Adaptive Embedded Control for a Ball and Plate System

Jun Xiao

Informatics Institute
University of Amsterdam
The Netherlands
Email: J.Xiao@uva.nl

Giorgio Buttazzo

Real-Time Systems Laboratory
Scuola Superiore Sant'Anna
Pisa, Italy
Email: giorgio@sssup.it

***Abstract*—In the application of embedded control systems, timing is an important factor for ensuring a desired system performance. However, the time properties are difficult to manage under the constraints of a given hardware platform and the application tasks have large computation time variations. These require the usage of adaptive mechanisms on several aspects of the control system, from sensory inputs to actuation outputs. Taking a control system consisting of ball and plate device as an example, this paper shows how to manage time and achieve performance improvements by using adaptive strategies in sensing and control. Sensory acquisition, control, and actuation are performed on a STM32F4 microcontroller. A camera mounted on the embedded board is used to detect the ball position. To cope with the limitations of computational speed of the embedded board, an adaptive approach that changes the capture area of the image acquisition process at every execution is proposed for improving the detection accuracy. A system simulator using a TrueTime kernel that provides multitasking environment has been developed as a support tool for designing the controller. Jitter compensation techniques, which adaptively update the control parameters at each sampling instant, are adopted to deal with the performance degradation due to the sampling jitter. Conditions for finding the set of proper control parameters, a practical issue of applying jitter compensation techniques, are also presented. Finally, the system has been implemented and tested on top of the Erika Enterprise real-time kernel.**

Keywords–Adaptive Embedded Control Systems; Camera Detection; Jitter Compensation; PID Control.

I. INTRODUCTION

Balancing is one of most challenging issues in the control field. There are lots of platforms for studying control algorithms for system balancing such as the ball-beam system and the inverted pendulum. Among those, the ball-and-plate system consists in controlling the angular position of a plate with two degrees of freedom (pitch/roll) in order to keep a ball always in the center of the plate in the presence of disturbances. When the ball starts moving, it will roll off the end of the plate if no control action is taken.

The first major challenge is to sense the ball position accurately and in a non-cumbersome, yet inexpensive way. In some implementations, a touch screen is used as the ball position sensor [1], whereas in most cases a Charge-coupled Device (CCD) camera is quite popular for ball detection. Christoph H. Lampert, et al. [2] proposed to overcome the computational bottleneck by making use of the massively parallel architecture of a Graphics Processing Unit (GPU) in a modern computer graphic card. This solution, however, is quite expensive. The second consideration is about the effectiveness of different control algorithms. Nonlinear control methods like Back-stepping control were successfully applied by Lin, et al, in [3] and Ker, et al, in [4]. In recent works [5] [6], intelligent control skills, such as fuzzy logic control, are also studied. However, most of these control methods work only if the sampling period of the control system is small, which strongly requires high speed processing hardware including sensor and microcontroller.

In this paper, sensory acquisition and control of the ball-and-plate system are performed by the STM32F4 microcontroller. A camera mounted on the embedded board behaves as sensor to detect the ball position. Under such hardware constraints, we propose an adaptive detection procedure to cope with the limited capacity of Random-Access Memory (RAM) storage and limited computational speed of the given embedded board. The proposed control strategy includes a jitter compensation technique and is adopted to deal with the performance degradation due to the sampling jitter introduced by the multitasking environment in the embedded board.

The rest of the paper is organised as follows. Section II presents an overview of the system architecture. Section III describes the decision of the sampling period and the adaptive camera detection procedure. Section IV analyzes the Proportional-Integral-Derivative (PID) control algorithm, the effect of the sampling jitter and the jitter compensation techniques. Simulation results and system testing are shown in Section V. Section VI draws the conclusions.

II. SYSTEM DESCRIPTION

The ball-and-plate control system considered in this work consists of two embedded boards: one board (referred to as board A hereafter), equipped with a camera, detects the ball, computes its position, and sends this information to the other board (board B) through a wireless network. Board B receives the data, applies the control algorithm and supplies a proper voltage to the servo motor for actuating the plate. The illustrative overview of the system is shown in Figure 1.

