A Realtime Filtering Method of Positioning Data with Moving Window Mechanism

Ha Yoon Song and Han-gyoo Kim

***Abstract*— Nowadays, advanced mobile devices can obtain current position with the help of positioning data systems such as GPS, GLONASS, Galileo, and so on. In addition cellular network positioning with base station locations or crowd source WIFI positioning approaches are available. However, positioning data sets usually have erroneous data for various reasons, mainly due to the environmental issues as well as inherent systematical issues. While doing research related to positioning data sets, authors experienced quite a large number of erroneous positioning data using Apple iPhone or Samsung Galaxy devices, and thus need to filter evident errors. In this paper, we will suggest relatively simple, but efficient filtering method based on statistical approach. From the user's mobile positioning data in a form of $< latitude, longitude, time >$ obtained by mobile devices, we can calculate user's speed and acceleration. From the idea of sliding window (moving window), we can calculate statistical parameters from speed and acceleration of user position data and thus filtering can be made with controllable parameters. We expect that the simplicity of our algorithm can be applied on portable mobile device with low computation power. For the possible enhancement of our method, we will focus on the construction of more precise window for better filtering. A backtracking interpolation was added in order to replace erroneous data with proper estimations in order to have more precise estimation of moving window. We also proposed this filtering algorithm with interpolation as a basis of future investigation in the section of conclusion and future research.**

***Keywords*— Human Mobility, Error Filtering, Positioning Data, Moving Window, Sliding Window**

I. Introduction

The recent advances of mobile devices enable various location based services over human mobility, especially the introduction of smart phone with GPS or other positioning equipment. The application of positioning system can be easily found in various mobile devices as shown in [1] including educational field as shown in [2]. However these positioning data sometimes have position errors according to the operational environment. In such cases, many of applications require filtering of such erroneous positioning data. As we experienced by our experiments, more than 12% of positioning data were erroneous by use of smart phones. This basic experiment was done by use of smart phone app over Samsung Galaxy Tab which internally uses the position of cellular base station, over portable GPS device [3], and Apple iPhone 3GS with iOS5 which uses combination of crowd sourced WIFI positioning, cellular networks, and GPS [4]. More precise result can be found in Kim and Song [5] along with researches on human mobility model. Another research field of complex system physics showed that up to 93% of human mobility can be predicted since peoples avoid the random selection of next destination instead selects their place frequented and their route frequented [6]. The sets of positioning data will be a basis for human mobility model construction as shown in [5]. In this paper, we will propose a filtering technique which filters erroneous positioning data with the use of moving window approach. Section II shows our idea using moving window with pre-experiments for algorithm set-up. Section III shows filtering algorithm and its detailed description. Section IV shows our consideration on user controllable parameters for experiment design and shows our experimental results. We will conclude and discuss about our future research in section V.

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0025875).

II. Backgrounds

A. Idea on Moving Window

Collected user position in a form of $< latitude, longitude, time >$ composes a set of user mobile trace and adding an identification parameter to the tuple will represent user's mobility data set. We call one tuple at time t as P_t, and latitude of P_t as lat_t, longitude of P_t as lon_t. From two consecutive position data tuples, we can calculate the distance D_i moved at time P_i with $< lat_i, lon_i >$ and $< lat_{i-1}, lon_{i-1} >$ according Vincenty's formula [7]. Of course, from the two consecutive distance, we can calculate speed at time of P_i, V_i, and acceleration at time of P_i, a_i. Therefore a tuple P_t has a core form of $< t, lat_t, lon_t, D_t, V_t, a_t >$ with possible auxiliary attributes.

Based on the speed values on actual position data set, we find a glitch on a series of speed such as $600m/sec$, which are meaningless for usual lifetime environment. Thus we investigated maximum possible speed values of usual human mobility as shown in table I. We define max speed MAX_{speed} as $250m/sec$. In addition, those maximum values cannot be reached instantly, i.e. the acceleration cannot be made abruptly. In addition to the MAX_{speed}, $MAX_{acceleration}$ can be defined.

For the next step we introduced the idea of moving average and moving standard deviation of speed. We define the moving average of speed at current time t, $MA_{speed}(n)$ where n stands for the number of past data of $\{P_x : t - n + 1 \leq x \leq t\}$. Similarly, we can define the moving standard deviation at time t, $MSD_{speed}(n)$ where n stands for the number of past data. Here, n is referred as window size by most of researchers. Once we obtain a new tuple P_t, then we can determine whether V_t is in the usual range of human mobility, and for the a_t, the same can be applicable. Once V_t is out of range for the normal distribution with average of $MA_{speed}(n)$

