A User Friendly Software Framework for Mobile Robot Control

Jesse Riddle, Ryan Hughes, Nathaniel Biefeld, and **Suranga Hettiarachchi**
Computer Science Department, Indiana University Southeast
New Albany, IN 47150

Abstract

We are interested in designing applications for autonomous mobile robots and robot swarms to accomplish tasks such as terrain analysis, search and rescue, and chemical plume source tracing. These tasks require robots to avoid obstacles and reach a goal. We use X80Pro mobile robots designed and developed by Dr.Robot Inc. for task applications. The vendor provided software framework with graphical user interface(GUI) allows robots only to be controlled remotely. The development of new robot control applications require developers to acquire in-depth knowledge of $MicrosoftActiveX$ controls and C# programming language. In this paper, we present a new software framework for X80Pro robots that will allow even a novice C++ programmer to design and implement autonomous mobile robot applications. We demonstrate the feasibility of our software framework using behavior-based and physics based control algorithms where a X80Pro robot avoid obstacles to reach a goal.

Introduction

It is important to have a stable software framework for developing applications for autonomous mobile robots, especially a software framework that allows quick prototype development, code implementation, and testing. A software framework with architectural deficiencies may increase development time and reduce the performance of robot applications. These issues become much more relevant when the developers are undergraduate research students with limited knowledge in multiple languages and complex software framework. Therefore, we are motivated to develop a user friendly software framework based on a commonly used programming language for mobile robot application development.

One of the major issues of the vendor provided software framework is the flexibility to modify it to provide sufficient user friendliness. Though these frameworks are user friendly for some tasks, they may not be user friendly for other applications. We are interested in designing applications for mobile robotic tasks such as terrain analysis, search and rescue, and chemical plume source tracing using physics based control algorithms. The vendor provided software framework for X80Pro robots provides insufficient flexibility and user friendliness required to developing applications for above tasks. Our effort is not to make superior software framework to vendor provided framework, but only to have a flexible software framework that suits our needs.

In this paper, we present our work with preliminary results from two robot control algorithms, behavior based and physics based. The algorithms make use of sonars, infrared sensors, and the camera of the X80Pro robot. The task of the robot is to navigate through a simple obstacle course and reach a light source as the goal. The rest of this paper provides an introduction to X80Pro robot, a description of the software framework, the two control algorithms, results and analysis of the robot behavior, a discussion on related work, and conclusion and future work.

X80Pro Robot

The X80Pro robot hardware framework developed by Dr.Robot is a off the shelf product for researchers that offers full WiFi (802.11g) wireless with dual serial communication channels, multimedia, sensing and motion capabilities (Dr. Robot 2012). The hardware framework contains two 12 volts motors, seven inch driving wheels, DC motor driver module with position and current feedback, six ultrasonic range sensors, seven Sharp infrared distance measuring sensors, one pyroelectric human motion sensor, color image camera, and audio modules (speaker and microphone), sufficient hardware components to serve in variety of applications. The Figure 1 shows the front and back views of the X80Pro robot with all the components.

X80Pro Software Framework

The software framework provided by the vendor for X80Pro robot is unique in that it depends on Win32 ActiveX Control. An ActiveX control is a reusable software component based on the Component Object Model (COM) that supports a wide variety of Object Linking and Embedding (OLE) functionality and can be customized to fit many software needs.

