TRANSFERRING SKILLS TO A SIMULATED ROBOT

MOHAMMAD H. SALAH
ECE Department, Clemson University, Clemson, SC 29634
msalah@ces.clemson.edu

ANDREW L. KUN
ECE Department, University of New Hampshire, Durham, NH 03824
andrew.kun@unh.edu

ABSTRACT:
A skill transfer and improvement framework was designed and implemented to transfer skills to a simulated robot without the help of a robotics expert. A skill expert first learns to perform the desired task by maneuvering the robot and giving actuator commands using the keyboard. The framework is then used to transfer the expert's skills to the robot by teaching the robot's controller to perform the skill. CMAC neural networks are responsible for learning skills. Learning is accomplished using imitation of the demonstrated movements that are mapped to a set of primitives. After transferring a skill, the expert can give advice to the controller to improve the robot's performance.

INTRODUCTION

The problem tackled in this research is how to transfer a skill from a skill expert to a robot. A skill expert is a person who possesses the skill to perform a task but is not a robotics expert. With today's technology we need robotics experts to program robots to perform some specific task. Consequently, a skill expert who is not a robotics expert cannot easily transfer skills to a robot.

Our first hypothesis is that an adaptive learning controller [1] can be used to implement the skill transfer. Adaptive control will allow adaptation to variations in the environment of the robot. The second hypothesis says that the learning controller can be based on the idea of cloning [1]. Cloning is also called learning by example, or imitation learning by researchers [2-4]. Cloning design allows imitating the skill(s) demonstrated by an expert. The third hypothesis proposes that the expert first needs to learn how to execute the skill by maneuvering the robot (controlling the robot's actuators) while observing the feedback from the robot's sensors. Then the expert should provide the imitation controller with pairs of state inputs (functions of sensor readings) and desired actuator outputs. The fourth hypothesis proposes that all robot skills should be described using action primitives, where primitives describe simple robot motions taking into account interactions with the environment. All skills should be described as concatenated primitives, where combined primitives represent skills. The fifth hypothesis deals with improving the performance of the imitation controller. The human expert can let the controller execute a skill and then give an advice to the controller on how to improve the performance of the skill. This advice can be advice about desired outcomes and advice about the state space variables.

Our research is constrained to robots with a small number of degrees of freedom interacting with a small number of static objects. Thus we will consider a two dimensional (2D) model of a truck with a trailer, a parking spot, and

obstacles. The technique of transferring skills will handle some of the kinematics of the robot and it will not handle any dynamics.

BACKGROUND

Daxwanger et al. have presented an approach to acquire and transfer an experienced driver's skills to an automatic parking controller [5] using an artificial neural network and fuzzy network-based controller. Skill transfer for an experimental space robot was implemented at the German Aerospace Center (DLR) [6]. As Schaal points out in [3], imitation learning of skills can be implemented by learning a control policy directly from a task expert or by building a new policy from observations of a behavior. In our system the teacher will control and move the robot directly based on the feedback obtained from the robot sensors. This means that the teacher's states and actions will all be known and observable. Schaal defines movement primitives as "sequences of action that accomplish a goal-directed behavior." The idea of motion primitives comes from studies that suggest the existence of force-field motor primitives in the spine. Studies on frogs and rats showed that these primitives are vector-additive, and can encode a large number of motor behaviors [7]. Mosemann and Wahl define a skill primitive as a 6-tuple that includes the movement to be performed, the set of sensors to observe the primitive execution, and the goals for sensor readings at the beginning and end of the primitive [8]. In this paper, we define motion primitives similarly to the definition of Mosemann and Wahl.

PRIMITIVE-BASED CONTROL

We will consider primitives as the basic units that describe any kind of motion. Primitives will describe our truck motions in terms of actuator commands and sensor feedback. We will define a primitive as:

$$P = < commands, transformation, stimuli, goals, references > \quad (1)$$

The *commands* element of the primitive represents the actuator commands that will result in the desired robot motion. The state input is created based on the outputs of relevant sensors and reference values provided by the expert as shown in Fig. 1 that shows a view of our primitive-based control architecture.