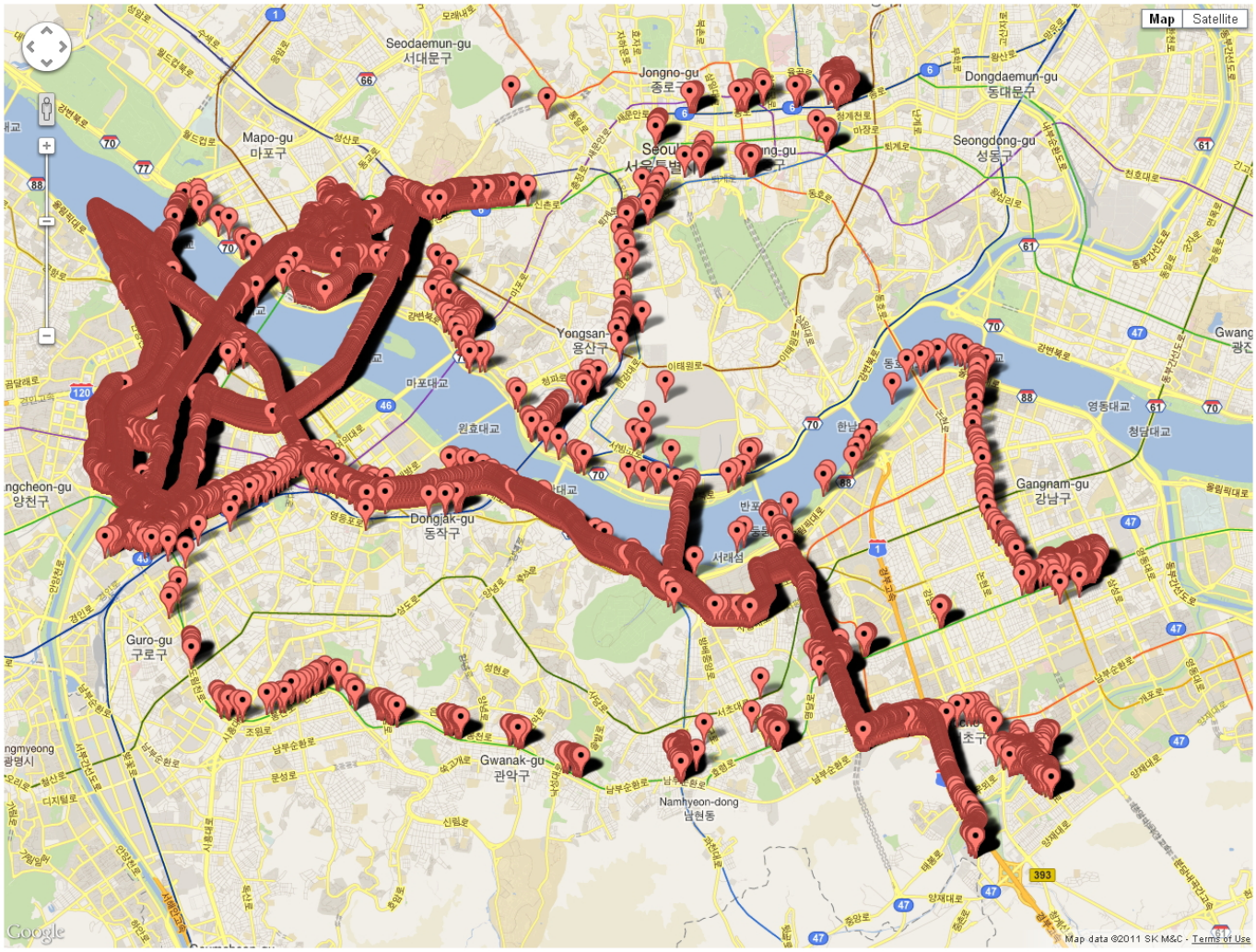

Fig. 1. A Trail of Positioning Data Set

TABLE I
MAXIMUM SPEED OF TRANSPORTATION METHODS

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

and standard deviation of $MSD_{speed}(n)$ we can discard P_t and this tuple will be filtered out from the series of human trace. The condition of filtering out P_t is:

$$V_t > MA_{speed}(n) + s \times MSD_{speed}(n) \quad (1)$$

where s stands for the sensitivity level of filtering and it is user controllable parameters. Otherwise, we can include the tuple P_t to the series as a valid positioning data, and thus can recalculate $MA_{speed}(n)$ and $MSD_{speed}(n)$. Note that this calculation can be made in real-time, and we intentionally introduced this approach because it requires relatively simple calculation. In other words, this algorithm can be executed on a device with low computing power such as smart phones or similar mobile devices. Our previous research includes similar work with more complicated statistical theory [8] however it is not likely to be introduced for real time environment. Other examples of moving window based applications can be found in [9] and [10].

B. Pre-experiments on Window Size

We will conduct experiments to see the effect of window size. For this experiment, we used a set of positioning data over the area of Seoul, Korea collected during more than 20 days. Note that this data is voluntarily collected by the author

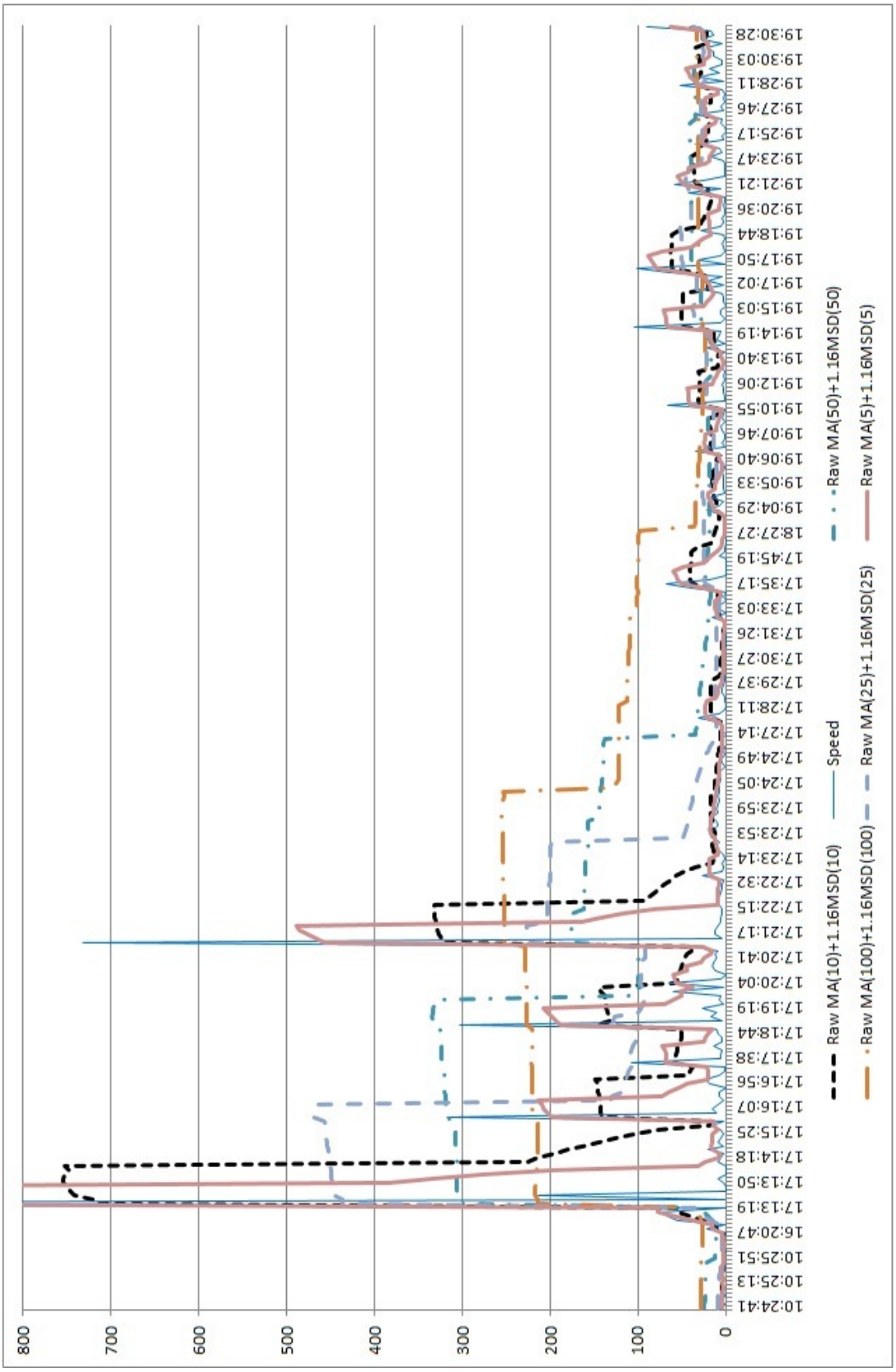

Fig. 2. Effects of Different Window Size

of this paper using iPhone 3GS with positioning data collection app. We will call it iPhone data set. The app records position data whenever it senses the location change of iPhone or for every user-defined interval (3 to 60 seconds) if the iPhone is in immobile state. We can draw the positioning data set on geographical map with various techniques. Among the various techniques, we choose Google map [11] for visualization of positioning data set. The visualization of raw data sets is shown in figure 1 and contains erroneous data. One of the notable phenomenon is that iOS5 sometimes give simultaneous report of position data from three different schemes. Our guess is that cellular base station locations, crowd source WIFI positioning, and GPS sometimes reports different positioning data at the same time. In case we meet multiple position values at the same time, the position data with smallest speed value was the correct one empirically. Therefore the first stage of filtering is trivially to choose the position data with the smallest distance to previous position among multiple position data of the same time. In addition, data sets are composed of several set of discontinued data, for example, position data collection was unable on the subway train or there was no need to collect data in the home bed. While traveling by subway a train, only at the stations was it possible to collect positioning data.

