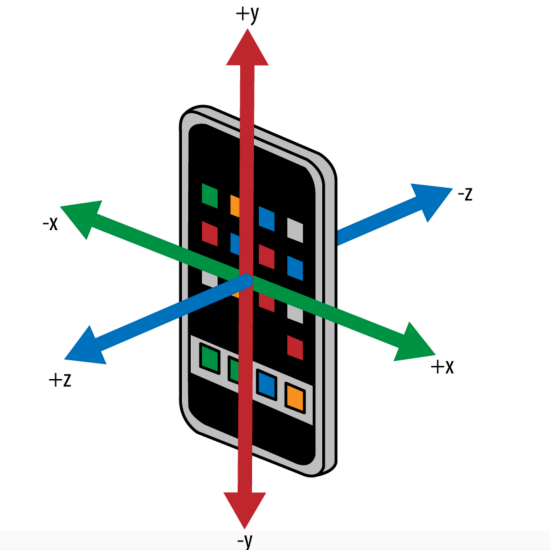
iOS 디바이스를 이용한 마우스 구현

1.개념 및 원리

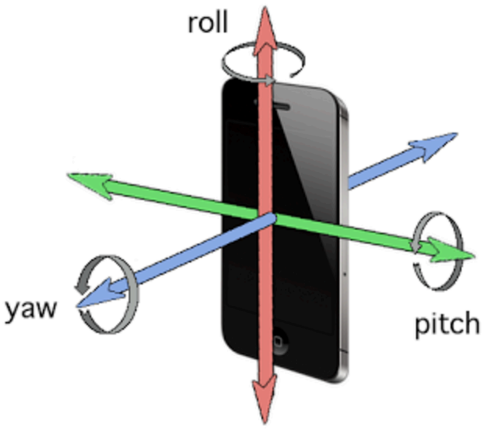
현재 대부분의 모바일 기기에는 자이로센서와 가속도 센서가 탑재 돼 있다. 이 두가지 기술은 사용자가 모바일 기기로 행하는 모션을 인식하는 것을 주요 목적으로 한다. 또한 저전력 블루투스 통신을 사용하여 저전력, 저용량 통신을 이용하여 근거리의 디바이스 간의 통신이 용이해졌다.

이 두 가지 기술을 이용하여 별도의 무소음 마우스를 가지고 다니지 않아도 어느 장소에서든 사용할 수 있는 마우스를 구현하는 것이 이 프로젝트의 목적이다.



모션 인식의 방법 1)

가속도 센서를 이용할 경우, 가속도계가 읽은 입력 값을 이중 적분하여 거리로 변환한다. 자이로 센서를 사용하는 경우보다 기존의 마우스의 작동 방식을 그대로 따름으로써 사용자가 디바이스를 사용하는데 보다 친숙함을 느낄 수 있을 것으로 예상한다.



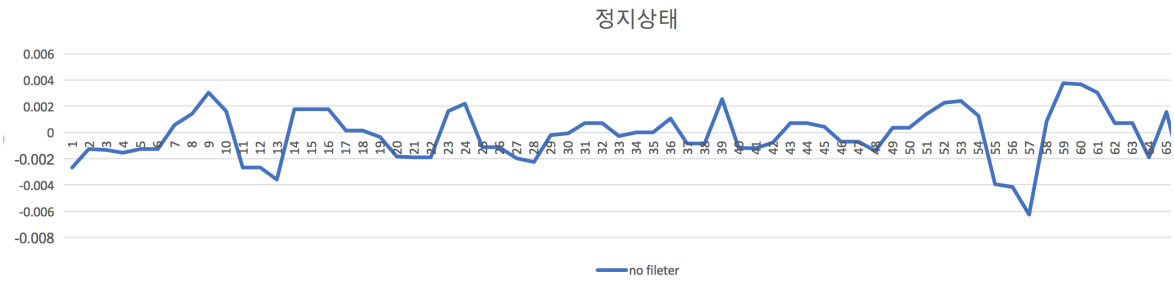
모션 인식의 방법 2)

자이로 센서를 이용할 경우, 모바일 기기의 기울기 변화를 입력 받아 기울기를 측정하고 이를 활용한 조작을 구현한다. 기존의 마우스 방식과는 좀 다르지만, 초기값을 설정하면 프레젠테이션을 위한 레이저 포인터와 컨트롤러의 대용으로도 사용할 수 있을 것으로 예상한다. 초기 상태의 디바이스의 y 축의 +값의 벡터에 직교하는 가상의 면을 둔다. 이 가상이 면은 움직이지 않는다. 이제 디바이스를 z 축이나 x 축을 기준으로 회전시키면 디바이스의 y 축의 +의 벡터와 가상의 면의 접점이 디바이스의 움직임에 따라 바뀌는데, 이것을 마우스 포인터로 하여 화면에 표시한다.