In this control architecture, the expert will select a primitive for execution from a set of stored primitives. Then the *stimuli*, *transformation*, *goals* and *references* elements of the selected primitive will be retrieved.

The *references* element makes the primitives independent of absolute locations in the state space. The *stimuli* element of the primitive provides information about which sensors are relevant. The *commands* element in the primitive *P* specifies the set of actuators that should be activated in order to execute the motion. The *transformation* element determines actuator commands based on the state of the robot (input). The sensor outputs associated with the references in turn can determine the robot's state. Once the state of the robot is quantified, actuator activation can be controlled by the transformation element of the primitive *P*.

Fig. 1 introduces the idea that the performance of the *transformation* element of a primitive can be adaptively learned as well as improved. Every group of actuators in the commands element of the primitive has an associated error. The *goals* element of the primitive specifies these functions. The transformation is adaptively changed to minimize the errors.

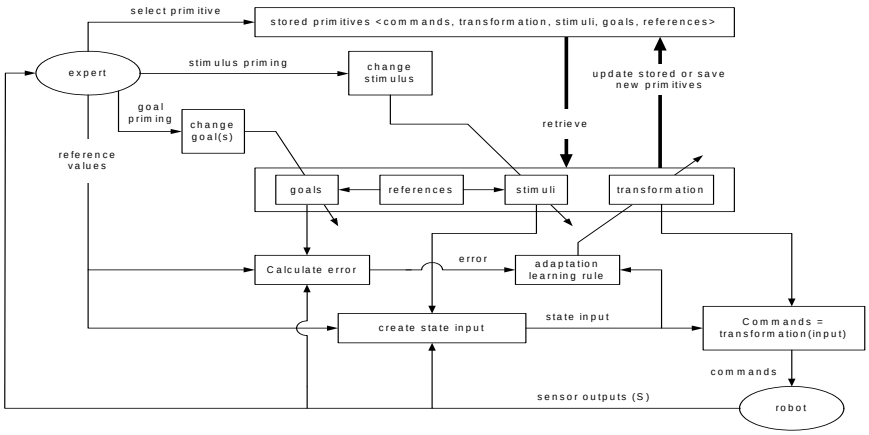

Fig. 1 *A view of a primitive-based control architecture*

SKILL EXPERT ADVICE

Adaptation allows improving the *transformation* element of primitives. However, when a primitive is executed, the skill expert may realize that training the *transformation* element using the existing goals does not improve the performance as desired. Fig. 1 shows a system that allows changing the *goals* element. In this system the expert can retrieve a stored primitive and can then initiate goal and/or stimuli priming. Goal/Stimuli priming refers to adding, removing or changing one or more of the goal/stimuli functions. Stimulus priming is needed if the primitive's state space has to be modified by the expert.

PRIMITIVE-BASED MOTION DESCRIPTION OF A TRUCK

The expert can create different combinations, sequentially, of three basic primitives to describe the desired path or skill to be transferred. For the sake of simplicity, we predefine three basic primitives as follows:

- Primitive 1 which is "Go_Straight." It is implemented just on the truck, with the trailer following. This primitive is implemented when the angle of the front wheels of the truck are aligned with the angle of the truck itself.
- Primitive 2 which is "Make_A_Right_Turn." The front wheels of the truck turn right by 20°, and the truck moves forward with the trailer following. This angle is the angle between the truck's angle and the front wheels' angle.
- Primitive 3 which is "Make_A_Left_Turn." It is the same as primitive 2 but executes a left turn.

The argument values of the primitive equation change depending on the type of primitive to be executed. According to equation (1), our basic primitives are:

$P_1 = < GF, T\#, Sgn(CDSR - RV), G, RV >$ where:

***P_1*:** "Go_Straight" primitive.
***GF*:** Actuator command of going forward.
***T#*:** Transformation number.
***CDSR*:** Current distance sensor reading.
***G*:** Goal that relies on (current distance sensor reading – reference value).
***RV*:** Reference value (either certain value or previous distance sensor reading).
Sgn: Sign of the difference.

