Name: Rodrigo Ignacio Rojas Garcia
Course: ECE 4310
Lab #7

Motion Tracking

In this project the student was to implement an algorithm to calculate the motion using accelerators and gyroscopes. The student was provided a text file with recorded data of an iPhone regarding Acceleration in the X, Y, and Z direction in units of gravities (G) as well as the Gyroscope movements for Pitch, Roll, and Yaw in units of radians per second. The sample time of each movement was of 0.05 seconds. The student was to use this information with the purpose of creating a C program which automatically segment the data into periods of motion and periods of rests and calculate the motion along and about each axis during the periods of motion. In order to complete the requisities, the following requirements were followed:

1. Read the "data.txt" text file
2. Smooth data
   1. Each Accelerometer and Gyroscope axes were smoothed, meaning, reducing "noise" from the raw data. This was achieved by using a window of size 25 which was used to determine the average value of the previous 24 data points (25th data point included) which replaced the current data point. This process repeated itself until all data points for both Accelerometer and Gyroscope were calculated.
   2. It should be noted that Smoothing of the data was not required, but was implemented with the purpose of visualizing and determining the Accelerometer and Gyroscope thresholds used to determine if object was in movement.
3. Determine Movement
   1. In order to determine if the iPhone was in movement, the variance of along all Accelerometer and Gyroscope axes was required to be calculated. The variance was used to determine if it was greater than a set threshold for Accelerometer and Gyroscope axes (thresholds for both Accelerometer and Gyroscope were different). It should be noted that a window size of 11 and 20 were used for variance. The following formula was used to calculate variance:

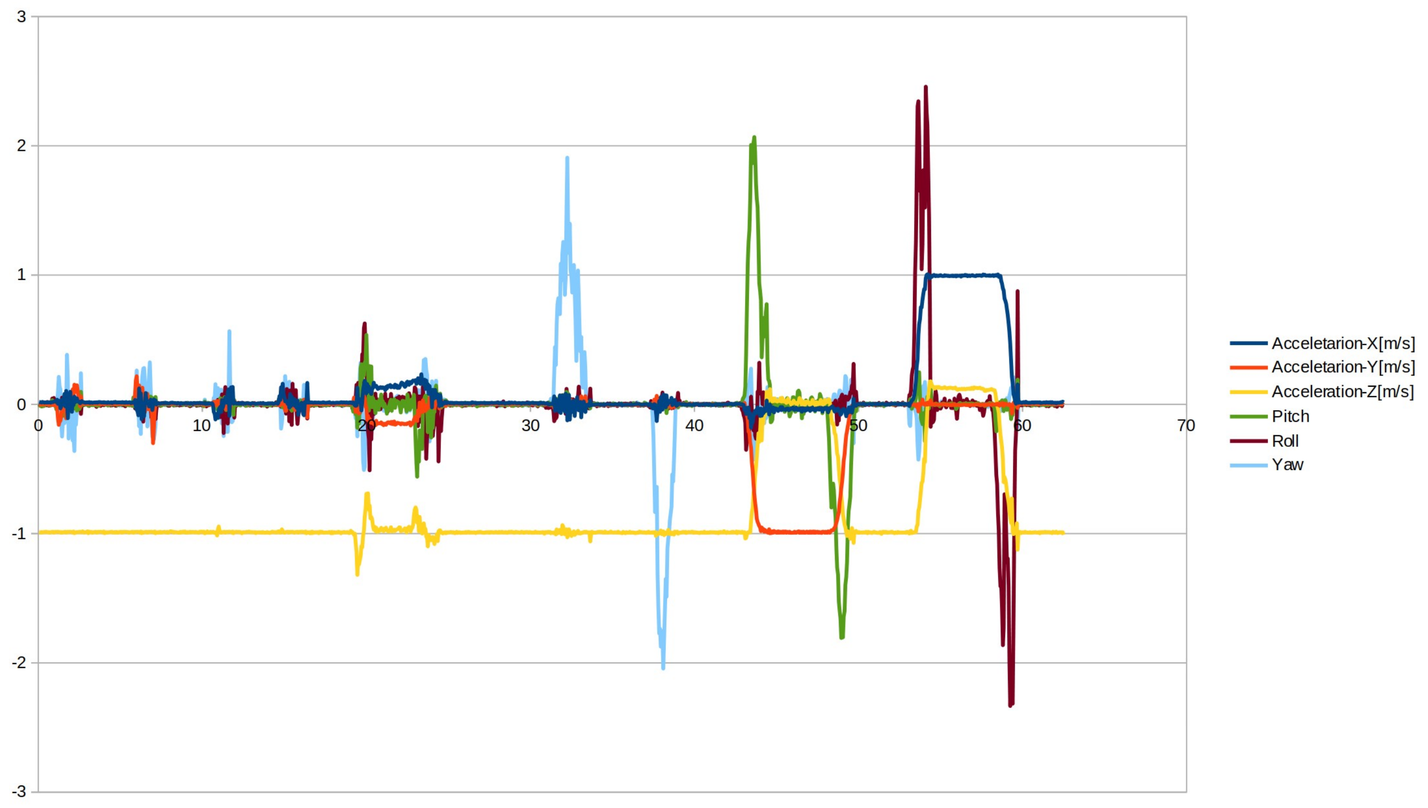
$$S^2 = \frac{\sum(X - \overline{X})^2}{n-1}$$