2. 가속도계를 이용한 구현 및 결과

I. 노이즈의 제거

i. Mechanical Filetering Window



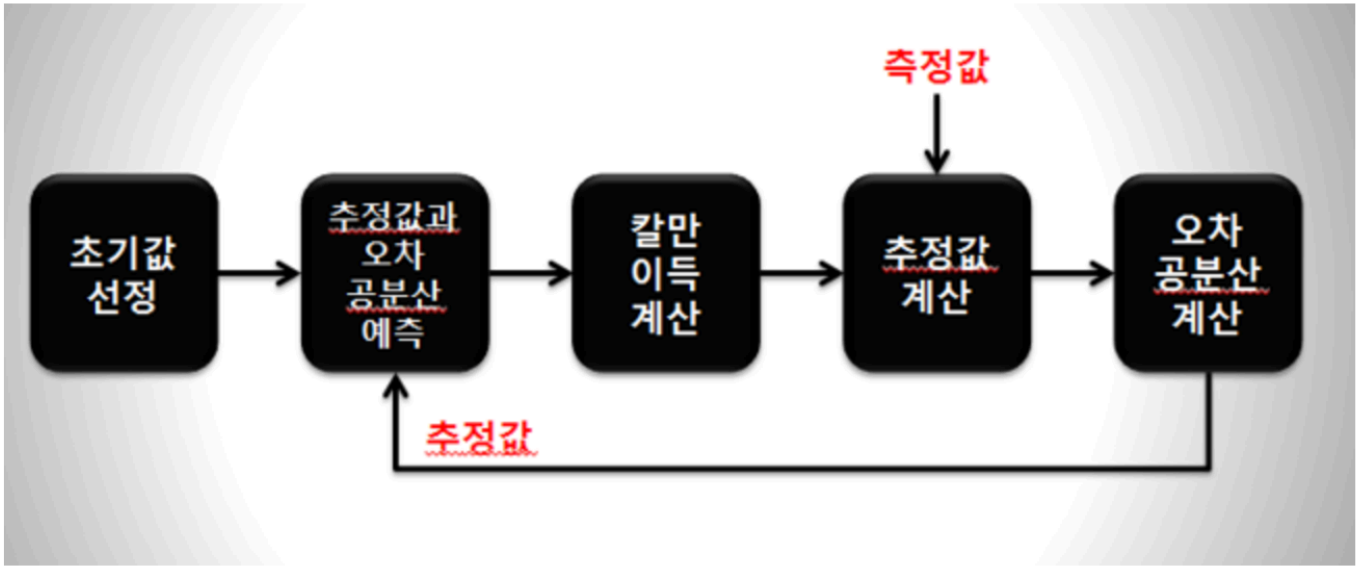
실제로 움직임이 없는 경우 Sample 들의 합이 0 이 되는 것이 이상적이지만, 실제 센서가 가지고 있는 오차와 작은 진동들로 인해 0 이 되지 않는다. 이것을 제거하기 위해 실제 움직임과 노이즈들을 구분하는 윈도우를 구현한다

Mechanical Filtering Window 실제로 정지상태에서 굉장히 좋은 성능을 발휘한다. 그러나 실제 사용에는 부적합한 측면이 크다. 사용자의 마우스의 위치 조작 방식은 다양하다. 같은 구간의 마우스의 이동이라고 할 지라도, 단편적인 시간 단위의 이동 속도를 다르게 동작할 수 있다. 이를 테면 천천히 긴 시간 동안 마우스를 움직일 수 있고, 빠르게 움직일 수도 있고, 빠르게 움직였다가 천천히 이동할 수도 있다. 문제는 역방향의 가속도가 매우 작게 긴 구간 이어지는 경우 발생한다. 어떤 목표지점으로의 정방향으로 동작을 시작한 후, 조작 동작의 속도가 느려지는 구간에서 역방향의 가속도가 매우 미세하게 작용하는 경우 이는 노이즈로 인식되어 정방향의 속도가 감소하지 못한다. 결국 마우스가 계속 움직이게 되고, 오작동을 야기한다. 이는 적분 오차의 문제와 결합하여 조작이 거의 불가능한 상태에 까지 이르게 한다.

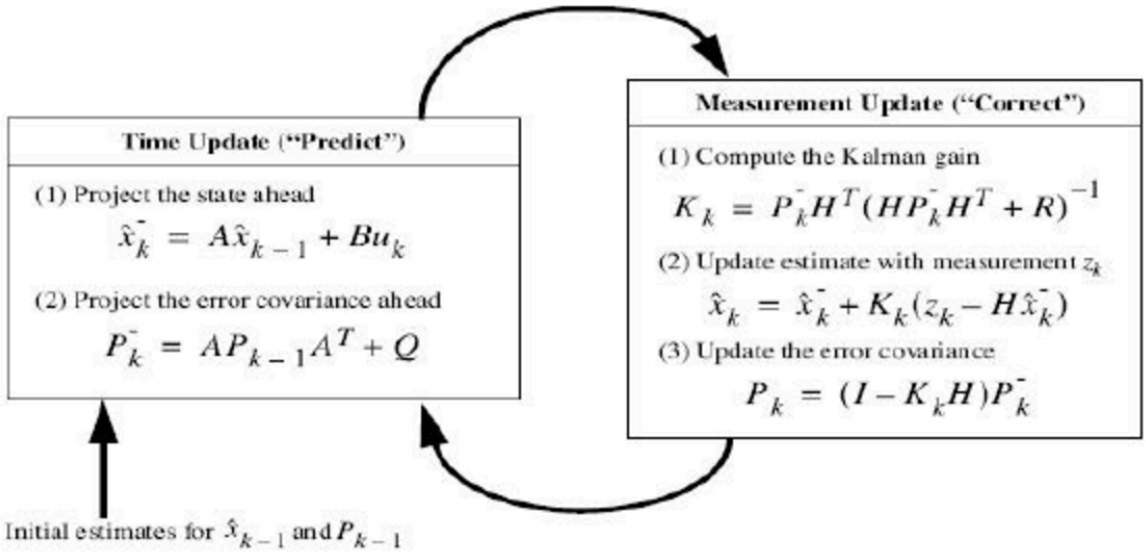
또한 이 Mechanical Filtering Window 를 사용할 경우 작은 단위의 입력 값이 무시되어 실제 디바이스의 운동 구간이 1 차 혹은 상수구간으로 입력되는데, 이는 적분의 오차를 줄이기 위한 Simpson method 의 오차를 오히려 크게 한다.