In addition, a system monitor has been developed under the Linux operation system to provide a graphic user interface for displaying the system state. The monitor records the last ten ball positions and plots dynamically the control error with respect to the reference. The monitor also allows users to modify and to update the controller parameters of the system at run time. The monitor communicates with board B through

 ISBN: 978-1-61208-463-3

a serial port. The design and implementation of the Computer-Monitor are out of the scope of this paper.

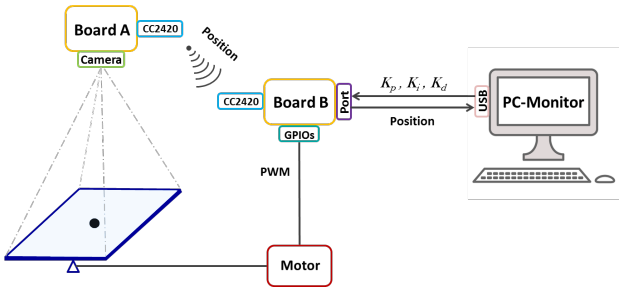

Figure 1. Overview of the ball-and-plate system.

The hardware boards used to control the ball-and-plate system are based on the STM32F4 platform and its expansion boards, CC2420 radio transceivers.They are briefly described in the following paragraph.

STM32F4 discovery board is a microcontroller featuring 32-bit ARM Cortex-M4F core, 1 MB Flash, 192KB RAM. The STM32F4 discovery board has up to 14 timers that can generate the Pulse Width Modulation (PWM) signals to control the servo motors [7].

Three boards extend the Discovery capabilities. The first board is a base board. It provides connectors with General Purpose Input Outputs (GPIOs), connector for camera board and connector for Liquid Crystal Display (LCD) board. The second board is a LCD board. It has a 3"5 display to visualize information such as the ball position and the controller parameters. The last board is a camera board. The resolution of the camera is set to be QQVGA (160×120). It is not possible to obtain an image with a higher resolution otherwise the image to be stored would exceed the RAM size (192KB).

Communication between board A and board B is realized by the CC2420, a true single-chip 2.4 GHz IEEE 802.15.4 compliant transceiver designed for low-power and low-voltage wireless applications. The CC2420 is connected with the base board. The wireless protocol adopts the IEEE 802.15.4 standard. The wireless personal area networks (LR-WPANs) is implemented by using the μ Wireless stack.

III. Detection Procedure

Ball detection is done based on the frame taken by the camera at the beginning of each sampling period. Accurate detection results can be obtained by processing all image pixels, but this comes at a cost of a high computational effort, especially when using a microcontroller with a limited computational speed. The execution time due to image processing determines the minimum sampling period of the control system, hence a compromise has to be found between the limit imposed by the computational speed of the STM32F4 board and the controller requirement of running at a faster sampling rate.

A. Image color data

The frame taken by the camera is expressed internally by a matrix $M_{m \times n}$ containing pixels. The size of the matrix $m \times n$ is determined by the image resolution. The pixels are encoded using a Red-Green-Blue (RGB) 16-bit model. For each element m_{ij} in the M, 5, 6 and 5 bits are used to represent the intensities of red, $green$ and $blue$ component, respectively. Thus, the values m_{red} and m_{blue} representing red and blue intensities are integer numbers ranging from 0 to 2^5, while for green m_{green}, the range is 0-2^6.

B. Computation of threshold

To facilitate the detection procedure, we used a dark ball on a white plate. Hence, the ball is detected by considering the pixels whose color levels are below certain thresholds. Each color component, namely red, $green$, $blue$, has one threshold, donated as THS_{red}, THS_{green} and THS_{blue} respectively. If the values of red, $green$ and $blue$ extracted from one pixel are all smaller than their corresponding thresholds, that pixel is considered to belong to the ball.

