Learning to Predict the Controllability of Containers and their Contents

Project Proposal for CprE 585X

Shane Griffith

Abstract - Robotic control of the contents of a container has challenged roboticists for a number of years. Traditionally, roboticists have created complex control algorithms that are engineered to demands of the system. It may be possible, however, for a robot to autonomously learn about the controllability of the contents of a container.

The research objective of this proposal is to show that the same self-detection algorithms that are can identify self, and that can predict the controllability of self, also can be used to predict the controllability of blocks that are inside containers. In this experiment the robot will interact with 20 different objects and 5 different blocks. Optical flow will be used to track objects, self-detection will be used to determine their controllability, and deep learning will be used to learn a shared representation between objects in the visual field and their controllability. We hope to show that this learning paradigm can provide a robot with a behavior–grounded definition of the *inside* spatial relationship.

Introduction

Robotic control of the contents of containers has challenged roboticists for a number of years. The slosh-free control of liquids inside containers is a particularly active area of research [1][2][3]. Many papers have proposed complex control algorithms for addressing this problem. However, these algorithms are heavily engineered to the demands of a specific system; they are not designed to work under different operating conditions. For example, the parameters for the size of the container and the exact contents of the container have to be specified beforehand in order to compute an engineered solution. It is surprising that research toward predicting the controllability of arbitrarily sized containers and their contents is not yet forthcoming, given that a large amount of research has already addressed robotic control of many specific containers.

Clearly, an algorithm *does* exist for predicting the controllability of many different types of containers and their contents, because some children demonstrate it every chance they get (see Fig. 1). However, the algorithm that these children use remains unknown. What is known is that children undergo a long progression of learning in order to perfect their container–manipulation abilities. Infants play with containers incessantly in order to learn about them. It is through their play and exploration that they first learn what is controllable and later gain control over a container's contents.

a)  b) 

Fig. 1: Cookie monsters. a) Retrieving a jar of cookies from the cupboard. b) Fetching a freshly poured glass of milk from the table.

Similarly, it may be possible for robots that play with and explore objects in order to learn to predict the controllability of containers and their contents. Indeed, a few papers have shown that a behavior–babbling robot can learn to identify itself as a controllable object [4][5][6][7]. In these papers, the robot can learn how to control itself once it detects itself. Some of the robots also learned about the controllability of different objects [6][7]. Little work, however, has addressed how self-detection methods can allow a robot to predict the controllability of objects inside containers.

This research will address the question: How can a robot learn to predict the controllability of containers and their contents? The work described in this proposal is based on an extension of self-detection. The robot will explore 20 objects and 5 blocks by dropping a block above the object, grasping the object, and then waving the object around. An optical flow algorithm will be used to track every pixel in the robot's visual field during the *wave* behavior. Self-detection will measure the uncertainty with which each pixel moves with the robot. The result will be a *controllability image*, which identifies how much each pixel in the image is affected by the robot's movement. Finally, a deep-learning architecture will learn a shared representation between the visual image and the controllability image. The research will be evaluated using images of different objects and blocks in different spatial configurations. The robot should be able use its learned, shared representation to predict that it will have good control of containers and other objects, some control of blocks that are inside containers, no control of blocks that are outside containers. Also, we hope to show that this learning paradigm can provide a robot with a behavior–grounded definition of the *inside* spatial relationship.

Related Work

Developmental Psychology

Containers are one of the first abstract object categories that infants learn [8]. Their single distinguishing feature—their concave shape—makes them one of the simplest kinds of objects. However, it takes infants years of play and exploration before they master how to detect and use containers [9]. Their progression of learning for containers begins at around 2.5 months and continues well into adolescence, but it is around 9 months of age that infants begin to understand the controllability of objects placed inside containers [10].

Nine-month-old infants are fascinated with inserting things into containers. First they try inserting their hands into containers [10]. Later they try inserting blocks into containers, which they subsequently shake. This is age at which infants learn how their own body affects the movement of blocks inside containers. Their insert behavior peaks when infants are around 15 months old [10], which suggests that infants have spent months learning about how different objects move around in different containers.

In this paper the robot learned about the controllability of containers using a similar interaction strategy as that of 9-month-old infants. The robot first dropped a block above an object and then waved the object around randomly inside the visual field several times. The motion of the block was tracked as the robot waved the container back and forth. The

same algorithm used for self-detection was used to identify the controllability of blocks inside containers.

