ECE 6310

Introduction to Computer Vision

Fall 2022

Lab 6

Motion Tracking

Harshal B. Varpe

Introduction

This lab is concerned with automatically segmenting given data into periods of motion and rest using the data collected from accelerometers and gyroscopes. The data was collected using an iPhone. The accelerometer data measures g force in X, Y, and Z directions. The gyroscope measures roll, pitch and yaw. The data was collected at ad 20 Hz frequency. We can use the accelerometer data to calculate the distance of movement. For this purpose, accelerometer data must be integrated twice. Moreover, since the sampling frequency is not high enough, velocity calculation may not be accurate enough, reducing accuracy in distance measurement. By integrating the gyroscope data, we can find the degree of rotation i.e. pitch, roll, and yaw in radians.

Implementation / Methods

I started by reading the text file of the given data. The given data had 1251 data points in 7 columns. Each column represented time, the accelerometer reading in X, the accelerometer reading in Y, the accelerometer reading in Z, pitch, roll, and yaw, respectively. For segmenting the data, variance in the data for each of the quantities needed to be calculated. The variance calculated was stored in the CSV file. Using this file, I plotted a simple line graph with a number of the data point on the X-axis and variance value on the Y-axis. Figure 1 shows an example of such a graph.

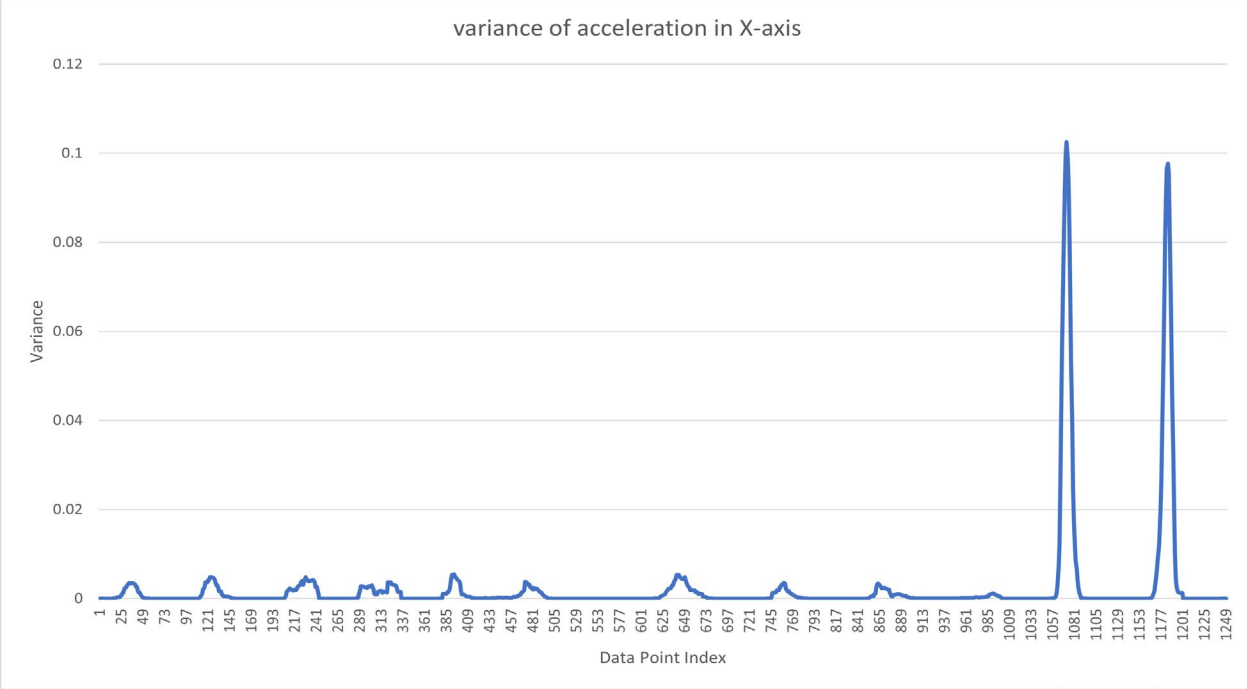

Figure 1 – How I decided the threshold values for each of the quantities

Using such graphs, I decided on the threshold for each quantity.

