6축 자이로 센서를 활용한 좌표 추출 및 당구 스트로크 자세 분석에 관한 연구

김정환, 김진형, 정윤호, 조형준, 김웅섭
동국대학교 정보통신공학과
e-mail : wjdghks94k@naver.com , rlawlsgud424@naver.com, saha382@naver.com, ssfhqhzkq@naver.com, woongsup@dongguk.edu

A Study on Coordinate Extraction and Ball Stroke Posture Analysis Using 6-Axis Gyro Sensor

Jung-Hwan Kim, Jin-Hyoung Kim, Yun-Ho Jung, Hyung-Joon Cho, Woongsup Kim
Dept. of info-communication, Dongguk University

요 약

본 연구를 통해 당구를 처음 접하는 사람들이 혼자서도 올바른 스트로크 자세를 연습할 수 있도록 하기 위한 시스템을 설계하였다. 6 축 자이로 센서를 활용하여 가속도, 각속도 센서 값을 안드로이드와 BLE 통신으로 수집하고 그 값으로 스트로크의 속도와 각도, 방향을 계산하여 유사율을 나타낸다. 또한 터치센서에 스트로크 시에 타격감을 주기 위하여 모바일의 진동을 울려주며, 센서에 터치 된 값을 이용하여 사용자가 실제 타격한 당구공의 타점을 모바일에서 실시간으로 보여준다. 동시에 앞서 계산된 유사율을 그래프와 수치상으로 확인할 수 있다. 모범 자세와 비교한 피드백을 통하여 올바른 스트로크 자세를 익힐 수 있도록 도와준다.

1. 서론

당구는 실내 스포츠로써 꾸준히 주목 받고 있으며 남녀노소 쉽게 접할 수 있는 스포츠이다. 우리는 어떤 스포츠를 시작하거나 배울 때 가장 먼저 자세를 익히게 된다. 당구 또한 마찬가지로 당구공의 여러 방향의 타점을 타격하고 제대로 즐기기 위해선 기본적으로 올바른 스트로크 자세를 익혀야 한다. 하지만 기존의 당구 관련된 애플리케이션은 시뮬레이션으로 당구공을 타격하였을 때의 경로를 보여준다. 이런 방법으로는 실제 당구를 칠 때 가장 중요한 기본자세를 연습할 수 없으며 몸으로 느낄 수 없다.

그래서 우리는 당구를 처음 접하는 사람들이 혼자서도 올바른 당구 스트로크 자세를 연습할 수 있도록 도와주기 위하여 6 축 자이로 센서와 터치센서를 활용하여 모범 자세와의 유사율을 비교해주며, 터치센서를 타격하였을 때 실제 당구공의 타점의 위치를 애플리케이션 상에서 보여주게 된다. 이와 같이, 이 논문에서 제시하는 애플리케이션은 당구 입문자에게 보다 나은 자세 연습 환경을 제공할 수 있다.

2. 요구사항

2-1) 당구 큐에 6 축 자이로 센서의 방향을 일정하게 부착해야 한다.

자이로 센서의 기준 방향에 따라 발생되는 각속도의 값이 다르므로 사용자가 센서의 방향을 고려해야 한다.

2-2) 디바이스 간의 통신이 원활해야 한다.

애플리케이션이 시스템 구조대로 진행되기 위해서는 아두이노와 안드로이드의 블루투스 통신, 6 축 자이로 센서와 안드로이드의 BLE 통신이 실시간으로 원활하게 통신 되어야 한다.

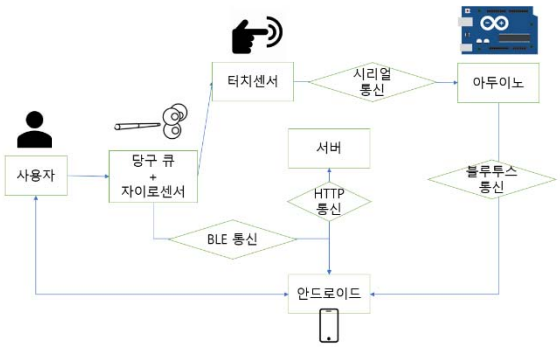
2-3) 모범 자세에 대한 스트로크의 유사율이 정확해야 한다.

