**최적화된 마우스 전달함수 세팅은 E스포츠 플레이어들의 실력에 유의미한 개선을 가져오는가?**

팀 APEX:

컴퓨터과학과 김령태

컴퓨터과학과 김민수

컴퓨터과학과 노준영

담당교수 이병주 교수님

Yonsei Esports Lab

**목차**

**1. 연구목표**

**2. 연구의 필요성**

**2.1 기존의 전달함수**

**2.2 게임 프로그램의 전달함수 override 불가**

**2.3 Autogain 함수의 Target 특정**

**3. 연구내용**

**3.1 준비물 – 듀얼 센서 마우스**

**3.2 AutoGain 설계 및 구현**

**3.3 연구 검증**

**4. 연구 결과**

**5. 추후 연구 가능성**

**1. 연구목표**

**- E스포츠 플레이어들에게 개인별로 최적화된 마우스 전달함수 세팅**

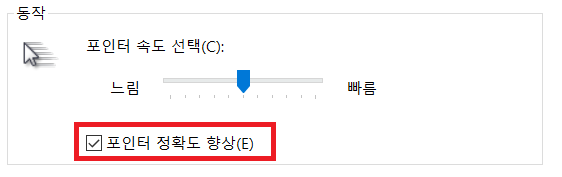
전달함수란 마우스가 실제로 움직인 거리와 마우스 커서가 공간상에서 움직인 거리 사이의 상관관계를 결정한다. 이것을 개인별로 최적화해 더 나은 게임 performance를 낼 수 있게 한다.

**2. 연구의 필요성**

**2.1 기존의 전달함수**

기존의 전달함수는 preset이 정해져 있다. 그 정도를 바꾸는 것도 힘들며, 어떠한 전달함수 값이 사용자한테 맞는지도 알 수 없는 상황이다.

텍스트, 장치, 측정기, 조종판이(가) 표시된 사진

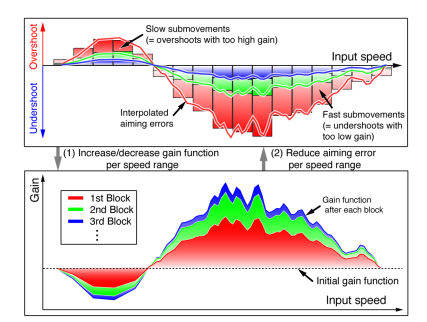
자동 생성된 설명

**2.2 게임 프로그램의 전달함수 override 불가**

위와 같이 게임들에서 확인할 수 있는 마우스 전달함수는 소프트웨어 레벨에서의 default 전달함수이다. 이러한 전달함수는 일반적인 소프트웨어로는 우회하게 힘들게 설계가 되어있다.

**2.3 Autogain 함수의 Target 특정**

Autogain함수란 속도와 gain값을 통하여 자동으로 오차를 계산, 최적화된 전달함수 값을 찾아주는 프로그램이다. 기존의 Autogain함수는 목표값(Target)과의 오차를 계산하여 overshoot, undershoot을 보정하는 방식이었다.



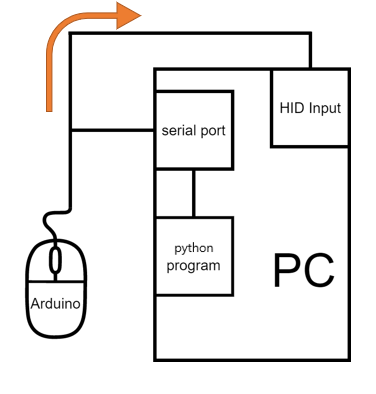
기존의 오차값 보정 과정

그러나 이 방식은 Target의 위치를 특정해야만 하는데 클릭한 위치가 꼭 Target의 위치와 일치하지 않을 수도 있다. 따라서 Target의 위치를 특정하지 않고 전달함수를 최적화시킬 수 있는 방식을 고려해야한다.

