|  |  |
| --- | --- |
| **Project Name** | Kneemetrics: Matlab to C |
| **Description** | Conversion of the Kneemetrics Matlab based algorithm into C, and improvement of the algorithms computational / time efficiency |
| **Project Sponsor** | WSN Group, Tyndall National Institute |
| **Author(s)** | Luke Jervis |
| **Date** | August 30, 2016 |

Abstract:

Knee injuries have a significant economic-social impact to the health care system (in terms of hospital cost, possible infections, medication, rehabilitation) and to the individual (as per lost working days, pain, medical expenses, and possible dissatisfaction), and it is expected to be even higher in the next years. As an example, the number of knee-related surgeries has increased by 8.2% between 1990 and 2002 in the US, and is predicted to increase by 673% to 3.5 million by 2030, impacting on 7.2% Medicare expenditure. The annual cost to society in the US was $2.78 billion for surgery and $4.4 billion for rehab (2012). Therefore, there is a pressing need for this work in order to provide for rehabilitation, cheaply and efficiently. Currently, a wide range of technologies could be deployed as viable solutions (e.g. optoelectronics systems – VICON, instrumented treadmills and force platforms). While these technologies can achieve very high performance, they also present several drawbacks. They are limited to lab-environments, are expensive and need a certain amount of training. Also, despite the availability in literature of several algorithms for gait analysis starting from inertial sensors, the technology is still not widely used by clinicians in real life cases, due to the fact that the systems hamper the normal operations and are validated by only considering a small amount of patients.

To avoid such issues, Tyndall National institute developed an essential, simple, low-cost, low-power, wireless, real-time, minimally-invasive system able to provide highly accurate, reliable data, in order to contain the accuracy of a lab into a simple wearable system. The software, in our initial proof-of-concept, has been currently implemented by means of a scientific tool (MATLAB) requiring license, which makes the system not affordable for most of clinicians. The aim of the internship is to reproduce the source code, in C language, with particular focus on constrained resources, such as memory and computational complexity. The new code will then be easily integrated into microcontrollers or real-time executable applications. The front-end implementation will also be considered as part of the project.

Introduction:

The following sections provide documentation for all of the classes and functions included in the project. I have made a note of functions that contain modifiable parameters to meet WIMU board specification. The first two sections give a list of the various classes and internal structures in the project that will be of use to a future programmer. The next seven sections provide descriptions of each function and the associated parameters and return values. The final section is a user guide on how to run the program using a command prompt interface.

# The following classes are included in the project

main.c

preprocessing.c | preprocessing.h

txt2mat.c | txt2mat.h

quaternions.c | quaternions.h

vector.c | vector.h

resampling.c | resampling.h

horizontal\_alignment.c | horizontal\_alignment.h

temporal\_interval\_modules.c | temporal\_intervals\_module.h

liir.h | liir.c

# The following structures are included in the project

filter

info

pair

prev\_files

sens\_files

vector

vectors

WIMU\_Files

# Description of functions in main.c

Class: main.c

---------------

Contains all function calls for the Knee Habilitation Project

Also contains auxiliary print functions

Function: main

---------------------

int main(void)

---------------------

main function that is invoked when the program is run.

includes all necessary function calls for the program

Returns: 0 if the program is completed successfully, 1

a problem was encountered

Note: The main function runs the knee joint angles through a low pass filter and a median filter, the parameters of this may need to be updated for the new boards.

Function: menu

---------------

void menu(Info i)

---------------

Displays a menu on the command line for the user

to input data

i: A struct holding the patients inputted information

Returns: void

Function: print\_euler\_final

--------------------------

void print\_euler(vector euler, int num)

--------------------------

used to print the various knee angles

t: Vector containing thigh angles

s: Vector containing shank angles

Returns: void

Function: print\_euler

---------------------

void print\_euler\_final(vector t, vector s, int num)

---------------------

Auxiliary function used to print the quaternion angles to

a .txt file

euler: Pointer to the quaternion angle data vector

num: Identification number to be used at the end of the

.txt filename

Returns: void

Function: print\_temporal

---------------------

void print\_temporal(vector vec, int type, int num)

---------------------

Auxiliary function used to print the gait parameter data to

a .txt file

vec: Pointer to the gait parameter data vector

type: 0 for Instants data, 1 for Intervals data

num: Identification number to be used at the end of the

.txt filename

Returns: void

Function prev\_main

--------------------------

void prev\_main()

--------------------------

for use with the older boards that the original

matlab code was based on. I have left it here for

legacy purposes just in case there is a reversion

back to the original boards.