Robotics

Little research has addressed the controllability of the contents of containers. Research that uses similar ideas to the ones proposed here have, however, shown how a robot can learn to detect and control the tip of tools, which is useful for keeping containers in an upright orientation. Kemp and Edsinger [11] showed how a robot could autonomously learn to detect and control the tip of a tool, which the robot used to control the open end of a bottle and a cup (the robot also interacted with a brush). The robot grasped each object and waved it around in order to identify the tip. The tip of the object was estimated from the points in the image sequence that had the highest optical flow. Their method works with arbitrarily sized containers (and other tools).

Also essential to container manipulation is the ability to control two objects at the same time. People routinely have to insert an object into a container (e.g., while preparing a drink that should be stirred). A robot can help people do this if it has the fine-tune control that is required to insert objects into containers. Edsinger and Kemp [12] showed how a robot could bimanually manipulate an object and a container in order to insert the object into the container. They presented a control system with which a robot manipulated five containers and five tools. The robot cued a person to hand it two objects, grasped both objects and then inserted one into the container, and afterword placed the cup on a shelf. When given a spoon, the robot stirred the contents of the cup before placing the cup on the shelf. The success of their control system derived from three themes of tool use: *cooperative manipulation*, *task relevant features*, and *let the body do the thinking*.

In contrast, this paper asks how can a robot *learn* about the controllability of containers and their contents. It is based on the assumption that robot learning should be grounded in its behaviors and the outcomes that they produce. Without behavior–grounded knowledge, what the robot learns would not verifiable by the robot; it would have no way to test, verify, and correct its knowledge autonomously. Thus, this paper used a developmental learning strategy, which is an important strategy toward creating a learning algorithm that can ultimately handle large changes in the robot's operating environment, and the containers that it has to control.

Experimental Setup

Robot

All experiments will be performed with the upper-torso humanoid robot shown on the cover page. The robot was built with two 7-DOF Whole Arm Manipulators (WAMs) by Barrett Technology, each equipped with the Barrett Hand as its end effector. The WAMs are mounted in a configuration similar to that of human arms. They are controlled in real time from a Linux PC at 500 Hz over a CAN bus interface. The robot is also equipped with two cameras (Quickcams from Logitech). The cameras capture 640x480 color images at 30 fps.

Objects

A set of 20 graspable objects will be used for these experiments (see Fig. 2.a). The objects will include 10 containers and 10 non-containers—the same 10 containers flipped over. The objects were used for previous container–learning experiments and worked well. They were selected to have a variety of shapes, sizes, and material properties.

A set of 5 blocks will also be used (see Fig. 2.b). The 5 blocks will be selected to have a variety of shapes, sizes, and materials. All of them will be small enough to fit inside the 10 containers.

a)

b)

Fig. 2: The objects used in the experiments. a) The set of 10 containers and 10 non-containers. The objects were selected to have a variety of shapes, sizes, and material properties. The 10 non-containers are the same objects as the containers, but flipped upside down. b) The set of 5 blocks, which were also selected to have various properties.

