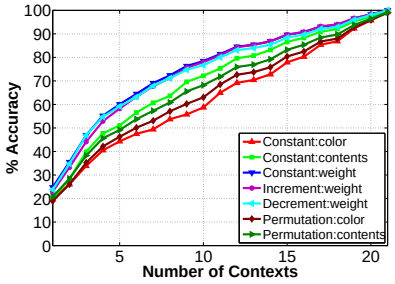
(b) Accuracy improves as the number of patterns present in each matrix increases from 2 to 6. Each line was computed using the context distances method without supervision over only the matrix reasoning tasks with that number of patterns. Figure 15a shows the overall number of matrix reasoning tasks with different number of patterns.



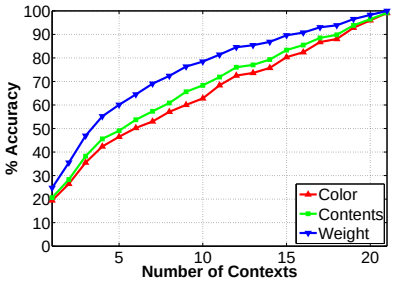
(c) Each of the three lines was computed using the category distances method with supervision by only allowing the robot to compute the objective function over the rows of the matrix, over only the columns of the matrix, or both.



(d) Each line was computed using the category distances method with supervision over only the tasks that had at least one instance of the corresponding rule (e.g., the red line was computed using only tasks that contained the rule *constant*). See also Figure 15b.

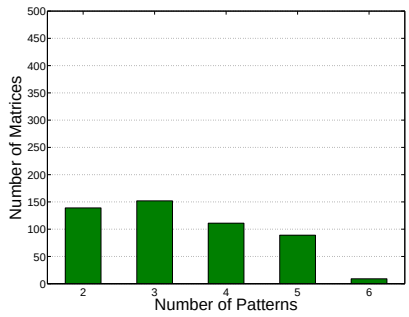


(e) Each line was computed using the category distances method with supervision over only the tasks that had the corresponding pattern present (e.g., the increment:weight line was computed only over tasks that contained the pattern *increment:weight*).

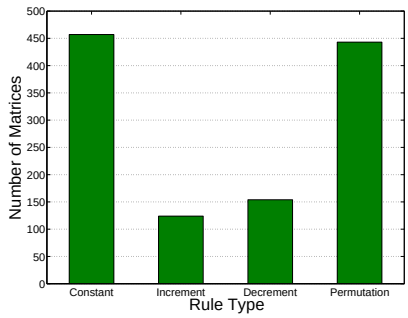


(f) Each line was computed using the category distances method with supervision over only the tasks that had at least one pattern over the corresponding property (e.g., the color line was computed only over tasks that contained at least one pattern over *color*). See also Figure 15c.

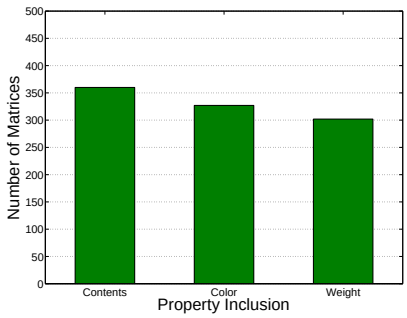
Fig. 14: Six figures that compare the performance as a function of the difficulty of the matrix completion tasks.



(a) The number of tasks that have 2 to 6 patterns.



(b) The number of tasks that have at least one instance of each of the four rules.



(c) The number of tasks that include at least one pattern over each object property.

Fig. 15: Three figures that show the number of matrix completion tasks for different task difficulty types.

constrained (either in the form of fewer candidate objects to choose from or more patterns present in the matrix). While this was not entirely unexpected, it was surprising to find that even the worst performing distance function method (context distances without supervision) was able to achieve 100% accuracy on tasks with 6 patterns when given enough contexts (see Figure 14b).

Figure 15 shows the number of matrix completion tasks for three different types of difficulty. There are many interdependencies between the patterns. This is illustrated in Figure 15a, which shows that, despite the fact that the task generation algorithm only generates tasks with 2 patterns, most tasks have more than 2 patterns. Also, Figure 15b shows that the *constant* and *permutation* rules tend to appear in tasks much more frequently than the *increment* and *decrement* rules. These interdependencies often mean that matrix completion tasks have redundant information. This is not the case for the properties of the objects, though, as Figure 15c shows that the distribution over tasks that include at least one pattern for each property is approximately uniform.

