Creation and Analysis of Sensory Drivers Using ERL

Matthew Robert Moore, TaeHoon Anthony Choi , Eric M. Schwartz,
Michael Nechyba, A. Antonio Arroyo

Abstract— **With most autonomous agents individual sensor drivers are coded for each agent that is constructed. This is not a viable solution for large scale production. This paper presents a process by which a universal sensor diver is created autonomously using environmental reinforcement. The agent is trained in a highly structured environment therefore eliminating the task of coding individual sensor drivers for each agent manufactured and allowing the programmer to focus on higher level behaviors. In the environment the agent is trained with specific scenarios to create the sensor driver, which will allow the agent to recognize those scenarios when outside of its environment. An autonomous mobile agent, Talrik, experimentally demonstrates which scenarios are detected when objects are placed at locations other than the ones for which the scenarios were originally created and which scenarios are detected when an object is slowly stretched into a wall. These experiments show that the sensor driver created using this process is versatile enough to recognize objects within a region rather than at just a specific location.**

*Index Terms*—**Robotics, sensor templates, sensor drivers.**

I. INTRODUCTION

In most of the current research in autonomous mobile agents the primary focus has been on high level learning and performing complicated tasks [1; 2; 3; 4]. This high level learning and complicated behaviors cannot take place without basic primitive behaviors. In order for the high level tasks to be performed the high level code must be able to communicate with the agent's sensors. Depending on the accuracies of the sensors used the driver that allows for this communication can require a great deal of time and work from the programmer. Due to variations in each sensor, this process has to be repeated for each agent that is constructed, even if the same model sensor is used. One way to eliminate this problem is to use more accurate sensors, however this usually results in much larger financial cost.

This paper describes a process by which the large amount of time spent by the programmer writing sensor drivers could be easily eliminated by instead creating these drivers using parts of Environmental Reinforcement Learning (ERL). Environmental Reinforcement Learning (ERL), is a learning process where the agent refines the primitive behavior through its interactions with a highly structured environment. The agent executes the behaviors in a highly structured environment from which the agent learns through Environmental Reinforcement (ER)[5]. Previous work has been done using Inate learning in the self calibration of senors[6] and using ERL to create primitive actuator drivers[5]. Using ERL a universal sensor driver could be created for each platform by simply training the agent in a structured environment. Once the agent has been trained and the universal driver created, higher level programming can be done without the need to worry about reading individual sensors; instead, the agent can be programmed to respond to different scenarios such as "if you see a dead end turn around." Since almost any agent can be trained using this method to create a universal sensor driver higher lever code can be easily transported across different robot platforms. This would allow programmers to share code and reduce development time that is wasted reproducing code to interface with each robots unique sensor suit.

II. CREATION OF SENSOR DRIVERS

In order to create the universal sensor drivers a highly structured environment must be created. In this environment the agent is exposed to different scenarios, where each scenario corresponds to a different configuration of objects. For each scenario a sensor template is created by taking six sensor reading, dropping the highest and lowest value for each sensor to eliminate environmental noise and then averaging the remaining values. These sensor templates are then used to create a library that will be used by the agent once it is outside

of the structured environment. This library will allow for the agent to distinguish between different scenarios in a real world setting. Since this library can be created for almost any sensor suit it would allow for a sensor driver to be easily created for almost any autonomous agent.
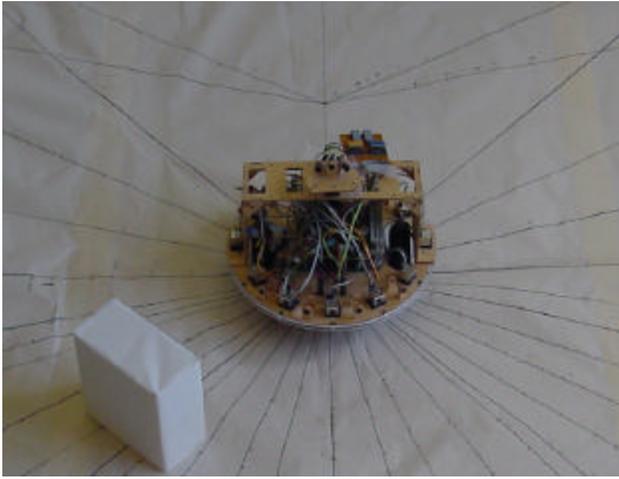
Figure 1: Talrik in its environment

