Developing Feedback-Control Prototypes using a Real-Time Simulink Environment

Jenelle Armstrong Piepmeier, Richard T. O'Brien, Jr.
U.S. Naval Academy
118 Maryland Ave (Stop 14A)
Annapolis, MD 21402

An effective undergraduate controls curriculum will have strongly coupled laboratory and classroom components. It is important that the students experience the application of the classroom theory. Mathwork's Simulink environment is ideal for teaching modeling and simulation of feedback control systems. However, with the addition of a few blocks from the Humusoft Real-Time Toolbox, the Simulink environment can be used for real-time feedback control of a variety of engineering systems. Humusoft's pre-compiled Simulink blocks allow the integration of hardware interfaces in the Simulink model. Options include serial port access as well as access to a variety of data acquisition boards. This approach allows students to quickly develop control systems within a familiar environment that is equipped with visualization tools such as displays. This software has been used to develop a number of different feedback control systems by students and faculty. The simplicity of implementation makes it ideal for engineering students at all levels. This paper discusses several successful examples including robotic visual servoing, scale-model vehicle identification and control, and automated vehicle navigation.

Background

Controls educators have long worked to bridge the gap between the conceptual theory and the practical implementation. Software environments such as MATLAB provide visualization tools such as graphs, displays, and animations that allow the student to develop engineering intuition about system behaviors under varying types of control and initial conditions. The Mathworks website[1] lists over 100 MATLAB-based controls textbooks including commonly used texts such as Nise [1] and Ogata [2]. With the Control Systems Toolbox, MATLAB and Simulink are ideal environments for the design and simulation of engineering control systems. However, recent literature has pointed to the need for an increase in hands-on or experiential learning in controls education [3-5]. Hands-on activities provide students with the chance to study all aspects of a system, and not just the controller [5]. Students need experience with more than just system modeling or controller design; they need to understand the technical issues associated with sensors, and actuators. Practicing control engineers spend much of their time dealing with these less theoretical issues, and it is important that students are exposed to these topics with hands-on exercises [6].

There are a number of solutions to this challenge. Within the commonly used MATLAB environment, the Instrument Control and Data Acquisition Toolboxes make it possible to interface with data acquisition boards, use RS-232, GPIB, VXI, TCP/IP and UDP communication

[1] http://www.mathworks.com

protocols, and call generic DLLs directly from MATLAB to interface with non-supported hardware. However, these capabilities are not all available in the graphical Simulink environment. The addition of Mathwork's Real-Time Workshop allows the user to generate optimized, portable, and customizable ANSI C code from Simulink models. This can be used with third-party products such as Quanser's Feedback Control experiments[2] or Educational Control Products[3]. Students can develop and test Simulink control experiments using systems such as a helicopter or inverted pendulum. At the Naval Academy, these experimental products have been invaluable in the controls education.

It is important that engineering educators be aware of limitations of these canned experimental products. Agrawal has correctly pointed out that the integration of the equipment can easily overwhelm limited resources of both time and departmental budgets [3]. Bissell has also noted that some of the canned experiments are so carefully engineered that they become more of an illustration in theory and less of an experience in control *system* design [6]. Even with a well-equipped laboratory, there are courses and student projects for which this equipment is either not available or suitable.

This paper discusses the use of a Simulink toolbox product developed and sold by Humusoft[4] that can be used to integrate a Simulink controller with actual engineering system. This product is simple to use, inexpensive, and allows students and faculty to quickly interface with sensors and actuators in a familiar computer environment. We believe this product is a useful tool for increasing the experiential learning of the controls systems student.

Humusoft Real-Time Toolbox

The Real-Time Toolbox product includes Simulink blocksets for data acquisition and control of a variety of products. The blocks do not require compilation (unlike the Mathwork's Real-Time Workshop) and come with a real-time kernel that allows the Simulink model to run real-time. The software comes with drivers for more than 100 industry-standard data acquisition boards available including A/D, D/A, digital I/O, quadrature encoders, counters, mouse, joysticks, etc. The capability for serial communication support Simulink is especially useful.