It is worthwhile to note that in Figure 15b the counts for the different rule types are far from uniformly distributed, and even in Figure 15a the counts are not uniform. This is due to the interdependencies between patterns. For example, the only way for a 3×3 matrix to have the *increment:weight* pattern present across the rows is for the first object in each row to be *light*, the second to be *medium*, and the third to be *heavy*. Since every row must have those values in order for *increment:weight* to be present across the rows, then that necessarily implies that all the weights are constant down each column, i.e., the pattern *constant:weight* is always present in the columns when *increment:weight* is present in the rows.

VII. Conclusion and Future Work

In this paper we proposed a framework for solving matrix completion tasks. The robot was tested on matrices composed of objects that varied by *contents*, *color*, and *weight*. It was able to gather information about the objects by interacting with them while simultaneously recording from multiple sensory modalities. We then generated a set of 500 matrix completion tasks using those objects and posed them to the robot. Using all 21 sensorimotor contexts and the best distance method, it was able to achieve 99.4% accuracy on the set of tasks. That is, it was able to pick the correct answer for all but 3 of the 500 tasks.

The tasks posed in this paper utilized objects that varied by multiple, independent properties. Because of this, the robot was required to synthesize information from multiple sensorimotor contexts in order to solve the tasks. In our previous work [15], [16] we found that when the tasks utilized objects that vary largely by a single property, a single, well-picked context is sufficient to solve the tasks with a high degree of accuracy. In this paper we extended the framework developed in [14], [15], [16] (see Figure 3) and showed that it can be used to solve complex tasks that require the perception of multiple properties.

The primary difference between our previous work [15], [16] and this paper is that in this paper the robot was required to perceive multiple patterns simultaneously using different sensory modalities and to integrate that information in order to solve each task. In contrast, the tasks in our previous work varied mostly by a single feature and along a single pattern. In other words, this paper showed that the general framework outlined in Figure 3 scales to even more complex and multivariate problems.

Overall, the robot was able to successfully solve a variety of matrix completion tasks using grounded, sensorimotor information. In previous work, it has been shown that robots can use exploratory behaviors to solve tasks such as object recognition [10], odd-one-out [14], object pairing [16], order completion [15], and now matrix completion. This shows that robots that use exploratory behaviors and ground their knowledge in their own sensorimotor contexts can not only perceive useful information about objects, but also can use that information to solve a variety of tasks.

One interesting avenue for future work would be to extend the robot's ability to learn more about the objects involved in the tasks. For example, a robot could be trained to recognize different categories of objects and then use that information to bootstrap its ability to solve matrix completion tasks. This could be done using current supervised category recognition methods. It could also be done in an unsupervised manner that is augmented by environmental cues, such as verbal cues provided by a human, in order to help direct the robot to better align its categories with human provided ones.

Acknowledgement

The research presented in this paper was partially supported by the National Science Foundation Graduate Research Fellowship (NSF Grant No. DGE1247194).

References

[1] J. C. Raven, *Progressive matrices*. Éditions scientifiques et psychotechniques, 1938.

[2] J. Raven, "The Raven's progressive matrices: Change and stability over culture and time," *Cognitive psychology*, vol. 41, no. 1, pp. 1–48, 2000.

[3] V. Prabhakaran, J. Smith, J. Desmond, G. Glover, and J. Gabrieli, "Neural substrates of fluid reasoning: an FMRI study of neocortical activation during performance of the Raven's progressive matrices test," *Cognitive psychology*, vol. 33, pp. 43–63, 1997.

[4] D. Wechsler, *Wechsler Adult Intelligence Scale third edition: Administration and scoring manual*. San Antonio, TX: Psychological Corporation, 1997.

[5] D. Watt, "Lionel Penrose, FRS (1898–1972) and eugenics: Part one," *Notes and Records of the Royal Society of London*, vol. 52, no. 1, pp. 137–151, 1998.

[6] E. Scerri, *The Periodic Table: A Very Short Introduction*. Oxford: Oxford University Press, 2011.