Returns: void

# Description of functions in preprocessing. c

Class: preprocessing.c

-----------------------------

Provides various filtering and interpolation functions to process WIMU data,

including a low pass and median filter

Function: lininterp1

--------------------

int lininterp1(vector euler)

--------------------

Preforms a linear interpolation on the timestamps of the samples

to account for sensor and read write errors

euler: The matrix containing sample data

Returns: 0 if the process was successful, 1 if an error occurred

Note: This function has a time stamp difference threshold of 2.0 when preforming linear interpolations. This may need to be updated for the new boards.

Function: lininterp2

---------------------

int lininterp2(vector euler)

---------------------

Interpolates timestamp data that dont meet the threshold and adds

precise data samples to the gaps

euler: The vector contianing sample data

Returns: 0 if the process was sucessful, 1 if an error occured

Note: This function has a time stamp difference threshold of 0.09 when preforming linear interpolations. This may need to be updated for the new boards.

Function: c\_coeff

-----------------

void c\_coeff(int n, double fcf, int sff, double ccof)

-----------------

Auxiliary function to find c coefficients for low pass butterworth filter

n: Filter order

fcf: Cutoff frequency

sff: Identifies whether to scale or not

ccof: array pointer that gets updated with c coefficients

Returns: void

Function: d\_coeff

-----------------

void d\_coeff(int n, double fcf, double dcof)

----------------

Auxiliary function to find d coefficients for low pass butterworth filter

n: Filter order

fcf: Cutoff frequency

dcof: array pointer that gets updated with d coefficients

Returns: void

Function: filter\_vector

--------------------

void filter\_vector(vector data, int order, double cut)

--------------------

Performs a low pass filter on the vectors columns

data: The vector to be filtered

order: The filter order

cut: The cutoff frequency as a fraction of PI

Returns: void

Function: filter\_data

--------------------

void filter\_data(double data, int order, double cut, int size)

--------------------

Performs a low pass filter on the array of doubles

data: The array to be filtered

order: The filter order

cut: The cutoff frequency as a fraction of PI

size: The size of the array to be filtered

Returns: void

Function: filter\_data\_helper

----------------------------

void filter\_data\_helper(double data, double dcof, double ccof, int nd, int nc, int size)

----------------------------

Auxiliary function to perform a low pass filter on the array of doubles

data: The array to be filtered

dcof: Array of d coefficients

ccof: Array of c coefficients

nd: The number of d coefficients

nc: The number of c coefficients

size: The size of the array to be filtered

Returns: void

Function: filter\_vector\_helper

-----------------------

void filter\_vector\_helper(vector data, double dcof, double ccof, int nd, int nc)

-----------------------

Auxiliary function to perform a low pass filter on the vectors columns

data: The vector to be filtered

dcof: Array of d coefficients

ccof: Array of c coefficients

nd: The number of d coefficients

nc: The number of c coefficients

Returns: void

Function: median\_filter\_set\_up

----------------------------

void median\_filter\_set\_up(vector vec)

----------------------------

Sets up a median filter for each column of a vector

vec: The vector to be filtered

Returns: void

Function: median\_filter

-----------------------

double median\_filter(double datum, filter filt)

-----------------------

Performs a median filter on numerical data

datum: The next data point entering the filter

filt: Filter struct that holds a columns filter information

Returns: The filtered data point

# Description of functions in txt2mat.c

Class: txt2mat.c

----------------

Provides functions to import data from .txt files and store them

in the underlying vector data structure

Function: open\_prev\_files

--------------------

int open\_prev\_files(Prev\_Files f, int number)

--------------------

Opens .txt files containing WIMU data to be imported

f: The Prev\_Files struct to be updated

number: The WIMU number

Returns: 0 if the files were opened correctly, 1 if

there was an error opening the files

Note: This function is for use with the old boards only

Function: open\_sens\_files

--------------------

int open\_sens\_files(Sens\_Files f)