ii. 칼만 필터

센서가 가지는 오차를 교정하는 필터이다. 잡음이 포함되어 있는 선형 역학계에서 상태를 추적하는 재귀 형식의 필터이다. 다음 단계의 움직임을 예측하고 다음 단계의 입력 값을 업데이트하여 오차를 교정하는 것이 이 필터의 핵심 원리이다. 즉, 다음 시점의 상태와 오차 공분산이 어떤 값이 될 것인지를 예측한 후 측정값과 예측값의 사이를 보상하여 새로운 추정 값을 계산하는 것이다.



<칼만필터 알고리즘의 프로세스>



<전체적인 칼만필터의 흐름도>

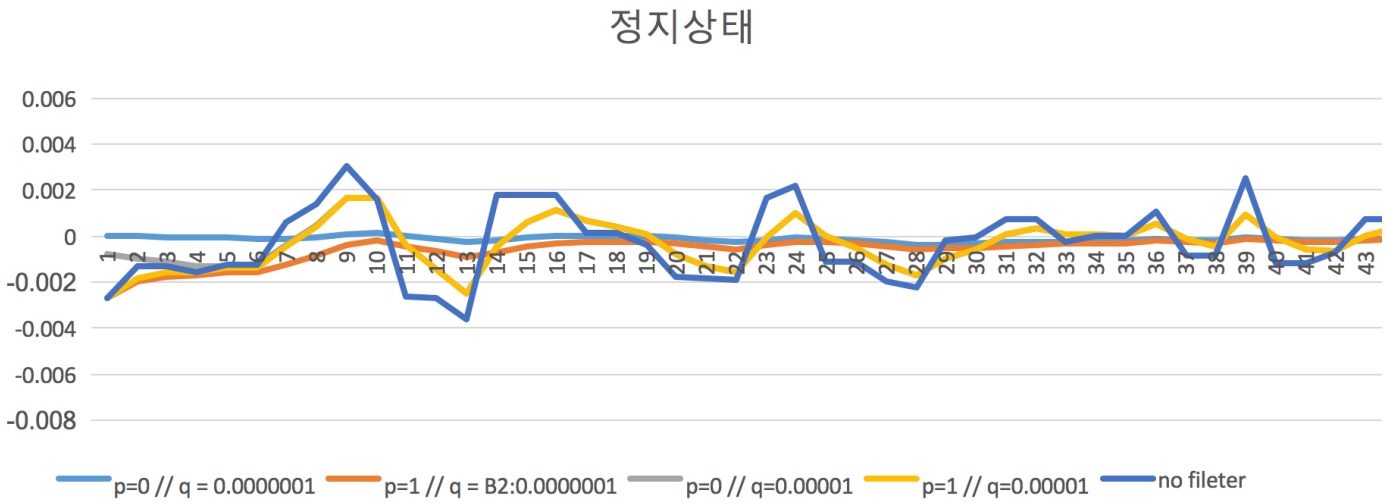
실제로 칼만 필터는 초기 입력값에 따라 그 성능의 달라지게 되는데, 오차값으로는 초기 공분산 오차(P), 시스템 오차(Q), 센서 오차(R)을 입력한다.

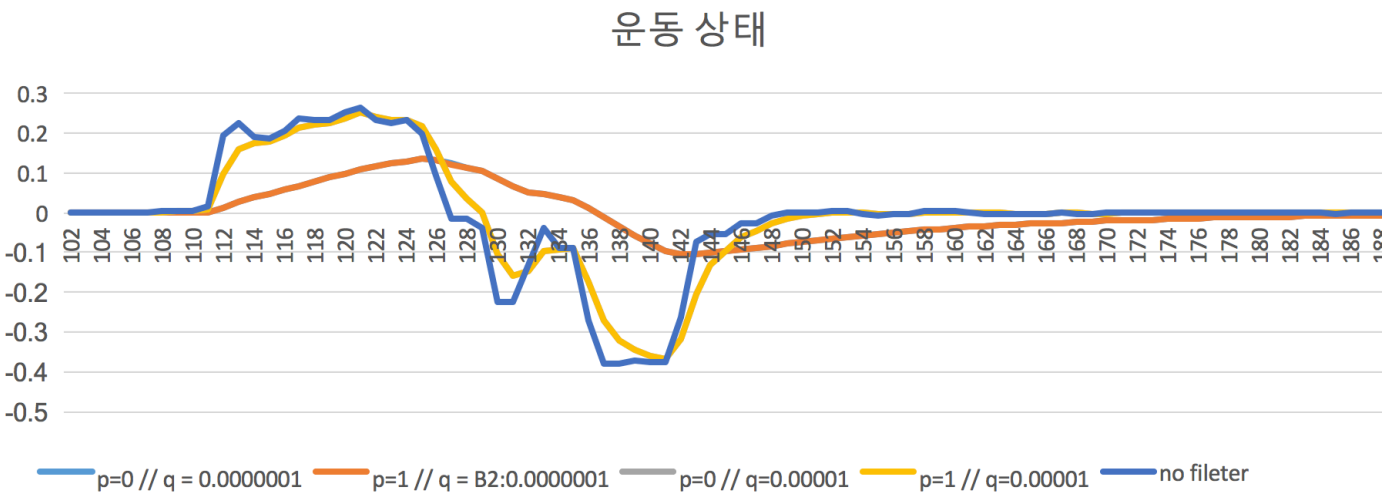
초기 오차 값에 따라 샘플링의 노이즈 값이 교정되는 것을 확인 할 수 있다.

