Computer Science Education
Vol. 00, No. 00, January 2008, 1–11

ORIGINAL ARTICLE

Introduction to Autonomous Robotics using Lego Mindstorms NXT

H. Levent Akın[a*], Çetin Meriçli[b], and Tekin Meriçli[a]

[a]*Department of Computer Engineering, Boğaziçi University, Istanbul, Turkey*; [b]*Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA*

(*v1.0 released May 2013*)

To thoroughly teach fundamentals of robotics to computer science undergraduates, a well-balanced curriculum should be designed and complemented with hands-on applications on a platform that allows rapid construction of complex robots, and implementation of sophisticated algorithms. This paper describes such a course where the Lego Mindstorms NXT kits are used as the robot platform. The aims, scope, and contents of the course are presented and the design of the laboratory sessions as well as the term projects, which address more than one core problem of robotics and artificial intelligence simultaneously, are explained in detail.

Keywords: Undergraduate robotics education, LEGO Mindstorms NXT, Using robots in Computer Science education

1 Introduction

Teaching robotics to computer science undergraduate students is a challenging problem as they lack a good grasp of the uncertainty notion associated with every physical part of a typical robotic system. Most of them also lack the experience and knowledge in electronics and mechanics to build a robot from scratch. Therefore, to efficiently teach robotics to such an audience, a carefully crafted curriculum should be complemented with a platform that allows rapid construction and task specific modification of the hardware, and implementation of sophisticated algorithms.

Lego Mindstorms NXT kit satisfies all these requirements, and the limited processing capability of the platform forces the computer science students to put forward their solid skills on efficient software development. The use of Lego Mindstorms for teaching basic robotics or as an educational aiding tool is common in K-12 level curricula (Mataric, 2004). On the other hand, it is mostly limited to teaching common computer science and programming concepts at the undergraduate level (Lew et al., 2010). We designed a robotics course that utilizes Lego Mindstorms NXT in an attempt to teach fairly advanced fundamental concepts of robotics without using expensive hardware platforms featuring abundant resources.

This paper describes the goals, design choices, course contents, and student evaluation methods of an introductory undergraduate robotics course offered in the Department of Computer Engineering at Boğaziçi University, Turkey. The course consists of weekly lectures explaining the theoretical aspects such as navigation, path planning, sensor and actuator calibration, probabilistic inference, and control theory. Each lecture is followed by a laboratory session where the covered topics are reinforced through hands-on application. A month-long term project structured as a competition is assigned to challenge the students to work as teams to design an end-to-end system in which they properly engineer the integration of the scientific methods learned during the lectures.

*Corresponding author. Email: akin@boun.edu.tr

ISSN: 0899-3408 print/ISSN 1744-5175 online

DOI: 10.1080/0899340YYxxxxxxxx
http://www.informaworld.com

2 Course Description

A good undergraduate level introductory robotics course should be designed to include as many topics as possible to form a solid basis for building and programming robots, yet it should be enjoyable and easy to follow for the students. With these goals in mind, we decided to cover the following topics that we think constitute the fundamentals of mobile robotics:

- Uncertainty in sensing and action, and methods for dealing with it
- Self localization
- Path planning and obstacle avoidance
- Behavior based and reactive controller architectures
- Control theory
- Mapping
- Advanced topics such as multi-robot systems, learning, and social robotics

Uncertainty is one of the major issues of robotics; however, it is not very well understood by the computer science undergraduates as they are not accustomed to interacting with the physical world and the noise that is associated with almost every physical entity. Therefore, we resolved to particularly emphasizing the uncertainty notion and the methods for handling it, including sensor and actuator calibration, filtering, and probabilistic inference algorithms. The covered topics are supported with an optional textbook (Mataric, 2007) and a set of reference books on robotics (Dudek & Jenkin, 2000; Martin, 2001; Murphy, 2000), and AI (Russell & Norvig, 2003).

No matter how well the topics are explained in the lectures, we believe that hands-on experience is essential in robotics education to help internalize the learned concepts. This opinion motivated us to structure the course as one hour of lecturing followed by two hours of laboratory sessions where the students apply the theory covered in the preceding lecture and immediately see the outcomes on the robot. To be able to achieve such a rapid development and experimentation process, we needed a hardware platform that is flexible enough to allow constructing easily programmable robots with a wide spectrum of features.