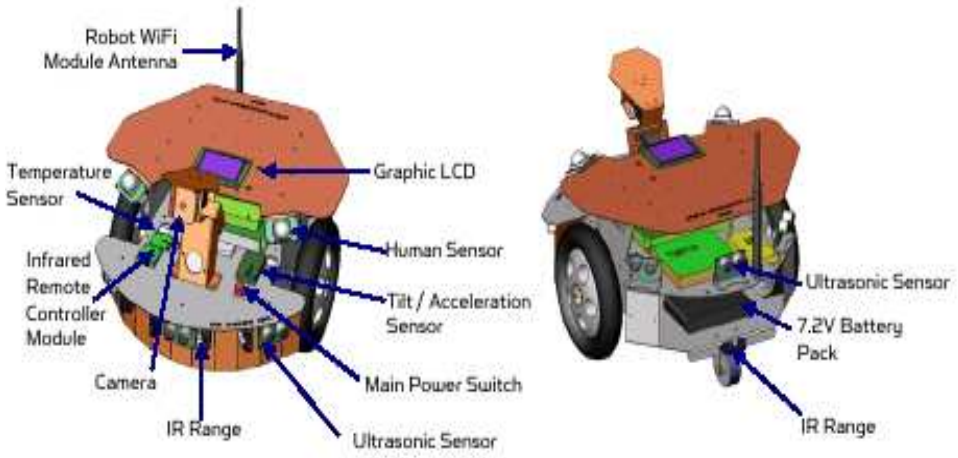

Figure 1: Front and back views of the X80Pro robot with the sensors mounted on the bottom.

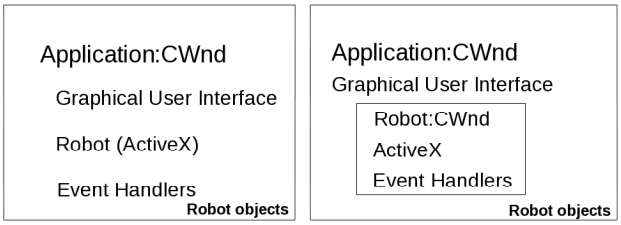

Figure 2: The vendor provided software framework(left) and user friendly software framework(right).

The developers can also create ActiveX controls with Microsoft Foundation Classes (MFC) (MFC ActiveX. 2005).

The complex software framework with multiple libraries implemented in C# treats the procedure calls as native. Behind the scenes though, when the ActiveX control is created in a C# application, the ActiveX component catches onto the application's message dispatcher automatically, without any extra special handling on the part of the application. In C# environment, the developer never have to work with the message dispatcher directly and can call the procedures as if they are native. However, when it comes to alternative programming languages, we realize that we could easily lose this simplicity. This is entirely because the vendor provided library is a framework dependent on Win32 ActiveX control, i.e. the application and the ActiveX representation builds an"is-a" relationship. Though the vendor's software framework provides sufficient development capabilities, one of the major inflexibility is the developed application's complete dependency on ActiveX controls making developers spend time trouble shooting software content possibly irrelevant to the task on hand. The left side box of the Figure 2 shows the high level architecture of the vendor provided software framework.

The right side box of the Figure 2 shows the high level architecture of the new user friendly software framework. In our new framework, the application keeps the event handlers contained in a class with an ActiveX representation for each robot by wrapping each ActiveX representation ("Robot:CWnd" in the inner box of the right box) into a Robot class, i.e. the application and the ActiveX representation builds a "has-a" relationship.

The simplest means to incorporate the provided library into an application in this framework is to first create either a Windows Forms Application using an MFC based application using C++. Next, using Microsoft Visual Studios form designer, developer should place the ActiveX control into a window container from the available toolbox. This causes the Visual Studio to automatically create a wrapper class to dispatch methods available in the ActiveX control. Finally the developer can assign methods to handle events coming from the ActiveX controls (ActiveX Events. 2005). We have also examined other possibilities, such as windowless activation of the control, with several container instances running asynchronously. However, without any special provision for multiple threads, the design of the framework becomes cumbersome, since we found that the event handlers are called asynchronously with up to 256 instances running in parallel.

Though the vendor supplied software framework seems simple, having all event handlers embed inside of the GUI design is a problem during application implementation and testing. We overcame this issue by compartmentalizing ActiveX controls with event handlers into an inner object independent of the GUI design. Though the implementation of our user friendly framework may be somewhat complicated, our main focus of robotic application development becomes much simpler with less time consumed on ActiveX trouble shooting.

Control Algorithms

We implemented two different control algorithms to test the feasibility of our user friendly software framework. The purpose of our experiments is to provide predictable results of control algorithms using our software framework. This allows us to evaluate the stability of our friendly software framework. The stability we refer here includes the accuracy of simple GUI for a control algorithm implemented with event handling to test sonar, infrared and motors.