즉, 시스템 오차에 따라 노이즈의 필터링 정도가 판가름 난다. Q 값이 0 작을 수록 노이즈에 대한 교정이 확실해 지지만, 실제 움직임에 대한 반응이 느리게 나타난다.

초기 센서 오차는 일상의 다양한 환경에서의 정지 상태에서의 샘플들의 분산 값을 측정하여 선정 했다.

(정지 상태와 운동 상태의 필터 초기값과 필터 미적용 상태를 그래프로 표현)

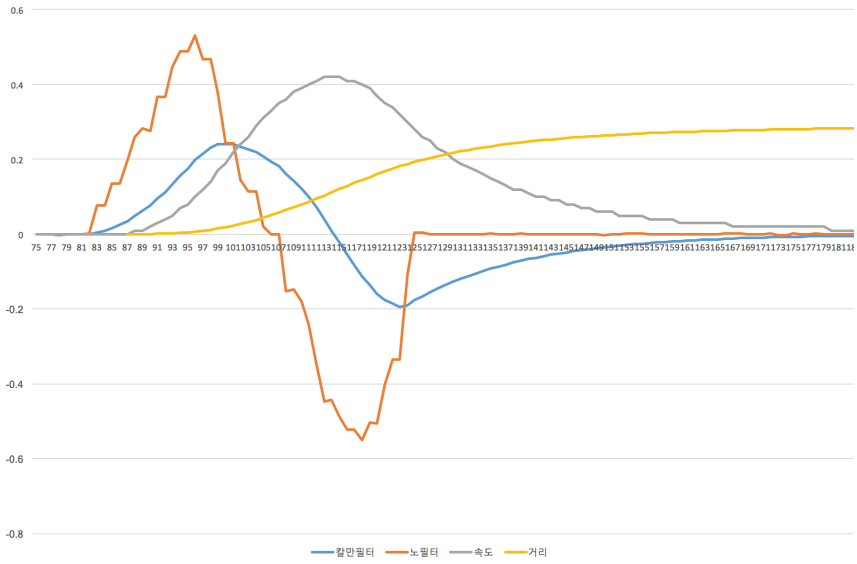




```objc
#import "KalmanFilter.h"

@implementation KalmanFilter


-(id) init: (float)aa: (float)bb: (float)cc: (float)dd: (float)ee{
    //시스템 오차
    q = aa;
    //센서 오차
    r = bb;
    //속도
    v = cc;
    //초기 공분산 오차
    p = dd;
    //칼만 이득
    k = ee;



    return self;
}

-(void) update{
    //칼만 이득의 업데이트
    k = (p + q)/(p + q + r);
    //공분산의 업데이트
    p = r*(p+q)/(r+p+q);
}

//업데이트된 변수들을 이용한 예측
-(float) calculate:(float)z{
    [self update];
    v = v + (z - v)*k;

    return v;
}

@end
```

칼만 필터의 코드

디바이스가 정지 상태에서는 속력을 0으로하고 디바이스가 작동 시 가속도 값에 반응하는 적당한 q를 찾으려고 시도했다. 그러나 문제는 다음과 같다. 센서의 노이즈가 상당하다. 작은 소리, 책상의 진동에도 반응한다. 따라서 상당히 작은 시스템 오차 값이나 큰 센서 오차 값을 주어야 하는데, 큰 시스템 오차 값을 부여하면 반응 시간에 문제가 생긴다. 마우스라는 입력 장치의 특성상 즉각적인 반응이 필수적이기에 반응 시간이 늦춰지는 문제는 심각하다.



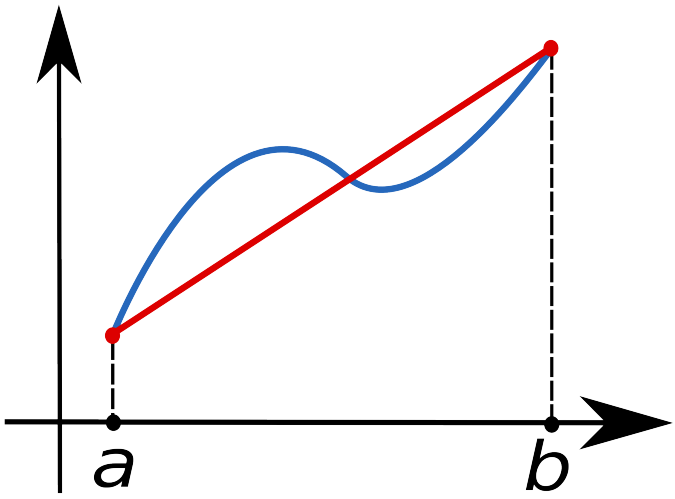
〔 칼만 필터의 적용 〕

II. 적분 오차의 제거

적분 오차의 문제는 중요하다. 수학적으로는 문제가 없지만, 실제로 이산적으로 입력되는 가속도를 적분하는 것은 오차를 야기한다. 가속도의 적분에서 오차가 발생하면 실제로 정지상태에서의 속력 값이 0 으로 수렴하지 않아 마우스의 좌표가 계속 움직이게 된다.

i. trapezoidal method

사다리꼴의 적분 방식으로 이산적으로 입력되는 가속도의 입력값을 적분하는데 사용한다. 원리와 구현이 그 어떤 적분법보다 간단하다. 그러나 입력되는 값들의 차이가 클 수록 오차가 발생한다. 따라서,Mechanical filtering window 를 사용할 경우에 적용하는 것이 적절하다.