Although image thresholding greatly simplifies the detection procedure, the result heavily depends on the light condition of the environment. The darker the environment, the lower the thresholds, and vice versa. In order to make the detection procedure more adaptive, the threshold is automatically computed by the system during an initial calibration procedure. In more details, board A takes a frame of the whole plate with the ball before the start of the system. Then the red intensity m_{red} is extracted from every pixel. For each possible value v (v=0,1,2,...,31), it counts the total number of pixels N_v^{red} whose m_{red} equals v. Finally, the threshold THS_{red} is determined by the following formula:

$$THS_{red} = 0.9 \times Index_{0-14}^{red} + 0.1 \times Index_{14-31}^{red}, \quad (1)$$

where:
$Index_{0-14}^{red} = k$ such that $N_k^{red} = \max N_i^{red}, i = 0, 1, ..., 14$,
$Index_{15-31}^{red} = k$ such that $N_k^{red} = \max N_i^{red}, i = 15, 16, ..., 31$.
The same steps is applied for the calculation of green and blue thresholds THS_{green}, THS_{blue}.

C. Sampling period

By comparing each element in the frame matrix M with the thresholds, we can transfer M to a matrix L containing logical values l_{ij}, where a value of 1 means the pixel corresponds to the ball.

Once the pixels belonging to the ball are identified, a $center\ of\ mass$ [14] algorithm is used to estimate the ball position. Based on the resulting matrix L, it records the coordinates of each black pixel (i, j) and computes the total number of dark pixels N. Then, the ball position is determined by computing the average coordinate of all black pixels. Specifically, a region to be scanned is delimited by the upper left pixel coordinating (x_{min}, y_{min}) and the lower right pixel coordinating (x_{max}, y_{max}). The ball position (X, Y) is computed by the following formulas:

$$N = \sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} l_{ij}, \quad (2)$$

$$X = \frac{\sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} i \times l_{ij}}{N}, \quad (3)$$

$$Y = \frac{\sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} j \times l_{ij}}{N}. \quad (4)$$

 ISBN: 978-1-61208-463-3

A test program was written to estimate the execution time of finding the ball position by scanning every pixel. It takes around 2.4 seconds to finish the computation on the STM32F4 microcontroller. As a sampling period, 2.4 seconds is too long for controlling the ball-and-plate system.

One way to reduce the sampling period is to scan less pixels. Considering the fact that the ball is in a square covered by several pixels, (in general $p \times q$ pixels and 4×4 in our case), one can scan only one pixel among every n_x and n_y ($n_x \leq p, n_y \leq q$) consecutive pixels along the two directions. Thus the total number of points N_{scan} to be scanned is reduced to $\frac{m \times n}{n_x \times n_y}$.

If only one pixel among four consecutive pixels is selected to be scanned, the execution time of detection decreases to around 0.15 seconds. Considering also the time needed for the execution of other tasks in board A, the sampling period of the control system was set at 0.2 seconds to avoid overload during the execution.

D. Adaptive scan

As mentioned above, to reduce the computation time of the scan, the basic detection procedure reads one pixel every four consecutive pixels. This is called constant capture area procedure because the capture area keeps always the whole plate, in other words, x_{min}, y_{min}, x_{max}, y_{max} are constant at each sampling period.

This procedure, however, fails to detect the ball if the scanning points are exactly the vertexes of the square covering the ball, as shown in Figure 2(a), where the full dots stand for the pixels scanned. Therefore, it is risky to adopt this method as it may lose detection in some occasions.

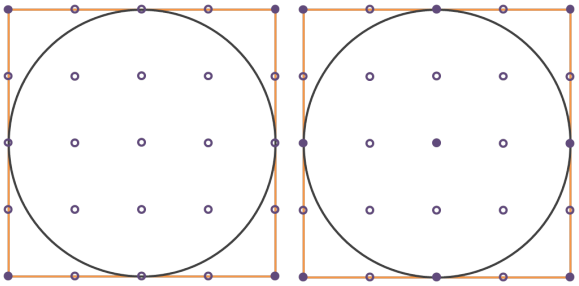

Figure 2. Ball coverage and scanning points.

To improve the detection rate, pixels need to be scanned more intensely without increasing N_{scan}. Noticing that most of computation is wasted for scanning the pixels unrelated to the ball, the idea of $region\ of\ interest$ [14] is used to reduce the computational cost by searching for the ball only in a smaller region of interest, moving with the ball.

