Research Proposal: Computational resource management of multi channel controller

Hodai Goldman (Hodaig@cs.bgu.ac.il)

Department of Computer Science,
Ben-Gurion University of the Negev, Beer Sheva, Israel

Supervisor: Dr. Gera Weiss

August 14, 2016

1 Background and Problem Formulation

Today's computer power allows for consolidation of controllers where a single computer can regulate many control loops, each with its varying needs of computation resources. This brings two research challenges that we intend to attack in the proposed thesis:

- How to schedule control tasks in order to achieve good performance in terms of control measures (overshoot, convergence speed, etc.)?
- What is a good interface for co-design of scheduling and control?

While it is possible to build control systems using standard operating systems, either real-time or desktop with static or with dynamic scheduling schemes, there is an agreed opinion in the control community that these do not serve well for the purpose outlined above [**?**]. Specifically, desktop type operating systems (Windows, Linux, etc.) schedule for computational efficiency, but do not allow for control performance guarantees of the individual loops. On the other hand, real-time operating systems sacrifice some efficiency for timing predictability, but they do relate timing information with control performance. When using such operating systems for control engineers usually apply controllers that work in a fixed periodic manner so that control behavior becomes deterministic and control performance can be guaranteed. This is not efficient because resources can be better utilized if controllers act at higher frequencies when only when needed. In this work we will develop methods to combine the efficiency of desktop operating systems with the predictability of real-time operating systems in a way that is more suitable for control systems then periods and deadlines

The control loops that we are analyzing are here of the form shown in Figure 1. A physical plant (controlled system) is connected via sensors and actuators...

The current state of the art is that control engineers design control tasks as periodic computations then they specify the required periodic frequency for the task and software engineers design a scheduler that ensures that the periodic frequency requirements are met usually using pre-computed knowledge of the expected (maximum) duration of the tasks. We claim that for control systems we can achieve better performance by using richer and more flexible set of requirements for the tasks. Specifically, we will develop tools with which the control engineers can specify in a natural way features of their control loop that the scheduler will use to allow for dynamic schedules that guarantee required control performance.

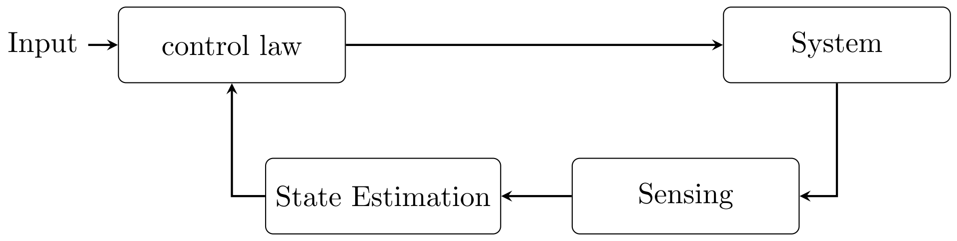

Figure 1: A typical control loop system.

2 Case study: Vision based controllers for drones

To test our concepts, we will examine their implementation in the context of a specific case study. Specifically, we will implement a quad-rotor that flies in corridors and greenhouses by a vision based feedback. The challenge in this case study, from our perspective, is that image processing is a heavy computational task that requires careful scheduling in order to preserve the system predictability and stability.

In Section 3 we suggest a novel framework model for such systems that we will implement in this thesis.

2.1 Nano-Satellite

Another interesting case study is the controlling of nano-satellite, in particular scheduling the sensing tasks of *IAI*[1] nano satellites. Their Satellites are controlled by very small and slower processor, and the developers say that the main issue in programing the satellite is how to schedule the sensing tasks. The common approach they are using is to periodically schedule all the sensing task every iteration, and they say that this is the main computational consumption and is too much for their slow processor.

We think that lots of the sensing are unnecessary and the controlling task can be achieved without "most up-to-date" information, some times we can base on good estimation of the sensing value or even use the last value. Although Vision based drones is our main case study, we will also check the possibility to improve the nano-satellite issue with our framework.

[1]IAI: Israel Aerospace Industries

3 Research Plan

Our research plan consists of: (1) Designing a methodology for effective allocation of computational resources in real-time control systems; and then (2) Demonstrating our methodology with a framework for developing such systems and with a case study. The details of these two steps are elaborated below.