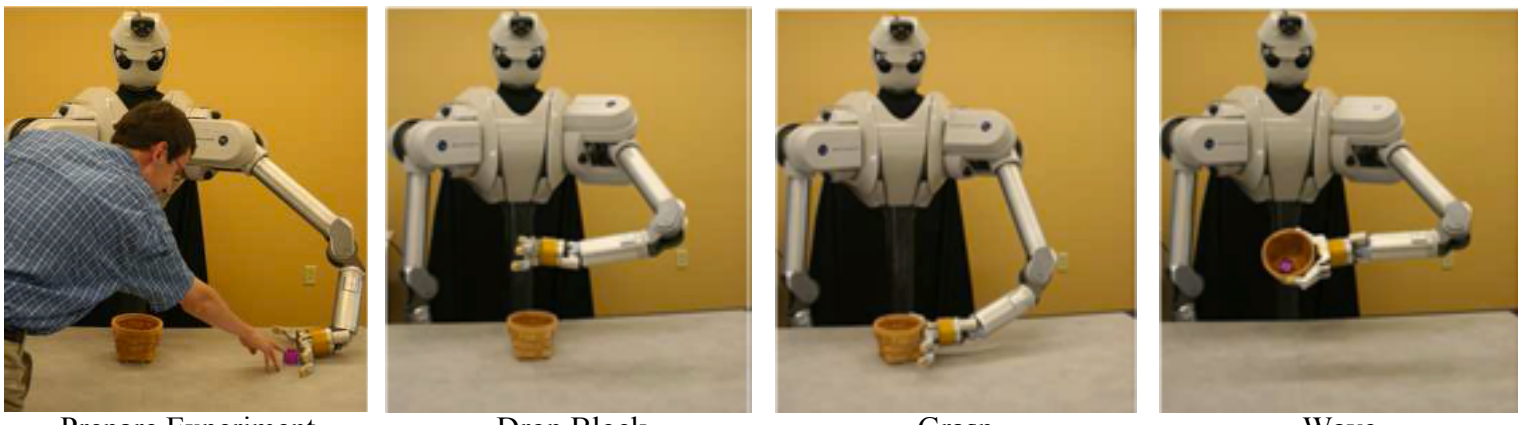
Behaviors

The robot will learn about the controllability of different objects and their contents by dropping a block above an object and then waving the object around, as shown in Fig. 3. The robot will do this 10 times for each block–object combination. A total of 20 objects * 5 blocks = 200 combinations of the objects will be used. The total number of trials performed will be 200 combinations * 10 trials each = 2000 trials. The sequence of behaviors performed during a trial are explained below:

Prepare Experiment – The block will be placed at a static location by an experimenter before every trial. It is anticipated that the blocks will frequently fall off the table or be moving at the start of the trial, which would prevent the robot from grasping it at a variety of locations. Ideally, the robot would be able to grasp the block at a variety of locations on the table. However, because round blocks will be used, they would likely be rolling around as the robot tried to grasp them or they would frequently roll off the table.

Drop block – The object will be placed at a random location in a circular area in front of the robot, which will have a diameter of roughly 30cm. The robot will locate the object in an image from its camera by performing background subtraction on the image. The location to drop the block will be computed using a mapping between the coordinates of the object in the image and the *drop block* location of the robot's arm. Linear regression with k-NN can be used to perform this mapping (Before the experiment a dataset of drop block positions matched with pixel coordinates has to be collected).

Grasp Object – The robot will grasp the object by locating it in the image and mapping the pixel coordinates to a grasp location. Linear regression with k-NN will again be used to perform the mapping. Working code for this already exists for the robot.



Prepare Experiment | Drop Block | Grasp | Wave

Fig. 3: Snapshots of the behaviors performed by the robot during an example trial with the wicker basket. The experiment starts after the block is placed at a marked location on the table. The robot will interact with the objects using the three behaviors shown.

Wave – The robot will next bring the object up toward its camera and wave it. Waving consists of moving the object back and forth using jerky motions. Each motion---back or forth---will consist of a single move motion. For this motion, the robot will move the object in a certain direction for a random distance. Following, it will move in a new direction after a random amount of time has passed. Based on preliminary experimental data, (see Fig. X) the robot should be able to perform self-detection after approximately *n* jerky movements.

Methodology

Data Collection

Experiment data will be collected during the *wave* behavior. Data will consist of a sequence of 640x480 color images, along with the timestamps of each image. The time at which the robot performs each jerky movement during the *wave* behavior will also be recorded.

Optical Flow

Objects in the robot's visual field will be tracked using optical flow. Tracking is necessary in order to estimate what objects in the robot's visual field move at the same time that a movement command is issued. A dense optical flow algorithm will be used, which can track every pixel from frame to frame without using pre-specified visual features. Figure 4 shows an example image sequence tracked using optical flow.

The Classic+NL algorithm was chosen for this project [13]. The algorithm gives state-of-the-art optical flow performance. It can produce dense optical flow for large, untextured regions of an image sequence. Example output for the algorithm is shown in Fig. 4.

