QBOT: AN EDUCATIONAL MOBILE ROBOT CONTROLLED IN MATLAB SIMULINK ENVIRONMENT

Rajibul Huq, Hervé Lacheray, Cameron Fulford, Derek Wight, and Jacob Apkarian

Quanser Consulting Inc., Markham, ON, Canada
Email: {rajibul.huq,herve.lacheray,cameron.fulford,derek.wight,Jacob.Apkarian}@Quanser.com

ABSTRACT

This paper describes Quanser's Mobile Robot Control Framework (QMRCF) for an educational robot called Qbot. The QMRCF accelerates the development of mobile robot control algorithms and hardware-in-the-loop (HIL) testing using model-based design techniques of MATLAB Simulink. This paper also validates the performance of QMRCF using several experiments in various research fields of mobile robotics, e.g., teleoperation, navigation, and obstacle avoidance.

***Index Terms*—** Mobile robot control, MATLAB Simulink, hardware-in-the-loop testing

1. INTRODUCTION

Researchers today are focusing on mobile robots because of their potential applications in hazardous environments [1], agriculture and harvesting [2], house-hold tasks [3], and medical applications [4]. The core technologies involved in these applications are sensory augmented remote control of mobile robots, capability of self-localized autonomous navigation, obstacles avoidance, and intelligent decision-making for task execution [5]. Successful implementation of these techniques largely depends on various sensors, e.g., shaft encoder for position estimation of the robot, range finder for obstacle avoidance, vision system for visual pattern recognition, and force feedback for haptic application.

Owing to their distinct automation capabilities, study on mobile robots and its sensors becomes an indispensable part of automation research. The development process of mobile robot control algorithms requires a suitable robotics platform equipped with appropriate sensors and a software framework for fast validation of concepts. The existing robotic platforms, e.g., iRobot [6] and Pioneer Robot [7], provide robust robotic architectures for different applications. However, their software framework requires hand-coded program for concept testing, which is a barrier for fast development process of control algorithms.

To overcome this problem, Quanser has developed a mobile robot control framework using the Simulink interactive graphical environment and a customizable set of block libraries. Quanser's Mobile Robot Control Framework (QMRCF) accelerates the development process with hardware-in-the-loop (HIL) testing in Simulink environment, which facilitates the design, simulation, and implementation of a wide variety of robotic applications. Glasgow Caledonian University has also developed a Simulink-based mobile robot team control toolbox [8]. The toolbox, however, was developed only for home-made microcontroller-based robots and it does not provide interfaces to existing commercially available robots and sensors. Quanser uses iRobot Create as the robotic platform and has added Gumstix-based [9] on-board computational power in addition to its basic platform. The Quanser robotic platform, called Qbot, is also equipped with a camera and range sensors in addition to the basic sensors provided with iRobot Create.

The rest of the paper is organized as follows. Section 2 describes the hardware and software framework for Qbot. Section 3 illustrates the development process of control algorithms using QMRCF. Experiments using QMRCF are presented in Section 4. Finally, Section 5 outlines the future extensions of this work and Section 6 draws the conclusion.

2. HARDWARE AND SOFTWARE TOOLS

2.1. Hardware framework

Fig. 1(a) and 1(b) show the basic iRobot Create robotic platform and the partially developed Qbot, respectively. The basic iRobot Create is a two-wheel differential robot and it is equipped with various built-in sensors, e.g., bump sensors and wheel encoder. The present form of Qbot includes:

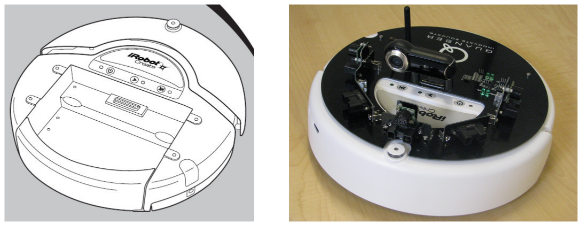
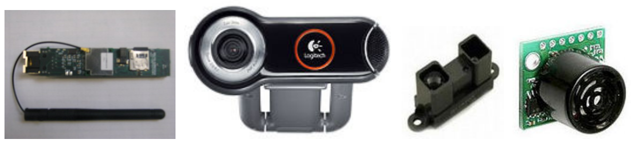

(a) iRobot Create (b) Qbot

Fig. 1. Quanser's robotic platform

- A Gumstix-based on-board computer that uses Verdex XL6P motherboard and PXA270 XScale 600 MHz processor (see Fig. 2(a)). It has 128MB RAM and 32MB Flash memory with on-board MicroSD memory slot. It provides both Ethernet and WiFi connectivity to facilitate networking capability. The on-board USB host controller enables to interface with a CCD camera. For further detail see [9].
- A Logitech Quickcam Pro 9000 USB camera (see Fig. 2(b)) for on-board image acquisition.
- Five Sharp GP2Y0A02YK IR range sensors (Fig. 2(c)) and three Maxbotix MaxSonar-EZ0 sonar range sensors (Fig. 2(d)).