--------------------

Opens .txt files containing WIMU data from the old boards to be imported

f: The Sens\_Files struct to be updated

Returns: 0 if the files were opened correctly, 1 if

there was an error opening the files

Note: This function is for use with the SAFE SENS boards only

Function scan\_prev\_files:

-----------------------

void scan\_prev\_files(Prev\_Files f, int leg\_num)

-----------------------

Scans time\_stamp, acc, gyr, txt files for ALL EXERCISES

on a single WIMU and stores

the data in a vector data structure

f: The Files struct containing file and vector data

leg\_num: Leg specifier: 0 = left, 1 = right

Returns: void

Note: This function is for use with the old boards only. This function preforms the JCS conversion

Function scan\_sens\_files:

-----------------------

void scan\_sens\_files(Sens\_Files f)

-----------------------

Scans SENS and EULER data files and stores

the data in vector data structures.

f: The Sens\_Files struct containing the files and vector data

Returns: void

Note: This function is for use with the SAFE SENS boards only. This function preforms necessary conversion from raw data. This function preforms the JCS conversion

Function: record\_size

---------------------

int record\_size(Prev\_Files f, int record\_number)

---------------------

Finds the number of rows that the cut vector will hold

f: The Files struct containing vector data

record\_number: The record number to count

Returns: The number of rows needed

Note: This function is for use with the old boards only

Function: cut\_record

--------------------

int cut\_record(Prev\_Files f, vector acc\_vec\_new, vector gyr\_vec\_new,

int record\_number)

--------------------

Cuts the WIMU data based on the record number and frees the

old vectors data

f: The Files struct containing vector data

acc\_vec\_new: The new acc vector that is filled with the cut data

gyr\_vec\_new: The new gyr vector that is filled with the cut data

record\_number: The record number to cut from the old vectors

Returns: 0 if function exited correctly, 1 if the record number

does not exist

Note: This function is for use with the old boards only

Function: print\_vectors

-----------------------

void print\_vectors(Prev\_Files f, int i)

-----------------------

Auxiliary function that Prints out the acc and gyc vector

arrays into a .txt file used for testing purposes

f: The Files struct containing vector data

i: Identification number to be used at the end of the

.txt filename

Returns: void

Function: compare\_double

------------------------

int compare\_double(double f1, double f2)

------------------------

Compares double f1 with double f2 for equality

f1: first double to compare

f2: second double to compare

Returns: 1 if they are equal and 0 if they are not equal

# Description of functions for quaternions.c

Class: quaternions.c

---------------------------

Performs a quaternion fusion algorithm on WIMU data to

compute knee-joint angles. the algorithm combines quaternion based

calculations from both gyroscope and accelerometer data. A gradient

descent algorithm is used to compute the direction of the gyroscope

measurement error as a quaternion derivative

Function: set\_up\_filters

------------------------

void set\_up\_filters(vector acc\_orig, double accf)

------------------------

Filters acc data and populates the accf vectors with the

vectors first 11 rows

acc\_orig: The acc vector to be filtered and added to accf

Returns: The populated accf array

Function: quaternion\_calc

-------------------------

void quaternion\_calc(vector acc\_orig, vector gyr\_orig, vector euler\_angles, double acc)

-------------------------

Performs the quaternion fusion algorithm on acc and gyr data

to find knee-joint angles angles

acc\_orig: The acc data from a WIMU

gyr\_orig: The gyr data from a WIMU

euler\_angles: Vector to be populated with knee-joint angles

acc: Array that contains the Accf filtered data

Returns: void

Function: my\_filter

-------------------

void my\_filter(double acc, double accf)

-------------------

Perform a low pass filter on the acc data,

returning the last X Y Z values

acc: The acc data to be filtered

accf: Array that will be populated with the

accf filtered data

Returns: void

Function: quaternion\_product

----------------------------

void quaternion\_product(double a, double b, double p)

----------------------------

Calculates the product of two quaternions

a: The first quaternion

b: the second quaternion

Returns: The product of a and b

Function: compute\_euler\_angles

------------------------------

void compute\_euler\_angles(double q, double result)