Behavior Based X80Pro Control

The behavior-based approach is a methodology for designing autonomous agents and robots; it is a type of an *intelligent agent architecture*. Architectures supply structure and impose constraints on the way robot control problems are solved. The behavior-based methodology imposes a general biologically-inspired, distributed, bottom-up philosophy, allowing for a certain freedom of interpretation (Matarić 1999).

The behavior based algorithms demonstrates a variety of behaviors in a heuristic manner. Behavior-based and rule-based techniques do not make use of potential fields or forces. Instead, they deal directly with velocity vectors and heuristics for changing those vectors.

Physics Based X80Pro Control

In physics based approaches, virtual physics forces drive a agents and robots to a desired configuration or state. The desired configuration is one that minimizes overall system potential energy, and the system acts as a molecular dynamics ($\vec{F} = m\vec{a}$) simulation (Spears *et al*. 2005).

"Physicomimetics" or artificial physics (AP) is motivated by classical physics. This approach provides excellent techniques for distributed control of large collections of mobile physical agents as well as theoretical foundations for analyzing swarm behaviors. The Physicomimetics framework provides an effective basis for self-organization, fault-tolerance and self-repair of robot control (Spears *et al*. 2011).

Our control algorithm *AP-lite (i.e. artificial physics lite)* uses the physics based approaches with Hooke's law as our force law.

Experimental Methodology

Since we are at the initial stages of our research we decided to conduct all of the experiments with one robot though the control algorithms have the capability to scale to swarm of robots. Our robots are nonholonomic and they always move in the forward direction.

The robot environment is modeled with two parallel walls and four obstacles in between with sufficient space for the robot to navigate. The Figure 3 shows the robot view of this environment. The goal, a light source, is kept at the far end of the two walls and all lights in the lab were turned off during the experiments. We do not make use of filters to filter out the noise in the environment.

Our behavior based control algorithm uses two sonars (the left and right most sonars in the front of the robot) to model the robot behavior of moving from start location to a goal location where goal is a light source. The robot constantly move forward searching for the light source using the robot camera, and reacts to obstacles in the environment depending on the sonar readings by turning left or right by 35° angle (rule-based). The motors are powered with consistent rate of voltage, increasing or decreasing the power with the same consistency. For the clarity of the graphs presented in the Results and Analysis section, we scaled the data differently.

Our AP-lite control algorithm uses two infrared sensors to model the robot behavior of moving from start location to a goal location where goal is a light source. The algorithm maintains a global attractive goal force that is always active; this drives the robot forward with an equal amount of power to both motors. Again, the clarity of the graphs presented in the Results and Analysis section, we scaled the data differently. When the robot reaches an obstacle, AP-lite computes the repulsive forces acting on the robot, and changes the power supply to the motors to change the robot heading. If the robot senses an obstacle from the right (i.e. right infrared sensor reads a low value), AP-lite reacts to this repulsion by decreasing power to the left motor and/or increasing power to the right motor. If the robot senses an obstacle from the left (i.e. right sensor reads a high value), AP-lite reacts to this repulsion by decreasing power to the right motor and/or increasing power to the left motor. AP-lite measures the turning angle based on the virtual attractive and repulsive forces acting on the robot. The force vector of the AP-lite is computed every four milliseconds giving sufficient time for wireless data communication between the robot and the host computer.

We present results of our experiments in next section. All experiments are conducted indoor and averaged over five runs.

Results and Analysis

We test the stability of our software framework using a behavior based control algorithm and a physics based control algorithm, Ap-lite. The Figure 4 shows the sonar readings during navigation of the robot in the y-axis over time in the x-axis, where robot uses the behavior based control algo-

Figure 3: Robot view of the environment. Light source at the far end.

rithm. According to the robot view (see Figure 3) the first obstacle to the right is detected by the right sonar between the time 100 and 150, and both the left and the right sonars detect the last obstacle to the left before the robot reaches the goal at time 250 and 350. This is due to the fact that our robot is directly facing the last obstacle.

