Sensor fusion Software based on Principal components

SYSC 5709 Course Project
Developer 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

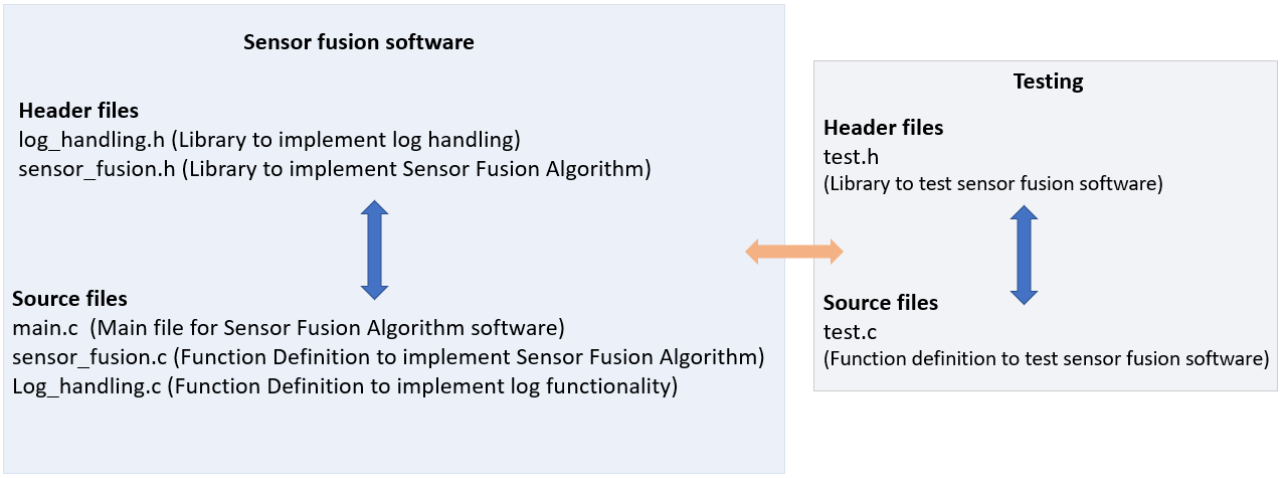
Table of Contents

Contents

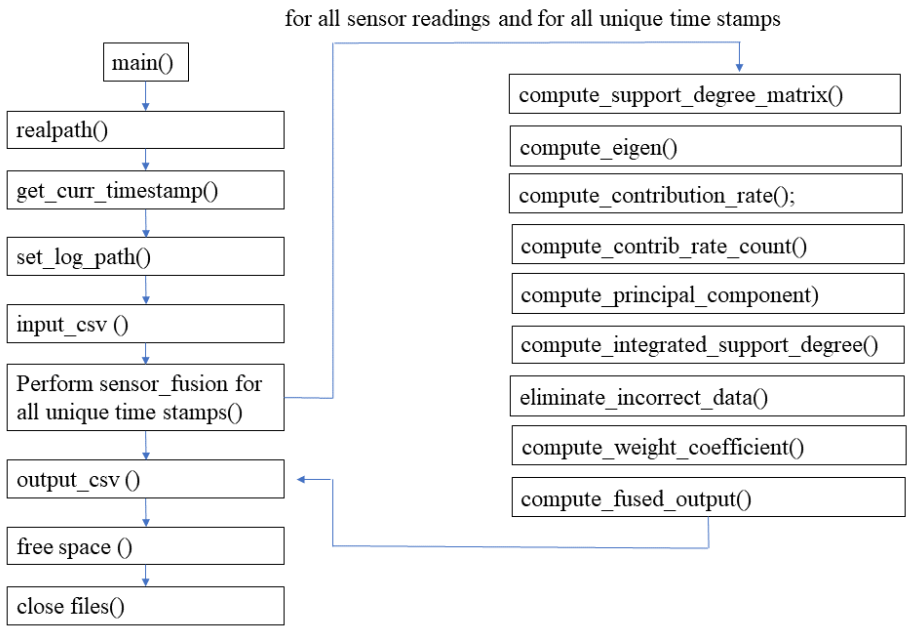
1. Introduction

This document outlines all the functions and programming components used in the development of sensor fusion software according to doxygen notation.