We decided that the *Lego Mindstorms NXT* robotic kit (http://mindstorms.lego.com/) was particularly suitable to be used as the hardware platform of the course since it enabled creating sophisticated robots in a relatively short amount of time and without requiring in-depth technical knowledge about either electronics or mechanics. Furthermore, it is possible to program the *Lego Brick*, which is the processing unit of the system, in Java using the *LeJOS* open-source firmware (http://lejos.sourceforge.net/). This capability made it even smoother for the students to begin programming the robots almost instantly as Java is taught as one of the primary programming languages in our curriculum. In addition to the reference books used for the lectures, we also used several resources for the Lego Mindstorms kits (Ferrari, 2002; Bagnall, 2007; Gasperi et al., 2007).

In addition to the laboratory sessions focused on isolated applications of the individual topics, we also designed term projects that would require integration and use of all the key concepts learned throughout the semester. The term project is assigned after the essential topics are covered and the students continue working on the project during the laboratory hours while the advanced topics without laboratory applications are being covered. The projects are designed in such a way to provide a competition environment that is based on a story and a set of rules to promote creative and smart solutions.

3 Course Syllabus

In this section, we elaborate on the contents of the weekly lectures and the laboratory sessions. The course material can be found at *http://robot.cmpe.boun.edu.tr/~cmpe434*.

3.1 *Week #1: Introduction*

The goal of the first week is to get the students acquainted with the general terms and concepts in mobile robotics and artificial intelligence (AI), such as definitions of intelligence, the "agent" concept, environment and agent types, brief history of robotics, the behavior concept, feedback control concept, manipulators vs. mobile robots, sensor and actuator concepts, sources of uncertainty, and three questions of mobile robotics.

In the first lab session, the students are asked to form teams of two, and each team is provided with the Lego Mindstorms NXT kits. They first build the *TaskBot* (Figure 1) from *CMU Robotics Academy* (Carnegie Mellon Robotics Academy, 2007), and learn how to use the Lego software for constructing simple programs by dragging and dropping basic programming blocks. They program the *TaskBot* to go forward or backward and change direction with the clapping sound. Next, they enhance the robot's sensing and actuation capabilities by the addition of a bumper and rotation movements. In its final configuration, the robot can wander around and react to collisions in addition to the clapping sound.

Figure 1. The *TaskBot* differential drive robot.

3.2 *Week #2: Programming Lego NXT with Java*

The goal of the second week is to introduce the *LeJOS* Java programming environment. In the lecture, a brief overview of the *LeJOS* environment and the API for accessing the sensors and actuators as well as programming behaviors are given, and the limitations of the *LeJOS* framework are discussed.

In this lab session, the students are first asked to update the firmware of their Lego Bricks. Following this update, they test various sample programs that come with the *LeJOS* environment, such as basic sensor and actuator access, simple line following behavior, and a controller for differential drive steering. Finally, the behaviors programmed using the Lego software in the previous laboratory session are re-implemented in Java.

3.3 *Week #3: Sensors and Actuators*

As opposed to the classical computer science understanding of I/O systems, sensors and actuators have uncertainty associated with them that require additional procedures to make them reliable. In this lecture, topics including different sensing and actuation modalities, the notion of uncertainty in sensing and action, sensor and actuator calibration, the difference between sensing and perception, active and passive sensing, sensor fusion, the state space concept, principles of motor operation, different types of motors, torque-power relation, servo motor basics, battery types, power regulation, and the sense-think-act paradigm are covered.

In order to demonstrate that the sensors and actuators do not behave exactly the way that they are expected to due to uncertainty, in this lab session, the students are asked to repeatedly measure the quantities reported by the light, ultrasound, sound, and touch sensors, and plot the

result to see if there is a deviation from the expected linear relationship. They are also asked to establish a relation between the motor speeds applied and the observed linear and rotational motion of the robot.

3.4 *Week #4: Locomotion*

There are numerous ways of making a robot move around, each having different advantages and disadvantages. In this lecture, we briefly touch upon the most popular locomotion architectures, and cover topics such as the effector and degree of freedom concepts, joints and links, holonomic, non-holonomic, and redundant systems, pros and cons of different locomotion systems, static and dynamic stability concepts, common gait types for legged locomotion, motion planning, forward and inverse kinematics, odometry and dead reckoning, detailed analysis of differential drive systems, Ackermann steering, synchronous drive, and omni-directional drive systems.

As a continuation from the previous lab session, in this session, odometry calibration of the *TaskBot* is performed by the application of *UMBmark* odometry calibration procedure (Borenstein et al., 1996). The students report the observed systematic and non-systematic errors as well as the level of improvement before and after the calibration. The robot is programmed to follow a 4m $\times$ 4m square trajectory several times both in clockwise and counterclockwise directions, and a motion model that compensates for the odometry errors is constructed based on the observed deviations from the desired path.