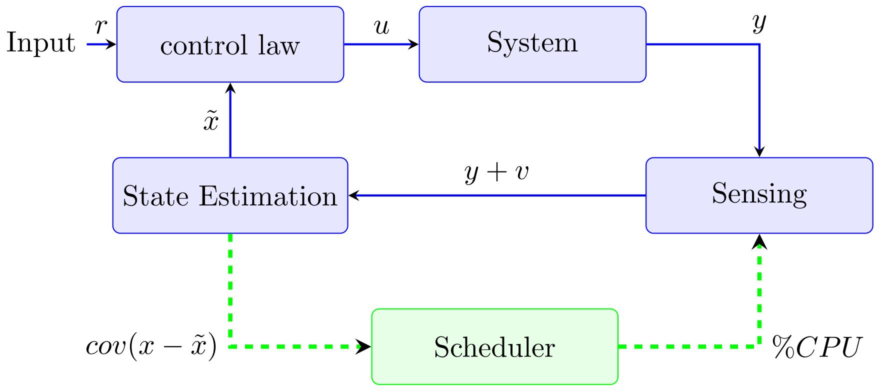

Figure 2: Our proposal of a general controller framework, each control loop (blue loop) will tell the resource allocator (*scheduler*) his Level of certainty and the allocator will allocate the CPU time among all the control loops corresponding to the loops certainty.

3.1 The proposed methodology

The general methodology that we will develop is illustrated if Figure 2. That methodology is based on modern controller architecture where the system has multiple controlling tasks (control loops) and a scheduler that schedule the task as they demand. In our methodology, shown in Figure 2, there is a strong relation between the *Scheduler* and the control loops. We suggest that each control loop (blue components) will tell the resource allocator (*scheduler*) his Level of certainty and the allocator will allocate the CPU time among all the control loops corresponding to the loops certainty in order to maintain some pre-defined specifications of state certainty or stability.

Our proposal is general and may be applicable in a wide range of applications. However, in this initial phase of the research, we believe that it is better to focus on a specific sub-domain and solve all technical issues in order to prof the concept. In this thesis we will develop and implement the

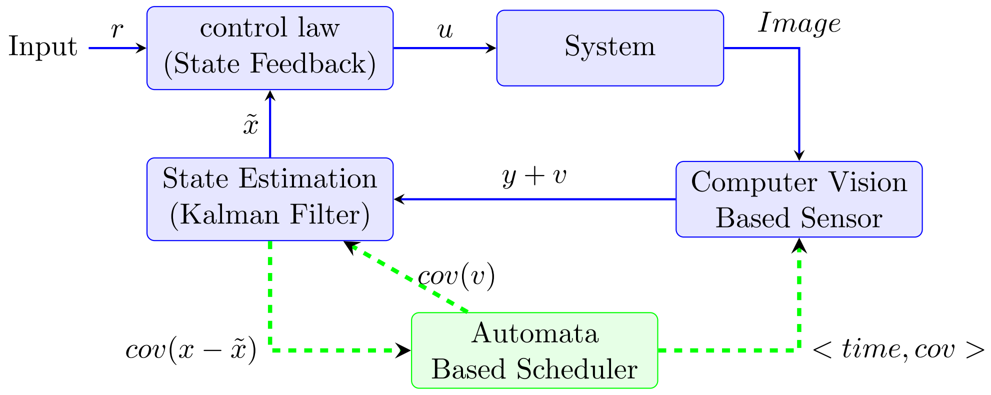

Figure 3: The controller framework we will implement, the *scheduler* will allocate CPU time ($< time, cov >$) for the $ComputerVision$ task base on the state estimation certainty using guarded Automata.

framework shown in Figure 3, a vision based controller for drone,

As shown in Figure 3 the suggested control system framework have strong relation between the tasks scheduler to the control loops, in this case the estimation task (*State Estimation*) accuracy is strongly depended on the sensing task ($Computer\ Vision$) and there both collaborate with the *scheduler* in order to achieve they **control objectives** (g.e. stability). In this framework the scheduler is part of the control logic and therefore it can make scheduling decisions based on the current control state, the scheduling of $Computer\ Vision$ task is depends on the accuracy of state estimation. For example if the vision is clear the $Computer\ Vision$ will produce good measurement of the *System* and therefore good accuracy will be achieved so the *Scheduler* can allocate less computation time to the heavy $Computer\ Vision$ task and still remain stable and allocate more CPU to others control loops or some background tasks (like navigation).

