(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년10월08일
(11) 등록번호 10-1447582
(24) 등록일자 2014년09월29일

(51) 국제특허분류(Int. Cl.)
G01S 19/40 (2010.01) *G01S 19/39* (2010.01)
(21) 출원번호 10-2013-0010032
(22) 출원일자 2013년01월29일
심사청구일자 2013년01월29일
(65) 공개번호 10-2014-0096883
(43) 공개일자 2014년08월06일
(56) 선행기술조사문헌
JP2008175788 A*
JP2010223692 A
KR1020030037891 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
홍익대학교 산학협력단
서울특별시 마포구 와우산로 94 (상수동)
(72) 발명자
송하윤
서울특별시 마포구 와우산로 94, 컴퓨터공학과 (상수동, 홍익대학교)
(74) 대리인
양기혁, 이인행, 김남식, 한윤호

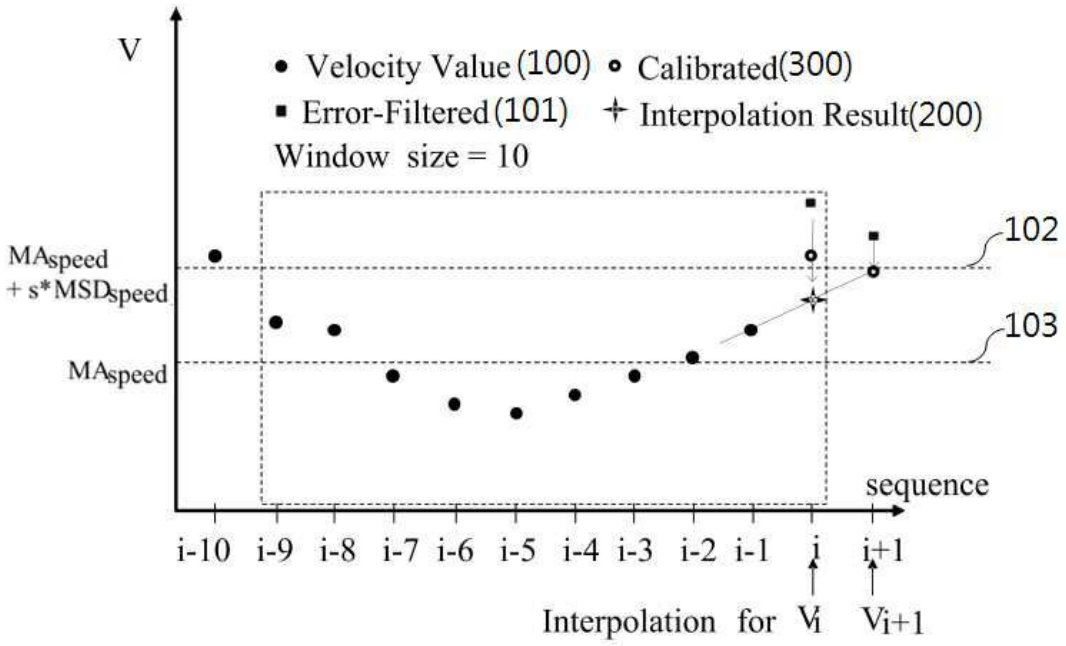
전체 청구항 수 : 총 8 항 심사관 : 변영석

(54) 발명의 명칭 **위치정보 보정방법**

(57) 요 약

위치정보 수집장치로부터 수집된 오류가 있는 위치의 참된 값을 추정하는 방법으로서, 상기 오류가 있는 위치의 전후에 수집된 위치 값들을 인터폴레이션하여 상기 참된 값을 추정(estimation)하되, 상기 오류가 있는 위치의 바로 다음에 수집된 위치 값에도 오류가 존재하는 경우에는 상기 바로 다음에 수집된 위치 값을 미리 교정(calibration)한 이후에 추정하는 방법이 공개된다.

대 표 도 - 도5



특허청구의 범위

청구항 1

위치정보 수집장치로부터 수집한 위도, 경도, 및 타임스탬프를 포함하여 이루어지는 튜플에 관한 정보를 처리하는 방법으로서,

타임스탬프[i+1]에서의 가속도[i+1](a_{i+1})에 오류가 있는 것으로 판단된 경우; 상기 타임스탬프[i+1](t_{i+1})에서 획득한 위도[i+1](lat_{i+1})를, 상기 타임스탬프[i]에서의 위도[i](lat_i)에 상기 타임스탬프[i] 이전의 한 개 이상의 위도값들의 대표값을 적용하여 생성한 보정위도[i+1]로 대체하고; 상기 타임스탬프[i+1]에서 획득한 경도[i+1](lon_{i+1})를, 상기 타임스탬프[i]에서의 경도[i](lon_i)에 상기 타임스탬프[i] 이전의 한 개 이상의 경도값들을 대표값을 적용하여 생성한 보정경도[i+1]로 대체하는 단계; 및

타임스탬프[i](t_i)에서의 튜플[i](P_i)에 오류가 존재하는 것으로 확인된 경우에는, 상기 보정위도[i+1]와 상기 타임스탬프[i] 이전에 획득한 한 개 이상의 위도들을 인터폴레이션하여 상기 타임스탬프[i]에서의 위도[i](lat_i)를 추정(estimation)하고, 상기 보정경도[i+1]와 상기 타임스탬프[i] 이전에 획득한 한 개 이상의 경도들을 인터폴레이션하여 상기 타임스탬프[i]에서의 경도[i](lon_i)를 추정하는 단계;

를 포함하는,

위치정보 처리방법.

청구항 2

삭제

청구항 3

제1항에 있어서, 상기 가속도[i+1]가 미리 설정된 최대가속도($MAX_{acceleration}$) 이상인 경우에는 상기 가속도[i+1]에 오류가 있는 것으로 결정하는, 위치정보 처리방법.

청구항 4

제1항에 있어서, 상기 타임스탬프[i]까지의 평균경도($MA_{Vlongitude}$) 및 상기 타임스탬프[i]에서의 경도[i](lon_i)를 이용하여 상기 경도[i+1]를 교정하고, 상기 타임스탬프[i]까지의 평균위도($MA_{Vlatitude}$) 및 상기 타임스탬프[i]에서의 위도[i](lat_i)를 이용하여 상기 위도[i+1]를 교정하도록 되어 있는, 위치정보 처리방법.

청구항 5

제1항에 있어서,

상기 추정하는 단계는, 상기 오류가 존재하는 것으로 확인된 경우에는, 타임스탬프[i+1]에서의 속도[i+1](V_{i+1})와 상기 타임스탬프[i] 이전에 획득한 한 개 이상의 속도들을 인터폴레이션하여 상기 타임스탬프[i]에서의 속도[i](V_i)를 추정(estimation)하는 단계를 포함하는, 위치정보 처리방법.

청구항 6

제1항에 있어서,

상기 추정하는 단계 이전에, 상기 타임스탬프[i+1]에서의 속도[i+1](V_{i+1})에 오류가 있는지 여부를 결정하는 단계를 더 포함하며,

상기 속도[i+1]에 오류가 있는 것으로 판단된 경우, 상기 추정하는 단계 이전에 상기 속도[i+1]를 미리 교정(calibration) 또는 제한(restriction)하도록 되어 있는,

위치정보 처리방법.

청구항 7

제6항에 있어서,

상기 속도[i+1]가 미리 결정된 최저속도(MIN_{speed}) 이하인 경우에는 상기 속도[i+1]에 오류가 없는 것으로 판단하며,

상기 최저속도는 허용오차거리(error tolerance of distance, ET_D)에 의해 결정되는,

위치정보 처리방법.

청구항 8

위도, 경도, 및 타임스탬프를 포함하여 이루어지는 튜플에 관한 정보를 수집하도록 되어있는 위치정보 수집부; 및

상기 튜플에 관한 정보를 처리하도록 되어 있는 처리부

를 포함하며,

상기 처리부는,

타임스탬프[i+1]에서의 가속도[i+1](a_{i+1})에 오류가 있는 것으로 판단된 경우; 상기 타임스탬프[i+1](t_{i+1})에서 획득한 위도[i+1](lat_{i+1})를, 상기 타임스탬프[i]에서의 위도[i](lat_i)에 상기 타임스탬프[i] 이전의 한 개 이상의 위도값들의 대표값을 적용하여 생성한 보정위도[i+1]로 대체하고; 상기 타임스탬프[i+1]에서 획득한 경도[i+1](lon_{i+1})를, 상기 타임스탬프[i]에서의 경도[i](lon_i)에 상기 타임스탬프[i] 이전의 한 개 이상의 경도값들을 대표값을 적용하여 생성한 보정경도[i+1]로 대체하도록 되어 있고, 그리고

타임스탬프[i](t_i)에서의 튜플[i](P_i)에 오류가 존재하는 것으로 확인된 경우에는, 상기 보정위도[i+1]와 상기 타임스탬프[i] 이전에 획득한 한 개 이상의 위도들을 인터폴레이션하여 상기 타임스탬프[i]에서의 위도[i](lat_i)를 추정(estimation)하고, 상기 보정경도[i+1]와 상기 타임스탬프[i] 이전에 획득한 한 개 이상의 경도들을 인터폴레이션하여 상기 타임스탬프[i]에서의 경도[i](lon_i)를 추정하도록 되어 있는,

사용자 기기.

청구항 9

제8항에 있어서,

상기 처리부는,

상기 추정을 수행하기 이전에, 상기 타임스탬프[i+1]에서의 속도[i+1](V_{i+1})에 오류가 있는지 여부를 결정하도록 되어 있으며,

상기 속도[i+1]에 오류가 있는 것으로 판단된 경우, 상기 추정을 수행하기 이전에 상기 속도[i+1]를 미리 교정(calibration)또는 제한(restriction)하도록 되어 있는,

사용자 기기.

명 세 서

기 술 분 야

[0001] 본 발명은 사용자 기기에 의해 수집된 위치데이터에 오류가 있는지 여부를 판단하는 방법 및 오류가 존재하는 위치값을 보정하는 기술에 관한 것이다.

