Self-Supervised Deep Visuomotor Learning from Motor Unit Feedback

Jay Ming Wong*, Takeshi Takahashi*, Roderic A. Grupen
Laboratory for Perceptual Robotics
College of Information and Computer Sciences, University of Massachusetts Amherst
{jayming, ttakahas, grupen}@cs.umass.edu

***Abstract*—Despite recent success in a number of domains with deep learning, expensive data collection and the need for large datasets becomes a major drawback for deep learning with real robotic platforms. As a result, many of the successful work in deep learning has been limited to domains where large datasets are readily available or easily collected. To address this issue, we leverage closed-loop controllers to reduce the dimensionality of output neurons and introduce a method of self-supervision derived from a dynamical systems approach that captures the robot's evaluation of its own interactions with the world. We focus on the accurate prediction of controller dynamics (or motor unit status) given visual observation and demonstrate that a real robot platform, the uBot-6, can quickly acquire visuomotor skills that are general enough to be applied to a set of novel objects.**

I. Introduction

With the recent resurgence of neural networks under *deep learning*, a number of great successes have been shown in vision and other difficult tasks. For instance, a remarkable results with deep learning have been demonstrated in the ImageNet competition [1], Atari [2, 3], and Go [4] gameplay. Yet, despite surprising success in many of these applications, deep learning is far from the norm in robotics. Concerns arise when these methods require many thousands of parameters to be set and are surprisingly easy to fool [5, 6]. Furthermore, a prerequisite for deep learning architectures is the requirement of large amounts of training data—upwards millions of training examples. In fact, it is particularly hard for robot systems to collect sufficiently large sets of data. As a result, some approaches rely on pretrained architectures to initialize the network using relatively good weights, with many leveraging the AlexNet result [7, 8].

Deep reinforcement learning via *Deep Q Networks* (DQNs) [2], leveraging convolutional architectures for Q-value function approximation with continuous observation spaces, have recently been introduced to extend ideas dating back to two decades of research [9, 10]. An actor-critic approach with "soft" target updates has been shown to extend these initial DQNs for continuous action spaces [11], however, despite its performance being competitive with optimal controllers, their algorithm has only been tested in simulation and have yet been demonstrated on a real physical system.

Much of the success in deep learning relating to robotics and control seem limited with demonstrations mainly in gameplay domains (e.g. Torcs, Atari, and Go), where *physical* control is not really considered—even fewer have even looked into applying such methods into real physical and dynamical systems. DQNs have been applied to some robotics applications, however, are limited to precisely defined domains for exploration in parameter space on real robot platforms [12]. Unfortunately, it is not the case that networks learned in simulation can be easily transferable to actual hardware and real world observations despite millions of training steps in over hundreds of hours [13]. Because of this, systems must collect and learn from real world experiences. A multi-staged architecture relying on exhaustive search over candidate rectangles in the scene for detection and execution of grasps have been shown [14] to avoid hand-designing features for grasping which have been popular approaches in the past [15, 16]. Meanwhile, Levine *et al.* showed that deep reinforcement architectures held promise by learning a number of manipulation tasks that require close coordination between vision and control but required optimal control theory to pretrain a network with motor torque outputs [12]. Extensions using deep spatial autoencoders with architectures like this allows for possibility in solving a wider range of tasks where suitable representations are not known *a priori* [17].

As many of these methods rely heavily on pretraining and unsupervised methods, we provide an approach that moves away from such necessities. A similar motivation by Pinto and Gupta has been discussed, where they obtained hundreds of hours of robot labeled grasp data for training leveraging a heuristic-based labeling metric [18]. Recent work by Levine *et al.* incorporated a similar predefined metric (gripper status and an image subtraction test) for self-supervision to train a network built to predict grasp success probability, driven by task-space motor commands [19]. Our approach differs in the way that self-supervision is derived from a dynamical systems approach and captures the robot's evaluation of its own actions through closed-loop control interactions with the environment. In this case, the output of our network corresponds to the robot's prediction of the status of its motor units.

This paper provides three main contributions in deep learning for robotics applications:

1) We propose a novel approach to simplify deep visuomotor skill acquisition using closed-loop motor units (or controllers) as output neurons (different from motor torque

*These authors contributed equally.