In order to determine n, we must be more considerate. With varying size of n with the algorithm over data set, we made several set of experiments. We can expect that large windows size cannot react to the situation of rapid speed change while it is useful in stable, immobile states, successfully coping with continuous errors. On the contrary small n will show rapid reaction to abrupt speed change for mobile states which maybe is considered to include continuous error tuples. For example, once we met m continuous errors we cannot filter out such errors if $m > n$. In order to figure out our conjecture, we make experiments on window size upon the iPhone data set. We calculated moving average and moving standard deviation over various window sizes such as 5, 10, 25, 50, and 100. Figure 2 shows the result of our simple experiment. It is clear that larger window size is dull. Once we met very large speed value, large window size tails the effect of large error speed thus leads to under-filtering of erroneous data. In case of window size with 100 or 50, we can clearly see the tailing effect on figure 2. On the contrary, small window size is sensitive and rapidly reacts on speed change while it over-filters correct data especially at the starting phase of speed change. We experienced two or more plausible tuples were discarded by the small window size while they look like correct speed data with window size 5 or 10. This phenomenon implies that we must introduce the throttling mechanism to moving window in order to avoid tailing effect of window size.

C. Pre-experiments on Positioning Data Error

We also conducted a basic test as our base experiment to check the positioning data accuracy as we mentioned in section I. We fix positioning devices both outside area and inside the building, and collected positioning data for several hours without moving any device. The first positioning device is Garmin GPSMAP62s [3] for pure GPS data collecting. The second positioning device is Samsung Galaxy Tab to obtain positioning data from its connected 3G base stations (3GBS). We guess Galaxy tab will show more error in both situation, and both of the data set from GPS and 3GBS shows positioning error, especially inside the building. The result of this basic experiment is listed in table II.

The variance in position data is regarded as errors and distance of error can be calculated form the position data. As we guessed, 3GBS shows larger error rate, larger error in distance, larger maximum error distance, and bigger standard deviation in error distance. Due to the producer's policy of Garmin GPSMAP62s which estimates the user's location upon past velocity while it lost the GPS signal, it shows drastic error value inside the building. Thus we think the GPS inside a building cannot be a meaningful data. GPS data from outside area is very accurate enough for precise localization and even the maximum error distance is in a reasonable range of 52 meters.

III. Filtering Algorithm

According to the considerations in section II, we build an algorithm for erroneous positioning data filtering as shown in algorithm 1. With the new acquisition of new position data P_{i+1}, this algorithm can determine whether P_{i+1} be filtered out or not. In practice, we must consider several situations:

- Initial construction of Moving Window: in case there are less than n tuples, we cannot construct complete moving average and moving standard deviation. Instead, we have to have incomplete window with less number of data: lines 2 to 5 in algorithm 1.
- Both acceleration and speed are considered as throttling parameters even though they have difference in filtering details. Positioning data tuple with out of range speed or with unreasonable acceleration will be filtered out: lines 6 to 8 in algorithm 1.
- Once the speed of a tuple is too big, which can affect the MA and MSD values and leads to filtering error as expansion of confidence interval which leads to erroneous filter-in of to-be-filtered tuple, we will calibrate the speed value with $MA_{speed}(n) + 2.57 \times MSD_{speed}$ in order to include possibly rapid change of speed and to avoid erroneous expansion of confidence interval: lines 9 to 11 in algorithm 1. The value s = 2.57 stands for 99.5% confidence interval of normal distribution. This is throttling which reduces the effect of erroneous speed to moving window: $s_{99.5}$ in lines 9 and 10 of algorithm 1.
- Window Construction: Even though a tuple be filtered out, we will include the speed of the tuple unless it is out of 99.5% confidence interval of normal distribution. It is intended to update moving window in order to cope with rapid change of speed, i.e. a tuple can be filtered out with rapid change of speed while it is a genuine one. In such cases, even though the tuple was filtered, moving window can reflect the change of speed for the next incoming tuples: line 10 in algorithm 1.
- Small speed less than $2.77m/s$ $(10Km/h)$ will not be filtered since it is always possible for a human ambulation

