Acquiring Transferrable Mobile Manipulation Skills

George Konidaris
MIT CSAIL
Cambridge MA 02139
gdk@csail.mit.edu

Scott Kuindersma, Roderic Grupen and Andrew Barto
Computer Science Department, University of Massachusetts Amherst
Amherst MA 01003
{scottk, grupen, barto}@cs.umass.edu

I. Introduction

This abstract summarizes recent research on the autonomous acquisition of transferrable manipulation skills. We describe a robot system that learns to sequence a set of innate controllers to solve a task, and then extracts transferrable manipulation skills from the resulting solution. Using the extracted skills, the robot is able to significantly reduce the time required to discover the solution to a second task.

II. The uBot-5

The uBot-5, shown in Figure 1, is a dynamically balancing, 13 degree of freedom mobile manipulator [1]. It has two arms with unactuated spherical "hands" that can be used for basic manipulation tasks[1] and a pan/tilt unit with two cameras.

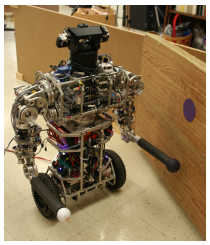

Fig. 1. The uBot-5.

We used the ARToolkit system [2] to identify target objects present in the robot's visual field. The uBot was able to identify the location and orientation of visible tags with an update rate of 8Hz.

The robot had access to a set of innate navigation and manipulation controllers. Given a target object, the navigation controller first aligned the robot with the wall normal at the object's location, then turned the robot to face the object and approach it. The manipulation controllers moved its hand to one of seven positions: withdrawn, extended, and then extended and moved to the left, right, upwards, downwards, or outwards. Each of these controlled the position of the hand relative to the centroid of the target object.

III. Learning to Solve a Mobile Manipulation Task

The uBot first learned to solve a mobile manipulation task. The robot began the task in a small room containing a button and a handle. When the handle was turned after the button had been pressed a door in the side of the room opened, allowing the uBot access to a compartment containing a switch. The goal of the task was to press the switch. Sensing and control for the objects in the room was performed using touch sensors, with state tracked and communicated to the uBot via an MIT Handy Board [7]. Figure 2 shows a schematic drawing and photographs of the first task.

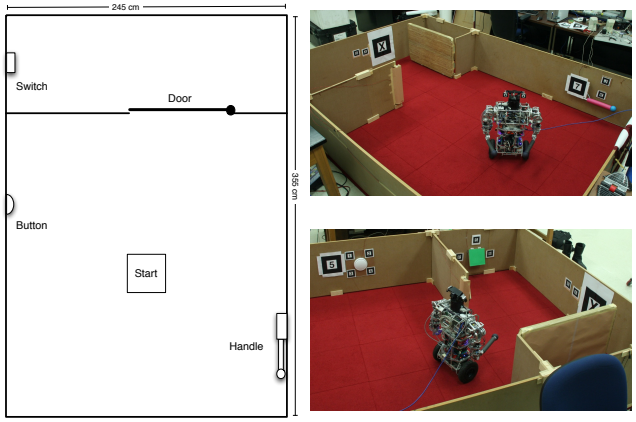

Fig. 2. The first task in the Red Room Domain.

The state of the first task at time i was described as a tuple $s_i = (r_i, p_i, h_i)$, where r_i was the state of the room, p_i was the position of the robot, and h_i was the position of its hand. The state of the room at time i consists of four state bits, indicating the state of the button (pressing the button flipped this bit), the state of the handle (this bit was only flipped once per episode, and only when the button bit was set), whether or not the door was open, and whether or not the switch had been pressed (this bit was also only flipped once per episode since the episode ended when it was set). The uBot could find itself at one of five positions: its start position, in front of the button, in front of the handle, through the door, and in front of the switch. Similarly, its hand could be in one of seven positions: withdrawn (allowing it to execute a navigation action), extended, and then extended and moved to the left, right, upwards, downwards, or outwards.

The uBot solved the task by interacting with the room to learn a discrete model of the task as a Markov Decision Process (MDP) and then planning using dynamic programming. It was able to find the optimal controller sequence after 5 episodes of interaction, reducing the time the robot required to

[1] A grasping hand prototype is expected to be working shortly.

complete the task from approximately 13 minutes to around 3. This did not include the time required by hardwired controllers not subject to learning (e.g., the controller that safely oriented the uBot from one object to another).

