

**Sensor Fusion Algorithm Software**

**Developer Manual**

SYSC5709F [35794]

Advance Topics in Software Engineering

**Authors:**

**Shreya Taneja:** [**shreyataneja@cmail.carleton.com**](mailto:shreyataneja@cmail.carleton.com)

**Ankit Kumar Vig:** [**ankitvig@cmail.carleton.ca**](mailto:ankitvig@cmail.carleton.ca)

***Introduction***

*Sensor fusion* is the process of merging data from multiple sensors such that to reduce the amount of uncertainty that may be involved in a robot navigation motion or task performing. Sensor fusion helps in building a more accurate world model for the robot to navigate and behave more successfully. The three fundamental ways of combining sensor data are the following:

* *Redundant sensors*: All sensors give the same information for the world.
* *Complementary sensors*: The sensors provide independent (disjoint) types of information about the world.
* *Coordinated sensors*: The sensors collect information about the world sequentially.

Algorithms used for sensor fusion must deal with noisy inputs and generate an estimation of stable state provided by all the sensors.

***A Simple Multi-Sensor Data Fusion Algorithm Based on Principal Component Analysis***

When we measure data (let’s focus on the example of getting data using sensors), we usually use the measures from different sensors to assure correctness. One of the methods we can use to calculate a single value from our input data is to perform a sensor fusion algorithm. The algorithm takes as inputs the row data from the sensors, perform some 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. There are different methods for doing this, but we will focus in one of them i.e. PCA.

***Data Fusion Based on PCA:***

*Data Fusion Based on PCA*

Fuzzy-index function *dij* only denotes the support degree of senor *i* by sensor *j*, but it cannot reflect the integrated support degree of senor *i* by all sensors in the same group.

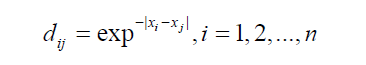
Equation proposed a method to determine the integrated support degree of sensors by computing eigenvector of relation matrix. But this method cannot reflect total information of

original variable.

So, in this paper, we adopt PCA to calculate the integrated support degree of all sensors.

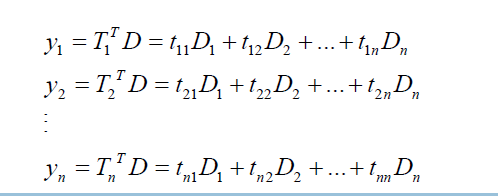
Detailed steps are as follows:

1. According to following equation, calculate the support degree matrix D.



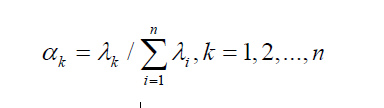
*2)* According to following equation, calculate all eigenvalues of the matrix *D*: *λ*1, *λ*2,…, *λ*n(*λ*1≥*λ*2≥…≥,*λ*n), and the corresponding eigenvectors:*T*1, *T*2,…, *T*n, where *Ti* =[*ti*1, *ti*2,…, *ti*n]*T* (*i*=1,2,…,*n*).

*3)* According to (3), calculate the principal components *yi* (*i*=1, 2,…,*n*):

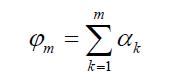


where *y*1, *y*2…, *yn* are the first principal component, the second principal component,…, and the *n*th principal component respectively; *Di* is the *i*th column of matrix *D*.

*4)* Calculate the contribution rate of the *k*th principal component by:

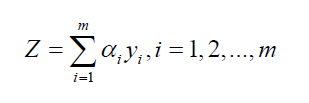


*5)* Calculate the accumulated contribution rate of the *m* principal components by:



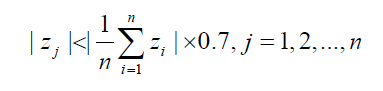
where ϕ*m* indicates the ratio of the first *m* principal components over all information of original variable. In Practical Application, the number of principal components is determined by criterion: ϕ*m*>85%. Thus, the first *m* principal components are *y*1, *y*2, …, *ym*.

*6)* Calculate the integrated support degree score of all sensors by:



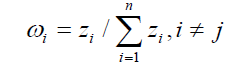
Obviously, vector *Z* is made up of *n* elements: *Z*= (*z*1, *z*2,…, *z*n). *zi* is defined as the integrated support degree of sensor *i*.

According to theory of fault-tolerant, if a sensor is not supported by more than 2/3 sensors, its observation value is regarded as invalid. So, in this paper, the validity of sensor’s observation value is determined by:

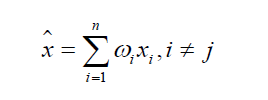


where | *zj* | denotes absolute value of *zj*.