[7] L. Fuchs, D. Fuchs, K. Stuebing, J. Fletcher, C. Hamlett, and W. Lambert, "Problem solving and computational skill: Are they shared or distinct aspects of mathematical cognition?" *Journal of educational psychology*, vol. 100, no. 1, p. 30, 2008.

[8] A. Dugbartey, P. Sanchez, J. Rosenbaum, R. Mahurin, J. Davis, and B. Townes, "WAIS-III matrix reasoning test performance in a mixed clinical sample," *The Clinical Neuropsychologist*, vol. 13, no. 4, pp. 396–404, 1999.

[9] K. Richardson, "Reasoning with Raven-In and out of context," *British Journal of Educational Psychology*, vol. 61, no. 2, pp. 129–138, 1991.

[10] J. Sinapov, T. Bergquist, C. Schenck, U. Ohiri, S. Griffith, and A. Stoytchev, "Interactive object recognition using proprioceptive and auditory feedback," *The Intl. J. of Robotics Research*, vol. 30, no. 10, pp. 1250–1262, 2011.

[11] J. Sinapov and A. Stoytchev, "Object category recognition by a humanoid robot using behavior-grounded relational learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 184–190.

[12] ——, "The boosting effect of exploratory behaviors," in *Proc. of the 24th National Conference on Artificial Intelligence (AAAI)*, 2010, pp. 1613–1618.

[13] J. Sinapov, C. Schenck, K. Staley, V. Sukhoy, and A. Stoytchev, "Grounding semantic categories in behavioral interactions: Experiments with 100 objects," *Robotics and Autonomous Systems (to appear)*.

[14] J. Sinapov and A. Stoytchev, "The odd one out task: Toward an intelligence test for robots," in *Proceedings of the 9th IEEE International Conference on Development and Learning (ICDL)*, 2010, pp. 126–131.

[15] C. Schenck, J. Sinapov, and A. Stoytchev, "Which object comes next? Grounded order completion by a humanoid robot," *Journal of Cybernetics and Information Technologies*, vol. 12, no. 3, pp. 5–16, 2012.

[16] C. Schenck and A. Stoytchev, "The object pairing and matching task: Toward Montessori tests for robots," in *Proceedings of the 2012 Humanoids Workshop on Developmental Robotics, Osaka, Japan*, 2012.

[17] P. Carpenter, M. Just, and P. Shell, "What one intelligence test measures: a theoretical account of the processing in the Raven's progressive matrices test." *Psychological review*, vol. 97, no. 3, pp. 404–431, 1990.

[18] D. Little, S. Lewandowsky, and T. Griffiths, "A Bayesian model of rule induction in Raven's progressive matrices," in *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, 2012, pp. 1918–1923.

[19] C. Darwin, *The Descent of Man and Selection in Relation to Sex*. New York: A.L. Burt, 1871.

[20] A. Kinnaman, "Mental life of two macacus rhesus monkeys in captivity," *The American Journal of Psychology*, vol. 13, no. 1, pp. 98–148, 1902.

[21] G. J. Romanes, *Animal intelligence*. London, Great Britain: Kegan Paul, Trench & Co, 1882.

[22] J. R. Slonaker, "The normal activity of the albino rat from birth to natural death, its rate of growth and the duration of life," *J. anim. Behav*, vol. 2, pp. 20–42, 1912.

[23] W. S. Small, "Notes on the psychic development of the young white rat," *The American Journal of Psychology*, vol. 11, no. 1, pp. 80–100, 1899.

[24] K. Lorenz, "Innate bases of learning," in *Learning as Self-Organization*, K. H. Pribram and J. King, Eds. Mahwah, NJ: Lawrence Erlbaum Pub., 1996.

[25] T. Power, *Play And Exploration in Children And Animals*. Mahwah, NJ: Lawrence Erlbaum Associates, Publishers, 2000.

[26] S. E. Glickman and R. W. Sroges, "Curiosity in zoo animals," *Behaviour*, pp. 151–188, 1966.

[27] G. C. Westergaard, "Object manipulation and the use of tools by infant baboons (papio cynocephalus anubis)," *Journal of Comparative Psychology*, vol. 106, no. 4, pp. 398–403, 1992.

