스마트폰 가속도 센서를 이용한 숫자인식

배석찬, 강보경
군산대학교 컴퓨터정보공학과

요 약

본 연구에서는 가속도 센서의 각 축의 값들을 이용해 숫자나 특정 입력 값을 기기에 전달할 수 있는 제스처 인식을 위한 센서 값들의 효율적인 사전 보정 알고리즘과 분류 알고리즘에 대해서 제안한다. 실험결과 보정 전과 보정 후의 X축과 Z축의 에러율을 통하여 전처리된 데이터가 생성됨을 알 수 있었다. 또한 전처리된 데이터에 적용할 정규화와 분류 알고리즘으로 구현한 인식기가 높은 인식률을 보여주었다.

키워드 : 가속기, 보정 및 분류, 제스처 인식

Number Recognition Using Accelerometer of Smartphone

Seok Chan Bae, Bo-Gyung Kang
Dept. of Computer Information Engineering, Kunsan Nat'l University

ABSTRACT

In this Paper, we suggest the effective pre-correction algorithm on sensor values and the classification algorithm for gesture recognition that use values for each axis of the accelerometer to send data(a number or specific input data) to device. we know that creation of reliable preprocessed data in experimental results through the error rate of X-Axis and Y-Axis for pre-correction and post-correction. we can show high recognition rate through recognizer using the normalization and classification algorithm for the preprocessed data.

Keywords: Accelerometer, pre-correction & classification algorithm, gesture recognition

1. 서 론

1.1 연구 배경 및 필요성

최근 출시되는 대부분의 스마트폰들에는 가속도 센서가 내장되어 있다. 본 연구에서는 이 가속도센서를 이용하여 숫자를 그리거나 특정 제스처를 취했을 때 기기에 원하는 데이터를 전달하는 것을 가능하게 하기 위한 사전 작업이다. 가속도 센서의 각 축의 센서 값들에서 기기의 기울기 값을 효과적으로 제거하기 위한 알고리즘과 숫자와 특정 기호를 전처리하여 스마트폰 가속도 센서를 이용해 인식하게 하는 알고리즘을 제안하고자 한다. 본 연구를 통해 기기의 기울기와 이동을 구별하기가 어려운 가속도센서에서 효과적으로 기울기를 제거하고 이동값만을 추출하는 것을 가능하게 하였다. 스마트폰을 이용해 공중에 숫자를 그리면 해당숫자가 디바이스에 입력되게 하거나 TV나 PC에 블루투스나 적외선 통신을 이용하여 숫자나 문자를 입력할 수 있는 진화된 개념의 리모콘으로써 사용이 가능하게 될

논문투고 : 2011-01-24
논문심사 : 2011-02-25
심사완료 : 2011-02-25

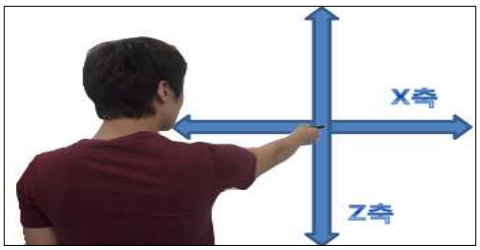
것이다. 가속도 센서를 통해 물체의 이동방향과 거리를 가속도 센서만으로 측정하는 경우는 거의 없으며 가속도 센서에서 기울기 값을 제거하거나 혹은 그 반대의 목적으로 자이로 센서와 같은 다른 센서를 사용하여 값을 보정하는 것이 일반적이다. 하지만 아이폰3GS를 포함하여 현재까지 출시된 안드로이드 폰들 중 자이로 센서를 포함한 단말기가 없다는 점을 감안하여(아이폰4는 제외) 자이로 센서를 이용한 애플리케이션을 개발 했을 시에 해당 애플리케이션을 이용 가능한 사용자층은 극히 제한되게 된다. 이는 가속도 센서에 비해 자이로 센서의 가격이 2~3배 이상이다. 이러한 사실에 기반하여 가속도 센서의 각 축의 값에서 기울기 값을 효과적으로 제거하고 전처리를 거친 데이터를 통해 숫자나 특정 기호의 입력을 인식할 수 있다면 다양한 분야에서 활용될 것이다.