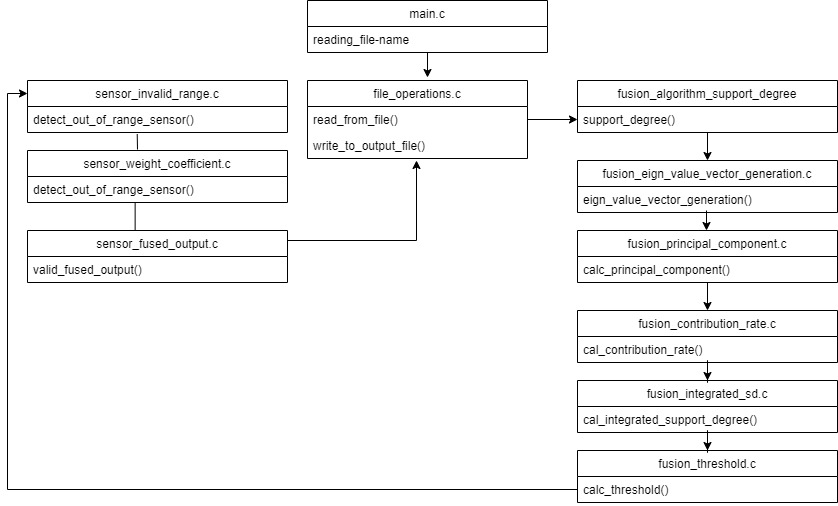
If *zj* is not satisfied (14), the sensor *j* should be disregarded before fusion. The rest observation values *xi*(*i*≠*j*) are allocated weight coefficient according to their integrated support degree:



Then, the final fusion expression is obtained by:



***Sensor Fusion Algorithm Software Design (Architecture)***



The purpose is to define the architecture of the Sensor Fusion algorithm project i.e. a brief description of the structure of files folders and behaviour of the file.

 We are following the approach so that most of the calculations are separated in between different functions and c files.

The main.c call the file input function ‘read\_from\_file()’ from file\_operations.c to read the sensor data from the input file  and then the data is parsed using the parse function formatting the data in a particular data structure (Sensor Structure).

Then the fusion algorithm calculations are performed as per the sensor fusion algorithm in multiple functions in different c files i.e.

* Deduce support degree matrix in support\_degree () function from fusion\_algorithm\_support\_degree.c.
* Calculate eigen Values and Eigen Vector of the Support Degree matrix in eigen\_value\_vector\_generation () function in fusion\_ eigen\_value\_vector\_generation.c.
* Calculate Principal Component with eigen vectors and the support degree matrix in calc\_principal\_component () in fusion\_principal\_component.c.
* Calculation of Contribution Rate and sum contribution rate with eigen value and degree matrixcalc\_contribution\_rate () function in Fusion\_contribution\_rate.c.
* Calculate Integrated support degree from contribution rate and sum of Integrated support degree Cal\_integrated\_support\_degree () from fusion\_integrated\_sd.c
* Calculate the threshold value of all the sensors from above calculations in calc\_threshold () in fusion\_threshold.c

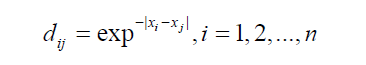
The fusion algorithm steps have been calculated and now it needs the out of range sensors to be eliminated, so that we can calculate weight coefficient and thus the fused output from the remaining within the range sensors.

* Detect\_out\_of\_range () sensor detects the output of range sensors, i.e. the sensors whose values are less than the threshold value calculated in previous step, and the sensors are eliminated in sensor\_invalid\_range.c
* The non-eliminated sensors integrated support degree and sum of integrated support degree is used to calculate the weight coefficient in sensor\_weight\_coefficient () in Sensor\_weight\_coefficient.c
* Valid\_fused\_output () calculates the fused outputs from non-eliminated sensors in sensor\_fused\_output.c.

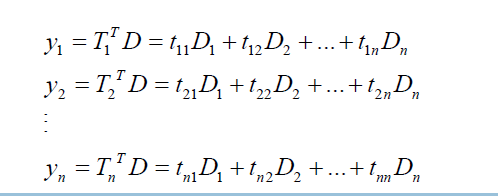
Finally, the values which are meant to be written in output file are returned to write\_to\_output\_file () function to write data from file\_operations.c.

***Implementation of Sensor Fusion Algorithm***

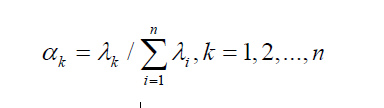
The program starts with passing Main.c the input file name as the argument. Then the control passes to the file\_operation.c where the csv file is read, the list is divided as per each time zone and the data is passed to each variable of type struct sensor. Then from read file the fusion\_algorithm \_support\_degree process the data as per the equation

.

Then we use the GSL- GNU Scientific Library to calculate the eigen values and the eigen vectors. The eigen values and vectors and then used to generate principal components by multiplying the eigen vector matrix and support degree matrix both are of size n\*n.

