Carleton UNIVERSITY
Canada's Capital University

Software Design Specification Sensor Fusion Software

SYSC 5709 Course Project

Revision 1.0

Jaspal Singh -101126853
Vinay Venkataramana Chary – 101118107
GitHub Repository: https://github.com/jaspal-carleton/SensorFusionAlgorithm

Table of Contents

Revision History

Version	Name	Reason for Changes	Date
1.0	*Jaspal Singh* *Vinay Chary*	*Initial Revision*	*31/10/2019*

Reviewed and Approved By

Name	Department	Comments	Date
Christina Ruiz Martin	*SYSC 5709 Project Supervisor*		

1. Introduction

1.1 Purpose

The purpose of the sensor fusion software is to calculate a single aggregate value from a stream of sensor data input by essentially adapting a sensor fusion algorithm. Sensor fusion algorithm in consideration is "***A Simple Multi-Sensor Data Fusion Algorithm Based on Principal Component Analysis"*** The algorithm takes row data from the sensors as inputs, perform Principal Component calculations and returns a value. The value generated by the algorithm is representative of the correct raw data from the sensors (i.e. there may be sensors giving wrong measures). The algorithm is executed every time new sensor data is received. The algorithm receives the data from all the sensors synchronously.

1.2 System Overview

The software will be built using C programming Language with internal standard libraries and use of external third-party libraries. Deliverable of the software will include a fused sensor output data in .csv format time stamped with software execution times, Logs for every software execution. Highlight out of range sensor inputs indicating sensor damages. Project will follow standard Coding conventions agreed upon from the reviewers and approvers mentioned. A version control system is provisioned for the project using Git and GitHub repositories for collaboration from team members of the project. Industry Standard software documentation will be provided for use of the software which includes necessary instructions for Data handling and execution of the software.

2. Design Considerations

2.1 Assumptions

- Sensor fusion algorithm Considered: A Simple Multi-Sensor Data Fusion Algorithm Based on Principal Component
- Software development Environment: C programming, Developer choice IDE
- External third-party libraries will be used for complex mathematical functions
- Input and output Data will be in csv format
- Software documentation will be English Language
- Exceptions will be handled with relevant error messages
- License: TBD
- Coding conventions followed will be approved by approvers and reviewers

2.2 Constraints/Risks

- Complexity and Accuracy of Sensor fusion algorithm
- Errors in Sensor fusion algorithm
- Reliable input test data sets
- Development environment setup issues

2.3 Software Development Environment

Software will be built using tools and compilers based on individual developer choice, make file instructions will be provided in the software documentation so as to enable any end user to execute the software

Tentative Development environment details

- Operating System: Windows 10.1
- Compiler: GCC
- Terminal: Cygwin
- Local Version Control repository: Git
- Cloud hosted version control repository: GitHub
- Document generation tool: Doxygen

2.4 Version Control system

Version control systems enables a software development team to manage changes to source code over the time of development and keeps track of every modification to the code in a dedicated exclusive database. Recently committed changes can be turned back the clock and compare earlier versions of the code to help fix any mistakes while minimizing disruption to all team members. In this software development exercise, poplar version control platform Git and GitHub will be used.

2.5 Git/GitHub Repository

Git will be used as a local repository for the project by individual developer and continuous integration and Continuous development will be integrated into cloud repository GitHub as the development progress.

GitHub repository for this project development
https://github.com/jaspal-carleton/SensorFusionAlgorithm

2.6 Folder Structure

Folder structure for the said project development environment is as shown in below in Fig 1. A local Git repository is initially created, and development components are pushed to GitHub by individual developers

```
$ tree SensorFusionAlgorithm -a
SensorFusionAlgorithm
├── .git
├── .gitignore
├── CHANGELOG.md
├── LICENSE.md
├── README.md
├── bin
├── build
├── data
│   ├── input
│   │   └── sample_input.csv
│   └── output
│       └── sample_output.csv
├── doc
├── include
├── lib
├── logs
├── makefile
├── src
│   └── main.c
└── test
    ├── data
    └── src
```