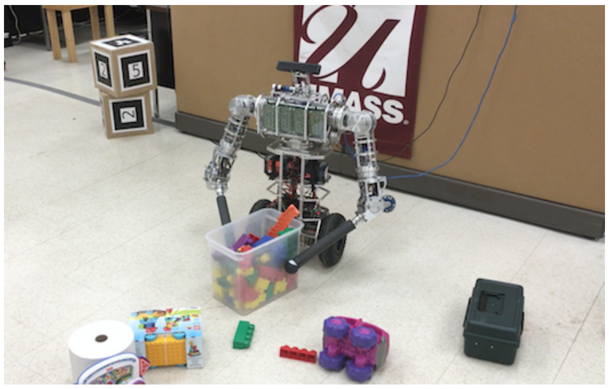

Fig. 1: Self-supervision: uBot-6 self-labels its own grasp experiences for training using closed-loop controller (motor unit) feedback.

outputs in the literature).

2) We show that the trajectory information of controllers alone provides a method for self-supervision in which robots label their own experiences for training.
3) And lastly, we discuss the performance using varying network architectures that implement these motor units.

II. Technical Approach

This paper leverages closed-loop motor units to reduce the dimensionality of the output layer of a deep convolutional network, whose purpose is to predict controller convergence or quiescence. By deriving the controller state representation from a dynamical system model of the interaction of the robot with the world, accurate prediction of controller dynamics suggests whether particular actions are afforded by the visual observation of the world. Executing a controller that is predicted to converge given some observation describes a visuomotor policy that performs these afforded actions. Such an approach learns visuomotor policies and skills along with perceptual representation.

A. Closed-loop Controller Dynamics

In this paper we employ a *Control Basis* formulation of closed-loop controllers. These controllers $\phi|_{\tau}^{\sigma}$, consist of a combination of potential functions ($\phi \in \Phi$), sensory inputs ($\sigma \subseteq \Sigma$), and motor resources ($\tau \subseteq \mathcal{T}$) [20, 21]. Such controllers achieve their objective by following gradients in the potential function $\phi(\sigma)$ with respect to changes in the value of the motor variables $\partial \boldsymbol{u_\tau}$, described by the error Jacobian $J = \partial\phi(\sigma)/\partial \boldsymbol{u_\tau}$. References to low-level motor units are computed as $\Delta u_\tau = -J^{\#}\phi(\sigma)$, where $J^{\#}$ is the pseudoinverse of J.

The interaction between the embodied system and the environment is modeled as a dynamical system, allowing the robot to evaluate the status of its actions as a state describing a time varying control system. The time history or trajectory of dynamics $(\phi, \dot{\phi})$ as a result of interactions with the environment by executing controllers have been shown to have predictive capability regarding the state of the environment [22]. The state description γ^t at time t, in the context of

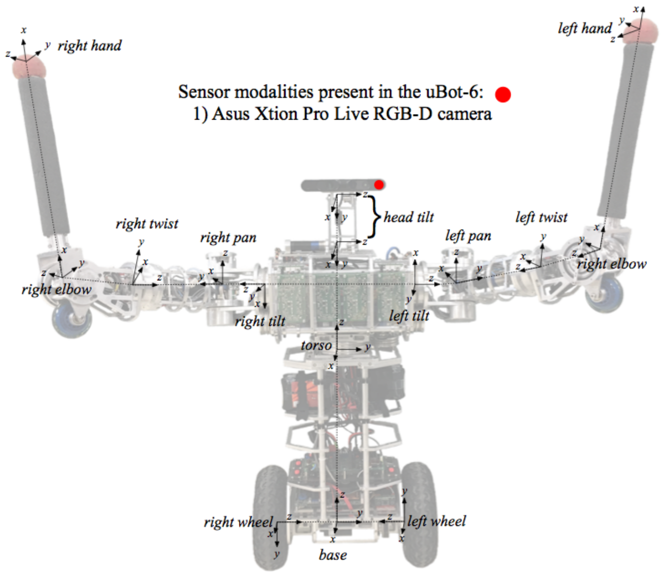

Fig. 2: uBot-6 robot platform: a 13 DOF, toddler-sized, dynamically balancing, humanoid mobile manipulator

this paper, is derived directly from the dynamics $(\phi, \dot{\phi})$ of the controller such that,

$$\gamma^t(\phi|_{\tau}^{\sigma}) = \begin{cases} \text{UNDEFINED}: & \phi \text{ has undefined reference} \\ \text{TRANSIENT}: & |\dot{\phi}| > \epsilon \\ \text{CONVERGED}: & |\dot{\phi}| \leq \epsilon \text{ and } \phi \text{ reaches } g \\ \text{QUIESCENT}: & |\dot{\phi}| \leq \epsilon \text{ and } \phi \text{ fails to reach } g \end{cases}$$

where g is the reference control goal.