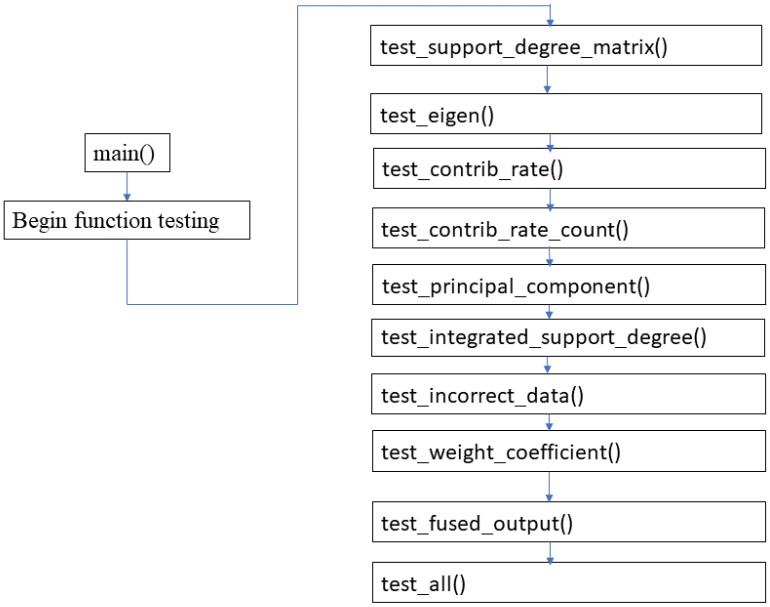
Below diagram represents the components and data flow of the Sensor fusion software



2. Data flow for Sensor fusion of software



3. Data Flow for testing



4. main function

Brief Description: Main file for Sensor Fusion Algorithm software
Authors: jaspalsingh@cmail.carleton.ca, vinayvenkataramanach@cmail.carleton.ca

Program data flow

1. get project absolute path: realpath(argv[0], NULL);
2. get current timestamp: get_curr_timestamp(time_now)
3. set log path: sprintf(log_path, "%s/logs/log_%s.txt", proj_abs_path, time_now)
4. input csv file: sprintf(temp_csv1, "%s/data/input/input.csv", proj_abs_path)
5. output csv file: sprintf(temp_csv2, "%s/data/output/output.csv", proj_abs_path)
6. perform sensor fusion for each unique time stamp and its sensor readings
    - compute_support_degree_matrix(sensor_values)
    - compute_eigen(spd)
    - compute_contribution_rate(eigen, spd->sensor_count);
    - compute_contrib_rate_count(contrib_rate, 0.5, spd->sensor_count);
    - compute_principal_component(spd, eigen->eigen_vector, contrib_rate_count)

- compute_integrated_support_degree(principal_components_matrix,contrib_rate, contrib_rate_count, spd->sensor_count)
- eliminate_incorrect_data(integrate_support_matrix, 0.7, spd->sensor_count)
- compute_weight_coefficient(integrate_support_matrix, spd->sensor_count)
- compute_fused_output(weight_coff, sensor_values, spd->sensor_count)

7. free space occupied by malloc's
   - free(input_csv)
   - free(output_csv)
8. close files
   - fclose(fp)
9. function testing API's: getopt(argc, argv, "t")

5. sensor_fusion functions

5.1 compute_support_degree_matrix()

struct support_degree_matrix *compute_support_degree_matrix(double *sensor_readings)

Brief description: Structure pointer for computed support degree matrix
Detailed description: Structure pointer for calculating support degree matrix
Input parameters: sensor_readings array of sensor readings
Return parameters:

- Success: pointer to the structure support_degree_matrix
- Failure: NULL

5.2 eigen_value_vector *compute_eigen()

struct eigen_value_vector *compute_eigen(struct support_degree_matrix *spd)

Brief description: Structure pointer calculate eigen value and eigen vector
Detailed description: Structure pointer for calculating eigen value and eigen vector
Input parameters: support_degree_matrix pointer to structure support degree matrix
Return parameters:

- Success: pointer to structure eigen_value_vector
- Failure: NULL

5.3 compute_contribution_rate()

double *compute_contribution_rate(struct eigen_value_vector *eigen, int sensor_count)

Brief description: Function to calculate the contribution rate of principal component
Detailed description: Compute the contribution rate of kth and mth principal components
Input parameters: eigen_value_vector pointer to structure of eigen_value_vector
Return parameters:

- Success: pointer to contribution_rate of type double
- Failure: NULL

5.4 compute_contrib_rate_count()

int compute_contrib_rate_count(double *contribution_rate,float threshold,int sensor_count)

Brief description: Function to compute contribution rate
Detailed description: Compute the count of contribution rates to be used for sensors
Input parameters: contribution_rate, threshold, sensor_count number of sensors to be used
Return parameters:

- Success: contribution rate count
- Failure: -1

5.5 compute_principal_component()

double **compute_principal_component(struct support_degree_matrix *spd,double **eigen_vector, int contrib_rate_count)

Brief description: Function to compute principal component
Detailed description: Computed principal component for each of eigen vector
Input parameters:

- support_degree_matrix pointer to structure support degree matrix
- eigen_vector pointer to pointer of eigen vector of type double
- contrib_rate_count number of contribution rate to be used

Return parameters:

- Success: principal of support degree matrix
- Failure: -1

5.6 compute_integrated_support_degree()

double *compute_integrated_support_degree(double **principle_components,double *contribution_rate, int contrib_rate_count, int sensor_count)

Brief description: Function to compute integrated support degree
Detailed description: Computed support degree matrix for each of the sensors

Input parameters:

