Shashi Shivaraju
XID: C88650674

REPORT
ECE 6310 Lab #7 - Motion Tracking

Objective:

To calculate motion using data of accelerometers and gyroscopes.

Sensor Data:

The accelerometers and gyroscopes data was recorded using an iPhone when it was moved individually along or about each axis independently for a period of 2-3 seconds and between each motion, the iPhone was held at rest for 2-3 seconds.

Implementation:

Implementation of motion tracking follows the steps listed below:

a) Read the accelerometer and gyroscope data from the file:

The provided file of data recorded using an iPhone contains 7 columns in the file, with the following data:
time x_acc y_acc z_acc pitch roll yaw

The units for time are seconds. Data was sampled at 20 Hz. The unit for the accelerometer data is gravities (G) and the unit for the gyroscope data is radians per second. The data is read from the file stored in a doubly linked list for further processing.

[Code snippet]:

```
while(0<fscanf(fp,"%f %f %f %f %f %f %f",\
      &time,&x_acc,&y_acc,&z_acc,&pitch,&roll,&yaw))/*will exit at EOF or fscanf error*/
{
      /*update the linked list with the new data from file*/
      new_data = (sensor_data*)malloc(sizeof(sensor_data));
      if(!new_data)
      {
            printf("memory allocation failed");
            return NULL;
      }
      memset(new_data,0,sizeof(sensor_data));

      new_data->time = time;
      new_data->x_acc = x_acc; new_data->y_acc = y_acc; new_data->z_acc = z_acc;
      new_data->pitch = pitch;new_data->roll  = roll;new_data->yaw   = yaw;
      if(!list_head)
      {
            list_head = new_data;
            list_head->prev = NULL;
      }
      if(prev_data)
      {
            prev_data->next = new_data;
            new_data->prev = prev_data;
      }

      new_data->next = NULL;
      prev_data = new_data;
}
```

Shashi Shivaraju
XID: C88650674

b) Smooth the sensor data:

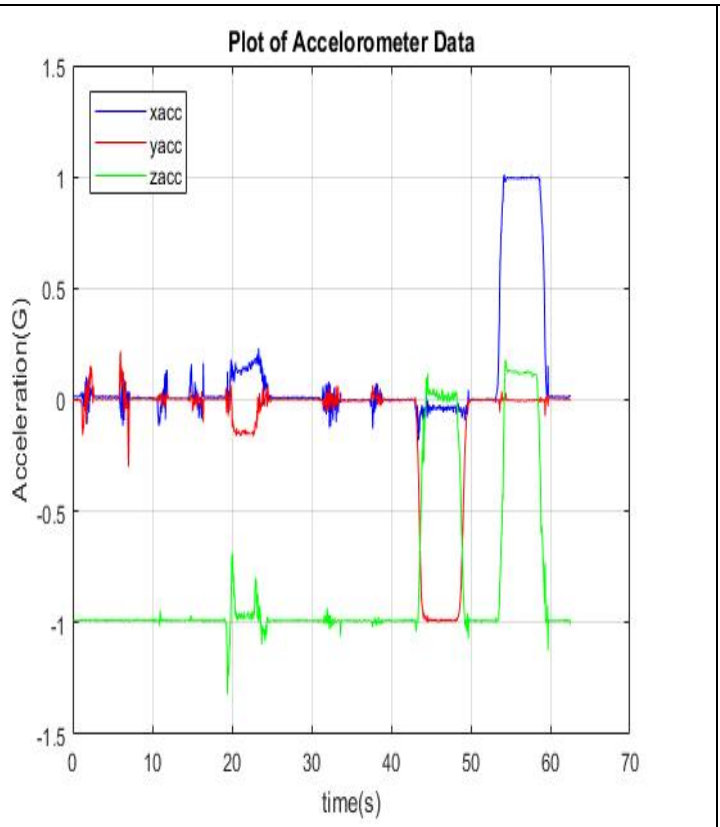
The accelerometer and gyroscope data is smoothened using a moving average filter with the span of 5 elements.