1.2 관련연구

2010년 10월 현재까지 NDSL과 같은 다양한 연구들의 검색이 가능한 국내 사이트에서도 가속도센서를 이용한 문자나 특정기호의 인식에 관한 연구는 찾기가 어렵다. 2축 혹은 3축 가속도 센서의 움직임을 패턴인식을 통해 게임이나 인터페이스에 적용한 사례들이 있는데 대부분이 사용자의 걸음수를 계산하거나 앉기 일어서기 등의 행동패턴을 분류하는 것에 대한 연구들이다. 그 중 인체의 운동량을 가속도 센서로 측정하기 위한 연구로 Extreme Learning Machine(ELM)을 사용하여 서기, 걷기, 뛰기, 앉기, 넘어지기 등의 일반적 행동 및 자세에 대하여 분류를 수행하였으며 ELM은 기존의 은닉층의 파라미터가 트레이닝 데이터와 완벽히 독립적인 단일계층-피드 포워드 네트웍(Single Layer Feedforward Networks, SLFNs)으로 다양한 활동함수(activation function)을 적용할 수 있고 단일-은닉계층을 사용하므로 기존의 신경회로망보다 처리속도가 빠르다는 장점이 있다. 그 연구에서는 2축 가속도 신호를 통해 6개의 일상생활 패턴에 대한 패턴분류를 수행하였고 대략 80%정도의 정확도를 확보하였다[2]. 3축 가속도 센서를 이용하여 보행 시 발생하는 데이터를 획득하고 사람의 걸음 수를 계산해 내는 알고리즘과 걷기, 제자리 걷기, 뛰기, 천천히 걷기 등의 각 상황별 걸음 수를 정밀하게 측정하기 위해 적응적인 임계값을 사용하는 보행 횟수 검출 알고리즘을 제안하였다[3]. 이는 고정 임계값을 이용하는 알고리즘 보다 5~10% 정확도가 높다고 하였다. 또한 가속도 센서의 기울기를 이용하여 진동과 소리 간 전환과 같은 기능을 구현하고자 시도한다[1]. 이처럼 가속도 센서를 이용한 특정 패턴을 인식하고자 하는 연구가 국내에서도 다방면에서 활발히 진행되고 있지만 가속도 센서만으로 특정문자와 기호를 그리고 이를 인식하고자하는 목표를 가지고 있다면 더 신뢰도 높은 가공된 입력데이터를 보장하는 전처리와 복잡한 패턴을 분류해 내기 위한 특별한 알고리즘이 필요하였다. 리모트 컨트롤러를 사용하여 특정 제스처를 인식하는 알고리즘을 기술하고 있다[8]. 그래서 본 연구에서는 스마트폰에 내장된 가속도 센서를 사용하고 제스처의 획수를 1회로 제한하는 등의 차이점이 있기에 사전 보정 알고리즘과 분류 알고리즘에 대해 다른 여러 가지 요소들을 고려하였다.

2. 기울기 값 제거를 위한 전처리 알고리즘

2.1 X와 Z축의 기울기 값을 상쇄하기 위한 방법



(그림 1)숫자인식을 위한 두 축의 사용

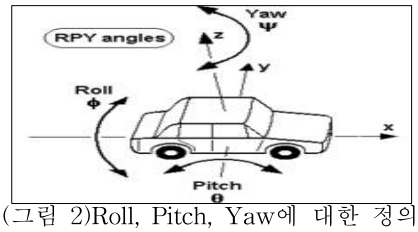
스마트폰의 가속도 센서를 이용하여 공중에 숫자를 그릴 때 스마트폰의 윗부분(통화 시 귀를 대는 부분)을 사용자의 앞쪽을 향하게 하고 화면을 위쪽으로 향하게 한다. 숫자의 입력은 이차원이므로 가속도센서 3축에 대해 두 축의 값만이 필요하게 된다. X축과 Y축을 이용해도 되며 X축과 Z축을 이용하여도 처리 가능하다. 하지만 스마트폰을 손에 쥐