The robot performs actions in the form of executing control programs defined as providing some goal reference to these closed-loop controllers. The purpose of this work to implement a system that predicts controller convergence or quiescence—an implication of the result forms a visuomotor policy indicating when visual stimuli in the scene affords the given action. Because this approach leverages the controller dynamics in defining state, the output of the network is reduced to an extremely small discrete space—as a result, less training data is required allowing real robot platforms to learn visuomotor policies with relatively small networks and short data collection and training times.

B. uBot-6 Mobile Manipulator

The uBot-6 robot platform (shown in Figure 1 and Figure 2) is a 13 DOF, toddler-sized, dynamically balancing, humanoid mobile manipulator equipped with an Asus Xtion Pro Live RGB-D camera, designed and built at the Laboratory for Perceptual Robotics [23]. The robot leverages the use of ROS (Robot Operating System) for a middleware, a publish-subscribe and message passing architecture [24].

Actions in the context of this paper corresponds to the execution of control programs. Primitive actions are defined basis controllers with some reference goal. In particular, the

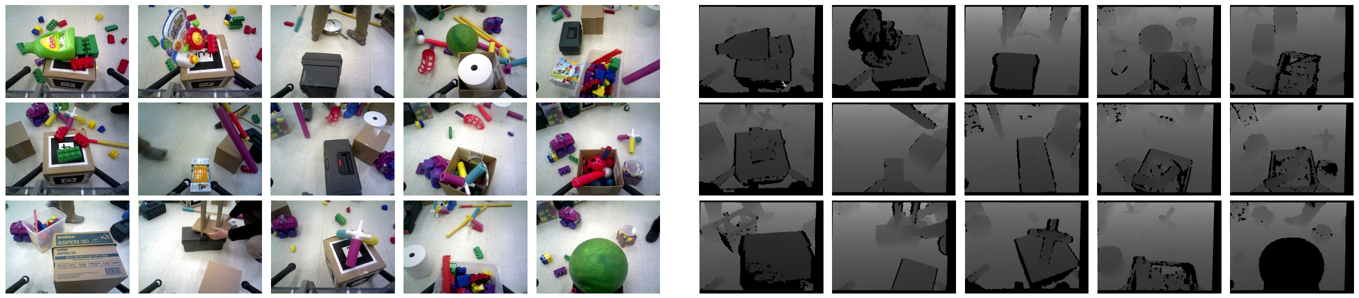
Fig. 3: Successful grasps: Selected grasp examples consisting of RGB (left) and depth (right) that resulted in grasp convergence.

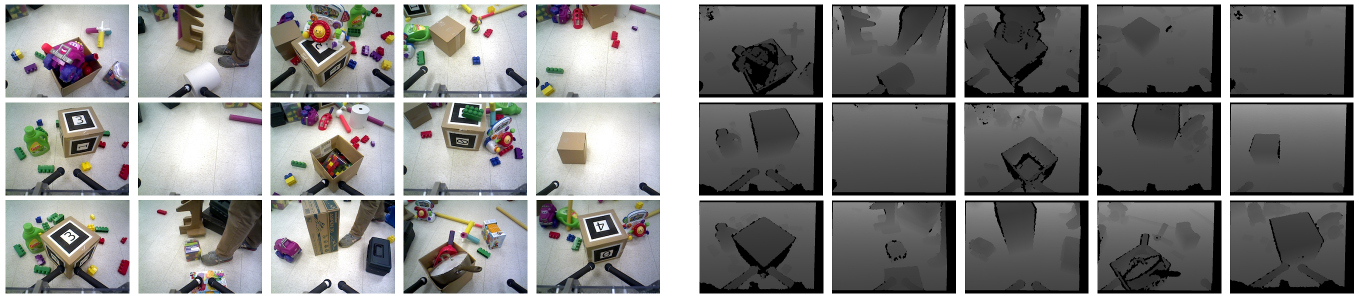
Fig. 4: Failed grasps: Selected grasp examples consisting of RGB (left) and depth (right) that resulted in grasp quiescence or failure.

GRASP control program that we demonstrate in this paper implements a particular target set-point in the form,

$$g = \left\langle \begin{bmatrix} x \\ y \\ z \end{bmatrix}_L, \begin{bmatrix} x \\ y \\ z \end{bmatrix}_R, M \right\rangle$$

where the control goal g is given by vectors describing the Cartesian positions $[x, y, z]_h$ to reach to for each hand h (with L and R denoting the left and right hands) and a particular desired compression magnitude M that the robot uses as a reference to track. This GRASP control program is a sequential composition of two primitives controllers described by control programs REACH and COMPRESS. The former is an endpoint position controller that drives the positions of the end effectors to some Cartesian position defined by $[x, y, z]_h$ and the latter is a compressive force tracking controller that aims to track some particular compression magnitude M. The robot evaluates the compressive force by tracking some set-point position error during the COMPRESS action. If the compress action converges to the Cartesian set-points that are analogous to a perceived force, then the grasping controller quiesces, unable to track the compression magnitude M. The COMPRESS action that leverages perceived set-point errors as a pseudo-force is further described in [25].