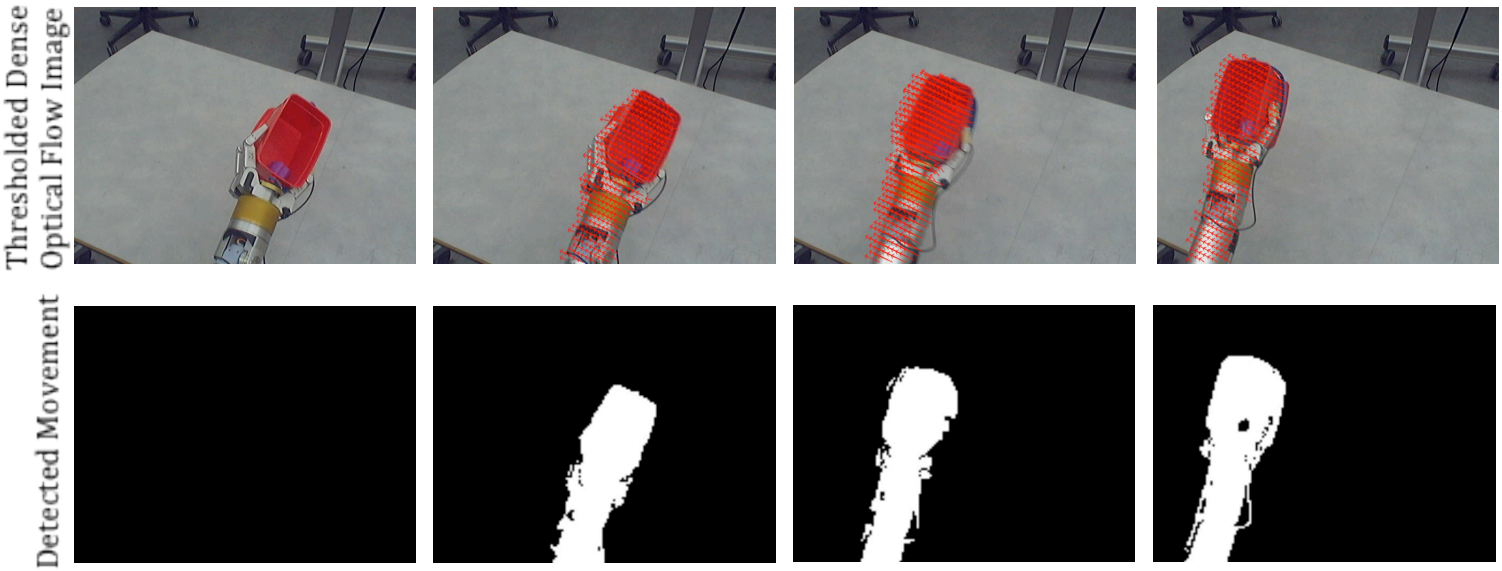


Fig. 4: Example image sequence showing the tracking algorithm and the detected movement. **Top row:** Optical flow is computed between each frame of the image sequence, which was captured during one execution of the wave behavior. Each pixel is tracked in order to create a dense optical flow image. Arrows are plotted only for the pixels that moved a distance larger than *d*. **Bottom row:** Movement detected between each frame, which was extrapolated from the optical flow image. Pixels are colored white to denote movement. **Far right column:** The robot's arm and the container registered movement, but the block did not. The images capture an instance when the robot had imperfect control of the block's movement.

The output of the optical flow step for trial T is a binary image sequence $B_T = B_T^1, ..., B_T^l$, whose binary values specify movement/no-movement for each pixel at its original position in the image sequence. First, the optical flow image, O^i, will be computed between the first image, F^1, in an image sequence, and the other images, $F^2, \ldots, F^l$ in the sequence (The first image is recurrently used in order to consistently track the same features. Otherwise the features may change with each frame, which would produce inaccurate tracking results.) Next, the amount of movement between successive frames F^i and F^j will be derived from the difference between optical flow images O^i and O^j. Finally, a threshold d will be applied in order to create each binary image B_T^i in the binary image sequence.

Self-Detection

By measuring the uncertainty in the delay between the times that robot issued a movement command and the times that a pixel moved in the image sequence, the robot can identify what pixels belong to its body and what pixels do not. It may also be possible for the robot to identify what pixels belong to the contents of containers, whose movement does not coincide as tightly with the robot's own movement. The confidence that a pixel is *self* can be measured for all tracked objects—thus defining a metric for controllability. An example confidence graph is illustrated in figure 5.