고 공중에 숫자를 그릴 때 그림 1과 같은 입력방식이 사용자들에게 좀 더 직관적이며 표준화된 입력방식을 제공 할 것이라 생각하여 본 연구에서는 X축과 Z축을 사용하였다. 가속도센서를 이용하여 물체의 이동거리와 이동방향을 구하는데 가장 문제가 되는 것은 가속도 센서가 이동할 때와 기울여 졌을 때 두 경우 모두 센서 값이 변하는 것이다. 그러므로 제스처 숫자 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 적용했는데 첫 번째 방법은 Z축을 보정하고자 한다면 기기가 한축으로 기울여 졌을 때 기울여진 축의 값을 이용하여 그 값에 따라 Z축 값을 상쇄시키는 것이었다. 하지만 이 방법은 정지 시의 기울임 값 제거에는 어느 정도 효과가 있으나 각 축의 값은 기울일 때 뿐 만이 아니라 이동 시에도 변하게 되니 이동 시 불확실한 값을 얻게 된다. 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 보정 값으로써 적절히 사용될 수 있는 적은 변화를 보여주었다.

2.2 보정과 분류 알고리즘

2.2.1. 보정 알고리즘

그림 2는 Roll, Pitch, Yaw에 대한 정의를 하였다. arctan()로 구한 roll값과 pitch값을 각각 X축 중심 기울기와 Y축 중심 기울기 값을 상쇄하기위한 입력 값 x로 정하고 해당 roll값과 pitch값이 추출되었을 때의 X축과 Z축 값에서 각각 0과 9.8을 뺀 수를 입력 값 y로 정하여 그래프의 곡선함수를 구해야 한다.

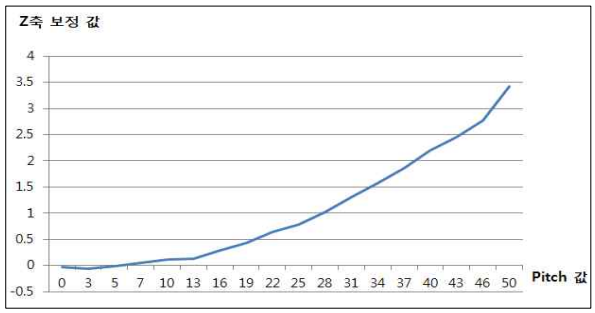


(그림 2)Roll, Pitch, Yaw에 대한 정의

pitch값과 roll값을 구하기 위한 수식은 아래와 같다.

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

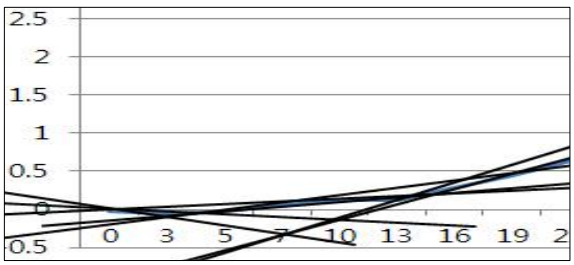
$$pitch = \arctan\left(\frac{x}{\sqrt{y^2 + z^2}}\right)$$



(그림 3) 0~50까지의 pitch값에 대한 Z축 보정 값

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 0~90이다[4,5,6]. 본 연구에서는 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 값이라 한다. 0~50까지의 pitch값에 대한 Z축 보정 값에 대한 그래프는 위의 그림 3과 같

다. 그림 4는 곡선의 함수를 구하기 위해서 고차방정식이 될 수 있으므로 임베디드 기기 성능상의 취약점을 고려하여 pitch값들을 약 3정도의 간격으로 끊어 직선의 방정식을 구하였고 각각의 방정식을 if와 else if문으로 분리하여 적용하였다.



(그림 4) 2개의 좌표씩 각각의 직선

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 5) Pitch값 각 구간별 직선의 함수

그림 5는 계산을 통해 얻어진 Pitch값의 구간별 직선의 함수이다. x값이 Pitch값이며 y값은 위에서 언급한 것처럼 9.8-Z축 값(9.8-9.8252=-0.0252)으로 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 결과에는 큰 변화가 없다.

