Computer Science Senior Software Engineering Project (CS462)
Winter 2017

HEAD-UP DISPLAY ALIGNMENT SYSTEM

TECHNOLOGY REVIEW

VERSION: 2.0

Group 65:
KRISNA IRAWAN
JIONGCHENG LUO
DREW HAMM

Sponsor:
ROCKWELL COLLINS, INC.

FEBRUARY 17, 2017

Abstract

This project is a proof concept to explore a potential technological innovation for Head-Up Display (HUD) system that present critical flight information to pilots. The primary objective of this project is to reduce the cost and time required to precisely align flight information to the HUD by introducing additional sensor to the system to make the alignment process more dynamic. To achieve this goal, there are eight different main technologies that will be critical for the development of the project. This document will compare three alternative options for each main technologies. This document will also include the option that we choose for each main technologies to develop this project.

Contents

I. Introduction

There are eight different main technologies that will be used or involved in this project. These technologies contain using of both software and hardware. For instance, for hardware, we will compare and discuss about some possible hardware options for building the demonstration system; for software, we will compare and discuss different algorithms or approaches for resolving the problem. Each of us takes an authorship (responsibility) for three technologies:

Jiongcheng (Roger):

- Hardware selection of Microcontroller
- Hardware selection of Represented MEMS IMU
- Hardware selection of Communication Protocol Type

Krisna:

- User Interface Toolkit
- Programming Language
- Statistical Analysis Method

Drew:

- IRU Data Representation
- Filter Technique

II. Technologies

A. Hardware selection of Microcontroller

1) Context: It is critical to choose the most appropriate microcontroller for this system since there are limitations from both hardware and software perspectives. A microcontroller plays an important role in the system, which it will have following functionalities in the system:

- Process input from the IMUs and output to the computer/display
- Process the alignment algorithm
- Debugging and testing purpose

2) Options: We consider to use a board between *Adafruit Pro Mini* [1], *Metro Mini 328* [2] and *InvenSense MPU-9250 CA-SDK Reference Board* [3]. Anyone of these board has its outstanding areas and lack, and we will choose one of them based on the following criteria.

3) Criteria:

1) **Support Language**: We expect to use a general programming language for the microcontroller since using familiar programming language will reduce our time on additional research and the save time on working on the algorithm itself.
2) **Clock Speed**: The alignment algorithm is required to within around 500 milliseconds time from taking the input from the IRUs to the output of aligned data, that require the board has a fast speed and running any processes.
3) **I2C Protocol**: This is necessary to have on the board since the MPU-9250 (our selected IMU) require I2C protocol to communicate.
4) **Connection with PC**: This is necessary to have on the board since we will use computer to any software program.
5) **Size**: Size is less important since this system is for demonstration purpose.
6) **Cost**: This is relatively important but as long as the price is within the expected budget, that will be acceptable.

4) Table of Detailed Comparison:

Model	*Arduino Pro Mini*	*Metro Mini 328*	*InvenseSense MPU-9250 CA-SDK Reference Board*
Core Chip	ATmega328	ATmega328	Texas Instrument MSP430
Support Language	C	C	C/C++
Clock Speed	8MHz (3.3V mode)	16MHz (3V mode)	16MHZ (1.8V 3.6V)
I2C Protocol/Number	Yes/1	Yes/1	Embedded Structure
Connection with PC	USB	FTDI/USB	USB/Bluetooth
Size	33mm x 18mm	18mm x 44mm x 4mm	Unknown
Cost (U.S Dollar)	$9.95	$12.50	$440.00
Advantage	Small size	Faster clock speed	With embedded IMU
Shortage	Slower Clock Speed	Lack of resource of guidance/datasheet	High cost

5) Overall Discussion: By comparing these boards by the above criteria, *Arduino Pro Mini* and *Metro Mini 328* have similar performance and specification, they both have the advantages of small size and low cost. *InvenseSense MPU-9250 CA-SDK Reference Board* is a special case, which its a board as well as the IRU itself since it has embedded on-board sensors. *MPU-9250 CA-SDK is very powerful* for multi-sensor system, other than an embedded MPU-9250 (accelerometer, gyroscope and compass), it also has pressure sensor, UV sensor, humidity and temperature sensor, light and proximity sensor. However, high cost is a noticeable shortage of *MPU-9250 CA-SDK*. In addition, and not all of its functionalities are necessary for this project. Now, we can only look at *Metro Mini 328* and *Arduino Pro Mini*. Based on the comparison and the criteria, *Metro Mini 328* will be more preferable. Because this module has a faster clock speed compares to the *Arduino Pro Mini*, although it has a larger size but this doesnt affect to its actual performance based on our project requirement, other than that, all other concerned criteria are same as the *Arduino Pro Mini*.