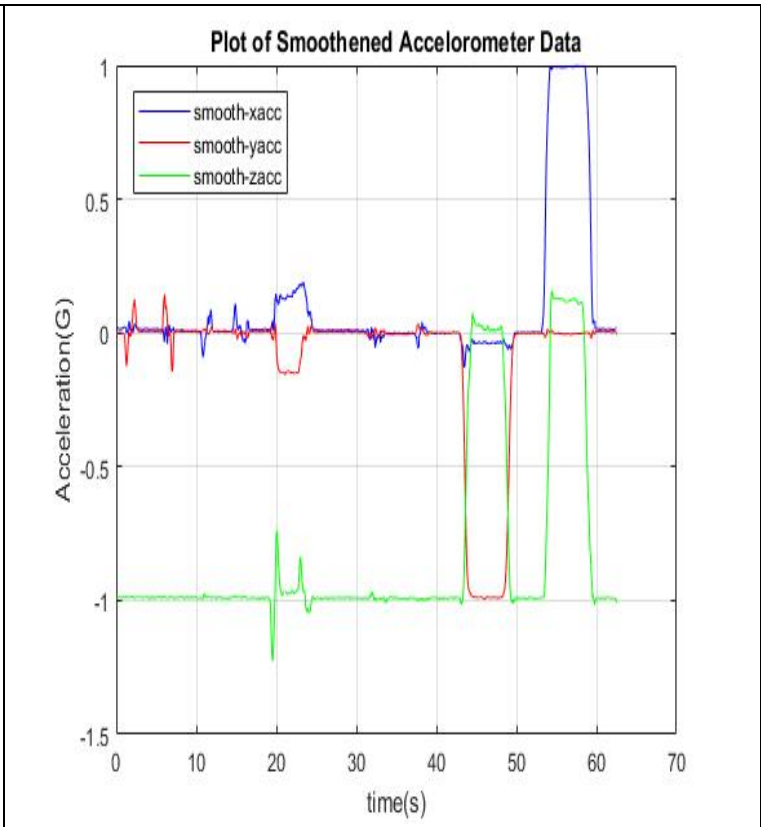
[Code snippet]:

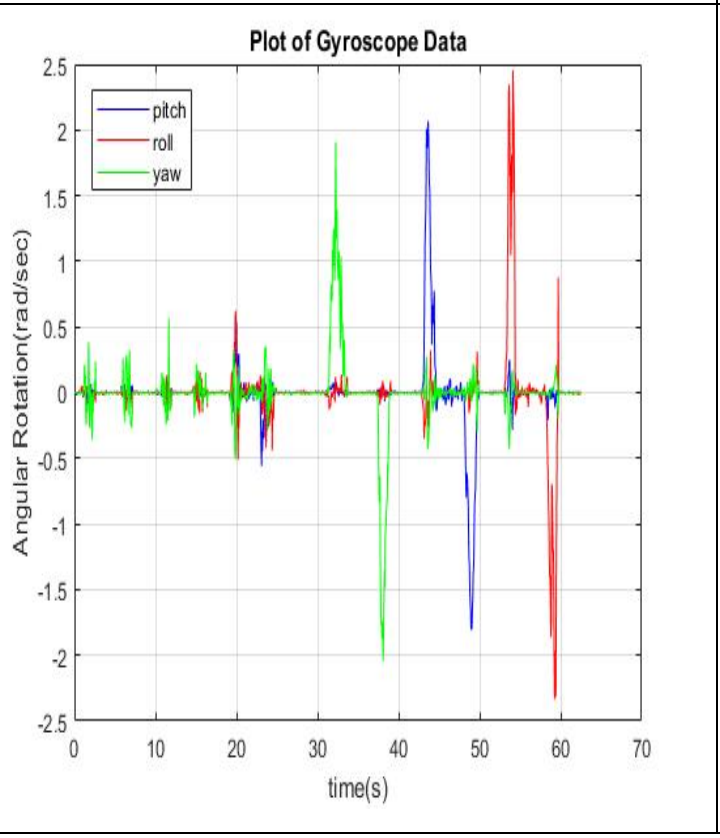
```
void smooth_sensordata(sensor_data* list_head)
{
	curr = list_head;
	while(curr)
	{
		temp2 = curr->prev;
		temp3 = curr->next;
		count ++;
		if(curr == list_head || curr->next == NULL)
		{
			curr->smoothx_acc = curr->x_acc;
			curr->smoothy_acc = curr->y_acc;
			curr->smoothz_acc = curr->z_acc;
			curr->smooth_pitch = curr->pitch;
			curr->smooth_roll  = curr->roll;
			curr->smooth_yaw   = curr->yaw;

		}
		else if(temp2->prev == NULL ||temp3->next == NULL)
		{
			curr->smoothx_acc = (temp2->x_acc + curr->x_acc + temp3->x_acc)/3;
			curr->smoothy_acc = (temp2->y_acc + curr->y_acc + temp3->y_acc)/3;
			curr->smoothz_acc = (temp2->z_acc + curr->z_acc + temp3->z_acc)/3;
			curr->smooth_pitch = (temp2->pitch + curr->pitch + temp3->pitch)/3;
			curr->smooth_roll  = (temp2->roll + curr->roll + temp3->roll)/3;
			curr->smooth_yaw   = (temp2->yaw + curr->yaw + temp3->yaw)/3;
		}
		else
		{
			temp2 = curr->prev;temp1 = temp2->prev;
			temp3 = curr->next;temp4 = temp3->next;
			curr->smoothx_acc = (temp1->x_acc + temp2->x_acc + curr->x_acc +
					     temp3->x_acc   + temp4->x_acc)/5;
			curr->smoothy_acc = (temp1->y_acc + temp2->y_acc + curr->y_acc +
					     temp3->y_acc + temp4->y_acc)/5;
			curr->smoothz_acc = (temp1->z_acc + temp2->z_acc + curr->z_acc +
					     temp3->z_acc + temp4->z_acc)/5;
			curr->smooth_pitch = (temp1->pitch + temp2->pitch + curr->pitch +
					     temp3->pitch + temp4->pitch)/5;
			curr->smooth_roll  = (temp1->roll + temp2->roll + curr->roll +
					     temp3->roll +temp4->roll)/5;
			curr->smooth_yaw   = (temp1->yaw + temp2->yaw + curr->yaw +
					       temp3->yaw + temp4->yaw)/5;
		}

		curr = curr->next;
	}
}
```