Examples

Robotic Visual Servoing

An experimental testbed was designed to test the controller and demonstrate the flexibility of uncalibrated visual servo control of an uncalibrated robotic manipulator. The system consists of a color camera, a two-link reconfigurable robotic manipulator, and a desktop PC running MATLAB with Simulink. The controller (implemented in an S-function) is a Gauss-Newton optimization technique that utilizes system estimation [7]. A CMUCam[5] provides color-tracking data of a moving colored object. The CMUCam features on-board processing and outputs color-tracking data via an RS232 serial connector at 115200 baud 17 times a second. Using Humusoft's Real-

[2] http://www.quanser.com
[3] http://www.ecpsystems.com
[4] http://www.humusoft.com
[5] http://www-2.cs.cmu.edu/~cmucam/

Time Toolbox, MATLAB's Simulink environment was used as a real-time control environment. The Real-Time Toolbox Serial-In and Serial-Out blocks obtain vision information from the camera, compute joint angles using Gauss-Newton algorithms, and output motor position commands. The overall system is running at 1 Hz. Each cycle includes image processing, calculations, and robot motion. Figure 1 shows the Simulink diagram used in this experimental work. Humusoft blocks are denoted with shadowed blocks. By changing the S-function, different control algorithms could quickly be tested. Displays such as the xy-plot *error* block allow the user to view the system behavior in real time.

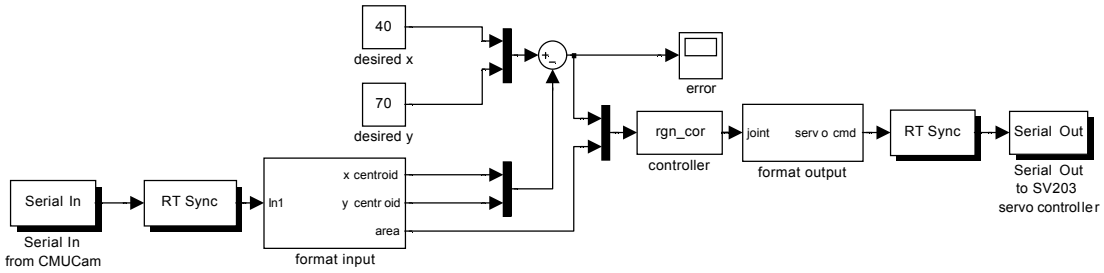


Figure 1 Simulink diagram implementing uncalibrated visual servo eye-in-hand control. Vision data from the CMUcam is compared with the desired target position, and the appropriate joint angles are calculated in the S-function *rgn_cor*, which is computing the dynamic recursive Gauss-Newton controller with velocity correction. The desired joint angles are sent to the serial motors using Humusoft's *Serial Out* block.

Motor commands were sent to Pontech's SV203 servomotor control board[6], which controls the robot's servomotors. The robot is a 2 DOF planar robot constructed using a Robix RCS-6 kit. The reconfigurable robot permits a demonstration of the robustness of the uncalibrated approach. The camera is affixed to the end of the second link of the robot arm looking down on a small target. The target consists of a one-inch diameter red object that is moved in an arc by an additional servomotor beneath the robot arm. The goal is to move the robot arm, and the camera, so that the moving target is maintained at the center of the camera's field of view.

Figure 2 shows the results of an XY graph, a Simulink display tool, which allows the user to easily note system performance. The figure shows the time history of the target location within camera image. This plot was captured in the early stages of implementation. By viewing the plot shown in Figure 2, it is easy to see that the controller is achieving the goal but with poor precision. Further investigation showed that the sampling time of the Simulink model was not allowing enough time for sensing the complete movement of the robot. Because of the optimization nature of the controller, it is algorithmically important that the effects of the change in joint motion be sensed by the vision system before the next joint angle is computed. The time it takes to sense, process, and send commands an example of a technical issue that is often ignored during simulation exercises.