**3. 연구내용**

**3.1 준비물 – 듀얼 센서 마우스**

앞서 언급했듯이 소프트웨어 레벨에서 프로그램들의 기존 전달함수를 우회하기는 복잡하다. 그러나 자체 아두이노 칩을 내장하고 있는 이 마우스를 사용하면 하드웨어 레벨에서 전달함수를 제어가능 하게 해준다. 따라서 일련의 최적화 과정을 가능하게 하는 핵심적인 장비이다.액세서리, 가방이(가) 표시된 사진

자동 생성된 설명

**Arduino Device:**

PMW3360\_dualsensor.ino

보드: **SparkFun Pro Micro.**

**프로세서: ATmega32U4 (5V, 16 MHz)**

**3.2 AutoGain 설계 및 구현**

**3.2.1 Serial Port를 통해 마우스의 움직임 정보(dx, dy, dt, 클릭)를 python 프로그램에 저장**

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

아두이노와 듀얼센서 마우스 간의 통신을 Serial port를 통해 구현하였다. 마우스로부터 일정 시간 간격의 마우스의 움직임 정보(dx,dy)와 클릭 유무를 수신한다. 마우스의 움직임 정보는 향후 최적화 과정에 이용할 속도 값을 얻기 위해서이고 클릭 유무는 마우스의 움직임의 기준점을 잡아 submovement화 하기 위해서이다.

전달함수 최적화 과정을 아두이노에서 처리하기에는 칩셋의 성능이 매우 부족하므로 아두이노(마우스)<->Python의 과정을 거치기로 하였다. 따라서 같은 serial port를 python에서 열어 마우스의 정보를 아두이노를 통해 수신하였다.

