Stabilisation of a Pan-and-Tilt Unit holding a camera

Final Report for CS39440 Major Project

Author: Edgar Ivanov (edi@aber.ac.uk)
Supervisor: Fred Labrosse (ffl@aber.ac.uk)

April 25, 2014
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Mobile And Wearable Computing (G421)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature ....

Date ....

Abstract

Computer Science department at Aberystwyth University has a large all-terrain rover that is equipped with a panoramic camera mounted on a Pan-and-Tilt Unit (PTU) that is stabilised using gyroscopes. One of the problems of the gyroscopes is that they tend to drift and therefore need to be regularly corrected to ensure the camera stays vertical. The project is about building a small hardware module and the software to drive it that will interface tilt sensor (inclinometer) with the PTU to ensure camera ”verticality”.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

This section should discuss your preparation for the project, including background reading, your analysis of the problem and the process or method you have followed to help structure your work. It is likely that you will reuse part of your outline project specification, but at this point in the project you should have more to talk about.

I was interested in getting experience with assembling and programming real world sensors and mini computers where low level programming would be involved. This project attracted my attention because is was solving real world problem, having a real customer and contained both hardware and software parts.

1.1 Problem Definition

Pan-and-Tilt Unit (PTU), which is mounted on the Idris electric vehicle, is used to hold a panoramic camera. This vehicle is used for the research and mainly drives over the curved surfaces. PTU that holds a camera has a built in functionality to stabilize at the chosen position, so that if rover drives up the hill camera stays vertical. To achieve camera verticality rover should be standing on the flat surface when stabilization command issued, otherwise it may end up stabilizing at the wrong angle. To ensure stability of the camera PTU uses gyroscopes. The well known problem of the gyroscopes is that their readings are affected by the different air temperature, magnetic effects, friction etc. [1].

This feature together with the data from the inclinometer about the current vehicle chassis position (inclination) can be used to ensure camera verticality when rover goes up or down the hill.

The aim of the project is to build an additional unit which will ensure that camera mounted on the rover is always in a vertical position.

I will be building an additional unit which will acquire current position using the more accurate inclinometer and then will provide the PTU with the calibration data. Essential my task is to write a program which will interact with the PTU and inclinometer. The program will fetch data from both devices and compare it. Based on the difference between PTU and inclinometer information it will calculate by how much degrees PTU should be adjusted and then will send this calibration data to the PTU. At the end of the project there should be a simple interface that allows to perform calibration by just sending one command from the unit that controls the rover.

1.2 Background

What was your background preparation for the project? What similar systems did you assess? What was your motivation and interest in this project?

1.3 Analysis

Taking into account the problem and what you learned from the background work, what was your analysis of the problem? How did your analysis help to decompose the problem into the main tasks that you would undertake? Were there alternative approaches? Why did you choose one approach compared to the alternatives?

There should be a clear statement of the objectives of the work, which you will evaluate at the end of the work.

In most cases, the agreed objectives or requirements will be the result of a compromise between what would ideally have been produced and what was felt to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.

1.4 Process

You need to describe briefly the life cycle model or research method that you used. You do not need to write about all of the different process models that you are aware of. Focus on the process model that you have used. It is possible that you needed to adapt an existing process model to suit your project; clearly identify what you used and how you adapted it for your needs.

Chapter 2

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

2.1 Overall Architecture

2.2 Some detailed design

2.2.1 Even more detail

2.3 User Interface

2.4 Other relevant sections

Chapter 3

Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on real users? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

Appendices

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the clients existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

Appendix B

Code samples

2.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [?].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*---------------------------------------------------*/
  /* Minimum Standard Random Number Generator          */
  /* Taken from Numerical recipies in C                */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity     */
  /* Always call with negative number to initialise    */
  /*---------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
```

```
static long iy=0;
static long iv[NTAB];
double temp;

if (*idum <=0)
{
  if (-(*idum) < 1)
  {
    *idum = 1;
  }else
  {
    *idum = -(*idum);
  }
  idum2=(*idum);
  for (j=NTAB+7;j>=0;j--)
  {
    k = (*idum)/IQ1;
    *idum = IA1 *(*idum-k*IQ1) - IR1*k;
    if (*idum < 0)
    {
      *idum += IM1;
    }
    if (j < NTAB)
    {
      iv[j] = *idum;
    }
  }
  iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
  *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
  idum2 += IM2;
}
j = iy/NDIV;
iy=iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
{
  iy += IMM1;
}
```

```
    if ((temp=AM*iy) > RNMX)
    {
      return RNMX;
    }else
    {
      return temp;
    }
  }
```

Annotated Bibliography

[1] Jacob Fraden, *Handbook of Modern Sensors*. Springer, 2010, p. 680. [Online]. Available: http://www.plentyofebooks.net/2011/05/download-handbook-of-modern-sensors-4th.html 1441964657.

This book gave me understanding of why different sensors a not perfect and may have problems with accuracy. On 339 p. it covers gyroscope architecture and in particular discusses factors that affect their accuracy.

[2] Y. Kreinin, "How to mix C and C++." [Online]. Available: http://yosefk.com/c++fqa/mixing.html

[3] Murata Electronics, "SCA121T dual axis inclinometer modules data sheet." [Online]. Available: http://www.muratamems.fi/sites/default/files/documents/sca121t_inclinometer_datasheet_82127400a2.pdf

This data sheet describes inclinometers features and characteristics. I was particularly interested in understanding the measuring directions.

[4] NXP Semiconductor, "I2C-bus specification and user manual," 2012. [Online]. Available: http://www.nxp.com/documents/other/UM10204_v5.pdf

This user manual gives in depth description of I2C bus; I was interested in analysing the fundamental architecture of this bus as well as reading about SDA and SCL signals

[5] Sagebrush Technology Inc, "Command Set Documentation, Pan-Tilt Gimbals,Servomotor Version," 2004.

This command set documentation defines a standard for the control of the PTU. I am interested in the contents of the tables with information about the error codes and general device control commands.

[6] ——, "Model 20 Pan-Tilt Gimbal User Manual," 2005.

This manual describes system set-up information and identifies the parameters necessary to operate a Model 20 Servo unit. In particular it describes available TASS stabilization commands.

[7] SensorWiki.org, "Gyroscope." [Online]. Available: http://sensorwiki.org/doku.php/sensors/gyroscope