In our demonstrations, a primitive grasp is defined by a reflexive action given by some fixed goal parameters describing the target set-points for the controller ϕ_{GRASP}. We will elaborate further on why this particular form of action is leveraged in the discussion section.

C. Self-Supervision via Controller State

We provide an approach in which robots can autonomously acquire large sets of training data from experiences—this relies on the fact that the robot can reliably self-label its training instances during the data collection session by querying the status of its controllers. All data collection for training was performed on the real robot system.

A data collection node was integrated with the ROS architecture allowing for the capture of depth images at 10 Hz during each grasp trial. Generally, grasp sessions take approximately ten seconds and as a result, the robot acquires at least 100 training images for each grasp. In addition to saving the depth image every 10 Hz, if at any time the state of the controller (γ^t) changes, an additional depth image is logged and saved into memory.

During the grasping session, the robot was presented with a subset of 12 training objects provided at random positions and orientations in the scene during each trial. Additional objects were added to the scene as distractors—they were either placed out of reach, on top of other objects, or provided occlusion to areas in the scene. Incorporating distractors in the scene allows us to produce training instances that incorporated a large amount of clutter. The data collection process consisted of 150 grasping trials consisting of about two hours of robot run time. For each trial, the robot performs a fixed depth grasp action while continuously saving the depth information. The grasp controller ϕ_{GRASP} is executed until $|\dot{\phi}| \leq \epsilon$ is satisfied indicating that the controller is no longer making progress towards its target fixed point. The controller state when this is satisfied becomes the label that is assigned to the whole grasp session—each of the saved images inherits this label. Some of these grasp trials are illustrated in Figure 3 and Figure 4 in which the RGB and depth images are illustrated for successful grasps (labeled by $\gamma^T(\phi_{\text{GRASP}}) = 1$ or converged) and failed grasp examples (labeled by $\gamma^T(\phi_{\text{GRASP}}) = 2$ or quiesced).

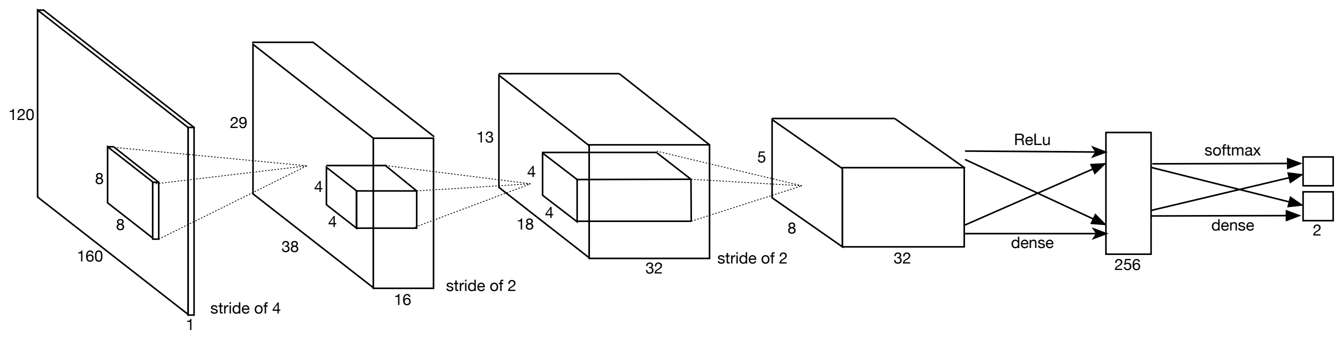

Fig. 5: Schematic of the network architecture consisting of two convolutional layers followed by a dense linear rectified layer which is densely connected to a softmax output layer. The network's input is of size 160×120 and the output corresponds to predictions regarding the convergence or quiescence of the controller ϕ_{GRASP}.