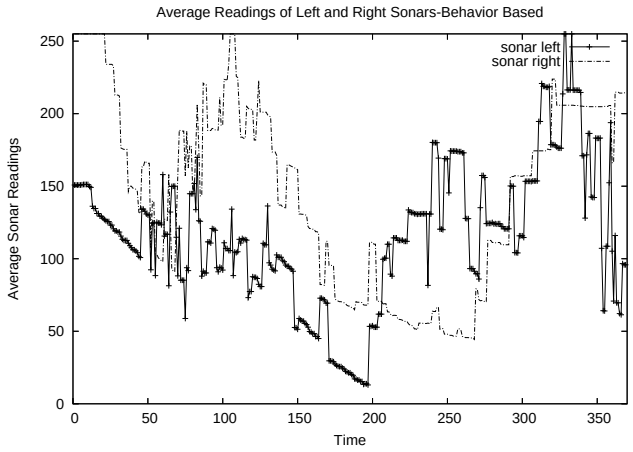

Figure 4: The sonar readings of the robot using behavior based algorithm.

The Figure 5 shows the power to motors during navigation of the robot in the y-axis over time in the x-axis, where robot uses the behavior based control algorithm. Though not very significantly, the power supply to the right motor increases when the robot reaches the first obstacle to the right, while the power to left motor remains unchanged. Then the robot takes several turns that we believe due to the fact that the robot is directly facing the last obstacle to the left before reaching the goal. This is also evident in the sonar readings in the Figure 4. The robot constantly change the power to two motors to keep the power consistent during these turns. We believe that this behavior can be corrected by finding an accurate balance of proper turning angle and proper filters to remove noise in the environment.

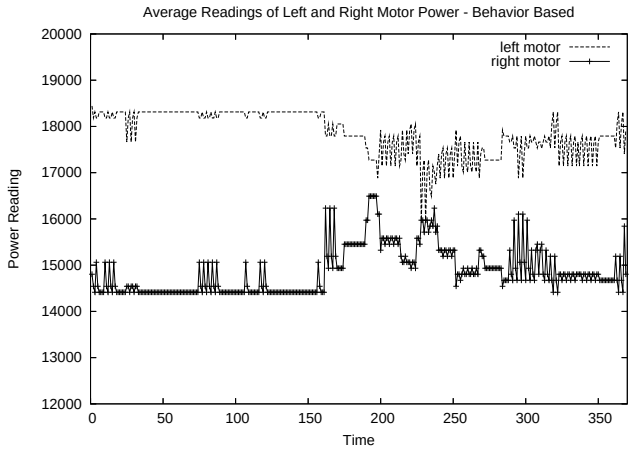

Figure 5: The power use by two motors of the robot using behavior based algorithm.

The Figure 6 shows the infrared sensor readings during navigation of the robot in the y-axis over time in the x-axis, where robot uses the AP-lite control algorithm. The results clearly show the robot reaching the first obstacle to the right where the right most infrared sensor reading decreases at times between 125 and 175, and the robot reaching last obstacle to the left before the goal at time 200 and 260. This behavior can clearly be seen in the power supply to the motor in the Figure 7 and the change in robot angle in the Figure 8.

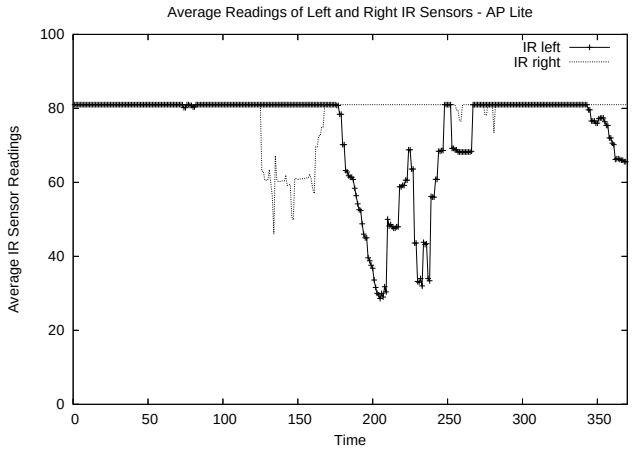

Figure 6: The infrared sensor readings of the robot using AP-lite algorithm.