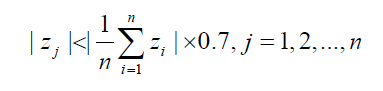


The next step pf the algorithm is to calculate the contribution rate which is an array formed by dividing each eigen vale to the sum of all eigen values.



In the next step we are supposed to calculate integrated support degree of each sensor value. We multiply the principal component matrix with the contribution rate array which leads to the generation of the integrated support degree matrix which is determined by criterion ϕ*m*>85%.

The next function calculates the threshold by the following equation. Which is the absolute sum of integrated support degree matrix divided by the number of elements into 0.7.



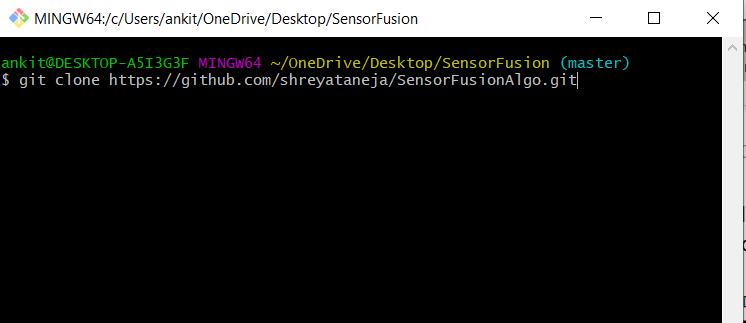
Then we eliminate the sensor values which are less than the threshold value. Then we calculate the weight coefficient by dividing the elements of integrated support degree array with the sum all the non eliminated elements.

Then the result output file is generated in in same folder as of the input file in the following format inputsfolder/inputfilename\_output.csv. In format fused Output and sensor time and the values.

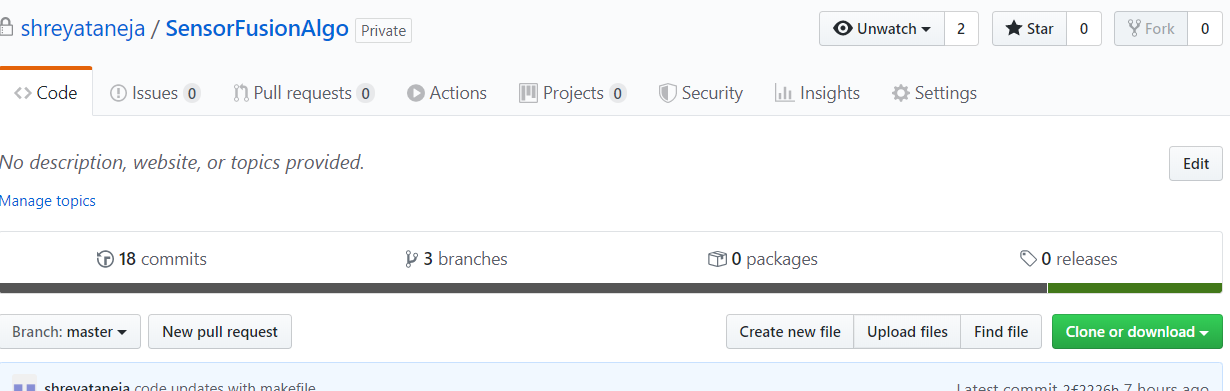
***Downloading the Sensor Fusion Algorithm Software***

1. Install Git (User can use Git bash or any other software of choice to clone the project).
2. Create a new folder for the project. Right Click inside the folder and open a git bash here. Right Click +” Git Bash here”.
3. Type the following command in the git terminal:

Git clone <https://github.com/shreyataneja/SensorFusionAlgo.git>



1. Or go to the link <https://github.com/shreyataneja/SensorFusionAlgo> and click on **Clone or Download** and download the zip folder of the software.