   2. After determining if the iPhone was moving, the data for Gyroscope was to be integrated once and Accelerometer data to be integrated twice. Gyroscope data was integrated by multiplying the data by the time between samples, in this case 0.05. For the Accelerometer, three values were calculated: velocity at end of sampling period, average velocity during the sampling period, and distance traveled during sampling period. Velocity was calculated by velocity at the time of previous sample plus acceleration reading multiplied by the time between samples (initial velocity is 0, and constant acceleration is assumed). Average velocity was calculated by using average of the initial and final velocities of sampling period. The distance traveled was calculated by using the average velocity multiplied by the time between samples during that sampling period.

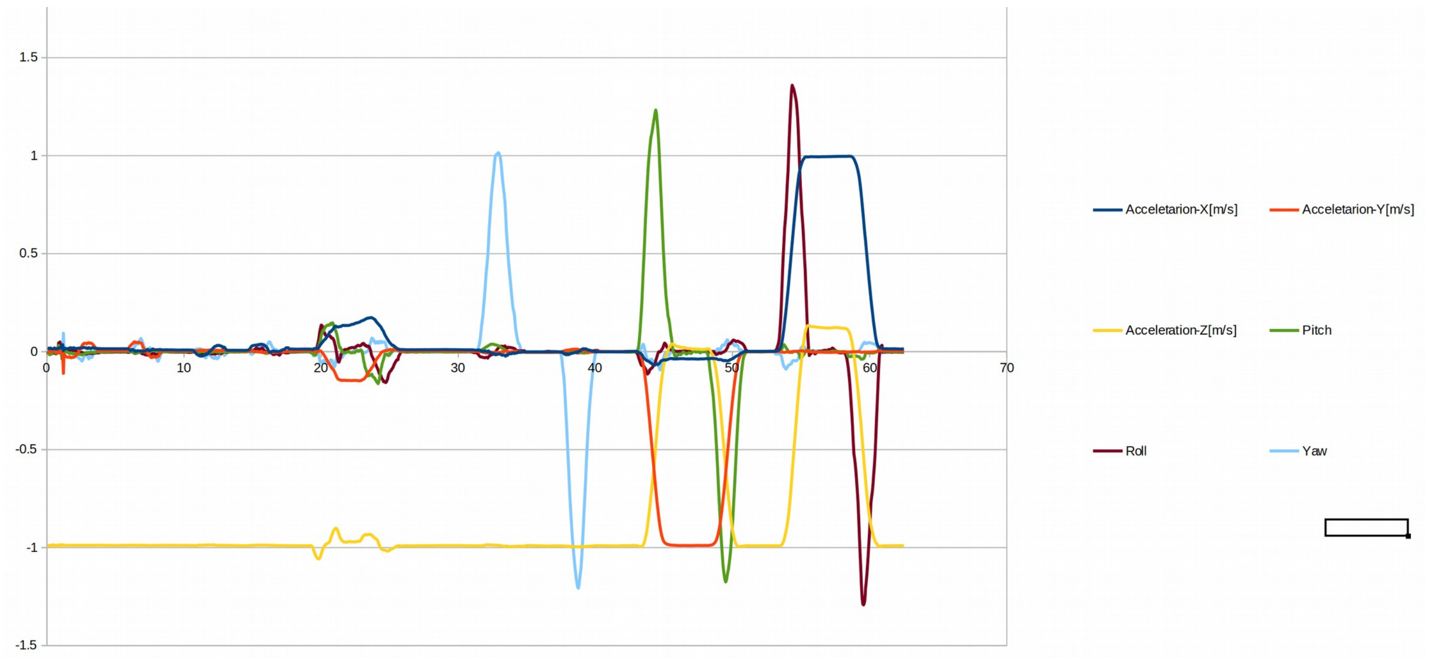
All C code can be seen at end of the report.