- s principle_components pointer to pointer of principle component of type double
- contribution_rate pointer to contribution rate of type double
- contrib_rate_count number of contribution rate to be used
- sensor_count number of sensors

Return parameters:

- Success: array of support degree matrix
- Failure: NULL

5.7 eliminate_incorrect_data()

int eliminate_incorrect_data(double *integrated_support_degree_matrix,double fault_tolerance_threshold, int sensor_count)

Brief description: Function to eliminate invalid sensor readings
Detailed description: Discard the faulty sensor readings from the support degree matrix
Input parameters:

- integrated_support_degree_matrix pointer to integrated support degree matrix
- fault_tolerance_threshold threshold value to cut off invalid readings
- sensor_count total count of sensors

Return parameters:

- Success: 0
- Failure: -1

5.8 compute_weight_coefficient()

double *compute_weight_coefficient(double *integrated_support_degree_matrix,int sensor_count)

Brief description: Function to compute weight coefficient for each sensor
Detailed description: Compute weighted coefficients for each sensors from the support degree matrix
Input parameters:

- integrated_support_degree_matrix pointer to integrated support degree matrix
- sensor_count total count of sensors

Return parameters:

- Success: array of weighted coefficients
- Failure: NULL

5.9 compute_fused_output()

compute_fused_output(double *weight_coefficient,double *sensor_data, int sensor_count)

Brief description: Function to compute fused output
Detailed description: Compute the aggregated value of sensors from the weighted coefficients
Input parameters:

- weight_coefficient pointer to weighted coefficient of sensors
- sensor_data pointer to data of sensors
- sensor_count total count of sensors

Return parameters:

- Success: fused value of sensor data
- Failure: -1

6. log_handling functions

6.1 log_info_t *init_log()

struct log_info_t *init_log(char *proj_abs_path, char *time_now)

Brief description: Initialize log info
Detailed description: Initialize the log base path, name and other details
Input parameters:

- ptr log file pointer
- path log file path

Return parameters:

- Success: 0
- Failure: 1

6.2 open_log()

int open_log(struct log_info_t *log_info)

Brief description: Open log
Detailed description: Open log file
Input parameters:

- ptr log file pointer
- path log file path

Return parameters:

- Success: 0
- Failure: 1

6.3 close_log()

int close_log(struct log_info_t *log_info)

Brief description: Close log
Detailed description: Close log file
Input parameters:
- ptr log file pointer

Return parameters:
- Success: 0
- Failure: 1

6.4 write_log()

void write_log(struct log_info_t *log_info, char *str)

Brief description: Write to log
Detailed description: Write to log file
Input parameters:
- ptr log file pointer
- str string message to write to log file

Return parameters:
- Success: 0
- Failure: 1

7. Testing Functions

7.1 test_support_degree_matrix()

void test_support_degree_matrix(void)

Brief Description: Function to test support degree matrix
Detail Description: Test the support degree matrix for given value of sensors
Input parameters: Void function
Return parameters: Void function

7.2 test_eigen()

test_eigen()

Brief Description: Function to test eigen
Detail Description: Test the eigne vector and value for given value of sensors
Input parameters: Void function
Return parameters: Void function

7.3 test_contrib_rate()

test_contrib_rate()

Brief Description: Function to test contribution rate
Detail Description: Test the contribution rate for given value of sensors
Input parameters: Void function
Return parameters: Void function

7.4 test_contrib_rate_count()

test_contrib_rate_count()

Brief Description: Function to test contribution rate count
Detail Description: Test the contribution rate count for given value of sensors
Input parameters: Void function
Return parameters: Void function

7.5 test_principal_component()

void test_principal_component(void)

Brief Description: Function to test principal components
Detail Description: Test the principal components for given value of sensors
Input parameters: Void function
Return parameters: Void function

7.6 test_integrated_support_degree()

void test_integrated_support_degree(void)

Brief Description: Function to test integrated support degree matrix
Detail Description: Test the integrated support degree matrix for given value of sensors
Input parameters: Void function
Return parameters: Void function

7.7 test_incorrect_data()

test_incorrect_data()

Brief Description: Function to test incorrect data
Detail Description: Test the incorrect data for given value of sensors
Input parameters: Void function
Return parameters: Void function

7.8 test_weight_coefficient()

void test_weight_coefficient(void)

Brief Description: Function to test weight coefficient
Detail Description: Test the weight coefficient for given value of sensors
Input parameters: Void function
Return parameters: Void function

7.9 test_fused_output()

void test_fused_output(void)

Brief Description: Function to test fused output
Detail Description: Test the aggregated value of sensors from the weighted coefficients
Input parameters: Void function
Return parameters: Void function

7.10 test_all()

void test_all(void)

Brief Description: Function to test all test cases
Detail Description: Automatically Test all the test cases for sensor fusion
Input parameters: Void function
Return parameters: Void function

8. Proposed Additional features

1. Email the fused output to authorized user recipients
2. Perform Data analysis of the fused output with the historical data for feedforward accuracy
3. Alerts and Notifications for pre-defined threshold fused outputs