(a) Gumstix (b) Camera (c) IR sensor (d) Sonar sensor

Fig. 2. Additional hardwares of Qbot

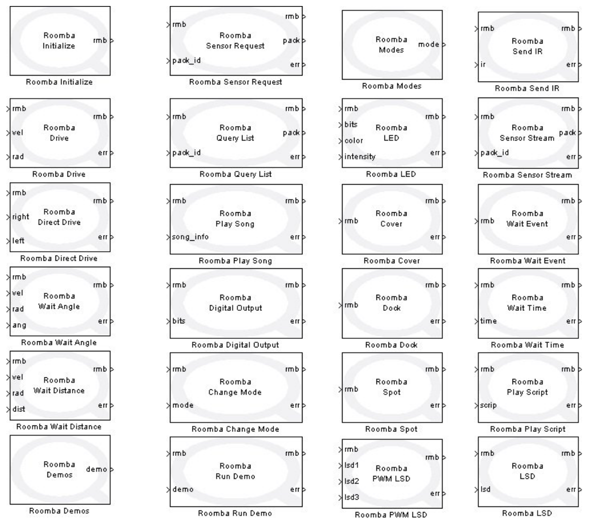
2.2. Software framework

The software framework for Qbot includes: 1) Open Embedded (OE) Linux operating system (OS) for Gumstix and 2) library blocksets for robotic control. The OE-Linux OS has been adapted to incorporate Quanser's rapid control prototyping (QuaRC) [10] environment on Gumstix. The GNU compiler is also included in the OS to facilitate on-board compilation of C/C++ code. The current library blocksets, based on QuaRC, has three-fold development areas: Interface, Application, and Image processing blocksets. These blocksets allow a user to build a model-based controller in Simulink and to generate suitable code for Gumstix on Windows or QNX for preliminary testing. The following sections describe the blocksets and code generation process for Gumstix.

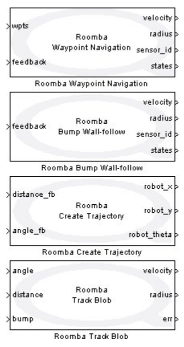
2.2.1. *Interface blockset*

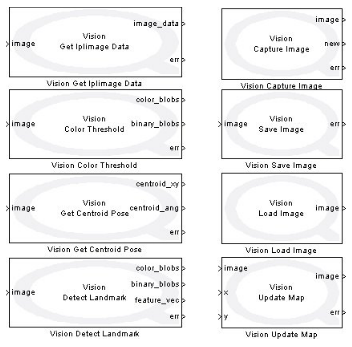
This blockset (see Fig. 3(a)) implements the basic application program interfaces (APIs) provided by iRobot Create. The APIs can be broadly categorized based on the following functionalities:

- serial connection configuration between the high level controller (e.g., PC or Gumstix) and Qbot,
- setting the operation mode of Qbot,
- accessing Qbot's sensory information,
- running built-in demos and playing songs, and
- other hardware configuration, e.g., setting digital and analog I/O ports, and changing LED colors.



(a) Interface





(b) Application (c) Image processing

Fig. 3. Library blocksets

2.2.2. *Application blockset*

The Application blockset (see Fig. 3(b)) allows a user to implement navigation algorithms using available sensory information. This blockset is currently under development and the present blocks use only wheel encoder data and bump sensors for navigation and obstacle avoidance.

2.2.3. *Image processing blockset*

This blockset (see Fig. 3(c)) implements image acquisition and processing functionalities using the Open Source Computer Vision (OpenCV) library [11]. The image processing blockset allows images to be captured from a USB camera, process it on-line, and save it to disk for further analysis. This blockset is currently undergoing extensive development and will include other available functionalities in OpenCV.