[6] http://www.pontech.com

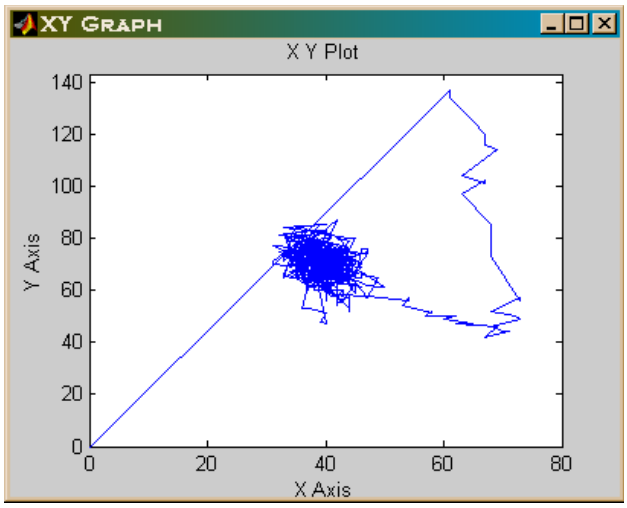


Figure 2 An example of a Simulink XY graph captured during testing of a vision-based robotic control system. The objective is to keep the target object centered in the camera's field-of-view. This plot was captured in the early stages of implementation. It is easy to see that the controller is achieving the goal but with poor precision. The graphical nature of the display allows the engineer to quickly diagnose problems such as this.

Figure 3 shows tracking error for this controller and target speed as generated in MATLAB. The average (x, y) error is (1.70, 2.01) *pixels*. An interesting aspect of the nature of the controller that was implemented was its independence from the system configuration. Theoretically, changing the position or orientation would have no effect on the system performance. However, when the camera was rotated 90°, the tracking error increased to (1.93, 7.48) *pixels*. This result is because the camera has a resolution of 80× 143 non-square pixels. The tracking performance was roughly the same in the Cartesian frame, but significantly larger in the sensor frame due to the nature of the sensor.

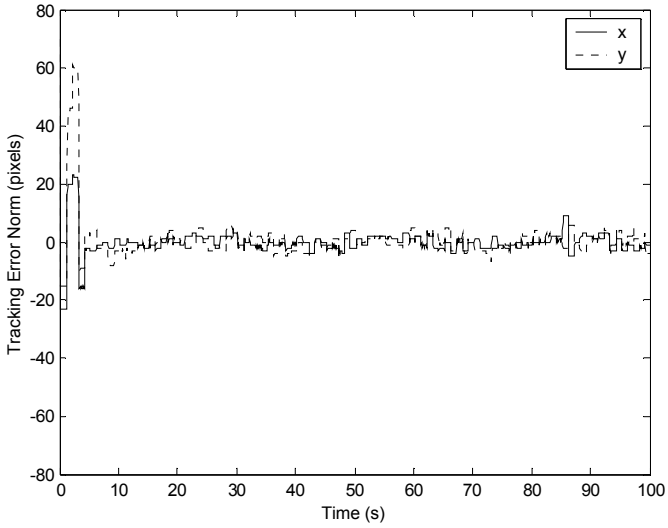
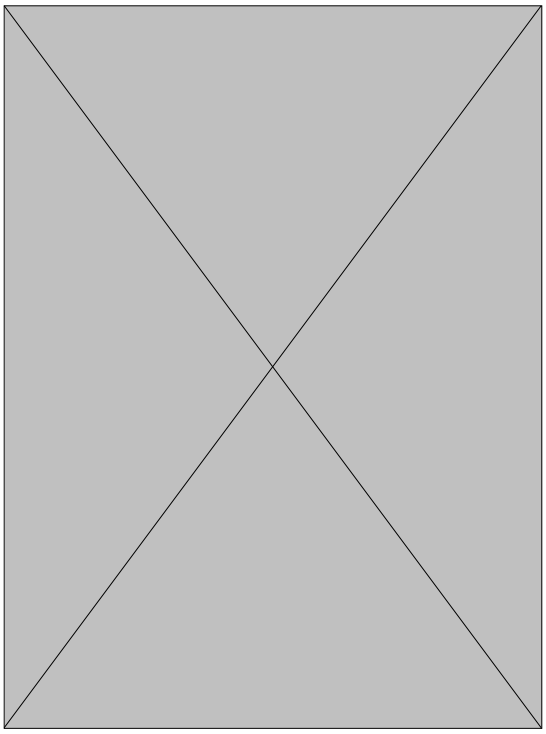