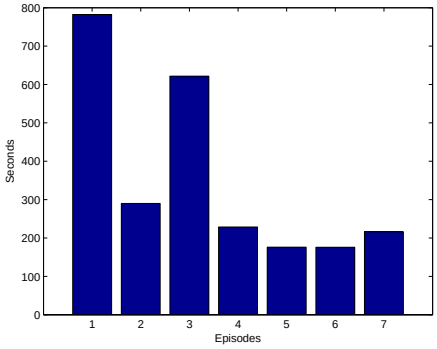

Fig. 3. The uBot's learning curve in the first task.

IV. Extracting Transferrable Skills

The resulting optimal sequence of controllers was used to generate 5 demonstration trajectories for use in CST, a skill acquisition algorithm [5]. The uBot was given a library of abstractions for use during skill acquisition. Each abstraction paired one of the uBot's motor modalities (body or hand) with a task object, and contained features describing the difference between the relevant effector and the relevant object. The use of skill-specific abstractions is critical in extracting transferrable skills because it allows the robot to abstract away the positions of the other objects in the environment when creating a skill policy. In addition, it significantly reduces the number of policy parameters and therefore greatly simplifies policy improvement. An example segmentation is shown in Figure 4. A description of each skill along with its relevant abstraction is given in Figure 5.

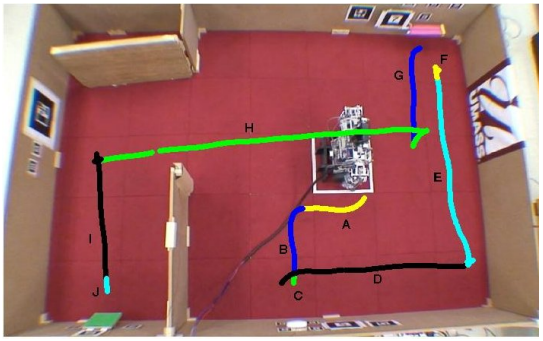

Fig. 4. A trajectory from the learned solution to the first task, segmented into skills.

CST extracted skills that corresponded to manipulating objects in the environment, and navigating towards them. We do not consider the resulting navigation skills further since they are room-specific and are not transferrable. In the manipulation case, sequences of two controllers were collapsed into a single skill: for example, extending the hand and then reducing the distance between its hand and the button to zero was collapsed into a skill which we might describe as *push the button*.

#	Abstraction	Description
A	body-button	Align with the button.
B	body-button	Turn and approach the button.
C	hand-button	Push the button.
D	body-handle	Align with the handle.
E	body-handle	Turn and approach the handle.
F	hand-handle	Turn the handle.
G	body-entrance	Align with the entrance.
H	body-entrance	Turn and drive through the entrance.
I	body-switch	Approach the switch.
J	hand-switch	Press the switch.

Fig. 5. A brief description of each of the skills extracted from the trajectory in Figure 4, with selected abstractions.

We fitted the resulting policies for replay using a single demonstrated trajectory, and obtained reliable replay for all manipulation skills. A new closed-loop controller was synthesized for each acquired skill by fitting a spline to the solution trajectory to identify a sequence of relative waypoints. This allowed robust replay while retaining the ability to learn to improve each controller using a policy search algorithm—e.g., Kohl and Stone [3]—if necessary.

V. Using Acquired Skills to Solve a Novel Task

We tested the performance of the uBot with acquired skills in a second task similar to the first: the robot was placed in a room with a group of manipulable objects and a door. In this case, it had to first push the switch, and then push the button to open the door. Opening the door hid a button in the second part of the room. The robot had to navigate to the second part of the room and turn a handle to close the door again. This revealed the second button, which it had to press to complete the task. The state of the second task was represented in a similar manner to that of the first.

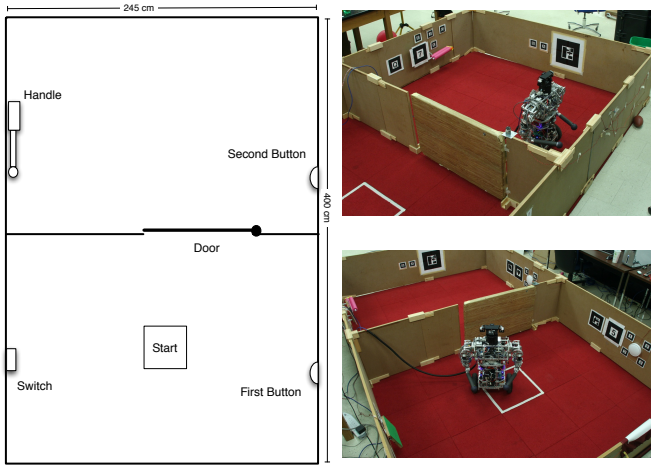

Fig. 6. The second task in the Red Room Domain.