A. Environment

The environment must remain static for each scenario since multiple robot platforms will all be trained using the same environment. Since the environment does not change between agents each agent will be exposed to exactly the same scenarios. When the sensor drivers a individually coded programmers can add their own personal bias and interpretation into the driver, this will not happen when the agent is trained in a static environment. Objects will be added and removed to create each of the different scenarios. The environment used for this experiment is shown in Figure 1. The figure shows the, agent, Talrik in its environment, which is currently set up for a specific scenario.

B. Scenarios

Each scenario consists of one or more objects arranged in a specific manor about the agent. The types of scenarios created so far include an "object to the front", "close object "a wall to the right", "a corridor", and "a dead end to the front." The "object" scenarios were each created by placing a 5 ¼ x 5 ¼ inch block into the environment a specific location. The "wall" scenarios were created by placing a 5 ¼ x 42 inch block into the environment. A complete list of scenarios currently used can be found in Appendix A.

C. TalrikII: An Autonomous Mobile Agent

The robot platform used to perform the experiments mentioned in this paper is Talrik II.Talrik II, shown in Figure 2, fits into a right circular cylinder of ten inch diameter and ten inch height. The Mekatronix MRC11 robot controller provides the Talrik II with an MC68HC11 microcontroller, 64 Kbytes of memory, and memory mapped I/O. The MRSX01 board has recharging circuitry with diode protection. Twelve IR emitters illuminate Talrik II's environment with invisible infrared light. The Talrik II's sensor suite includes 14 IR detectors,10 bumper switches, 6 CDS photoresistors, battery voltage level detect, and recharge current detect.[6]

III.   ANALYSIS OF SENSOR DRIVERS

After the sensor drivers were created they were first tested within the same environment in which they were created to determine which scenario was the most likely candidate for the current scenario. The current sensor reading was compared to the library of templates using a modified Euclidian distance . The formula for this modified distance is shown in Equation 1, where s(k) is the current sensor value from the kth sensor, st(k) is the sensor template value for the kth sensor. This is done for each sensor up to the nth sensor. This value is computed for each sensor template, and the sensor template with the minimum value is chosen as the current scenario.

$$\sum_{k=0}^{n} (s(k) - \text{st}(k))^2$$

Equation 1

With the sensor drivers created they were first tested within the same environment in which they were created. A portion of the results calculated using Equation 1 is shown in Table 1. This table shows that for each scenario the minimum value occurred for the same scenario, resulting in a diagonal of minimum values.

A. Sensor Template Mapping
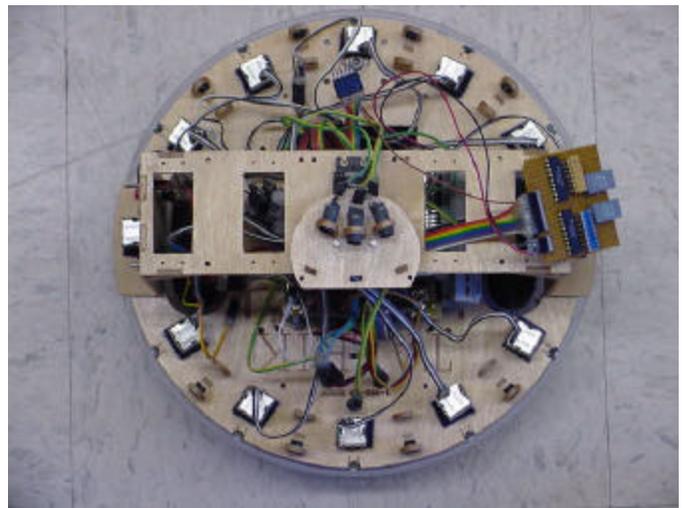
Figure 2: Autonomous mobile agent Talrik

Table 1: Distance between some of the sensor templates (partial table)