RESULTS

RAW DATA:



SMOOTH DATA (WINDOW SIZE 25) :



Movement Segments:

Accelerometer Threshold: 0.0009 | Gyroscope Threshold: 0.03 | Variance Window Size: 11

Start Index	Stop Index	Start Time	End Time	X [m]	Y [m]	Z [m]
14	49	0.65	2.4	0.256064	-0.316698	-14.850183
109	142	5.4	7.05	0.067466	0.536124	-13.201827
206	238	10.25	11.85	-0.217162	0.081988	-12.399938
286	329	14.25	16.4	0.620282	-0.025468	-22.390909
378	407	18.85	20.3	0.325531	-0.030281	-10.653658
448	487	22.35	24.3	3.054094	-1.728742	-17.662994
622	663	31.05	33.1	-0.106901	-0.110559	-20.372381
742	775	37.05	38.7	-0.13283	0.117453	-13.269423
854	889	42.65	44.4	-0.655107	-4.136676	-12.600515
956	991	47.75	49.5	-0.598791	-13.440938	-2.957079
1059	1088	52.9	54.35	2.599228	-0.019205	-9.004156
1158	1195	57.85	59.7	14.997234	-0.016012	-2.800284
	Total Distance:			20.209107	-19.089014	-152.163346

Start Index	Stop Index	Start Time	End Time	Pitch [radia	Roll [radians	Yaw [radians]
14	49	0.65	2.4	-0.017767	0.001065	-0.046859
109	142	5.4	7.05	0.00121	-0.015365	0.041551
206	238	10.25	11.85	-0.001187	-0.00968	-0.021367
286	329	14.25	16.4	-0.004582	0.002037	0.042248
378	407	18.85	20.3	0.160875	0.106606	-0.057485
448	487	22.35	24.3	-0.181188	-0.113467	0.050882
622	663	31.05	33.1	0.046811	-0.002195	1.506832
742	775	37.05	38.7	-0.001093	-0.00997	-1.505605
854	889	42.65	44.4	1.57538	-0.09729	-0.08077
956	991	47.75	49.5	-1.520516	0.008876	0.052772
1059	1088	52.9	54.35	-0.005122	1.702601	-0.073388
1158	1195	57.85	59.7	-0.038776	-1.604323	0.044563
	Total Angular Rotation:			0.014047	-0.031105	-0.046625

Accelerometer Threshold: 0.0009 | Gyroscope Threshold: 0.03 | Variance Window Size: 20

