스마트폰 가속도 센서를 이용한 제스처 문자인식을 위한 센서 값 보정

강보경*, 배석찬*
*군산대학교 컴퓨터정보공학과
le2090@hanmail.net*, scbae@kunsan.ac.kr*

Calibration of Accelerometer for Character Recognition through Moving Smartphone

Bo-Gyung Kang*, Seok Chan Bae*
*Dept of Computer Engineering, Kunsan University

요 약

최근 출시된 IPhone이나 Android Phone들을 보면 인터페이스의 편리성과 엔터테이먼트적인 요소의 극대화를 위한 가속도센서, 디지털 나침반, 근접센서, 조도센서 등, 다양한 센서들을 디바이스에 포함하고 있다. 이들 중 가속도 센서를 이용한 다양한 인터페이스가 여러 스마트폰 게임과 어플리케이션들에서 사용되고 있는데 본 논문에서는 가속도 센서의 각축의 값들을 이용해 문자나 특정 입력 값을 기기에 전달할 수 있는 제스처 인식을 위한 센서 값들의 효율적인 사전 보정 알고리즘에 대해서 제안하고자 한다.

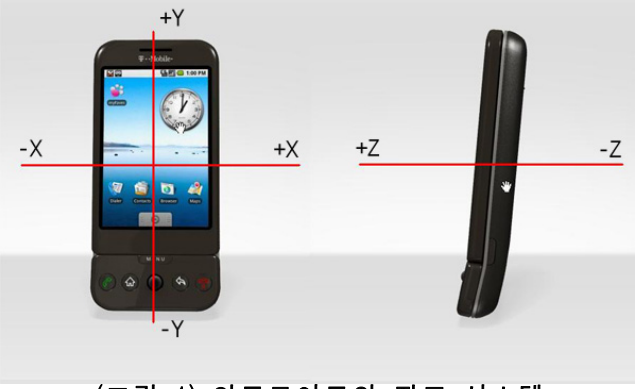
1. 서론

몇 년 전부터 아이폰OS, 윈도우모바일, 안드로이드, 심비안, 블랙베리 등의 다양한 스마트폰 OS들이 각각의 다양한 디바이스들에 사용되면서 스마트폰은 이전에 시장을 지배하던 단순한 휴대전화의 위치에서 탈피하여 엔터테이먼트(고성능 3D게임, 멀티미디어), 소셜네트워크, 일정관리, Mobile Office 등의 강력한 기능을 제공함으로써 휴대용 컴퓨터로써의 입지를 구축해 나가고 있다. 특히 아이폰과 안드로이드 디바이스들, 그리고 윈도우 모바일폰들은 강력한 성능의 하드웨어와 디지털 나침반, 근접센서, 조도센서 등과 같은 다양한 센서들을 포함함으로써 사용자가 흥미를 느낄 수 있는 다양한 인터페이스구현을 가능하게 한다. 최근 출시되는 거의 대부분의 스마트폰들에는 가속도 센서가 내장되어 있는데 본 논문에서는 이 가속도센서를 이용하여 문자를 그리거나 특정 제스처를 취했을 때 기기에 원하는 데이터를 전달하는 것을 가능하게 하기 위한 사전 작업으로써 가속도 센서의 각축의 센서 값들에서 기기의 기울기 값을 효과적으로 제거하기위한 알고리즘을 제안하고자 한다. 이를 통해 기기의 기울기와 이동을 구별하기가 어려운 가속도센서에서 효과적으로 기울기를 제거하고 이동 값만을 추출하는 것을 가능하게 하여 기기를 이용해 공중에 문자를 그리면 해당문자가 디바이스에 입력되게 하거나 TV나 PC에 블루투스나 적외선 통신을 이용하여 문자를 입력할 수 있는 진화된 개념의 리모콘으로써 사용이 가능하게 될 것이다. 흔히 가속도 센서를 통해 개발 시에 물체의 이동방향과 거리를 가속도 센서만으로 측정하는 경우는 거의 없으며 가속도 센서에서 기울기 값을 제거하거나 혹은 그 반대의 목적으로 자이로 센서와 같은 다른 센서를 사용하여 값을 보정하는 것이 일반적이다. 하지만 아이폰3GS를 포함하여 현재까지 출시된 안드로이드 폰들 중 자이로 센서를 포함한 단말기가 없다는 점을 감안하여(아이폰4는 제외) 자이로 센서를 이용한 어플리케이션을 개발 했을 시에 해당 어플리케이션을 사용가능한 유저층은 극히 제한되게 된다. 이는 가속도 센서에 비해 자이로 센서의 가격이 2~3배 이상이라는 사실 때문이기도 할 것이다. 이러한 사실에 기반하여 가속도 센서를 이용해 스마트폰의 기울기만을 알아내는 것이 아닌 스마트폰의 이동거리와 이동방향을 적절한 값으로 효율적으로 얻어낼 수 있다면 많은 스마트폰 어플리케이션에서 유용한 입력방식이 될 수 있을 거라고 생각한다.