B. Hardware selection of Represented MEMS IMU

1) Context: An Inertial Measurement Unit (IMU) is the most significant part of the hardware components of this system. An MEMS IMU will be used to measure the acceleration, velocity position of the aircraft and output data for alignment algorithm. Therefore, we are looking for a model of IMU that is highly accurate, well performed as well as with affordable expense for the demonstration of this project, following chart compare these three model with a list of comparison.

2) Options: We are looking at three models that are considered to be the most likely options for representing MEMS IRU. *9DoF Sensor Stick* [4], *MPU-9250 IMU* [5] and *9DoF IMU* [6] are three models of IMU. All these three IMUs are with 9 degrees of freedom/9 axis sensor, that indicates all of these IMUs contain MEMS sensors of accelerometer, gyroscope and magnetometer(compass), which is a basic requirement for choosing the IMU in this project. Following are some specific criteria that we look and compare for choosing the most proper MEMS IRU.

3) Criteria:

1) **Operating voltage range**: This is determines based on the model we use for the microcontroller, which an operable voltage range of an IMU should be less than the output voltage of its microcontroller.
2) **Support I2C protocol**: This is significant since I2C could be the only protocol for an IMU to communicate with the microcontroller.
3) **Accelerometer output resolution**: This will determine the accuracy of the acceleration data output.
4) **Gyroscope output resolution**: This will determine the accuracy of the angular velocity data output.
5) **Magnetometer output resolution**: This will determine the accuracy of the orientation data output.
6) **Cost**: This is relatively important but as long as the price is within the expected budget, that will be acceptable.

4) Table of Detailed Comparison:

Model	*9DoF Sensor Stick*	*MPU-9250 IMU*	*9DoF IMU*
MEMS Sensors	Accel.: ADXL345 Gyro: ITG-3200 Magn.: HMC5883L	MPU-9250	LSM9DS1
Operating Voltage Range	2.1 — 3.6V	2.4 — 3.6V	1.9 — 3.6V
Output Type	Unknown	16 bits ADC	16 bits ADC
Support I2C Protocol	2g, 4g, 8g, 16g	2g, 4g, 8g, 16g	2g, 4g, 8g, 16g
Gyroscope Output Resolution (Angular Velocity)	Full scale = 2000 degree/s	250, 500, 1000, 2000 degree/s	250, 500, 1000, 2000 degree/s
Magnetometer Output Resolution (gauss)	8 gausses	48 gausses	4, 8, 12, 16 gausses
Size	22.22 X 18.48 mm	Unknown	14.3 mm X 20.5 mm
Cost (U.S. Dollar)	$49.95	$14.95	$24.95
Advantage	Higher output resolution	Least expensive	Wide spectrum of sensor range
Shortage	High expense	Weaker performance	A little expensive

5) Overall Discussion: In overall, these three models have similar performance and specification of out resolution, however, *9DoF Sensor Stick* will be the least preferred option since it has the highest expense and it lacks of resources of its datasheet the relevant guidance. By comparing between *MPU-9250 IMU* and *9DoF IMU*, they both have specific datasheet that can be found, and their prices are both within the acceptable range, so by looking at more specific detail of their hardware performance, *9DoF IMU* has wider sensor range to choose for its output resolution, which is more functional than the *MPU-9250 IMU*. Therefore, the *9DoF IMU* model would be the first preference for selecting the represent MEMS IMU.

C. Hardware selection of Communication Protocol Type

1) Context: There are many different types of protocol for hardware component devices communicating with each other. We chose three different protocol types that are the most common types to use, as well as doable in our hardware system.