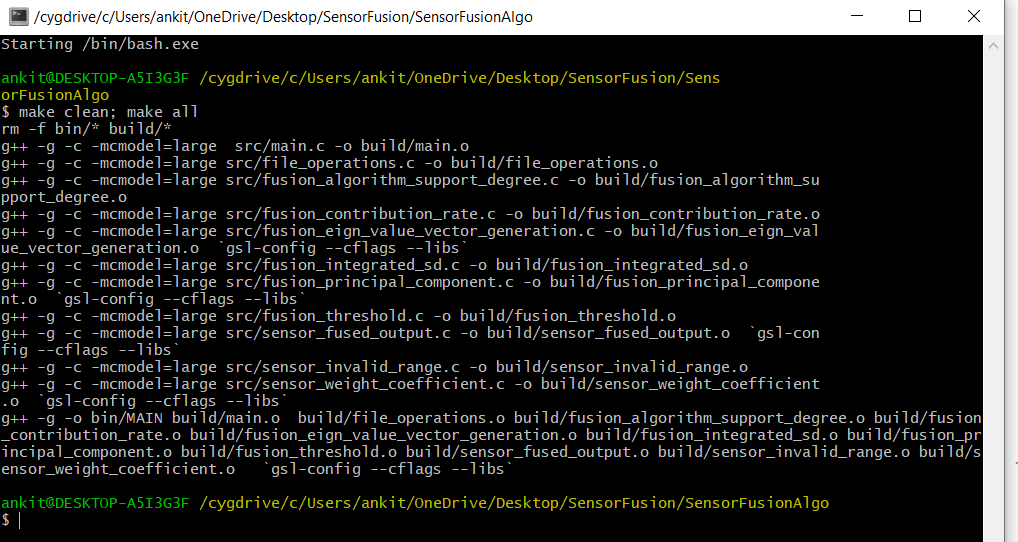


**Compiling and Running the Sensor Fusion Algorithm Software**

To compile and run the tests for the software, follow these steps:

1. Compile the project
2. Open the folder containing the cloned or downloaded software from git.
3. Open Bash Prompt inside the folder i.e. Right Click + ”Bash Prompt here”.
4. To compile the project, type in the bash prompt:

Make clean; make all

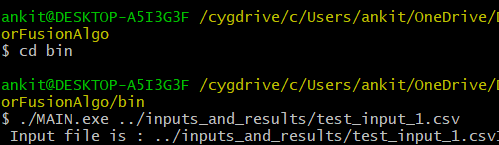


1. Run the software and test with input file of your choice. Place the input file in the inputs\_and\_results folder.
2. From the previous command, you now have new folders created , i.e. bin and build , the tests will run in the bin folder .

Type: cd bin

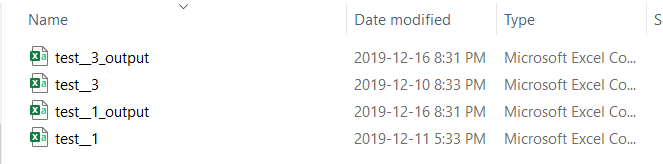
1. To run test with input file of your choice, for example, input\_sensorfusion.csv, type:

./MAIN.exe ../inputs\_and\_results/test\_input\_1.csv



1. To run the software with different input files, simply create a .csv file with desired inputs and replace in command with new file name
2. The output file will be created in **same folder as of the input file** in the following format: inputsfolder/inputfilename\_output.csv

Example:



**Testing**

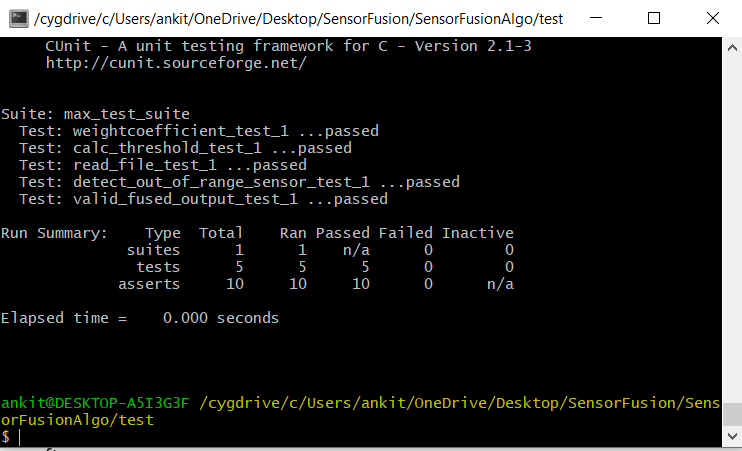
Testing is done using CUnit. CUnit was installed using Cygwin. As Cygwin provides direct installation of CUnit package. So we install CUnit using the following command:

**apt-cyg install CUnit**

Test folder contains the code for all the unit test cases performed. In order to evaluate the proper functioning of each function defined for development of software.

Makefile for the test is also provided. Tests can be compiled and executed in the same way as mentioned before like the same program.

Test output results are displayed as:



***References***

## [Sensors and Data Acquisition](https://www.sciencedirect.com/science/article/pii/B9780128111536000014) Diego Galar, Uday Kumar, in [eMaintenance](https://www.sciencedirect.com/book/9780128111536), 2017

1. A Simple Multi-sensor Data Fusion Algorithm Based on Principal Component
2. Analysis GAO Hongyan College of Information and Electrical Engineering Shandong University of Science and Technology Qingdao, China
3. J.Z. Sasiadek, “Sensor Fusion,” *Annual Reviews in Control*, vol. 26,pp. 202-228, 2002.