(원리)

$$\int_a^b f(x)\,dx \approx (b-a)\frac{f(a)+f(b)}{2}.$$

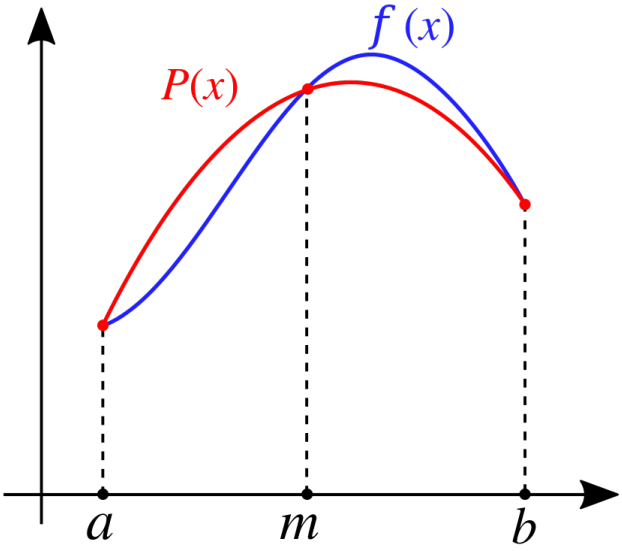
(수식)

```
//trapezoidal 적분
- (float) CalIntegration:(float) base:(float) diff0 : (float) diff1 :(float) time{
    float ret = 0;
    ret = base + ( diff0 + diff1 )/2 *time;
    return ret;
}
```

(코드)

ii. Simpson method

칼만 필터를 사용할 경우 입력되는 가속도 값은 2 차이상의 형태를 보이기 때문에 Simpson 알고리즘을 사용하는 것이 적절한다.



심프슨 공식은 $P(x)$라는 이차방정식을 이용해 $f(x)$의 근사값을 구한다. 이때 $P(x)$는 a, b, 그리고 둘의 중간값 $m = \frac{a+b}{2}$에서 $f(x)$와 같은 값을 갖는 근사식이다. 라그랑주의 다항식 보간법을 사용해서 $P(x)$를 구하면 다음을 얻는다.

$$\int_a^b f(x)\,dx \approx \int_a^b P(x)\,dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

(Wiki 백과)

```
- (float) simpson:(float)t0 :(float)t1 :(float)t2{
    float ret = 0;
    ret  = (2*times/6)*(t0 + 4*t1 + t2);
    return ret;
}
```