Thus, we shrink the capture area for catching the ball at each sampling period. The values representing the upper left (x_{min}, y_{min}) and lower right (x_{max}, y_{max}) corner are updated at each sampling period due to ball movement. Specifically, the next focusing area is a square centered by the last ball position and composed of $\frac{m \times n}{s_x \times s_y}$ pixels, where s_x and s_y (2×2 in our case) are the reduction factors. The matrix size of the focusing area is $\frac{1}{s_x \times s_y}$ of matrix M. Therefore, one pixel among every $\frac{n_x \times n_y}{s_x \times s_y}$ pixels can be scanned without increasing the time for the frame processing, as shown in Figure 2(b).

However, the second solution is based on the assumption that the ball moves slowly such that at the next sampling time, it is always inside the scope of the region of interest. Otherwise, the ball will not be detected.

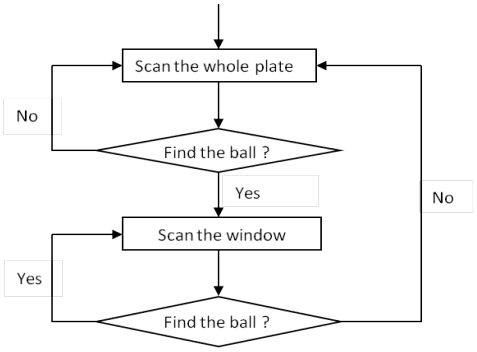

Figure 3. Block diagram for the adaptive detection procedure.

To improve detection, the two solutions are combined to overcome their drawbacks. The adaptive detection procedure is called by adaptive capture window, as shown in Figure 3. The detection starts from executing the procedure in a constant capture area. Once the ball is detected, the detection switches to scan the adaptive window, which is the square centered by the last detected ball position. If the ball moves out of the region of interest, the detection goes back to the procedure of constant capture area.

IV. Control Design

This section describes how the controller has been designed to meet the performance specifications and how the jitter compensation techniques can be applied for improving the control performance to deal with the performance degradation due to the sampling jitter. Considering that the mutual interference between the two actuation axes, it is possible to design two independent controllers for the two (x and y) axes. The following analysis focuses on one direction, but can be applied to both axes.

A. Control performance metrics

The primary criterion for evaluating the performance of control systems is to meet stability requirements and response time specifications. But these criterions do not provide a universal value to judge the control performance. We use the notion of $Quality\ of\ Control$ (QoC), called performance rate function, to evaluate the control performance of closed-loop systems. The QoC in [11] is defined as:

$$QoC(T_0, t) = \frac{1}{IAE(T_0, t)}, \tag{5}$$

where:

- IAE is the integral of the absolute system error, which is the difference between the desired response of the system $y_{ref}(t)$ and its actual response $y_{act}(t)$.

$$IAE = \int_0^\infty |y_{act}(t) - y_{ref}(t)| dt. \tag{6}$$

- $IAE(T_0, t)$ denotes the IAE value obtained by a controller designed with a nominal period T_0, but running with a period t.

B. System model

The full derivation of the system state model is out of scope of this paper; hence, this part only shows the resulting system model.

The linearized relationship between the rotation angle of the motor α and the rotation angle of the plate θ is:

$$\theta = \frac{-68.28(\alpha - 0.49)}{194.28(\alpha - 0,49) - 141.52}. \quad (7)$$

The transfer function of the servo motor having the desired rotation angle α_d as input and the actual rotation angle α_{act} as output is:

$$G(s) = \frac{\alpha_{act}(s)}{\alpha_d(s)} = \frac{1}{0.002s + 1}. \quad (8)$$

The state space form of dynamic model of the ball movement is:

$$\dot{x}_1 = x_2, \quad (9)$$

$$\dot{x}_2 = \frac{3}{5} g \sin\theta, \quad (10)$$

where x_1 represents the ball position of x-axis, x_2 denotes the speed of the ball.

C. Control design

1) PID control law: The PID control [12] algorithm using the closed-loop feedback mechanism is commonly applied in the industrial control systems.

Since the sampling period T is fixed to be $0.2s$, we directly design the PID controller in the discrete time domain. The input to the controller is the ball position error with respect to the reference $e(k)$ at time kT, the output of the controller is the rotation angle of motor $\alpha(k)$ at time kT.