If the variance of a data point of a particular quantity was greater than the threshold for that quantity, then the phone was said to be in motion; otherwise, it was resting. If the phone was said to be in motion, then double integration of accelerometer data and single integration of gyroscope data was done and sent out to the CSV file. This CSV file contained the start index, stop index, start time, stop

time, and distance in X, Y, and Z, pitch, roll, and yaw during that motion period. For the rest period, only the start and stop times, along with indices, were recorded.

A total of 12 motions were found. The table below shows both the rest and motion periods.

State	start_idx	stop_idx	start_time	stop_time	dist_x(m)	dist_y(m)	dist_z(m)	pitch(rad)	roll(rad)	yaw(rad)
rest	0	16	0.05	0.85						
move	16	59	0.85	3	0.406029	-0.236764	-22.408235	-0.015601	-0.002523	-0.025836
rest	59	110	3	5.55						
move	110	150	5.55	7.55	0.092237	0.624258	-19.39885	0.001236	-0.023356	0.0215
rest	150	206	7.55	10.35						
move	206	244	10.35	12.25	-0.201117	0.124855	-17.490764	-0.003269	-0.013998	-0.027016
rest	244	286	12.25	14.35						
move	286	335	14.35	16.8	0.763667	-0.05356	-29.081781	-0.004254	0.001629	0.041848
rest	335	377	16.8	18.9						
move	377	498	18.9	24.95	18.577558	-14.497411	-175.287625	-0.011974	-0.036922	-0.020487
rest	498	621	24.95	31.1						
move	621	680	31.1	34.05	-0.322209	-0.173315	-42.232474	0.045546	0.005694	1.567627
rest	680	742	34.05	37.15						
move	742	783	37.15	39.2	-0.142198	0.176683	-20.49362	-0.004103	-0.003566	-1.540908
rest	783	853	39.2	42.7						
move	853	1001	42.7	50.1	-11.380061	-202.430169	-78.686839	0.002165	-0.026866	-0.02147
rest	1001	1054	50.1	52.75						
move	1054	1101	52.75	55.1	9.746275	-0.021305	-19.727023	0.001572	1.672613	-0.070368
rest	1101	1112	55.1	55.65						
move	1112	1127	55.65	56.4	2.742092	-0.009238	0.338924	-0.00278	-0.001376	0.001498
rest	1127	1144	56.4	57.25						
move	1144	1148	57.25	57.45	0.195577	-0.000182	0.024634	0.000027	-0.002637	0.00077
rest	1148	1149	57.45	57.5						
move	1149	1201	57.5	60.1	28.882653	-0.053418	-4.947092	-0.038379	-1.617972	0.045119
rest	1201	1249	60.1	62.5						

Table 1. Motion (in green) and Rest (in black) periods, along with distances traveled in X, Y, and Z and roll, pitch, and yaw

However, the distance traveled and rotation about the axis does not make much sense. For example, for the first data point, we can see that phone moved 22 meters or roughly 72 feet in the negative z direction.

One of the problems with accelerometers is that their working depends on which axis is under the effect of gravity. Under normal circumstances, the Z axis is considered to be under the influence of gravity. However, when the orientation of the accelerometer is changed (or, in our case, the phone is flipped), this is not the case. Accelerometers cannot differentiate between acceleration due to gravity and applied acceleration. Moreover, accelerometer values are also affected by angular rotation as well. This causes errors in the measurement of acceleration. Since we are double integrating the acceleration to calculate distance, the error in distance will be huge. This is what we see here in the table.

Similarly working of the gyroscope is also affected by the earth's magnetic field. A gyroscope is erroneous, which is why it is used in conjunction with a magnetometer.