배 경 기 술

[0002] 사람들은 그들이 빈번하게 방문한 장소와 경로를 선택하는 것 대신에 다음 목적지를 임의로 선택하는 것을 꺼리는 경향이 있으며, 따라서 인간의 이동 목적지가 상당한 수준으로 예측될 수 있다는 다양한 연구결과가 존재한다. 어떤 사람의 이동을 예측하기 위해서는 우선 그 사람에 대한 정확한 위치데이터를 얻을 수 있어야 한다. 이를 위하여, GPS, GLONASS 및 Galileo와 같은 잘 알려진 위치데이터 시스템을 이용하거나, 이동통신 기지국 또는 크라우드 소스(crowd source)로서의 WIFI 위치 접근법을 이용한 무선 통신망 포지셔닝을 이용하는 등의 다양한 방법을 이용하여 위치데이터를 알아낼 수 있다. 이러한 방법들을 구현하기 위한 기술은 최근 기술의 발달로 이동형 사용자 기기에 제공되고 있다. 그러나 실제로 측정된 실측 위치데이터에는 여러가지 이유로 에러가 존재할 수 있다. 애플사의 아이폰이나 삼성의 갤럭시를 이용할 때에 꽤 많은 에러가 있는 위치데이터를 경험할 수 있다. 명백한 에러들을 필터링할 필요성이 있고, 나아가 에러라고 판단된 위치를 보정하여 제공할 필요가 있다.

발명의 내용

해결하려는 과제

[0003] 사용자 기기에 의해 실측된 위치데이터는 작동환경에 따른 에러들을 포함할 수 있다. 실제로, 본 발명의 검증을 위하여, 상용의 스마트폰, 휴대용 GPS 장치, 크라우드 소싱(crowd sourcing)을 위한 WIFI 위치검출장치, 및 셀룰러 네트워크 장치 등으로부터 수집한 실측 위치데이터의 12% 이상에 에러가 나타났다. 따라서 에러가 있는 위치데이터를 필터링 아웃할 필요가 있다. 나아가 단순히 에러가 있는 위치데이터를 버리는 것이 아니라 이를 보정한 새로운 위치데이터로 대체할 필요가 있다. 본 발명에서는 이러한 문제점을 해결하기 위한 방법을 제공하고자 한다. 또한 이러한 문제점을 해결하기 위한 방법이 컴퓨팅 능력이 제한된 사용자 기기에서 잘 수행될 수 있도록 실시간으로 구현하는 방법을 제공하고자 한다.

과제의 해결 수단

[0004] 사용자 기기에 의해 실측된 위치데이터에 오류가 있는지 여부를 판단하기 위한 본 발명의 일 양상에 따른 방법은 실측된 속도에 대하여 계산된 평균값(mean, expectation) 및 표준편차(standard deviation)를 이용하는 통계적 기법을 이용할 수 있다. 또한 실측된 가속도의 값을 이용하는 기법을 이용할 수 있다.

[0005] 그리고 오류가 있다고 판단된 위치값을 보정하기 위한 본 발명의 일 양상에 따른 방법은 실측된 위치값에 대하여 계산된 차이(difference)를 이용하는 통계적 기법을 이용할 수 있다.

[0006] 본 명세서에서 '실측'이라 함은, 본 발명에 따른 위치데이터 필터링방법 및 위치데이터 보정방법을 이용하지 않은 사용자 기기로부터 위치, 속도, 가속도에 관한 정보를 직접 획득하는 것을 의미할 수 있다.

[0007] 사용자 기기로부터 실측된 <위도, 경도, 시간>의 형식을 갖는 이동 위치데이터로부터 사용자의 속도 및 가속도의 실측값을 계산할 수 있다. 슬라이딩 윈도우 또는 이동윈도우에 의하여 선택된 데이터를 이용하여 위치 및 속도의 통계량을 계산하고 이 계산값들을 반복적으로 갱신할 수 있다. 그리고 실측된 위치데이터에 대한 필터링을 제어가능한 파라미터를 이용하여 수행할 수 있다. 본 발명의 일 실시예에 따른 알고리듬은 간단하기 때문에 계산능력(computing power)이 작은 이동형 사용자 기기에도 적용될 수 있다.

[0008] 본 발명의 일 실시예에서는 성능의 향상을 위하여 이동윈도우의 크기를 최적화할 수 있다. 또한 에러가 있는 것으로 결정된 속도값 및/또는 위치값을 적당한 추정값으로 교체하기 위해 백트래킹(backtracking) 인터폴레이션 방법이 사용될 수 있다.

[0009] 속도는 일정 수준 이상으로 급격하게 증가하지 않는다고 볼 수 있기 때문에, 일정 시구간에서의 속도의 평균값과 표준편차를 알 수 있다면, 예컨대 정규분포를 따르는 유의구간(significant interval)을 설정하고 이 유의구간을 넘어가는 속도는 오류값이라고 판단할 수 있다.

발명의 효과

[0010] 본 발명에 따르면 에러가 있는 위치데이터를 실시간으로 필터링 아웃할 수 있으며, 에러가 있다고 판단된 위치데이터를 버리는 것이 아니라 보정한 새로운 위치데이터를 이용하여 실시간으로 대체하는 기술을 제공할 수 있다.

도면의 간단한 설명

[0011] 도 1은 본 발명의 일 실시예에 따른 속도교정 알고리듬을 설명하기 위한 것이다.

도 2는 본 발명의 다른 실시예에 따라, 최소 자승법으로 속도값을 인터폴레이션하는 속도교정방법을 설명하기 위한 것이다.

도 3 내지 도 5는 본 발명의 일 실시예에 따른 위치튜플 필터링 방법, 인터폴레이션을 이용한 속도 추정방법, 및 속도에러 교정방법을 설명하기 위한 것이다.

도 6은 본 발명의 일 실시예에 따른 속도보정없이 검출한 속도에러를 설명하기 위한 것이다.

도 7는 본 발명의 일 실시예에 따른 인터폴레이션 기법에 따른 속도교정을 적용하여 검출한 속도에러를 설명하기 위한 것이다.

발명을 실시하기 위한 구체적인 내용

[0012] 이하, 첨부된 도면들을 참조하여 본 발명의 실시예를 상세히 설명한다. 그러나 본 발명은 이하에서 개시되는 실시예에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있다.

[0013] <위도, 경도, 시간>의 형식(이하, '튜플(tuple)' 또는 '위치데이터 튜플'이라고 지칭할 수 있음)으로 실측되어 수집된 한 세트의 사용자 위치를 이용하여 사용자의 이동궤적(자취)을 나타낼 수 있다. 이 튜플(tuple)에 각 튜플을 식별할 수 있는 식별 파라미터를 추가함으로써 사용자 이동성을 나타내는 데이터 세트를 얻을 수 있다. 본 명세서에서 시각 t에 대한 하나의 튜플을 P_t 라고 지칭하고, P_t 에 의해 제공되는 위도 및 경도를 각각 lat_t , lon_t 라고 지칭할 수 있다.

[0014] 두 개의 연속적인 위치데이터 튜플들로부터, Vincenty의 공식[1]에 따라, $<lat_{i-1}, lon_{i-1}>$와 $<lat_i, lon_i>$를 이용하여 시각 P_i 에서의 이동거리 D_i 를 계산할 수 있다. 물론, 이 이동거리로부터 시각 t_i 에서의 속도 V_i 를 계산할 수 있고, 연속된 3개의 튜플에서 연속된 2개의 속도를 계산하여 P_i 가 가진 시각 t_i 에서의 가속도 a_i를 계산할 수 있다. 그러므로 튜플 P_t 는 부가적인 속성을 포함하는 $<t, lat_t, lon_t, D_t, V_t, a_t>$와 같은 코어 형식(core form)을 가질 수 있다. 상술한 t, lat_t, lon_t, D_t, V_t, a_t 은 본 명세서에서 실측값이라고 지칭될 수 있다.

[0015] [1]: T. Vincenty, "Direct and Inverse Solutions of Geodesics on the ellipsoid with Application of Nested Equations," Survey Review, Volume 23, Number 176, April 1975 , pp. 88-93(6).

[0016] 실측된 위치데이터 세트에 의해 실측된 속도 값에는, 예컨대 600m/s라는 값이 존재하였는데 이러한 값은 보통의 일상환경에서 무의미한 값으로서 오류가 발생한 것으로 판단할 수 있다. 표 1에는 운송수단에 따라 나타날 수 있는 일반적인 최대속도를 조사하여 나타냈다. 예컨대 본 발명의 일 실시예에 적용하기 위한 최대속도 MAX_{speed} 을 250m/s로 정의할 수 있다. 또한 일상환경에서 가능하다고 여겨지는 가속도의 최대값을 정의하여 $MAX_{acceleration}$ 라고 표기할 수 있다.

표 1

[0017]

Transportation Method	Maximum speed (m/sec)
Ambulation	3.00
Bicycle	33.33
Automobile	92.78
Sports-Car	244.44
High-Speed Train	159.67
Air-plain	528.00

[0018] 그 다음, 실측되거나 저장되어 있는 속도로부터 이동평균(moving average of speed) 및 이동표준편차(moving standard deviation of speed)를 계산할 수 있다.

[0019] 임의의 시각 t에서 속도의 이동평균 값을 $MA_{speed}(n)$라고 표기할 수 있는데, 여기서 n은 튜플의 집합인 $\{P_x : t-n+1 \leq x < t\}$의 과거 데이터의 수를 나타낸다. 마찬가지로, 상기 시각 t에서의 이동표준편차 값을 $MSD_{speed}(n)$라고 표기할 수 있는데, 여기서 n은 과거 데이터의 수를 나타낸다. 이때, 과거의 n개의 데이터는 크기 'n'을 갖는 윈도우를 적용함으로써 얻을 수 있다. 일단 새로운 튜플 P_t 를 얻게되면, 이것으로부터 계산된 V_t 및 a_t 가 일상 환경에서 용인될 수 있는 값인지 여부를 확인할 수 있다. 일단 상기 V_t 가 속도의 이동평균값 $MA_{speed}(n)$과 속도의 이동표준편차 $MSD_{speed}(n)$에 의한 정규분포(normal distribution)의 범위 밖에 존재하는 경우에는 상기 튜플 P_t 가 일련의 자취로부터 필터링 아웃(즉, 제거)될 수 있다. P_t를 제거하기 위한 조건은 수학식 1과 같이 주어질 수 있다.