	FC	F	FRC	FR	FLC	FL	LC	L	RC	R	BC	B
FC	3	116	281	223	306	194	473	288	512	276	431	265
F	43	0	176	45	285	34	258	65	288	52	207	41
FRC	206	181	0	74	548	231	424	231	310	180	373	207
FR	42	44	60	0	388	69	261	68	234	40	210	44
FLC	399	322	565	406	1	199	286	342	631	395	550	384
FL	39	33	234	65	172	0	172	43	289	49	203	37
LC	237	253	419	255	291	174	0	106	477	241	389	230
L	26	49	225	55	319	35	108	0	278	38	189	26
RC	264	289	329	238	614	292	483	290	0	152	432	266
R	18	43	183	34	368	46	237	44	141	0	186	20
BC	184	209	385	215	534	212	396	207	438	198	8	137
B	4	29	205	35	354	32	223	30	258	18	132	0

Mapping of the sensor templates was created by moving an object to various coordinate locations in a semi circle region, as shown in figure 3. A block, 5 ¼ x 5 ¼ inch, was placed at different locations within a 180 degree region about the robot. For this experiment the object was placed from 7 to 19 inches radialy outward from the center of the agent. This was done every 10 degrees for the 180-degree section in front of the agent. The results are shown in Figure 3 where each color corresponds to a different object seen by the agent. Looking at Figure 3 we see each colored region in the graph corresponds to an area where a certain scenario will be detected when an object is placed within that region. This shows that the sensor driver created is versatile enough to recognize a scenario when the object is not placed at the exact location for which a scenario was created. The scenario will be identified as long as it resides within that objects region.

B. Object Morphing

Till now all the experiments have focused on testing the sensor drivers with the same sized objects used to create the sensor templates for each scenario. In the real world the agent will encounter objects that are different sizes. One question is how long must an object be to be considered a wall. A normal 5 ¼ x 5 ¼ inch object was extended by 2 inches each time a sensor reading was taken and the ends of the object were plotted in Figure 4 with each color corresponding to a different object seen by the agent. Figure 4 shows that the sensor suit detects a far wall as the object gets longer, however for each object tested the transition to the wall at that position occurred

with the object at the same length. This shows that the sensor driver can adapt to objects of varying size and still be able to recognize the scenario.

IV. CONCLUSION

In this paper the creation of sensor drivers using ERL was examined. Including the importance of the static environment and the creation of the each sensor template corresponding to its own unique scenario, resulting in a universal driver. This universal driver will allow the agent will be able to recognize different scenarios outside of its environment and react accordingly. With a universal driver, time that would have otherwise been used to code and tweak a sensor driver could now be spent on other projects.