Start Index	Stop Index	Start Time	End Time	X [m]	Y [m]	Z [m]
6	48	0.25	2.35	0.352533	-0.309568	-21.381299
100	142	4.95	7.05	0.191492	0.578579	-21.376094
197	237	9.8	11.8	-0.150589	0.105197	-19.386427
277	329	13.8	16.4	0.68473	0.018411	-32.78159
369	407	18.4	20.3	0.434299	-0.003757	-17.976666
440	485	21.95	24.2	3.981427	-2.823389	-23.638944
613	661	30.6	33	-0.00152	-0.075387	-27.943726
733	774	36.6	38.65	-0.139125	0.087022	-20.451682
845	889	42.2	44.4	-0.641987	-4.129016	-21.225546
947	991	47.3	49.5	-0.909985	-22.070812	-2.855779
1050	1088	52.45	54.35	2.612688	-0.019962	-16.336424
1150	1195	57.45	59.7	23.021133	-0.03924	-1.887527
	Total Distance:			29.435096	-28.681921	-227.241702

Start Index	Stop Index	Start Time	End Time	Pitch [radia	Roll [radians]	Yaw [radians]
6	48	0.25	2.35	-0.017175	0.000298	-0.044526
100	142	4.95	7.05	0.001452	-0.01667	0.040136
197	237	9.8	11.8	-0.000551	-0.012135	-0.013817
277	329	13.8	16.4	-0.005372	0.001318	0.041893
369	407	18.4	20.3	0.16069	0.105549	-0.059015
440	485	21.95	24.2	-0.192021	-0.093844	0.049514
613	661	30.6	33	0.047987	-0.0047	1.4599
733	774	36.6	38.65	-0.001611	-0.010253	-1.4764
845	889	42.2	44.4	1.575496	-0.097429	-0.080252
947	991	47.3	49.5	-1.513114	0.00934	0.051571
1050	1088	52.45	54.35	-0.005032	1.702818	-0.073442
1150	1195	57.45	59.7	-0.040427	-1.616448	0.043937
	Total Angular Rotation:			0.010323	-0.032157	-0.060501

```c
/*  Name: Rodrigo Ignacio Rojas Garcia
    Lab #7
*/

// Library Declaration Section
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

// Define Declaration Section
#define MAXLENGTH 256
#define ACC_THRESHOLD_X 0.0009
#define ACC_THRESHOLD_Y 0.0009
#define ACC_THRESHOLD_Z 0.0009
#define GYRO_THRESHOLD_PITCH 0.03
#define GYRO_THRESHOLD_ROLL 0.03
#define GYRO_THRESHOLD_YAW 0.03
#define SMOOTH_WINDOW_SIZE 25
#define VARIANCE_WINDOW_SIZE 11
#define SAMPLE_TIME 0.05
#define GRAVITY 9.81

// READ IN TEXT FILE AND EXTRACT DATA
void read_text_file(char *file_name, int *file_size, double **time, double **accX,
double **accY,
                    double **accZ, double **pitch, double **roll, double **yaw)
{
    // VARIABLE DECLARATION SECTION
    FILE *file;
    int i = 0;
    double d1, d2, d3, d4, d5, d6, d7;
    char c;
    char line[MAXLENGTH];
    *file_size = 0;

    // OBTAINS FILE LENGTH AND REWINDS IT TO BEGINNING
    file = fopen(file_name, "r");
    if (file == NULL)
    {
        printf("Error, could not read in initial contour text file\n");
        exit(1);
    }
    while((c = fgetc(file)) != EOF)
    {
        if (c == '\n')
        {
            *file_size += 1;
        }
    }
    rewind(file);

    // SKIPS FIRST LINE OF FILE
    fgets(line, sizeof(line), file);

    // ALLOCATES MEMORY
    *time = calloc(*file_size, sizeof(double *));
    *accX = calloc(*file_size, sizeof(double *));
    *accY = calloc(*file_size, sizeof(double *));
    *accZ = calloc(*file_size, sizeof(double *));
    *pitch = calloc(*file_size, sizeof(double *));
    *roll = calloc(*file_size, sizeof(double *));
    *yaw = calloc(*file_size, sizeof(double *));

    // EXTRACTS TDATA FROM TEXT FILE