[0020] [수학식 1]

[0021] $$V_t > MA_{speed}(n) + s * MSD_{speed}(n)$$

[0022] 여기서 s는 필터링의 민감성 수준을 나타내며 사용자가 통제 가능한 파라미터이다. 그렇지 않다면(즉, V_t가 수학식 1을 만족하지 않는다면), 새로운 튜플 P_t 이 유효한 값을 갖고 있다고 판단하고, 이 새로운 값을 이용하여 $MA_{speed}(n)$와 $MSD_{speed}(n)$의 값을 갱신할 수 있다. 이 계산은 비교적 간단하기 때문에 이동형 사용자 기기와 같은 저전력 장치에서 실시간으로 수행될 수 있다.

[0023] 상술한 윈도우의 크기에 따라 필터링 특성이 달라질 수 있다. 따라서 윈도우의 크기에 따른 영향을 파악하기 위한 실험을 수행하였는데, 실험에 사용된 위치데이터는 위치검출을 수행할 수 있는 상용의 이동형 사용자기기에 의해 실측된 값을 사용하였다. 사용자기기는 위치변화를 감지할 때마다 위치데이터를 저장하고 출력할 수 있다. 또는 사용자기기가 정지 상태에 있다면 사용자가 정의한 매 간격, 예컨대 3초에서 60초마다 위치데이터를 저장하고 출력할 수 있다. 이렇게 획득한 위치데이터들을 다양한 기법을 이용하여 지도(map) 상에 표시할 수 있다.

[0024] 위치데이터를 제공한 사용자기기는 3개의 서로 다른 방식에 따라 얻은 위치데이터를 가끔 동시에 제공하였다. 즉, 셀룰라 기지국을 이용한 위치, 크라우드 소스(crowd source)를 위한 WiFi 장치를 이용한 위치, 및 GPS를 이용한 위치가 동시에 제공되었는데, 이때 동일한 시각에 제공된 3가지 종류의 위치데이터의 값이 서로 다른 값을 나타내는 경우가 발생했다. 동시에 다수의 위치값들을 얻게되는 경우, 경험에 따르면 가장 작은 속도값을 가진 위치데이터가 유효한 데이터일 확률이 크다. 그러므로 위치데이터를 필터링하기 위한 첫 단계에서는 같은 시간에 서로 다른 방식에 의해 제공된 다수의 위치데이터 중 이전의 위치에 대해 가장 조금 이동한 위치데이터를 고르는 방식을 선택할 수 있다.

[0025] 위치데이터 필터링을 위하여 사용되는 알고리듬에 적용되는 윈도우의 크기 n 의 최적값을 결정하기 위하여 n 의 크기를 변화시키면서 다양한 실험을 수행하였다. 윈도우 크기가 큰 경우에 상기 알고리듬은, 이동이 없는 상태에서는 연속적인 에러들에 대해 성공적으로 대처할 수 있지만, 속도가 빠르게 변하는 상황에는 반응하지 못할 것이라고 예상할 수 있다. 반면에, 윈도우의 크기가 작은 경우라면, 연속적인 에러 튜플들을 포함하는 것으로 여겨지는 이동 상태에 대한 갑작스런 속도 변화에 대한 빠른 반응을 보여줄 것이다. 그러나 m개의 연속적인 에러가 존재하는 경우에 $m > n$이라면, 즉 윈도우의 크기가 작다면 상기 에러들을 필터링할 수가 없다. 이를 확인하기 위하여, 상기 실측된 한 세트의 데이터세트에 대하여 윈도우 크기를 변경하면서 다양한 실험을 하였다. 각 실험에서 n = 5, 10, 25, 50, 100와 같이 주어졌으며, 각 경우에 대한 이동평균과 이동표준편차를 계산했다. 실측된 속도값이 매우 큰 에러가 있는 데이터가 입력되는 경우, 윈도우 크기가 클수록 큰 속도의 영향을 긴 시간 동안 유지된다. 그 결과 에러가 있는 데이터에 대한 언더 필터링(under filtering) 현상이 발생한다. 윈도우 크기가 100 또는 50인 경우에, 테일링 효과(tailing effect)가 있음을 확인하였다.

[0026] 반면에, 윈도우 크기가 작은 경우에는, 특히 속도변화의 시작단계에서, 속도 변화에 빠르게 반응하고 민감한 반면, 올바른 데이터를 오버 필터링하는 경향이 있음을 알 수 있다. 윈도우 크기가 5 또는 10과 같이 작을 때에, 올바른 속도 데이터인 것으로 보이는 2개 이상의 튜플들이 제거된 것을 알 수 있다. 이 현상을 통해, 윈도우 크기에 따른 테일링 효과(tailing effect)를 피하기 위해서는 이동윈도우에 대한 조절 메카니즘을 도입해야만 한

다는 점을 알 수 있다.

[0027] 이하 실측된 위치데이터의 에러를 파악하기 위한 사전실험(pre-experiments)에 대해 설명한다.

[0028] 이 사전실험에서 위치검출 장치들을 실외와 실내에 고정한 상태에서 수 시간 동안 위치데이터를 수집하였다. 첫 번째 위치검출 장치는 상용의 Garmin GPSMAP62s 으로서 순수하게 GPS 데이터 수집을 위해 사용되었다. 두 번째 위치검출 장치는 상용의 삼성 갤럭시 탭으로서, 이 장치에 연결된 3G 기지국(3GBS)으로부터 위치검출 데이터를 얻기 위해 사용되었다. 3GBS를 이용하여 얻은 위치데이터에 더 많은 에러를 나타낼 것이라고 추측하였으며, GPS 및 3GBS로부터 얻은 위치데이터 모두 에러를 나타낼 것이라고 추측하였다. 특히 실내에서도 위치검출 에러를 보일 것이라고 추측하였다. 이 사전실험의 결과는 표 2에 표시되어 있다. 표 2는 위치데이터 실측 시 발생하는 에러를 나타낸 것이며, 표기 단위는 미터(meter)이다.

표 2

[0029]

Building	*3G Base Station*		*GPS*	
Inside	n(Data Point)	893	n(Data Point)	2186
	n(Error Point)	434	n(Error Point)	939
	Error Rate	48.6%	Error Rate	43.0%
	E[Error Dist]	52.5530m	E[Error Dist]	43.5506m
	Max(Error Dist)	156.7578m	Max(Error Dist)	10769.72m
	$\sigma_{ErrorDist}$	32.6859m	$\sigma_{ErrorDist}$	370.6034m
Outside	n(Data Point)	331	n(Data Point)	1690
	n(Error Point)	122	n(Error Point)	208
	Error Rate	36.9%	Error Rate	12.3%
	E[Error Dist]	52.6618m	E[Error Dist]	4.4498m
	Max(Error Dist)	206.3526m	Max(Error Dist)	51.7789m
	$\sigma_{ErrorDist}$	23.5953m	$\sigma_{ErrorDist}$	7.1696m

[0030] 위치데이터로부터 에러거리가 계산되며, 에러거리의 분산(variance)과 평균을 계산할 수 있다. 추측한 바와 같이, 3GBS가 더 큰 에러율을 나타냈으며, 에러의 크기도 더 컸고, 에러 크기의 최대값도 더 컸으며, 에러 크기의 표준편차도 더 컸다. Garmin GPSMAP62s 에서는 GPS 신호를 잃은 경우에도 과거의 속도에 근거하여 사용자의 위치를 계산하도록 되어 있는데, 이 때문에 실내에서도 매우 큰 에러를 나타냈다. 따라서 실내의 GPS 데이터는 의미있는 데이터가 아닌 것으로 생각할 수 있다. 실외에서의 GPS 데이터는 정교한 위치데이터를 얻기 위하여 충분한 수준의 정확성을 나타내며, 최대 에러 크기도 52 미터라는 합리적인 수준의 범위 내에 들어왔다.

[0031] 이하 본 발명의 일 실시예에 따른 필터링 알고리듬을 설명한다. 표 3의 [알고리듬 1]은 위치데이터에 오류가 있는지 여부를 판단하고, 오류가 있는 위치데이터를 필터링하기 위한 것이다. 새로운 위치데이터 P_{i+1} 를 획득하면, 이 [알고리듬 1]은 P_{i+1} 를 필터링할지 여부를 결정할 수 있다.

표 3

[0032]

[알고리듬 1]: 이동윈도우에 의한 위치에러검출

Require: P_0 // At least one initial tuple is required
Require: window size n
Require: user sensitivity level s
Ensure: Check validness of new position tuple
Ensure: Calibrated series of tuple $\{P_i : t \geq i > 0\}$ for t inputs
Require: i=0
1: repeat
2: Get P_{i+1} // Acquisition of new tuple, if exist
3: Construct $MA_{speed}(n)$ with $\{P_x : \max(i-n+1, 0) \leq x \leq i\}$
4: Construct $MSD_{speed}(n)$ with $\{P_x : \max(i-n+1, 0) \leq x \leq i\}$
5: Set $MA_{speed} = MA_{speed}(n)$
6: Set $MSD_{speed} = MSD_{speed}(n)$ // Moving Window Construction
7: Set $V_{i+1} = (dist(P_{i+1}, P_i))/(t_{i+1}-t_i)$
8: if $((V_{i+1} > MA_{speed} + s \times MSD_{speed})$ or $(a_{i+1} \geq MAX_{acceleration}))$ and $(V_{i+1} > MIN_{speed})$ then
9: Mark P_{i+1} as filtered. // Filtering
10: end if
11: if $(V_{i+1} \geq MA_{speed} + s_{99.5} \times MSD_{speed})$ and $(V_{i+1} > MIN_{speed})$ then
12: Set $V_{i+1} = MA_{speed} + s_{99.5} \times MSD_{speed}$ // Calibration of Speed
13: end if
14: if $a_{i+1} \geq MAX_{acceleration}$ then
15: Mark P_{i+1} as filtered
16: Set $V_{i+1} = MA_{speed}$
17: Set $a_{i+1} = MAX_{acceleration}$ // Restriction by Maximum Acceleration
18: end if
19: Set i = i + 1
20: until Exist no more input of positioning tuple

[0033] 표 3의 [알고리듬 1]의 각 부분의 기능을 아래에 설명한다.