Thus, the discretized form of the PID controller is:

$$\alpha(k) = K_p e(k) + K_i T \sum_{i=0}^{k} e(i) + K_d \frac{e(k) - e(k-1)}{T}. \quad (11)$$

A system simulator is developed to help in controller design. By tuning the three control parameters in the simulator, when $K_p = 10$, $K_i = 0.5$, $K_d = 15$, the system is stable, meeting the control performance.

2) Multitasking and jitter compensation: Nowadays, multitasking is common in real applications. More specifically, in embedded control applications, usually there are multiple tasks executed concurrently in one processor. In particular, in the ball-and-plate implementation, besides the control task, there are other periodic tasks to toggle LEDs to indicate system running state, to print data such as PID parameters on the LCD screen.

At each execution of the discretized PID control task, the output is computed based on the value of three parameters (K_p, K_i, K_d). The three parameters are constant during the execution of each job in a traditional PID control task, assuming that the sampling is performed at equidistant sampling time instants.

However, the assumption is not realistic due to the existence of sampling jitter. In fact, we suppose that there are two additional tasks τ_1 and τ_2 with execution times $5ms$ and $15ms$ and periods $100ms$ and $150ms$ running concurrently with the control task τ_3 including both the work of sensory detection and control algorithm execution in the microcontroller. If the task set composed of τ_1, τ_2, τ_3 is scheduled by Rate Monotonic, the task scheduling of one hyperperiod is illustrated in Figure 4. From Figure 4, it is easy to see that

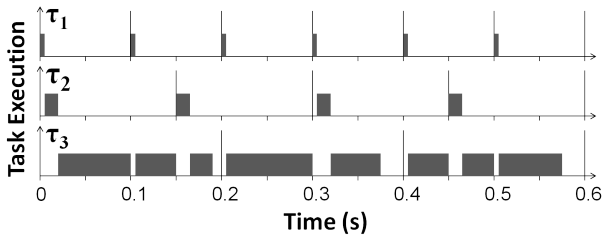

Figure 4. Task scheduling of one hyperperiod by rate monotonic.

the start times $s_{i,k}$ of the k^{th} job in the control task i are not always $nT(n = 0, 1, 2...)$. The difference between the start times (relative to the request times) of two or more instances of a periodic task introduces sampling jitter. The set of all possible sampling jitters, $SJ(\tau_i)$, for task τ_i is defined as [13]:

$$SJ(\tau_i) = \{h_{i,k} | h_{i,k} = s_{i,k+1} - s_{i,k}, k = 0, 1, 2, ...\}. \quad (12)$$

Sampling jitters that will appear at the run time of control task τ_3 can be analyzed offline. The jitters of the first three jobs repeat since the schedule repeats every hyperperiod. Therefore, $SJ(\tau_3) = 0.185, 0.2, 0.215$.

As integral (K_i) and the derivative (K_d) actions depend on the time interval between the last and current sampling time, the actual control performance of the controller running within the multitasking environment is jeopardized by the sampling jitter. The jitter compensation technique updating the PID controller parameters based on the sampling jitter, proposed in [15], can be applied to improve the control performance.

3) Setting the control parameters: To apply jitter compensation, the PID parameters for all possible jitters have to be chosen at the design phase. However, as a practical issue, finding a proper set of parameters is not easy since a bad composition of PID parameters would make the system response even worse.

Two conditions to judge whether the set of PID parameters is proper is based on the performance-rate functions. Specifically, for a control task τ, $\forall T_0 \in SJ(\tau)$, $\forall T_0' \in SJ(\tau)$, and $T_0 \neq T_0'$, if the QoC obtained by running the PID controllers prepared for the compensation satisfies the following two conditions:

- if $T_0 < T_0'$, $QoC(T_0, T_0) > QoC(T_0', T_0')$,
- $QoC(T_0, T_0') < QoC(T_0, T_0)$ and $QoC(T_0', T_0) < QoC(T_0, T_0)$,

then the PID parameters set is proper.

The first condition derives from the fact that smaller sampling periods lead to better QoC. The second item requires the optimal selection of the PID control parameters such that for a specific sampling period, the best QoC is produced only when the actual execution period of the control task equals the designed sampling period.