Code

```
/*
ECE 6310
Motion tracking
Harshal Varpe
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define SQR(x) ((x)*(x))
#define thresh_acc 0.0001
#define thresh_gy 0.001
#define sample_t 0.05
#define g 9.81

int main(int argc, char *argv[]){
    FILE *fpt,*fpt1;
    int data_pts, filt_win ,x,y;
    char
head1[320],head2[320],head3[320],head4[320],head5[320],head6[320],head7[320];
    double
t[1300],acc_x[1300],acc_y[1300],acc_z[1300],roll[1300],pitch[1300],yaw[1300]; //
for no header data
    // long double
t[640],acc_x[640],acc_y[640],acc_z[640],roll[640],pitch[640],yaw[640]; //
unusable
    double
f_acc_x[1300],f_acc_y[1300],f_acc_z[1300],f_roll[1300],f_pitch[1300],f_yaw[1300];
    double sum_x, sum_y, sum_z, sum_p, sum_r, sum_yaw, avx, avy, avz, avp, avr,
avyaw;
    double temp_x, temp_y, temp_z, temp_r, temp_p, temp_yaw;
    double
var_accx[1300],var_accy[1300],var_accz[1300],var_pitch[1300],var_roll[1300],var_y
aw[1300];

    double start_t, stop_t,start_rest,stop_rest;
    double int_acc[3],in_vel[3],vel[3];
    double int_pitch, int_roll, int_yaw ;
    int start_idx, stop_idx, start_rest_idx, stop_rest_idx, move;


    fpt = fopen("acc_gyro.txt","r");
    if(fpt == NULL){
        printf("Unable to open IMU readings!\n");
```

```
        exit(0);
    }

    fscanf(fpt,"%s %s %s %s %s %s %s",head1,head2,head3,head4,head5,head6,head7);

    data_pts = 0;    // calculating the number of data points
    while(fscanf(fpt,"%lf %lf %lf %lf %lf %lf %lf
",&t[data_pts],&acc_x[data_pts],&acc_y[data_pts],&acc_z[data_pts],&pitch[data_pts
],&roll[data_pts],&yaw[data_pts]) != EOF){
        data_pts++;
    }
    fclose(fpt);

// variance calculation
int var_win = 15;

// Just like mean calculation above we will deal with edge cases first and then
move on to
// variance calculation.
for(x=0;x<var_win/2;x++){
    var_accx[x] = 0;
    var_accy[x] = 0;
    var_accz[x] = 0;
    var_pitch[x] = 0;
    var_roll[x] = 0;
    var_yaw[x] = 0;
}
for(x=data_pts-(var_win/2);x<data_pts;x++){
    var_accx[x] = 0;
    var_accy[x] = 0;
    var_accz[x] = 0;
    var_pitch[x] = 0;
    var_roll[x] = 0;
    var_yaw[x] = 0;
}
for(x=var_win/2;x<(data_pts - (var_win/2));x++){
        sum_x = 0; sum_y =0; sum_z=0;
        sum_p = 0; sum_r = 0; sum_yaw=0;
        for(y=(-var_win/2);y<=(var_win/2);y++){
            sum_x = sum_x + acc_x[x+y] ;
            sum_y = sum_y + acc_y[x+y] ;
            sum_z = sum_z + acc_z[x+y] ;
            sum_p = sum_p + pitch[x+y] ;
            sum_r = sum_r + roll[x+y] ;
            sum_yaw = sum_yaw + yaw[x+y] ;
        }
        avx = sum_x / var_win;
```

```
        avy = sum_y / var_win;
        avz = sum_z / var_win;
        avp = sum_p / var_win;
        avr = sum_r / var_win;
        avyaw = sum_yaw / var_win;
        temp_x = temp_y = temp_z = temp_r = temp_p = temp_yaw = 0;
        for(y=(-var_win/2);y<=(var_win/2);y++){
            temp_x = temp_x + SQR(acc_x[x+y] - avx) ;
            temp_y = temp_y + SQR(acc_y[x+y] - avy) ;
            temp_z = temp_z + SQR(acc_z[x+y] - avz) ;
            temp_p = temp_p + SQR(pitch[x+y] - avp) ;
            temp_r = temp_r + SQR(roll[x+y] - avr) ;
            temp_yaw = temp_yaw + SQR(yaw[x+y] - avyaw) ;
        }
        var_accx[x] = temp_x / (var_win - 1);
        var_accy[x] = temp_y / (var_win - 1);
        var_accz[x] = temp_z / (var_win - 1);
        var_pitch[x] = temp_p / (var_win - 1);
        var_roll[x] = temp_r / (var_win - 1);
        var_yaw[x] = temp_yaw / (var_win - 1);
    }

    fpt = fopen("variance.csv","w");
    fprintf(fpt," %s, %s, %s, %s, %s, %s, %s
\n","time","var_acx","var_acy","var_acz","pitch","var_roll","vra_yaw");
