Teaching Deliberative Navigation Using the LEGO RCX and Standard LEGO Components

Gary R. Mayer*, Jerry B. Weinberg†, Xudong Yu‡
Department of Computer Science, School of Engineering
Southern Illinois University at Edwardsville
*malekith@alum.wpi.edu, †jweinbe@siue.edu, ‡xyu@siue.edu

Abstract

A number of universities are using inexpensive robotic platforms to teach artificial intelligence and robotics courses – for examples, see the IEEE Robotics & Automation Magazine, vol. 10, no. 2, June 2003 [1]. The LEGO Mindstorms set is one of the most popular platforms. This set is frequently chosen for its low cost and ease of use. However, most of the work being done with it is in teaching reactive robotic architectures. Other, more expensive platforms are used for the deliberative and hybrid robotic architectures. This paper explains how an off-the-shelf LEGO Mindstorms kit can be used to teach deliberative navigation, including path planning and mapping.

Introduction

The intent of this project is to integrate LEGO Mindstorms into the curriculum as a tool for teaching deliberative robotic control. It consists of using a standard LEGO Robotic Command Explorer (RCX) – a small computer built around a Hitachi H8 controller – to control a deliberative robotic architecture that performs mapping and path planning to achieve the goal of gathering targets in an arena. To test the validity of its use as an educational tool, students in an introductory artificial intelligence class were given a project requiring them to build a deliberative robotic architecture using only commercially available LEGO components.

The LEGO Mindstorms kits offer a lot of advantages over other inexpensive platforms. First, they are one of the cheapest platforms available. The Mindstorms kit, complete with programmable computer, building components, sensors, and motors, costs $200. The second advantage to the LEGO Mindstorms kit is that it requires no special knowledge for construction. LEGO pieces, sensors, and motors are standardized to snap together, making connecting pieces very easy. Lastly, almost all students are familiar with LEGO bricks. Thus, building robots from LEGO bricks is much less intimidating for students building their first robots. They are much more likely to explore different configurations. For this reason, the project focuses on using only commercially available components.

Lectures

Students are provided an introduction to robotic control as a part of the introductory artificial intelligence course. When teaching the students about a deliberative robotic paradigm, goal-directed actions and creating plans to meet those goals is emphasized. To help the students understand the process of how the robot meets its goals, the "Sense-Plan-Act" organization of the architecture is discussed in detail (see Murphy [2]). Sensory input comes from either direct sensor input or *a priori* information. The robot then plans an action with what knowledge it has. Finally, it acts on the plan. Then, the process repeats until the goal is met. If the plan fails, the robot re-plans.

The explanation of the process then leads to a discussion of the need for a world model and the use of a knowledge representation language that can represent that model. The robot requires a language it can use to store information and later retrieve and process the information for planning purposes. To help students better understand what the robot does with the information it receives, there is a discussion on short term and long term memory.

Next, the students are presented with the strengths and weaknesses of deliberative robotic control. The benefits include the ability to solve problems requiring cognitive skills, the ability to generate an optimal solution, and predictability. The deliberative robotic control's biggest drawback is explained to be its dependence upon the world model. This includes requiring a closed world model, the symbol grounding problem,

the frame problem, the qualification problem, and localization.

Once students are familiar with what a deliberative robotic control does, they are introduced to methodologies of design. Specifically, planning algorithms. Tree searches are discussed in detail earlier in the course. They are reviewed again and the students are then introduced to the wavefront algorithm (see Murphy [2]) with the intent of having them implement it on the robot during the laboratory assignments. The wavefront algorithm was chosen because of its small memory requirement. Its biggest drawback is the amount of time it takes to propagate the wave on the RCX. The wavefront algorithm is typically used with a grid-based world representation (see Figure 1).