2. 안드로이드 스마트 폰의 가속도 센서 값 이용

가속도 센서는 크게 기계식과 반도체식을 이용한 두 제품이 있으며 반도체식이 주로 쓰이는데, 이는 소형이면서 정밀한 검출이 가능하여 전화에 적합한 형태이기 때문이다. 기계식의 경우는 용수철(혹은 이와 비슷한 성질의 것)에 매달린 추와 댐퍼로 구성되어 있다. 특정 수학식을 이용하면 추가 이동한 거리를 통해 가속도를 구해낼 수 있다. 그러나 기계식 센서는 제한된 범위의 가속도만 측정할 수 있는데다 소형화가 어려워 휴대 기기에는 쓰이지 않고 있다. 본 논문의 실험에 쓰인 스마트폰은 삼성전자의 SHW-M110S(갤럭시S)이며 사용OS는 Android 2.1(Eclair)

이고 Bosch Sensortec사의 smb380 3축 가속도 센서가 내장되어 있다. 안드로이드 플랫폼에서의 가속도 센서 좌표 시스템은 (그림 1)과 같다.



(그림 1) 안드로이드의 좌표 시스템

센서 값은 SensorManager에 등록한 SensorEventListener의 콜백메서드에서 접근가능하며 X, Y, Z 의 좌표에 대해서 각각에 해당하는 값을 배열로 받아와서 처리할 수 있게 되어있다. 각각의 좌표축에 대한 정의는 아래와 같다.

-**X축** : 화면에 수평축(portrait 모드에서 짧은 에지를 landscape 모드에서 긴 에지)을 나타내며 오른쪽을 가리킨다.

-**Y축** : 화면에 수직축을 나타내며 화면 위쪽을 가리킨다.

-**Z축** : 단말이 화면을 위로 해서 테이블에 올려져 있다고 생각했을 때 하늘을 가리킨다.

콜백메서드의 매개변수로 넘겨져오는 SensorEvent객체의 멤버변수로 선언된 float[]타입 변수 values를 사용해 각 축의 값들에 접근할 수 있으며 각 배열의 값에 대한 설명은 아래와 같다.

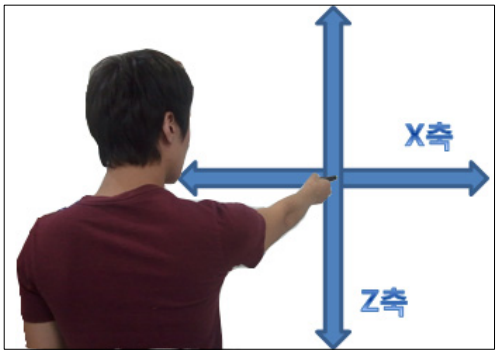
-**values[0]** : X축에 적용되는 힘

-**values[1]** : Y축에 적용되는 힘

-**values[2]** : Z축에 적용되는 힘

각 축의 가속도 단위는 (m/s^2)이며 단말기의 화면이 위를 향하게 하고 평평한 바닥에 올려놨을 때 Z좌표의 값은 약 9.8(m/s^2)의 값을 가지게 되며 이는 지구상에 발생하는 중력가속도 1G에 해당한다.