V. EXPERIMENTAL RESULTS

In this section, we illustrate both simulation results and system testing.

A. Simulation results

1) Simulation model: To carry out the simulation experiments, a system model has been developed in Simulink [10]. Figure 5 is a general view of the whole model. In this model, the block *motor* functioning is defined by (8), which describes the transfer function of a servo motor. The *plate* block represents the plate rotation performed by (7) .The dynamic model (9) and (10) is simulated by the *rolling ball* block.The tasks execution (three periodic tasks including control task τ_3, together with τ_1 and τ_2, discussed in Section IV-C2.) is simulated using the $TureTime\ Kernel$ block, provided by TrueTime [8], which facilitates co-simulation of controller task execution in real-time kernels. The two detection procedures discussed in Section III-D are also implemented.

2) Camera detection: Camera detection results and corresponding system responses results by using the adaptive capture window and the constant capture area are compared in Figure 6 and Figure 7, respectively. In Figure 6, 1 stands for the successful detection, while 0 represents missing detection at the current execution.

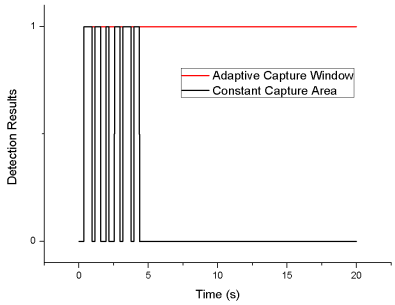

Figure 6. Comparison of the camera detection results using two different detection procedures.

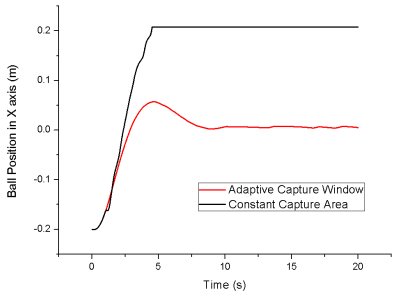

Figure 7. Comparison of system responses using two different detection procedures.

Experiments showed that the system was unstable using the constant sampling area, since the ball was no more detected after few seconds and stayed near the beam. On the other hand, the experiments showed that the system became stable using the adaptive capture window, due to the higher achievable detection rate.

3) Jitter compensation: Following the conditions in IV-C3, the three controllers used for the compensation with sampling periods T_0 equal to $0.185s$, $0.2s$ and $0.215s$ were designed. The values of the compensated PID controller parameters dealing with each possible jitters are reported in Table I. The

TABLE I. COMPENSATED PID CONTROL PARAMETERS.

T_0	0.185	0.2	0.215
K_p	10.8	9.8	10.5
K_i	0.15	0.12	0.1
K_d	23.85	22	23.5

QoC of the three controllers running at different sampling periods are represented in Figure 8.

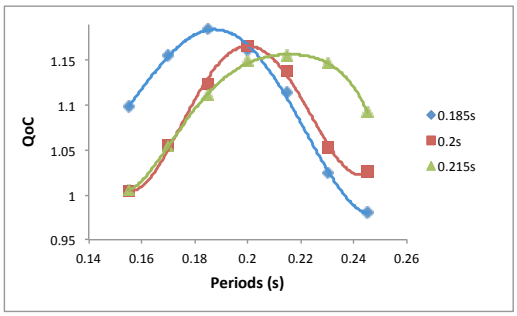

Figure 8. QoC of the three controllers running at different sampling periods.

We compared the system response of PID controllers with and without jitter compensation. Figure 9 shows the jitter degrading effect and the improvement achieved by the compensation. The ideal system response was obtained by running the controller designed with sampling period $T = 0.2s$ without the interference of multitasking. When the PID control task executes at the background of multitasking, the system response of the compensated controller catches more quickly and stays closer to the ideal response with respect to the controller without compensation. Moreover, the system response of the controller without compensation suffers from a larger overshoot and a lower rising speed.

