이동 윈도우 메커니즘을 이용한 위치데이터 오류 검출 및 교정

온경운, 송하윤, 김현욱
홍익대학교 정보 컴퓨터 공학부 컴퓨터공학 전공
okw0311@hotmail.com, hayoon@wow.hongik.ac.kr, gizingun@gmail.com

Detection and Correction of Erroneous Positioning Data With Moving Window Mechanism

Kyoung-woon On, Ha Yoon Song, Hyunuk Kim
Department of Computer Engineering, Hongik University

요 약

오늘날 모바일 스마트 기기의 발전은 위치기반의 새로운 기술을 이끌었다. 현재 위치 데이터를 사용하는 많은 응용프로그램들이 소개되었고, 또한 널리 사용되고 있다. 하지만 아직 이러한 위치 데이터들은 환경적인 요소 등으로 인해 오류가 많다. 본 논문에서는 연속적인 위치 데이터들 사이에서 오류를 찾아내는 알고리즘을 제안할 것이다. 이동 윈도우에서의 이동 평균과 이동 표준편차가 이동 유의 구간을 구성할 것이고, 이 이동 유의 구간은 오류 데이터들을 찾아내는 데에 사용될 것이다. 또한 오류 데이터를 교정하는 방식도 제안할 것이다. 이러한 일련의 과정을 본 논문에서는 알고리즘으로 나타낸 후 실험을 통해 입증할 것이다. 이러한 방식의 접근이 다른 위치기반 응용프로그램이나 인간 이동 연구에 도움이 될 것이라 생각한다.

1. 서론

스마트 폰과 같이 휴대가 용이한 모바일 기기의 발달로 사용자들의 위치를 추적, 분석할 수 있는 다양한 위치기반 시스템의 사용이 가능하게 되었다. 또한 현재 GPS, GLONASS, Galileo 같은 Wifi나 이동통신망을 통한 위치 기반 시스템이 널리 이용되고 있다. 그러나 위치 데이터들은 환경적 요소 등 다양한 이유로 인해 상당수 오류를 가지고 있다. 모바일 기기 측면에서 보면 이러한 오류는 기기 스스로 필터링 하고 교정할 필요가 있다. 그러나 모바일 기기는 연산능력이 일반 PC보다 떨어지므로 복잡한 필터링 메커니즘은 지양해야 한다.

본 논문에서, 우리는 이동 윈도우의 사용을 통한 오류 위치 데이터 필터링 및 교정 기술을 제안할 것이다. 다음에 올 섹션에서는 이동 윈도우의 개념 및 위치 데이터에 관한 정보, 알고리즘에서 사용될 기본 개념들을 설명하고, 섹션3 에서는 알고리즘을 제안할 것이고, 섹션 4에서는 실험, 섹션 5에서는 결과 및 향후 연구방향에 대해서 논의하게 될 것이다.

2. 위치 데이터 및 이동 윈도우

수집된 데이터의 형태는 gps 데이터로 <위도, 경도, 시간>의 형태로 되어있다. 우리는 이러한 형태의 데이터에서 <위도$_t$, 경도$_t$>, <위도$_{t-1}$, 경도$_{t-1}$>를 이용하여 Vincenty의 공식을 통해 이동거리를 계산하였다.[1] 또한 이러한 이동거리를 이용하여 속력을 계산하고 그에 따른 가속도를 계산하여 한 개체가 <위도, 경도, 시간, 속도, 가속도>의 데이터를 가질 수 있도록 하였다.

필터링의 기준은 이전 위치에서 다음 위치로의 이동시 속력으로 하였는데 우리는 600m/s같은 실제 사람이 이동시에 발견할 수 없는 의미 없는 속력을 제한하기 위해 최대속력을 250m/s으로 정의하고 이에 더하여 최대 가속도도 10.8m/s^{2}정의하였다[3]. 다음으로 우리는 위치 데이터들의 최근 경향을 감안하여 필터링을 하기 위하여 이동 윈도우의 개념을 도입하였다. 이는 최근 n개의 데이터들의 통계량을 통해 필터링의 기준을 내세우는 것으로 이동 윈도우 내의 평균속도를 MA_{speed}(Moving Average of Speed), 그것의 표준편차를 MSD_{speed}(Moving Standard Deviation)으로 표현하였다.