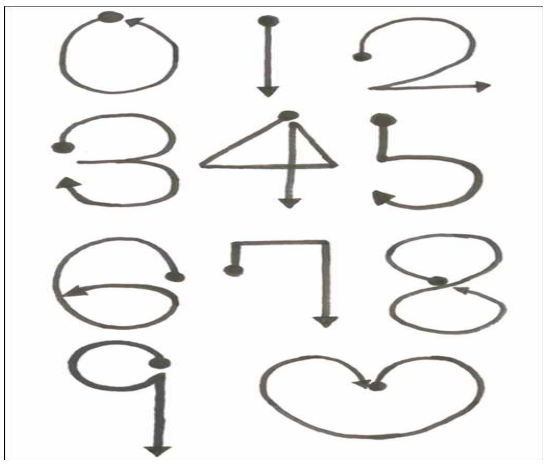
2.2.2 임계구간 설정에 의한 추가 보정

사람이 스마트폰을 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 이용하여 제스처를 입력할 때 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. 하지만 본 연구에서 숫자와 기호를 식별하는 분류기를 구현시에 속도와 이동거리는 사용하지 않는다.

2.2.3. 분류 알고리즘

그림 6은 각 숫자 제스처를 입력할 때의 시작점과 끝점을 나타내며 제스처의 입력은 시작점(동그란점)에서 시작하여 끝점(화살표)에서 끝나게 된다. 제스처 입력에 대한 가이드라인을 정해 놓은 이유는 사람마다 숫자를 입력할 때의 시작점과 끝점이 각자 다르고 같은 숫자를 쓰더라도 완전히 다른 방향과 방식으로 쓰기 때문에 모든 패턴에 대한 처리를 하는 것은 분류기의 구현을 어렵게 할 뿐아니라 오인식률을 굉장히 많이 증가시킬 수 있다. 그렇기 때문에 보편적으로 사람들이 숫자를 쓰는 방향과 시작점을 조사하여 가이드라인을 정하였다. 가이드라인을 정해 놓고 제스처를 분류하고 이를 인식하고 있다[7]. 이는 가속도 센서를 이용하여 문자를 그릴 때 같은 문자에 대해서도 그리는 방향에 따라 값이 다르기 때문이다. 가이드라인을 보면 직관적으로 어떤 방식으로 숫자를 입력해야 하는지 쉽게 알

수 있을 것이다. 분류기를 구현하는데 가장 큰 문제는 가이드라인을 정해 주었다 해도 사람마다 조금씩 입력 값과 그 패턴이 다르다는 것이다. 또한 천천히 이동시키는 사람과 빨리 이동시키는 사람들이 있으므로 분류기 테스팅에 사용될 DB데이터에는 다양한 사람들의 시험데이터가 존재하여야 했다.



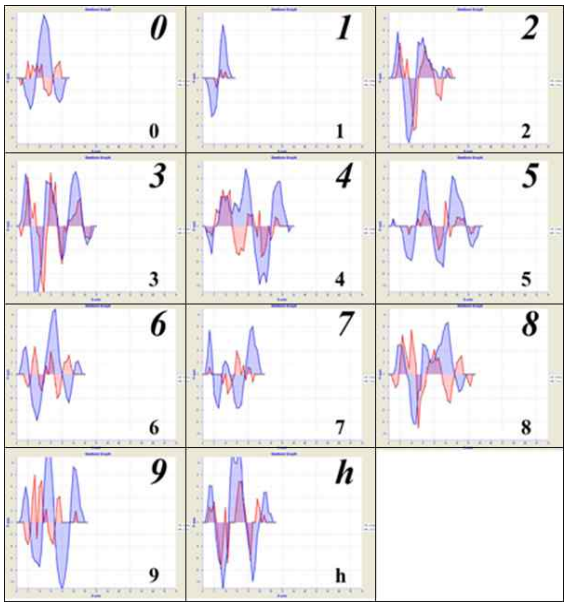
(그림 6)제스처들의 입력 가이드라인

3. 데이터의 정규화

```
for(i = 0 to The size of XZData - 3){
  if(x_i ≠ 0 or z_i ≠ 0){
    if((x_i ≠ 0 and x_i+1 = 0 and x_i+2 = 0) or
       (x_i ≠ 0 and x_i+1 ≠ 0 and x_i+2 = 0 and x_i+3 = 0) and
       (z_i = 0 and z_i+1 = 0 and z_i+2 = 0))
    {
      Garbage Value로 판단하고 삭제한다.
    }
  }
}
```