Fig 1: Folder and File structure of GitHub Repository

Details of the folders

- CHANGELOG.md: Contains Change log details
- LICENSE.md: Contains License info the software
- README.md: Contains general description of the software and usage instructions
- bin: binary files of the project will be stored and maintained here
- build: Contains executable
- data: Contains input and output data
- doc: Contains Software documentation
- include: Contains header files
- lib: Standard libraries and external third-party libraries
- logs: Contains log files with error codes
- makefile:
- src: Contains Source code
- test: Contains tests cases and source code to test cases

3. Software Architecture

The architecture provides the top-level design view of the sensor fusion software system. This architecture serves as basis for more detailed design work

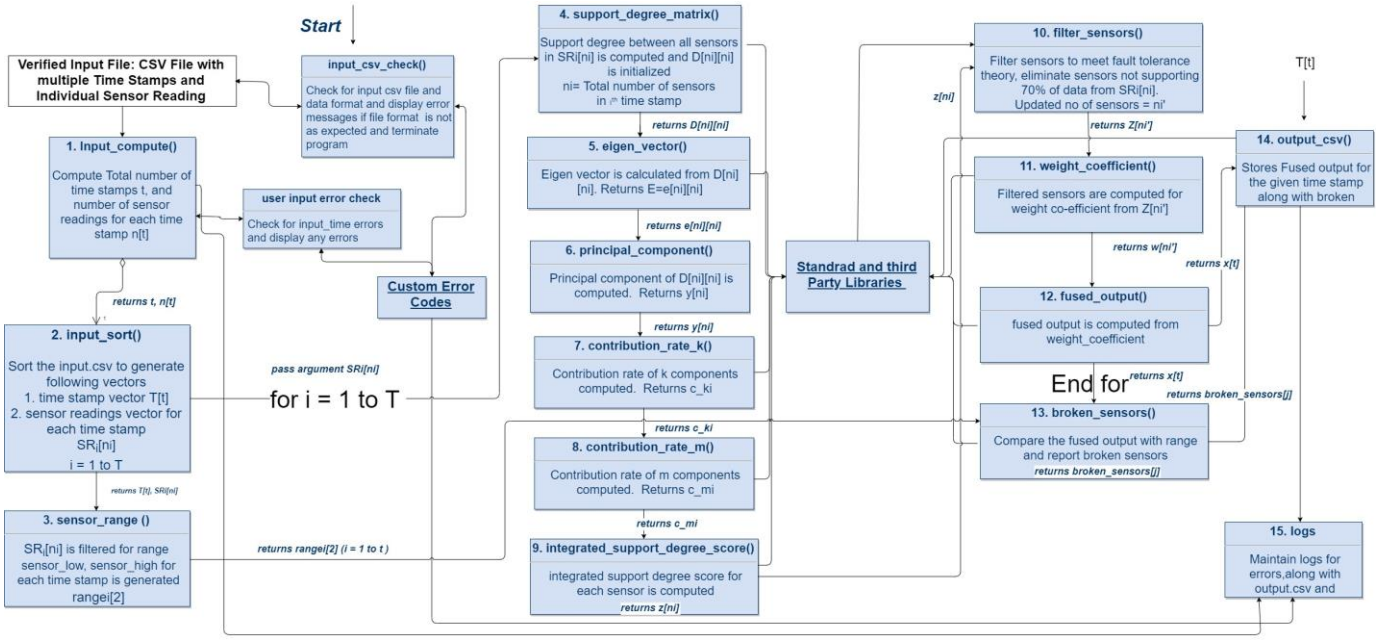

Fig 2: Proposed Software architecture diagram for Sensor Fusion Software Algorithm

3.1 Software Components and data flow

Fig 2. Shows the comprehensive high-level architecture for Sensor fusion software architecture. Data flow depicted in this architecture represents the overall software functionality. Each component of the software is represented as various functions and associated libraries with it. Detailed description of the data flow is presented as below

User is expected to provide the input in csv file with the format of three column data with Time stamp, Sensor and Sensor value fields. Below is the example of the input.csv file

time_24hr	sensor_name	value
13:20	sensor1	10
13:20	sensor2	20
13:50	sensor1	30
13:50	sensor2	30.5