2) Options: I2C (Inter-Integrated Circuit), *SPI (Serial Peripheral Interface)* and *UART (Universal Asynchronous Receiver/Transmitter)* are three possible communication protocol types for our project, each of them has its own advantages and lack. In regards to this project, we will consider the following criteria for choosing the most appropriate protocol type for our hardware devices communication [7].

3) Criteria:

1) **High Transmission Speed**: This is one of the most critical criteria to be consider, since our hardware system requires multi-devices to process near synchronously, a fast communication speed between hardware devices is necessary.
2) **Transmission Distance**: This criterion is less important within this system since we consider all of the hardware devices will be implemented as a whole device, that means transmission distance of the protocol type wont be a necessary criterion.
3) **Multi-Devices Support**: Since we may use more than 1 IMU to work together in order to gathering more accurate data, the protocol we choose to use must support multi-devices communication.
4) **Number of wire needed to connect with microcontroller**: This criterion is less important but less number of wires between hardware device provides higher portability of the system, and it also reduces the difficulties on the hardware set-up process.

4) Table of Detailed Comparison:

Protocol Type	*I2C/IIC*	*SPI*	*UART*
Transmission Speed (Standard Speed Mode)	>1 Mbit/s	∼10 Mbit/s	0.3 Kbit/s ∼ 1 M bit/s
Transmission Distance	Short (Within integrated circuit components)	Short (Within integrated circuit components)	Long (wireless)
Multi-Devices Support	Yes	Yes	Yes
Number of Wire Needed	2	3 + 1 for each signal line	none
Advantage	Easy to implement	Fast transmission speed	can be implemented in wireless environment
Shortage	Slow transmission speed	More number of wires needed	Hard to implement

5) Overall Discussion: By comparing the above criteria, we first may eliminate UART because it has the slowest transmission speed within these three types, so its not expected for using in our system. SPI has significant fast speed however since we will use multiple IMUs connecting together in the system, SPI doesnt provide a good portability that it requires 1 more physical wire for each additional devices. Therefore, I2C is the most preferable option, it has relatively high transmission speed, and its easy to implemented for a complicated system.

D. User Interface Toolkit

1) Context: Graphical user interface will be a crucial of for our project presentation and demonstration system. The algorithm from our project will output a raw alignment data and it will be hard to understand the output without the help of graphical user interface. The graphical user interface will put contexts to the data and presents the correct symbology of the alignment data output. User interface toolkits is needed to develop our project user interface. We are looking for a user interface toolkit that will not add more complexity to the project, have minimal impact on the development process of the project, and have a useful library for user interface functionality.

2) Options: There are three options for the user interface toolkit. The first option is the visual studio user interface toolkit. Visual studio user interface toolkit is a built-in toolkit in visual studio that used for programmer to create a window application user interface. The second option is GTK+ toolkit. GTK+ toolkit is an open source cross-platform widget toolkit for creating graphical user interfaces [8]. The last option is the IUP portable user interface. IUP is a computer software development kit that provides a portable, scriptable toolkit to build a graphical user interface [9].

3) Criteria:

1) **Complexity**: Our clients encouraged us to spend most our time in developing and refining the dynamic alignment algorithm. Having a less complex toolkit will give us extra time to work on the alignment algorithm.
2) **Accessibility**: Toolkit and library that is easily accessible is desired in this project. A free and easy to access toolkit is desired.
3) **Adoption Rate**: Toolkit that well adapted in the programmer community will have more resources for debugging and refining our user interface.
4) **Time Commitment**: Using a user interface toolkit that will require huge learning curve and time to learn it will not add much benefits to the overall project.
5) **Library**: More powerful library will have more tool to further refine our graphical user interface.
6) **Overall cost and benefits**: Key strengths and weaknesses of using a particular user interface toolkit.