[0034] * [알고리듬 1]의 3~6 번째 줄 : n개 미만의 튜플이 존재하는 경우, 크기 n의 윈도우를 채우기 위한 완전한 데이터를 준비할 수 있다. 대신에 적은 수의 정보를 이용하여 불완전한 윈도우를 구성할 수 있다.

[0035] * [알고리듬 1]에서 8~10 번째 줄 : 가속도와 속도는 필터링을 위한 파라미터로 사용될 수 있으며 서로 다른 방식으로 사용될 수 있지만, 모두 조절 파라미터(throttling parameter)로서 간주될 수 있다. 속도 V_{i+1} 이 $MA_{speed} + s \times MSD_{speed}$ 로 표현되는 소위 '이동유의구간(moving significant interval)'의 값을 범위를 벗어나거나 지나친 가속도를 가진 위치데이터 튜플은 필터링된다. s는 정규분포의 유의수준(significant level)을 나타낸다.

[0036] * [알고리듬 1]에서 11~13 번째 줄 : 하나의 튜플의 속도가 너무 크면 MA_{speed}와 MSD_{speed} 값에 영향을 주게되며 신뢰구간(confidence interval)의 확장과 같은 필터링 에러를 초래하기 때문에, 필터링-아웃되어야 하는 튜플이 필터링-인(filter-in) 되는 결과가 나타난다. 이와 같이 하나의 튜플의 속도가 너무 크면, 가능한 빠른 속도의 변화를 포함하고 신뢰구간의 확장 오류를 피하기 위해, $MA_{speed}(n) + s_{99.5} * MSD_{speed} = MA_{speed}(n) + 2.57 * MSD_{speed}$ 를 이용하여 속도값을 수정할 수 있다.

[0037] * [알고리듬 1]의 11~13 번째 줄에서 $s_{99.5}$: $s_{99.5} = 2.57$은 정규분포의 99.5%의 신뢰구간을 나타낸다. 이것은 이동윈도우에 대한 에러가 있는 속도의 영향을 감소시키는 조절 파라미터이다.

[0038] * [알고리듬 1]에서 12 번째 줄 : 어떤 하나의 튜플이 수학식 1에 의하여 필터링 아웃(filtered out)되어야 하

는 경우라고 하더라도, 이 튜플의 속도를 이동 윈도우에 반영하기 위하여 정규분포의 99.5% 신뢰구간 내에 있도록 제한(restriction) 또는 교정(calibration)하여, 이 튜플의 속도를 포함할 수 있다. 이는 속도의 빠른 변화에 대처하기 위해 이동윈도우가 빠른 속도값을 포함하여 업데이트 되도록 의도한 것이다. 즉, 오류가 없는 튜플이라고 하더라도 급격한 속도 변화가 있는 것은 필터링 아웃될 수 있다. 그러한 경우에, 튜플이 필터링되더라도, 이동윈도우는 이 튜플에 대한 용납할 수 있는 가능한 최대치의 속도 변화를 반영할 수 있다.

[0039] * [알고리듬 1]에서 11번째 줄 $MIN_{velocity}$: 인간 보행은 보통 2.77m/s (10Km/h)보다 작은 속도 내에서 가능하며 또한 GPS 에러범위에 있기 때문에, 2.77m/s (10Km/h)보다 적은 속도는 필터링 아웃되지 않는다.

[0040] * [알고리듬 1]에서 14번째 줄 $MAX_{acceleration}$: 비현실적인 가속도를 가진 튜플들은 필터링 아웃된다.

[0041] * [알고리듬 1]에서 12~16 번째 줄 : 일단 과도한 가속도를 갖는 튜플은 필터링 아웃되고, 이 튜플의 가속도 값은 $MAX_{acceleration}$ 으로 교정(calibration)되며, 속도값은 비현실적인 속도값의 영향을 무효화시키기(nullify) 위해 $MA_{speed}(n)$으로 교정된다.

[0042] * 차량이 커다란 음의 가속도를 가지고 급하게 멈추는 것은 항상 가능하기 때문에, 음의 가속도 값들을 가진 위치데이터 튜플들은 에러가 없는 유효한 것으로 간주된다. 반면에, $MAX_{acceleration}$보다 더큰 양의 가속도값을 가진 튜플들은 에러가 있는 것으로 간주된다.

[0043] [알고리듬 1]에서 2개의 변수인 n과 s는 사용자에 의해 설정될 수 있다. n은 윈도우 내의 튜플의 개수, 즉 윈도우의 크기이고, s는 필터링의 민감도 수준(즉, 정규분포의 유의수준)이다. 사용자 민감도 수준 s는 상대적으로 간단히 결정할 수 있다. 정규분포의 특성으로부터 적절한 신뢰구간을 갖는 s를 얻을 수 있다. 필터링을 위해 정규분포의 양의 영역만을 사용할 것이기 때문에, 95%의 신뢰구간에 대해서는 s=1.64로 설정하고, 99%의 신뢰구간에 대해서는 s=2.33으로 설정할 수 있다. 사용자들은 그들 자신의 목적에 따라 s를 결정할 수 있다. 예를 들어, 민감도 수준 s를 아래와 같이 선택할 수 있다. [표 2]는 기기가 움직이지 않을 때, 위치데이터 수집에 대한 보통의 에러율(error rate)을 보여준다. GPS의 경우에 대하여, 12.3%의 위치데이터에 에러가 존재하였고, 3GBS 셀룰러 위치검출 시스템에 대하여는 36.75%의 위치데이터에 에러가 존재하였다. 따라서 이 경우, GPS 데이터에 대하여는 s=1.16으로 설정하고, 셀룰러 위치검출 데이터에 대해서는 s=0.34로 설정할 수 있다.

[0044] 상술한 바와 같이 윈도우의 크기가 크면 트레일링 효과가 발생하기 때문에 이를 피하기 위하여 윈도우의 크기를 더 작게 할 수 있다. [알고리듬 1]에 의해 부정확한 속도 값이 교정(calibrate)되고, 비정상적인(abnormal) 가속도 값은 제한되며, 속도 값은 이동윈도우에 의해 산출된 평균속도로 대체될 수 있다. 이러한 교정 방식(calibration mechanism)에 따른 윈도우 크기의 효과를 살펴보기 위해 몇 가지 실험을 하였다. 첫 번째 실험에서는 윈도우 크기를 n = 5, 10, 25, 50, 100으로 선택하였고, 민감도 수준을 s=1.16으로 선택하여 실측된 위치데이터 중의 88%가 정확한 데이터인 것으로 가정하고 필터링하였다. 윈도우 크기가 작은 경우 트레일링 효과가 제한되지만 속도변화에 따른 반응(reaction)이 더 유연한 결과를 얻었다. 첫 번째 실험에서 윈도우의 크기가 5 또는 10인 경우 좋은 결과를 얻었다.

[0045] 두 번째 실험에서는 연속적인 에러의 효과를 고려하였다. 연속적인 에러들은 본 알고리듬에서 사용되는 이동평균과 이동표준편차에 영향을 줄 수 있다. 실제 위치검출 데이터 세트에서 최대 4개의 연속적인 에러가 존재하였고, 따라서 두 번째 실험에 대하여는 n=5로는 부족하다는 결론을 내렸고, n=10인 경우에 연속적인 에러에 대하여 대응할 수 있으며 더 큰 윈도우 크기에 의한 테일링 효과(tailing effect)를 줄일 수 있다고 결론내렸다. 따라서 이 실험에 있어서 윈도우 크기는 최종적으로 10으로 결정되었다. 더 많은 개수의 연속적인 에러를 경험한다면 이에 적합한 윈도우 크기를 선택하거나 윈도우의 크기를 동적으로 증가시킬 수 있다.

[0046] 본 발명에 따른 [알고리듬 1]을 이용한 다양한 실험 결과 두 가지 결론을 얻을 수 있다. 첫째, 윈도우 크기와 민감도 수준을 적절히 선택하면 적절한 필터링 결과를 얻을 수 있다. 둘째, 윈도우 크기와 민감도 수준의 여러 가지 조합에 따라, 모든 위치검출 데이터 세트에 대해 필터링을 수행할 수 있다. 표 4는 파라미터들의 각 조합에 대하여 필터링-아웃된 튜플들의 백분률을 나타낸다. 본 발명의 일 실시예에 따른 알고리듬의 사용자는 그들의 환경에 따라 표 4에 따른 윈도우 크기와 민감도 수준을 선택할 수 있다. [알고리듬 1]에서 윈도우 크기, 민감도 수준, 최대 속도 및 최대 가속도와 같은 알고리듬의 다양한 파라미터들은 사용자에 의해 정의될 수 있다.

사용자가 $MAX_{accelertaion}$, $s_{99.5}$(최대 민감도 수준)과 $MIN_{velocity}$(필터링을 위한 속도의 최소 임계값)와 같은 알고리듬의 상수(constant)를 변화시키는 것도 가능하다.

표 4

[0047]

Size of Sliding Window	s=0.34	s=1.16	s=1.64	s=2.33
5	42.20	29.07	24.06	19.31
10	38.67	24.10	18.81	14.31
25	37.37	21.43	16.07	11.85
50	37.71	20.54	15.23	11.07
100	37.05	20.03	14.90	10.65

[0048] [알고리듬 1]은 아래와 같은 사항을 고려하여 변형할 수 있다.

[0049] 첫째, 윈도우의 크기를 하나의 윈도우 내의 튜플들의 개수로 정의하는 대신에 윈도우가 포함하는 시간 간격으로 정의할 수 있다. 속도는 시간의 함수라는 점을 고려하면, 위치데이터를 규칙적으로 수집할 때에 이 변형된 방식이 효과적이며 또한 정확할 것이라는 것을 알 수 있다.