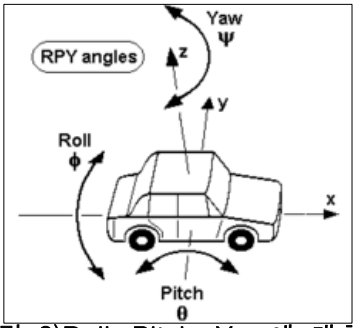
2. X와 Z축의 기울기 값을 상쇄하기 위한 방법



(그림 2)문자인식을 위한 두 축의 사용

스마트폰의 가속도 센서를 이용하여 공중에 문자를 그린다고 할 때 스마트폰의 윗부분(통화 시 귀를 대는 부분)을 사용자의 앞쪽을 향하게 하고 화면을 위쪽으로 향하게 한다. 문자의 입력은 이차원이므로 개발자는 가속도센서 3축에 대해 두 축의 값만이 필요하게 된다. X축과 Y축을 이용해도 되며 X축과 Z축을 이용하여도 무방하다. 하지만 스마트폰을 손에 쥐고 공중에 문자를 그릴 때 (그림 2)와 같은 입력방식이 사용자들에게 좀 더 직관적이며 편리한 입력방식을 제공 할 것이라 생각하여 본 논문에서는 X축과 Z축을 사용하기로 한다. 가속도센서를 이용하여 물체의 이동거리와 이동방향을 구하는데 가장 걸림돌이 되는 것은 가속도 센서가 이동할 때와 기울여 졌을 때 두 경우 모두 센서 값이 변하는 것이다. 그렇기 때문에 제스처 인식을 하기위해 스마트폰을 이동시킬 때나 혹은 공중에 정지 상태로 있을 때, 단말기가 한 축 혹은 두 축으로 기울여진 상태라면 평행으로 이동하거나 평행인 상태로 정지된 것과는 완전히 다른 값이 얻어지게 되어 결과 값에 패턴을 인식 하기 위한 알고리즘을 적용하는 것이 거의 불가능해 진다. 이를 해결하기 위해 두 가지 방법을 적용해 보았는데 첫 번째 방법은 단순히 말해 만약 Z축을 보정하고자 한다면 기기가 한축으로 기울여 졌을 때 기울여진 축의 값을 이용하여 그 값에 따라 Z축값을 상쇄시키는 것이었다. 하지만 이 방법은 정지 시의 기울임 값 제거에는 어느 정도 효과가 있을지 모르지만 각 축의 값은 기울일 때 뿐만이 아니라 이동 시에도 변하게 되니 이동 시 불확실한 값을 얻게 된다는 결론을 실험을 통해 얻게 되었다. 두 번째 방법은 각축의 기울기를 arctan()로 구하여 제스처 인식에 이용하려는 각 축에 값을 상쇄시키는 것이었다. 이 방법은 스마트폰을 손으로 잡고 멈춰있을 때 효과적인 값 보정 결과를 보여 주었으며 이동시에도 보정값으로써 적절히 사용될 수 있는 미미한 변화를 보여주었다.

3. 기울기 값 제거



(그림 3)Roll, Pitch, Yaw에 대한 정의

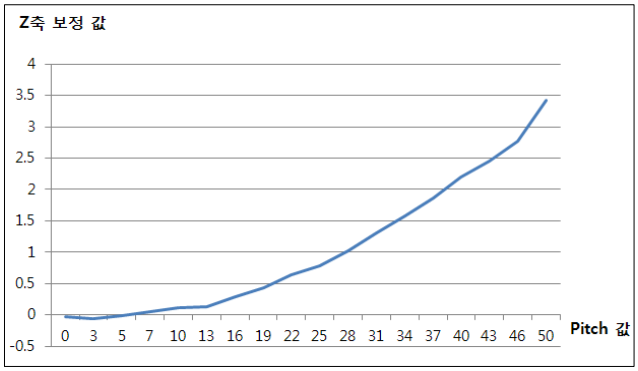
arctan()로 구한 roll값과 pitch값을 각각 X축 중심 기울기와 Y축 중심 기울기 값을 상쇄하기위한 입력 값 x로 정하고 해당 roll값과 pitch값이 추출 되었을 때의 X축과 Z축 값에서 각각 0과 9.8을 뺀 수를 입력 값 y로 정하여 그래프의 곡선함수를 구해야 한다. 여기서 pitch값과 roll

값을 구하기 위한 수식은 아래와 같다.