4) Table of Detailed Comparison:

Toolkit Name	Visual Studio Toolkit	GTK+	IUP
Complexity	Least complex	Less complex	Most complex
Accessibility	Accessible for free. Already have it in our computer.	Accessible for free online.	Accessible for free online.
Adoption Rate	Have a lot of tutorials and references online	The most popular toolkit for graphical user interface.	Less popular than GTK+ toolkit.
Time Commitment	Minimal	Medium	High
Library	Least powerful	Less powerful	Most powerful
Overall Cost and Benefit	Simple and easy to use toolkit. The least powerful toolkit, with the least learning curve.	More powerful option than visual studio, simpler than IUP.	The most powerful option. A complete development tool. Have the most cost and learning curve

5) Overall Discussion: By comparing criteria visual studio toolkit is the simplest and ready to use user interface toolkit option. All of the user interface toolkit options are easily accessible and available for free online. Although, IUP is the most powerful option that we have, it will also bring a lot of complexity to our project. IUP is a development tool that completely different than visual studio, using IUP will change the workflow of the development process. The learning curve and time commitment to learn IUP is huge and it will reduce our time to work on the algorithm of the project. Although GTK+ has the best overall cost and benefits, there is learning curve associated with it and it will also reduce our time to work on the algorithm of the project. Although a user interface is a critical part in our project presentation, this functionality is not a critical part of the project as a whole. Our clients strongly emphasize the quality of our algorithm. The most important piece of this project is the dynamic alignment algorithm and our clients encouraged us to spend most our time in developing and refining the dynamic alignment algorithm. Thus, using the toolkit with the least learning curve is encouraged for our project. This makes us decided to use the built-in visual studio toolkit to create our projects graphical user interface.

E. Programming Language

1) Context: We are looking for a better programming language that will help us to improve the quality of our project in terms of complexity, speed, and memory allocation. The goal of using a particular programming language is to reduce the complexity of writing code for our project, while having an acceptable speed and spaces memory. Programming language that has compatibility with the hardware is also desired.

2) Options: Our clients give us freedom to choose the programming language that we want to use to create this project. There are three options for programming language that we can use for this project. The options for the programming languages are C, C++, and Assembly language.

3) Criteria:

1) **Language Orientation**: Language orientation will change the way we implement the algorithm. More familiar language orientation is desired.
2) **Language Level**: Low-level programming language is a programming language that provides little or no abstraction from a computer's instruction set architecture [10].
3) **Complexity**: This criteria will determine the difficulty and complexity of implementing algorithms in a particular programming language. Less complex programming language is desired.
4) **Speed**: This criteria will determine the running speed of a compiled software written in a particular programming language. A fast running software is desired for this project.
5) **Memory Allocation**: This criteria will determine the memory allocation required to store a compiled software written in a particular programing language. This criteria will also determine the memory usage of a software during running time. More efficient memory allocation is desired for this project.
6) **Hardware Compatibility**: This criteria will determine the communication efficiency between the software and the hardware. More compatible programming language is desired for this project.
7) **Overall Cost and Benefit**: Key strengths and weaknesses of writing software in a particular programming language.

4) Table of Detailed Comparison:

Language Name	C	C++	Assembly
Language Orientation	Object oriented	Object oriented	No orientation
Language Level	High level language	High level Language	Low level language
Complexity	Medium	Medium	Complex
Speed	Fast	Fast	Fastest
Memory Allocation	More efficient	Most efficient	Most efficient
Hardware Compatibility	More Compatible	Compatible	Most Compatible
Overall Cost and Benefit	The best high level programming language to interact with hardware	The less hardware compatibility version of C	The most complex programming language with the best performance to communicate with hardware

5) Overall Discussion: Based from the criteria above, it is clear that assembly language has the best performance and benefits in writing a program that works with hardware. However, the low level language of assembly makes writing code in assembly really complex. With little or no abstraction from a computer's instruction set architecture, this will slow our development time for the algorithm of this project. C++ is a great programming language that is adopted greatly in the programmer community. However, in terms of our project requirement, it is also clear that C++ is the less hardware compatibility version of C. C is arguably one of the best high level programming language to interact with hardware. This makes us decide to use C as our programming language for this project.

F. Statistical Analysis Method

1) Context: The goal of using a statistical analysis method is to find the credibility of the output that we get from the algorithm. There are a lot of options of using statistical method to further refine our alignment data. We are looking for a statistical analysis method that will allow us to find the credibility on our alignment data or error tolerance.

2) Options: Confidence interval, credible interval, and tolerance interval are three possible options of statistical analysis method that we can use to gain credibility of our alignment data. Confidence interval will measure the frequency of repeated events [11]. How much true value that we get from running the test repeatedly. Credible interval will give us the degree of certainty about a values, given the observed data, there is a probability that the true value of falls within the credible region [11]. Tolerance interval will give us the spread of error in our alignment data.