III. 결론

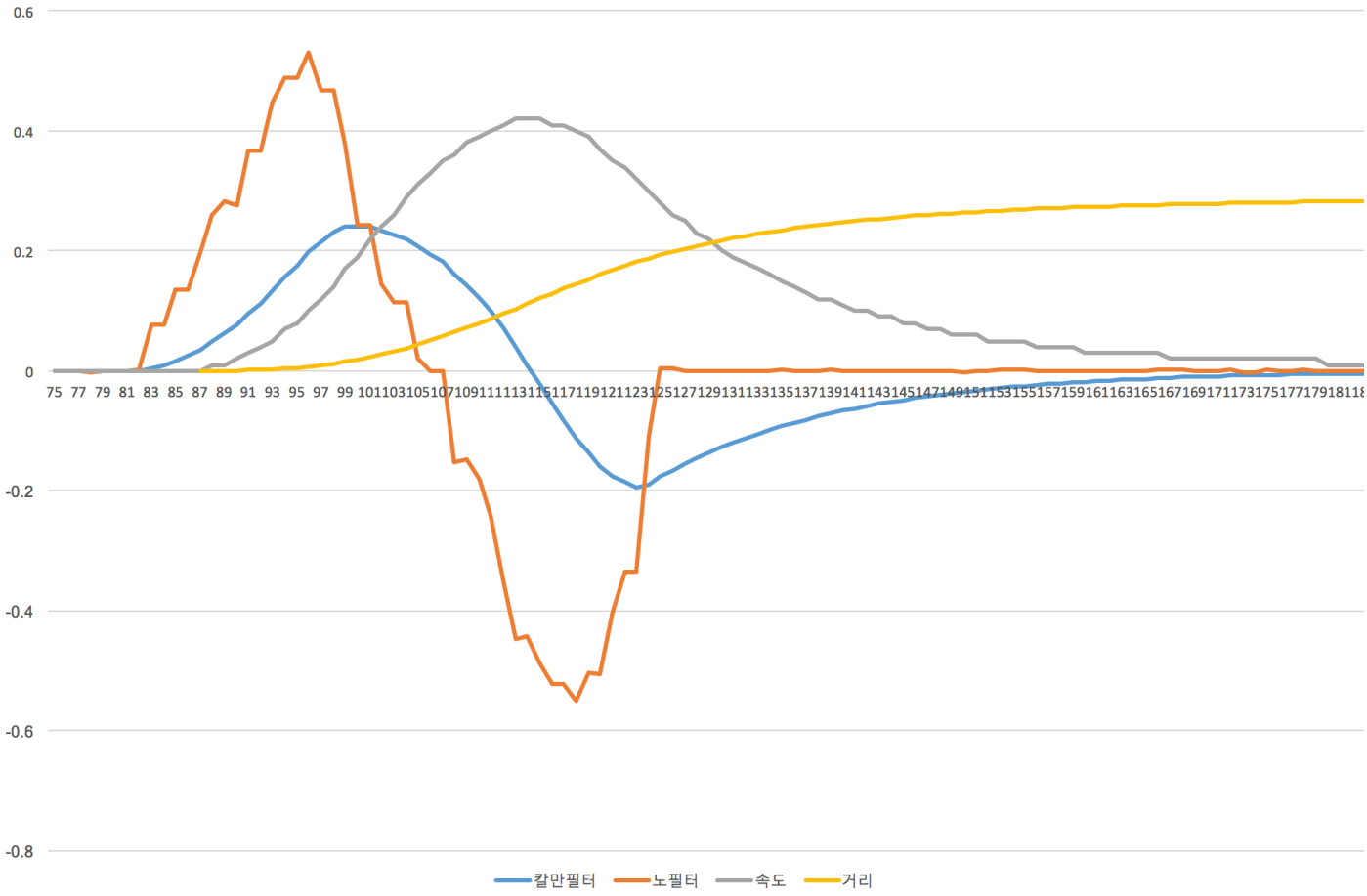
i. 등속 운동 상태의 인식 문제

등속 운동을 하는 물체와 정지 상태의 있는 물체의 공통점은 가속도가 모두 0 이라는 것이다. 적분 오차의 문제점을 제거하기 위해 일정 횟수 이상 가속도 값이 0 으로 입력되면 속도를 0 으로 강제했다.

그러나 실제로 마우스를 사용하다 보면, 드래그, 선 긋기 작업을 할 때 등속 운동을 하게 된다. 가속도계는 100Hz 간격으로 값을 읽어 들인다. 따라서 등속 운동이 일정 시간 지속될 경우 속도가 0 으로 강제되어 등속 운동을 멈추고 감속 구간에서 역방향 가속도가 더욱 크게 작용한다. 이것은 마우스 포인터가 실제 움직인 방향의 역방으로 움직이게 한다. 이 문제는 실제 사용에 큰 장애 요인이 된다.

ii. 반응 시간의 문제

칼만 필터를 이용할 경우 노이즈에 대한 교정이 뛰어나고 Simpson's Rule 을 적용할 수 있기 때문에 적분 오차에도 비교적 안정적이다. 그러나 칼만 필터의 가장 큰 문제점은 반응 시간이다. 일정 시간 이상의 입력 값에 반응하는 특성에 따라 실제 움직임에 대한 반응시간이 0.6 초 가량 늦어지게 되어 실제 사용하기에 부적합하다.



3. 자이로 센서를 이용한 구현 및 결과

가속도 센서를 이용한 구현이 상기된 원인에 의해 실패함에 따라, 자이로센서로 마우스 구현을 시도했다. 이 구현은 가속도계와 달리 입력값에 대한 별도의 처리 과정이 필요 없기 때문에, 단순히 기울기 변화량이 입력된 값을 증폭하여 사용한다.

실제의 사용에도 매우 적합하게 반응한다.

```
if([manager isGyroAvailable]){
    [manager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue] withHandler:^(CMDeviceMotion*
        motion, NSError* error){
        CMRotationRate rot = motion.rotationRate;


        //자이로 센서의 x축 기준 회전이 화면상의 y값을 변화
        x -= sensitivity*rot.z;

        //자이로 센서의 z축 기준 회전이 화면상의 x값을 변화
        y -= sensitivity*rot.x;



        NSLog(@" 현재의 마우스 좌표 ( %+3.5f, %+3.5f )", x, y);
        NSString* msg = [NSString stringWithFormat:@"x:%f y:%f",x ,y];
        //마우스의 x좌표와 y좌표를 전송
        [self sendData:msg];
    }];
}
```

자이로 센서를 통해 입력 값을 가공하여 마우스의 좌표값을 계산한 후, 블루투스 통신으로 값을 전달한다.