사용자가 유사율을 통해 본인의 스트로크 자세를 판단하기 위해서는 정확한 알고리즘을 통한 스트로크 간의 유사율을 계산해주어야 한다. 특히, 서로 다른 데이터의 길이를 가진 스트로크를 명확하게 비교해주어야 할 알고리즘이 필요하다.

2-4) 사용자의 스트로크에 대한 피드백이 필요하다.

사용자가 애플리케이션을 통해 스트로크의 흔들림을 교정하기 위해서는 자세를 교정할 수 있는 피드백이 필요하다. 당구 큐가 상하좌우로 흔들리는 것을 판단하고 해결책을 제시해주어야 한다.

요구사항에서 분석한 결과를 바탕으로 이번 연구에서 설계한 스트로크 자세 분석 시스템의 설계도와 전반적인 구동 방법, 사용된 기술, 애플리케이션 구조를 설명한다. 본 시스템의 설계도는 (그림 1)과 같다.



(그림 1) 시스템 설계도

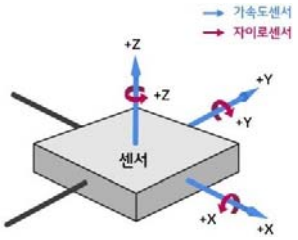
사용자가 6 축 자이로 센서(이하 '자이로 센서')가 부착된 당구 큐를 이용하여 스트로크를 한다. 정해진 기준 방향에 맞추어 자이로 센서를 큐에 부착해야 기준 자세와 비교할 정확한 데이터를 추출할 수 있다. 스트로크 시에 큐에 달린 자이로 센서로부터 추출된 가속도x, y, z, 각속도x, y, z 의 데이터들을 BLE 통신을 통해 안드로이드로 전송한다.

아두이노의 터치센서로 제작된 당구공을 타격하게 되면, 터치센서에서 발생된 (x, y) 형태의 좌표 값이 시리얼 통신을 통해 아두이노로 전송된다. 아두이노의 데이터를 블루투스 통신으로 안드로이드에 전송해줌과 동시에 안드로이드에서 자이로 센서와의 BLE 통신을 종료한다. 사용자의 스트로크에 대한 데이터를 실시간으로 서버에 저장하기 위하여 HTTP 통신을 사용한다.

3. 스트로크 자세 분석

4-1) 가속도 값과 각속도 값의 각도 변환

6 축 자이로 센서의 경우 (그림2) 와 같이 가속도x, y, z, 각속도x, y, z 총 6 개로 구성된다.



(그림 2) 6 축 자이로 센서

가속도 센서의 경우 x, y, z 에 입력되는 센서의 값을 그대로 사용하지 않고 각 축에 대해 센서가 회전한 각도로 변환해주는 작업을 진행하였다. Roll 은 직선 y 축이 곡선 x 축에 따라 기울어진 각도로서 센서의 '좌우로의 기울어짐' 을 의미한다. Pitch 는 직선 x 축이 곡선 y 축에 따라 기울어진 각도로서 센서의 '앞뒤로의 기울어짐' 을 의미한다. Roll 과 Pitch 값은 (수식 1) 식을 통해 계산하였다.

$$Roll = \arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right)$$

$$Pitch = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right)$$

$$A_k = k\text{축 가속도 } (k = x, y, z)$$

(수식 1) Roll, Pitch

Yaw 는'z 축 방향으로의 기울어짐' 을 의미한다. 가속도 센서의 경우 z 축에 대한 회전감지가 어려워 각속도 센서를 이용하여 구할 수 있다. 하지만 각속도 센서의 값의 경우 누적오차의 문제가 발생하여 지자기 센서가 추가적으로 필요하므로 Yaw 값은 사용하지 않았다. 각속도 센서의 경우에도 x, y, z 에 입력되는 센서의 값을 그대로 사용하지 않고 단위 시간당 각속도 센서의 값의 변화량에 대해 적분하여 Gyro 각도로 변환해주는 작업을 진행하였다. Gyro 각도의 경우 (수식 2) 를 통해 계산해주었다.

