Embedded Control System Design for Inverted Pendulum Type Mobile Robots Based on High-Level Petri Nets

Gen'ichi Yasuda
Nagasaki Institute of Applied Science, Japan

INTRODUCTION

In recent years, functionality and complexity of real-time controllers dramatically increased, beyond the complexity of hardware components. Due to the growing complexity of the software components in advanced multiprocessor-based robot controllers, including timing requirements of run-time elaboration of information from multiple external sensors, the need of early validation of the design of control software is increasing (David & Alla, 1994; Hruz & Zhou, 2007). Usually, control laws for non-linear physical systems like robotic vehicles and arms are specified in continuous time and allows for checking mainly qualitative properties like stability. At run time, these control laws are integrated and implemented as multi-tasks programs controlled by a real-time operating system on a single or multiprocessor target. The performance indices depend on the actual implementation, including sampling rates and computing latencies, besides the algorithm in use. Conventional research on real-time operating systems for complex robotic system control do not provide tools to analyze or synthesize such sampled control laws with respect to performance indices, measuring the impact of the organization of the control system on the overall controlled process. For the design and implementation of the overall software of real-time robotic control systems, while traditional techniques, such as finite state automaton, offer limited support for validating control algorithm design, the use of formal methods can increase the possibility of validating design.

This chapter presents a formal approach to designing software for sensor based non-linear motion control with a backstepping method (Khalil, 2002), focusing on the design of control software for non-holonomic two-wheeled inverted pendulum mobile robots (Nomura, 2009). Because the hardware of the control system exhibits subsystems with different dynamics, for example, feedforward paths to update some parameters and measurements to come from sensors of different kind running at different rates, control algorithms are decomposed into groups of cooperating tasks; where these tasks with different executing durations run in sequence or in parallel. Multi-task programs with multi-rate controllers permit modular programming and software reusability. The feature can be useful to optimize computing resources, because both onboard space and energy are strongly limited, especially for autonomous robots working in unstructured or critical environments.

BACKGROUND

To perform complex control software using multiple sensors and effectors, single or multiple computer control systems are essentially employed, where the overall management of a computer system is under

DOI: 10.4018/978-1-7998-3479-3.ch004

control of a multitasking operating system master program, which is referred to as a supervisor, monitor, or kernel. The master program is a collection of programs to schedule the jobs, to manage the hardware resources, and to provide a more hospitable interface to the user programs. So centralized operating systems have become so essential to efficient computer operation, viewed as inseparable from the hardware.

The simple Petri nets associate events with transitions and conditions with places. Places and transitions are linked by directed edges called arcs. Since the presence of a token in a place indicates that the corresponding condition is true, the marking describes the state of the Petri net in terms of conditions which are true and those which are false. When each of the input places becomes marked, i.e. has a token, the transition fires, i.e. the event occurs, and the firing removes a token from each of the input places, making the preconditions false, and deposits a token in each of the output places, making the postconditions true.

The Petri net is gaining increasing importance for the discrete event modeling and analysis of robotic control systems, since it offers a convenient way of expressing system behavior which is both parallel and asynchronous, and therefore distributed (Wang & Sarides, 1993; Yasuda, 2014; 2015a). In addition to the precedence constraints among computing modules, including sequences of modules and the repetition of certain modules, looser couplings associated with shared resources, such as memory and devices, can also be directly expressed. Besides modeling capabilities, the Petri net can be analyzed in a formal way to obtain information about the dynamic behavior of the modeled system (Du et al., 2013).

DISTRIBUTED CONTROLLER DESIGN BASED ON BACKSTEPPING APPROACH

A wheeled inverted pendulum mobile robot is a self-balancing, traveling vehicle with two wheels attached on the sides of its body. Two motors are attached between the wheels and the body to control traveling speed and direction, besides its own posture. Both the traveling motion and the self-balancing are achieved by independent actuators providing the necessary torques to the wheels (Fierro & Lewis, 1997; Nomura, 2009; Takei et al., 2009). The complete dynamics of the traveling motion consists of the steering kinematics and the vehicle dynamics. The nonholonomic constraint states that the vehicle can only move in the direction normal to the axis of the driving wheels, satisfying the conditions of pure rolling and non-slipping. For traveling control, divided into tracking a reference trajectory, following a path, and point stabilization, the software generates robot trajectories based on on-line elaboration of sensors information and manages kinematic inversion. According to a backstepping approach, the main idea behind nonlinear feedback control for solving these problems is to define velocity control inputs based on the perfect velocity tracking assumption. According to a backstepping approach, nonlinear feedback control for solving these problems velocity control inputs are defined based on the perfect velocity tracking assumption. So, velocity control inputs should be converted into torques, taking into account the actual vehicle dynamics.

For balancing control, the posture angle x_1 and its derivative $x_2 = \dot{x}_1$ are used as the state variables, where the posture angle is 0 when the center of gravity of the robot is right above the contact point of a wheel to the ground. The reference values of the posture angle and the angular velocity are zero. The state-space equations of balancing control are generally represented as follows:

Related Content