2.2.4. *Code generation*

The code generation functionality allows a Simulink diagram to be executed real-time on Qbot. The diagram can include library blocksets mentioned in Scetions 2.2.1-2.2.3 as well as regular Simulink and QuaRC library blocks. MATLAB Real Time Workshop (RTW) is employed to generate target-specific code, which is ported to Gumstix using TCP/IP. The final executable is obtained by compiling the generated code on Gumstix using GNU compiler. It should be noted that this process will still provide all of the functionalities of the Simulink diagram during the model execution on Qbot, e.g., the ability to modify gains or view any signal in the original diagram within its own window.

3. DEVELOPMENT PROCESS OF CONTROL ALGORITHMS

This section describes the development process of robot control algorithms using QMRCF. The following steps are undertaken in order to design, launch, and monitor an application on Qbot:

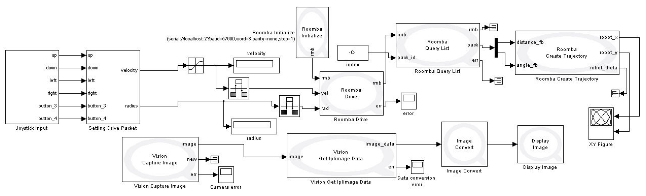
1. Open MATLAB/Simulink as the application design environment for QMRCF.
2. From the Quanser Toolboxes use Interface blockset to initialize serial connection of Qbot.
3. Design an application using appropriate blocks from Interface, Application, and Image Processing block libraries. Note that any regular blocks in Simulink and QuaRC libraries can be used to design an application.
4. Generate code for Qbot (which is then automatically downloaded and compiled on Qbot).
5. Start and monitor the application progress.

4. EXPERIMENTS

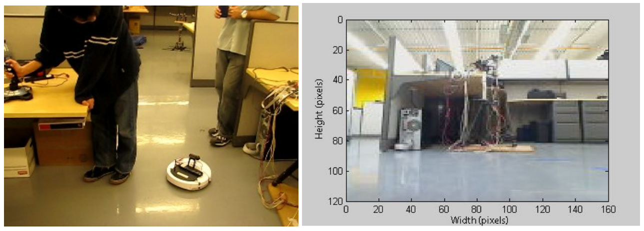
This section employs the development process described in Section 3 to design, launch, and monitor mobile robot applications on Qbot using QMRCF. The experimental results are demonstrated in the following fields of mobile robot applications: teleoperation, navigation, and obstacle avoidance.

4.1. Teleoperation

The goal of this application is to remotely control a mobile robot with a joystick. Fig. 4(a) shows the Simulink diagram for this task, which uses Quarc's *Host Game Controller* block under *Joystick Input* subsystem to generate motion commands for Qbot with a joystick. The Interface library blocks *Roomba Initialize*, *Roomba Drive*, and *Roomba Query List* are used to setup Qbot's serial connection, drive the wheels, and access wheel encoder data, respectively. The Application library block *Roomba Create Trajectory* is used to generate robot's trajectory. The Image Processing library blocks *Vision Capture Image* and *Vision Get Iplimage Data* are used for image acquisition and data extraction from the camera. The rest of the blocks from Simulink and QuaRC libraries are used for monitoring different signals and images. Fig. 4(b) shows the



(a) Simulink model



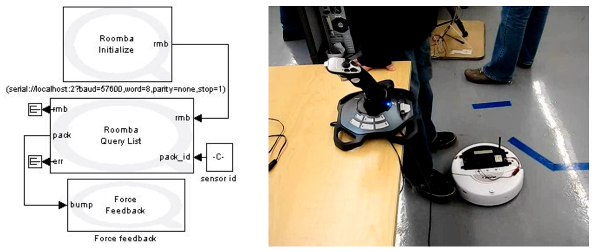
(b) Experimental setup (c) On-board camera view

Fig. 4. Qbot teleoperation

experimental setup where a teleoperator is driving Qbot with a joystick. Fig. 4(c) depicts the image data streamed back from Qbot to the host Windows PC.

4.2. Force feedback

The goal of this experiment is to demonstrate application of force-feedback for teleoperation. The force-feedback is associated with Qbot's bump sensors. The operator feels the reaction force on a COTS joystick while the bump sensor collides with any objects (see Fig. 5(b)). QuaRC's Force Feedback



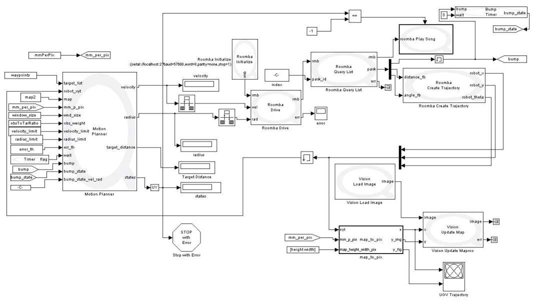
(a) Simulink model (b) Experimental setup

Fig. 5. Qbot force feedback

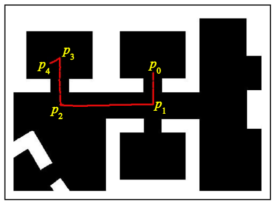
Game Controller blockset is used (inside *Force Feedback* subsystem) in association with Interface blockset to design the Simulink diagram described in Fig. 5(a).