텍스트이(가) 표시된 사진

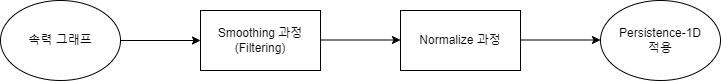
자동 생성된 설명

python을 이용한 속력 변환 코드

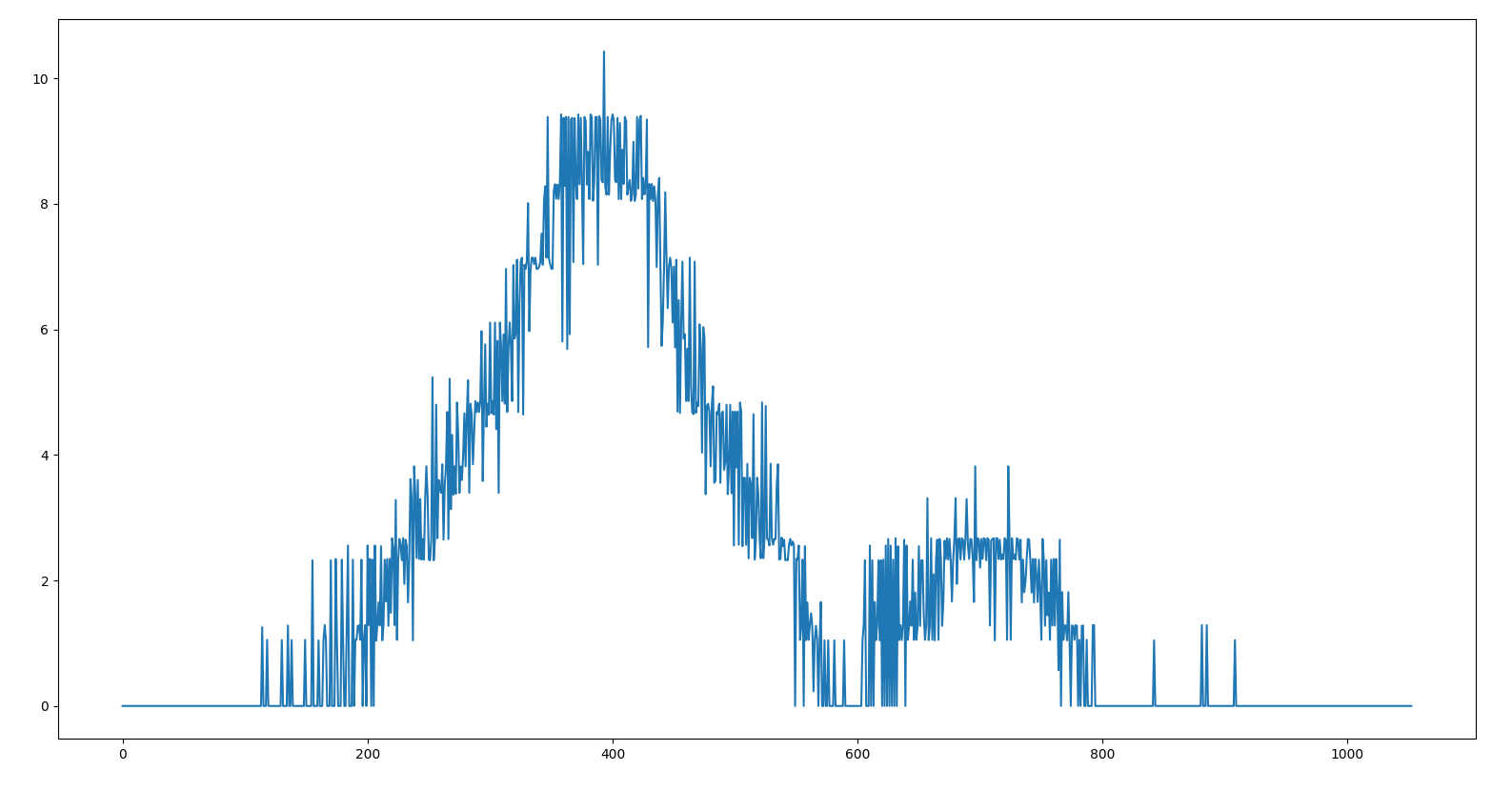
마우스의 기본 cpi값과 아두이노로부터 수신한 마우스의 이동정보(dx,dy)를 활용해 이것을 실제 마우스 이동거리(distance)로 나타내어, 정확한 속력(velocity)값으로 변환해주었다.

**3.2.2 클릭과 클릭 사이의 속도 변화에서 submovement를 추출**

앞서 말했듯이 최적화 과정을 진행할 Autogain함수는 submovement라고 불리는 수많은 마우스 움직임들을 대입해주어야 한다. 따라서 변환해는 속력값들로부터 submovement를 찾아내야 하는데 이것을 persistence-1D 알고리즘을 통해 수행한다.



위의 순서도는 persistence-1D과정이 어떻게 수행되는지를 보여준다. persistence-1D알고리즘은 그래프로부터 극소-극대-극소 값을 찾아내어 해당 구간을 submovement로 분류하는데, 따라서 속력그래프로부터 전처리 과정이 요구된다. 그렇지 않다면 수많은 submovement가 나오기 때문이다.



smoothing 과정을 거치지 않은 그래프

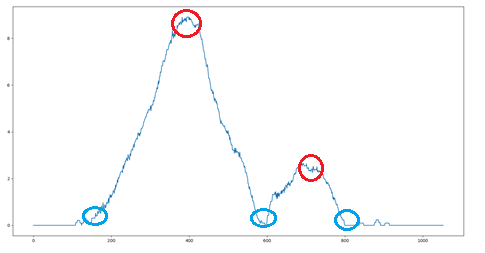
테스트 결과, 예상보다 미세 submovement가 매우 많이 관측되었고, 처음엔 gaussian kernel을 통해 smoothing을 수행했으나 좀 더 강력한 filtering이 필요하다고 생각되어 box filter(average filter)를 사용하게 되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

크기 11의 box filter을 통해 smoothing과정을 진행했다.