    while((fscanf(file, "%lf %lf %lf %lf %lf %lf %lf\n", &d1, &d2, &d3, &d4, &d5, 
&d6, &d7)) != EOF)
    {
        (*time)[i] = d1;
        (*accX)[i] = d2;
        (*accY)[i] = d3;
        (*accZ)[i] = d4;
        (*pitch)[i] = d5;
        (*roll)[i] = d6;
        (*yaw)[i] = d7;
        i++;
    }
    fclose(file);

    // CREATE CSV FILE
    file = fopen("initial-data.csv", "w");
    fprintf(file, "Time[s],Acceletarion-X[m/s],Acceletarion-Y[m/s],Acceleration-Z[m/
s],Pitch,Roll,Yaw\n");
    for (i = 0; i < *file_size; i++)
    {
        fprintf(file, "%lf,%lf,%lf,%lf,%lf,%lf,%lf\n", (*time)[i], (*accX)[i], (*accY)
[i], (*accZ)[i], (*pitch)[i], (*roll)[i], (*yaw)[i]);
    }
    fclose(file);
}

// SMOOTH DATA POINTS
void smooth_data(int arr_length, double **data, double **smoothed_data)
{
    // VARIABLE DECLARATION SECTION
    int i, j;
    double sum;

    // ALLOCATE MEMORY
    *smoothed_data = calloc(arr_length, sizeof(double *));

    for (i = 0; i < SMOOTH_WINDOW_SIZE; i++)
    {
        (*smoothed_data)[i] = (*data)[i];
    }

    for (i = SMOOTH_WINDOW_SIZE - 1; i < arr_length; i++)
    {
        sum = 0;
        for (j = 1; j < SMOOTH_WINDOW_SIZE; j++)
        {
            sum += (*data)[i - j];
        }
        (*smoothed_data)[i] = (sum + (*data)[i]) / SMOOTH_WINDOW_SIZE;
    }
}

// CALCULATE VARIANCE
double calc_variance(double **data, int arr_length, int index)
{
    // VARIABLE DECLARATION SECTION
    int i;
    int window = 0;
    double mean = 0;
    double variance = 0;

    // HANDLES ERROR WHEN WINDOW SIZE IS TOO BIG
    if ((index + VARIANCE_WINDOW_SIZE) < arr_length)
    {
        window = VARIANCE_WINDOW_SIZE;
    }
    else
    {
        window = abs(arr_length - index);
    }

    // CALCULATE THE SUM OF DATA SET
    for (i = index; i < (index + window); i++)
    {
        variance += (*data)[i];
    }

    // USED SUM CALCULATED TO OBTAIN MEAN OF DATA SET
    mean = variance / window;
    variance = 0;

    // SUBTRACT MEAN FOR THE DATA SET AND SUM RESULTS
    for (i = index; i < (index + window); i++)
    {
        variance += pow(((*data)[i] - mean), 2);
    }

    // CALCULATE VARIANCE OF EACH DATA SET
    variance = variance / (window - 1);

    return variance;
}

// DETERMINES IF OBJECT HAS MOVED AND STORES RESULTS ON TEXT AND CSV FILES
void move_or_still(double **accX, double **accY, double **accZ, double **pitch, double
**roll, double **yaw, int arr_length)
{
    // VARIABLE DECLARATION SECTION
    FILE *text_file, *csv_file;
    int i, j, k;
    int start_movement = 0;
    int end_movement = 0;
    int moving = 0;
    double start_time = 0;
    double end_time = 0;
    double acc_variance[3];
    double gyro_variance[3];
    double gyro_distance[3];
    double velocity[3];
    double distance_traveled[3];
    double previous_velocity[3];
    double sum[6];
    char file_name[MAXLENGTH];
    k = 1;

    sprintf(file_name, "movement-%d.txt", VARIANCE_WINDOW_SIZE);
    text_file = fopen(file_name, "w");
    sprintf(file_name, "movement-%d.csv", VARIANCE_WINDOW_SIZE);
    csv_file = fopen(file_name, "w");