The Figure 7 shows the power to motors during navigation of the robot in the y-axis over time in the x-axis, where robot uses the AP-lite control algorithm. When the robot reaches the first obstacle to the right, repulsive forces are high from the right side of the robot. The resulting force vector causes the robot to reduce power to the left motor, but increase power to the right motor, allowing the robot to

take a quick and sharp turn. Since this sharp turn causes the robot to keep an easy focus on the goal robot does not reach the last obstacle to the right, but detects the last obstacle to the left before the goal. To avoid the obstacle to the left of the robot at time 180, AP-lite control algorithm reduces the power to right motor while keeping the power to left motor the same.

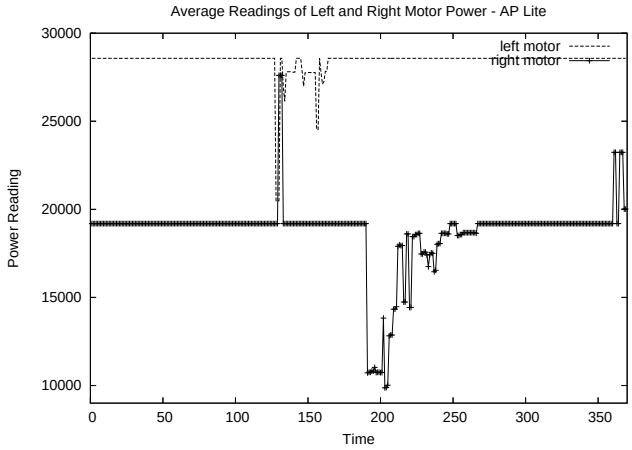

Figure 7: The power use by two motors of the robot using AP-lite algorithm.

We also measure the turning angle of the robot in AP-lite algorithm, since AP-lite computes the turning angle based on the attractive and repulsive forces exerted on the robot by obstacles and the goal. The Figure 7 shows the turning angle during navigation of the robot in the y-axis over time in the x-axis. Once again, it is apparent that the robot's left turn occurs with the first obstacle to the right, and the robot's right turn occurs with the last obstacle to the left before the goal. We believe that the robot force vector computation favors the least resistant path from the starting point to the goal.

Related Work

Both behavior-based and rule-based systems have proved quite successful in exhibiting a variety of behaviors in a heuristic manner. Fredslund and Matarić studied the problem of achieving global behavior in a group of distributed robots using only local sensing and minimal communication, in the context of formations (Fredslund and Matarić 2002). The key concept of their algorithm is that each robot follows a designated "friend" robot at the appropriate angle and distance using a proximity sensor that can provide the angle and distance information of the friend. By panning the sensor appropriately, the algorithm simply keeps the friend centered in the sensor's view. They presented their results using four and eight robots in different formations. Balch and Arkin accomplished robot formations using the following two step process: "detect-formation-position" which is a perceptual process that determines the robot's

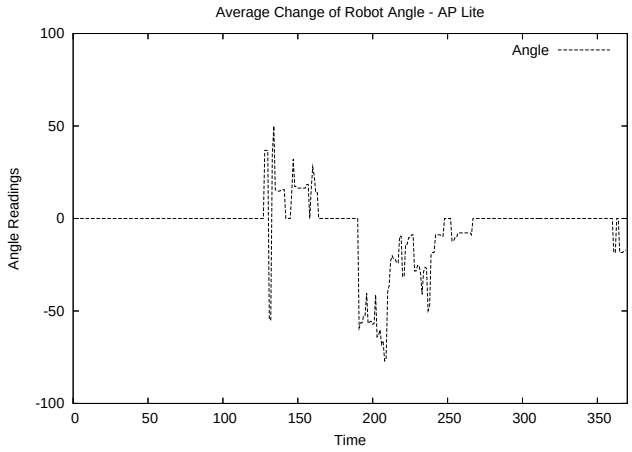

Figure 8: The turning angle of the robot using AP-lite algorithm.

position in the formation based on the current environment data, and "maintain-formation" which generates motor commands to direct the robot towards the correct location (Balch and Arkin 1998).