$$k\text{축 } Gyro_n = k\text{축 } Gyro_{n-1} + k\text{축 } \omega_n \times \Delta t$$

$$k = x, y, z$$
$$Gyro_n = \text{현재의 각도}$$
$$\omega_n = \text{현재의 각속도}$$
$$\Delta t = sampling\ rate\ (\approx 0.02)$$

(수식 2) Gyro 각도

최종적으로 사용자가 스트로크를 진행한 후의 Roll, Pitch, Gyro(x), Gyro(y), Gyro(z) 총 5 개의 데이터를 각각 5 개의 배열로 불러들여 작업을 진행하였다.

4-2) 상보필터

가속도 센서의 경우 긴 시간 동안에는 비교적 값이 정확하지만 짧은 시간 동안에는 노이즈와 짐벌락(두 회전축이 겹치는 현상)이 발생하는 문제점이 있다. 각속도 센서의 경우 짧은 시간 동안에는 비교적 값이 정확하지만 긴 시간 동안에는 적분 드리프트(적분 오차가 누적되어 값이 한쪽으로 이동하는 현상)이 발생한다.

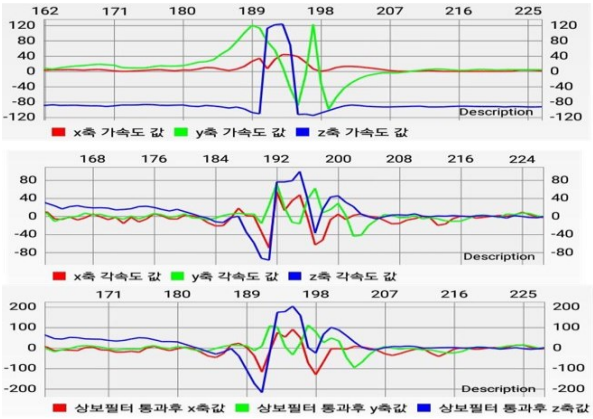
따라서 이를 해결하기 위해 상보필터를 이용하여 필터링 작업을 수행하였다. 상보필터란 상호보완필터로서 가속도 센서와 각속도 센서의 단점을 보완해주고 장점만을 살려 각도로 추정하는 방법을 말한다. 상보필터를 거친 Filtered 각도의 경우 (수식 3)을 통해 계산하였다.

$$x\text{축 } Filtered = \alpha \times x\text{축 } Gyro + (1-\alpha) \times Roll$$
$$y\text{축 } Filtered = \alpha \times y\text{축 } Gyro + (1-\alpha) \times Pitch$$
$$z\text{축 } Filtered = \alpha \times z\text{축 } Gyro$$

$$k\text{축 } Filtered = k\text{축의 필터링된 각도 } (k = x, y, z)$$
$$\alpha\,(\text{필터계수}) \approx 0.98$$

(수식 3) 상보필터

최종적으로 사용자가 스트로크를 진행한 후, 3 개의 축에서 상보필터를 통과한 각도 Filtered(x), Filtered(y), Filtered(z) 총 3 개의 데이터를 각각 3 개의 배열로 불러들여 작업을 진행하였다. 임의의 스트로크에 대해 가속도x, y, z, 각속도x, y, z, Filtered(x), Filtered(y), Filtered(z) 의 그래프를 보면 (그림 3)과 같다.



(그림 3) 상보필터 통과 후 그래프

4-3) EditDistance 알고리즘을 이용한 유사율 비교

안드로이드 스튜디오에서 mp android chart 라이브러리를 이용하여 사용자가 스트로크 시행 시 가속도, 각속도 별로 꺾은선 그래프로 시각화시켰다. 추가로, 모범적인 스트로크의 샘플 그래프를 미리 확보한 후에 사용자의 스트로크 그래프와 비교 분석 후 유사도를 구현하는 알고리즘을 구성하였다. 비교 분석 시에는 (수식 4)와 같이 두 벡터 사이의 각도를 이용하는 코사인 유사도를 이용하였다.