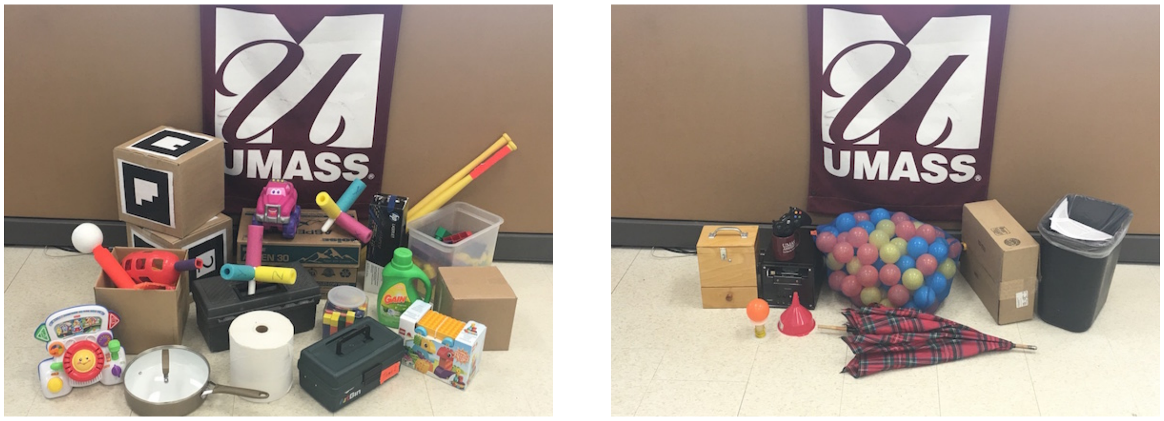

Fig. 6: Training set (left) and test set (right). Training set consists of twelve varying objects with a number of distractors such as blocks, bats, and toys. Two of the twelve objects were from the YCB grasping dataset. The test set consisted of ten novel objects with half being physically not graspable given the properties of the object and the robot.

Each grasping trial, when completed, calls a homing procedure to reposture the robot.

In total, uBot-6 self-labeled 150 grasp experiences, consisting of $17,714$ depth images, which is then used as input to a deep convolutional neural network. About a half of the grasp experiences are CONVERGED examples. A fraction of the total image set was used to train the network. As these images contribute to a time history during the grasp, many of these images are similar. To prevent overfitting, we used 4556 depth images, about one fourth of the overall set of collected data.

D. Network Architecture

The depth image obtained by the Asus Xtion Pro Live RGB-D camera collected and self-labeled by the robot is subsampled into an input of size 160×120. At the first convolutional layer, 16 filters of size 8×8 with a stride of four is applied to the input image at the first convolutional layer. The second layer convolves the resulting $38 \times 29 \times 16$ output with 32 filters of size 4×4 with a stride of two. Another 32 filters of size 4×4 convolves this $18 \times 13 \times 32$ output using a stride of two at the third convolutional layer. This is then fully connected to 256 neurons with Rectified Linear Units (ReLu) as a nonlinear activation function. Lastly, a softmax is applied connecting to the output layer consisting of two neurons for each controller in the set Φ. The purpose of each of these two neurons is to predict the controller state $\gamma^t(\phi|^{\sigma}_{\tau})$ at the time $t = T$ where $|\dot{\phi}| \leq \epsilon$ is satisfied and the controller terminates by either convergence or quiescence. In the case of this paper, we consider solely a grasp controller $\phi_{\text{GRASP}} \in \Phi$. A pictorial image of the network architecture used in this paper is illustrated in Figure 5. We investigated network architectures with the different number of convolutional layers. The comparisons are shown in Figure 9. The difference in the performance between the network with five convolutional layers and the network with three convolutional layers is minimal therefore we adopted three convolutional layers described above as it requires less computation than that of the 5-convolutional layers network.

Learning with this network employs the categorical cross-entropy between network predictions $p_{i,j}$ and training targets $t_{i,j}$ describing the network's loss function,

$$\mathcal{L} = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{S} t_{i,j}\log(p_{i,j}).$$

where i is a batch number, N is the number of batches, j is a controller state, and S is the number of terminal controller states ($|\dot{\phi}| \leq \epsilon$). We use RMSProp [26] to update the weights of the network with learning rate $\alpha = 0.00001$ and decay