    for(x = 0; x < data_pts; x++){
        fprintf(fpt," %lf, %lf, %lf, %lf, %lf, %lf, %lf
\n",t[x],var_accx[x],var_accy[x],var_accz[x],var_pitch[x],var_roll[x],var_yaw[x])
;}
    fclose(fpt);
// printf("Good till here");


start_rest_idx = 1; // index 1 means not resting
move = 0;
start_t = stop_t = stop_rest = start_idx = stop_idx = start_rest = 0;
stop_rest_idx = 0;

int_acc[0] = int_acc[1] = int_acc[2] = 0;
int_pitch = int_roll = int_yaw = 0;
in_vel[0] = in_vel[1] = in_vel[2] = 0;
vel[0] = vel[1] = vel[2] = 0;
//Write out the data in txt file
fpt = fopen("output.csv","w");
fprintf(fpt,"%s  , %s  , %s  ,%s  ,%s  ,%s  ,%s  ,%s  ,%s  ,%s  ,%s  \n",
"state","start_idx","stop_idx","start_time","stop_time","dist_x(m)","dist_y(m)","
dist_z(m)","pitch(rad)","roll(rad)","yaw(rad)");
```

```
//(work till here)
for(x=0;x<data_pts;x++){
    // if the variance is above threshold motion is detected
    if(var_accx[x]>thresh_acc || var_accy[x]>thresh_acc || var_accz[x]>thresh_acc
|| var_pitch[x]>thresh_gy || var_roll[x]>thresh_gy || var_yaw[x]>thresh_gy){
        move = 1;
        // printf("%d",x);
    }
    else{move = 0;
    //printf("%d",x);
    }
    // if move is 1 and start_idx is zero, then start_idx
    if(move == 1 && start_idx == 0){start_idx = x; start_t = t[x];}

    //if it is moving, stop_rest_idx is zero, and start_rest-idex is not what
initilised
    //(meaning we are in rest period) ,  then stop_rest_idx
    // if(move == 1 || start_rest_idx != 2 && stop_rest_idx == 0 )
    // if(move == 1 && stop_rest_idx == 0 || start_rest_idx != 2 && x == data_pts
- 1)
    if((move == 1 && stop_rest_idx == 0 && start_rest_idx != 1) || x == data_pts
- 1)
    {stop_rest_idx=x; stop_rest = t[x];}

    // if it is not moving, and stop index is zero but start index is not zero,
then stop index
    if(move == 0 && stop_idx == 0 && start_idx != 0){stop_idx = x ; stop_t =
t[x];}

    // if it is not moving and start_rest_index is initilised value, then start
rest
    if(move == 0 && start_rest_idx == 1){start_rest_idx = x; start_rest = t[x];}

    if(start_idx != 0 && stop_idx != 0){

        for(y = start_idx; y<stop_idx; y++){
            int_pitch = int_pitch + pitch[y] * sample_t ;
            int_roll = int_roll + roll[y] * sample_t ;
            int_yaw = int_yaw + yaw[y] * sample_t ;

            in_vel[0] = vel[0]; in_vel[1] = vel[1]; in_vel[2] = vel[2];

            vel[0] = vel[0] + (acc_x[y] * g * sample_t);
            vel[1] = vel[1] + (acc_y[y] * g * sample_t);
            vel[2] = vel[2] + (acc_z[y] * g * sample_t);
```

```
            int_acc[0] = int_acc[0] + (((in_vel[0] + vel[0])/2)*sample_t); //
dist x
            int_acc[1] = int_acc[1] + (((in_vel[1] + vel[1])/2)*sample_t); //
dist y
            int_acc[2] = int_acc[2] + (((in_vel[2] + vel[2])/2)*sample_t); //
dist z

        }
        fprintf(fpt,"%s   ,%d   ,%d   ,%lf   ,%lf   ,%lf   ,%lf   ,%lf   ,%lf   ,
%lf   ,%lf
\n","move",start_idx,stop_idx,start_t,stop_t,int_acc[0],int_acc[1],int_acc[2],int
_pitch,int_roll,int_yaw);
        start_idx = stop_idx = start_t = stop_t = 0;
        int_acc[0] = int_acc[1] = int_acc[2] = 0;
        // int_gy[0] = int_gy[1] = int_gy[2] = 0; // may not need this
        int_pitch = int_roll = int_yaw = 0;
        in_vel[0] = in_vel[1] = in_vel[2] = 0;
        vel[0] = vel[1] = vel[2] = 0;
    }
    if(start_rest_idx != 1 && stop_rest_idx != 0){

        fprintf(fpt,"%s   ,%d   ,%d   ,%lf   ,%lf  \n","rest",start_rest_idx,stop
_rest_idx,start_rest,stop_rest);
        stop_rest_idx = stop_rest = start_rest = 0;
        start_rest_idx = 1;
    }
    move = 0;
}
fclose(fpt);

}
```