0	0	0	0	0	**2**	0	0
0	0	0	**1**	0	0	0	0
0	0	**1**	**1**	**1**	**1**	0	0
0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	0

Figure 1 – World Representation (pre-propagation)
[Key: 2 = Goal, 1 = Obstacle, 0 = Non-obstacle]

Only one goal can be evaluated at a time. So, if there are multiple goals, multiple waves must be propagated. When the wavefront algorithm begins, the world space with the goal of interest is designated with a value of 2. World spaces with obstacles are designated with a value of 1. All other locations are given a value of 0. To propagate the wave, the algorithm must check each grid space. If it has a value of 2 or greater, it then assigns the non-obstacle spaces around it with a value one greater than the current space. This continues until all reachable, non-obstacle spaces are assigned a new value (see Figure 2).

It is important to define how the robot can move about its world so that the wavefront can correctly determine the shortest path. In Figure 2, you should note that only grids horizontally and vertically adjacent to a grid space are incremented. This implies that the robot does not have the capability to travel diagonally from grid space to grid space.

7	6	5	4	3	**2**	3	4
8	7	6	**1**	4	3	4	5
9	8	**1**	**1**	**1**	**1**	5	6
10	9	10	11	10	**1**	6	7
11	10	11	10	9	8	7	8

Figure 2 – World Representation (post-propagation)
[Key: 2 = Goal, 1 = Obstacle]

Once the wave propagation is complete, the grid location of the robot's current position is examined. That grid space's value, minus 2, is the number of moves that the robot is away from the goal. Following the values down from the current location value to the goal value of 2 generates a path.

The next topic in the introduction to deliberative robotic control is localization. Both landmarking and deductive reckoning are discussed. Students are made aware of how to use landmarking to predict where the robot is in the world without being given a start location. They are also informed of the requirement to provide a start location and orientation when using deductive reckoning. The discussion also stresses the difficulty of maintaining accuracy by solely using deductive reckoning. Additionally, students are told about physical errors caused by things like wheel slippage and an imbalanced physical design. They are cautioned that these errors are typically immeasurable with an internal localization method like deductive reckoning.

While deductive reckoning has many drawbacks, the intent to use only off-the-shelf components limits our projects to this approach. The LEGO Mindstorms main external sensor is the light sensor. It does not contain enough granularity in readings or enough detection range to provide an easy methodology to incorporate landmarking.

Finally, students are walked through the creation of a deliberative robotic control architecture.

Specifically, Murphy's [2] processes of a cartographer, a planner, a navigator, a sensor monitor, and a pilot were discussed as a feasible functionality software breakdown.

Diane, the deliberative LEGO robot created for the project, was used to demonstrate a working model. The robot was programmed with known goals, obstacles, and a start location and orientation. As Diane progressed toward a goal, an obstacle was placed in the way causing the plan to fail, the map to be updated, and another plan to be generated and executed for a successful completion (see Figure 3 for sample arena layout).

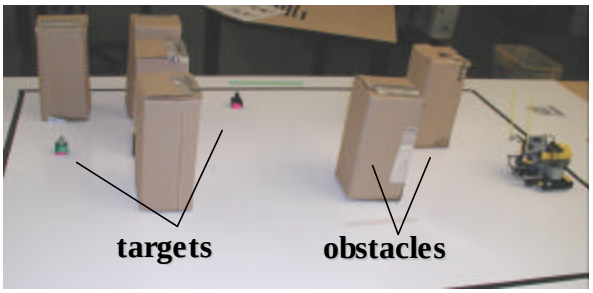

Figure 3 – Sample Arena Layout

Diane's Hardware Construction

Diane's current hardware configuration (see Figure 4) is a result of analysis of the strengths and weaknesses of five previous hardware architectures designed by Mayer [3]. This design was discussed in depth with the students to provide them knowledge of how the hardware design can impact the robot's functionality. It also provided the students with a real feasible solution.

