**William Benton**

**Lab 7: Motion Tracking**

**11/25/2019**

**Background**

The purpose of this lab was to try to calculate the distance a phone traveled based on data from accelerometers in the X, Y, and Z directions and data from gyroscopes measuring pitch, roll, and yaw sampled every 0.05 seconds. The accelerometers give acceleration data. To calculate the total distance from the acceleration, the acceleration data must be integrated twice. Since the sample time is not infinitely small, there will be error in calculating the velocity. Since the velocity is used to calculate the position, the distance traveled cannot possibly be calculated accurately. The gyroscope data is in radians per second, so the rotation in radians can simply be calculated by multiplying the sample time by the value of the radians per second.

**Implementation**

I started by reading the data into one large 2D array. It was organized into 1250 rows by 7 columns. The columns represented the time, acceleration in the x direction, acceleration in the y direction, acceleration in the z direction, pitch, roll, and yaw. After the data is read in, to visualize the data a little bit better, the variance is calculated for every value of every data point and that data is printed to an excel file. Viewing this excel file was beneficial and was how I chose my threshold values. From there, I looped through the 1250 rows of the array that contained the 7 different data points. I calculated the variance of each category in the current row. Then a check was done to determine if the phone was moving. The instructions were to compare all 6 variances at once (acceleration X, acceleration Y, acceleration Z, pitch, roll, and yaw) to use threshold values. So using the graph of the variances, I determined some threshold values to capture the data that appeared to be movement. The instructions were to start recording movement if any of the values were greater than the threshold associated with that category. The problem with this will be discussed later in the results section. Once one of these categories exceeded their threshold, a start time was recorded. The loop continued through the data until all 6 categories fell back below their thresholds. Once that occurred, the end time was recorded. With the time frame, the integrals could be performed. After the integrals were performed, the data was printed out to a text file and the start and end times were cleared and the process started again.

**Results and Discussion**

A total of 9 movements were found. Below are the results of the calculations. All X, Y and Z data is measured in meters and all pitch, roll and yaw data is measured in radians.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **X Movement** | **Y Movement** | **Z Movement** | **Pitch Rotation** | **Roll Rotation** | **Yaw Rotation** | **Start Time** | **End Time** |
| 0.324663 | -0.28033 | 0.216977 | -0.016488 | -0.00299 | -0.02564 | 0.75 | 2.65 |
| 0.077216 | 0.556208 | 0.171221 | 0.001593 | -0.02214 | 0.022497 | 5.45 | 7.15 |
| -0.208693 | 0.086416 | 0.172758 | -0.004715 | -0.00977 | -0.02667 | 10.25 | 11.90 |
| 0.650188 | -0.024251 | 0.311491 | -0.003550 | 0.002754 | 0.042411 | 14.25 | 16.45 |
| 16.408960 | -12.96649 | 3.811162 | -0.013730 | -0.03350 | -0.02282 | 18.80 | 24.55 |
| -0.221626 | -0.147999 | 0.409759 | 0.046144 | 0.002978 | 1.566631 | 31.0 | 33.70 |
| -0.134216 | 0.136253 | 0.102390 | -0.001799 | -0.01122 | -1.54073 | 37.05 | 38.85 |
| -10.50441 | -186.3172 | 174.996948 | 0.009000 | -0.03555 | -0.02369 | 42.60 | 49.75 |
| 179.58739 | -0.274640 | 182.312164 | -0.037802 | 0.067978 | -0.02369 | 52.80 | 59.75 |

As you can see above, the rotational data did not turn out very well and this was the easier set of data to calculate. Why is this? The answer is in how it was determined the phone was in motion and more specifically, how the phone was determined to stop moving. Below is a graph of the variance for Z used to demonstrate the problem.

Figure : Variance in the Z Direction

The first big spike is where the phone was moved in the Z direction. The later spikes in data are all very important for the calculations of the rotational components. Since in order for the program to believe the phone has stopped moving, all categories must fall below the threshold. As you can see, the variance of in the Z direction stays relatively high. This is because the phone has now rotated and flipped which axis is feeling gravity. Since the Z axis is not feeling gravity, the variance is varying greatly. It is also not possible to raise the threshold because then the Z movement seen in the first spike would not be seen. This is why checking to make sure all values of the variance are within the threshold creates issues for the rotational components. Since the end time doesn’t occur until the phone has flipped back over, the integral occurs over the entirety of the rotation one direction and then the rotation back the other direction. Since the signal roughly mirrors itself rotating one direction and then rotating back, the result of the area under the curve is roughly zero. This is why the rotational fields are all so low especially for where the actual rotations occur. Along the same lines, the positional components vary wildly when rotation occurs because which accelerometer feels gravity changes. This will create a large spike in the acceleration data and will create a large integral and then a large second integral. This lab was very difficult to begin and debug because the right answer could not necessarily be found but it taught a lot about why we cannot accurately estimate movement using accelerometers. It also taught about the difficulty of working with accelerometers and gyroscopes and the difficulty associated with having one axis feel the effects of gravity and then have that axis change.