3.5 *Week #5: Control*

In order to make the robot perform the way we want it to, we need to *control* it in such a way to minimize the discrepancy between the desired and the observed behavior. In this lecture, we make an introduction to the control theory by briefly explaining topics like open and closed loop control concepts, step response concept, types of feedback control systems, linear and non-linear control, adaptive control, detailed coverage of PID control, and tuning PID controllers with Ziegler-Nichols method.

In this lab session, the students are asked to build a two wheeled self-balancing robot. It uses a light sensor pointing downwards as its feedback mechanism which provides information about the tilt of the robot inferred using the sensed light intensity. An example construction from one of the groups is shown in Figure 2. After constructing the robot, the students try to find appropriate coefficients for a complete PID controller code to keep the robot balanced for as long as possible. The bonus part of this lab session is to also make the robot move forward or backward, and perform turning motions in a controlled manner.

Figure 2. An example self-balancing robot construction.

3.6 *Week #6: Reactive Architectures*

A mobile robot does not always have to be very intelligent in order to exhibit sophisticated behaviors. In this lecture, we present a generic view to robot control architectures, a brief description of deliberative architectures, a more detailed view into the sense-plan-act paradigm

and its problems, principles of reactive control, Braitenberg vehicles, hybrid control, three-layer architectures, and pros and cons of hybrid approaches.

The aim of this lab session is to demonstrate to the students that complex behaviors can be obtained by using very primitive sensory-motor couplings. The students are asked to implement the first four *vehicles* from Valentino Braitenberg's book "Vehicles" (Braitenberg, 1986), and a vehicle of their own design.

3.7 *Week #7: Behavior Based Architectures*

Continuing from the previous week's lecture, we present the definition of behavior, emergent and reactive behavior concepts, behaviors in animals, behavior vs. action, representation of behaviors, properties of behaviors, behavior notations (stimulus-response diagrams, functional notation, and finite-state acceptors), assembling behaviors, evaluating behavior based architectures, and the Subsumption Architecture (Brooks, 1986).

In this lab session, the students first build a robot with an ultrasonic sensor on a turning turret to be used for obstacle detection. Then they are asked to develop a behavior scheme using the Subsumption Architecture for seeking a light source and making the robot go towards it while avoiding obstacles in the environment.

3.8 *Week #8: Localization*

Knowing its whereabouts in the world is crucial for a mobile robot in order to perform goal-directed tasks. However, the uncertainty in sensing and actuation imposes several difficulties with inferring the robot's pose accurately. In this lecture, we briefly explain typical problems in localization, the difference between active and passive localization, classification of localization algorithms, triangulation, probabilistic localization methods, Markov localization, Kalman localization, and Monte Carlo and Reverse Monte Carlo localization algorithms.

In the lab session the students develop observation and motion models for a Monte-Carlo Localization implementation.The students use a wireless monitoring tool to observe the performance of the developed self localization system. As the robot moves along a line, it continuously checks the side-facing ultrasonic sensor readings in addition to computing its displacement since the last odometry reading to update the belief values of individual hypotheses on its possible locations (i.e. particles). A sample screenshot from the remote monitoring tool is shown in Figure 3. In the figure, the blue lines denote the map, individual hypotheses on the location of the robot are represented as thin red lines, and the inferred position is represented with a green line.

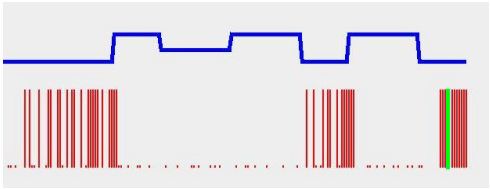

Figure 3. A snapshot from the visualization tool used for testing the localization algorithm.

3.9 *Week #9: Path Planning*

Knowing its own position and where to go do not provide complete information to the robot without a way of connecting these two points in a safe and efficient manner. Depending on the application, different metrics, such as the path length and the average distance from the obstacles, can be used for deciding how to connect the start and end positions. In this week's lecture, we present definitions of the path planning problem, configuration space, complexity of

path planning, bug algorithms, visibility graphs, generalized Voronoi graphs, grid based methods and cell decomposition, potential fields, sampling based methods, probabilistic roadmaps (PRM), and rapidly exploring random trees (RRT).