[0050] 둘째, 윈도우 크기를 동적으로 교정(dynamic calibration)할 수 있다. [알고리듬 1]로부터 변형된 다른 실시예에서는 연속적인 에러의 개수에 따라 윈도우 크기를 동적으로 증가 또는 감소시킬 수 있다. 일단 연속적인 에러의 개수가 많다는 것을 발견하면, 연속적인 에러가 이동평균과 이동표준편차에 미치는 영향을 최소화하기 위하여 윈도우 크기를 증가시킬 수 있다. 만일 연속적인 에러의 개수가 작은 경우에는 윈도우 크기를 감소시킴으로써 급격한 속도 변화에 대해 적절한 반응을 보일 수 있고 필터링을 위한 계산량을 감소시킬 수 있다.

[0051] 셋째, [알고리듬 1]을 실시간 알고리듬이 아닌 의사 실시간 알고리듬(pseudo real time)으로 변형할 수 있다. 윈도우 크기 n에 대해 우리는 (n+1)번째 튜플을 필터링하는 것 대신에 윈도우의 대략 중간에 있는 $\lceil \frac{n}{2} \rceil$ 번째 튜플을 필터링할 수 있다. 이 방법은 완벽하게 실시간으로 구현될 수 없을지라도 언더 필터링과 오버 필터링의 경향을 줄일 수 있다.

[0052] 넷째, [알고리듬 1]에 인터폴레이션(interpolation) 기법을 추가할 수 있다. 아래 표 5에 제시한 [알고리듬 2]는 [알고리듬 1]에 인터폴레이션을 위한 단계가 추가된 것이다([알고리듬 2]의 18~21번째 행 참조). 새로운 튜플(인덱스 i+1)을 획득하였을 때에, 현재 존재하는 윈도우 내의 마지막 튜플(인덱스 i)이 필터링 아웃된 것으로 확인된다면, 이 마지막 튜플의 속도를 인터폴레이션된 값으로 대체할 수 있다. 이러한 인터폴레이션을 이용하면, 위치데이터 튜플의 필터링을 위해 도입된 이동윈도우를 더 정밀하게 제공할 수 있다([알고리듬 2]의 18~21행 참조). 즉, [알고리듬 2]의 가장 마지막 부분에 의해, 새로운 튜플을 얻을 때마다 윈도우 내의 필터링 아웃되었다고 마킹된 마지막 튜플을 인터폴레이션 할 수 있다. 좀 더 정밀한 근사화를 위한 이동윈도우 구성을 위한 또 다른 변형예는, 윈도우의 대략적 중간지점에 있는 $\lceil \frac{n}{2} \rceil$ 번째 튜플을 윈도우 내의 n개의 튜플들을 이용하여 인터폴레이션 하는 것이다. 더 정밀하게 추정(estimation)하기 위하여 n 개의 튜플들로부터 추정된 점근곡선(asymptotic curve)을 이용하여 윈도우 내의 중앙 튜플(middle tuple)을 인터폴레이션하여 더 정밀한 인터폴레이션을 가능하게 할 수 있다. 그러나 이 때문에 계산 오버헤드(computational overhead)가 발생할 수 있기 때문에 이동형 사용자기기에 적용하기 어려울 수 있다.

표 5

[0053]

Algorithm 2 이동윈도우 안정화를 위한 속도의 추정

Require: P_0 // At least one initial tuple is required
Require: window size n
Require: user sensitivity level s
Ensure: Check validness of new position tuple
Ensure: Calibrated series of tuple $\{P_i : t \geq i > 0\}$ for t inputs
Require: i=0
1: **repeat**
2: Get P_{i+1} // Acquisition of new tuple, if exist
3: Construct $MA_{speed}(n)$ with $\{P_x : \max(i-n+1, 0) \leq x \leq i\}$
4: Construct $MSD_{speed}(n)$ with $\{P_x : \max(i-n+1, 0) \leq x \leq i\}$
5: Set $MA_{speed} = MA_{speed}(n)$
6: Set $MSD_{speed} = MSD_{speed}(n)$ // Moving Window Construction
7: **if** $((V_{i+1}>MA_{speed}+s\times MSD_{speed})$ or $(a_{i+1}\geq MAX_{acceleration}))$ and $(V_{i+1}>MIN_{speed})$ **then**
8: Mark P_{i+1} as filtered. // Filtering
9: **end if**
10: **if** $(V_{i+1}\geq MA_{speed}+s_{99.5}\times MSD_{speed})$ AND $(V_{i+1}>MIN_{speed})$ **then**
11: Set $V_{i+1} = MA_{speed} + s_{99.5} \times MSD_{speed}$ // Calibration of Speed
12: **end if**
13: **if** $a_{i+1} \geq MAX_{acceleration}$ **then**
14: Mark P_{i+1} as filtered
15: Set $V_{i+1} = MA_{speed}$
16: Set $a_{i+1} = MAX_{acceleration}$ // Restriction by Maximum Acceleration
17: **end if**
18: **if** (P_i marked as filtered) **then**
19: Set $V_i = ((V_{i+1} - V_{i-1}) \times (t_i - t_{i-1})) / (t_{i+1} - t_{i-1}) + V_{i-1}$
20: Mark P_i as interpolated // Linear Interpolation
21: **end if**
22: Set i = i + 1
23: **until** Exist no more input of positioning tuple

[0054] 도 1은 본 발명의 일 실시예에 따른 속도보정 알고리듬을 설명하기 위한 것이다.

[0055] 도 1의 가로축은 연속적으로 얻은 위치데이터 튜플의 인덱스(즉, 타임스탬프)를 나타내며, 세로축은 메모리에 저장된 각 위치데이터 튜플에 따른 속도이다. 도 1에서는 이동윈도우의 크기 n이 10인 것으로 예시되어 있다. 이동윈도우 내에 존재하는 10개의 속도들로부터 MA_{speed}와 $MA_{speed} + s_{99.5} \times MSD_{speed}$를 계산할 수 있으며, 이 값의 예가 도 1에 표시되어 있다. 도 1에서 인덱스 [i+1]에서의 가속도 a_{i+1}는 $MAX_{acceleration}$ 보다 큰 것으로 가정하였다. 따라서, 인덱스 [i+1]에서의 속도 V_{i+1}는 MA_{speed}으로 보정되었다([알고리듬 2]의 제15행). 도 1에서 인덱스 [i]에 대한 이동 데이터 튜플 P_i는 [알고리듬 2]에 따라 이미 필터링 아웃된 것으로 마킹되었다고 가정하였다([알고리듬 2]의 제18행). (어떤 튜플이 필터링 아웃되었다는 것은 그 튜플이 나타내는 위치에 오류가 있다고 판단했다는 뜻이지만, 그 튜플의 속도나 가속도 값은 여전히 저장되어 있을 수 있다.) 따라서 [알고리듬 2]의 제19행에 따라, 이동 데이터 튜플 P_i의 속도 V_i는 이동 데이터 튜플 P_{i-1}의 속도 V_{i-1}과 이동 데이터 튜플 P_{i+1}의 속도 V_{i+1}를 이용하여 선형 인터폴레이션 된 값으로 대체된다. 이와 같이 [알고리듬 2]에 의하면 새로 획득한 튜플의 속도는 [알고리듬 2]의 제10~17행에 의해 1차적으로 교정되고, 그 후 또 다른 새로운 튜플이 획득될 때에 이 교정된 속도는 [알고리듬 2]의 제18~21행에 의해 2차적으로 교정될 수 있다. 임의의 튜플의 속도에 대하여 상술한 1차적 교정과 2차적 교정이 모두 일어나거나, 둘 중 하나만 일어나거나 또는 전혀 일어나지 않을 수 있

음을 이해할 수 있다.

[0056] 상술한 튜플 P_i의 속도 V_i가 인터폴레이션되어 교정된 값은 [알고리듬 2]의 제11행에 의해 교정된 값(MA_{speed} + $s_{99.5}$ × MSD_{speed})보다 작은 것이 보통이지만, 그 다음 타임스탬프에서의 이동유의구간 또는 교정된 값에 따라 더 크게 나올 수도 있다.

[0057] [알고리듬 2]에서는 새롭게 획득한 이동 데이터 튜플(P_{i+1}) 및 이동윈도우에 존재하는 가장 최신의 이동 데이터 튜플(P_i)에 대한 처리가 이루어졌다. 이와 달리 본 발명의 다른 실시예에서는 이동윈도우 내의 임의의 타임 스탬프에 대한 이동 데이터 튜플(P_m)에 대한 처리를 수행하도록 변형될 수 있다. 이를 위하여 [알고리듬 2]의 제18~21행을 표 6과 같이 변형할 수 있다.

표 6

[0058]

```
17:   if (P_m marked as filtered) then //  Linear Interpolation
18:     Set V_m = ((V_m+1 - V_m-1)×(t_m - t_m-1)) / (t_m+1 - t_m-1) + V_m-1
19:     Mark P_m as interpolated
20:   end if
```

단, m=i-k이고, k는 사용자 조절가능 파라미터

[0059] 도 2는 본 발명의 다른 실시예에 따라, 최소 자승법으로 속도값을 인터폴레이션하는 속도보정방법을 설명하기 위한 것이다.

[0060] 도 2에서는 [알고리듬 2]의 제18행 내지 제21행에 의한 인터폴레이션 방법으로부터 변형된 인터폴레이션 방법을 사용한 결과를 나타낸다. 도 2에서는 이동 윈도우 내의 복수 개의 데이터를 이용한 비선형 인터폴레이션 방법을 이용한다. 예컨대, 이동 윈도우 내에 포함된 복수 개의 데이터와 최소 자승법을 이용하여 근사화한 점근선에 해당하는 값을 이용하여 인덱스 [i-4]를 갖는 이동 데이터 튜플을 보정할 수 있다.

[0061] 도 3 내지 도 5는 본 발명의 다른 실시예에 따른 위치튜플 필터링 방법, 인터폴레이션을 이용한 속도 추정방법, 및 속도에러 교정방법을 설명하기 위한 것이다.