**Code**

*MotionTracking.h*

enum {

TIME,

ACCX,

ACCY,

ACCZ,

PITCH,

ROLL,

YAW

};

int DATA\_SIZE;

#define VAR\_WINDOW 10

#define ACC\_THRESH 0.0001

#define PITCH\_THRESH 0.005

#define ROLL\_THRESH 0.05

#define YAW\_THRESH 0.005

#define SAMPLE\_TIME 0.05

#define GRAVITY 9.81

float variance (float \*input, int curr\_index);

void PrintData(FILE \*fp, float distance[7], float start\_time, float end\_time, int start\_index, int end\_index);

float integrate\_acc(float \*input, int start, int end);

float integrate(float \*input, int start, int end);

bool isMoving(float var\_arr[7]);

*MotionTracking.c*

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#include <stdbool.h>

#include "MotionTracking.h"

#define SQR(x) ((x)\*(x))

int main()

{

float \*\*Table;

FILE \*fpt;

FILE \*fpt2;

FILE \*Output;

bool moving;

int i, j;

char c;

float var;

float dist\_arr[7] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0};

float var\_arr[7] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0};

float start\_time = 0.0;

float end\_time = 0.0;

float startIndex = 0;

float endIndex = 0;

// Read beginning contour points

if ((fpt=fopen("acc\_gyro.txt", "rb")) == NULL)

{

printf("Unable to open data file\n");

exit(0);

}

i = 0;

for (c = getc(fpt); c != EOF; c = getc(fpt))

{

if (c == '\n')

i++;

}

DATA\_SIZE = i;

rewind(fpt);

Table = (float \*\*)malloc(7\*sizeof(float \*));

for (i = 0; i < 7; i++)

{

Table[i] = (float \*)malloc(DATA\_SIZE\*sizeof(float));

}

i = 0;

while(!feof(fpt) && !(i == DATA\_SIZE))

{

if (i == 0)

{

fscanf(fpt, "time accX accY accZ pitch roll yaw \n");

}

else

{

fscanf(fpt, "%f %f %f %f %f %f %f \n", &Table[TIME][i-1], &Table[ACCX][i-1], &Table[ACCY][i-1], &Table[ACCZ][i-1], &Table[PITCH][i-1], &Table[ROLL][i-1], &Table[YAW][i-1]);

}

i++;

}

for (i = 0; i < DATA\_SIZE; i++)

{

Table[ACCZ][i] += 1;

}

fpt2 = fopen("variance.csv", "w");

fprintf(fpt2, "TIME,ACCX,ACCY,ACCZ,PITCH,ROLL,YAW\n");

for (i = 0; i < DATA\_SIZE; i++)

{

fprintf(fpt2, "%f,", Table[TIME][i]);

for (j = ACCX; j <= YAW; j++)

{

var = variance(Table[j], i);

fprintf(fpt2, "%f,", var);

}

fprintf(fpt2, "\n");

}

fclose(fpt2);

Output = fopen("Results.txt", "w");

fprintf(Output, "RESULTS:\n");

for (i = 0; i < DATA\_SIZE; i++)

{

for (j = ACCX; j <= YAW; j++)

{

var\_arr[j] = variance(Table[j], i);

}

moving = isMoving(var\_arr);

if (moving && start\_time == 0.0)

{

start\_time = Table[TIME][i];

startIndex = i;

}

else if (!moving && start\_time != 0.0 && end\_time == 0.0)

{

moving = false;

end\_time = Table[TIME][i];

endIndex = i;

}

if (start\_time != 0.0 && end\_time != 0.0)

{

dist\_arr[ACCX] = integrate\_acc(Table[ACCX], startIndex, endIndex);

dist\_arr[ACCY] = integrate\_acc(Table[ACCY], startIndex, endIndex);

dist\_arr[ACCZ] = integrate\_acc(Table[ACCZ], startIndex, endIndex);

dist\_arr[PITCH] = integrate(Table[PITCH], startIndex, endIndex);

dist\_arr[ROLL] = integrate(Table[ROLL], startIndex, endIndex);

dist\_arr[YAW] = integrate(Table[YAW], startIndex, endIndex);

PrintData(Output, dist\_arr, start\_time, end\_time, startIndex, endIndex);

start\_time = 0.0;

end\_time = 0.0;

}

}

fclose(Output);

return(0);

}

void PrintData(FILE \*fp, float distance[7], float start\_time, float end\_time, int start\_index, int end\_index)

{

fprintf(fp, "########################################################\n");

fprintf(fp, "X Movement: %f [m]\nY Movement: %f [m]\nZ Movement: %f [m]\n", distance[ACCX], distance[ACCY], distance[ACCZ]);

fprintf(fp, "Pitch Movement: %f [rad]\nRoll Movement: %f [rad]\nYaw Movement: %f [rad]\n", distance[PITCH], distance[ROLL], distance[YAW]);

fprintf(fp, "Start Time: %0.2f\t\tEnd Time: %0.2f\n", start\_time, end\_time);

fprintf(fp, "Start Index: %d\t\tEnd Index: %d\n", start\_index, end\_index);

fprintf(fp, "\n\n");

return;