4.3. Navigation and obstacle avoidance

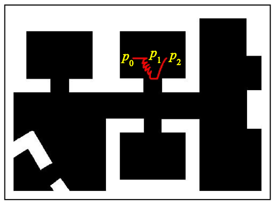
Fig. 6(a) shows the navigation model, where a set of waypoints is defined using an occupancy map (see Fig. 6(b)) to be reached by the robot. White pixels of the occupancy map indicate obstacles whereas black pixels indicate empty space. The



(a) Simulink model



(b) Trajectory of waypoint-based navigation



(c) Obstacle avoidance

Fig. 6. Qbot map-based navigation and obstacle avoidance

Motion Planner subsystem, a MATLAB Embedded Function, generates motion commands according to the current position of the robot and the target waypoint. The *Play Song* subsystem employs *Roomba Play Song* block to generate a beep sound when Qbot reaches the final target point. The robot's trajectory is continuously updated with the help of *Vision Update Map* block. Fig. 6(b) depicts the complete trajectory (red pixels) where the robot starts at point p_0 and reaches the waypoints p_1, p_2, p_3, and p_4 to accomplish the navigation task.

The *Motion Planner* also maintains a short-term memory (defined by *Bump Timer* subsystem) of the previously bumped obstacle's position. The robot, first, moves opposite to the obstacle and then rotates and moves forward to avoid the obstacle. Fig. 6(c) shows an instance where the robot starts at p_0 and tries to avoid obstacle at point p_1 and finally, reaches the target position at p_2.

5. FUTURE WORKS

The ongoing extension of this work includes the development of hardware and software interfaces between Gumstix and range sensors. Further development will include a Multi-agent blockset to facilitate co-operative task execution of mobile robots.

6. CONCLUSION

This paper described a robotic control framework in MATLAB Simulink environment. The control framework has been developed using Quanser's rapid control prototyping library. This framework includes a mobile robot platform called Qbot, which is adapted from iRobot Create. Qbot is equipped with a Gumstix-based on-board computer, a USB camera, and infrared range sensors in addition to the built-in sensors provided by iRobot. The software framework consists of three blocksets, namely, Interface, Application, and Image Processing that facilitate to design, launch, and monitor a mobile robot application on Qbot. The performance of the developed architecture has also been tested using several experiments in various research fields of mobile robots, e.g., teleoperation, navigation, and obstacle avoidance. The future goal is to develop a Multi-agent blockset, which will provide communication and data management libraries for co-operative task execution of mobile robots.

7. REFERENCES

[1] G. Kantor, S. Singh1, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer, "Distributed search and rescue with robot and sensor teams," in *Field and Service Robotics*. 2006, vol. 24, pp. 529–538, Springer Berlin / Heidelberg.

[2] B. Astrand and A.J. Baerveldt, "An agricultural mobile robot with vision-based perception for mechanical weed control," *Autonomous Robots*, vol. 13, no. 1, pp. 21–35, July 2002.

[3] M. Hans, B. Graf, and R.D. Schraft, "Robotic home assistant care-Obot: Past-present-future," in *Proc. IEEE International Conference on Robot and Human Interactive Communication*, 2001, pp. 407–411.

[4] J. Eriksson, M.J. Mataric, and C.J. Winstein, "Hands-off assistive robotics for post-stroke arm rehabilitation," in *Proc. IEEE International Conference on Rehabilitation Robotics*, June 2005, pp. 21–24.

[5] R. Murphy, *Introduction to AI Robotics*, MIT Press, 2000.

[6] iRobot, *http://www.irobot.com/*.

[7] Pioneer Robot, *http://www.mobilerobots.com/*.

[8] Mobile Robot Team Control Toolbox, *http://www.dziewierz.pl/moroteco/index.html*.

[9] Gumstix, *http://www.gumstix.com*.

[10] QuaRC, *http://www.quanser.com/quarc*.

[11] Open Source Computer Vision Library, *http://en.wikipedia.org/wiki/OpenCV*.