------------------------------

computes the eulerian angles given a quaternion

q: The quaternion to convert

Returns: Array populated with x,y,z eularian angles

Function: gradient\_descent

--------------------------

void gradient\_descent(double acc, double q, double mu, double gradient\_result)

--------------------------

Performs the gradient descent portion of the quaternion

fusion algorithm

acc: Array populated with the Accf filterd data

q: Array with the qosserv\_prev data

mu: The calculated mu value to be used

Returns: The newly calculated qosserv array

# Description of functions for resampling.c

Class: resampling.c

-------------------

This module temporally alignes the data from all of the WIMUs

Note: this assumes that the timestamps are already aligned

with an absolute starting timestamp, this is to be completed

by sending a signal for all of the WIMUs to begin calculations

over bluetoothwifi

Function: resampling\_right

--------------------------

int resampling\_right(vec v)

--------------------------

Resamples the data from the right leg

v: Vec struct containing the right leg data

Returns: 0 if the function is completed sucessfully,

or 1 if there was an error

Note: this function assumes that the timestamps are sorted in ascending order (Which they should be)

Function: resampling\_left

-------------------------

int resampling\_left(vec v)

-------------------------

Resamples the data from the left leg

v: Vec struct containing the right leg data

Returns: 0 if the function is completed sucessfully,

or 1 if there was an error

Note: this function assumes that the timestamps are sorted in acending order (Which they should be)

Function: resampling\_both

--------------------------

int resampling\_both(vec v)

--------------------------

Resamples the data from both legs

v: Vec struct containing the right leg data

Returns: 0 if the function is completed successfully,

or 1 if there was an error

Note: This function should only be called after the

right and left legs are resampled individually

Function: set\_up\_resample

-------------------------

void set\_up\_resample(vector orig\_vec, vector new\_vec)

-------------------------

Sets up the resampling functions by populating a

temporary vector with the original vectors data

orig\_vector: The original vector to copy

new\_vector: The new vector that will be populated

with the same data as orig\_vector

Returns: void

# Description of functions for horizontal\_alignment.c

Class: horizontal\_alignment.c

-----------------------------

This module horizontally aligns the shank WIMU's so that the

the shank and thigh knee-joint angles are in a common reference frame

Function: horzontal

-------------------

void horizontal(vector thigh, vector shank, vector euler, int leg)

-------------------

Performs a horizontal rotation of the shank knee-joint

angle data using the seel method

thigh: The thigh WIMU's acc angles

shank: The shank WIMU's knee-joint angles

euler: The shank knee-joint angles to be rotated

Returns: void

Note: The number of random samples used in the algorithm

may be changed to accommodate different sensor sample rates

Function: singular\_value\_decomposition

--------------------------------------

void singular\_value\_decomposition(double a, double PINV)

--------------------------------------

Computes the pseudoinverse of a matrix using the

singular value decomposition (SVD)

a: The matrix (vector in this application) to be

inverted

PINV: Storage for the pseudoinverse

Returns: void

Function: horizontal\_rotation

-----------------------------

void horizontal\_rotation(double theta, vector euler)

-----------------------------

Rotates the shanks knee-joint angles

horizontal plane (XY) using a rotation matrix

theta: Angle of horizontal rotation in radians

euler: A vector holding the knee-joint angles to be rotated

Returns: void

Function: norm

--------------

double norm(double v)

--------------

Computes the norm of a length 3 vector

v: Vector containing 3 elements

Returns: Norm of the vector

# Description of functions for temporal\_intervals\_module.c

Class: temporal\_intervals\_module.c

----------------------------------

Computes various gait parameters using the WIMU data and

knee-joint angles

Function: temporal\_instants\_1

-----------------------------

void temporal\_instants\_1(vector roll, vector gyr, int peaks, vector instants, int num\_peaks)

-----------------------------

Finds and stores the index and X axis gyro data for

the Toe-Off, Heel\_Strike, and Shank Vertical tempral

instants

roll: The knee-joint angles contianting roll data

in the first column of the vector

gyr: The gyro vector

peaks: An ordered array containing the index of each peak

instants: A vector to be updated with the temporal instants

in the following order [T\_O, T\_O\_I, H\_S, H\_S\_I, S\_V, S\_V\_I]