$$\cos\Theta = \frac{\vec{A} \cdot \vec{B}}{|A|^2 \times |B|^2}.$$

(수식 4)

사용자의 스트로크 벡터는 (Filtered(x), Filtered(y), Filtered(z)) 와 같이 3 차원 벡터로 구성하였다. 위 벡터의 경우 Filtered(x), Filtered(y), Filtered(z) 총 3 개의 배열에서 인덱스 순으로 불러들여 인덱스 순으로 불러들여 생성하였다. 모범 스트로크 벡터 역시 동일하게 구성하였다. 두 벡터 사이의 각도는 최소 0 도부터 최대 180 도까지 형성되며 0 도일 경우 두 벡터는 100% 일치하는 것으로 간주하여 1 을 출력해주며 180 도일 경우 두 벡터는 완전히 불일치 하는 것으로 간주하여 -1 을 출력해준다.

따라서 -1 부터 1 까지의 값을 유사도 0%에서부터 100%까지로 매칭해주었다. 하지만 코사인 유사도의 경우 그래프가 유사한데도 불구하고 두 벡터의 개수차이가 클 경우 유사율이 저조해지는 문제점이 발생하였다. 따라서 두 데이터의 개수차이와 관계없이 사용할 수 있는 Editdistance 알고리즘을 이용하였다. Editdistance 알고리즘은 두 문자열의 유사도를 판단하는 알고리즘이다. 유사도를 판단하는 방법은 기준이 되는 문자열과 같게 하기 위해 비교하고자 하는 문자열을 삽입, 삭제, 변경을 몇 번을 수행해줘야 하는지의 최솟값을 구해 그 값을 유사도 판단의 척도로 다룬다. Editdistance 의 경우 (수식 5)의 의사코드를 참고하여 구현하였다.

```
EditDistance(A[1..m], B[1..n]):
  for j ← 0 to n
      Edit[0, j] ← j
  for i ← 1 to m
      Edit[i, 0] ← i
      for j ← 1 to n
          if A[i] = B[j]
              Edit[i, j] ← min {Edit[i − 1, j] + 1, Edit[i, j − 1] + 1, Edit[i − 1, j − 1]}
          else
              Edit[i, j] ← min {Edit[i − 1, j] + 1, Edit[i, j − 1] + 1, Edit[i − 1, j − 1] + 1}
  return Edit[m, n]
```

(수식 5) Editdistance

Editdistance 알고리즘을 적용시킬 때에는 사용자의 Filtered(x)와 모범 스트로크의 Filtered(x)를 비교한 후에 x 축의 유사율을 출력해주었다. y, z 축 의 경우도 동일하게 진행하였다. 최종 유사율은 x 축, y 축, z 축의 유사율의 평균치로 출력해주었다.

유사율을 백분율(%)로 환산해주는 작업은 (수식 6)을 참고하여 계산하였다.

$$\text{유사율}(\%) = \frac{len - e}{len} \times 100$$

len = 사용자 $Filtered$ 배열과 모범 $Filtered$ 배열 중 긴 값
e = 최소편집거리

(수식 6) 유사율 (%)

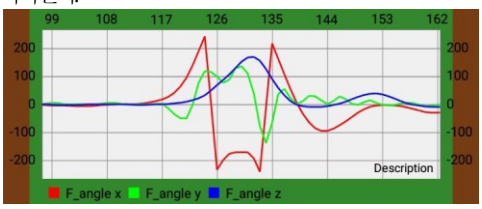
4-4) 큐의 움직임에 관한 피드백

사용자가 스트로크 시행시에 큐가 위로 흔들렸을 경우 (그림 4)와 같이 나타난다.

(그림 4) 큐가 위로 흔들렸을 때

Filtered(x) 의 값(적색 그래프) 이 감소하다가 위로 흔들리는 지점에서 값이 순간적으로 증가했다가 다시 감소하는 양상을 보인다.

사용자가 스트로크 시행시에 큐가 아래로 흔들렸을 경우 (그림 5)와 같이 나타난다.