마우스의 클릭과 같은 버튼에 대한 동작은 터치 버튼으로 구성하여, 이벤트가 발생할 때마다 블루투스 통신으로 어떤 버튼에 어떠한 이벤트가 발생했는지를 전송한다.

```objc
#import <UIKit/UIKit.h>
#import <CoreMotion/CoreMotion.h>
#import <CoreBluetooth/CoreBluetooth.h>


#define TRANSFER_SERVICE_UUID           @"1A2B"
#define TRANSFER_CHARACTERISTIC_UUID    @"3C4D"


@interface ViewController : UIViewController<CBPeripheralDelegate>{
    //왼쪽 더블탭 트리플탭 변수
    UITapGestureRecognizer *doubleLeftTapGestureRecognizer;
    UITapGestureRecognizer *tripleLeftTapGestureRecognizer;

    //오른쪽 더플탭 트리플탭 변수
    UITapGestureRecognizer *doubleRightTapGestureRecognizer;
    UITapGestureRecognizer *tripleRightTapGestureRecognizer;

    //자이로 모션 관리자
    CMMotionManager *manager;

    //자이로 센서 변수
    float x;
    float y;

    //민감도 설정
    float sensitivity;

    //블루투스 uuid
    CBUUID *uuid;
}


@property (strong, nonatomic) CBPeripheralManager *peripheralManager;
@property (strong, nonatomic) CBMutableCharacteristic *transferCharacteristic;
@property (strong, nonatomic) NSData *dataToSend;
@property (nonatomic, readwrite) NSInteger sendDataIndex;

@property (strong, nonatomic) IBOutlet UIButton *leftButton;
@property (strong, nonatomic) IBOutlet UIButton *rightButton;


- (IBAction)leftButtonClick:(id)sender;  //touch up inside
- (IBAction)leftButtonDrag:(id)sender;   //touch down

- (IBAction)rightButtonClick:(id)sender; //touch up inside
- (IBAction)rightButtonDrag:(id)sender;  //touch down

- (void)peripheralManagerDidUpdateState:(CBPeripheralManager *)peripheral;
- (void)peripheralManager:(CBPeripheralManager *)peripheral central:(CBCentral *)central
    didSubscribeToCharacteristic:(CBCharacteristic *)characteristic;
- (void)sendData:(NSString *)message;
- (void)startAdvertising:(NSDictionary<NSString *,id> *)advertisementData;



@end
```

```objc
#import "ViewController.h"
#import <CoreBluetooth/CoreBluetooth.h>
@interface ViewController ()

@end

@implementation ViewController
@synthesize leftButton, rightButton;

- (void)viewDidLoad {

    [super viewDidLoad];
    //자이로 센서 객체 생성 및 주기설정
    manager = [[CMMotionManager alloc]init];
    manager.deviceMotionUpdateInterval = 0.1; //10Hz
    sensitivity = 10;

    NSLog(@"확인");
    //블루투스 관련 인스턴스 생성
    _peripheralManager = [[CBPeripheralManager alloc]
        initWithDelegate:self queue:nil];
    CBUUID *uuid = [CBUUID UUIDWithString:TRANSFER_SERVICE_UUID];
    NSArray *uuids = @[uuid];
    NSDictionary *info = @{CBAdvertisementDataServiceDataKey: uuids,
                           CBAdvertisementDataLocalNameKey:@"nil"};
    NSLog(@"블루투스 게시 시작");
    [_peripheralManager startAdvertising:info];


    x,y= 0;


    if([manager isGyroAvailable]){
        [manager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue] withHandler:^(CMDeviceMotion*
            motion, NSError* error){
            CMRotationRate rot = motion.rotationRate;


            //자이로 센서의 x축 기준 회전이 화면상의 y값을 변화
            x -= sensitivity*rot.z;

            //자이로 센서의 z축 기준 회전이 화면상의 x값을 변화
            y -= sensitivity*rot.x;



            NSLog(@" 현재의 마우스 좌표 ( %+3.5f, %+3.5f )", x, y);
            NSString* msg = [NSString stringWithFormat:@"x:%f y:%f",x ,y];
            //마우스의 x좌표와 y좌표를 전송
            [self sendData:msg];
        }];
    }
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

```
-(void)sendData:(NSString *)message{
    NSData *data = [message dataUsingEncoding:NSUTF8StringEncoding];
    BOOL didSend = [_peripheralManager updateValue:data forCharacteristic:_transferCharacteristic
        onSubscribedCentrals:nil];
    if(didSend){
        NSLog(@"Sent : %@",message);
    }
}



- (IBAction)leftButtonClick:(id)sender {

    //왼쪽 더블 클릭 인식을 위한 GestureRecognizer 생성 - handleLeftDoubleTapGesture에서 동작 내용 구현
    doubleLeftTapGestureRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector
        (handleLeftDoubleTapGesture:)];
    //왼쪽 트리플 클릭 인식을 위한 GestureRecognizer 생성 - handleLeftTripleleTapGesture에서 동작 내용 구현
    tripleLeftTapGestureRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector
        (handleLeftTripleTapGesture:)];

    //왼쪽 더블 클릭 인식을 위한 GestureRecognizer 설정 : 횟수 2회
    doubleLeftTapGestureRecognizer.numberOfTapsRequired = 2;
    //왼쪽 트리플 클릭 인식을 위한 GestureRecognizer 설정 : 횟수 3회
    tripleLeftTapGestureRecognizer.numberOfTapsRequired = 3;

    //더블 클릭은 트리플 클릭이 인식될 시에 실패로 규정함
    [doubleLeftTapGestureRecognizer requireGestureRecognizerToFail:tripleLeftTapGestureRecognizer];

    //현재 뷰(UIButton)에 생성한 제스쳐 인식자들을 추가함
    [self.view addGestureRecognizer:doubleLeftTapGestureRecognizer];
    [self.view addGestureRecognizer:tripleLeftTapGestureRecognizer];


    NSLog(@"왼쪽 - 싱글 탭이 인식되었습니다.");
    NSString* msg = [NSString stringWithFormat:@"LS"];
    //동작에 의한 신호를 전송
    [self sendData:msg];
}

- (IBAction)leftButtonDrag:(id)sender {
    NSLog(@"왼쪽 - 드래그 탭이 인식되었습니다.");
    NSString* msg = [NSString stringWithFormat:@"LG"];
    //동작에 의한 신호를 전송
    [self sendData:msg];