(그림 7) 입력데이터의 정규화

그림 7은 정규화 과정 중 데이터 입력의 시작부분의 Garbage Values를 제거하기 위한 일부 알고리즘이다. 그림 8은 가속도 센서를 이용하여 숫자 0에서 9까지를 입력했을 때, 기울기 값을 제거하는 전처리를 거친 후 다시 각 축의 값들을 간단히 정



(그림 8)전처리와 정규화 후의 센서 값의 그래프

규화 한 데이터에 대한 그래프 변화를 시각적으로 나타낸 것이다. 그러므로 전처리를 거친 Garbage Values들을 정규화 과정에서 제거하기가 보다 용이해 진다. 정규화 과정을 거친 각 제스처 데이터의 대표적인 그래프들을 캡처한 것이다. 정규화가 필요한 이유는 사용자가 제스처 입력 전 혹은 입력 후 스마트폰을 완벽한 수평으로 유지하기는 어렵고 또 손이 떨릴 수도 있기 때문에 이 입력의 처음과 끝의 Garbage Values 를 제거해 주어야 한다. 이러한 정규화 작업을 거치기 전에 본 연구에서 제안한 전처리 알고리즘을 먼저 거치고 나면 Garbage Values들은 일정 횟수 이하로 처음과 끝에 발생하며 또한 그 값의 크기도 미세하게 된다. 동일 숫자에 대한 가속도 값 그래프들은 비슷한 모양의 패턴을 보여주기 때문에 평균값에 근접한 그래프들을 캡처하였다. 파란색 선은 가속도 센서의 Z축의 값이며 빨간색 선은 가속도 센서의 X축의 값이다. 캡처된 그래프의 X축은 시간의 변화이며 Y축은 가속도 값의 변화이다. 안드로이드폰에서 숫자 제스처를 입력하면 본 저자가 구현한 프로그램이 가속도 센서의 X축과 Z축 값들을 XML형태로 만들어 3G데이터 통신망을 통해 데스크탑 PC에 Socket으로 전

송하며 이 데스크탑 PC(서버PC)에서는 Socket으로 데이터가 들어올 때마다 전송된 XML데이터를 파싱하여 그래프로 그리게 하였다. 또한 구현한 분류기의 테스팅을 위해 XML데이터가 전송되어 올 때마다 서버의 DB에 저장하였다. 최종적으로는 분류기를 스마트폰 애플리케이션 안에 내장하여야 하지만 분류기를 구현하는 과정에서는 효율성을 위해 데이터를 서버 PC에 모으고 분류기를 서버에 구현한 후 실제 스마트폰 프로그램 구현 시에 다시 애플리케이션에 분류기를 내장시켰다. 14명에게서 0에서 9와 하트 제스처(총 11개의 제스처)에 대한 입력을 받아 XML형태로 서버의 DB에 저장을 하였으며 최종적으로 DB에 저장된 XML파일의 수(샘플 데이터의 수)는 887개가 되었다. 서버 프로그램은 Java와 SWT로 작성하였다. 그림 8에서 보면 각각의 숫자 제스처들이 입력되었을 때 입력그래프의 모양이 많이 다르다는 것을 알 수 있다. 분류기를 구현하기 위해서 먼저 가장 큰 분류 기준을 정하였고 그것은 처음 검출되는 Z좌표(파란색선)값이 음수인지 양수인지에 따라 2그룹으로 나뉘게 된다는 것이다. 또한 분류할 수 있는 특징이 많으면 많을수록 최종 인식 확률이 높아지게 되고 비슷한 패턴의 다른 제스처 숫자들의 오인식 확률이 낮아지게 되므로 분류 가능한 특징이 많은 숫자들을 분류 순서의 앞에 두었다. 0과 1은 처음에 출현하는 Z좌표의 값이 음수 덩어리일 가능성이 100%이므로 같은 부류로 묶고 하트와 2, 3, 8 또한 Z값의 덩어리 출현 순서가 양수-음수-양수 로 동일하므로 같은 부류로 묶는다. 그래프의 패턴이 비슷한 6, 7, 9 또한 같은 부류로 묶고 다른 패턴들과 다소 상이한 패턴을 보여주는 4, 5를 한 부류로 묶는다. 같은 부류의 숫자들끼리는 분류를 가능하게 하는 특징들의 공집합 요소가 많게 되므로 같은 부류의 숫자들에서 적당한 분류 방법을 구현하지 못하게 되면 오인식률이 급증하게 될 것이다. 오인식률을 줄이기 위해 위에서 언급한 것처럼 특징점이 많은 숫자들을 분류 순서의 앞부분에 두었으며 예를 들어 같은 부류인 0과 1에서 1보다는 0이 특징점이 많으므로 0을 분류의 앞쪽에 새운다. 이 부류의 패턴이 입력되었을 때 0분류기를 통과 하였다면(0이 아니라면) 1일 확률이 매우 높아질 것이다. 같은 방식으로 2, 3, 8에서 2보다는 3이 많은 특징점을 가지고 있으므로 3을 앞쪽에 새운다. 모든 분류기들을 모아 놓은 것을 “인식기”라고 했을 때 인식기 안에서의 분류기의 배치 순서는 다음과 같다.