[28] ——, "Development of combinatorial manipulation in infant baboons (papio cynocephalus anubis)," *Journal of comparative psychology*, vol. 107, no. 1, pp. 34–38, 1993.

[29] I. Inglis and D. Shepherd, "Rats work for food they then reject: Support for the information-primacy approach to learned industriousness," *Ethology*, vol. 98, no. 2, pp. 154–164, 1994.

[30] M. E. Verbeek, P. J. Drent, and P. R. Wiepkema, "Consistent individual differences in early exploratory behaviour of male great tits." *Animal Behaviour*, vol. 48, no. 5, pp. 1113–1121, November 1994.

[31] A. Weisler and R. B. McCall, "Exploration and play: Resume and redirection," *American Psychologist*, vol. 31, pp. 492–508, 1976.

[32] J. Piaget, *The Origins of Intelligence in Children*. New York: International Universities Press, 1952.

[33] E. J. Gibson, "Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge," *Annual review of psychology*, vol. 39, no. 1, pp. 1–42, 1988.

[34] J. Vauclair and K. A. Bard, "Development of manipulations with objects in ape and human infants," *Journal of Human Evolution*, vol. 12, no. 7, pp. 631–645, 1983.

[35] L. J. Yarrow, S. McQuiston, R. H. MacTurk, M. E. McCarthy, R. P. Klein, and P. M. Vietze, "Assessment of mastery motivation during the first year of life: Contemporaneous and cross-age relationships." *Developmental Psychology*, vol. 19, no. 2, p. 159, 1983.

[36] D. J. Messer, M. E. McCarthy, S. McQuiston, R. H. MacTurk, L. J. Yarrow, and P. M. Vietze, "Relation between mastery behavior in infancy and competence in early childhood," *Developmental Psychology*, vol. 22, no. 3, pp. 366–372, 1986.

[37] D. J. Messer, D. Rachford, M. McCarthy, and L. Yarrow, "Assessment of mastery behavior at 30 months: Analysis of task-directed activities." *Developmental Psychology*, vol. 23, no. 6, p. 771, 1987.

[38] H. A. Ruff and K. Dubiner, "Stability of individual differences in infants'manipulation and exploration of objects," *Perceptual and Motor Skills*, vol. 64, no. 3c, pp. 1095–1101, 1987.

[39] D. A. Caruso, "Dimensions of quality in infants' exploratory behavior: Relationships to problem-solving ability," *Infant Behavior and Development*, vol. 16, no. 4, pp. 441–454, 1993.

[40] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action-initial steps towards artificial cognition," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, 2003, pp. 3140–3145.

[41] R. Brooks, C. Breazeal, R. Irie, C. C. Kemp, M. Marjanovic, B. Scassellati, and M. Williamson, "Alternate essences of intelligence," in *15-th National Conference on Artificial Intelligence (AAAI-98)*, 1998, pp. 961–968.

[42] E. Krotkov, R. Klatzky, and N. Zumel, "Robotic perception of material: Experiments with shape-invariant acoustic measures of material type," in *Experimental Robotics IV*. Springer, 1997, pp. 204–211.

[43] E. Torres-Jara, L. Natale, and P. Fitzpatrick, "Tapping into touch," in *Proceedings of the Fifth International Workshop on Epigenetic Robotics*, Osaka, Japan, 2005.

[44] L. Natale, G. Metta, and G. Sandini, "Learning haptic representation of objects," in *Proc. of the Intl. Conf. on Intelligent Manipulation and Grasping*, 2004.

[45] A. Stoytchev, "Some basic principles of developmental robotics," *IEEE Transactions on Autonomous Mental Development*, vol. 1, no. 2, pp. 122–130, 2009.

[46] S. Griffith, Sinapov, V. J., Sukhoy, and A. Stoytchev, "A behavior-grounded approach to forming object categories: Separating containers from non-containers," *IEEE Transactions on Autonomous Mental Development*, vol. 4, no. 1, pp. 54–69, 2012.