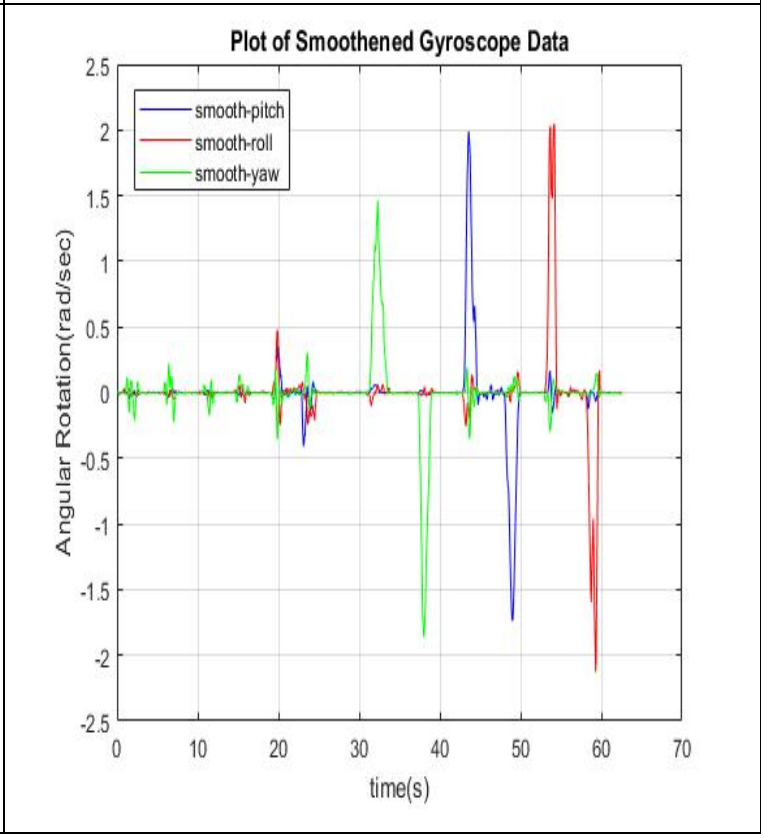
Shashi Shivaraju
XID: C88650674

[Plots of Sensor Data]:









Shashi Shivaraju
XID: C88650674

c) Data Segmentation and Variance Calculation:

A window size of 2.5 second is considered and variance is calculated along each of the 6 axis for these windows.