$P_2 = <GF \cup TR, T\#, (RV - TOAS), G, RV >$ where:

***P_2*:** "Make_A_Right_Turn" primitive.
***TR*:** Actuator commands of turning the wheels right.
***TOAS*:** Truck's orientation angle sensor.

$P_3 = < GF \cup TL, T\#, (TOAS - RV), G, RV >$ where:

***TL*:** Actuator command of turning the wheels left.

The parallel activation operator, $\cup$, indicates that the actuators are activated concurrently. Fig. 2 depicts an example of how we can execute the basic primitives by determining the *stimuli*, *goals* and *references* elements.

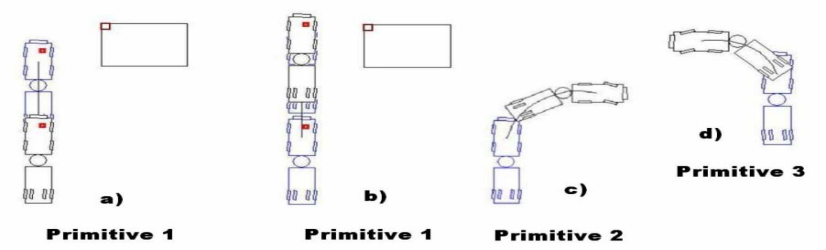

Fig. 2 *Primitives 1, 2, and 3*

In panel a) of Fig. 2 we use a distance sensor to measure the distance from a location on the truck to a corner of the obstacle (the upper left corner in this example). In this example our reference is a specific distance *d* between the sensor and the corner. Our goal is to minimize the difference between the distance sensor reading and the reference value. In panel b), our goal is to obtain the minimum distance to the same corner while keeping the current truck heading. In order to minimize this distance we observe the sign of the difference between the current and past distance sensor readings. When the sign of this difference changes, it can be said that the truck has reached the minimum distance to the corner. The reference value is the last distance reading. For panel c) of Fig. 2, a right turn is executed. The sensor that measures the truck's angle is activated and the reference value is adjusted to be 0°. The goal is to minimize the difference between the angle of the truck and the reference value. Panel d) of Fig. 2 is the same as panel c) except the sign of the angles change.

SIMULATION RESULTS

Fig. 3 shows an example of a skill to be transferred to the truck.

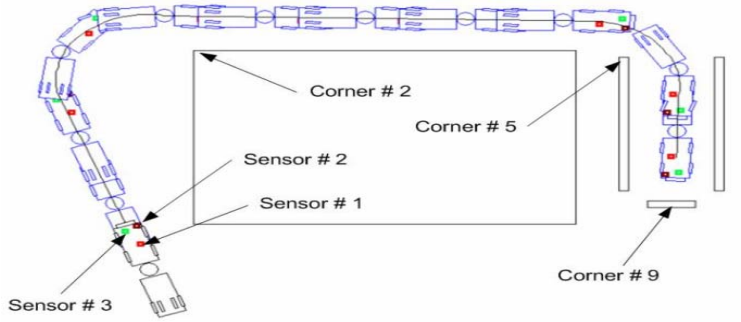

Fig. 3 *Skill to be transferred to the truck*

The expert will try to reach the parking spot by making a big turn around the obstacle. This skill can be divided into five primitives in the order: P_1, P_2, P_1, P_2, and then P_1. Next, the expert has to let the truck's controller learn how to implement these primitives. The learning will be done by the CMAC Neural Networks (NNs) [9] located in the imitation-learning controller. The CMAC NNs represents the transformation element defined in equation (1). After learning, the expert has to combine these two learned primitives to accomplish the skill sought. The truck's controller will learn how to execute the primitives sequentially according to the primitive. Our controller will perform correctly even if the truck's position or angle changes (within limits) compared to Fig. 3. The truck follows the same order of basic primitives and makes turns as desired.