Fig 3: Input.csv file format

Upon reception of input.csv file, program execution begins and following operations will be executed

input_csv_check(): Validates input csv file and data format and display error messages if file format and fields are not as expected and terminate program if any errors. function invokes standard and third-party libraries, custom error code function for display of appropriate error messages

Verified input.csv file will be passed to input_compute() function

1. ***input_compute():*** Compute Total number of time stamps in given input file t, and number of sensor readings for each time stamp n[t]. This function returns n[t] and is passed onto *input_sort()*

2. ***input_sort():*** Sort the input.csv to generate following vectors
   - time stamp vector T[t]
   - sensor readings vector for each time stamp SRi[ni] i = 1 to T
   This function returns T[t], SRi[ni] and is passed onto *sensor_range()* and to *support_degree_matrix()* in a for loop for over i=1 to T

3. ***sensor_range():*** SRi[ni] is filtered for range sensor_low, sensor_high for each time stamp is generated. This function returns rangei[2] (i = 1 to t) and is passed onto broken_sensors()

4. ***support_degree_matrix():*** Support degree between all sensors in SRi[ni] is computed and D[ni][ni] is initialized ni= Total number of sensors in i^{th} time stamp. This function returns D[ni][ni] and is passed onto *eigen_vector()*

5. ***eigen_vector():*** Eigen vector is calculated from D[ni][ni] and this function returns E=e[ni][ni] and is passed onto *principal component()*

6. ***principal component():*** Principal component of D[ni][ni] is computed and this function Returns y[ni] and is passed onto *contribution_ratek()*

7. ***contribution_ratek():*** Contribution rate of k components is computed and this function returns coke and is passed on to *contribution_ratem()*

8. ***contribution_ratem():*** Contribution rate of m components computed and this function returns c_mi and is passed onto ***integrated_support_degree_score()***

9. ***integrated_support_degree_score()***: integrated support degree score for each sensor is computed and returns z[ni] and is passed onto *filter sensors()*

10. ***filter_sensors():*** Filter sensors to meet fault tolerance theory, eliminate sensors not supporting 70% of data from SRi[ni]. Updated no of sensors = ni'. This functions returns Z[ni'] and is passed onto *weight_coefficient()*

11. ***weight_coefficient():*** Filtered sensors are computed for weight co-efficient from Z[ni'] and this function returns w[n'] and is passed onto fused_output()

12. ***fused_output():*** fused output is computed from weight_coefficient and this function at the end of for loop returns a vector of fused output for all the time stamps X[t]. Further, returned value will be passed to *output_csv()*. X[t] is also passed to *broken_sensors()* function.

13. ***broken_sensors():*** Accepts sensors range rangei[2] and fused output x[t] to determine sensors which are reporting out of range values and will be reported as broken sensors. This function returns broken_sensors[j] and is passed onto *output_csv().*

14. ***output_csv():*** Renders X[t] with respective time stamps and broken_sensors[j] in csv format

15. ***logs():*** Maintain error logs for respective input and output files, will be generated upon request by user via flag.

All functions and software components will make use of standard and third-party libraries
Best practice test cases will be built for Unit and functional testing of all the functions.

Appendix A: Project Timeline

Project Deliverables and Timelines with developer tasks

SL No	Deliverable	Developer	Status	Timeline
1	Create Git Repo	JS	Completed	Oct-31
2	Implement Folder structure of the project	JS	Completed	Oct-31
3	Software Architecture	VV	Completed	Oct-31
4	Software Design Document	VV	Completed	Oct-31
5	Finalize Project Development Environment	VV	In Progress	Nov-10
6	Create makefile	VV	In Progress	Nov-10
7	Finalize Coding Conventions	JS	In Progress	Nov-10
8	Implement CSV File parser	VV	Planned	Nov-15
9	Identify all required libraries	JS	Planned	Dec-01
10	Implement the Sensor Fusion algorithm	VV & JS	Planned	Dec-01
11	Test Case development, Unit testing and Functional testing of Sensor Fusion algorithm	VV & JS	Planned	Dec-01
12	Software Documentation	VV	Planned	Dec-05
13	Readme Documentation	JS	Planned	Dec-05
14	GitHub Project Management	JS & VV	Planned	Dec-10
15	Additional features/ Branches	JS & VV	Planned	Dec-15