3) Criteria:

1) **Type of statistics**: Each statistics have their own philosophy of finding credibility of their data. We will determine which type of statistics is most suitable for the project.
2) **Parameters**: In our project case, the parameter will be the expected value of alignment data.
3) **Data**: The data will be the alignment data outputs that we get from the algorithm.
4) **Bound**: We are looking to achieve error within an acceptable bound for confidence and credible interval. Tolerance interval will give us bound values of the sample after the calculation.
5) **Result**: The result will vary for each method. We will determine which result will be the most appropriate.

4) Table of Detailed Comparison:

Method Name	Confidence Interval	Credible Interval	Tolerance Interval
Type of Statistics	Frequentist	Bayesian	Can be both
Parameters	Fix	Random	Fix
Data	Random	Random	Fix
Bound	Fix	Fix	Random
Result	Frequency of true value output	Degree of uncertainty about a value	Statistics of acceptable error tolerance

5) Overall Discussion: The confidence interval will give us the frequency of the true value output being generated by the algorithm over a time period. While being run repeatedly, our algorithm will produce the same result, which implies the consistency and the credibility of our algorithm data. However, until the time of calculation, we are not sure what kind of value that we will get from the algorithm. Hence, leaving us with no parameter to work on confidence interval. The tolerance interval will give us the spread of error in alignment data being generated by the algorithm. Tolerance interval can be a great way for us to make sure that we have fulfil the error tolerance requirement for this project. This statistical method can also be useful for us to further refine our algorithm. We can adjust the alignment data by the error interval that tolerance interval gave us to make our alignment data more accurate. Credible interval will give us the credibility to state the certainty of a true value within an interval of data. This method will give us the true value that might be a great representation of our alignment data. We think that credible interval method fits the most with our project. Thus, this makes us decide to do credible interval for our statistical analysis method.

G. IRU Data Representation

1) Context: Our solution requires an accurate reference point to compare against when determining the correct alignment offset. The reference point in an aircraft is the data output from the aircrafts IRU. Since access to an expensive IRU is limited, we will need to find another method that is sufficient to test against.

2) Options: There are three simulation techniques to represent the sensor output from the aircrafts IRU for our demonstration system. The first technique would be to use a cheap MEMS for the sensor data when tracking motion. The second technique would involve redundant MEMS and an algorithm to average across for higher accuracy. The third technique would be to fully simulate the IRU data [12]. This last technique would require the precise input of motion that could be simulated at runtime [13].

3) Criteria:

1) **Accuracy**: As the IRU is used as a reference point, the method to represent the IRU's data must maintain acceptable accuracy.
2) **Cost**: Cost is a driving consideration as the IRU is already too expensive. The chosen method should stay within our budget.
3) **Difficulty**: The chosen method should be one in which we are able to implement within the allotted timeframe.

4) Table of Detailed Comparison:

Method	Single MEMS Sensor Data	Redundant MEMS Improved Sensor Data	Fully Simulated Sensor Data
Accuracy	Low	Medium	High
Cost	Low	Medium	Medium/High
Difficulty	Low	Medium	Low/Medium

5) Discussion: The single MEMS sensor data simulation technique is the most predictable and straightforward. This technique is dependent on the chosen hardware. Although the redundant MEMS improved sensor data simulation technique also depends on the chosen hardware, the accuracy may be higher as a result of overlapping data being used to reduce error. The noise reduction for redundant MEMS is expected to be 1/sqrt(n). Lastly, the fully simulated sensor data technique would ideally provide the highest accuracy. The approach would be to mount the demonstration system to a MEMS testing device. The simulated data would match the scripted motion of the testing device at runtime. This approach could be less difficult but the cost is currently unknown.

6) Selection: At this time we are planning on following the single MEMS sensor data simulation technique as the cost and difficulty are both low. Although the accuracy is less than the other two methods, it can be improved with the selection of hardware.

H. Filter technique

1) Context: In order to find the alignment offset we need data that represents the HUD position in 3d space. We have chosen to use a quaternion output when calculating position as it provides the necessary information. Not only must the output be in a usable form, it is also desired to be of high accuracy. Individual sensor output will have some amount of error as specified by hardware. By combining sensor data, performance issues can be reduced [14].

