Carleton UNIVERSITY
Canada's Capital University

Sensor fusion Software based on Principal components

SYSC 5709 Course Project
User Manual

Course Supervisor:
Dr. Christina Ruiz martin
Department of Systems and Computer Engineering,
Carleton University, Ottawa, ON, Canada

Collaborators:
Scott Jordan, Tarun Panuganti

Revision 1.0

Student Name: Jaspal Singh
Student Name: Vinay Venkataramanachary
Submitted on Dec 17 2019

Table of Contents

Contents

1. Introduction

This document outlines the procedure to use the Sensor fusion Software based on Principal components available on the below GitHub repository
https://github.com/jaspal-carleton/SensorFusionAlgorithm.git

General Introduction to Software: 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 software can be executed for any new sensor data is via input.csv file. The algorithm receives the data from all the sensors synchronously via input.csv file

Software Development Environment:

1. Programming Language: C
2. IDE Used: Sublime
3. Version Control System: GitHub
4. External Library used: GSL, GNU Scientific Library
5. Testing:
   - Unit API Testing
   - Manual Testing
6. Project Planning: GitHub Project Board
7. Project Management Model: Agile
8. Manual Documentation
9. External Reviewers

2. Compatible Platforms

This software is tested and compatible on below platforms

1. Ubuntu 18.04
2. MAC Mojave 10.14
3. Windows 10

3. Pre-requisites

Before cloning the repository to the local machine, kindly ensure to meet the following pre-requisites on your local machine

1. Installation of Git

   Reference link to install git on Linux/windows/MAC: Click Here

2. gcc compiler and GNU packages (make, cmake)

   - Reference link to install gcc and GNU on Windows: Click Here
   - Installation on Linux and MAC is covered in section 4

3. GSL library

   Reference link to install GSL libraries on Windows: Click Here

   Installation on Linux and MAC is covered in section 4

4. Path variables set to use GSL library:

   Reference link to set path variable on Windows : Click Here

   Path associated with lib folder of GSL library will be added as a new path variable in the environment.

   Setting path variables to use GSL library on Linux and MAC is covered in section 4

4. Steps to Install Pre-requisites on Ubuntu 18.04/ MAC OS

Step1: Open Terminal Window in the desired directory

Step 2: Install Git by using below command

sudo apt install git

Step 3. Install GCC compiler by issuing below command

sudo apt install gcc

Step 4: Install GNU packages by issuing below command

sudo apt install build-essentials

Step 5: Clone the repository by issuing below command

git clone https://github.com/jaspal-carleton/SensorFusionAlgorithm.git

Step 6. Navigate to the Git Repository folder by issuing below command

cd SensorFusionAlgorithm

Step 7. Provide permissions to run env_setup script

chmod 777 env_setup.sh

Step 8. Install GSL Environment setup by issuing below command

./env_setup.sh

Step 9. Set the newly installed GSL Library as a path variable by issuing below command

source ~/.bashrc

or

edit ~/.bashrc file using vi/vim and add a path variable pointing to lib/gsl/lib folder

5. Running the Software

Note: Please ensure that all pre-requisites have been met.
If you are using Non-Linux machine, please ensure to meet all the prerequisites mentioned above
Ensure to install make, cmake, wget packages in Cygwin to run the make commands

Step 1: Open a terminal with make access and navigate to the root folder of repository

Step 2: Clean residual files

make clean

Step 3: Build the project (Ignore warnings if any)

make

Step 4: Navigate to bin folder

cd bin

Step 5: run the build main file by issuing below command

./main

Software executes and displays fused outputs on the terminal screen, and also the output is stored in /data/output folder. Also, a log file with the execution time stamp details will be generated in /logs folder.