Figure 3 Tracking error for uncalibrated 2DOF eye-in-hand tracking using the Recursive Gauss-Newton controller with a target moving 7.5 pixels/second. The average steady state tracking error is (1.70, 2.01) pixels.

Scale-Model Vehicle Identification and Control

An automotive scale-model vehicle testbed has been developed to investigate driver-assistance steering control system [8] as shown in Figure 4. The driver-assistance steering control system activates only in an emergency and performs an analogous function to an anti-lock braking system. The control system augments the driver's steering input to prevent the driver from steering the vehicle into a roll or spin. While testing on full-size vehicles would be preferable, scale-model testing has been a successful alternative to expensive and potentially dangerous full-size testing [9].

The testbed consists of an off-the-shelf, remote-control (RC) vehicle, an exercise treadmill, a DVT camera[7], and a PC-based-controller. Using the camera, the x- and y-coordinates of the RC vehicle are computed and the controller computes the inputs to the vehicle's electric motor and steering servo. The objective is to track a stationary position and, therefore, regulate the RC vehicle to the same speed as the treadmill. As a result, the RC vehicle simulates the motion of a full-size vehicle on a highway. Originally, the controller was programmed in C but it was difficult to implement complex control schemes and to achieve a consistent sampling rate. Currently, the apparatus is being updated and the control system is being implemented using the Humusoft software. The software facilitates the implementation because the *Serial/In* and *Serial/Out* blocks interface with a DVT SmartSensor camera and the SV203 servomotor control board. The student is able to implement the controller in the environment that was used for preliminary designs and simulations.



[7] http://www.dvtsensors.com

Figure 4: Automotive scale-model vehicle testbed developed at the Naval Academy

Automated Vehicle Navigation

Two student groups are working together to develop a vision-based, steering autopilot for a full-size industrial cart a small color digital camera as the feedback sensor. Initial design and testing has been done using MATLAB and Simulink. A servomotor has been attached to the steering column to produce the desired steering input, and the cart has been equipped with a separate cruise control system to maintain the desired speed. The cart will navigate by determining its position relative to a marked path and steering the vehicle towards the path. The desired path is defined by tape that has a high level of color contrast with its background. The Simulink model of the autopilot is shown in Figure 6.

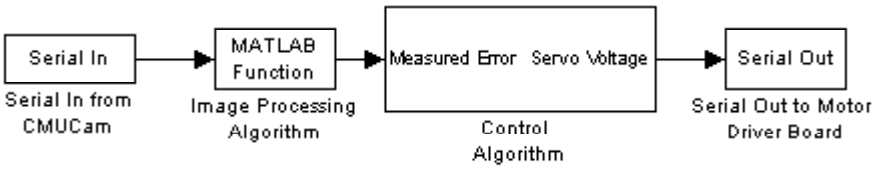


Figure 5: Simulink model of image processing and control algorithms for vision-based, steeing autopilot.