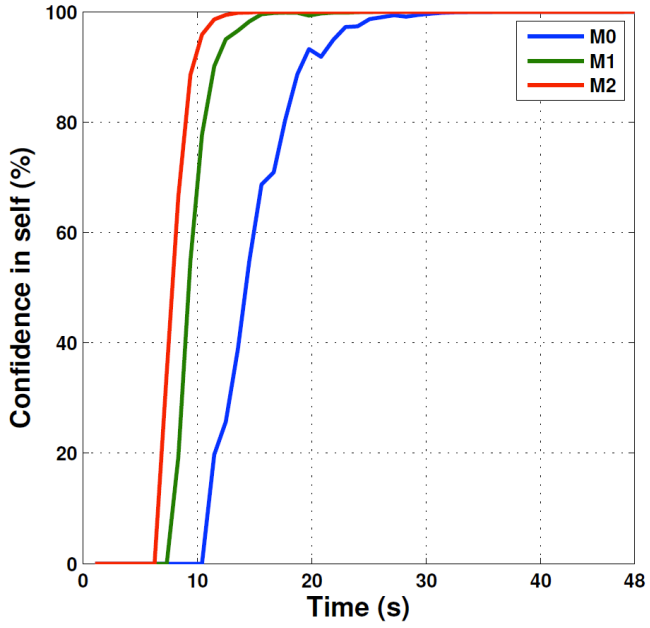


Fig. 5: Confidence graph generated for an example image sequence. Instead of tracking pixels using optical flow, three objects were tracked using color: MO represents the container; M1 represents the block; and M2 represents the robot's wrist. The self-detection algorithm first predicted that M2 was a part of its body—the wrist always moved when the *wave* command was issued. The block's position was tracked less-reliably due to limitations of color-based tracking, which explains why M1 was detected as *self* more quickly than M0. Because movement was detected for the block more than it actually moved, the block appeared to move with the robot's *wave* behavior more tightly than it actually did.

Given a feature, the self-detection algorithm predicts if the robot controls it. The assumption is that a feature controlled by the robot belongs to the robot. In this case, however, the feature may belong to an object, or the contents of a container. Because the container is rigidly grasped, it should move when the robot moves. However, because the block is free to move around inside a container, its movements will not be at exactly the same time as the robot's own movement. Thus, the controllability of the block should be lower than the controllability of the robot's body, or rigidly grasped objects.

The relationship between the movement of the feature, quantified by a variable M, and the temporal delay since the last motor command, quantified by a variable D, is used for self-detection. The mutual information $I(M; D)$ quantifies the amount of information shared between the two variables. The criterion for self-detection:

Self if $I(M; D) > 0$
Not Self if $I(M; D) = 0$

Since the precise distribution of the variables is not known but only the data is available, self-detection is performed as a statistical hypothesis test where the null hypothesis is *Not Self*. In practice, exponential convergence is observed for the p-value due to statistical properties of entropy estimation.

In this case, the features passed to the self-detection algorithm were the individual pixels from the binary image sequence $B_T = B_T^1, ..., B_T^l$, the time that each frame was captured, and the time that each *wave* movement command was issued. The result of self-detection is a controllability image, C_T, which identifies the confidence in the controllability of each pixel in the image.

Deep Learning

A deep learning architecture is a special configuration of a hierarchical neural network. To understand deep learning architectures, it is helpful to have an understanding of neural networks, and the limitations of traditional hierarchical neural networks.

> *Neural Networks* – One kind of deep learning architecture is based on the neural network algorithm. This deep architecture can be applied to any algorithm that has two layers of nodes, in which the top layer nodes are hidden units and the bottom layer nodes are the input data. In practice, most neural networks have weighted, directed edges connecting every node of the input layer to every node of the hidden layer. The result is a densely connected graph between the input units and the hidden units. The hidden layer often has fewer nodes than the input layer for the purposes of dimensionality reduction.
>
> The edge weights, W, and a bias term, b, determine the mapping between the input and the output of the neural network. A weight w_{ij} is the weight between an input node n_i and a hidden node h_j. Initially, each weight is assigned to a random value near 0.01. However, through a training process, these values are modified in order for the neural network to learn a mapping from the input data to the output

data. A bias term, *b*, feeds into each hidden unit and is also modified during the training process.

A neural network can be trained to learn a mapping from a set of training data *T* to their output labels *L*. First, the neural network is structured to resemble this mapping. That is, the number of nodes in the hidden layer is set to match the dimension of the value of *L*. Following, a supervised learning process is used to modify the weights and bias term of the network using the tuple (T^u, L^u).

That is, for each training data T^u, the output of the network is predicted using the sigmoid function of a weighted combination of the inputs. Following, the error of the network is measured using the difference between the value L and the output O^u. The weights between the nodes in the neural network are modified using a process called *backpropagation* in order to minimize this error. The training process is finished either when the weights in the network are stabilized. At this point, the neural network can be used to predict an output label *l* given a test input T^{test}. For some tasks, neural networks can do this fairly well.