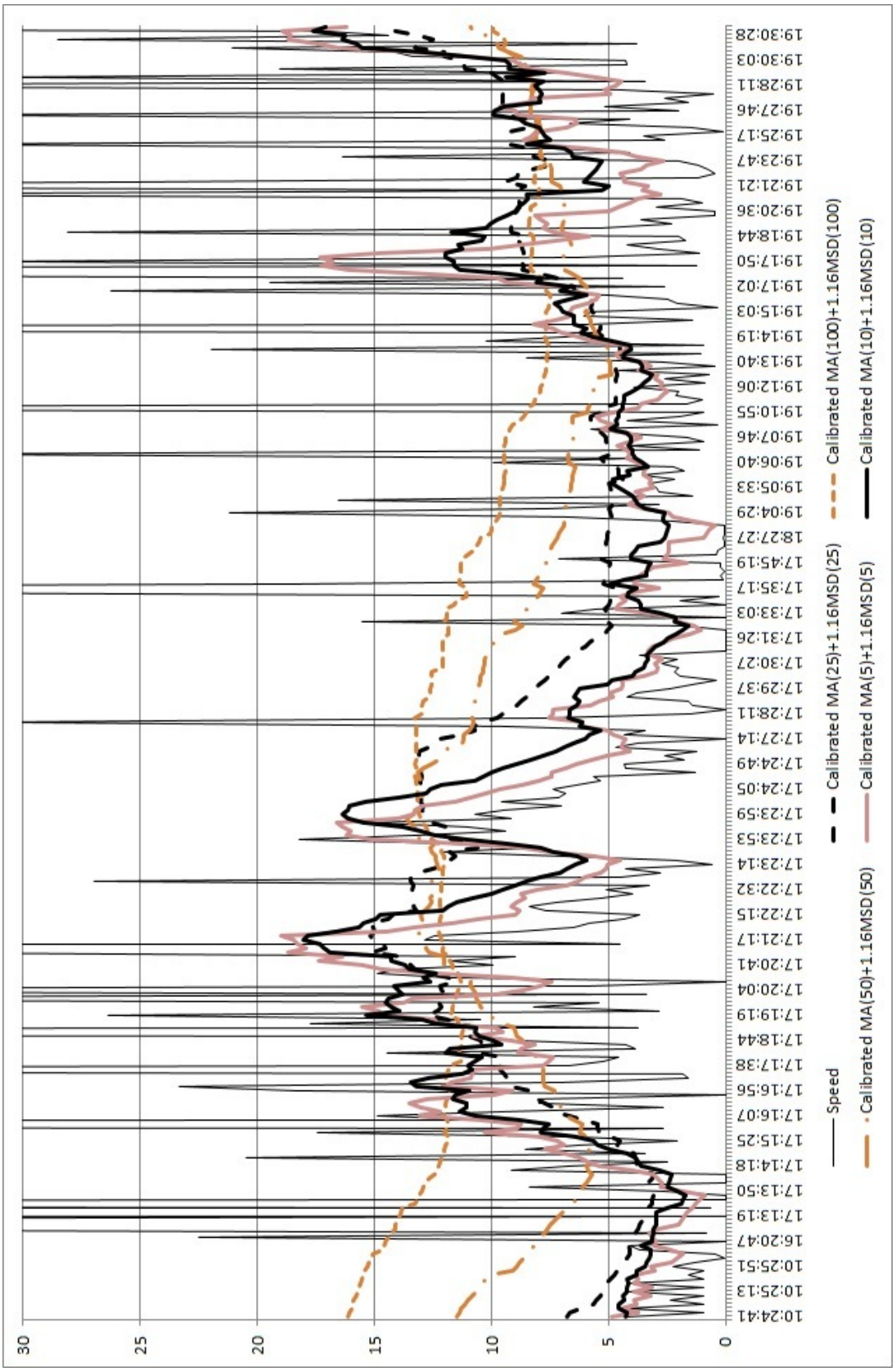

Fig. 3. Effect of Window Size with s=1.16 and Calibrated Speeds

TABLE II
TYPICAL ERRORS IN POSITIONING DATA ACQUISITION (UNIT: METER)

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

within this small speed, and it is also in GPS error range: $MIN_{velocity}$ line 9 in algorithm 1.

- Tuples with unreal acceleration must be filtered. As reported in [12], $10.8m/s^2$ is the current biggest value for sport-cars: $MAX_{acceleration}$ as line 12 in algorithm 1.
- Once a tuple be filtered out due to excessive acceleration, the tuple must be filtered, the acceleration value of the tuple is forced to set as $MAX_{acceleration}$, and the speed value is forced to set as $MA_{speed}(n)$ to nullify the effect of unreal speed value: lines 12 to 16 in algorithm 1.
- Tuples with positive acceleration values more than $MAX_{acceleration}$ will be regarded as errors while positioning data tuples with negative acceleration values will not be regarded as errors since it is always possible for a vehicle to stop emergently with large negative acceleration.

IV. EXPERIMENTS

A. Experiment Design

For the algorithm from section III, we left two parameters unspecified: n as a number of tuples in the window and s as a sensitivity level of filtering. Both of the parameters can be specified by user of this algorithm. The user sensitivity level s has relatively simple to determine. From the properties of normal distribution, we can obtain s with proper confidence interval. Since we use only the positive part of normal distribution for filtering, we can set $s = 1.64$ for confidence interval of 95% and $s = 2.33$ for confidence level of 99%. Users can determine s as their own purpose. For example, we choose the sensitivity level s as follows. Table II shows a typical error rate for position data collecting when the device is immobile. For GPS case, 12.3% data was erroneous and for 3GBS cellular positioning system, 36.75% of data has errors. So we can choose $s = 1.16$ for GPS data or $s = 0.34$ for cellular positioning data in our experiment.

Algorithm 1 Moving Window Filtering at real-time t

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** Get P_{i+1} ▷ Acquisition of new tuple, if exist
2: Construct $MA_{speed}(n)$ with $\{P_x : max(i-n+1, 0) \leq x \leq i\}$
3: Construct $MSD_{speed}(n)$ with $\{P_x : max(i - n + 1, 0) \leq x \leq i\}$
4: Set $MA_{speed} = MA_{speed}(n)$
5: Set $MSD_{speed} = MSD_{speed}(n)$ ▷ Moving Window Construction
6: **if** $(V_{i+1} > MA_{speed} + s \times MSD_{speed})$ OR $(a_{i+1} \geq MAX_{acceleration})$ **then** ▷ Filtering
7: Mark P_{i+1} as filtered.
8: **end if**
9: **if** $(V_{i+1} \geq MA_{speed} + s_{99.5} \times MSD_{speed})$ AND $(V_{i+1} > MIN_{velocity})$ **then** ▷ Calibration of Speed
10: Set $V_{i+1} = MA_{speed} + s_{99.5} \times MSD_{speed}$
11: **end if**
12: **if** $a_{i+1} \geq MAX_{acceleration}$ **then** ▷ Restriction by Maximum Acceleration
13: Mark P_{i+1} as filtered
14: Set $V_{i+1} = MA_{speed}$
15: Set $a_{i+1} = MAX_{acceleration}$
16: **end if**
17: Set i = i + 1
18: **until** Exist no more input of positioning tuple

B. Reconsideration of Window Size

We already discussed the effect of window size in section II. Due to the trailing effect of large window, we may choose smaller window. However, algorithm 1 calibrates incorrect speed values. As well, an abnormal acceleration value is restricted and the speed value will be replaced by average speed of moving window. With these calibration mechanisms, we need to see the effect of window size again. Figure 3 shows the effect of window size under our calibration mechanism. The x-axis stands for wall clock time on 11th of November, 2011. We choose window size $n = 5, 10, 25, 50, 100$ and sensitivity level $s = 1.16$ which stands that 88% of positioning data are regarded as correct ones. Even though the trailing effect is restricted, smaller windows show more flexible reaction according to speed change. Thus window size of 5 or 10 is