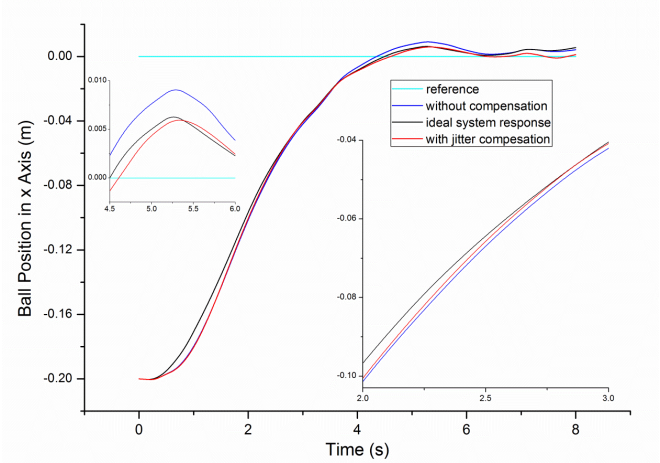

Figure 9. System responses with and without jitter compensation.

B. System testing

Finally, the ball-and-plate system was implemented and tested. The software developed on the two boards runs on

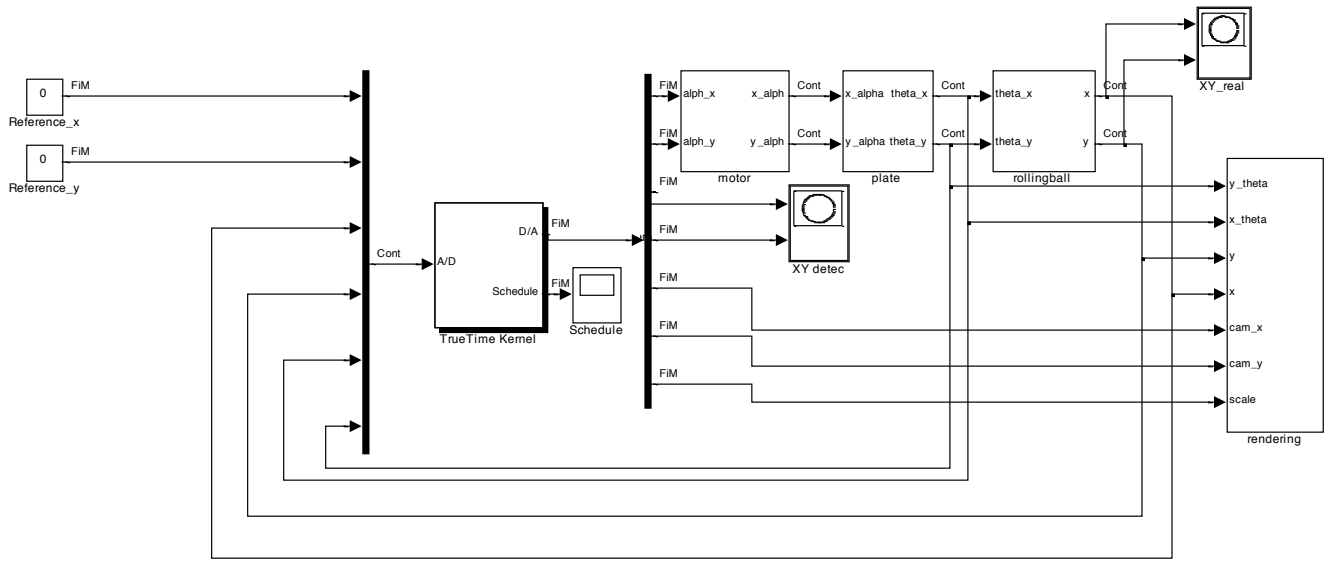

Figure 5. Simulation model.

top of the Erika Enterpise real-time kernel [9]. The desired control performance was obtained by adopting the adaptive mechanisms. Figure 10 illustrates the physical connections and running state of the system.