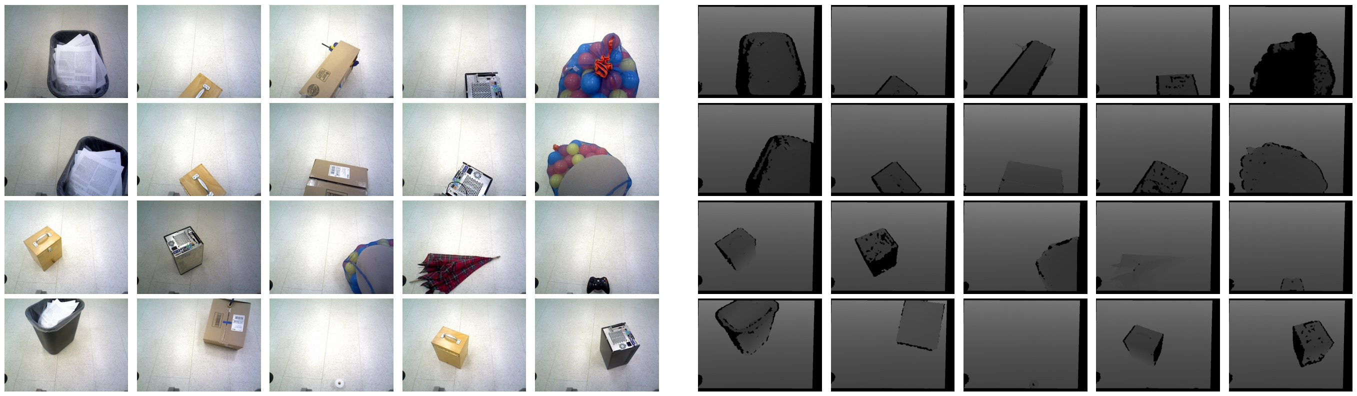
Fig. 7: Correct Prediction: Selected correct prediction examples consisting of RGB (left) and depth (right) that were taken before the grasp action started. The images in the first and the second rows illustrates correct convergence predictions while those in the third and fourth rows depicts the correct quiescence predictions.

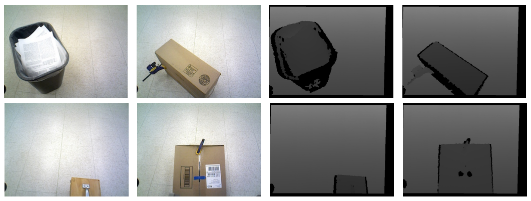
Fig. 8: Incorrect Prediction: Selected incorrect prediction examples consisting of RGB (left) and depth (right) that were taken before the grasp action started. The images in the first row depict the case where the robot has predicted a convergent grasp, but in reality resulted in quiescence. The second row shows the case where the robot predicted a quiesced grasp outcome, but the grasp resulted in convergence. The bottom rightmost panel illustrates a cardboard box that is tilted with respect to the horizontal plane.

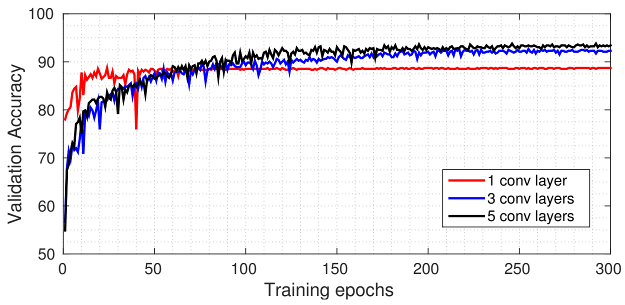

Fig. 9: Training plot: comparing the validation accuracy of three different network architectures: network with one, three, and five convolutional layers. The graph is illustrating the validation accuracy. The network with one convolutional layer uses 16 filters of size 8×8 with a stride of four. The network with five convolutional layers uses 16 filters of size 8×8 with a stride of four in the first layer, 32 filters of size 4×4 with a stride of two in the second layer, 32 filters of size 4×4 with a stride of two in the third layer, 32 filters of size 2×2 with a stride of one in the fourth layer, and 32 filters of size 2×2 with a stride of one in the fifth layer. The last two layers are the same for all the networks. (Best viewed in color).

factor $\rho = 0.99$. This particular convolutional neural network was implemented using Lasagne[1] and ROS.

III. Experimental Methodologies

Our experiments employ the uBot-6 mobile manipulator with a primitive fixed target grasp controller ϕ_{GRASP}. We train the network using the set of training data that was obtained through self-labeling via control basis state information.

The trained network was tested using ten novel objects that were present not in the training set—five of these objects were physically graspable by the robot, while the other five were not graspable due to the physical properties of the object and the robot. For instance, objects may not be tall enough or massive enough for the fixed setpoints to afford a bimanual grasp by the uBot-6 platform. The objects used in this study is illustrated in Figure 6—two of the objects in the training set were from the YCB grasp dataset [27]. These objects were randomly presented in ten different positions and orientations in a meter by meter workspace directly in front of the robot. The robot performs a prediction on controller state that the particular configuration of objects in the scene affords. Afterwards, the robot performs the grasp action by executing the controller to validate its prediction. A successful prediction is one that matches the controller state after the validating action is performed.