0 - 1 - 3 - 하트 - 2 - 8 - 6 - 9 - 7 - 4 - 5

4. 모의 실험 및 분석

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 9) 보정 전과 보정 후의 에러율 비교

그림 9에서와 같이 보정 전과 보정 후를 비교하면 이동 시와 정지 시 낮은 에러율을 보여준다. 아래로 이동 시(Z축 이동 시) X축의 가속도 값이 변하게 되는 에러율은 다소 높은 편이나 이를 상쇄시키기 위해 제안한 전처리 알고리즘을 적용하면 분류에 미치는 영향이 적어졌다.

제스쳐 종류	샘플수	인식 개수	인식 확률	오인식 개수	오인식 확률
0	76	76	100%	0	-
1	54	54	100%	0	-
2	78	78	100%	0	-
3	79	79	100%	0	-
4	77	77	100%	0	-
5	44	43	97%	1	3%
6	106	102	96%	4	4%
7	86	85	98%	1	2%
8	88	88	100%	0	-
9	106	105	99%	1	1%
하트	93	93	100%	0	-
	총887개				

(그림 10) 각 제스쳐별 인식확률과 오인식 확률

그림 10에서 테스팅을 위한 총 샘플의 수가 887개이다. 이는 14명의 다양한 사람들에게서 수집한 샘플이며 제스쳐 입력을 하기 전 가이드라인에 대해 2~3분 사전 설명을 한 후 샘플데이터 수집을 진행하였다. 1번 제스쳐 샘플수가 상대적으로 적은데 다른 숫자의 제스쳐에 비해 입력 패턴이 단순하여 인식하기가 가장 쉬우므로 거의 언제나 100%의 인식률을 보장하므로 54개까지만 DB에 저장을 하였다. 그림 10에서와 같이 각 제스쳐의 인식률은 매우 높은 편이며 가이드라인을 준수하였을 때 각 제스쳐 별로 최대 100%, 최저 96%의 인식률을 기록하였다.

5. 구현한 Android 응용

5.1 Gesture Alarm Call

(그림 11) Gesture Alarm Call 메인화면

(그림 12)제스쳐 입력을 통한 전화연결

그림 11은 Gesture Alarm Call 앱의 메인화면이며 그림 12는 바로전화걸기 기능에 대한 설명이다.

(그림 13)제스쳐 입력을 통한 산술 알람 기능

그림 13은 제스쳐 알람 기능에 대한 설명이다. 2010년 12월에 Android마켓에 릴리즈하였으며 인식 만족도는 본 연구에서 보다 좋을 것으로 예상한다.