}

float integrate\_acc(float \*input, int start, int end)

{

int i;

float prev\_velocity = 0.0;

float velocity = 0.0;

float distance = 0.0;

for (i = start; i <= end; i++)

{

prev\_velocity = velocity;

velocity += input[i] \* GRAVITY \* SAMPLE\_TIME;

distance += ((velocity + prev\_velocity) / 2) \* SAMPLE\_TIME;

}

return (distance);

}

float integrate(float \*input, int start, int end)

{

float ret\_val = 0.0;

int i;

for (i = start; i <= end; i++)

{

ret\_val += input[i] \* SAMPLE\_TIME;

}

return (ret\_val);

}

float variance (float \*input, int curr\_index)

{

int i;

int local\_var\_window;

float mean = 0;

float var;

if (curr\_index + VAR\_WINDOW <= DATA\_SIZE)

{

local\_var\_window = curr\_index + VAR\_WINDOW;

}

else

{

local\_var\_window = DATA\_SIZE;

}

for (i = curr\_index; i < local\_var\_window; i++)

{

mean += input[i];

}

mean = mean / (VAR\_WINDOW + 1);

for (i = curr\_index; i < local\_var\_window; i++)

{

var += SQR(input[i] - mean);

}

var = var / (VAR\_WINDOW + 1);

return var;

}

bool isMoving(float var\_arr[7])

{

bool ret\_val = false;

if (var\_arr[ACCX] > ACC\_THRESH || var\_arr[ACCY] > ACC\_THRESH || var\_arr[ACCZ] > ACC\_THRESH)

{

ret\_val = true;

}

if (var\_arr[PITCH] > PITCH\_THRESH || var\_arr[ROLL] > ROLL\_THRESH || var\_arr[YAW] > YAW\_THRESH)

{

ret\_val = true;

}

return (ret\_val);

}

*Results.txt*

RESULTS:

########################################################

X Movement: 0.324663 [m]

Y Movement: -0.280333 [m]

Z Movement: 0.216977 [m]

Pitch Movement: -0.016488 [rad]

Roll Movement: -0.002989 [rad]

Yaw Movement: -0.025635 [rad]

Start Time: 0.75 End Time: 2.65

Start Index: 14 End Index: 52

########################################################

X Movement: 0.077216 [m]

Y Movement: 0.556208 [m]

Z Movement: 0.171221 [m]

Pitch Movement: 0.001593 [rad]

Roll Movement: -0.022141 [rad]

Yaw Movement: 0.022497 [rad]

Start Time: 5.45 End Time: 7.15

Start Index: 108 End Index: 142

########################################################

X Movement: -0.208693 [m]

Y Movement: 0.086416 [m]

Z Movement: 0.172758 [m]

Pitch Movement: -0.004715 [rad]

Roll Movement: -0.009771 [rad]

Yaw Movement: -0.026665 [rad]

Start Time: 10.25 End Time: 11.90

Start Index: 204 End Index: 237

########################################################

X Movement: 0.650188 [m]

Y Movement: -0.024251 [m]

Z Movement: 0.311491 [m]

Pitch Movement: -0.003550 [rad]

Roll Movement: 0.002754 [rad]

Yaw Movement: 0.042411 [rad]

Start Time: 14.25 End Time: 16.45

Start Index: 284 End Index: 328

########################################################

X Movement: 16.408960 [m]

Y Movement: -12.966489 [m]

Z Movement: 3.811162 [m]

Pitch Movement: -0.013730 [rad]

Roll Movement: -0.033550 [rad]

Yaw Movement: -0.022816 [rad]

Start Time: 18.80 End Time: 24.55

Start Index: 375 End Index: 490

########################################################

X Movement: -0.221626 [m]

Y Movement: -0.147999 [m]

Z Movement: 0.409759 [m]

Pitch Movement: 0.046144 [rad]

Roll Movement: 0.002978 [rad]

Yaw Movement: 1.566631 [rad]

Start Time: 31.00 End Time: 33.70

Start Index: 619 End Index: 673

########################################################

X Movement: -0.134216 [m]

Y Movement: 0.136253 [m]

Z Movement: 0.102390 [m]

Pitch Movement: -0.001799 [rad]

Roll Movement: -0.011219 [rad]

Yaw Movement: -1.540733 [rad]

Start Time: 37.05 End Time: 38.85

Start Index: 740 End Index: 776

########################################################

X Movement: -10.504405 [m]

Y Movement: -186.317215 [m]

Z Movement: 174.996948 [m]

Pitch Movement: 0.009000 [rad]

Roll Movement: -0.035547 [rad]

Yaw Movement: -0.023689 [rad]

Start Time: 42.60 End Time: 49.75

Start Index: 851 End Index: 994

########################################################

X Movement: 179.587387 [m]

Y Movement: -0.274640 [m]

Z Movement: 182.312164 [m]

Pitch Movement: -0.037802 [rad]

Roll Movement: 0.067978 [rad]

Yaw Movement: -0.023690 [rad]

Start Time: 52.80 End Time: 59.75

Start Index: 1055 End Index: 1194