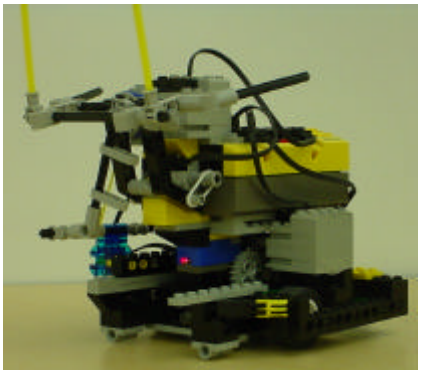

Figure 4 – Diane's Current Architecture

To successfully complete the tasks assigned, Diane required two rotation sensors for deductive reckoning, a light sensor for detecting targets, and a touch sensor for monitoring impacts with obstacles. Since the RCX only provides three input ports, the touch sensor was multiplexed with the light sensor. This is possible because when the touch sensor is closed, the RCX (reading the port signal as a light source) indicates a value equivalent to the brightest light source. It is very rare for the light sensor to provide a signal this bright even with the brightest of sources. The only time a conflict arises is when the RCX converts the values from raw form to an estimated value percentage. Physically keeping the light source from getting too close to the sensor is the easiest way to prevent the actual light value from reaching the touch sensor value. In addition to the sensors, Diane uses three standard LEGO motors. One motor is for linear motion, the other is for turning, and the third manipulates a target trap.

One of the hardest parts of designing a deliberative robotic architecture using deductive reckoning is ensuring that the robot travels a straight path and makes accurate turns. To facilitate the success of these two things, Diane was built around a dual-differential gear system (see Figure 5).

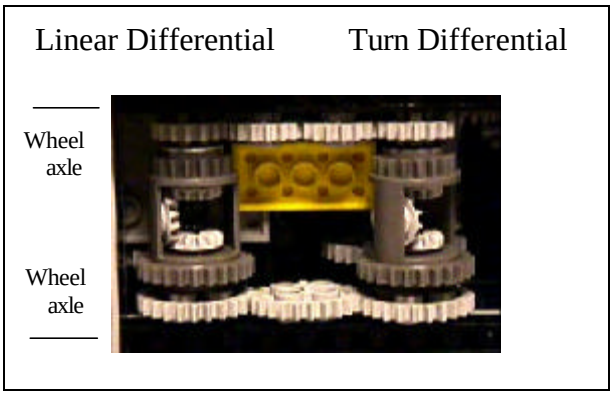

Figure 5 – Dual-Differential

The dual-differential system has many advantages. First, only one wheel axle is required. The same axle is used for both linear and turning motions. The robot is designed to rest on a rear skid plate for balance. With this axle mounted at

the center of the robot, the robot can essentially turn in place, removing turning radius calculations from movement equations. The second advantage is ensured equal speed at both wheels at all times. To obtain both linear and turning motions, two motors are required – one connected to each differential shell. By powering one motor and braking[1] the other (ensuring its connected differential shell doesn't move), the wheels are forced to move at the same speed. Mathematically, this can be seen by understanding that the angular velocity (speed and direction) of the differential shell is equal to the average angular velocity of the two axles connected to it.

$$n_{shell} = \frac{n_{axle1} + n_{axle2}}{2} \qquad \text{(Eq. 1)}$$

Re-examine Figure 5 and note how the differential shells are geared together. In particular, notice that there are an odd number of gears along one side and an even number of gears on the other. The effect is that when one differential shell is turning, all axles are turning with the same angular velocity but the axles on the other differential are turning in opposite directions. Since the gears are all of the same size between the differentials, the speed of axle rotation on the second differential is equal to the speed of the first differential. Since the angular velocities of the axles on the second differential are of equal speed but opposite direction, by equation 1 we can determine that the second differential shell has a velocity of zero. The opposite motion of the two axles translates through the second differential shell via the beveled gears inside the shell.

By rotating one differential shell and keeping the other stationary – by braking the motor connected to the second differential – the speed of the two wheels remain equal, regardless whether they travel in the same or opposite directions. If the second differential shell were to turn, then the speed of each wheel would have to change. An additional advantage is the design's flexibility. A turn with a non-zero turning radius can be executed by turning on both motors at once. The exact radius of the turn is determined by the difference in speeds. This third advantage gives great flexibility in execution with the same design.