One of the earliest physics-based techniques is the *potential fields* approach (e.g., (Khatib 1986)). Most of the PF literature deals with a small number of robots (typically just one that is similar to our experimental setup) that navigate through a field of obstacles to get to a target location. The environment, rather than the robots, exert forces. Obstacles exert repulsive forces while goals exert attractive forces (Kim 1991; Koren 1991).

The *social potential fields* (SPF) framework is highly related to physicomimetics framework (Reif 1998). Reif and Wang rely on a force-law simulation that is similar to the physicomimetics approach, allowing different forces between different robots. Their emphasis is on synthesizing desired formations by designing graphs that have a unique potential energy (PE) embedding.

Conclusion and Future Work

We are interested in designing applications for autonomous mobile robots and robot swarms. We use X80Pro mobile robots designed and developed by Dr.Robot Inc. for our applications. The vendor provided software framework was inflexibility and lack user friendliness required to develop software applications with undergraduate student researchers. We presented a user friendly software framework for X80Pro robots that will allow even a novice C++ programmer to design and implement autonomous mobile robot applications. We explored the feasibility of our software framework using behavior-based and physics based control algorithms where a X80Pro robot avoid obstacles to reach a goal. We are capable of producing predictable robot be-

havior using these control algorithm. We believe that the behavior based control algorithm needs to be studied further to provide a proper conclusion. The AP-lite shows significant predictability of the robot behavior in our user friendly software framework.

Future work of this research will focus on a Java based software framework for X80Pro robots and improved control algorithms to test the feasibility of the software framework. We would also extends the implementation of our algorithms to multi-robot systems since the algorithms are already theoretically extended to handle swarm of robots. This will also allow us to make use of techniques presented in (Reif 1998). Another improvement would be to implement filters to eliminate noise in the environment and test the robot behavior in more complex environments.

Acknowledgements

Authors are especially greatful to the IU Southeaset research and grant committee for providing financial support to this work. Authors would like to acknowledge the valuable support by IU Southeast Dean for Research, Dean of School of Natural Sciences, Department of Computer Science, and contributions by all the students who took C458 Intelligent Robots course in spring 2011.

References

Dr. Robot Inc. (2012). Dr. Robot Inc - Extend Your Imagination. *http://www.drrobot.com/products_item.asp?itemNumber=X80Pro*

Matarić, M. (1999). Behavior-Based Robotics. *MIT Encyclopedia of Cognitive Sciences, Robert A. Wilson and Frank C. Keil, eds., MIT Press,*: 74-77.

MFC ActiveX Controls (2005). Microsoft Visual Studio - MFC ActiveX Controls. *http://msdn.microsoft.com/en-us/library/k194shk8(v=vs.80).aspx*

ActiveX Controls: Events (2005). Microsoft Visual Studio - ActiveX Controls: Events. *http://msdn.microsoft.com/en-us/library/aa268929(v=vs.60).aspx*

Spears, W., Spears, D., Hamann, J. and Heil, R. (2005). Distributed, Physics-Based Control of Swarm of Vehicles. Autonomous Robots, *Kluwer,* **17**: 137–164.

Balch, T. and Arkin, R. (1998). Behavior-based Formation Control for Multi-Robot Teams. *IEEE Transactions on Robotics and Automation,* **14**: 1–15.

Spears, W. and Spears, D., eds. (2011). Physicomimetics-Physics Based Swarm Intelligence. Springer.

Fredslund, J. and Matarić, M. (2002). A General Algorithm for Robot Formations Using Local Sensing and Minimal Communication. *IEEE Transactions on Robotics and Automation,* **18**: 837–846.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research,***5, (1)**: 90–98.

Kim, J. and P. Khosla (1991). Real-time obstacle avoidance using harmonic potential functions. *IEEE International Conference on Robotics and Automation,*: 790796.

Koren, Y. and J. Borenstein (1991). Potential eld methods and their inherent limitations for mobile robot navigation. *IEEE International Conference on Robotics and Automation,* 1398–1404.

Reif, J. and H. Wang (1998). Social potential elds: A distributed behavioral control for autonomous robots. *Workshop on the Algorithmic Foundations of Robotics.*