    fprintf(csv_file, "Start Index,Stop Index,Start Time,End Time,X [m],Y [m],Z
[m],Pitch [radians],Roll [radians],Yaw [radians]\n");
```

```
    for (i = 0; i < 6; i++)
    {
        sum[i] = 0;
    }


    for (i = 0; i < arr_length; i++)
    {
        for(j = 0; j < 3; j++)
        {
            acc_variance[j] = 0;
            gyro_variance[j] = 0;
            gyro_distance[j] = 0;
            velocity[j] = 0;
            distance_traveled[j] = 0;
            previous_velocity[j] = 0;
        }

        // CALCULATE VARIANCE FOR BOTH ACCELOREMETER AND GYROSCOPE
        acc_variance[0] = calc_variance(accX, arr_length, i);
        acc_variance[1] = calc_variance(accY, arr_length, i);
        acc_variance[2] = calc_variance(accZ, arr_length, i);
        gyro_variance[0] = calc_variance(pitch, arr_length, i);
        gyro_variance[1] = calc_variance(roll, arr_length, i);
        gyro_variance[2] = calc_variance(yaw, arr_length, i);

        // IF TRUE, ACCELOREMETER IN MOTION
        if ((acc_variance[0] > ACC_THRESHOLD_X) || (acc_variance[1] >
ACC_THRESHOLD_Y)
            || (acc_variance[2] > ACC_THRESHOLD_Z))
        {
            moving = 1;
        }

        // IF TRUE, GYROSCOPE IN MOTION
        if ((gyro_variance[0] > GYRO_THRESHOLD_PITCH) || (gyro_variance[1] >
GYRO_THRESHOLD_ROLL)
            || (gyro_variance[2] > GYRO_THRESHOLD_YAW))
        {
            moving = 1;
        }

        // CHECKS WHEN MOVEMENT STARTED AND ENDED
        if (start_movement == 0 && moving == 1)
        {
            start_movement = i;
            start_time = start_movement * SAMPLE_TIME;
        }
        if (end_movement == 0 && moving == 0 && start_movement != 0)
        {
            end_movement = i;
            end_time = end_movement * SAMPLE_TIME;
        }

        // ONCE MOVEMENT PERIOD OBTAINED, GYROSCOPE AND ACCELERATION INTEGRATION IS
CALCULATED
        if (start_movement != 0 && end_movement != 0)
        {
            // GYROSCOPE INTEGRATION
            for (j = start_movement; j < end_movement; j++)
            {
                gyro_distance[0] += ((*pitch)[j] * SAMPLE_TIME);
                gyro_distance[1] += ((*roll)[j] * SAMPLE_TIME);
                gyro_distance[2] += ((*yaw)[j] * SAMPLE_TIME);
            }

            // ACCELEROMETER DOUBLE INTEGRATION
            for (j = start_movement; j < end_movement; j++)
            {
                previous_velocity[0] = velocity[0];
                velocity[0] += ((*accX)[j] * GRAVITY * SAMPLE_TIME);
                distance_traveled[0] += (((velocity[0] + previous_velocity[0]) / 2) *
SAMPLE_TIME);

                previous_velocity[1] = velocity[1];
                velocity[1] += ((*accY)[j] * GRAVITY * SAMPLE_TIME);
                distance_traveled[1] += (((velocity[1] + previous_velocity[1]) / 2) *
SAMPLE_TIME);

                previous_velocity[2] = velocity[2];
                velocity[2] += ((*accZ)[j] * GRAVITY * SAMPLE_TIME);
                distance_traveled[2] += (((velocity[2] + previous_velocity[2]) / 2) *
SAMPLE_TIME);

            }

            sum[0] += distance_traveled[0];