(그림 5) 큐가 아래로 흔들렸을 때

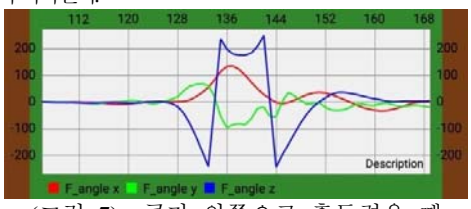
Filtered(x) 의 값(적색 그래프) 이 증가하다가 아래로 흔들리는 지점에서 값이 순간적으로 감소했다가 다시 증가하는 양상을 보인다.

사용자가 스트로크 시행시에 큐가 오른쪽으로 흔들렸을 경우 (그림 6)과 같이 나타난다.

(그림 6) 큐가 오른쪽으로 흔들렸을 때

Filtered(z) 의 값(파랑색 그래프) 이 증가하다가 오른쪽으로 흔들리는 지점에서 값이 순간적으로 감소했다가 다시 증가하는 양상을 보인다.

사용자가 스트로크 시행시에 큐가 왼쪽으로 흔들렸을 경우 (그림 7)과 같이 나타난다.



(그림 7) 큐가 왼쪽으로 흔들렸을 때

Filtered(z) 의 값(파랑색 그래프) 이 감소하다가 왼쪽으로 흔들리는 지점에서 값이 순간적으로 증가했다가 다시 감소하는 양상을 보인다.

따라서 큐가 위로 흔들리는 경우와 왼쪽으로 흔들리는 경우에는 각각 Filtered(x), Filtered(z) 값이 설정한 max 값을 도달할 때로 판별해주었다. 큐가 아래로 흔들리는 경우와 오른쪽으로 흔들리는 경우에는 각각 Filtered(x), Filtered(z) 값이 설정한 min 값을 도달할 때로 판별해주었다. max 값과 min 값의 경우 각각 50, -50 으로 설정해주었다.

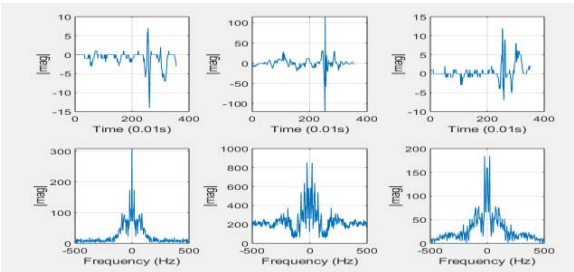
4-5) 밀어치기와 끌어치기에 관한 피드백

사용자가 스트로크 시행시에 밀어치기와 끌어치기를 할 경우 모두 아두이노 터치센서에서의 좌표 값을 기준으로 판별해주었다. 밀어치기의 경우 설정한 좌표이상으로 인식될 경우 사용자가 밀어치기의 스트로크를 시행한 것으로 판별하였다. 끌어치기의 경우 설정한 좌표 이하로 인식될 경우 사용자가 끌어치기의 스트로크를 시행한 것으로 판별하였다.

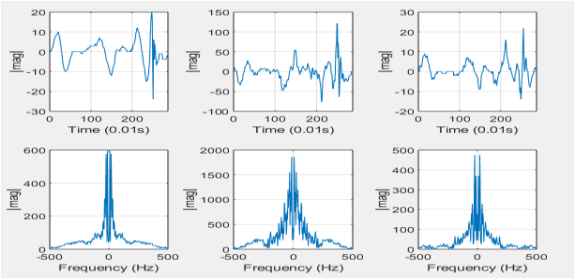
밀어치기의 경우, 좌표는 y 값이 2~4 일 때로 하고 끌어치기의 경우, y 값이 10~12 일 때로 설정하였다.