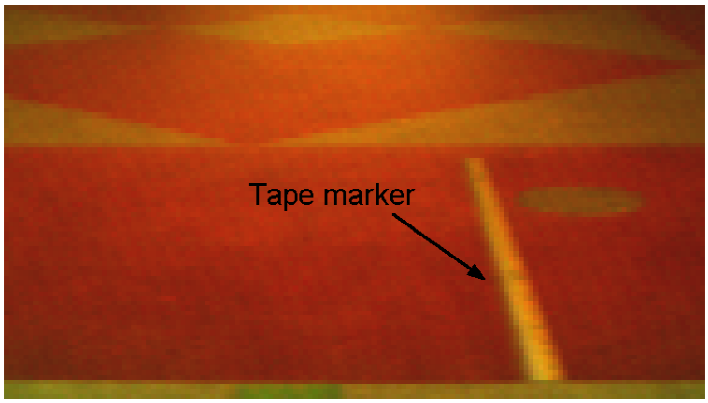


Figure 6 A test image from the CMUCam showing the tape marker on an indoor surface. This image was used to perform an open-loop test of the image-processing and control algorithms.

The first group is working on implementing the vision system. Image data from the CMUCam (such as in Figure 7) will be brought into the Simulink environment using the Humusoft interface. An S-function will be used to process the data and determine the angle and relative position of the reference line. In developing the system, students developed the necessary processing techniques and determined the calibration between image positions and Cartesian distances. Again, the Humusoft blocks will allow them to continue to implement the system in the environment they have been using for testing and development. System latencies, such as image processing will result in a very slow sampling time, but the system will serve as a proof-of-concept that can lead to further implementation using a microprocessor.

A second group has been developing the control algorithm as discussed in [10, 11]. Using a 3952 Full-bridge PWM motor driver, the output of the control algorithm drives the servomotor. This

nature of the algorithm makes a Simulink implementation much easier than a digital implementation that requires multiple interrupts.

An open-loop test of the image-processing and control algorithm was performed using the image in Figures 7 and the Simulink diagram in Figure 9. From the image-processing algorithm, the orientation and lateral offset of the vehicle. The control algorithm was designed to steer the cart to the path marker using a two-second period and a unity proportional feedback gain. As a result, the controller will steer the cart to path marker in two seconds.

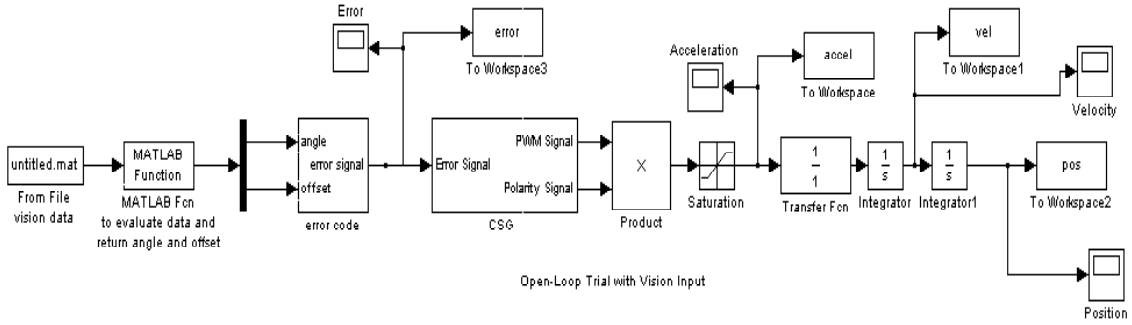


Figure 7 SIMULINK diagram for the open-loop test of the image-processing and control algorithms

The cart is modeled as a pure double integrator and the simulation results are shown in Figure 10. The control algorithm generates the piecewise constant signal (labeled as the acceleration) that represents the voltage to the steering servomotor. The position (in centimeters) changes by the desired value ($1.78cm$) and the heading between the cart and the path returns to $-0.194°$ after the two-second interval.