[0062] 도 3 내지 도 5의 가로축은 타임스탬프의 인덱스를 나타내며 세로축은 속도를 나타낸다. 참조번호 100은 각 타임스탬프에서 실측된 속도를 나타내며, 참조번호 101은 본 명세서에 공개한 알고리듬 중 하나를 사용하여 필터링 아웃한 튜플로부터 얻은 실측속도를 나타낸다. 도 3 내지 도 5에서는 윈도우 크기가 10인 예를 나타내었다. 참조번호 103은 윈도우 내의 속도의 평균값을 나타내며 참조번호 102는 상술한 이동유의구간(moving significant value)의 값을 나타낸다. 도 3에 나타낸 참조번호 101의 위치튜플은, 그 속도가 상기 이동유의구간의 값보다 크기 때문에 필터링 아웃되었다.

[0063] 도 4의 참조번호 101(인덱스 i)은 본 발명의 알고리듬에 따라 필터링 아웃된 위치튜플로부터 얻은 속도를 나타낸다. 도 4의 참조번호 200은 인덱스 i의 속도값이 그 전후의 인덱스 i-1 및 인덱스 i+1의 속도값들에 의해 선형 인터폴레이션되어 제공된 속도의 추정값(estimated value)을 나타낸다.

[0064] 도 5의 경우에는 인덱스 i 및 인덱스 i+1의 튜플이 모두 필터링 아웃된 경우를 나타낸다. 이때, 인덱스 i의 실측속도(인덱스 i의 참조번호 101 참조)가 이동유의구간의 값보다 크기 때문에 소정의 크기로 교정(calibration)된다(인덱스 i의 참조번호 300 참조). 그 다음, 인덱스 i+1의 실측속도(인덱스 i+1의 참조번호 101 참조) 역시 이동유의구간의 값보다 크기 때문에 소정의 크기로 교정(calibration)된다(인덱스 i+1의 참조번호 300 참조). 그 다음, 인덱스 i의 튜플은 필터링 아웃되었기 때문에, 인덱스 i의 속도는 인덱스 i+1과 인덱스 i-1의 속도를 이용하여 인터폴레레이션 된다. 말하자면, 본 발명의 일 실시예에서는, 속도의 교정(calibration)이 속도의 추정(estimation)을 정확하게 하기 위한 전단계로서 이용될 수도 있다.

[0065] [알고리듬 2]를 이용하면 실측된 튜플 중 필터링 아웃된 튜플에 대한 속도를 추정할 수 있지만, 필터링 아웃된

튜플의 위치값은 단순히 제거된 상태이다. 아래 표 3의 [알고리듬 3]을 이용하면 상기 필터링 아웃된 튜플에 대한 위치값을 추정할 수 있다. [알고리듬 2]의 속도 추정에서는 양(+)의 속도값만을 고려하면 되었지만, 이에 비하여 위도와 경도에는 방향성이 존재하기 때문에 [알고리듬 3]에 추가된 위치 추정에서는 위도와 경도의 음(-)의 차이(difference)를 무시할 수 없다는 차이점이 있다.

표 7

[0066]

Algorithm 3 : 이동윈도우를 이용한 위치추정

==========

Require: P_0 // At least one initial tuple is required

Require: window size n

Require: user sensitivity level s

Require: error tolerance of distance ET_D

Ensure: Check validness of new position tuple

Ensure: Calibrated series of tuple $\{P_i : t \geq i > 0\}$ for t inputs

Require: i=0 // V_i is speed calculated from P_{i-1} to P_i

1: **repeat**

2: Get P_{i+1} // Acquisition of new tuple, if exist

3: Set $V_{i+1} = dist(P_{i+1}, P_i)/(t_{i+1} - t_i)$

// dist(): distance between two geographic coordinate system points

4: Set $MIN_{speed} = ET_D/(t_{i+1} - t_i)$

5: Construct $MA_{speed}(n)$ with $\{P_x : max(i-n+1, 0) \leq x \leq i\}$

6: Construct $MSD_{speed}(n)$ with $\{P_x : max(i-n+1, 0) \leq x \leq i\}$

7: Set $MA_{speed} = MA_{speed}(n)$

8: Set $MSD_{speed} = (MSD_{speed}(n) > MIN_{speed})\ ?\ MSD_{speed}(n) : MIN_{speed}$

// Allow minimum room for moving significant interval

9: Set $Vlat_{i+1} = ||lat_{i+1}, lat_i||/(t_{i+1} - t_i)$

10: Construct $MA_{Vlatitude}(n)$ with $\{P_x : max(i-n+1, 0) \leq x \leq i\}$

11: Set $MA_{Vlatitude} = MA_{Vlatitude}(n)$

12: Set $Vlon_{i+1} = ||lon_{i+1}, lon_i||/(t_{i+1} - t_i)$

13: Construct $MA_{Vlongitude}(n)$ with $\{P_x : max(i-n+1, 0) \leq x \leq i\}$

14: Set $MA_{Vlongitude} = MA_{Vlongitude}(n)$

15: **if** $((V_{i+1} > MA_{speed} + s \times MSD_{speed})$ or $(a_{i+1} \geq MAX_{acceleration}))$ and $(V_{i+1} > MIN_{speed})$ **then**

16: Mark P_{i+1} as filtered. // Filtering

17: **end if**

18: **if** $(V_{i+1} \geq MA_{speed} + s_{99.5} \times MSD_{speed})$ and $(V_{i+1} > MIN_{speed})$ **then**

19: Set $V_{i+1} = MA_{speed} + s_{99.5} \times MSD_{speed}$ // Calibration of Speed

20: **end if**

21: **if** $a_{i+1} \geq MAX_{acceleration}$ **then**

22: Mark P_{i+1} as filtered

23: Set $V_{i+1} = MA_{speed}$

24: Set $a_{i+1} = MAX_{acceleration}$

25: Set $lat_{i+1} = lat_i + sign(lat_{i+1}-lat_i) \times MA_{Vlatitude} \times (t_{i+1}-t_{i-1})$

26: Set $lon_{i+1} = lon_i + sign(lon_{i+1}-lon_i) \times MA_{Vlongitude} \times (t_{i+1}-t_{i-1})$

// Restriction by Maximum Acceleration

27: **end if**

28: **if** (P_i marked as filtered) **then**

29: Set $V_i = ((V_{i+1} - V_{i-1}) \times (t_i - t_{i-1})) / (t_{i+1} - t_{i-1}) + V_{i-1}$

30: Mark P_i as interpolated // Linear Interpolation of speed

31: Set $lat_i = ((lat_{i+1} - lat_{i-1}) \times (t_i - t_{i-1})) / (t_{i+1} - t_{i-1}) + lat_{i-1}$

32: Set $lon_i = ((lon_{i+1} - lon_{i-1}) \times (t_i - t_{i-1})) / (t_{i+1} - t_{i-1}) + lon_{i-1}$

// Estimation of Position by interpolation

33: **end if**

34: Set i = i + 1

35: **until** Exist no more input of positioning tuple

[0067] 이하 상기 [알고리듬 3]의 내용을 이용하여 본 발명의 일 실시예에 따른 위치교정방법을 설명한다.

[0068] [알고리듬 3]에서는 통상의 거리오류를 허용하기 위한 파라미터 ET_D가 도입된다(4행). 통상의(usual) 포지셔닝 시스템의 CEP(Circular Error Probable) 개념으로부터, 예컨대 SA가 활성화된 GPS 시스템에 대하여 ET_D를 15m로 설정할 수 있다. 물론 포지셔닝 시스템의 정확성에 따라 ET_D의 값을 변경할 수도 있다. ET_D에 의해 최저속도 및 위도와 경도의 최소 차이(difference)가 결정될 수 있으며, 이동 표준편차에 영향을 줄 수 있으며, 그 결과 이동 유의구간이 0이 아닌 값을 갖도록 할 수 있다. 즉, ET_D를 이용함으로써 포지셔닝 시스템의 CEP라는 개념을 본 발명의 일 실시예에 따른 [알고리듬 3]에 도입할 수 있으며, 포지셔닝 시스템의 정확도 내의 작은 가능한 속도 오류 때문에 튜플들을 오버필터링(overfiltering)하는 것을 회피할 수 있다.

[0069] [알고리듬 3]에 따르면, [알고리듬 2]의 속도추정 프로세스와 비슷하게, 필터링 된 튜플에 대하여 위치추정 프로세스를 도입할 수 있다. [알고리듬 3]에 포함된 위치추정 프로세스의 주된 다른 점은 위도와 경도의 방향성에 있다. 즉, 속도값에 대해서는 양의 값만을 고려하면 되었으나, 위치를 추정하기 위해서는 위도와 경도의 음의 차이(difference)를 무시할 수 없다. [알고리듬 3]은 위치 추정에 대한 전체 프로세스를 나타낸 것이며, 이는 [알고리듬 2]의 속도 추정을 위한 전체 프로세스에 대응한다.

[0070] [알고리듬 3]의 Vlat와 Vlon은 각각 위도와 경도의 차이(difference)를 지칭한다. [알고리듬 3]은 Vlat와 Vlon의 이동 평균을 계산한다. 하나의 튜플이 과다한 속도에 의해 필터링되는 경우 이는 위치의 차이가 과다함을 나타낸다. 일단 이렇게 되면, 올바른 위치를 추정할 필요가 있다. 가속도 오류가 발생한 경우에는, 속도에 대하여 수행한 것과 마찬가지로, 위치 데이터를 차이값의 이동평균(MA of difference) 값으로 보정한다(25 및 26행). 마지막으로, 속도 추정에 대하여 수행한 바와 마찬가지로, 이동 윈도우의 끝에서 위도와 경도에 대하여 인터폴레이션한다(31 및 32행). 그리고 차이값(difference)의 방향성도 보존된다(25, 26, 31, 및 32행).

[0071] 4행에서 위치 튜플의 시간 간격이 커지는 경우 ET_D에 의한 MIN_{speed} 값이 매우 작아지는 경우가 있다. 이 때 MIN_{speed} 값의 일정 하한선을 보장하기 위하여 다음과 같이 4행을 변경 가능하다. 여기서 MINSPEED는 발명의 사용자가 임의로 정한 최저 속도이다.

[0072] 4: Set MIN_{speed} = $(ET_D/(t_{i+1}-t_i)>MINSPEED)?ET_D/(t_{i+1}-t_i):MINSPEED$

[0073] 이하 본 발명의 일 실시예에 따른 위치정보 처리방법은 [알고리듬 3]을 참조하여 설명한다.