}
```

```
- (IBAction)rightButtonClick:(id)sender {
    //오른쪽 더블 클릭 인식을 위한 GestureRecognizer 생성 - handleRightDoubleTapGesture에서 동작 내용 구현
    doubleRightTapGestureRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector
        (handleRightDoubleTapGesture:)];
    //오른쪽 트리플 클릭 인식을 위한 GestureRecognizer 생성 - handleRightTripleTapGesture에서 동작 내용 구현
    tripleRightTapGestureRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector
        (handleRightTripleTapGesture:)];

    //오른쪽 더블 클릭 인식을 위한 GestureRecognizer 설정 : 횟수 2회
    doubleRightTapGestureRecognizer.numberOfTapsRequired = 2;
    //오른쪽 더블 클릭 인식을 위한 GestureRecognizer 설정 : 횟수 3회
    tripleRightTapGestureRecognizer.numberOfTapsRequired = 3;

    //더블 클릭은 트리플 클릭이 인식될 시에 실패로 규정함
    [doubleRightTapGestureRecognizer requireGestureRecognizerToFail:tripleRightTapGestureRecognizer];

    //현재 뷰(UIButton)에 생성한 제스쳐 인식자들을 추가함
    [self.view addGestureRecognizer:doubleRightTapGestureRecognizer];
    [self.view addGestureRecognizer:tripleRightTapGestureRecognizer];


    NSLog(@"오른쪽 - 싱글 탭이 인식되었습니다.");
    NSString* msg = [NSString stringWithFormat:@"RS"];
    //동작에 의한 신호를 전송
    [self sendData:msg];
}

- (IBAction)rightButtonDrag:(id)sender {
    NSLog(@"오른쪽 - 드래그 탭이 인식되었습니다.");
    NSString* msg = [NSString stringWithFormat:@"RG"];
    //동작에 의한 신호를 전송
    [self sendData:msg];

}
```

```objc
//왼쪽 더블 클릭 : doubleLeftTapGestureRecognizer가 인식될 시 작동 방식의 구현
- (void)handleLeftDoubleTapGesture:(UITapGestureRecognizer *)sender{
    if(doubleLeftTapGestureRecognizer.state == UIGestureRecognizerStateEnded){
        NSLog(@"왼쪽 - 더블 탭이 인식되었습니다.");
        NSString* msg = [NSString stringWithFormat:@"LD"];
        [self sendData:msg];
    }
}

//왼쪽 트리플 클릭 : handleLeftTripleTapGesture가 인식될 시 작동 방식의 구현
- (void)handleLeftTripleTapGesture:(UITapGestureRecognizer *)sender{
    if(tripleLeftTapGestureRecognizer.state == UIGestureRecognizerStateEnded){
        NSLog(@"왼쪽 - 트리플 탭이 인식되었습니다.");
        NSString* msg = [NSString stringWithFormat:@"LT"];
        [self sendData:msg];
    }
}


//오른쪽 더블 클릭 : handleRightDoubleTapGesture가 인식될 시 작동 방식의 구현
- (void)handleRightDoubleTapGesture:(UITapGestureRecognizer *)sender{
    if(doubleRightTapGestureRecognizer.state == UIGestureRecognizerStateEnded){
        NSLog(@"오른쪽 - 더블 탭이 인식되었습니다.");
        NSString* msg = [NSString stringWithFormat:@"RD"];
        [self sendData:msg];
    }
}

//오른쪽 트리플 클릭 : handleRightTripleTapGesture가 인식될 시 작동 방식의 구현
-(void)handleRightTripleTapGesture:(UITapGestureRecognizer *)sender{
    if(tripleRightTapGestureRecognizer.state == UIGestureRecognizerStateEnded){
        NSLog(@"오른쪽 - 트리플 탭이 인식되었습니다.");
        NSString* msg = [NSString stringWithFormat:@"RT"];
        [self sendData:msg];
    }
}


- (void)peripheralManagerDidUpdateState:(CBPeripheralManager *)peripheral {

    CBManagerState state = peripheral.state;

    if(state != CBManagerStatePoweredOn) {
        NSLog(@"peripheralManagerDidUpdateState error = %ld",state);
    }
    else{
        NSLog(@"bluetooth on ");
        CBUUID *uuidService = [CBUUID UUIDWithString:TRANSFER_SERVICE_UUID];
        CBUUID *uuidCharacteristic = [CBUUID UUIDWithString:TRANSFER_CHARACTERISTIC_UUID];


        CBCharacteristicProperties porperties = CBCharacteristicPropertyWrite | CBCharacteristicPropertyRead
            CBCharacteristicPropertyNotify;
        CBAttributePermissions permissions = CBAttributePermissionsReadable | CBAttributePermissionsWriteable

        self.transferCharacteristic = [[CBMutableCharacteristic alloc] initWithType:uuidCharacteristic
            properties:porperties value:nil permissions:permissions];

        CBMutableService *transferService = [[CBMutableService alloc] initWithType:uuidService primary:YES];
        transferService.characteristics = @[_transferCharacteristic];

        [_peripheralManager addService:transferService];
    }
}

- (void)peripheralManager:(CBPeripheralManager *)peripheral central:(CBCentral *)central
    didSubscribeToCharacteristic:(CBCharacteristic *)characteristic {

    //NSString *init = @"start talking";

    NSString *hello = [NSString stringWithFormat:@"connected"];

    [self sendData:hello];
    NSLog(@"start talking check:");
}




@end
```