[1]https://github.com/Lasagne/Lasagne

To compensate for sensor noise, the robot performs several predictions before deciding the controller outcome. To predict controller convergence, $\gamma^T(\phi_{\text{GRASP}}) = \text{CONVERGED}$, or quiescence, $\gamma^T(\phi_{\text{GRASP}}) = \text{QUIESCENT}$, the robot computes,

$$\underset{j \in \{\text{QUIESCENT, CONVERGED}\}}{\arg\max} \left(\sum_i^K (p_n(\gamma^{t_i}(\phi_{\text{GRASP}} = j)) \right)$$

where $p_n(\gamma^{t_i}(\phi_{\text{GRASP}} = \text{QUIESCENT}))$ and $p_n(\gamma^{t_i}(\phi_{\text{GRASP}} = \text{CONVERGED}))$ are outputs from the convolutional neural network describing the predicted controller state, and K is the number of predictions. In our experiments, we used $K = 5$.

IV. Results

Correctly and incorrectly predicted trials are illustrated in Figure 7 and Figure 8. These results are the outcome of a network that was trained using grasp experiences over 12 random objects in addition to a number of distractors that were also placed in the scene. The training result is shown in Figure 9. The validation accuracy converges to about 92% after 250 epochs. All 4556 depth images as mentioned earlier was used in random selections during each epoch. Despite the task seeming simple, it is in reality difficult to achieve high validation accuracy with a single convolutional network. This result is present in Figure 9 in which we compare the accuracies between networks that leverage slightly shallower and deeper architectures. A single convolution is insufficient in achieving high accuracy—we hypothesize that this is because observing solely temporal information or depth discontinuities in the depth image (two aspects that are likely good indicators of being able to grasp) is not enough in deciding the control state. It is also important to note that using a cross-fire sensor or examining point cloud information between target setpoints may not be able to achieve higher performance either since these techniques are unable to take into account events in which the robot's hands slip along the surface of the object due to orientation or other properties that are inherent in the identity of the object. It appears that additional convolutional layers beyond three adds little marginal value—we observe that an additional two layers adds less than 1% validation accuracy.

The experiment results using novel objects consisting of 75 grasp instances by the uBot-6 platform are summarized in Table I with an overall accuracy about 93.33% over all grasp trials. The robot was able to predict the grasp controller state correctly for the most of the objects despite the fact that the robot has never seen these objects in training. Furthermore, robot predicted the outcome perfectly for non-graspable objects. Correct prediction examples are shown in Figure 7. This results illustrates that the neural network does not exhibit an overfitting phenomenon despite the limited dataset size. We assume that this is due to the reduction in output neurons by closed-loop control motor units, resulting in the number of weights necessary to optimize being relatively small. Further, it may be also attributed to having a high variety of object properties (in different shapes and sizes) in a cluttered environment in the training data set. Lastly, we suspect that learning in clutter may help reduce the chance of overfitting since provides more stimulus in the scene, forcing the network to not tend to resist drawing conclusions from a small subset of features. Of the 75 grasp experiments, the network incorrectly predicted five trials. A selected number of these incorrect predictions are depicted in Figure 8. Of these mispredictions, we suspect that the network has trouble predicting state correctly when the object is presented extremely close to the base of the robot and is not flat on the ground (an instance of this is illustrated in the lower right of Figure 8. However, since the robot has a self-labeling mechanism that exploits the control basis framework in self-labeling real world experiences, it has the capabilities to continuously improve the prediction performance by exhibiting more data collection trials.

V. Conclusion and Future Work

In this paper, we demonstrated a preliminary study integrating deep learning architectures with a control theoretic framework for robot control, using a single closed-loop control motor unit. Extentions to this preliminary work looks into a larger set of motor units describing a *Control Basis* in which

	$\gamma^T = 1$	$\gamma^T = 2$	Accuracy
Trash can	5/5	4/5	90%
Brown Box	4/5	5/5	90%
Cardboard box	4/5	4/5	80%
Computer	5/5	4/5	90%
Pile of balls	5/5	5/5	100%
Small ball	–	5/5	100%
Wire roll	–	5/5	100%
Funnel	–	5/5	100%
Umbrella	–	5/5	100%
Mug & Joystick	–	5/5	100%
Total	23/25	47/50	93.33%

Table I: Correct predictions of controller state: Real robot testing results on the uBot-6 where each fraction denotes the number of correctly predicted trials over the number of grasp attempts that resulted in a particular controller state when $|\dot{\phi}_{\text{GRASP}}| \leq \epsilon$. Of the testing objects, half of them were not physically graspable given the geometries and properties of the object and robot, this is indicated with a '–.' In the case of this table, $\gamma^T(\phi_{\text{GRASP}})$ is abbreviated as γ^T.

reinforcement learning opportunties arise. Our results indicate that the real hardware systems can learn visuomotor skills through control basis state predictions and gives support to perhaps a new form of learning robot skills through employing an extended version of this approach.

