Reactive Scheduling of Computational Resources in Control Systems

Gera Weiss
Ben-Gurion University
Beer-Sheva, Israel
geraw@cs.bgu.ac.il

Hodai Goldman
Ben-Gurion University
Beer-Sheva, Israel
hodaig@cs.bgu.ac.il

ABSTRACT

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

- How to schedule control tasks in order to achieve good performance in terms of control measures (overshoot, convergence time, etc.), within the still limited computation resources?
- What is a good interface for co-design of scheduling and control?

Desktop type operating systems, like Windows and Linux, schedule for computational efficiency but do not allow for worst-case performance guarantees. Real-time operating systems, on the other hand, sacrifice some efficiency for timing predictability, but the type of timing guarantees that such systems provide usually are not directly guaranteeing the control performance of the system. When using such operating systems for control, engineers usually apply controllers that work in a fixed periodic manner witch the 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 only when needed.

In this work we show how to combine the efficiency of dynamic scheduling with the predictability of real-time scheduling, in a way that is more suitable for control systems then periods and deadlines. We show that applying control computations at dynamically adjustable periodic, and with variable computational demands, based on real-time information, allows for better utilization of the computational resources and therefor better control performance.

1. INTRODUCTION

Sections go here.

2. ARCHITECTURE: AUTOMATA BASED SCHEDULER

Sections go here.

3. SIMULATIONS

4. APPLICATION TO AUTONOMOUS QUADROTOR FLYING IN-DOOR

TODO - The quadrotor basics

The specific case study we used to test our concept is the ability of fly thru windows, this task require a very stable hovering in front of the window. We implement Autonomous controller for that task, and we evaluate the performance in terms of the horizontal linear axis parallel to the window surface, as this is the most critical axis in this task, this axis noted by x see Figure **??**.

The drone consist of Raspberry pi with navio [**?**, **?**] and is controlled by modified ArduPilot [**?**]. We implement standard PID [**?**] controller that regulate the position and attitude of the drone relatively to the window. We used inertial sensors (Gyroscope and Accelerometer) for attitude control relative to earth, and we use camera [**?**] and simple image processing for position and attitude relatively to the window **??**. The image processing have few different operation modes which have different image resolution, better resolution image produce more accurate measurement but consume more processing time, changing operation modes allows us to control the trade off between mesurment quality and processing time as shown in Table **??**, for simplicity we examine only two modes, *High quality* with resolution of 960p and *Low quality* with resolution of 240p.

As shown in the simulation at Section 3 we should use Kalman filter estimator, in our case we have non-linear system... For simplicity, inspiring from Kalman filter [**?**], we use two steps estimator that *predict* the current state evolved from the previous state using the system model ($x_{k|\hat{k}-1}$) and then *update* the prediction with current state measurement from image (z_k). The result estimation ($x_{\hat{k}|k}$) is weighted average: $x_{\hat{k}|k} = Kx_{k|\hat{k}-1} + (1-K)z_k$, the K gain is defined different for each resolution for achieving estimation in each resolution.

4.1 specification automatons

The performances of the system is measured by

TODO - only High vs only Low (with performances results)

TODO - error ($y_{k|k}$) derived schedulers

TODO - position ($x_{k|k}$) derived schedulers

TODO - complex / agregated schedulers

TODO - gyro derived??

4.1.1 results

TODO - graphs and tables

5. CONCLUSIONS

Conclusion goes here.

Acknowledgments

Acknowledgement goes here.

6. REFERENCES

APPENDIX

Appendix goes here.