$$roll = \arctan\left(\frac{y}{\sqrt{x^2 + z^2}}\right)$$

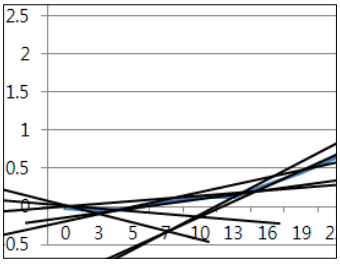
$$pitch = \arctan\left(\frac{x}{\sqrt{y^2 + z^2}}\right)$$

각 축의 Roll과 Pitch에 대한 보정작업은 전부 동일한 방식으로 진행되는데 여기서는 Z축을 Pitch에 대하여 보정하는 과정에 대해 설명하겠다. 스마트폰의 머리부분이 앞을 향하게 하고 화면이 위를 바라볼 때 왼쪽으로 회전시켜 화면이 왼쪽을 향하게 했을 때의 Pitch값의 변화는 이론상으론 0~90이다(하지만 실제로 테스트해보면 0~84정도의 값이 나온다). 본 논문에서는 0~50까지의 Pitch값에 대해서 보정을 하였다. 예를 들어 스마트폰을 Y축을 중심으로 살짝 기울여 Z축을 pitch에 대하여 보정하고자 할 때 pitch값이 3.0이 나오면 이때 Z축의 값은 9.8561이 나오며 스마트폰이 기울임에 영향을 받지 않기 위해서는 9.8-9.8561 의 값(-0.0561)을 다시 Z축 값에 더해주어야 한다. 여기서 이 -0.0561을 pitch값 3.0에 대한 Z축 보정 값이라고 말하겠다. 0~50까지의 pitch값에 대한 Z축 보정 값에 대한 그래프는 아래의 (그림 4)와 같다.



(그림 4) 0~50까지의 pitch값에 대한 Z축 보정 값

그래프에 보여지는 것처럼 곡선그래프가 그려지는데 곡선의 함수를 구하기가 쉽지 않으며 정확한 곡선의 함수를 구해도 고차방정식이 될 가능성이 높기 때문에 스마트폰이라는 임베디드 기기의 성능상의 취약점을 고려하여 pitch값들을 약3정도의 간격으로 끊어 직선의 방정식을 구하였고 각각의 방정식을 if와 else if문으로 분리하여 적용하였다.



(그림 5) 2개의 좌표씩 각각의 직선을 구한다.

x		y		수식_f(x)
0	3	-0.0252	-0.0561	-0.0103*x+-0.0252
3	5	-0.0561	-0.0178	0.01915*x+-0.11355
5	7	-0.0178	0.0558	0.0368*x+-0.2018
7	10	0.0558	0.1192	0.021133*x+-0.092133
⋮				
43	46	2.4574	2.77	0.1042*x+-2.0232
46	50	2.77	3.429	0.16475*x+-4.8085

(그림 6) Pitch값 각 구간별 직선의 함수

계산을 통해 얻어진 Pitch값의 구간별 직선의 방정식은 (그림 6)과 같으며 x값이 Pitch값이며 y값은 위에서 언급한 것처럼 9.8-Z축 값으로 하였다. 만약 pitch값으로 3.0이 들어온다면 -0.0103*x+-0.0252 식의 x에 pitch값을 대입하여 얻어진 보정 값을 다시 Z축 값에 더하면 기울임 값을 효과적으로 제거할 수 있다. 위와 반대로 기기를 오른쪽으로 기울여 화면이 오른쪽을 향하게 하면 Pitch값이 0.0~-90이 되는데 위에서 언급했던 방식으로 pitch값의 구간별 직선의 방정식을 구해 얻어진 값을 Z값에 더해주면 Z축의 Pitch에 대한 보정이 완료된다. 같은 방식으로 Z축 값을 roll값에 대하여 보정하고 X축 또한 Pitch값에 대해 보정한다. X축 가속도 값은 Roll값에 대해 큰 영향을 받지 않기 때문에 Pitch에 대해서만 보정해주어도 결과에 큰 변화는 없다.