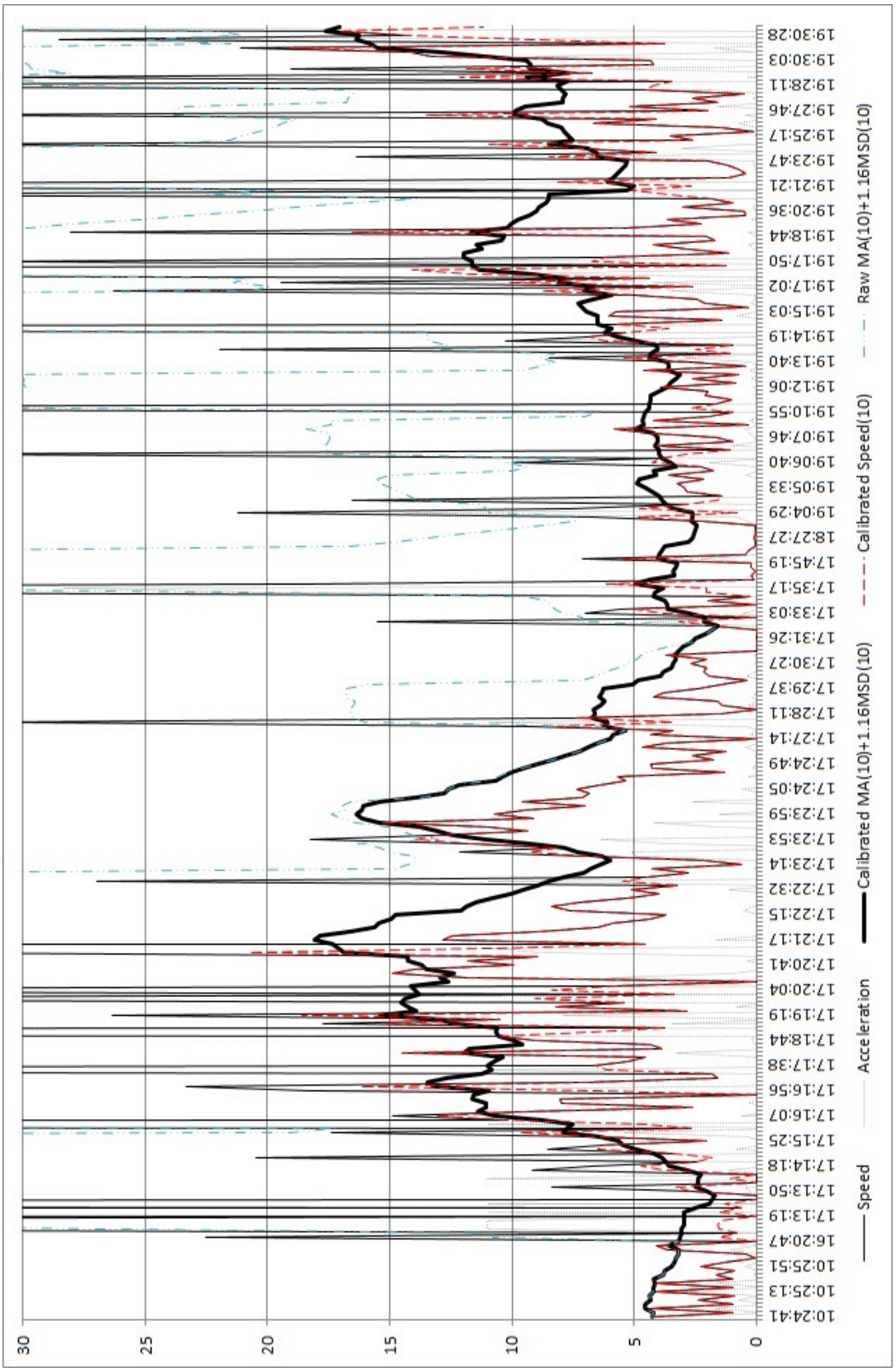

Fig. 4. Filtering Investigation with s=1.16 and n=10

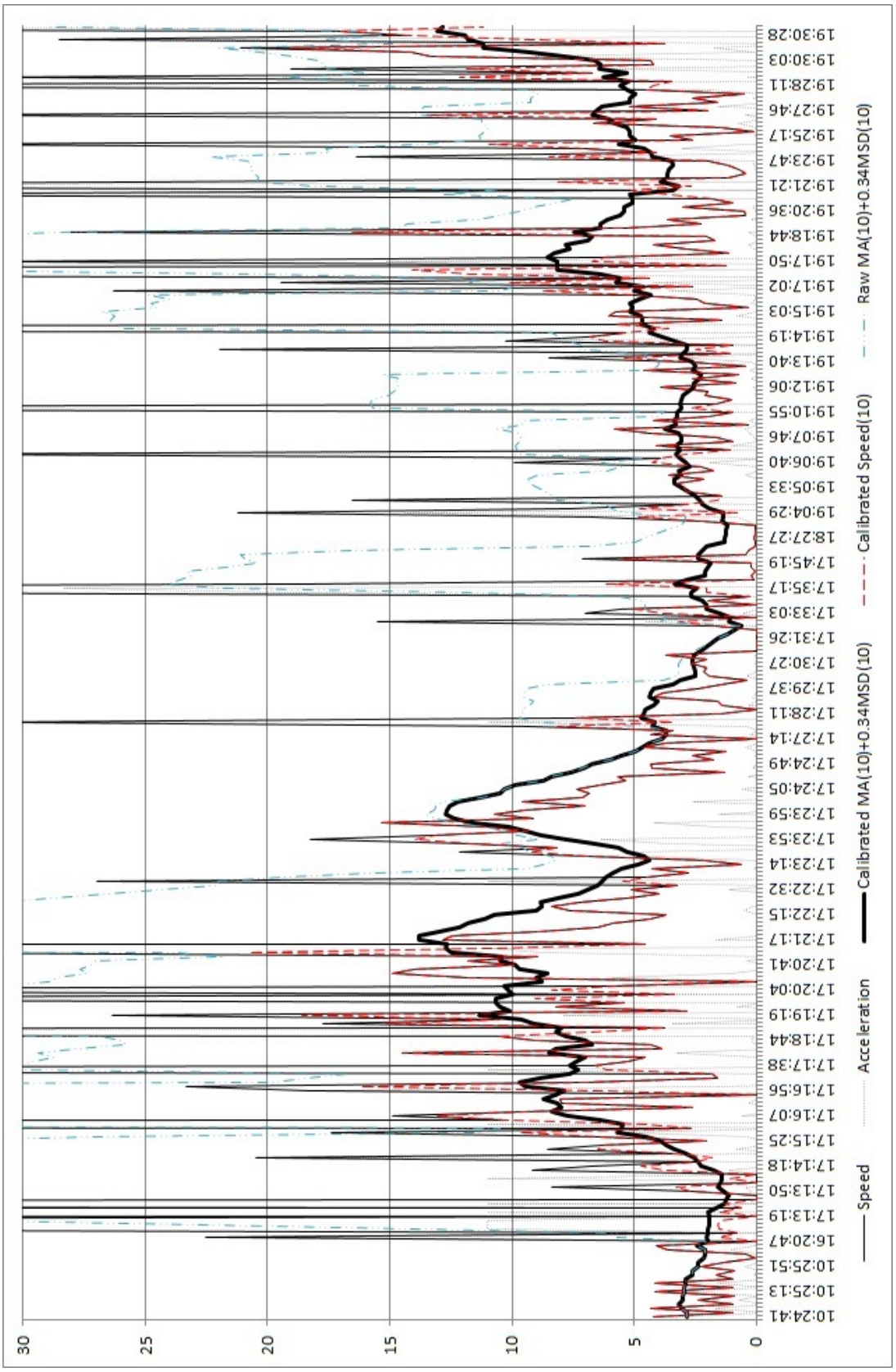

Fig. 5. Filtering Investigation with s=0.34 and n=10

better choice.