            sum[1] += distance_traveled[1];
            sum[2] += distance_traveled[2];
            sum[3] += gyro_distance[0];
            sum[4] += gyro_distance[1];
            sum[5] += gyro_distance[2];

            // TEXT FILE PRINTS
            fprintf(text_file, "-------------------------------------\n");
            fprintf(text_file, "Movement #%d:\n", k);
            fprintf(text_file, "Movement X-axis: %.6f[m]\nMovement Y-axis: %.6f[m]
\nMovement Z-axis: %.6f[m]\n", distance_traveled[0], distance_traveled[1],
distance_traveled[2]);
            fprintf(text_file, "Movement Pitch: %.6f[radians]\nMovement Roll: %.
6f[radians]\nMovement Yaw: %.6f[radians]\n", gyro_distance[0], gyro_distance[1],
gyro_distance[2]);
            fprintf(text_file, "Movement Start Time: %.2f | Movement End Time: %.
2f\n", start_time, end_time);
            fprintf(text_file, "Movement Start Index: %d | Movement End Index: %d\n",
start_movement, end_movement);
            fprintf(text_file, "-------------------------------------\n\n");

            // CSV FILE PRINTS
            fprintf(csv_file, "%d,%d,%.2f,%.2f,%.6f,%.6f,%.6f,%.6f,%.6f,%.6f\n",
                    start_movement + 1, end_movement + 1, start_time, end_time,
                    distance_traveled[0], distance_traveled[1], distance_traveled[2],
                    gyro_distance[0], gyro_distance[1], gyro_distance[2]);

            start_movement = 0;
            end_movement = 0;
            start_time = 0;
            end_time = 0;
            k++;
        }
        moving = 0;
    }
    fprintf(csv_file,"Total Distance:,,,,%.6f,%.6f,%.6f,%.6f,%.6f,%.6f\n",
sum[0],sum[1],sum[2],sum[3],sum[4],sum[5]);
    fclose(text_file);
    fclose(csv_file);
}

int main(int argc, char *argv[])
{
    /* VARIABLE DECLARATION SECTION */
    FILE *file;
    int file_size;
    int i;
    double *time, *accX, *accY, *accZ, *pitch, *roll, *yaw;
    double *smooth_accX, *smooth_accY, *smooth_accZ, *smooth_pitch, *smooth_roll,
*smooth_yaw;

    if (argc != 2)
    {
        printf("Usage: ./executable text_file.txt\n");
        exit(1);
    }

    /* EXTRACT DATA FROM TEXT FILE */
    read_text_file(argv[1], &file_size, &time, &accX, &accY, &accZ, &pitch, &roll,
&yaw);

    /* SMOOTH EXTRACTED DATA */
    smooth_data(file_size, &accX, &smooth_accX);
    smooth_data(file_size, &accY, &smooth_accY);
    smooth_data(file_size, &accZ, &smooth_accZ);
    smooth_data(file_size, &pitch, &smooth_pitch);
    smooth_data(file_size, &roll, &smooth_roll);
    smooth_data(file_size, &yaw, &smooth_yaw);

    /* EXPORT SMOOTHED DATA AS CSV */
    file = fopen("smoothed_data.csv", "w");
    fprintf(file, "Time[s],Acceletarion-X[m/s],Acceletarion-Y[m/s],Acceleration-Z[m/
s],Pitch,Roll,Yaw\n");
    for (i = 0; i < file_size; i++)
    {
        fprintf(file, "%lf,%lf,%lf,%lf,%lf,%lf,%lf\n", time[i], smooth_accX[i],
smooth_accY[i], smooth_accZ[i], smooth_pitch[i], smooth_roll[i], smooth_yaw[i]);
    }
    fclose(file);

    /* DETERMINE MOVEMENT */
    move_or_still(&accX, &accY, &accZ, &pitch, &roll, &yaw, file_size);

    return 0;
}
```