Fig. 4 depicts a situation where a new obstacle (red circle) is added to the environment and set in the path of the truck. In this case, an advice can be made by changing the state input to get desired outcomes in order to overcome a variation in the truck's environment. This could be done by adding a sensor to the sensors installed on the truck to measure the distance from the truck's position to the center of the new obstacle. In our implementation, the advice requires adding a CMAC NN to prevent the truck from moving over it.

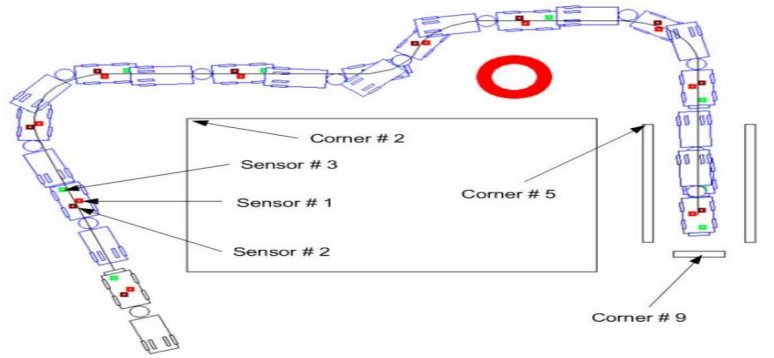

Fig. 4 *Advising the truck by changing the state input and adjusting the path*

Under some circumstances, giving advice will not solve the problem (variations in the environment) encountered by the truck. In this case, a new combination of primitives (a new skill) is needed.

CONCLUSION

In this paper we proposed a machine-learning framework that allows a layperson to transfer skills to a simulated robot. The elements that make up this imitation technology are adaptation, imitation-based learning and primitives that describe the truck's motion. Skill transfers are performed using imitation where the demonstrated movements are mapped to a set of primitives to implement novel skills. Creating a similar framework to transfer skills to physical robots could have a tremendous effect on our everyday lives that would be similar to the effect that software applications for PCs had on our society.

ACKNOWLEDGMENT

This research was funded in part by the University of New Hampshire.

REFERENCES

[1] P. J. Werbos, "*Neurocontrollers*," in J. G. Webster (ed.) Wiley encyclopedia of electrical and electronics engineering, John Wiley, 1999, pp. 350-366.

[2] M. Mataric, "*Sensory-motor primitives as a basis for imitation: linking perception to action and biology to robotics*," in C. Nehaniv and K. Dautenhahn (eds.) Imitation in Animals and Artifacts, MIT Press, 2001.

[3] S. Schaal, "*Is imitation learning the route to humanoid robots?*," Trends in Cognitive Sciences, vol. 3, no. 6, pp. 233-242, 1999.

[4] A. Ude, C. Man, M. Riley, and C. G. Atkeson, "*Automatic generation of kinematic models for the conversion of human motion capture data into humanoid robot motion*," First IEEE-RAS International Conference in Humanoid Robots, Cambridge, MA, 2000.

[5] W. A. Daxwanger and G. K. Schmidt, "*Skill-based visual parking control using neural and fuzzy networks*," IEEE International Conference on Systems, Man and Cybernetics, 1995, 'Intelligent Systems for the 21st Century', vol. 2, no. 1695-1664, Vancouver, BC , Canada.

[6] B. Brunner, G. Hirzinger, K. Landzettel, and J. Heindl, "*Multi-sensory shared autonomy and tele-sensor-programming - key issues in the space robot technology experiment ROTEX*," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Yokoham, Japan.

[7] F. A. Mussa-Ivaldi and E. Bizzi, "*Motor learning through the combination of primitives*," Philosophical Transactions of the Royal Society of London: Biological Sciences, vol. 355, no. 1404, pp. 1755-1769, 2000.

[8] H. Mosemann and F. M. Wahl, "*Automatic decomposition of planned assembly sequences into skill primitives*," IEEE Transactions on Robotics and Automation, vol. 17, no. 5, pp. 709-718, 2001.

[9] J. S. Albus, "A new approach to manipulator control: the Cerebellar Model Articulation Controller (CMAC)," Trans. of ASME, Journal of Dynamic Systems, Measurement, and Control, vol. 97 pp. 220-227, Sep 1975.