적당한 수의 submovement를 추출하기 위하여 box filter의 크기도 여러가지를 진행해 보았다. 이때 크기 5, 7은 smoothing의 강도가 너무 약했고 11정도의 크기가 적당하다고 판단되어 크기 11짜리 filter를 사용하였다.

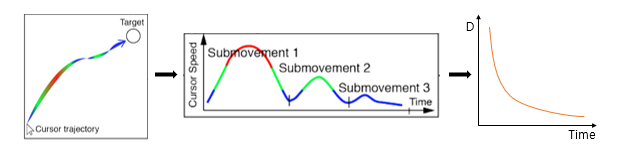


smoothing 과정을 거친 속력 그래프

위와 같이 smoothing과정을 거쳐 submovement의 개수를 의미 있게 줄일 수 있었다. 이 때, 마우스의 최대 속도가 클릭과 클릭 사이의 speed profile 마다 최대 속도가 달라지기 때문에 이 speed profile을 normalize하는 과정을 거쳐 준 후에 persistence-1D를 진행하였다. persistence-1D를 진행하면 극소-극대-극소로 이어지는 submovement들을 모두 찾아주게 된다. 이때 어느 구간을 submovement로 인식할지를 결정하는 persistence값은 0.03으로 고정해주었다.

**3.2.3 추출한 submovement를 기반으로 마우스 전달함수 최적화**

현재 Autogain의 방식으로는 사용자의 클릭이 Target의 위치와 최대한 일치해야 한다. 목표값과 오차값이 생기는 overshoot과 undershoot을 계산하여 보정하기 때문이다. 그러나 Target의 위치를 특정하지 못한다면 기존의 Autogain으로는 오차값 보정에서 한계가 있기 마련이다. 따라서 우리 팀은 Target의 위치와 관계없이 전달함수를 최적화하는 과정을 생각해보았다.



개선된 submovement 보정 과정

Submovement 1,2,3이 모니터상에서 이동한 거리를 각각 D1, D2, D3라고 하자. 마우스가 이상적으로 움직인다면 D1은 다른 D2, D3, D4 등등 보다 훨씬 더 큰 값을 가질 것이다. 이에 착안해 새로운 AutoGain을 설계했다. 먼저 각 마우스의 속력과 그에 따른 gain값을 저장해 둔 후 submovement간의 이동거리의 비율을 분석했다. D1은 인접하는 D0, D2보다 매우 큰 값을 가질 것이기 때문에 Submovement 1을 판별하는 방법은 특정한 양수 n에 대하여 D1/D0>n이고 D1/D2>n을 만족하는지 여부를 보는 것이다. 여러 번의 실험을 통해 Submovement 1을 판별할 수 있는 최적의 n을 찾아보았고 n=2.0일 때 좋은 결과를 나타내었다. 판별한 Submovement 1으로 하나의 마우스 움직임을 구별하고 하나의 마우스 움직임에 속하는 속력의 gain 값을 조정하여 전달함수를 최적화시켰다.

**3.2.4 최적화를 마친 마우스 전달함수를 듀얼 센서 마우스에 적용**

최적화시킨 전달함수를 듀얼 센서 마우스에 전달하는 방법에는 여러가지가 있었다. Python 프로그램에서 최적화를 시킨 후 이 전달함수 인자들을 serial 포트를 통해서 듀얼 센서 마우스의 아두이노 칩에 전달해주는 방식과 최적화에 필요한 인자들만 serial 포트를 통해서 듀얼 센서 마우스의 아두이노 칩에 전달해 준 후 아두이노 칩에서 전달함수를 최적화시키는 방식이 있었다. 전자의 경우에 많은 전달함수 인자들 때문에 실시간으로 최적화하기 힘들다고 판단해 후자의 방법을 택했다. 마우스 속력의 최솟값, 최댓값, D1 그리고 target까지의 거리를 마우스에 전달해준 후 내장된 gain값을 D1과 예상 target의 거리의 차의 일정 비율만큼 조정해주었다. **텍스트, 실내, 컴퓨터, 노트북이(가) 표시된 사진