Hierarchical Neural Networks – Additional layers of hidden units can be stacked on top of the existing structure in order to create a hierarchical learning architecture, which sometimes improves prediction accuracy. For example, a three-layer network is created by densely connecting the output from the second-layer hidden units to an additional layer of hidden units, the third layer. The first layer is still the training data, which is used as the input to the second-layer hidden units. The output from the second-layer hidden units is treated as the input data for the third-layer hidden units. Additional layers of hidden units can be connected to the hierarchy in a similar fashion.

The performance gain of hierarchical neural networks is often marginal, however. In these hierarchical networks, the weights and bias values of the hidden layer often converge to local minimum error values, which can be far from the optimal solution. There is no guarantee that the network can produce an accurate mapping of the input data. Until Hinton figured out how to do deep learning [14], however, most implementations used this type of hierarchical structure.

Deep Learning – A deep learning architecture is a hierarchical neural network structure, which is trained in a special way. Instead of connecting all the layers of nodes together and training all at once, the hierarchical network is trained layer by layer. This *pre-training* phase initializes the weights in each layer to near the optimal solution. Once the first layer is trained, the weights in that layer are frozen and the outputs from the hidden units are used as input the second layer. This is shown in Fig. 6.Pretraining.

After the training phase, the hierarchical network is unfolded in order to create encoder and decoder networks. Unfolding means copying all of the trained layers

into a decoder network, which has the exact same weights as the encoder network. This is shown in Fig. 6.Unrolling.

In the last training phase the weights of the network are fine-tuned toward the optimal solution. The training data is fed to the network again, and the backpropagation of error derivatives through the whole network hierarchy are used to update the weights. This is shown in Fig. 6.Fine-tuning.