The above collaboration requires to re-adjust some parts of the controlling system, e.g. the *State Estimator* needs to work with variable error covariance of $Computer\ Vision$ measurement and the $Computer\ Vision$ needs to be able run within variable time limits, and also the tasks predefined requirements need to be re-adjust in order to define the relation between tasks, like $Computer\ Vision$ and $State\ Estimator$. Below we will dive in each part of the above system (Figure 3) and explain how it will be adjusted (changed) and how we will achieve that adjustment in our demonstration.

Finally, we need to re-consider the system Analyzing technique because variable sensing characteristics ($Computer\ Vision$ time limits) will lead to variable estimation characteristics...

3.1.1 Sensors ($Computer\ Vision\ Based\ Sensor$)

$Computer\ Vision\ Based\ Sensor$ is the module that is responsible for taking picture or series of pictures and produce the goal measurement. In our research we will use the $Computer\ Vision$ in order to detect two dimensional movement in the camera surface (i.e. the drone speed). There are many such algorithms (optical flow), they differ by their running time and by the accuracy of the solution, mostly more time lead to more accuracy.

In our case we need to be able to control the $Computer\ Vision$ running time and to have some god knowledge of the solution accuracy in order to achieve optimal state estimation (see Section 3.1.2). We assume that the $Computer\ Vision$ error is spread normally, and we will use the error covariance as the solution accuracy. This is acceptable assumption because it is the sum of many independent random variables, as follows. Optic flow algorithms usually go by identifying similar regions in constitutive pictures and then averaging the distances that each feature "traveled". Assuming that the error in measurement of each feature is independent of the other errors, we get that the total error is the average of independent random variables. We will validate this assumption by experimentation with different parameters of different algorithms.

In order to control the running time, we will use a "contract based" vision algorithms show by Pant [4]. This type of algorithms will run until we stop them, and when we stop them they will provide a solution with accuracy that is proportional to the amount of time he was running, that way we can control the solution accuracy by controlling the running time. In our implementation, in order to lower the complexity, we will pre-define few different "operation modes" of the $Computer\ Vision$ task, that differ by their running time and they are identified by a pair $< RunTime, Covariance >$ where $Covariance$ is the error covariance of running time $RunTime$.

3.1.2 State Estimator

3.1.3 Control Tasks

The control task himself ($ControlLaw$) is responsible to reduce the difference between current state (x) where $\tilde{x} = x +$ "$estimation\ error$" (in Figure 2) and target state (u in Figure 2), by changing the drone motors speed.

This task is usually a very low CPU consumer and been well studied [**?**], hence, in this thesis we will use a well known technique from control theory called **PID**[2]. In this technique the control output is based on the physical knowledge of the system dynamics, and have 3 variable parameters (P, I and D) that define the controller convergence behavior. This is also the technique used in the APM software (Section 3.2.2) that we will use and we will not change it.

3.1.4 Automata Based Scheduler

3.2 Experimental environment

3.2.1 Drones

3.2.2 Ardu-Pilot-Mega as base controller software

4 Preliminary Results

Blablabla said Nobody [2]. Blablabla said Nobody [3]. Blablabla said Nobody [1].

References

[1] Rajeev Alur and Gera Weiss. Rtcomposer: A framework for real-time components with scheduling interfaces. *8th ACM/IEEE Conference on Embedded Software*, 2008.

[2] Merav Bukra. GameComposer: A Framework for Dynamic Scheduling. Master's thesis, Ben-Gurion University, Israel, October 2014.

[3] Ardupilot Developers. ArduPilot Mega (APM): open source controller for quad-copters. `www.arducopter.co.uk`.

[4] Yash Vardhan Pant, Kartik Mohta, Houssam Abbas, Truong X. Nghiem, Joseph Devietti, and Rahul Mangharam. Co-design of anytime computation and robust control (supplemental). Technical Report UPenn-ESE-15-324, Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, May 2015.

[2]PID controller - A proportional-integral-derivative controller