[Code snippet]:

```
#define DATA_WINDOW_SIZE 50/*50x0.05 = 2.5s*/
void calculate_variance(sensor_data* list_head,int win_size)
{
	head = list_head;
	cur = head;

	while(head)
	{
		temp = head;
		cur = head;
		/*calculate the mean for the window*/
		for(i=0;i<win_size && cur;i++)
		{
			x_acc_sum = x_acc_sum + cur->smoothx_acc;
			y_acc_sum = y_acc_sum + cur->smoothy_acc;
			z_acc_sum = z_acc_sum + cur->smoothz_acc;
			pitch_sum = pitch_sum + cur->smooth_pitch;
			roll_sum  = roll_sum  + cur->smooth_roll;
			yaw_sum   = yaw_sum   + cur->smooth_yaw;
			cur = cur->next;
		}
		x_acc_avg = x_acc_sum/win_size;y_acc_avg = y_acc_sum/win_size;
		z_acc_avg = z_acc_sum/win_size;pitch_avg = pitch_sum/win_size;
		roll_avg = roll_sum/win_size;yaw_avg = yaw_sum/win_size;

		cur = temp;
		/*calculate the variance for the window*/
		for(i=0;i<win_size && cur;i++)
		{
			x_acc_var = x_acc_var+SQR(cur->smoothx_acc-x_acc_avg);
			y_acc_var = y_acc_var+SQR(cur->smoothy_acc-y_acc_avg);
			z_acc_var = z_acc_var+SQR(cur->smoothz_acc-z_acc_avg);
			pitch_var = pitch_var+SQR(cur->smooth_pitch-pitch_avg);
			roll_var  = roll_var+SQR(cur->smooth_roll-roll_avg);
			yaw_var   = yaw_var+SQR(cur->smooth_yaw-yaw_avg);
			cur = cur->next;
		}
		/*Store the variance calculation of
		window data at first data in window*/
		head->x_acc_var = x_acc_var/(win_size-1);head->y_acc_var = y_acc_var/(win_size-1);
		head->z_acc_var = z_acc_var/(win_size-1);head->pitch_var = pitch_var/(win_size-1);
		head->roll_var = roll_var/(win_size-1);head->yaw_var = yaw_var/(win_size-1);

		x_acc_sum=0;y_acc_sum=0;z_acc_sum=0;
		pitch_sum=0;roll_sum=0;yaw_sum=0;
		x_acc_avg=0;y_acc_avg=0;z_acc_avg=0;
		pitch_avg=0;roll_avg=0;yaw_avg=0;
		x_acc_var=0;y_acc_var=0;z_acc_var=0;
		pitch_var=0;roll_var=0;yaw_var=0;
		head = cur;
	}
}
```

Shashi Shivaraju
XID: C88650674

d) Motion Detection and Estimation:

The values of the variance for each axis are checked against their respective thresholds `ACCEL_THRESHOLD` and `GYRO_THRESHOLD`. When any of the 6 variances is greater than the threshold, the iPhone is detected to be in motion else it is at rest. To calculate the motion, the data must be integrated for gyroscopes and doubly integrated for accelerometers. The value of gravities or G-forces (G) is assumed to be 9.8m/s^2 for calculating linear distance in meters.

[Code snippet]:

```
#define SAMPLE_TIME            0.05
#define G                      9.8
#define ACCEL_THRESHOLD  0.003
#define GYRO_THRESHOLD   0.009
#define SQR(x)     (x)*(x)
#define CONV_DEGREE(x) (x*180)/3.14159265358979323846

void motion_estimation(sensor_data* list_head,int win_size)
{
      fp = fopen("motion_tracking_result.txt","w+");
      head = list_head;
      cur = head;
      while(head)
      {
            temp = head;cur = head;
            if(cur->x_acc_var > ACCEL_THRESHOLD ||cur->y_acc_var > ACCEL_THRESHOLD ||
               cur->z_acc_var > ACCEL_THRESHOLD ||cur->pitch_var > GYRO_THRESHOLD  ||
               cur->roll_var  > GYRO_THRESHOLD  ||cur->yaw_var   > GYRO_THRESHOLD)
            {
                  motion_detection_flag = 1;
            }
            if(motion_detection_flag)
            {
                  /*calculate total linear distance and total angular rotation in the window*/
                  for(i=0;i<win_size && cur;i++)
                  {
                        total_pitch_rotation = total_pitch_rotation +
                                               cur->smooth_pitch * SAMPLE_TIME;
                        total_roll_rotation = total_roll_rotation +
                                               cur->smooth_roll * SAMPLE_TIME;
                        total_yaw_rotation = total_yaw_rotation +
                                               cur->smooth_yaw * SAMPLE_TIME;
                        prev_x_vel = x_velocity;
                        x_velocity = x_velocity+cur->smoothx_acc * SAMPLE_TIME;
                        avg_x_vel  = (x_velocity + prev_x_vel)/2;
                        total_x_dist = total_x_dist + avg_x_vel*SAMPLE_TIME;
                        prev_y_vel = y_velocity;
                        y_velocity = y_velocity+cur->smoothy_acc * SAMPLE_TIME;
                        avg_y_vel  = (y_velocity + prev_y_vel)/2;
                        total_y_dist = total_y_dist + avg_y_vel*SAMPLE_TIME;
                        prev_z_vel = z_velocity;
                        z_velocity = z_velocity+cur->smoothz_acc * SAMPLE_TIME;
                        avg_z_vel  = (z_velocity + prev_z_vel)/2;
                        total_z_dist = total_z_dist + avg_z_vel*SAMPLE_TIME;

                        cur = cur->next;
                  }
```