· $MA_{speed}(n)$: $\{P_x : t - n + 1 \le x \le t\}$내의 평균속력
· $MSD_{speed}(n)$: $\{P_x : t - n + 1 \le x \le t\}$내의 표준편차
(t : 현재시각)

이러한 최근 n개 데이터들은 여러 확률 밀도 함수로 표현할 수 있는데 우리는 경향을 가장 잘 나타낼 수 있는 표준정규분포로 나타낼 것이다. 따라서 만약 ($V_{i+1} > MA_{speed} + s \times MSD_{speed}$)이면 새로 얻어진 V_{i+1}은 오류로 간주되며

우리는 (MA_{speed} + s × MSD_{speed})를 이동 유의 구간이라고 칭할 것이다. 또한 이동 유의 구간을 넘어서는 속력이라도 미리 정의한 MIN_{speed}, 즉, 최소 속도를 넘어서지 않으면 사람이 이동할 수 있는 정상적인 속도로 간주하여 필터링 하지 않을 것이다.

우리는 오류 데이터를 교정하기 위해 모바일 기기 같은 연산 능력이 낮은 기기에서도 용이하게 수행할 수 있는 선형 보간법을 사용하였다. 새로운 위치 데이터가 들어오면 이동 윈도우의 끝에 위치한 오류 데이터는 선형 보간법에 의해 조정되는 과정을 거치는 방식으로 알고리즘이 진행된다. 또한 이동 윈도우 기반의 필터링 알고리즘이 효과적이긴 하지만 아직 과하게 필터링 되거나 의도된 것보다 덜 효율적으로 필터링 될 여지가 남아있다. 이는 비현실적인 속도나 가속도의 데이터가 들어오는 경우인데 이들은 이동 윈도우 안의 평균속도, 표준 편차에 큰 영향을 주게 되기 때문에 우선적으로 보정할 필요가 있다. 따라서 이를 위해 전자의 경우 (MAspeed + $s_{99.5}$ × MSDspeed) 보다 큰 속도는 이 값으로 맞추어 조금 더 타당한 에러 속력으로 보정하고(Calibration speed), 후자의 경우 속도는 MAspeed로, 가속도는 MAXacceleration 으로 값을 미리 보정하는 방법을 사용하였다.

3. 알고리즘

섹션 2에 의거하여 우리는 <표 1>에서 볼 수 있는 에러 데이터를 필터링하고 교정하는 알고리즘을 만들었다. 새로 들어오는 데이터를 P_{i+1}이라 할 때 이 알고리즘은 P_{i+1}이 오류인지 아닌지를 판명하고 그 후에 P_i가 오류 데이터로 판명 되었으면 그것을 선형 보간법을 이용해 교정할 것이다. 실제로 사용될 때 우리는 몇 가지 사항을 더 고려하여야한다.

먼저 초기 이동 윈도우의 설계를 살펴보면, 만일 들어온 데이터의 개수가 n보다 작으면 우리는 완전한 MA_{speed} 및 MSD_{speed}를 정의할 수 없다. 그 대신에 우리는 보다 적은 수의 데이터를 가지고 위의 것을 정의해야 한다.

둘째로 속도가 MIN_{speed}를 넘고 이동 유의 구간 이상이거나, 가속도가 $MAX_{acceleration}$을 넘으면 오류데이터로 표시한다.

셋째로 일단 속력이 MA_{speed} + 2.57($s_{99.5}$) × MSD_{speed}를 넘어서고 MIN_{speed}보다 크면 속력을 MA_{speed} + 2.57($s_{99.5}$) × MSD_{speed}로 조정한다. $s_{99.5}$는 표준 정규 분포에서 신뢰도 99.5%에 해당하는 값이다.

다음으로 비현실적인 가속도는 필터링 되어야 한다. 또한 이때의 속도는 MA_{speed}, 가속도는 $MAX_{acceleration}$로 보정한다. [3]에서 볼 수 있듯이 10.8m/s^2은 스포츠카의 최대 가속도 값이다.

마지막으로 오류로 판명된 데이터는 선형 보간법을 통해 속도를 교정한다.