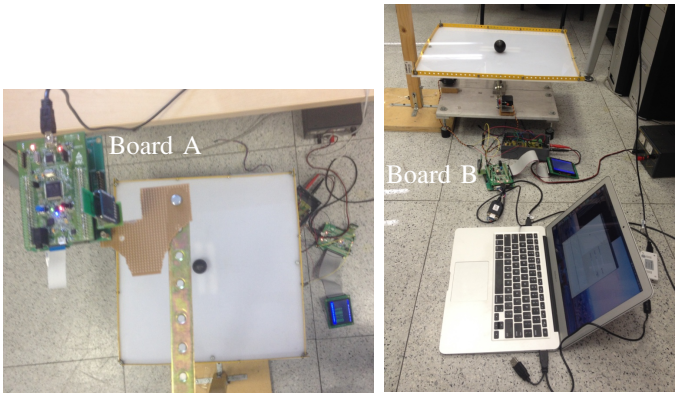

Figure 10. Board A detects the ball position, while Board B controls the motors and communicates with the laptop for monitoring.

VI. Conclusions

This paper illustrated how the usage of adaptive mechanisms in sensing and control activities can improve the system performance in the presence of stringent hardware constraints. To cope with the limitations of computational speed of the embedded board, we presented an adaptive approach that adapts the capture window of the image acquisition process at every execution to improve the detection accuracy. To cope with the performance degradation due to the sampling jitter, jitter compensation techniques adaptively updating the control parameters at each sampling instant proved to be effective in improving control performance. Conditions for finding the set of proper control parameters, a practical issue of applying jitter compensation techniques, are also presented.

As a future work, we plan to investigate the possibility of applying similar adaptive mechanisms to other embedded control applications, especially for those whose system performance is heavily affected by the hardware constraints. Another interesting research direction is to formally prove the correctness of the condition to choose the control parameters for jitter compensation.

References

[1] S. Awtar, et al., “Mechatronic Design of a Ball on Plate Balancing System“, Mechatronics, 12(2), 2002, pp. 217–228.

[2] C. H. Lampert and J. Peters, “Real- time detection of colored objects in multiple camera streams with off-the-shelf hardware components“, Journal of Real-Time Image Processing, 7(1), March 2012, pp. 31–41.

[3] C. E. Lin and C. C. Ker, “Control Implementation of a Magnetic Actuating Ball and Plate System“, International Journal of Applied Electromagnetics and Mechanics, Vol. 27, No. 1-2, 2008, pp. 133-151.

[4] C. C. Ker, C. E. Lin, R. T. Wang, “Tracking and Balance Control of Ball and Plate System“, Journal of the Chinese Institute of Engineering, 30(3), 2007, pp.459-470.

[5] C. H. Lampert and J. Peters, “Image Fuzzy Control on Magnetic Suspension Ball and Plate System“, International Journal of Automation and Control Engineering, 3(2), May 2012, pp. 35-47.

[6] M. A. Moreno-Armendariz, C. A. Perez-Olvera, F. O. Rodriguez, E. Rubio, “Indirect Hierarchical FCMAC Control for the Ball and Plate System“, Neurocomputing, Vol. 73, No. 13-15, 2010, pp. 2454-2463

[7] STMicroelectronics, “http://www.st.com“.

[8] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, K. E. Arzen, “How Does Control Timing Affect Performance? Analysis and Simulation of Timing Using Jitterbug and TrueTime“, IEEE Control Systems Magazine, 23(3): June 2003 pp. 16–30.

[9] Evidence srl Pisa, “http://evidence.eu.co“.

[10] The mathworks,“Simulink User’s Guide“ Nattick, MA USA, 2000.

[11] G. Buttazzo, P. Marti, M. Velasco, “Quality-of-Control Management in Overloaded Real-Time Systems“, IEEE Transactions on Computers, vol. 56, no. 2, February 2007, pp. 253–266.

[12] K.J. Astrom and B. Wittenmark, “Computer-controlled systems“ Third Edition. Prentice-Hall, 1997.

[13] G. Buttazzo, “Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications“ Third Edition, Springer, 2011.

[14] T. Bradski and A. Kaehler “Learning OpenCV“ First Edition O’Reilly 2008.

[15] P. Marti, G. Fohler, K. Ramamritham, J. M. Fuertes, “Jitter Compensation for Real-Time Control Systems“, Proc. 22nd IEEE Real-Time System Symp., Dec. 2001, pp. 39-48.

 ISBN: 978-1-61208-463-3