본 연구에서 제안한 알고리즘이 실제 모바일 장치에 적용하도록 Android기반 애플리케이션을 구현하였다. 스마트폰에 내장된 가속도센서를 이용하여 각각의 제스처를 입력하고 이를 인식해 바로 전화걸기 기능과 산술문제의 답을 제스쳐로 입력해야지만 알람이 꺼지는 알람기능을 제공하는 애플리케이션을 구현하였다. 스마트폰 장치는 삼성전자의 SHW-M110S(갤럭시S)를 사용하였으며 OS Version은 Android 2.1(Eclair)이다.

6. 결 론

본 연구에서는 가속도 센서를 이용하여 숫자와 특정 기호를 인식하기 위한 사전 보정 알고리즘과 분류 알고리즘에 대해 제안하였다. 보정 전과 보정 후의 X축과 Z축의 에러율 실험 결과를 통해 신뢰성있는 전처리된 데이터가 생성됨을 알 수 있었다. 또한 전처리된 데이터에 적용할 정규화와 분류 알고리즘으로 구현한 인식기가 높은 인식확률을 보여주었다. 이를 사용하는 Android 애플리케이션을 구현함으로써 모바일 환경에서도 실시간으로 제스쳐 인식이 적용 가능함을 확인하였다. 본 연구에서는 제한된 입력패턴에 대해서만 분류와 인식이 가능하다. 그래서 앞으로 연구방향은 사람들마다 숫자와 문자를 쓰는 방식은 여러가지이며 이에 대한 표준을 말하기는 어려우므로 보다 자연스러운 인식과 인식률이 높은 애플리케이션을 구현하기 위해서는 다양한 패턴들에 대한 분류기를 구현하여야 한다.

참고문헌

[1] 백종훈,이기혁(2002). 가속도 센서 기반 휴대폰 단말기 사용자 인터페이스 구현, 한국정보과학회 추계 학술발표논문집, 29-02, 223 - 225.

[2] 신항식,이영범,이명호(2007). 2축 가속도 신호와 Extreme Learning Machine을 사용한 행동패턴 분석 알고리즘, 대한전기학회 전기학회논문지, ,56-07, 1324-1330.

[3] 유향미,서재원,차은종,배현덕(2008). 3축 가속도 센서를 이용한 보행 횟수 검출 알고리즘과 활동 모니터링, 한국콘텐츠학회 한국콘텐츠학회논문지, 8-08, 253-260.

[4] Kimberly Tuck.(2007), Implementing Auto-Zero Calibration Technique for Accelerometers, Accelerometer Systems and Applications Engineering Tempe, AZ. Freescale Semiconductor AN3447.

[5] Kimberly Tuck.(2007), Tilt Sensing Using Linear Accelerometers, Accelerometer Systems and Applications Engineering Tempe, AZ. Freescale Semiconductor AN3461.

[6] Kurt Seifert and Oscar Camacho(2007), Implementing Positioning Algorithms Using Accelerometers Freescale Semiconductor AN3397.

[7] Matthias Rehm, Nikolaus Bee, Elisabeth Andre(2008), Wave Like an Egyptian-Accelerometer Based Gesture Recognition for Culture Specific Interactions, British Computer Society Swinton.

[8] Tian Seng Leong, Jeffrey Lai, Jeffrey Panza(2008), Wii Want to Write: An Accelerometer Based Gesture Recognition System, Electrical and Computer Engineering Dept. Carnegie Mellon University,

저자소개

배 석 찬

1988 전남대학교 전산통계학과(이학석사)

1995 전남대학교 전산통계학과(이학박사)

1995 - 현재 군산대학교 컴퓨터정보공학과 교수

관심분야 : 의료정보. 정보보안, 데이터베이스,애플리케이션 개발

e-mail : scbae@kunsan.ac.kr

강 보 경

2011 군산대학교 컴퓨터정보공학과(공학석사)

2011 - 현재 삼성전자 연구원

관심분야 : 스마트폰 플랫폼, 스마트폰 애플리케이션 개발, 패턴인식

e-mail : le2090@hanmail.net