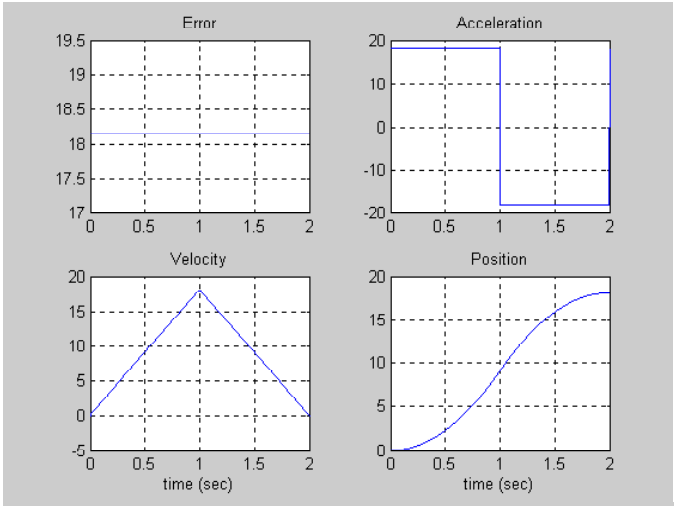


Figure 8 Simulations results for open-loop test of the image-processing and control algorithms.

The student groups working on this project were able to use the Simulink environment to model the cart behavior, design the controller, and prototype processing the sensor data. As part of an undergraduate design project, the students will implement their design within the same environment using the Humusoft toolbox.

Conclusion

This paper has described several applications of a software product that enables real-time control of engineering systems within the Simulink environment. The product is simple to implement and provides several advantages to control educators. Students are able to develop prototypes of controllers rapidly for a variety of engineering systems using commercially available products such as color cameras, RC model cars, or an industrial cart. Students develop experience not only in the design of controllers, but also in the implementation of sensors and actuators. This type of hands-on experience is invaluable in the education of undergraduate engineering students.

References

[1] N.S. Nise, *Control Systems Engineering*, 3rd Edition, John Wiley & Sons, New York, 2000.
[2] K. Ogata, *Modern Control Engineering*, 4th Edition, Prentice-Hall, New Jersey, 2002.
[3] Agrawal, Sunil K. "Undergraduate Control Education: An ME Perspective", *Proceedings of the American Control Conference*, San Diego, CA, June 1999, 983-986.
[4] Bernstein, Dennis S. "Enhancing Undergraduate Control Education", *IEEE Control Systems Magazine*, Vol. 19(5), Oct 1999, 40-43.
[5] Dorato, Peter. "Undergraduate Control Education in the U.S.", *IEEE Control Systems Magazine*, Vol. 19(5), Oct 1999, 38-39.
[6] Bissell, C.C. "Control Education, Time for a Radical Change?", *IEEE Control Systems Magazine*, Vol. 19(5), Oct 1999, 44-49.
[7] Piepmeier, J.A., Gumpert, B.A., and H. Lipkin. "Uncalibrated Eye-in-Hand Visual Servoing", *Proceedings of the IEEE International Conference on Robotics and Automation,* Washington, D.C., May 2002, 568 –573.
[8] Burns, S.R., O'Brien, Jr., R.T., and Piepmeier, J.A. "Driver Assistance Steering Control Compensation of Accelerating Vehicle Motion", *Proceedings of the Southeastern Symposium on Systems Theory*, Huntsville, AL, March 2002, 476-478.
[9] Hoblet, P.C., O'Brien, Jr., R.T., and Piepmeier, J.A. "Scale Model Vehicle Analysis for the Design of a Steering Controller", *Proceedings of the Southeastern Symposium on Systems Theory*, Morgantown, W. Va., March 2003, Accepted for publication.
[10] Vasek, M.G. and O'Brien, Jr., R.T. "Bang-Bang Control of Double Integrator Systems", *Proceedings of the Southeastern Symposium on Systems Theory*, Huntsville, AL, March 2002, 275-278.
[11] O'Brien, Jr., R.T., Boernke, E.P., and Gorsky, L.M. "Sampled-data Control of Double Integrator Systems", *Proceedings of the Southeastern Symposium on Systems Theory*, Morgantown, W. Va., March 2003, Accepted for publication.