[0074] 이 방법은, 위치정보 수집장치로부터 수집한 위도(lat), 경도(lon), 및 타임스탬프(t)를 포함하여 이루어지는 튜플(P_i)에 관한 정보를 처리하는 방법으로서, 타임스탬프[i](t_i)에서의 튜플[i](P_i)에 오류가 있는지 여부(즉, 필터링된 것인지 여부)를 확인하는 단계(28행); 및 상기 오류가 존재하는 것으로 확인된 경우에는, 타임스탬프[i+1](t_{i+1})에서의 위도[i+1](lat_{i+1})와 상기 타임스탬프[i] 이전에 획득한 한 개 이상의 위도들을 인터폴레이션하여 상기 타임스탬프[i]에서의 위도[i](lat_i)를 추정(estimation)하고(31행), 상기 타임스탬프[i+1]에서의 경도[i+1](lon_{i+1})와 상기 타임스탬프[i] 이전에 획득한 한 개 이상의 경도들을 인터폴레이션하여 상기 타임스탬프[i]에서의 경도[i](lon_i)를 추정하는 단계(32행)를 포함할 수 있다.

[0075] 이때, 상기 추정하는 단계 이전에, 상기 타임스탬프[i+1]에서의 가속도[i+1](a_{i+1})에 오류가 있는지 여부($a_{i+1} \geq MAX_{acceleration}$)를 결정하는 단계를 더 포함하며(21행), 상기 가속도[i+1]에 오류가 있는 것으로 판단된 경우, 상기 추정하는 단계 이전에 상기 위도[i+1] 및 상기 경도[i+1]를 미리 제한(restriction)하도록 되어 있을 수 있다(25 및 26행).

[0076] 이때, 상기 결정하는 단계에서, 상기 가속도[i+1]가 미리 설정된 최대가속도($MAX_{acceleration}$) 이상인 경우에는 상기 가속도[i+1]에 오류가 있는 것으로 결정할 수 있다(21행).

[0077] 이때, 상기 타임스탬프[i]까지의 평균경도($MA_{Vlongitude}$) 및 상기 타임스탬프[i]에서의 경도[i](lon_i)를 이용하여

상기 경도[i+1]를 교정하고, 상기 타임스탬프[i]까지의 평균위도($MA_{Vlatitude}$) 및 상기 타임스탬프[i]에서의 위도[i](lat_i)를 이용하여 상기 위도[i+1]를 교정하도록 되어 있을 수 있다(25 및 26행).

[0078] 이때, 상기 추정하는 단계는, 상기 오류가 존재하는 것으로 확인된 경우에는, 타임스탬프[i+1]에서의 속도[i+1](V_{i+1})와 상기 타임스탬프[i] 이전에 획득한 한 개 이상의 속도들을 인터폴레이션하여 상기 타임스탬프[i]에서의 속도[i](V_i)를 추정(estimation)하는 단계를 포함할 수 있다(29행).

[0079] 또한, 상기 추정하는 단계 이전에, 상기 타임스탬프[i+1]에서의 속도[i+1](V_{i+1})에 오류가 있는지 여부를 결정하는 단계를 더 포함할 수 있으며(18행), 상기 속도[i+1]에 오류가 있는 것으로 판단된 경우, 상기 추정하는 단계 이전에 상기 속도[i+1]를 미리 교정(calibration)하도록 되어 있을 수 있다(19행).

[0080] 상기 속도[i+1]가 미리 결정된 최저속도(MIN_{speed}) 이하인 경우에는 상기 속도[i+1]에 오류가 없는 것으로 판단하며(18행), 상기 최저속도는 허용오차거리(error tolerance of distance, ET_D)에 의해 결정될 수 있다(4행).

[0081] 본 발명의 다른 실시예에 따른 사용자기기는, 위도, 경도, 및 타임스탬프를 포함하여 이루어지는 튜플에 관한 정보를 수집하도록 되어있는 위치정보 수집부; 및 상기 튜플에 관한 정보를 처리하도록 되어 있는 처리부를 포함할 수 있다.

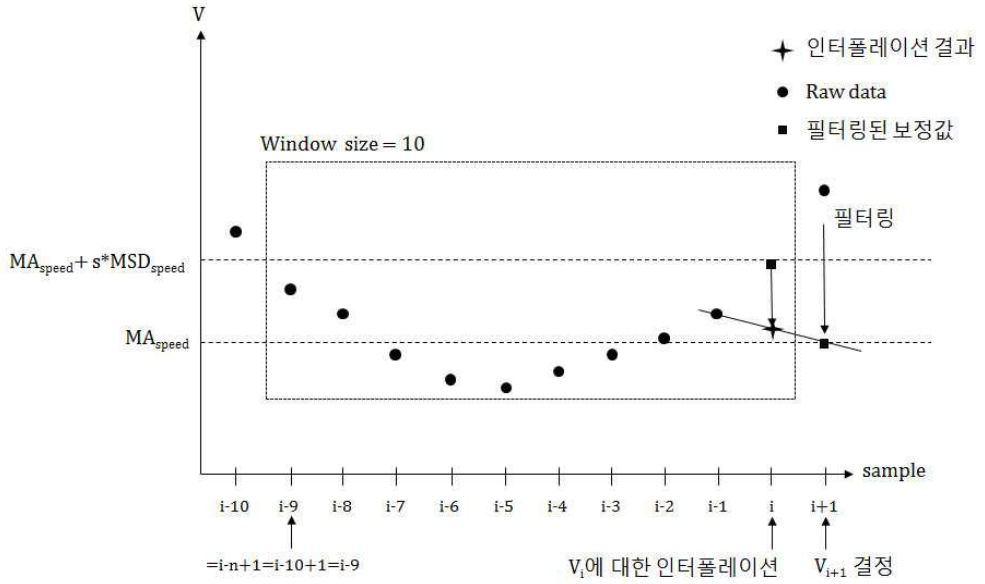
[0082] 이때, 상기 처리부는, [알고리듬 3]의 28행, 31행 및 32행의 단계를 수행할 수 있다. 또한, 18행 및 19행의 단계를 더 수행할 수 있다.

[0083] 도 6은 본 발명의 일 실시예에 따른 속도보정 없이 검출한 속도에러를 설명하기 위한 것이고, 도 7는 본 발명의 일 실시예에 따른 인터폴레이션 기법에 따른 속도보정을 적용하여 검출한 속도에러를 설명하기 위한 것이다. 각각의 도면은 실측 속도(original speed), 가속도, 이동유의구간, 교정속도(calibrated speed), 추정속도(estimated speed), 검출된 가속도 에러에 관한 정보를 포함한다. 가속도 에러는 역삼각형 모양으로 도시된다. 다만, 도 6은 인터폴레이션 된 속도에 관한 정보를 포함하지 않는다. 속이 빈 사각형(empty rectangle)은 본 발명의 일 실시예에 따른 교정기법(calibration mechanism)에 따라 제한된 속도를 나타내며, 검은색 사각형(dark rectangle)은 인터폴레이션 기법에 의한 추정속도를 나타낸다. y축의 속도는 m/s 단위이고, x축은 하루 중의 시각을 나타낸다. 13:33:06 근처에서 실측 속도가 이동 유의 구간의 바깥에 위치하는 에러가 발견되었음에도 불구하고, 본 발명의 일 실시예에 따른 알고리듬은 가능한 속도값을 잘 추정하였다. 2012년 12월 18일 13:32:59에 가속도 제한에 의한 명백한 에러가 발행하였지만, 본 발명의 교정방식에 의해 적절한 범위 안으로 속도 오류가 제한되었으며, 그 다음에는 속도 추정 방식에 의해 속도가 보정된다. 오류 속도를 교정하는 방법보다는, 인터폴레이션 방식에 의해 속도를 추정하는 방법이 이동 윈도우에 미치는 영향을 더 줄일 수 있다. 따라서 도 7에서 인터폴레이션을 뒤따르는 이동 유의 구간은 차이점을 나타내는데, 이는 인터폴레이션된 속도 값에 의해 이동 윈도우의 통계값이 달라지기 때문이다. 13:33:04에서의 차이점은 상당히 크다.

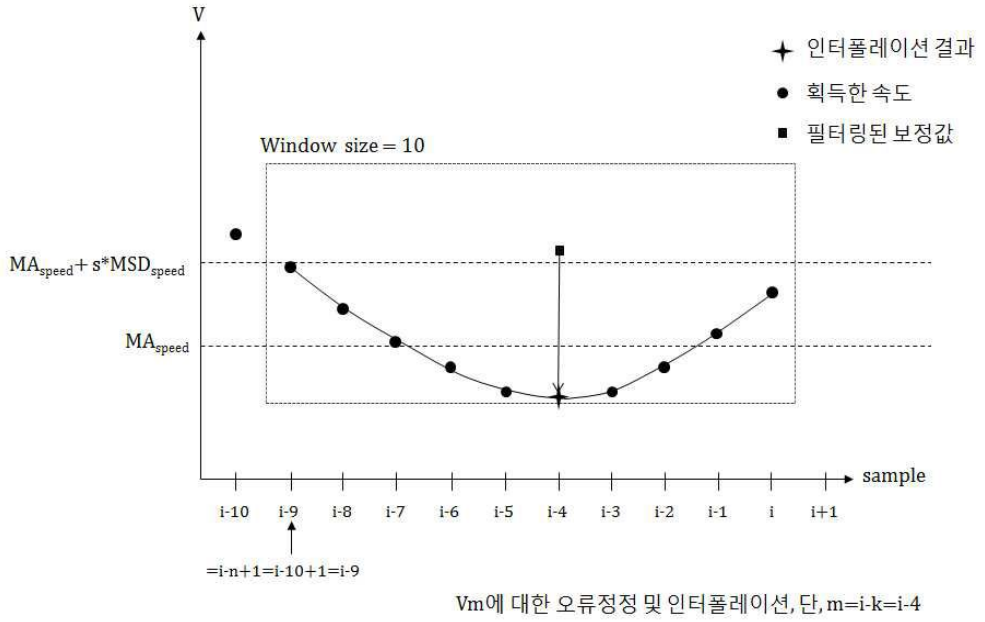
[0084] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 당해 기술분야에서 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 다른 실시예가 가능하다는 점을 이해할 것이다. 따라서 본 발명의 진정한 기술적 보호 범위는 첨부된 특허청구범위의 기술적 사상에 의하여 정해져야 할 것이다.