One more consideration is effect of consecutive errors. Even if we have throttling mechanism of speed, consecutive errors will have to effect on the moving average and moving standard deviation, and thus leads confusion to our filtering algorithm. We experienced up to four consecutive errors in real positioning data set and thus concluded that $n = 5$ is deficient for our experimental environment while $n = 10$ will cover consecutive errors and will reduce the tailing effect of larger window size. Thus our final choice of window size for our main experiment is 10. Of course, once we experienced more number of consecutive errors, we choose proper window size or dynamically increase the size of window as we will see in section V

Figure 4 shows the progression of filtering algorithm. The same positioning data set as in figure 3 was chosen for fair combination. The x-axis stands for wall clock time on 11th of November, 2011. For figure 4, window size is $n = 10$ and sensitivity level $s = 1.16$ which stands that 88% of positioning data are regarded as correct ones. Thin black solid line shows the change of real speed in m/s and thin gray dashed line shows calibrated speed by our filtering algorithm. Calibrated speed usually overlapped with speed while it shows calibration once a tuple be filtered by filtering algorithm. Dotted line shows the values of acceleration.

Thick black solid line denotes the coverage (acceptance range of speed) of moving window with n=10. Note that the moving window shown in the figure is based on calibrated speed values. It reacts rapidly with the change of speed while successfully filters erroneous tuples. Double dotted line shows the coverage of moving windows without speed calibration (raw coverage). Comparing calibrated coverage and raw coverage, the effect of speed calibration or limitation is clear. Speed calibration in our algorithm successfully suppresses the trail of moving window due to gigantic speed errors. Thus moving window composed of calibrated speeds successfully eliminates the effect of speed errors and keeps proper estimation of positioning tuple values.

Figure 5 shows the progression of filtering algorithm similar to figure 4. For figure 5, sensitivity level $s = 0.34$ and every other conditions is the same.

C. Filtering Results

For the final representation of our filtering experiment, we will use two kinds of representation.

First, we conduct the filtering over our whole positioning data set and represent the filtering result on real map. The visualization is also done with Google maps [11]. Figure 6 shows the filtering result of $n = 10$ and $s = 0.34$. Figure 7 shows the filtering result of 88% confidence interval for $n = 10$. The collector of positioning data set cannot find errors in this figure while it contains more data for positioning than the result shown in figure 6. Therefore, we can conclude that proper selection of window size and sensitivity level will leads to adequate filtering results.

Second, we conduct the filtering on the combinations of window size and sensitivity level over the whole positioning data set. Table III shows the percentage of filtered-out tuples in each combination of parameters. Users of our algorithm may choose windows size and sensitivity level according to table III for their own environment.

V. Conclusion and Future Work

In this research we build algorithm for erroneous position data filtering and experienced the combinational effect of window size and sensitivity level. A real set of positioning data collected by author is used for algorithm verification and we found successful filtering results in general. Several parameters of the algorithm must be defined by user such as window size, sensitivity level, maximum speed and maximum acceleration. Even it is possible for a user can change constant parameters of the algorithm such as $MAX_{acceleration}$, $s_{99.5}$ (maximum sensitivity level) and $MIN_{velocity}$ (minimum threshold of speed for filtering) of the algorithm.

While investigating filtering process one by one, we find several minute frailties of our algorithm. The first one is that out algorithm cannot work at the starting phase of data collection because at the stage of initial window construction there were not enough tuples to fill the whole window. We think it is a compulsory demerit for every approach based on moving window.

The second problem is a tendency of over-filtering and under-filtering. The arrival of new tuple with rapid increase of speed will be filtered out regardless of its correctness. This tendency is clear when we have a large window size since large window cannot react fast enough to catch the rapid change of speed. In compensation, we include the velocity of filtered tuple to preserve the coverage of window in order to cope with speed change unless the change of speed is out of 99% of confidence interval of normal distribution. In case the velocity is out of 99% confidence interval, we calibrate the velocity for the future construction of moving window. As we noticed in figure 3 the tendency of under-filtering is clear with larger window size.

With smaller windows we cannot filter if number of consecutive errors is bigger than window size. As we experienced four consecutive errors in our data, we think n=10 is better choice than n=5. Another benefit of small window size is a small computation time and small memory capacity for moving window construction so that this algorithm can work on mobile devices with low computational power in real time.

We must express several sorts of further considerations. The first one is consideration on windows size. We can express window size as time duration instead of number of tuples in a window. This will be effective once we regularly collect position data and somewhat accurate since speed is a function of time. Another thought on window size is dynamic calibration of windows size. We can increase or decrease the window size dynamically according to the number of consecutive errors. Once we found larger number of consecutive errors we can increase window size in order to minimize the effect of consecutive errors to moving average and moving standard deviation. If we have smaller number of consecutive errors, we can decrease the window size so that we achieve more