Returns: void

Fuction: single\_leg\_temporal\_intervals

--------------------------------------

void single\_leg\_temporal\_intervals(vector intervals, vector instants, vector t)

--------------------------------------

Computes the Gait Cycle Time (GCT), Swing Phase (SW\_PH),

and the Stance Phase (ST\_PH)

intervals: A vector to be updated with the temporal intervals

in the following order: [GCT, SW\_PH, ST\_PH]

instants: A vector containing the temporal instants

t: the gyro vector containing the time stamps in position [0]

Returns: void

Function: combined\_temporal\_intervals

-------------------------------------

void combined\_temporal\_intervals(vector r\_instants, vector l\_instants, vector ds, vector t)

-------------------------------------

Computes the Initial Double Support (IDS),

Terminal Double Support (TDS), and Double Support

Interval (DSI)

r\_instants: A vector containing temporal instants

for the right leg

l\_instants: A vector containing temporal instants

for the left leg

ds: A vector that will be populated with the double

support interval data

t: A vector containing the time stamps in the [0]

position

returns: void

Function: compute\_stridelength

------------------------------

void compute\_stridelength(vector stridelength, vector instants, vector acc, vector gyr, int sample\_f, int segment)

------------------------------

Computes the stridelength

stridelength: A vector that will be populated with the stridelength data

instants: A vector containing the temporal instants data

acc: A vector containing the acc data

gyr: A vector containing the gyr data

sample\_f: The sample freq of the inertial sensors (WIMU's)

segment: The length from the shank WIMU to the ground (cm)

Returns: void

Function: get\_peaks

-------------------

int get\_peaks(vector gyr, int start, int num\_peaks)

-------------------

Counts the number of peaks in the gyro X

axis data

gyr: The gyro vector

start: The row to begin counting peaks

Returns: The number of peaks

Function: sample\_freq

---------------------

double sample\_freq(vector vec)

---------------------

Computes the sample frequency of the

data set

vec: The data set with time stamp in the [3]

position

Returns: The sample frequency

Function: print\_gait

--------------------

void print\_gait(gait g) {

--------------------

Auxiliary function to print the gait

parameters to a .txt file

g: A structure containing necessary

vectors

Returns: void

# User guide for command prompt interface

The current build (.exe) file is located in the Debug folder of the project directory. If moving this file, make sure to also move the lapack DLL files that are in the directory. Without these DLL’s the program will not run. Make sure that any input files are in a folder named “input” in the same directory as the .exe file. The input files should have the form “SENS\_X.txt” if working with raw data, and “EULER\_X.txt” if working with computed orientation angles directly from the board. The X in both of these file names should be a number the differentiates the board. Start the program by double clicking the executable. A command prompt window will appear:



Follow the on scree prompts to fill in patientWIMU data. This menu prompts the user for the same data as the Matlab menu. After inputting your information, you will be given a summary of the data and asked for a conformation before proceeding. If you made a mistake you can fill out the form again. After confirming that the information is correct, the program will begin. If you did not specify a walking test (“W”) then five .txt files will be outputted:

output\_euler\_X.txt: These files will hold the [Roll, Pitch, Yaw, Time] calculations for each of the four WIMU’s in that order

output\_euler\_1.txt = right thigh

output\_euler\_2.txt = right shank

output\_euler\_3.txt = left thigh

output\_euler\_4.txt = left shank

output\_knee\_joint\_X: These files will hold the [Time, FE Angle Knee Joint, VV Angle Knee Joint, IE Angle Knee Joint, FE Angle Thigh, FE Angle Shank] for the right and left leg in that order.

output\_knee\_joint\_1 = right leg

output\_knee\_joint\_2 = left leg

If you specified a walking test (“W”) then you will also have the following two files:

gait\_parameters\_right\_leg: This file holds the gait parameter data for the right leg

gait\_parameters\_left\_leg: This file holds the gait parameter data for the left leg

The data will be displayed in this order, [R GCTs (s), R Stance Phase (s), R Swing Phase (s), R Stride Length (m), R Stride Speed (ms), R Clearance (m), R Cadence (GCmin), R Gate Variability]

All of these files can easily be copied and pasted into excel spreadsheets for analysis.