자동 생성된 설명**

gain값들을 조정되어 전달함수가 변화되면서 최적화되어가는 과정

**3.3 연구 검증**

연구 목표를 완성하기 위해선 E스포츠 플레이어의 실력을 판단해야한다. E스포츠에는 AOS, FPS, TCG 등 다양한 장르도 있고 게임을 혼자하는 싱글플레이 방식도 있고 다른 사람들과 함께하거나 경쟁하는 멀티플레이 방식도 있다. 이렇게 E스포츠 플레이어의 실력을 판단하기에는 많은 변수가 존재한다. 게임 내적인 데이터로 실력을 검증하는 것은 변수가 너무 많아 신뢰성이 떨어진다고 생각해서 실력을 객관적으로 판단할 수 있는 다른 방식을 생각해보았다. 최적화된 마우스 전달함수 세팅은 사용자가 마우스를 원하는 위치까지 빠르고 정확하게 이동할 수 있도록 도움을 준다. 따라서 사용자가 마우스를 이용해 빠르고 정확하게 target을 찾아서 클릭할 수 있으면 게임 실력이 늘었다고 말할 수 있을 것이다.

이에 착안해서 저희는 에임랩이라는 프로그램을 사용하였다. 에임랩 프로그램은 마우스 움직임의 퍼포먼스를 점수를 환산해주고 마우스의 정확도, 그리고 TTK(time to kill: 목표에 다가가는데 걸리는 시간)와 같은 결과를 나타내 주는 프로그램이다. 에임랩 프로그램을 이용해 최적화 전과 최적화 후 점수, 정확도, TTK를 측정하고 서로 비교하였다. 프로그램에 익숙해짐에 따른 점수 변화를 줄이기 위해 최적화 전 프로그램에 익숙해지기 위해 20번 프로그램을 실행한 후 최적화를 진행하였다.

**4. 연구 결과**

|  |  |  |  |
| --- | --- | --- | --- |
| 변화율 | 점수 | 정확도 | TTK |
| 최소 | 10.97 | 1.04 | 8.10 |
| 최대 | 29.21 | 5.01 | 18.21 |
| 평균 | 21.48 | 2.66 | 15.00 |

최적화 전후 지표 비교 종합

전체적인 마우스 움직임의 퍼포먼스를 표시하는 점수가 전체적으로 향상되었다. 정확도는 모든 실험자들이 최적화 전후 모두 높은 정확도를 보여 향상폭이 높지는 않았다. TTK가 평균적으로 15프로 줄어든 것도 고무적인 변화였다.

Autogain은 목표지점을 향해 이동하는 속력을 보정하여 목표에 더욱 빠르게 다가갈 수 있게 해주는 시스템이다. 따라서 비슷한 정확도로 목표지점을 향할 수 있을 때, 그 속력이 빨라진다면 Autogain이 효과가 있다고 판단할 수 있다. 위에서 볼 수 있듯이 Autogain으로 최적화를 진행했을 때 정해진 목표를 찾아가는 속도가 향상되었습니다. 이는 새롭게 설계한 Autogain으로 최적화가 잘 진행되었음을 보여줌과 동시에 e스포츠 플레이어의 실력 향상에 도움을 준다는 것을 의미한다.

**5. 추후 연구 가능성**

연구를 진행하면서 x축과 y축으로만 gain 값을 구분하고 dx 또는 dy의 절댓값이 같을 경우에는같은 gain 값을 사용하였다. 실험을 하면서 마우스 움직임이 각각 왼쪽과 오른쪽, 위쪽과 아래쪽일 때 서로 달랐다. 추후에 이 연구를 더 진행하게 된다면 x축과 y축으로만 나누는 것이 아닌 x축과 y축의 부호가 다를 때도 서로 다른 gain 값을 사용해야 할 것이다.