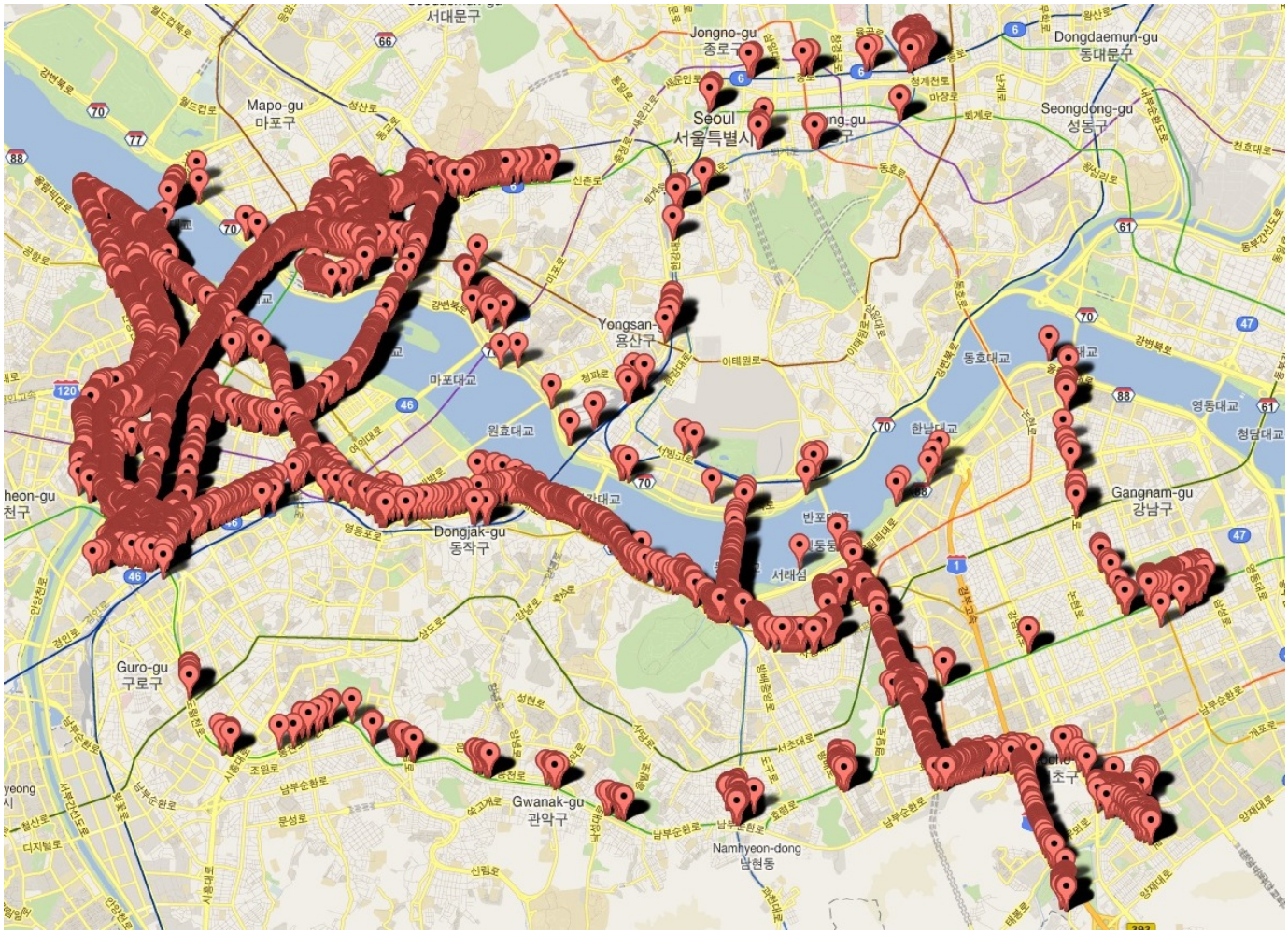

Fig. 6. Trail of Filtered Positioning Data with n=10 and s=0.34 (63% confidence interval)

TABLE III
RATIO (%) OF FILTERED-OUT TUPLES WITH RESPECT TO SENSITIVITY LEVEL

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

proactive reaction of rapid speed change and less computation for filtering.

The second one is pseudo real time algorithm rather than real time one as algorithm 1. For a window size n, we can decide filtering of $\lceil n/2 \rceil$-th tuple in a window instead of filtering the new incoming tuple with the existing window including n tuple. Filtering $\lceil n/2 \rceil$-th tuple in a window stands for the filtering will be made in the middle of window rather than at the very end of window. Even though it cannot be used in real time, this approach can reduce the tendency of under-filtering and over-filtering.

Another idea of algorithm enhancement is an introduction of interpolation. Algorithm 2 has extra operation for interpolation rather than algorithm 1. Here, we add extra stages to the algorithm for better approximation of moving window statistics. Upon the arrival of new tuple, we would replace the velocity of the last tuple in existing window with linearly interpolated value once the last tuple in the window found marked. This interpolation will give more precise approximation of moving window for filtering purpose: lines 17 to 20 in algorithm 2. In other words, we can interpolate the marked last tuple in a window whenever a new tuple is obtained by the latest part of algorithm 2. Another variance of moving window construction with more precise approximation is to interpolate $\lceil n/2 \rceil$-th tuple with n tuples in a window. For better estimation, interpolating the middle tuple in a window using asymptotic curve estimated from n tuples will enable more precise interpolation. However it could introduce computational overhead so that the