Two encoders are required to properly conduct deductive reckoning. One is directly attached to the output shaft of the linear motor and the second is directly attached to the turning motor. Thus, both have a 1:1 ratio in angular velocity with their respective motors. LEGO rotation sensors are capable of registering 16 ticks per rotation of the rotation sensor internal axle. This yields 22.5 degrees per tick, which is equivalent to approximately 6 mm on Diane's 30.4 mm diameter wheel. This is the amount of error that can be incurred each time Diane starts and stops. To improve the granularity between rotation sensor ticks and actual wheel rotations, the rotation sensor can be geared up (increase its angular velocity in comparison to the motor) or the wheels can be geared down from the motor. The problem with the former is that the rotation sensors have a limited angular velocity before they start loosing accurate count. Per Ferrari and Ferrari [4], this limit is somewhere around the speed of the motor. So, to maintain accurate count, Diane's wheel axle is geared down from the motor while maintaining a 1:1 ratio between rotation sensor and motor. The motor to wheel ratio is 1:6, resulting in 96 rotation sensor ticks per wheel revolution. This provides a more acceptable accuracy of 1 mm per tick.

While gearing is important for accuracy when using the LEGO rotation sensors, the drawbacks and other benefits of Diane's design should also be noted. First, given the 1:6 ratio, the wheels are traveling at $1/6^{th}$ of the motor speed. Approximating that the LEGO motors turn at about 250 rpm when loaded [4], Diane's wheels are only turning at about 40 rpm – about 40 cm per minute. However, torque through a gear train is inversely proportional to speed. Thus, Diane has 6 times the torque at her wheels than her motor, enabling her to push a battery-laden target and roll over small bits of dirt and debris in her environment. Also, Diane's gear train involves 5 gears between each motor and wheel axle. This requires the need to account for gear slop – the

[1] Note that the Handy Board does not support braking with a motor. It is left free floating when set to off.

amount of rotation you can achieve in your input gear without moving your output gear. Diane's deliberative architecture tracks which way each motor last moved and adds a correction if it is opposite the current direction.

Physical construction also plays an important role. When using only internal references for navigation, it is extremely important that all immeasurable errors be reduced as much as possible. The physical design should be balanced. If one side is more heavily weighted than the other, then the robot tends to drift toward the heavier side. This drift is typically very hard to calculate and even harder to compensate for accurately. Similarly, it is best if the wheel axles are supported on both sides of the wheel. If unsupported, the weight of the RCX alone causes the ends of the axles to bow, leading to further immeasurable errors. Finally, the wheel axle should be somewhat centered on the robot. The more off-balance the load on the two sides of the gear axle, the harder the motors have to work to turn. This usually leads to immeasurable wheel slippage and reduced battery life.

Diane's Software Construction

Diane is programmed using Interactive C version 4.2 (IC4). IC4 was chosen over Not-Quite-C (NQC) because IC4 allows more room for program code by replacing the standard LEGO firmware. With extra memory, a single variable can be used for representation of each 6 x 8 grid space. If NQC is used, bitwise encoding of the grid is required due to decreased memory availability. While not impossible, it is felt that the added complexity of managing a bitwise representation may detract from the core lessons.

Diane's programming begins with initialization functions that note the average ambient and target light levels. She also receives *a priori* user input such as known obstacle, goal, and start location and heading. After initialization, Diane runs two concurrent processes. A retrieve process constantly makes and executes plans and a monitor process watches for plan failure. Diane keeps track of world information in a multidimensional array, using one variable per grid space. She manages the vertices, the grids themselves, vice the edges between them. Thus, an obstacle or target occupies an entire grid space.