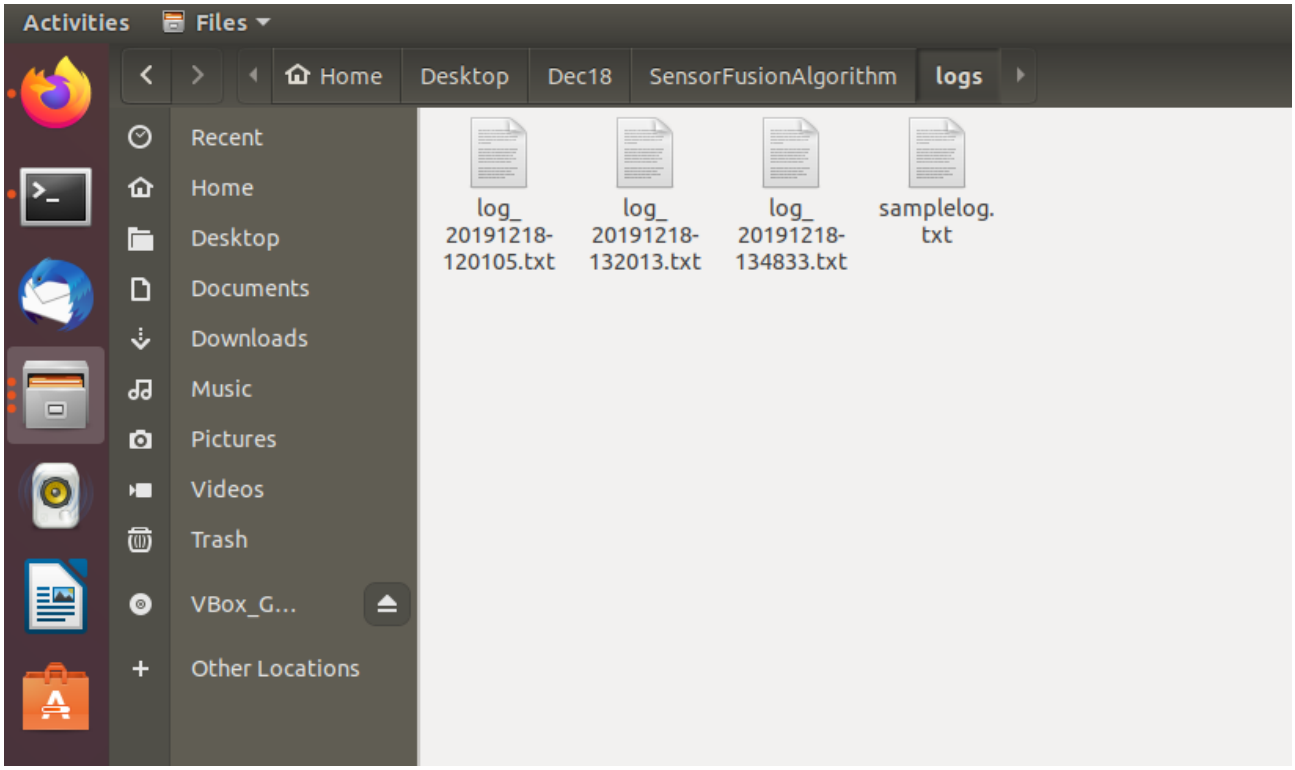
Below is the screen shot of the terminal screen.

```
vinay@vinay-VirtualBox:~/Desktop/Dec18/SensorFusionAlgorithm$ ./bin/main
DEBUG: Absolute path => /home/vinay/Desktop/Dec18/SensorFusionAlgorithm
DEBUG: Current timestamp => 20191218-142215
DEBUG: Log Path => /home/vinay/Desktop/Dec18/SensorFusionAlgorithm/logs/log_20191218-142215.txt
DEBUG: Input CSV Path => /home/vinay/Desktop/Dec18/SensorFusionAlgorithm/data/input/input.csv
DEBUG: Output CSV Path => /home/vinay/Desktop/Dec18/SensorFusionAlgorithm/data/output/output.csv
INFO : Computed support degree matrix DEBUG: Value => 1.000000
DEBUG: Eigen Value => 1
DEBUG: Eigen Vector =>
1
INFO : Compute contribution rate
DEBUG: Rate => 1
DEBUG: Count of Contribution rates => 1
INFO : Compute support degree matrix
1.000000
INFO: Principal Component => 0-1.000000, Contribution Rate => 0-1.000000
DEBUG: Time - 13.20, Fused Value: 30.000000

INFO : Computed support degree matrix DEBUG: Value => 1.000000
DEBUG: Eigen Value => 1
DEBUG: Eigen Vector =>
1
INFO : Compute contribution rate
DEBUG: Rate => 1
DEBUG: Count of Contribution rates => 1
INFO : Compute support degree matrix
1.000000
INFO: Principal Component => 0-1.000000, Contribution Rate => 0-1.000000
DEBUG: Time - 13.50, Fused Value: 20.000000

INFO : Computed support degree matrix DEBUG: Value => 1.000000
DEBUG: Eigen Value => 1
DEBUG: Eigen Vector =>
1
INFO : Compute contribution rate
DEBUG: Rate => 1
DEBUG: Count of Contribution rates => 1
INFO : Compute support degree matrix
1.000000
INFO: Principal Component => 0-1.000000, Contribution Rate => 0-1.000000
DEBUG: Time - 14.10, Fused Value: 40.000000
```