알고리즘 : 속도 교정을 이용한 필터링

Require: P_0
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

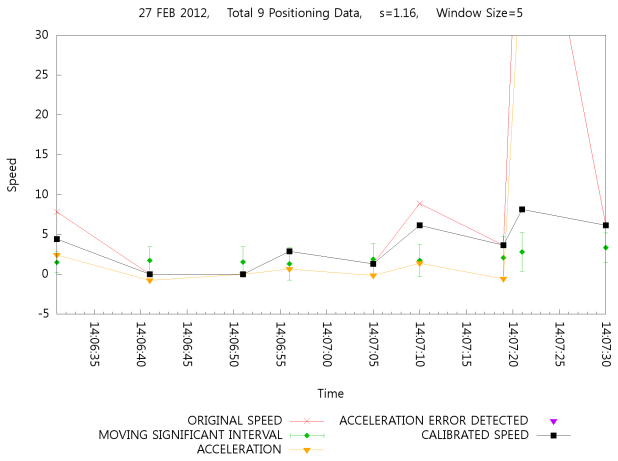
repeat Get P_{i+1}
  Construct $MA_{speed}(n)$ with $\{P_x : \max(i \leq n+1 < 0) \leq x \leq I\}$
  Construct $MSD_{speed}(n)$ with $\{P_x : \max(i \leq n+1 < 0) \leq x \leq i\}$
  Set $MA_{speed} = MA_{speed}(n)$
  Set $MSD_{speed} = MSD_{speed}(n)$
  if $((V_{i+1} > MA_{speed} + s \times MSD_{speed})$ **or** $(a_{i+1} \geq MAX_{acceleration}))$ **and** $(V_{i+1} > MIN_{speed})$ **then**
    Mark P_{i+1} as filtered.
  end if
  if $(V_{i+1} \geq MA_{speed} + s_{99.5} \times MSD_{speed})$ **and** $(V_{i+1} > MIN_{speed})$ **then**
    Set $V_{i+1} = MA_{speed} + s_{99.5} \times MSD_{speed}$
  end if
  if $(a_{i+1} \geq MAX_{acceleration})$ then
    Mark P_{i+1} as filtered
    Set $V_{i+1} = MA_{speed}$
    Set $a_{i+1} = MAX_{acceleration}$
  end if
  if (P_i marked as filtered) **then**
    Set $V_i = (V_{i+1} - V_{i-1}) \times (t_i - t_{i-1}) / (t_{i+1} - t_{i-1}) + V_{i-1}$
    Mark P_i as interpolated
  end if
  Set i = i + 1
until Exist no more input of positioning tuple

<표 1>

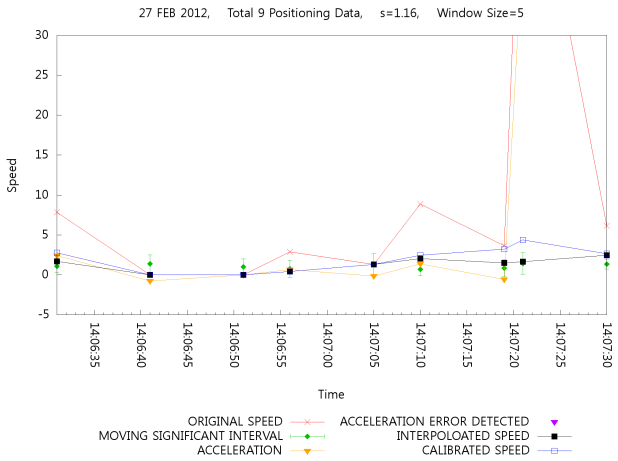
4. 실험

본 실험에서 우리는 2012년도에 서울(대한민국), San Francisco Bay Area(미국), Lake Ontario(캐나다)에서 Iphone3GS, Iphone 4의 위치 정보 수집 프로그램을 이용해 수집된 데이터를 사용하였다. 해당 프로그램은 사용자의 위치가 바뀔 때마다 해당 위치를 기록한다. 또한 위치가 변하지 않을 시에는 미리 지정한 시간간격에 따라 위치를 기록한다. 우리는 다양한 방법으로 해당 데이터를 시각화 할 수 있는데 이 중 Google Earth를 이용해 실제 지도에 데이터를 표시[2]하고 gnuplot을 이용, 결과를 그래프로 나타내었다.

우리는 가장 먼저 iOS의 위치 시스템의 정확성을 확인하기 위한 실험을 수행하였다. 3GS iphone 하나는 빌딩 내에서 움직임 없이 몇 시간 동안 데이터를 수집했는데, 놀랍게도 15553개의 데이터 중 단 16개만이 오류로 판명되었다. 그러나 움직이는 상황이나, 시외로 나간 상황 등에서는 더 많은 오류가 발견되었다.



<그림 1> 속도 교정이 없는 오류 필터링의 부분 진행 과정



<그림 2> 속도 교정이 있는 오류 필터링의 부분 진행 과정

우리는 직접 모은 수많은 위치 데이터들을 Google Earth를 통해 실제 지도에 시각화하였고[2], <그림 4>를 통해 2012년 2월 27일자 데이터를 시각화 한 모습을 볼 수 있다. 앞으로의 실험에서 우리는 s=1.16, window size=5로 인수를 지정하여 수행한다.