Diane begins by determining which known target is closest. She then makes a plan from the start location to the nearest target. If an obstacle is encountered, Diane adds the obstacle to the world map and re-plans. She determines the new closest goal and plans a path there. If a new target is encountered, Diane adds the new target to the world map and re-plans from her current location. Once a target is captured, she makes a plan to return to the start location to release the target. If an obstacle is encountered while returning to the start location, Diane reacts similarly to an obstacle encounter when retrieving a target. A drawback of Diane's design is that when a target is in the trap, she has no way of detecting new targets as the captured target blocks the target-detecting light sensor.

Lab Exercise

To test the applicability of this educational approach to deliberative robotics, 21 students – in teams of 2 – were assigned a deliberative robotic task of finding targets. IC4 was the required programming environment. To try and emphasize the necessity of a good physical design and its interface with the software, students were required to build their robotic platforms from scratch (though they were encouraged to use design ideas from Diane). To simplify their task, the world was modified to use a black tape silhouette around a grid as an obstacle and a green silhouette as a target. Since there were no physical targets, the students did not have the concern of designing adequate target traps or planning return trips. The lack of a return trip requirement also reduced localization errors. If the student's robot encountered an unknown obstacle, it was required to go back to the last known empty grid space, modify its world representation, and replan. If it encountered an unknown target space, it was required to audibly acknowledge the new target with a series of beeps and could then proceed with its original plan through the new target grid space.

Students were graded on logic and coding, physical construction, actually finding the goals and avoiding the obstacles, and an overall design

integration category. They were provided with the wavefront algorithm and had two weeks to complete the project. Further, students were told that they were allowed 3 "nudges" during a run to assist with localization should the robot begin to go awry.

The results were very positive. The students showed understanding through both successful execution and comprehension of the causes of failure. All students used the wavefront algorithm for path planning. One team mentioned attempting to use A* but found the memory requirements too restrictive even with IC4. Four of the teams made it a personal goal to use as few nudges as possible. One of these teams used 0; the others used 1 each. Eight of 11 teams found the target and avoided all obstacles. While not all teams were successful, a discussion was held with each team after their run to provide feedback on the positive and negative aspects of their design and logic. At the end of the laboratory, a feedback form was handed out.

Nineteen of 21 feedback forms were received. While many students seemed to grasp the difficulty of developing a mobile robot with a good hardware/software interface and the ability to localize, a few negative comments were received regarding the annoyance of sensor limitations and amount of emphasis on hardware systems and the related physics. The material may need to be modified to ensure that the importance of the role that the physical world plays on robotics systems is stressed and understood by all of the students. The other major comments received regarded the amount of time required to build the robot.

The deliberative robotic assignment was provided to the students after a reactive robotic assignment was complete. Thus, each team built two separate robots that often had large variations in design. In future instruction it may be best to provide the deliberative lectures first, have the teams build one robot to the more restrictive deliberative hardware requirements required for proper localization, and, in a later assignment, incorporate reactive functionality to make a single hybrid robot.

Overall, the project proves that deliberative robotic architectures can be taught successfully using commercially available LEGO components. All teams showed comprehension of the required task. Even those teams that did not successfully complete the assignment still made significant progress toward the end goal. While student feedback reveals areas to be improved upon, the majority felt that they had learned something and enjoyed doing it. More specific details about the project can be found in Mayer's thesis [3], a copy of which can be obtained by sending e-mail to the address at the top.

Works Cited

[1] Weinberg, Jerry B., and Xudong Yu. "Robotics in Education: Low-Cost Platforms for Teaching Integrated Systems." IEEE Robotics & Automation Magazine. vol. 10, no. 2. June 2003: 4 – 6.

[2] Murphy, Robin. Introduction to AI Robotics. Cambridge: MIT Press, 2000.

[3] Mayer, Gary. "Implementation of a Deliberative and Reactive Robot Control Architecture on an Inexpensive Platform." Thesis Southern Illinois University at Edwardsville, 2003.

[4] Ferrari, Mario, and Giulio Ferrari. Building Robots with LEGO Mindstorms: The Ultimate Tool for Mindstorms Maniacs!. Ed. Ralph Hempel. Rockland: Syngress Publishing, Inc., 2002.