4-6) 주파수를 이용한 스트로크 분석

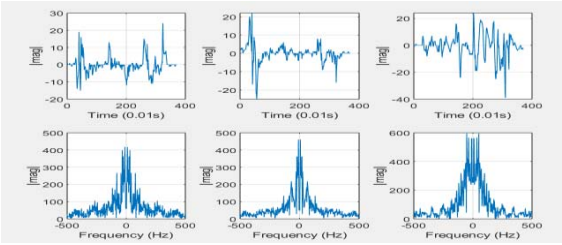
6축 자이로 센서의 값을 일정 시간 간격으로 받아오며 신호를 분석하는 프로젝트를 진행하였다. Time Domain에서 x축, y축, z축의 각속도 및 가속도에 대한 그래프 분석을 해본 결과 당구의 특성상 큐의 움직임이 제한적이면서 상대적으로 다른 운동에 비해 일관적이기 때문에 분석하는 것에 제한이 있었다.



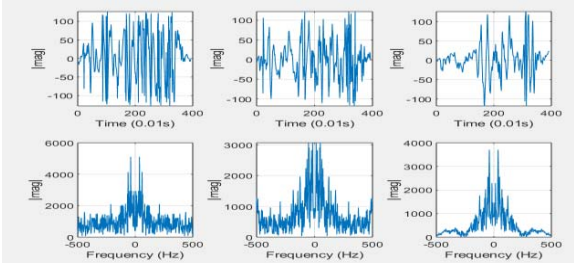
(그림 8) 올바른 스트로크



(그림 9) 흔들린 스트로크



(그림 10) 올바른 스트로크



(그림 11) 흔들린 스트로크

그림 8, 그림 9은 당구를 잘 치는 사람에 대한 올바른 스트로크와 흔들린 스트로크이며, 그림 10, 그림 11는 당구 초보자의 올바른 스트로크와 흔들린 스트로크에 대한 결과이다. 그림은 위에서부터 각속도에 대한 x축, y축, z축의 Time Domain에서의 값이며 아래의 3가지 그래프는 Frequency Domain에서의 값이다. 스트로크 시에 큐가 흔들림에 따라 각속도의 값이 큰 폭으로 발생하여 주파수 변환을 했을 때 Frequency Domain에서 magnitude 값이 더 커지는 것을 볼 수 있다. 그림1의 경우에는 x축, y축, z축 순서대로 약 200,800,200의 값을 갖는 반면, 흔들린 결과인 그림 8의 경우에서는 600, 2000, 500까지 값이 올라갔다.

그림 8, 9에 비해 그림 10, 11은 초보자가 친 스트로크이기 때문에 상대적으로 흔들림이 많았고 이에 따라 각속도의 변화량이 커져서 주파수축에서 magnitude 값이 올바른 스트로크의 경우에는 400,500,600 흔들린 스트로크의 경우에는 4000,3000, 3000까지 커졌다.

이에 따라 우리는 스트로크 시에 흔들림이 생기면 각속도의 값이 발생하게 되고 Time Domain에서 값이 발생함에 따라 주파수 변환을 하였을 때 Frequency Domain에서 magnitude 값이 커지는 것을 알 수 있다. 우리는 주파수 변환을 해서 그래프를 plot 하였을 때 FFT(Fast Fourier Transform) 방법을 사용하였으며 주파수 축에서의 값을 더 눈으로 잘 보기 위하여 matlab에서 shiftfft 함수를 사용하였다.

4. 결론

본 논문에서는 6 축 자이로 센서를 이용하여 유사율을 계산해주는 것과 터치센서를 이용하여 타점을 보여주는 것을 목표로 하였다. 초보자를 위한 당구 스트로크 자세 교정을 목표로 하였기 때문에 기본기를 연습하는 것을 중점으로 잡았다.

그래서 다른 당구 애플리케이션에 있는 시뮬레이션 기능을 구현하지 않았다. 한계점으로는 터치센서를 실제 공에 적용하지 못한 점이다. 실제 공에 내장된다면 본 논문의 한계점을 극복할 수 있을 것으로 기대된다.

5. 사사

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW 중심대학지원사업의 연구결과로 수행되었음"(2016-0-00017)

참고문헌

-Alan V.Oppenheim, Signal and Systems 2nd edition, Prentice Hall, 2008

-Simon Haykin and Michael Moher, Communication Systems, John Willey and Sons, 2010

-정재곤, Do it! 안드로이드 앱 프로그래밍, 이지스퍼블리싱, 2019