속도 교정(보간법)이 포함되지 않은 필터링 과정은 <그림 1>에 나타나 있고 보간법을 이용한 속도 교정이 있는 필터링 과정은 <그림 2>에서 볼 수 있다. 그림에서 ORIGINAL SPEED는 교정 전 속도, ACCELERATION은 가속도, MOVING SIGNIFICANT INTERVAL는 이동 유의 구간, INTERPOLATED SPEED는 선형 보간법에 의한 교정 속도, CALIBRATED SPEED는 교정 전 선 보정 속도, ACCELERATION ERROR DETECTED는 가속도에 의한 오류 데이터를 각각 나타낸다.

두 그림을 보면 오류로 판명된 데이터가 <그림 2>에서 더 많은 것을 알 수 있는데 이는 선형 보간법에 의해 속도 교정을 한 결과, 윈도우 내의 평균 속도 및 표준 편차의 값이 달라져 더 많은 오류를 찾아냄을 알 수 있다.(14:06:56, 14:07:30 데이터) 또한 14:06:30, 14:07:10, 14:07:21의 데이터를 살펴보면 이동 유의 구간을 벗어나는 속도는 오류로 판명, Calibration에 의해 속도가 효과적으로 감소하고, 그 후 보간법에 의해 성공적으로 교정이 됨을 확인할 수 있다. 이를 통해 단순히 속도의 오류를 필터링 하는 것보다 보간법에 의해 속도 교정을 하는 것이 윈도우의 역할을 더 성공적으로 수행하게 하고 더 나은 결과를 보여주는 것을 확인 할 수 있다.

<그림 3>은 37개의 데이터로 알고리즘 수행의 전체적인 효과를 육안으로 확인할 수 있다.

5. 결과분석 및 향후 연구 방향

이번 연구에서, 우리는 새로운 알고리즘 제안을 통해 몇 가지 목표를 달성하고자 하였다 : 첫 번째로는 이동 윈도우를 통해 이동의 경향을 필터링 결과에 반영하고자 한 것이고, 두 번째로는 선형 보간법을 이용한 속도 교정을 통해 좀 더 세련된 필터링 기법을 제시하는 것이다. 세 번째로는 과한 필터링, 또는 약한 필터링을 완화시키고자 앞에 설명한 Calibration 과정을 포함하는 것이다. 또한 이것은 연산 능력이 비교적 뒤떨어지는 모바일 기기에서 충분히 사용가능한 것이어야 한다.

미래에는 분명히 연산능력이 더 좋은 여러 모바일 기기들이 등장할 것이고 그에 따라 우리는 좀 더 세련된 알고리즘(연산이 복잡한 알고리즘)을 제시하는 것이 가능해진다. 예를 들면 보간법이 선형보간법이 아닌 비선형 보간법을 통해 더 현실에 가까운 속도 교정을 수행하는 것 등이 있겠다. 또한 속도 교정뿐만 아니라 위치(위도, 경도)값도 보간법 등을 통하여 교정할 수 있겠고, 위치 정보 자체도 위도, 경도만이 아닌 고도 정보도 사용함으로써 좀 더 정확한 속력도 도출해 낼 수 있을 것이다.

사사(謝辭)

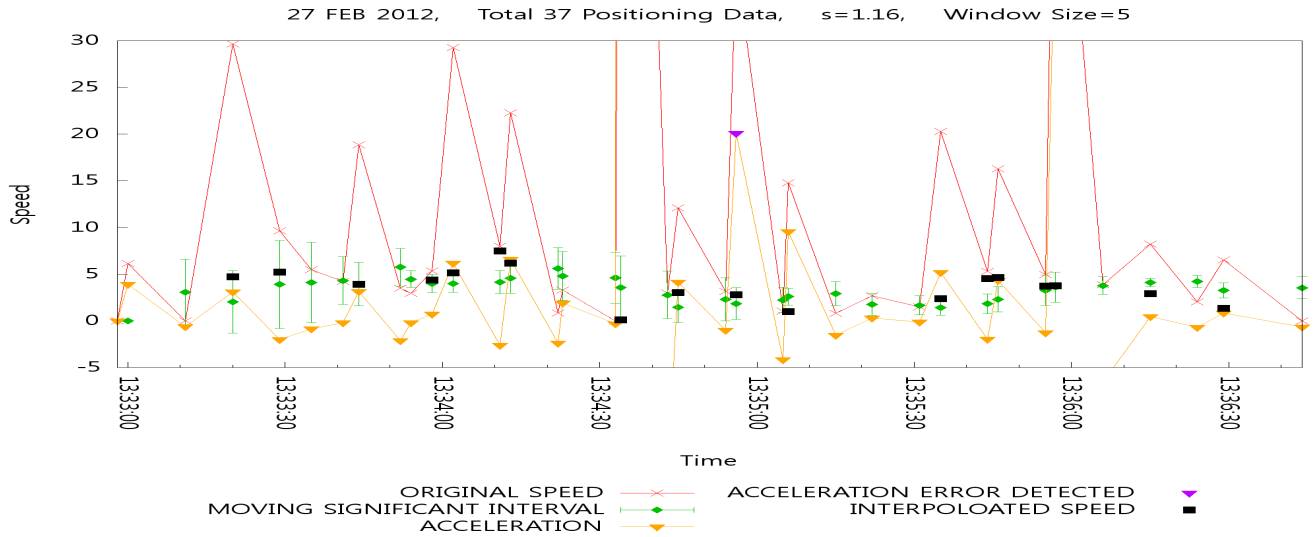
이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2012046473 및 No. 20120007162).