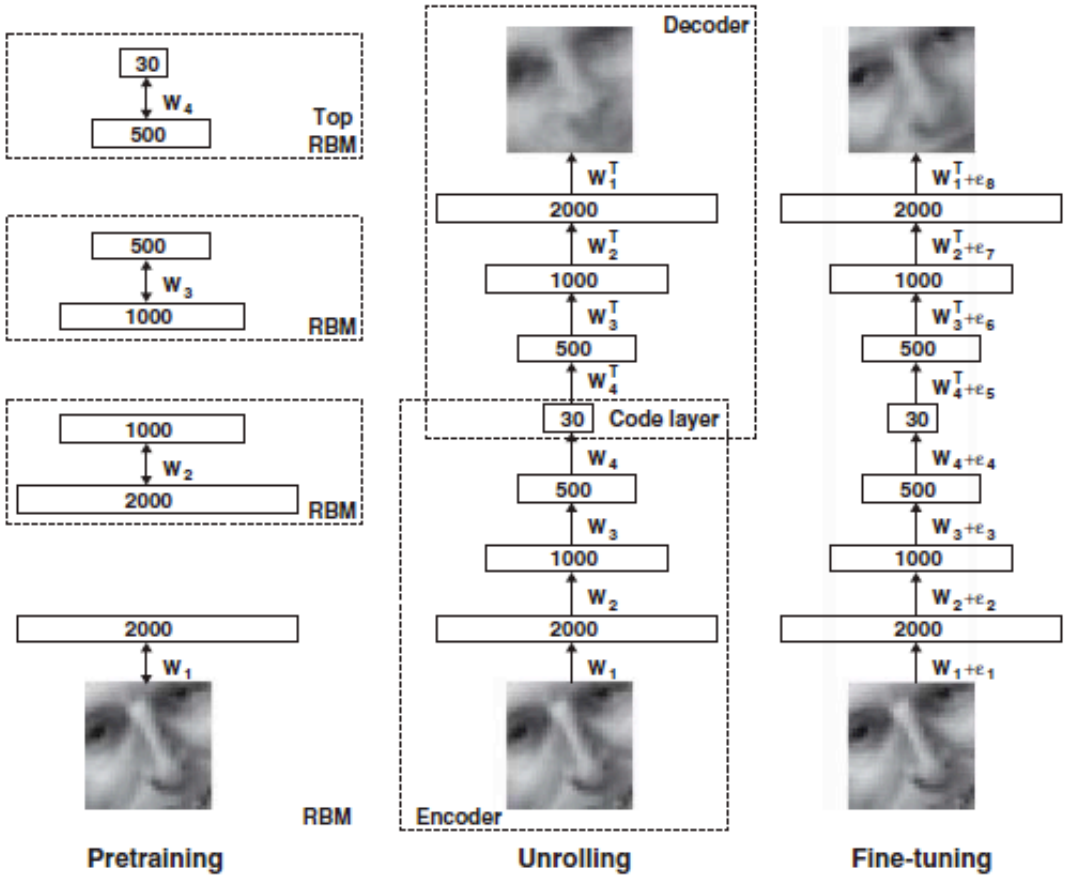


Fig. 6. Pretraining consists of learning a stack of restricted Boltzmann machines (RBMs), each having only one layer of feature detectors. The learned feature activations of one RBM are used as the "data" for training the next RBM in the stack. After the pretraining, the RBMs are "unrolled" to create a deep autoencoder, which is then fine-tuned using backpropagation of error derivatives.

For the case of this project, a deep learning autoencoder network will be used, based on the work of Ngiam et al. [14]. Ngiam et al showed that a shared feature representation can be learned between modalities by connecting two deep learning networks at the code layer. For this proposed research, the two different inputs to the algorithm will be the first image captured during the *wave* behavior and the resulting controllability image computed using self-detection (see Fig.7). Once trained, the multimodal deep autoencoder network can be used to reconstruct both types of input data when given only one type, since the shared representation

encodes something about both inputs. This allows the robot to predict the structure of the controllability image given only the visual image.

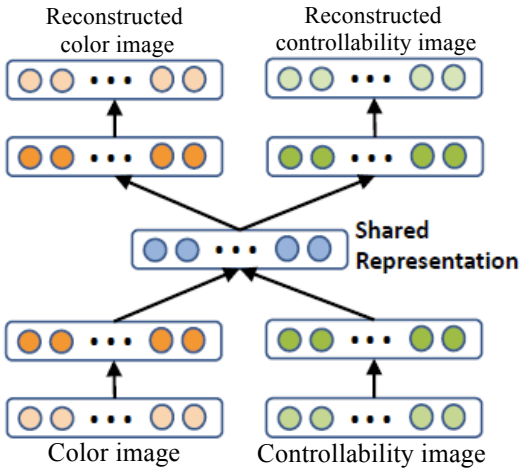


Fig. 7: Bi-modal deep autoencoder.

Evaluation

The learning algorithm will be evaluated by testing how well the robot can detect the controllability of objects in a novel image. The novel image will be passed to the trained deep learning architecture. The controllability for objects in the image will be unknown, but should be constructed by the shared representation in the deep learning architecture. This research will be successful if the robot can accurately construct the controllability image. In theory, the algorithm should be able to predict that a block inside a container is moderately controllable. If it does, the robot will have successfully gained a representation that unites the controllability of a container's contents along with a visual description of the *inside* spatial relationship.

Potential for Future Work

If successful, the research proposed here leaves open several possibilities for future work. Future work could show how the robot can gain control the contents of containers. The proposed research only seeks to show that the robot can discriminate between controllability of objects based on their spatial configuration with other objects. A robot could use what it learned in this study to start to try and improve its control over the blocks inside containers. If successful, the robot could solve common puzzle games like the bead puzzle problem, in which the objective is to maneuver several tiny beads into equally tiny holes on a sheet of paper, which is all enclosed inside a clear object.

Future work could also explore the robot's ability to autonomously learn several parameters that determine the controllability of blocks inside containers. For example, how tightly fitting the block is with the container determines the amount of controllability

the robot will have over the block. Perhaps the robot can learn to predict controllability based on the size of the block. Another variable tat the robot could learn is controllability vs. jerkiness during movements. The more jerky the robot's movement, the less the block will appear to be moving with the container and the robot. A robot could learn that, given a block inside a container, the best strategy for manipulation is the strategy that has the least jerky movements. If a robot can learn about these variables using our algorithm, then it would be useful for solving the slosh-free motion of containers task, even with arbitrarily sized containers.

Qualifications

The project proposed here is combines my knowledge from several domains of developmental robotics and computational perception. My experience in each of these domains is described below:

Robot – I have performed numerous experiments and demonstrations of the upper-torso humanoid robot with the Developmental Robotics Laboratory. My continued use of the robot has helped me resolve many unforeseen challenges in order to complete experiments. My experience taught me that maximal planning and preparation for an experiment helps, and that we cannot foresee all bugs before an experiment starts. Also, I learned that proposed ideas don't always work out as planned due to the assumptions we make about the robot's perceptual world. However, the reasons for failures are often enlightening and encouraging.