Note that this room contained the same object types (but in a different geometric configuration) as the first task, and so the robot was able to apply its acquired skills to manipulate them. In general, object classification would require visual pattern matching, but for simplicity we provided object labels.

Figure 7 shows the interaction time required for the uBot's first attempt at completing the second task, given either its original innate controllers or, additionally, the manipulation skills acquired in the first Red Room task (again, this does not include controllers not subject to learning, which are the same in both cases). We performed 8 runs of each condition.

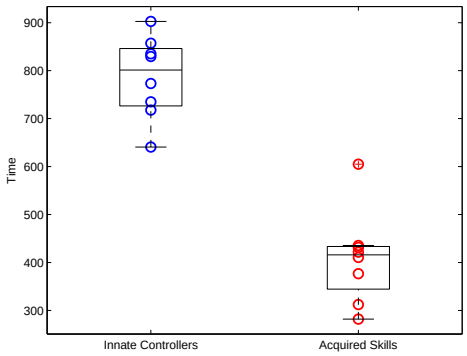

Fig. 7. The time required for the uBot-5 to first complete the second task, given innate controllers or acquired skills.

The presence of acquired skills nearly halved the mean interaction time required (from 786.39 seconds to 409.32 seconds), and this difference is significant (Welch two-sample t-test, $t(13.785) = 8.22, p < 0.001$); moreover, the sampled times for the two conditions do not overlap.[2]

VI. Summary and Conclusions

We have described a robot system that can acquire manipulation skills by learning to solve one problem, and then apply the resulting skills to improve performance in another. It is worth considering the implications of these results. Although the uBot started off with the capacity to *learn* to, for example, push the button, this was accomplished through a laborious process of trial-and-error exploration through many combinations of manipulation actions within a particular task. However, since this sequence of manipulation actions happened to be useful in *solving a problem*, it was extracted as a single action that can be deployed as a unit—requiring only a single action selection decision—when the robot encounters a new problem. Had the uBot attempted transfer its *entire* policy from the first task to the second, it would have performed very poorly. Instead, transfer was achieved via the isolation and retention of skills—*policy components*—that are feasible for learning and suitable for reuse in later tasks due to their use of skill-specific abstractions. Please see Konidaris et al. [6] and Konidaris [4] for more details.

Versatile mobile manipulators will require a combination of planning (using what is known about their own actuators and the world) and learning (to discover the remainder) to operate effectively in open environments. We expect that the ability to retain, refine, and redeploy learned procedural knowledge—particularly in the form of acquired manipulation skills—will both simplify planning (regardless of the sophistication of the planner used) and reduce the actual effort expended by the such robots when solving new tasks.

[2]Note that one of the runs using skill acquisition is marked as an outlier (with a cross) in Figure 7. During this run, the robot explored virtually all transitions available in the MDP before finally finding the solution. This corresponds to near worst-case behavior using acquired skills; it still requires less time (by about 30 seconds) than the fastest observed run using only innate controllers.

Acknowledgments

We would like to thank the members of the LPR for their technical assistance. AGB and GDK were supported in part by the AFOSR under grant FA9550-08-1-0418. GDK was also supported in part by the AFOSR under grant AOARD-104135 and the Singapore Ministry of Education under a grant to the Singapore-MIT International Design Center. SRK is supported by a NASA GSRP fellowship from Johnson Space Center. RAG was supported by the Office of Naval Research under MURI award N00014-07-1-0749.

References

[1] P. Deegan, B. Thibodeau, and R. Grupen. Designing a self-stabilizing robot for dynamic mobile manipulation. In *Proceedings of the Robotics: Science and Systems Workshop on Manipulation for Human Environments*, August 2006.

[2] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, 1999.

[3] N. Kohl and P. Stone. Machine learning for fast quadrapedal locomotion. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 611–616, 2004.

[4] G.D. Konidaris. *Autonomous Robot Skill Acquisition*. PhD thesis, University of Massachusetts Amherst, May 2011.

[5] G.D. Konidaris, S.R. Kuindersma, A.G. Barto, and R.A. Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In J. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1162–1170, 2010.

[6] G.D. Konidaris, S.R. Kuindersma, R.A. Grupen, and A.G. Barto. Autonomous skill acquisition on a mobile manipulator. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, 2011.

[7] F.G. Martin. *The Handy Board Technical Reference*. MIT Media Lab, Cambridge MA, 1998.