6. 참고문헌

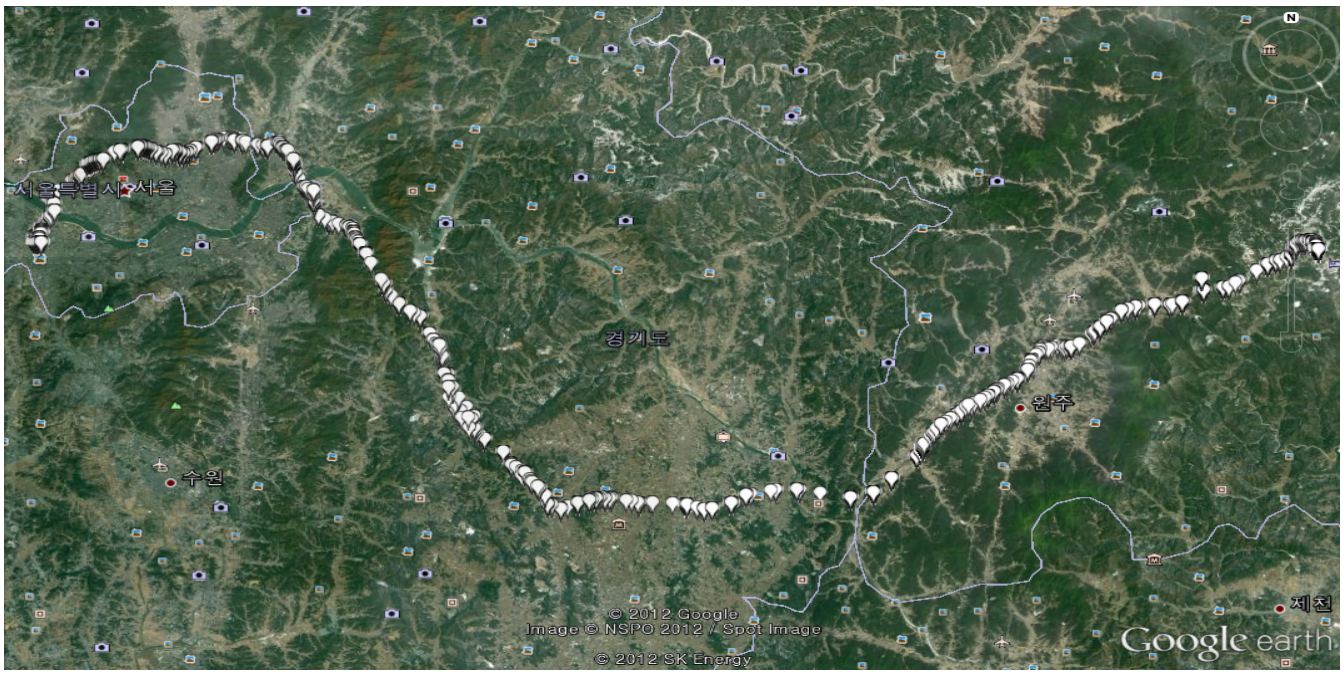
[1] T. Vincenty, ”"Direct and Inverse Solutions of Geodesics on the ellipsoid with Application of Nested Equations,"" Survey Review, Volume 23 Number 176, April 1975 , pp. 88-93(6).

[2] Google Earth, Available: http://www.google.com/earth/index.html

[3] List of fastest production cars by acceleration, Available: http://en.wikipedia.org/wiki/LisP_of_fastesP_cars_by_acceleration



<그림 3> 오류 필터링 및 속도 교정의 전반적 과정



<그림 4> 실제 지도에 이동 경로를 시각화한 모습