In this lab session, the students implement the *Bug1* and *Bug2* algorithms (Lumelsky & Stepanov, 1987) for a robot that tries to reach a light source while avoiding the obstacles on its way. The light source is visible to the robot from anywhere in the environment; however, the direct path is obstructed. The robot runs the *Bug* algorithms to follow the walls of the obstacle until its path intersects with the line that connects the starting point and the light source.

3.10 *Week #10: Mapping*

In situations where a map of the environment to be used for self localization purposes is not available, the robot has to build one by itself. In this lecture, after briefly discussing why mapping is needed, we present environment modeling and representation methods, problems in mapping, continuous representation, single and multi-hypothesis approaches, exact, fixed, and adaptive cell decomposition methods, occupancy grids, topological decomposition, simultaneous localization and mapping (SLAM) problem, and applications of SLAM.

In this lab session, the students are asked to build a robot base with an ultrasonic distance sensor. The robot is programmed to traverse a square path of 1 m $\times$ 1 m with its distance sensor facing outwards. Various objects are placed around the square trajectory, and the students are asked to develop a method for building a map of the environment using the estimated position of the robot on the square trajectory, and the distance reported by the ultrasonic sensor at that position.

3.11 *Week #11: Learning*

Equipping robots with the ability to improve their performance through learning not only saves human developers from programming every detail, but may also be necessary for problems without a known algorithmic solution. We start this lecture by giving the definitions of learning and machine learning. We then cover topics such as supervised learning, unsupervised learning, reinforcement learning (RL) systems, reward and value functions for RL systems, exploration-exploitation dilemma, Markov Decision Processes (MDPs), Markov property, policy, reward, and value functions for MDPs, policy evaluation, Monte Carlo sampling, temporal difference (TD) learning, Q-Learning, policy gradient learning, Poincare map based RL, and example applications of learning.

3.12 *Week #12: Multirobot Systems*

Using multiple robots to perform a task has two potential advantages. First, the overall execution performance can be improved if the task at hand can be divided into proper subtasks that can be performed by different robots. Second, with a proper design, the introduced redundancy can make the system more robust against possible failures. In this lecture, we cover topics including the terminology in multirobot systems, classification of multirobot systems according to interaction, biological inspirations, motion coordination, communication in multirobot systems, cooperative object transport and manipulation, reconfigurable robotics, multirobot localization, mapping and exploration, multirobot learning, swarming/flocking/schooling, advantages and disadvantages of multirobot systems, types of collective systems and multirobot domains, competitive domains, control approaches, centralized vs. distributed control, reactive, behavior based, intentional, and hybrid approaches, taxonomies for multirobot systems, and market based task allocation methods with robot soccer as an example application domain.

3.13 *Week #13: Social Robotics*

Depending on the application, robots may need to cohabitate with humans in certain environments, such as factories (as co-workers), hospitals (as nurses), and houses (as servants). In that case it becomes necessary for robots to be able to form social bonds with humans through various modalities of communication and get socially accepted by them. This lecture covers the definition of social robots, social robot classes and forms, common design problems of socially interactive robots, interaction modalities, artificial emotions, user models, robot social learning, imitation, and examples of potential applications.

4 Term Projects

To solidify the hands-on experience gained in laboratory sessions, a relatively complex term project with a theme is assigned to the students. The term project includes the simplified versions of the main problems in the field of robotics and AI, some of which are inference, path planning, map making, perception, and navigation. The remainder of this section describes in detail the five term project themes that have been assigned since 2007.

4.1 *Wumpus World*

In this project, the students are asked to implement a simplified and modified version of the famous *Wumpus World* problem (Russell & Norvig, 2003). The simplified *Wumpus World* is a 5×5 grid of squares surrounded by walls. The grid cells may be empty, or may contain a bottomless pit, a stinky beast called **Wumpus**, or a chest of gold. The goal of the robot is to find the gold and return to the starting position without falling into a pit or being eaten by the Wumpus. The Wumpus creates a stench percept, and the pit creates a breeze in the adjacent cells while the gold creates a glitter percept in its own cell. The stench, breeze, and glitter percepts are represented with different colors and sensed via a light sensor, whilst the bump percept is sensed through a bump sensor. A custom made simulator written in Java is provided to isolate the development of the AI part from the mechanical design and to keep the students concentrated on the strategy part first. In parallel to the software development, the students were asked to develop the necessary low level functionality required to execute the sensing and action functions that the simulated agent uses. The selection of Java language for simulator development enabled the students to run their agent software on the real robot without any modifications. An example *Wumpus World* setup in the simulator is shown in Figure 4.