Shashi Shivaraju
XID: C88650674

```
                motion_detection_flag = 0;
                fprintf(fp,"MOTION Detected from time %f to time %f with:\n",
                      head->time,head->time+win_size*SAMPLE_TIME);
                fprintf(fp,"Total Angular Rotation along Pitch Axis %f degree :\n",
                            CONV_DEGREE(total_pitch_rotation));
                fprintf(fp,"Total Angular Rotation along Roll Axis %f degree:\n",
                            CONV_DEGREE(total_roll_rotation));
                fprintf(fp,"Total Angular Rotation along Yaw Axis %f degree:\n",
                            CONV_DEGREE(total_yaw_rotation));
                fprintf(fp,"Total Linear Distance along X Axis %f meter :\n",total_x_dist*G);
                fprintf(fp,"Total Linear Distance along Y Axis %f meter:\n",total_y_dist*G);
                fprintf(fp,"Total Linear Distance along Z Axis %f meter:\n\n",
                            total_z_dist*G);

                total_pitch_rotation=0;total_roll_rotation=0;
                total_yaw_rotation=0;
                x_velocity=0;y_velocity=0;z_velocity=0;
                prev_x_vel=0;prev_y_vel=0;prev_z_vel=0;
                avg_x_vel=0;avg_y_vel=0;avg_z_vel=0;
          }
          else
          {
                /*traverse to the next window*/
                for(i=0;i<win_size && cur;i++)
                {
                      cur = cur->next;
                }
                fprintf(fp,"REST Detected from time %f to time %f\n\n",
                      head->time,head->time+win_size*SAMPLE_TIME);
          }

          head = cur;
    }
    fclose(fp);
    fp = NULL;
}
```

Shashi Shivaraju
XID: C88650674

[RESULTS]:

Time Interval [s]	State	Linear Distance for [X axis] in [meter]	Linear Distance for [Y axis] in [meter]	Linear Distance for [Z axis] in [meter]	Angular Rotation for [Pitch axis] in [degree]	Angular Rotation for [Roll axis] in [degree]	Angular Rotation for [Yaw axis] in [degree]
0.05 - 2.55	**MOTION**	0.509861	-0.29458	-30.2821	-1.2467	-0.13861	-2.2153
2.55 - 5.05	REST	0	0	0	0	0	0
5.05 - 7.55	**MOTION**	0.763422	0.381372	-60.5499	0.100927	-1.39107	1.21214
7.55 - 10.05	REST	0	0	0	0	0	0
10.05 - 12.55	REST	0	0	0	0	0	0
12.55 - 15.05	REST	0	0	0	0	0	0
15.05 - 17.55	REST	0	0	0	0	0	0
17.55 - 20.05	**MOTION**	1.207716	0.485227	-91.2641	5.505589	8.814102	-3.81967
20.05 - 22.55	**MOTION**	5.175993	-3.76947	-119.644	3.65087	-1.2769	-0.9615
22.55 - 25.05	**MOTION**	9.614574	-5.66053	-148.981	-9.82438	-9.69651	3.477279
25.05 - 27.55	REST	0	0	0	0	0	0
27.55 - 62.55	REST	0	0	0	0	0	0
30.05 - 62.55	**MOTION**	9.78368	-5.67189	-179.295	2.810356	-1.42605	**61.29553**
32.55 - 62.55	**MOTION**	9.496688	-5.7445	-209.732	-0.22706	1.323984	**28.28197**
35.05 - 62.55	REST	0	0	0	0	0	0
37.55 - 62.55	**MOTION**	9.421457	-5.45462	-240.2	-0.29282	-0.02978	**-86.4859**
40.05 - 62.55	REST	0	0	0	0	0	0
42.55 - 62.55	**MOTION**	8.193569	-16.5215	-262.663	**91.38357**	-5.09009	-4.01934
45.05 - 62.55	REST	0	0	0	0	0	0
47.55 - 62.55	**MOTION**	7.007069	-41.9405	-270.388	**-90.2301**	3.195326	3.132099
50.05 - 62.55	REST	0	0	0	0	0	0
52.55 - 62.55	**MOTION**	16.10992	-41.9538	-294.202	0.105348	**95.98617**	-4.08637
55.05 - 62.55	REST	0	0	0	0	0	0
57.55 - 62.55	**MOTION**	42.86979	-42.0025	-298.812	-2.22408	**-92.5261**	2.586902
60.05 - 62.55	REST	0	0	0	0	0	0

[Inference]:From the above result it is clear that estimation of orientation using Gyroscope is more accurate than estimation of linear distance using Accelormeter.

<END>