**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.

Chart, application

Description automatically generated with medium confidence  
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.

  
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\_yaw[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);

}