Object Tracking – I have four years of image processing experience. Almost all of my major research projects have required some type of object tracking. Because of this, I have ready-to-use code for color-based object tracking, as well as optical flow-based feature tracking.

Self-Detection – This part of the project will be outsourced to Vlad Sukhoy. My project is a perfect application for his uncertainty-driven self-detection algorithm.

Deep Learning – I attended the Deep Learning and Unsupervised Feature Learning Workshop at Neural Information Processing Systems in December 2010. The work presented there introduced me to the concept of deep learning architectures. I learned why they work so well and that they can be applied to many different machine learning problems, from feature extraction to object tracking. In the time since the workshop ended, I identified a few different applications for their use in my research. I have also taken steps toward an implementation. However, due to the fact that this will be my first implementation of a deep learning architecture, this will be the most challenging part of the project.

Publication – My ability to propose a publishable project is demonstrated by my publication track record. Ritika Sahai and I published the project we proposed for computational perception 575X. Also, I have first-authored four papers and co-

authored 3 papers for the field of developmental robotics. For this project, I plan to submit the work to the International Conference on Development and Learning, which means that it has to be completely finished by March 28 (two days before your review of this proposal is due).

References

[1] J. Feddema, C. Dohrmann, G. Parker, R. Robinett, V. Romero, and D. Schmitt. "Control for slosh-free motion of an open container." IEEE Control Systems Magazine. v. 17. no. 1. pp. 29-36. 1997.

[2] V.J. Romero and M.S. Ingber, "A Numerical Model for 2-D Sloshing of Pseudo-Viscous Liquids in Horizontally Accelerated Rectangular Containers", *Proc. 17th International Conference on Boundary Elements*, pp. 1995.

[3] Tzamtzi, M.P.; Koumboulis, F.N.; Kouvakas, N.D.; , "A two stage robot control for liquid transfer," *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on* , vol., no., pp.1324-1333, 25-28 Sept. 2007

[4] K. Gold and B. Scassellati. "Learning about the self and others through contingency." 2005 AAAI Spring Symposium "Developmental Robotics." Stanford, CA. Mar. 2005.

[5] Stoytchev, A., "Self-Detection in Robots: A Method Based on Detecting Temporal Contingencies", Robotica, volume 29, pp. 1-21, 2011. Cambridge University Press 2011.

[6] Giorgio Metta and Paul Fitzpatrick. Early integration of vision and manipulation. Adaptive Behavior, 11:2, pp. 109-128, June 2003.

[7] Kemp, Charles C. and Edsinger, Aaron. "What Can I Control?: The Development of Visual Categories for a Robot's Body and the World that it Influences". Proceedings of the Fifth International Conference on Development and Learning, Special Session on Autonomous Mental Development. 2006.

[8] M. Casasola, L. Cohen, and E. Chiarello. "Six-month-old infants' categorization of containment spatial relations." Child Development. vol. 74. no. 3. pp. 679-693. 2003.

[9] R. Baillargeon. "How do infants learn about the physical world?" Current Direction in Psychological Science. vol. 3. no. 5. pp. 133-140. 1994.

[10] R. Largo and J. Howard. "Developmental progression in play behavior of children between nine and thirty months: II: Spontaneous play and language development." Developmental Medicine and Child Neurology. vol. 21. no. 4. pp. 492-503. 1979.

[11] Kemp, Charles C. and Edsinger, Aaron. "Robot Manipulation of Human Tools: Autonomous Detection and Control of Task Relevant Features". Proceedings of the Fifth

International Conference on Development and Learning, Special Session on Classifying Activities in Manual Tasks. 2006.

[12] Edsinger, Aaron and Kemp, Charles. "Two Arms are Better than One: Designing Robots that Assist People in Everyday Manual Tasks". Proceedings of the IEEE International Conference on Advanced Robotics (ICAR). 2007.

[13] D. Sun, S. Roth, M. Black. “Secrets of optical flow estimation and their principles.” IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2432-2439. 2010.

[14] G. Hinton and Salakhutdinov. “Reducing the Dimensionality of Data with Neural Newtorks.” Science. v. 313. pp. 504-507.

[15] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee and Andrew Y. Ng. “Multimodal Deep Learning.” *in NIPS Workshop on Deep Learning and Unsupervised Feature Learning. 2010.*