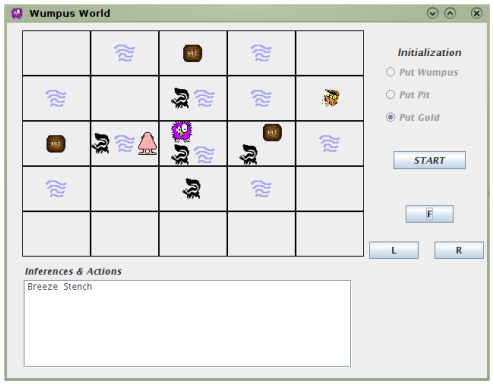

Figure 4. A snapshot from the simulator for the Wumpus World problem.

4.2 *Rescue the Princess*

In this project, the students are asked to design a convertible robot that can both navigate a maze, and climb a pole. The goal is to find the kidnapped "princess" being kept in an underground dungeon, and help her out by climbing a pole. The dungeon world is implemented as a rectangular area consisting of square cells. The area is surrounded by walls and each square can either be empty or contain only one type of object. The valid object set consists of: i) an obstacle (no specific color information), the pole (in yellow), and the princess (in blue). The students could use extra sensors but they were limited to using three motors and only one Lego brick. The initial position of the robot was determined randomly and was the same for all student groups. Figure 5 shows one design by the students that could easily be modified into a pole climber.

Figure 5. An example robot design for the Rescue the Princess project. The robot used for navigating and searching for the princess on the floor (left) could easily be modified into a pole climber after finding the princess (right).

4.3 *Mouse in the Maze*

The objective of this project is to build a robot, namely a "Micromouse", which can negotiate a specified maze in the shortest time. Micromouse is an engineering design competition created by IEEE where small robotic mice solve a 16×16 maze. The mice are completely autonomous robots that must find their way from a predetermined starting position to the central area of the maze unaided. A micro mouse needs to keep track of where it is, discover walls as it explores, build a map of the maze, and detect when it reaches the goal. Having reached the goal, the mouse will typically perform additional searches of the maze until it finds an optimal route from the start to the center. Once the optimal route is found, the mouse will run that route in the shortest possible time.

We downscaled the original maze to 8×8 squares of size 33 cm$\times$33 cm each for simplification purposes and space constraints. At the center of the maze is a large opening that is composed of 4 unit squares. This central square is the destination and is marked with a different color. A snapshot from the maze setup is given in Figure 6.

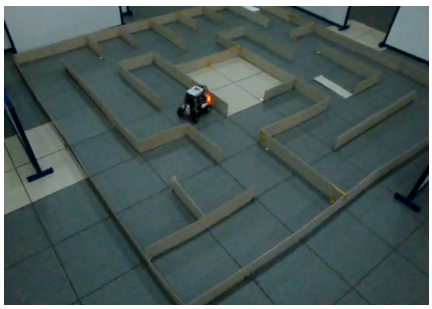

Figure 6. A snapshot from the downscaled "Micromouse" maze setup.

4.4 *Search and Rescue*

The objective of this project is to build a search and rescue robot to operate in a maze, symbolizing a post-disaster debris. The debris is modeled as a discrete world consisting of grids, and the victims with different conditions (trapped, wounded, or dead) are represented by different A5-sized color patches on the walls. The students are asked to develop a search and rescue robot to explore the maze and build a map of it on which the victims marked. The robot starts exploring the map from one of the four corners of the maze, and it should return to its starting position once the exploration is completed. The final map should be communicated over Bluetooth and displayed on a host PC.

4.5 *Treasure Hunter*

The goal of this project is to build a treasure hunter robot to operate in a room-based environment with 2 floors, constructed based on a RoboCup Junior Rescue B competition field (http://rcj.robocup.org/rcj2011/rescueB_2011.pdf) without the high contrast lines in the original setup for making navigation easier (Figure 7). The environment is modeled as a discrete world consisting of grid cells whose dimensions are 30×30 cm. The grid cells that contain treasures of different classes (gold, silver, or bronze), treasure boxes and traps are represented by different A5-sized color patches on the floors. A treasure box is used to store the treasures collected. When the robot moves into a trap, all treasures collected by the robot that have not yet put into the treasure box disappear. If the robot collects a treasure, the corresponding grid cell is reloaded with the same type of treasure after 3 minutes. The goal is to collect as much treasure as possible. The robot starts exploring the map from one of the four corners of the first floor. It should communicate its actions like collecting and dropping treasures through different sounds. Furthermore, a visualization software should be developed to display the final map and the collected treasures on a host PC connected over Bluetooth.