The sensor driver was created and tested to see what would happen when the same object was placed near the original location. We observed that the regions created for "object front," "object right," etc. were contiguous and took on a value that was physically nearest to the original template location. The second test was to see what would happen when the size of the object was stretched. As an object was lengthened, we observed that it transitioned smoothly from "object front close" to "wall front far" to "wall front." When the objects changed shape by small amounts, they were still recognized; however, when they were stretched, they would eventually have a clear transition to a wall at that location after and intermediate scenario.
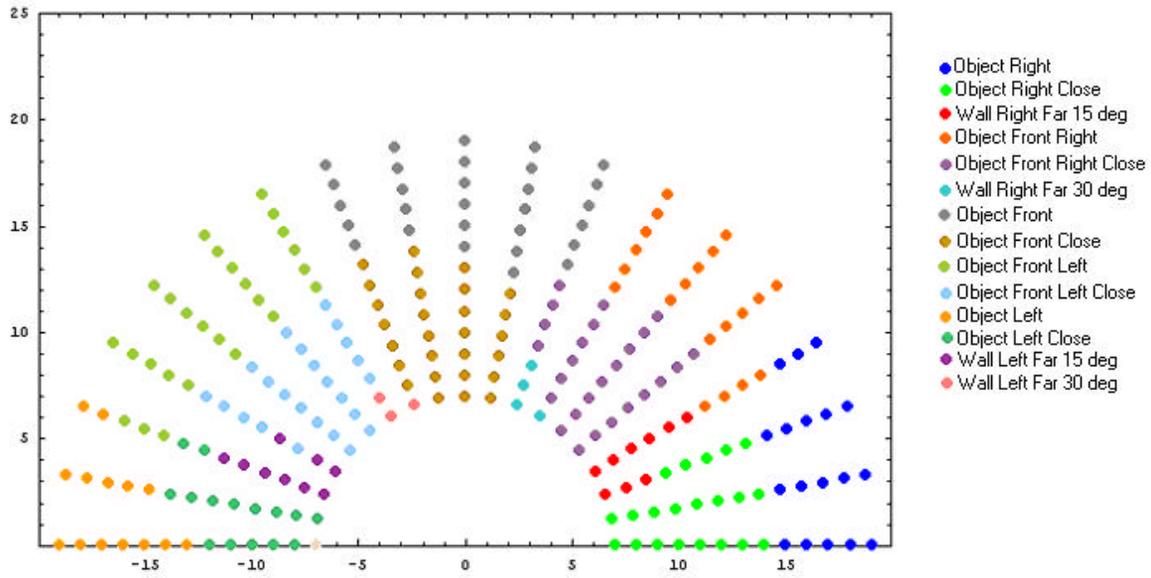
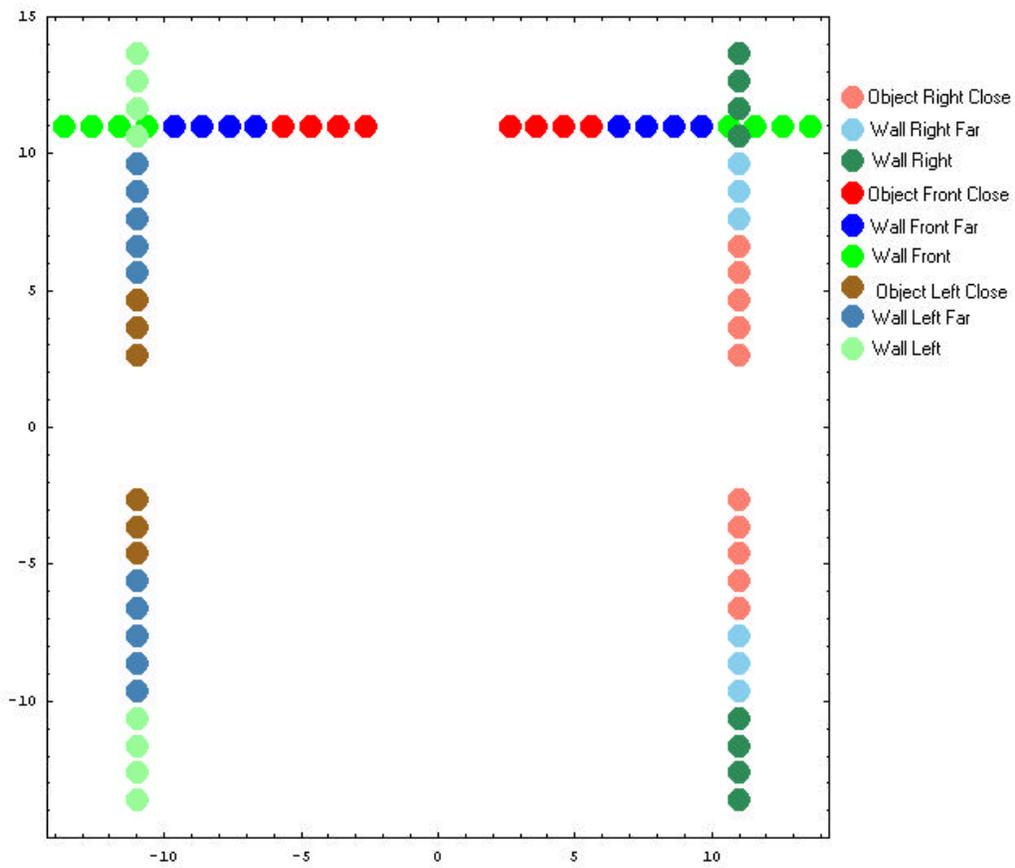**Figure 3:Mapping for object region**

Figure 4: Object morphing due to change in length of an object

V. References

[1] Long-Ji Lin. Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching. In *Machine Learning* **8**, pages 293-321, Kluwer Academic Publishers, 1992

[2] Lynne E. Parker. Adaptive Action Selection for Cooperative Agent Teams. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 442-450, MIT Press, 1992

[3] Ashwin Ram and Juan C. Santamaria. Multistrategy Learning in Reactive Control Systems for Autonomous Robotic Navigation. In *Informatica* **17 (4)**, pages 347-369, 1993.

[4] Mark B. Ring. Continual Learning in Reinforcement Environments. *Ph.D. Thesis*, University of Texas, 1994

[5] T. A. Choi, E. A. Yim, and K. L. Doty, "Environmental Reinforcement Learning: A Real-Time Architecture for Primitive Behavior Refinement," *ROBOLEARN 96: An International Workshop on Learning for Autonomous Robots*, May 19-20, 1996, Key West, Florida, pages 20-27.

[6] T. A. Choi, E. A. Yim, A. A. Arroyo, and K. L. Doty, Automatic Configuration of Sensors and Actuators Through Innate Learning, *Robotics 98: The 3rd International Conference and Exposition on Robotics for Challenging Environments*, Albuquerque, NM, 26-30 April 1998, 64-70.

I. Appendix A

Object Front Close

Object Front

Object Front Right Close

Object Front Right

Object Front Left Close

Object Front Left

Object Right Close

Object Right

Object Left Close

Object Left

Object Back Close

Object Back

Object Back Right Close

Object Back Right

Object Back Left Close

Object Back Left

Wall Front Close

Wall Front

Wall Front Far

Wall Right Close

Wall Right

Wall Right Far

Wall Left Close

Wall Left

Wall Left Far

Wall Front Right 15 deg Close

Wall Front Right 15 deg

Wall Front Right 15 deg Far

Wall Front Left 15 deg Close

Wall Front Left 15 deg

Wall Front Left 15 deg Far

Wall Back Right 15 deg Close

Wall Back Right 15 deg

Wall Back Right 15 deg Far

Wall Back Left 15 deg Close

Wall Back Left 15 deg

Wall Back Left 15 deg Far

Wall Front Right 30 deg Close

Wall Front Right 30 deg

Wall Front Right 30 deg Far

Wall Front Left 30 deg Close

Wall Front Left 30 deg

Wall Front Left 30 deg Far

Wall Back Right 30 deg Close

Wall Back Right 30 deg

Wall Back Right 30 deg Far

Wall Back Left 30 deg Close

Wall Back Left 30 deg

Wall Back Left 30 deg Far

Corner Front

Corner Front Right

Corner Front Left

Corner Right

Corner Left

Corner Back Right

Corner Back Left

Corner Back

Corridor

Corridor Shifted Right

Corridor Shifter Right Twice

Corridor Shifted Left

Corridor Shifter Left Twice

Corridor Front Right 15 deg

Corridor Front Right 15 deg Shifted Right

Corridor Front Right 15 deg Shifter Right Twice

Corridor Front Left 15 deg

Corridor Front Left 15 deg Shifted Right

Corridor Front Left 15 deg Shifter Right Twice

Corridor Front Right 30 deg

Corridor Front Right 30 deg Shifted Right

Corridor Front Right 30 deg Shifter Right Twice

Corridor Front Left 30 deg

Corridor Front Left 30 deg Shifted Right

Corridor Front Left 30 deg Shifter Right Twice

Dead-end Front

Dead-end Right

Dead-end Left

Dead-end Back