4. X축과 Z축의 임계구간 설정

스마트폰을 사람이 한손으로 잡고 공중에 정지해 있을 때에도 미세한 흔들림과 기울임에 대해 가속도 센서 값이 계속해서 변화하므로 X축 값이 x>-0.5 && x <0.5 라면 X축 값에 0을 대입하고 Z축 값이 z> 9 && z < 10.6 이면 0을 대입하였다. 또한 Z축 값이 임계구간안의 값이 아닐 때는 중력가속도 9.8을 뺀 값을 Z값으로 설정해 주었다. 이를 통해 스마트폰을 사용해 제스처를 입력할 때 좀 더 신뢰성 있는 이동방향을 구하는 것을 가능하게 하였다. 하지만 만약 제스처 분류를 하기 위한 패턴인식을 하는 과정에서 이동거리를 사용해야 한다면 위에 설정한 임계구간을 통해 손실되는 값들이 생기게 되므로 패턴인식에 이동거리를 사용하게 될 때는 임계구간에 의해 손실된 가속도 값을 제스처 입력의 끝에서 되살려주는 별도의 알고리즘이 필요하게 될 것이다.

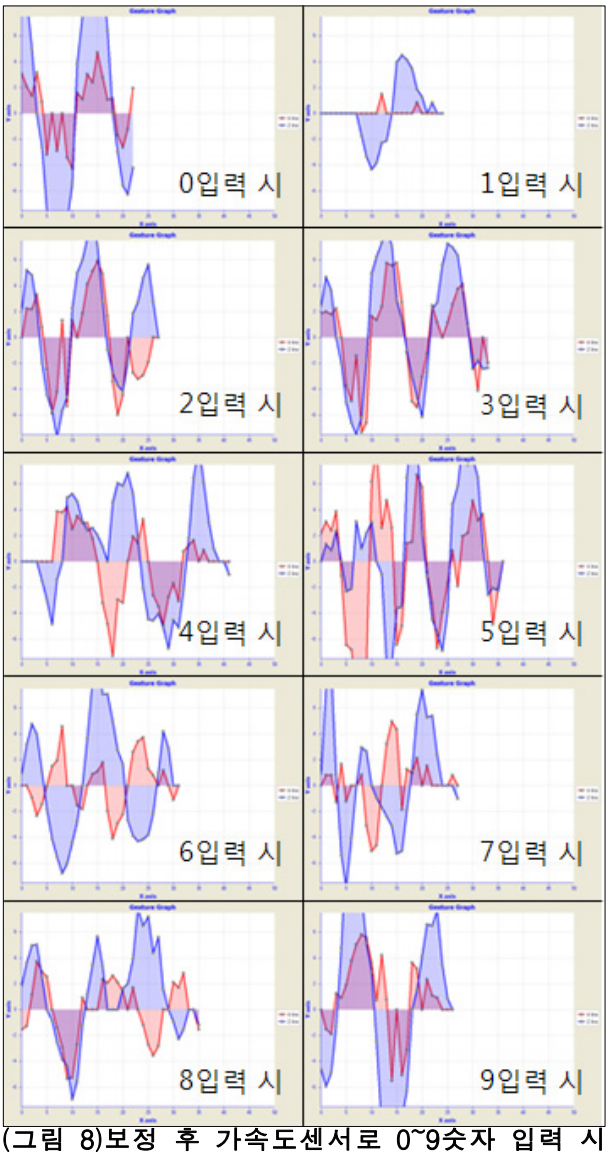
5. 결론

기울임 값을 제거하여 보정한 X축과 Z축 값을 실험을 통해 보정 전의 에러율과 비교해 보았다. 실험은 “이동시”와 “손으로 잡고 있을 때”로 나뉘어 진행하였으며 여기서 에러율이란 오른쪽으로 이동시에는 X축 가속도 값만 변경되고 Z축 값은 0을 벗어나면 안 되는데 Z축 값이 0을 벗어난 횟수를 백분율로 환산한 것을 말하며 반대로 아래로

이동시에는 X축 값이 0을 벗어난 횟수를 백분율로 나타낸 것이다. “손으로 잡고 있을 때”라는 것은 사용자가 스마폰으로 제스처를 입력하기 직전 혹은 직후 공중에 스마트폰을 들고 있는 상태를 말하는 것인데 이때 사람은 조금씩 손목이나 팔을 움직일 수 있고 기기를 완전한 수평으로 들고 있을 수 없기 때문에 손 떨림에 의한 X와 Z축 에러율을 말하는 것이다. 임계구간의 설정은 “보정 값 미 적용 시”와 “보정 값 적용 시“ 두 경우 모두 설정되었다.