[47] V. Chu, I. McMahon, L. Riano, C. McDonald, Q. He, J. Perez-Tejada, M. Arrigo, N. Fitter, J. Nappo, T. Darrell, and K. Kuchenbecker, "Using robotic exploratory procedures to learn the meaning of haptic adjectives," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[48] D. Xu, G. Loeb, and J. Fishel, "Tactile identification of objects using bayesian exploration," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[49] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, "Cognitive developmental robotics: A survey," *IEEE Transactions on Autonomous Mental Development*, vol. 1, no. 1, pp. 12–34, 2009.

[50] A. Lovett, K. Forbus, and J. Usher, "A structure-mapping model of Raven's progressive matrices," in *Proc. 32nd Annual Meeting of the Cognitive Science Society*, vol. 10, 2010, pp. 2761–2766.

[51] D. Rasmussen and C. Eliasmith, "A neural model of rule generation in inductive reasoning," *Topics in Cognitive Science*, vol. 3, no. 1, pp. 140–153, 2011.

[52] M. Kunda, K. McGreggor, and A. Goel, "Taking a look (literally!) at the Raven's intelligence test: Two visual solution strategies," in *Proc. 32nd Annual Meeting of the Cognitive Science Society*, 2010.

[53] S. Cirillo, "An anthropomorphic solver for Raven's progressive matrices," Master's thesis, Chalmers University of Technology, Göteborg, Sweden, 2010.

[54] S. Nolfi and D. Marocco, "Active perception: A sensorimotor account of object categorization," in *From Animals to Animats: 7*, 2002, pp. 266–271.

[55] N. Mavridis and D. Roy, "Grounded situation models for robots: Bridging language, perception, and action," in *AAAI Workshop on Modular Construction of Human-Like Intelligence*, 2005, pp. 32–39.

[56] T. Nakamura, T. Nagai, and N. Iwahashi, "Multimodal object categorization by a robot," in *Proceedings of the IEEE/RSJ IROS*, 2007, pp. 2415–2420.

[57] S. Takamuku, K. Hosoda, and M. Asada, "Object category acquisition by dynamic touch," *Advanced Robotics*, vol. 22, no. 10, pp. 1143–1154, 2008.

[58] J. Sun, J. Moore, A. Bobick, and J. Rehg, "Learning visual object categories for robot affordance prediction," *The Intl. J. of Robotics Research*, vol. 29, no. 2-3, p. 174, 2010.
[59] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *Proceedings of the 7th IEEE International Conference on Development and Learning (ICDL)*, 2008, pp. 91–96.
[60] P. Forssén, D. Meger, K. Lai, S. Helmer, J. Little, and D. Lowe, "Informed visual search: Combining attention and object recognition," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 935–942.
[61] J. Sinapov, M. Wiemer, and A. Stoytchev, "Interactive learning of the acoustic properties of household objects," in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 2518–2524.
[62] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1–4.
[63] K. Lee, H. Hon, and R. Reddy, "An overview of the SPHINX speech recognition system," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 1, pp. 35–45, 1990.
[64] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.
[65] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

Connor Schenck (S09) received the M.S. degree in computer science and human-computer interaction in 2013 from Iowa State University, Ames. He is currently working towards a Ph.D. degree in computer science and engineering at the University of Washington, Seattle. His research interests include artificial intelligence, machine learning, robotics, and developmental robotics.

Jivko Sinapov (S'09) received the Ph.D. degree in computer science and human-computer interaction in 2013 from Iowa State University, Ames. He is currently a Postdoctoral Researcher and Instructor affiliated with ISU's Human-Computer Interaction program. His current research interests include developmental robotics, robotic perception, autonomous manipulation, and machine learning.

David Johnston is currently pursuing undergraduate degrees in Software Engineering and Mathematics at Iowa State University, Ames. His research interests are in the areas of Data Analysis, Bayesian Statistics, Developmental Robotics, Functional Programming, and Programmable User Interface Design.

Alexander Stoytchev (S'00-M'07) received the M.S. and Ph.D. degrees in computer science from the Georgia Institute of Technology, Atlanta in 2001 and 2007, respectively.

He is currently an Assistant Professor of Electrical and Computer Engineering and Director of the Developmental Robotics Laboratory at Iowa State University, Ames. His research is in the areas of Developmental Robotics, Autonomous Robotics, Computational Perception, and Machine Learning.