2) Options: Three options are presented to improve MEMS accuracy. The first option is to the Extended Kalman Filter (EKF). The next option is to use the Mahony Filter. Lastly, we could simply enable the quaternion output mode if supported by hardware such as Invensenses Digital Motion Processor (DMP).

3) Criteria:

1) **Accuracy**: When trying to find the correct alignment offset, accurate positional data is required.
2) **Complexity**: Given that our budget limits the available hardware we have access to, the chosen algorithm must be quick enough to run in realtime.
3) **Availability**: The choice must be available for use in our demonstration system.

4) Table of Detailed Comparison:

Method	**EKF**	**Mahony**	**DMP**
Accuracy	High	Medium	Medium
Complexity	High	Medium	Low
Availability	Open Source	Open Source	Hardware Dependent

5) Discussion: The EKF provides the highest accuracy among the other two options [15]. The high accuracy can be attributed to the high complexity of the algorithm itself. This high complexity is the main drawback to choosing EKF as our hardware must be able calculate the result fast enough to keep up with its output. The EKF is available as open source. Next, similar accuracy is achieved by both the Mahony Filter and Invensense DMP. While the Mahony Filter is also open source, the Invensense DMP is dependent on hardware as the chip is only on select boards [3]. Both the Mahony Filter and Invensense DMP are able to run efficiently on most boards where applicable.

6) Selection: At this time we are considering the MPU-9250 IMU which supports DMP. We will plan on using this feature unless a change of hardware is made. Higher performance equipment will enable us to use the EKF in place of the Mahony Filter.

III. Conclusion

Throughout this technology review we covered each of the eight different technologies that were deemed critical to our project. Each technology was provide three potential options to choose between for inclusion in our proposed solution. Before making any decisions, we performed research to effectively compare the options against each other. As a result we have made the appropriate implementation decisions to move forward with in the development of our solution.

REFERENCES

[1] "Arduino pro mini 328 - 3.3v/8mhz," Sparkfun Electronics. [Online]. Available: https://www.sparkfun.com/products/11114
[2] "Adafruit metro mini 328 - 5v 16mhz," Adafruit. [Online]. Available: https://www.adafruit.com/product/2590
[3] "Invensesense mpu-9250 ca-sdk reference board user guide," InvenSense Inc, 2012. [Online]. Available: https://store.invensense.com/datasheets/invensense/MPU-9250CA-SDK.pdf
[4] "https://www.sparkfun.com/products/10724," Sparkfun Electronics. [Online]. Available: SparkFun9DegreesofFreedom-SensorStick
[5] "Sparkfun imu breakout - mpu-9250," Sparkfun Electronics. [Online]. Available: https://www.sparkfun.com/products/13762
[6] "Sparkfun 9dof imu breakout - lsm9ds1," Sparkfun Electronics. [Online]. Available: https://www.sparkfun.com/products/13284
[7] "Introduction to ic and spi protocols." [Online]. Available: http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/
[8] "Gtk+," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/GTK%2B
[9] "Iup (software)," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/IUP_(software)
[10] "Low-level programming language," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Low-level_programming_language
[11] "Confidence vs. credibility intervals." [Online]. Available: http://freakonometrics.hypotheses.org/18117
[12] "Improving accuracy with multiple sensors: Study of redundant mems-imu/gps configurations," Swiss Federal Institute of Technology Lausanne. [Online]. Available: https://www.researchgate.net/profile/Stephane_Guerrier/publication/255962728_Improving_accuracy_with_multiple_sensors_Study_of_redundant_MEMS-IMUGPS_configurations/links/00463520ff0b1c5507000000.pdf?origin=publication_list
[13] "Precision testing for mems accelerometers," Nist. [Online]. Available: https://www.nist.gov/news-events/news/2016/04/precision-testing-mems-accelerometers
[14] "Solutions for mems sensor fusion," Mouser. [Online]. Available: http://www.mouser.com/applications/sensor_solutions_mems/
[15] "Experimental comparison of sensor fusion algorithms for attitude estimation." [Online]. Available: http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2014/media/files/1173.pdf