Below is the screenshot of output.csv file

Time: 13.20 Fused Value: 30.000000
Time: 13.50 Fused Value: 20.000000
Time: 14.10 Fused Value: 40.000000
Time: 14.50 Fused Value: 50.000000
Time: 15.20 Fused Value: 10.000000
Time: 16.50 Fused Value: 20.000000

Below is the screen shot of logs folder



6. Input and output accessibility

If the user needs to run the program again for a different set of time stamps and sensor readings, follow the below steps

Step1: Navigate to /data/input folder

Step2: Replace the existing input.csv with the desired input file

Note: Ensure that the file name is input and is on .csv format

Step 3: Go to terminal and navigate to the root folder of the repository

Step 4: Clean previous build files by issuing below command

make clean

Step 5: Build the program for the new input file by issuing below command

make

Step 5: Build the program for the new input file by issuing below command

make

Step6: Navigate to the newly created bin folder by issuing below command

cd bin

Step7: Execute the program by issuing below command

./main

Fused output for new input file will be displayed on the screen

/data/output folder will have output.csv file which contains fused output values for corresponding timestamps
Please note: Previous output.csv files any, will be overwritten

/logs folder will contain log files

7. Testing

To perform function testing follow below steps

Step 1: Open a terminal with make access and navigate to the root folder of repository

Step 2: Clean residual files

make clean

Step 3: Build the project (Ignore warnings if any)