	이동할 때		손으로 잡고 있을 때	
	오른쪽 이동시	아래쪽 이동시		
	Z축 에러율	x값 에러율	z축 에러율	x축 에러율
보정 값 적용시	3%	23%	0%	3%
보정 값 미 적용시	54%	83%	43%	80%

(그림 7) 보정 전과 보정 후의 에러율 비교



(그림 8)보정 후 가속도센서로 0~9숫자 입력 시 그래프의 변화

(그림 7)에서 보이듯이 보정전과 비교하여 보정 값을 적용했을 때는 이동 시와 정지 시 모두 확실히 낮아진 에러율을 보여준다. 다만 아래로 이동 시(Z축 이동 시) X축의 가속도 값이 변하게 되는 에러율은 아직도 조금 높은 편인데 임계구간의 확대나 적절한 보조적 보정 알고리즘의 적용으로 10%미만으로 떨어뜨릴 수 있을 거라고 생각하며 보정 전에 발생한 에러 값들에 비해 훨씬 적은 가속도 에러 값이 입력되므로 데이터를 정규화 시키고 패턴을 찾을 때 무시할 만한 수치가 될 수 있다. (그림 8)은 각 축을 보정 후 가속도 센서를 이용하여 숫자 0에서 9까지를 입력했을 때의 그래프 변화를 시각적으로 나타낸 것 이다. 즉, 각 숫자의 대표적인 그래프들을 캡쳐한 것인데, 동일 숫자에 대한 가속도 값 그래프들은 비슷한 모양의 패턴을 보여줌으로 평균값에 근접한 그래프들을 캡쳐하였다. 파란색 선은 가속도 센서의 Z축의 값이며 빨간색 선은 가속도 센서의 X축의 값이다. 캡쳐된 그래프의 X축은 시간의 변화이며 Y축은 가속도 값의 변화이다. 안드로이드폰에서 숫자 제스쳐를 입력하면 프로그램이 가속도 센서의 X축과 Z축 값들을 XML형태로 만들어 3G데이터 통신망을 통해 데스크탑 PC에 Socket으로 전송하며 이 데스크탑PC(서버 PC)에서는 Socket으로 데이터가 들어올 때마다 전송된 XML파일을 파싱하여 그래프로 그리게 하였다. 서버 프로그램은 Java와 SWT로 작성하였다. (그림 8)의 “1입력 시”에서 보면 숫자 1을 입력할 때는 가속도 센서의 Z축의 값만이 검출되어야 하고 X축 값이 되도록 검출되지 않아야 하는데 사용자는 숫자 1을 입력할 때 지면과 수직으로 정확하게 기기를 이동시키면서 값을 입력하지 않을 수 있고 또한 보정 후 가속도 센서의 X축 값 에러율 발생에 의해 X축 값(빨간색 선)이 미약하게나마 검출순자의을 확인할 수 있다. (그림 8)에서 입력된 0~9까지의 숫자들의 그래프들을 보면 가속도 센서의 X축과 Z축의 값들이 겹치는 구간와 겹치지 않는 구간의 개수와 넓이가 저마다 다르다는 것을 알 수 있으며 또한 X축과 Z축의 값이 처음 검출되기 시작할 때를 보면 양수일 때와 음수일 때로 나뉜다는 것을 알 수 있다. 이러한 입력된 데이터들은 정규화를 반드시 거쳐야 하며 차후 고려될 만한 여러 패턴인식 알고리즘을 거쳐 입력된 숫자들을 분류하는데 사용될 수 있으며 숫자뿐만이 아니라 한글과 영어 알파벳 등의 문자입력을 인식할 때에도 사용될 수 있을 거라고 생각한다.

참고문헌

[1] Kimberly Tuck ,Accelerometer Systems and Applications Engineering Tempe, AZ, “Implementing Auto-Zero Calibration Technique for Accelerometers”

[2] Kurt Seifert and Oscar Camacho, "Implementing Positioning Algorithms Using Accelerometers"

[3] Kimberly Tuck ,Accelerometer Systems and Applications Engineering Tempe, AZ, "Tilt Sensing Using Linear Accelerometers"