도면

도면1

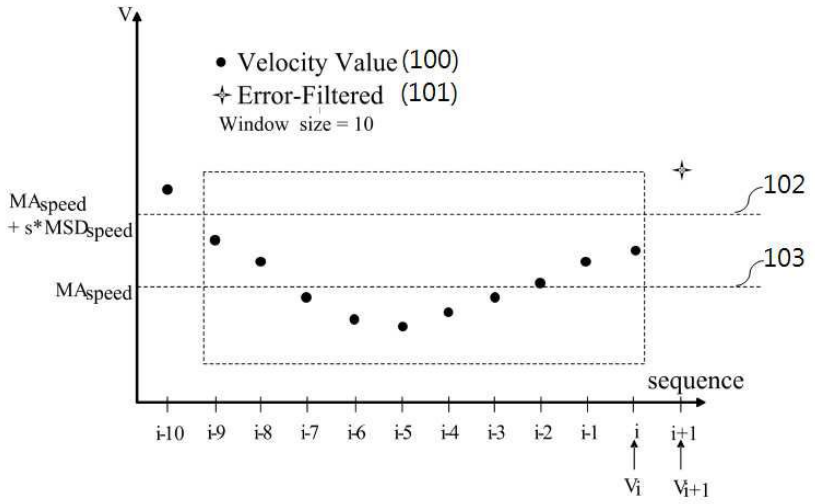


도면2

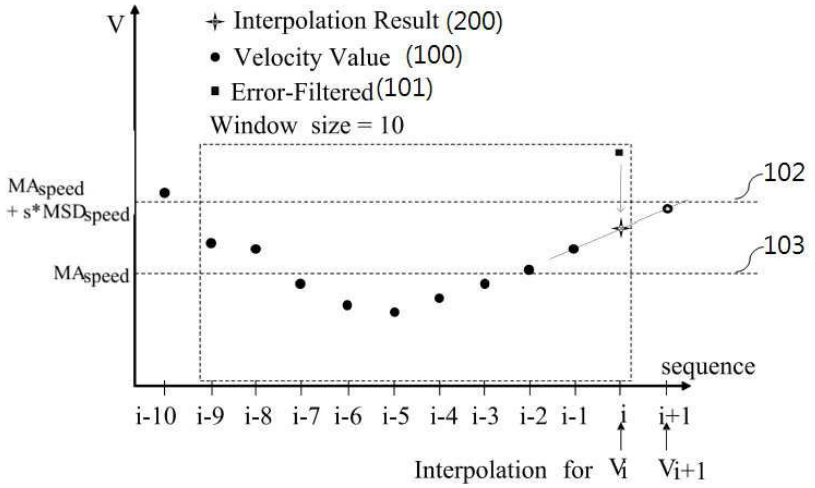


Vm에 대한 오류정정 및 인터폴레이션, 단, m=i-k=i-4

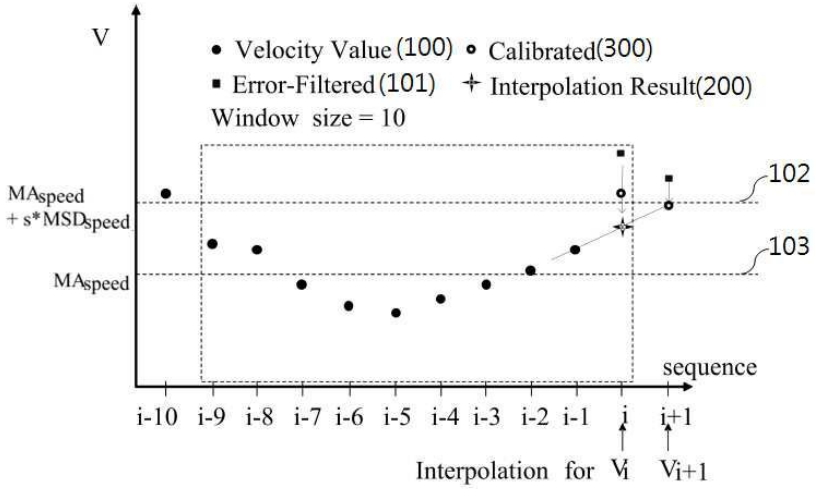
도면3


V
Velocity Value (100)
Error-Filtered (101)
Window size = 10
MAspeed + s*MSDspeed
MAspeed
102
103
sequence
i-10 i-9 i-8 i-7 i-6 i-5 i-4 i-3 i-2 i-1 i i+1
Vi Vi+1

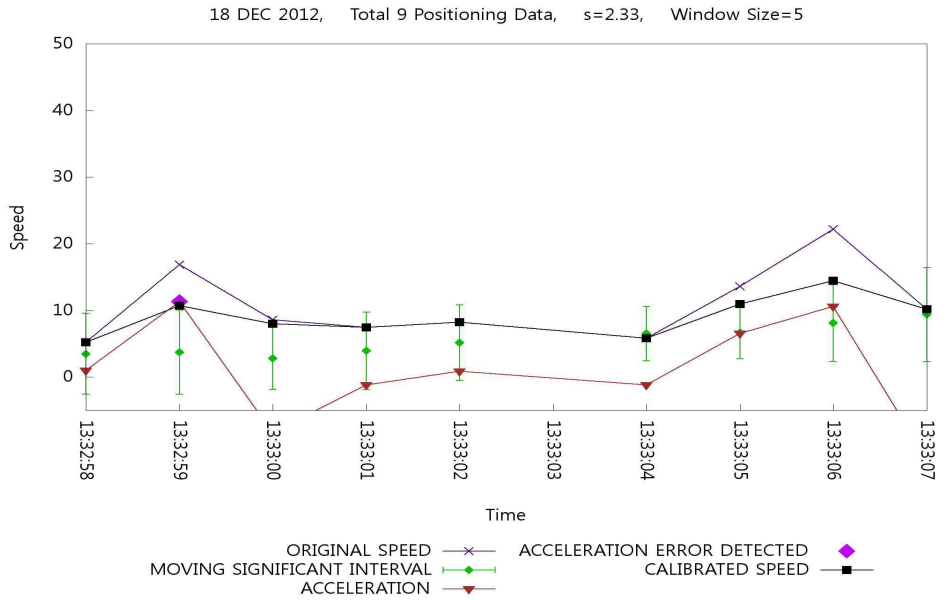
도면4


V
Interpolation Result (200)
Velocity Value (100)
Error-Filtered (101)
Window size = 10
MAspeed + s*MSDspeed
MAspeed
102
103
sequence
i-10 i-9 i-8 i-7 i-6 i-5 i-4 i-3 i-2 i-1 i i+1
Interpolation for Vi Vi+1

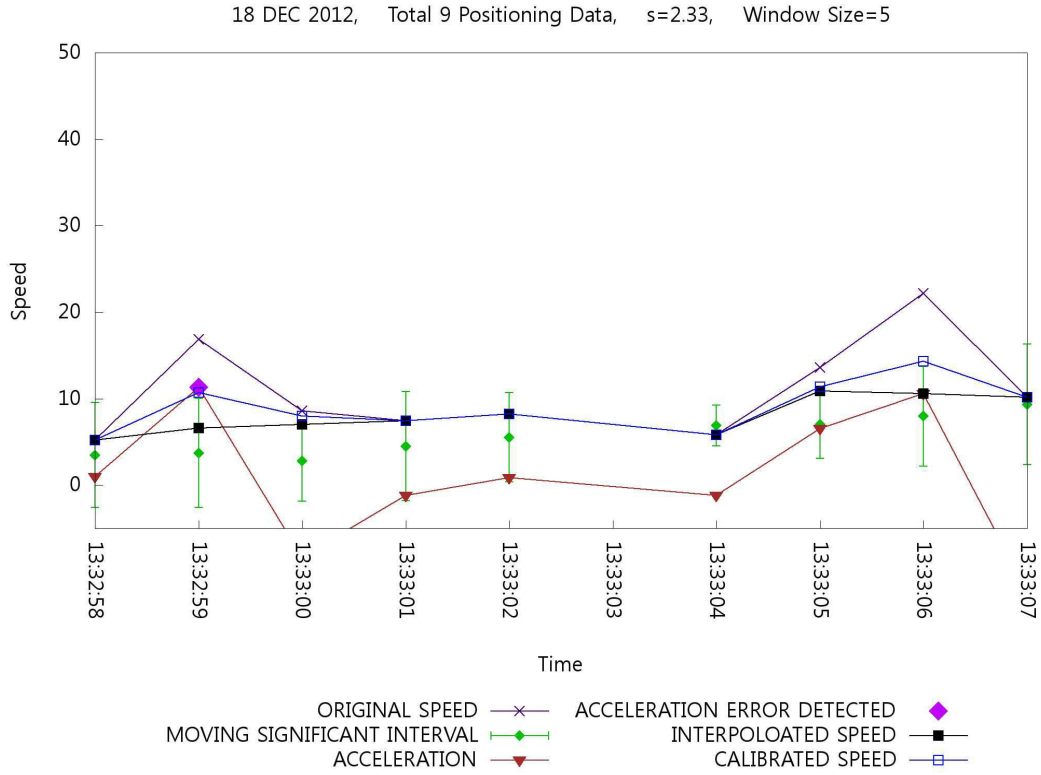
도면5


V
Velocity Value (100)
Calibrated (300)
Error-Filtered (101)
Interpolation Result (200)
Window size = 10
MAspeed + s*MSDspeed
MAspeed
102
103
sequence
i-10 i-9 i-8 i-7 i-6 i-5 i-4 i-3 i-2 i-1 i i+1
Interpolation for Vi Vi+1

도면6



도면7



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 청구항 3 발명

【변경전】

"상기 결정하는 단계에서,"

【변경후】

내용 삭제