make

Step 4: Navigate to bin folder

cd bin

Step 5: For testing the various sensor APIs issue below command

./main -t

Test Cases will execute and display the result on terminal or screen

Below is the screenshot of test functions

```
-----------------------------------
Starting Automated APIs Testing
-----------------------------------

----------------------------------------
Testing API: compute_contrib_rate_count()
----------------------------------------
DEBUG: Count of Contribution rates => 2
Test Failed. Expected Value = 2, Actual Value = 2

-----------------------------------------------
Testing API: compute_integrated_support_degree()
-----------------------------------------------
INFO: Principal Component => 0-1.000000, Contribution Rate => 0-0.100000
INFO: Principal Component => 0-1.000000, Contribution Rate => 1-0.100000
INFO: Principal Component => 0-1.000000, Contribution Rate => 2-0.100000
INFO: Principal Component => 1-2.000000, Contribution Rate => 0-0.200000
INFO: Principal Component => 1-2.000000, Contribution Rate => 1-0.200000
INFO: Principal Component => 1-2.000000, Contribution Rate => 2-0.200000
INFO: Principal Component => 2-3.000000, Contribution Rate => 0-0.300000
INFO: Principal Component => 2-3.000000, Contribution Rate => 1-0.300000
INFO: Principal Component => 2-3.000000, Contribution Rate => 2-0.300000
INFO: Principal Component => 3-4.000000, Contribution Rate => 0-0.400000
INFO: Principal Component => 3-4.000000, Contribution Rate => 1-0.400000
INFO: Principal Component => 3-4.000000, Contribution Rate => 2-0.400000
Test Passed

-------------------------------------
Testing API: eliminate_incorrect_data()
-------------------------------------
Test Passed. Expected Value = 0, Actual Value = 0

----------------------------------------
Testing API: compute_weight_coefficient()
----------------------------------------
Test Passed. Expected Value = 0.212121, Actual Value = 0.212121

-----------------------------------
Testing API: compute_fused_output()
-----------------------------------
Test Passed. Expected Value = 6.200000, Actual Value = 6.200000

-----------------------------------
Finished Automated APIs Testing
-----------------------------------
```

8. Additional Logging Feature (Bonus)

This software has an additional logging feature which allows the user to perform some debugging in the case of program termination or errors. Upon execution of the program, a txt file with timestamp of the execution will be generated in /logs folder.

However, this feature is not complete and has following limitations: Upon execution of the program, a txt file with timestamp of the execution will be generated but the txt file is empty and is not recording any of the log information that is required for debugging.

This can be fixed and implemented in future for better user experience

9. Bug Reports

We performed a manual testing to verify all the functionalities of the software and below results were observed

Test case Number	Description	Input File	Tested Platform	Expected result	Actual result	Test case Status
1	Platform testing	6 Unique Time stamps	Ubuntu 18.04	Compile and produce Fused Output	Compiled and produced Fused Output	Pass
2	Platform testing	6 Unique Time stamps	MAC OS 10.14 Mojave	Compile and produce Fused Output	Compiled and produced Fused Output	Pass
3	Platform testing	Developer Provided	Windows 10	Compile and produce Fused Output	Compiled and produce Fused Output	Pass
4	Input Function testing	No Input File	Ubuntu 18.04	Should generate an file error message	Generated a file error message	Pass
5	Input Function testing	Input File with incorrect file name	Ubuntu 18.04	Should generate an file error message	Generated a file error message	Pass
6	Zero values Input testing	6 Time stamps with few 0 values	Ubuntu 18.04	Produce accurate fused output for all 6 time stamps	Produced accurate fused output for all 6 time stamps	Pass

Test case Number	Description	Input File	Tested Platform	Expected result	Actual result	Test case Status
7	Negative value Input testing	6 Time stamps with few negative values	Ubuntu 18.04	Produce accurate fused output for all 6-time stamps	Produced accurate fused output for all 6-time stamps	Pass
8	Single reading Input testing	6 Time stamps with 1 reading foreach time stamp	Ubuntu 18.04	Produce accurate fused output for all 6-time stamps	Produced accurate fused output for all 6-time stamps	Pass
9	Character values Input testing	6 Time stamps with characters as sensor readings	Ubuntu 18.04	Produce accurate fused output for all 6-time stamps (Character inputs should be converted to ASCII Values)	Produced accurate fused output for all 6-time stamps (Character inputs was converted to ASCII Values)	Pass
10	Scale Testing	10 unique time stamps	Ubuntu 18.04	Produce accurate fused output for all 10-time stamps	Produce accurate fused output for all 10-time stamps	Pass
11	Log Handling Additional feature testing	6 Unique Time stamps	Ubuntu 18.04	Produce a txt file for every software execution with execution timestamp as file name	Produced a txt file for every software execution with execution timestamp as file name	Pass
12	Log Handling Additional feature testing	6 Unique Time stamps	Ubuntu 18.04	Produced log file should contain Fused output and its components	Text File is empty	Fail