The grasping reflexive action presented in this paper draws inspiration from some developmental phenomena. Insight from cognitive psychology indicates that infants are inherently encoded by primitive reflex repertoire and must learn to encode this set of motor skills into intentional actions—as psychologist Zelazo describes, this is one of the first functions of the cognitive process [28]. We demonstrate this using a relatively small convolutional neural network architecture with output neurons mapping to control state (ϕ_{GRASP} in this paper) that is derived from the asymptotic behavior of the controller. We observe that in Figure 9 the validation accuracy with a small number of training epochs is actually higher using a shallower network in the particular case. It may be possible to look into developmental strategies in which robots first learn using shallower networks with a small set of training data to learn coarse behavior quickly. As more data is possible, network structure and number of training epochs may be increased—this assumes that the information encoded in the shallow convolutional layers are similar between varying depth network architectures.

One of the most promising results of work dating back to two decades regarding the *Control Basis* framework is that the underlying controllers are reusable in the way that they are defined independent of hardware—therefore, they can be used to control arbitrary robot morphologies [20]. Extensions of this work look into incorporating a larger number of controllers $\phi|_\tau^\sigma$ at the output layer allowing for the acquisition of time-dependent visuomotor policies over many basis controllers. This allows the visuomotor skill learned in this work to be extended to acquire more complex behaviors—a benefit of adopting the *Control Basis* framework.

This work serves as the preliminary basis for these future extensions supporting the integration between control theoretic architectures with deep learning for visuomotor skills. For instance, since the network encodes the time history of all grasping experiences, we hypothesize that continuous prediction may lead to fine grained error detection. We suspect that this is the case since each forward propagation of the depth image corresponds to prediction whether there is stimulus in the scene that affords a particular action or control program (in this case ϕ_{GRASP}). If at any point the performed action is no longer afforded by the scene, it may be possible to detect and preempt the execution of the selected control program. We hope to explore this further in upcoming work. Other plausible extensions look into developing a reformulation of the network architecture to integrate DQNs with continuous action spaces using actor-critic approaches [11], allowing controllers to act outside fixed stationary goals or reflexive parameters. Using approaches like this, robots may be able to not only self-label their training instances but at the same time, learn goal parameters that result in likely desired controller states. Lastly, we look in a subsequent study to incorporate a deeper motor network involving an abstraction of spinal motor units and cerebellar motor circuitry (the *Control Basis*) that may be able to accomplish force closure grasps in a much wider range of conditions. The perceptual guidance from range data would be more sophisticated as well and more layers may be warranted.

Acknowledgements

This material is based upon work supported under Grant NASA-GCT-NNX12AR16A. Any opinions, findings, conclusions, or recommendations expressed in this material are solely those of the authors and do not necessarily reflect the views of the National Aeronautics and Space Administration.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[6] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[7] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[8] E. Wilkinson and T. Takahashi, "Efficient aspect object models using pre-trained convolutional neural networks," in *EEE-RAS 15th International Conference on Humanoid Robots*, Nov 2015, pp. 284–289.

[9] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[10] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998, vol. 116.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *arXiv preprint arXiv:1504.00702*, 2015.

[13] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[14] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[15] D. Kragic and H. I. Christensen, "Robust visual servoing," *The international journal of robotics research*, vol. 22, no. 10-11, pp. 923–939, 2003.

[16] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[17] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," *arXiv preprint arXiv:1509.06113*, 2015.

[18] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," *arXiv preprint arXiv:1509.06825*, 2015.

[19] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *arXiv preprint arXiv:1603.02199*, 2016.

[20] M. Huber, W. S. MacDonald, and R. A. Grupen, "A control basis for multilegged walking," in *IEEE International Conference on Robotics and Automation*, Apr 1996.

[21] S. Sen and R. Grupen, "Integrating task level planning with stochastic control," University of Massachusetts Amherst, Tech. Rep. UM-CS-2014-005, 2014.

[22] J. A. Coelho Jr and R. Grupen, "A control basis for learning multifingered grasps," 1997.

[23] D. Ruiken, M. W. Lanighan, and R. A. Grupen, "Postural modes and control for dexterous mobile manipulation: the umass ubot concept," in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[25] H.-T. Jung, T. Takahashi, and R. A. Grupen, "Human-robot emergency response-experimental platform and preliminary dataset technical report# um-cs-2014-006," 2014.

[26] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, p. 2, 2012.

[27] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *International Conference on Advanced Robotics*, 2015.

[28] P. R. Zelazo, "The development of walking: new findings and old assumptions," *Journal of Motor Behavior*, vol. 15, no. 2, pp. 99–137, 1983.