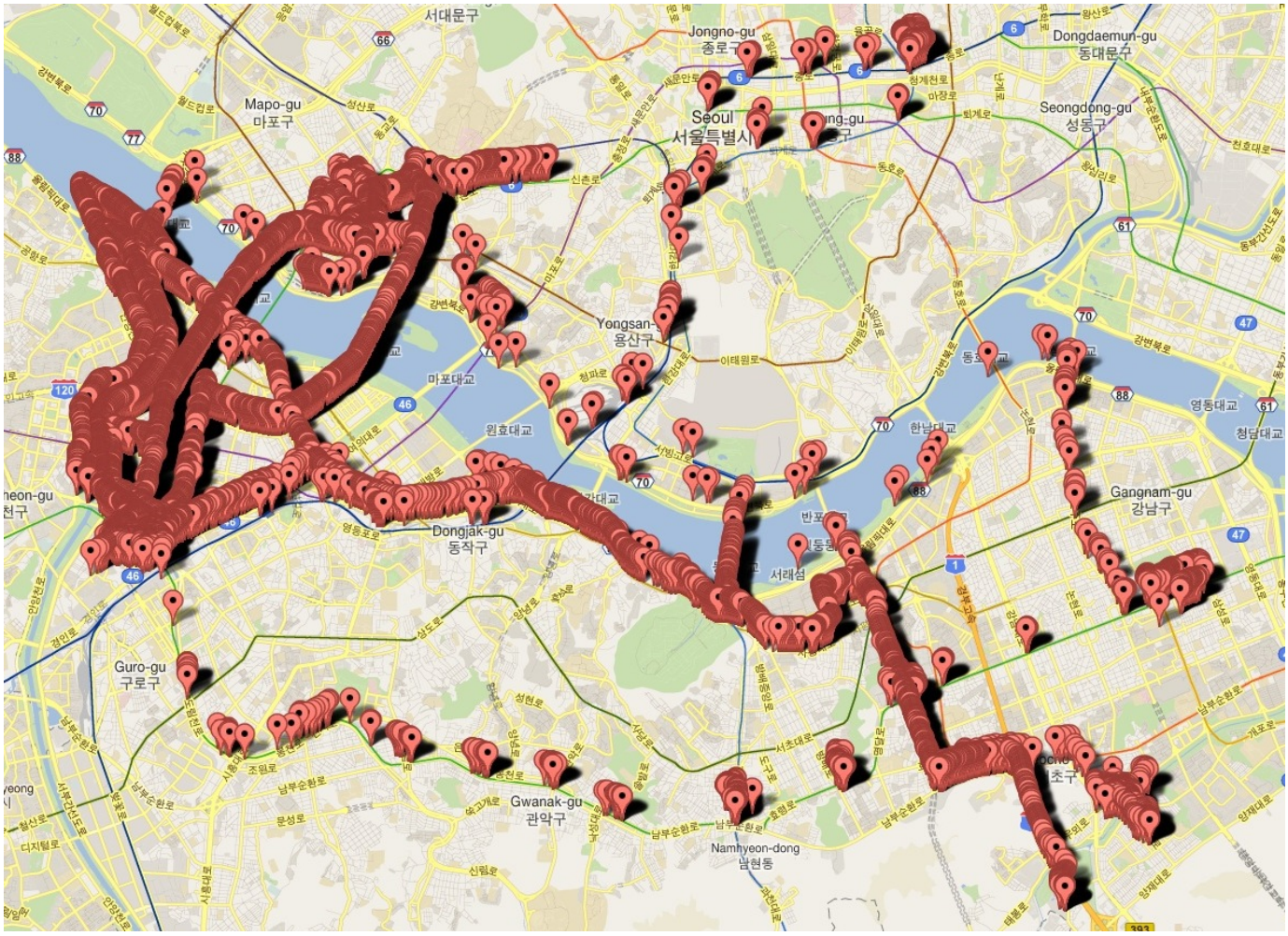

Fig. 7. Trail of Filtered Positioning Data with n=10 and s=1.16 (88% confidence interval)

application of the algorithm to mobile device is difficult. We consider these enhancements of filtering algorithm as our next research. We will see the effect of speed interpolation to the filtering accuracy.

Finally, we need to investigate the effect of probability distribution for filtering. In general, normal distribution is a usual candidate for various sources of errors and filtering. However, a research showed that human mobility pattern is in a heavy tailed distribution such as Levy Walk [6]. We will therefore see the effect of Levy Walk for filtering since the distribution of positioning data will be likely to be in a Levy Walk form.

REFERENCES

[1] Nicolae-Iulian Enescu, Dan Mancas, and Ecaterina-Irina Manole, "Locating GPS Coordinates on PDA," *8th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC08)*, Rhodes, Greece, August 20-22, 2008, pp.470-474.

[2] Ph. Ddondon, T. TSING, and F. Sandoval, "Initiation to GPS localization and navigation using a small-scale model electric car: An illustration of learning by project for graduated students," *Proceedings of the 8th WSEAS International Conference on EDUCATION and EDUCATIONAL TECHNOLOGY*, 2009, pp.21-26.

[3] Garmin GPSMAP62s, Available: `https://buy.garmin.com/shop/shop.do?pID=63801`

[4] iOS 5: Understanding Location Services, Available: `http://support.apple.com/kb/ht4995`

[5] Hyunuk Kim and Ha Yoon Song, "Daily Life Mobility of A Student: From Position Data to Human Mobility Model through Expectation Maximization Clustering," *Communications in Computer and Information Science*, Volume 263, December 2011, pp. 88-97.

[6] Marta C. Gonzalez and A. Hidalgo and Albert-Laszlo Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, pp.779-782, 5 June 2008.

[7] 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).

[8] Woojoong Kim and Ha Yoon Song, "Optimization Conditions of OCSVM for Erroneous GPS Data Filtering," *Communications in Computer and Information Science*, Volume 263, December 2011, pp. 62-70.

[9] Camelia Ucenic and Atsalakis George, "A Neuro-fuzzy Approach to Forecast the Electricity Demand," *Proceedings of the 2006 IASME/WSEAS International Conference on Energy & Environmental Systems*, Chalkida, Greece, May 8-10, 2006, pp.299-304.

[10] Wiphada Wettayaprasit, Nasith Laosen, and Salinla Chevakidagarn, "Data Filtering Technique for Neural Networks Forecasting," *Proceedings of the 7th WSEAS International Conference on Simulation, Modelling and Optimization, Beijing*, China, September 15-17, 2007, pp.225-230.

[11] Google Maps API, Available: `https://developers.google.com/maps/`

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

Algorithm 2 Moving Window construction with Interpolation

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** Get P_{i+1} ▷ Acquisition of new tuple, if exist
2: Construct $MA_{speed}(n)$ with $\{P_x : max(i-n+1, 0) \leq x \leq i\}$
3: Construct $MSD_{speed}(n)$ with $\{P_x : max(i-n+1, 0) \leq x \leq i\}$
4: Set $MA_{speed} = MA_{speed}(n)$
5: Set $MSD_{speed} = MSD_{speed}(n)$ ▷ Moving Window Construction
6: **if** $(V_{i+1} > MA_{speed} + s \times MSD_{speed})$ OR $(a_{i+1} \geq MAX_{acceleration})$ **then** ▷ Filtering
7: Mark P_{i+1} as filtered.
8: **end if**
9: **if** $(V_{i+1} \geq MA_{speed} + s_{99.5} \times MSD_{speed})$ AND $(V_{i+1} > MIN_{velocity})$ **then** ▷ Calibration of Speed
10: Set $V_{i+1} = MA_{speed} + s_{99.5} \times MSD_{speed}$
11: **end if**
12: **if** $a_{i+1} \geq MAX_{acceleration}$ **then** ▷ Restriction by Maximum Acceleration
13: Mark P_{i+1} as filtered
14: Set $V_{i+1} = MA_{speed}$
15: Set $a_{i+1} = MAX_{acceleration}$
16: **end if**
17: **if** (P_i marked as filtered) **then** ▷ Linear Interpolation
18: Set $V_i = \frac{(V_{i+1}-V_{i-1})\times(t_i-t_{i-1})}{t_{i+1}-t_{i-1}} + V_{i-1}$
19: Mark P_i as interpolated
20: **end if**
21: Set i = i + 1
22: **until** Exist no more input of positioning tuple

Ha Yoon Song received his B.S. degree in Computer Science and Statistics in 1991 and received his M.S. degree in Computer Science in 1993, both from Seoul National University, Seoul, Korea. He received Ph.D. degree in Computer Science from University of California at Los Angeles, USA in 2001. From 2001 he has worked at Department of Computer Engineering, Hongik University, Seoul, Korea and is now associate professor. In his sabbatical year 2009, he worked at Institute of Computer Technology, Vienna University of Technology, Austria as a visiting scholar. Prof. Song's research interests are in the areas of mobile computing, performance analysis, complex system simulation and human mobility modeling.

Han-gyoo Kim is now Associate Professor in Department of Computer Engineering, Hongik University, Seoul, Korea 121-791 from August 1994 to present..

He received Ph.D in Computer Science from Computer Science Department, University of California at Berkeley, USA. In February 1981, he received B.S degree from Mechanical Engineering Department, Seoul National University.

Tel: +1-82-2-320-1469
E-mail: hgkim@hongik.ac.kr

Major Areas of Interests : network system, storage system