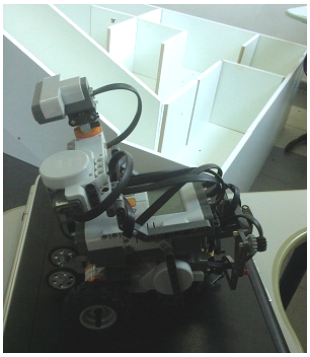

Figure 7. An example robot design for the Treasure Hunter project with the competition field in the background.

5 Evaluating Student Progress

We monitor the students' progress weekly through live demos and technical reports summarizing the laboratory work done in the previous week. We ask the students to address several issues in their reports such as the technical challenges faced, the methods used to overcome these difficulties, the design choices for both software and hardware, and the rationale behind these choices. Similarly, the term project is also evaluated through technical reports and a live final demo. After the assignment of the term project, the student teams first prepare a design document where they speculate about their design decisions along with the justifications of these decisions. At the end of the project, they also submit a final report explaining in detail everything they

have done, accompanied by a live demo. We also hold a competition during the final demos and award the teams with outstanding performance with bonus points. The theoretical part of the lecture is evaluated through a standard paper-based written exam.

For the first three years, we collected the weekly reports as well as the term project reports on paper. In 2010, we switched to an online feedback system where we utilized our wiki-based course web page for collecting and evaluating progress reports. Seeing the designed system in action play a very important role when evaluating robotic systems; therefore, the positive impact of heavily utilizing web technologies were huge, as the new feedback system enabled the students to easily embed videos they take outside the laboratory sessions. Another side benefit of this web-based feedback system is the accumulation of a significant amount of media (videos and pictures) over the semester for archival purposes.

6 Conclusions

In this paper, we present an introductory undergraduate robotics course that is being offered in the Department of Computer Engineering at Boğaziçi University since 2007. Our course started as a special topics course and as a result of the great interest and positive feedbacks of the students, it became a regular course in 2011. Also for the first time in 2011, we increased the student quota from 10 to 22 to meet the student demand.

Our curriculum and laboratory design proved itself to be successful as we consistently received very positive feedback from the students. As a consequence, the content of the course has almost stayed the same with the exception of introducing a new theme for the term projects each year. We also think that Lego Mindstorms is an excellent platform to be used in an introductory robotics course as it enables rapid development of fairly complex robotic systems, especially when combined with a widely popular language like Java.

Encouraged by the success of this course, we are planning to extend both laboratory sessions and the term projects to cover more advanced topics like multi-robot systems, vision based navigation and mapping, and learning, and write a textbook with very detailed descriptions of the laboratory sessions in addition to the theoretical material to make it easy to replicate for other instructors and institutions. Meanwhile, we would be glad to share our experiences with any interested parties.

References

Bagnall, B. (2007). *Maximum Lego NXT: Building Robots with Java Brains*. Variant Press.

Borenstein, J., Feng, L., & Borenstein, C.J. (1996). Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, *12*, 869–880.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. The MIT Press.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*(1), 14–23.

Carnegie Mellon Robotics Academy (2007). Http://www.education.rec.ri.cmu.edu.

Dudek, G., & Jenkin, M. (2000). *Computational Principles of Mobile Robotics*. Cambridge University Press.

Ferrari, M. (2002). *Building Robots with Lego Mindstorms*. Syngress Pub.

Gasperi, M., Hurbain, P.E., & Hurbain, I.L. (2007). *Extreme NXT: Extending the LEGO MINDSTORMS NXT to the Next Level*. Apress.

Lew, M.W., Horton, T.B., & Sherriff, M.S. (2010). Using LEGO MINDSTORMS NXT and LEJOS in an Advanced Software Engineering Course. In The 23rd Annual IEEE-CS Conference on Software Engineering Education and Training .

Lumelsky, V., & Stepanov, A. (1987). Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, *2*(1), 403–430.

Martin, F.G. (2001). *Robotic Explorations: A Hands-on Introduction to Engineering*. Prentice Hall.

Mataric, M. (2004). Robotics Education for All Ages. In AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education .

Mataric, M. (2007). *The Robotics Primer*. MIT Press.

Murphy, R.R. (2000). *An Introduction to AI Robotics*. MIT Press.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall.