Available online at www.sciencedirect.com
ScienceDirect
Computers, Environment and Urban Systems
30 (2006) 757–772
Computers, Environment and Urban Systems
www.elsevier.com/locate/compenvurbsys

GPS accuracy estimation using map matching techniques: Applied to vehicle positioning and odometer calibration

George Taylor [a,*], Chris Brunsdon [b], Jing Li [a], Andrew Olden [a], Dörte Steup [a], Marylin Winter [a]

[a] *GIS Research Centre, School of Computing, University of Glamorgan, Glamorgan CF37 1DL, Wales, United Kingdom*
[b] *Department of Geography, University of Leicester, Leicester LE1 7RH, United Kingdom*

Received 15 July 2004; accepted in revised form 15 July 2005

Abstract

A test-bed application, called Map Matched GPS (MMGPS) processes raw GPS output data, from RINEX files, or GPS derived coordinates. This developed method uses absolute GPS positioning, map matched, to locate the vehicle on a road centre-line, when GPS is known to be sufficiently accurate. MMGPS software has now been adapted to incorporate positioning based on odometer derived distances (OMMGPS), when GPS positions are not available. Relative GPS positions are used to calibrate the odometer. If a GPS position is detected to be inaccurate, it is not used for positioning, or for calibrating the odometer correction factor. In OMMGPS, GPS pseudorange observations are combined with DTM height information and odometer positions to provide a vehicle position at '1 s' epochs. The described experiment used GPS and odometer observations taken on a London bus on a predefined route in central of London. Therefore, map matching techniques are used to test GPS positioning accuracy, and to identify grossly inaccurate GPS positions. In total, over 15,000 vehicle positions were computed and tested using OMMGPS.

In general, the position quality provided by GPS alone was extremely poor, due to multipath effects caused by the urban canyons of central London, so that odometer positioning was used much more often to position the vehicle than GPS. Typically, the ratio is 7:3 odometer positions to GPS

* Corresponding author. Tel.: +44 1443 483618; fax: +44 1443 482715.
E-mail addresses: getaylor@glam.ac.uk (G. Taylor), cbrunsdo@glam.ac.uk (C. Brunsdon), jli@glam.ac.uk (J. Li), aolden@glam.ac.uk (A. Olden), dsteup@glam.ac.uk (D. Steup), mwinter@glam.ac.uk (M. Winter).

0198-9715/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compenvurbsys.2006.02.006

positions. In the case of one particular trip, OMMGPS provides a mean error of position of 8.8 m compared with 53.7 m for raw GPS alone.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: GPS; Odometer calibration; Map matching

1. Introduction

Global navigation satellite systems (GNSS), such as the Global Positioning System (GPS), have been increasingly used in real time tracking of vehicles. Especially, when GPS is integrated with ever powerful geographic information system (GIS) technologies, the accuracy and reliability of low cost standalone GPS receivers can be significantly improved to meet the technical requirements of various transportation applications of GPS, such as vehicle navigation, fleet management, route tracking, vehicle arrival/schedule information systems (bus/train) and on demand travel information. With the autonomous European Satellite Navigations System Galileo, expected in 2008, an opportunity of a joint system 'GPS + Galileo' with more than 50 satellites will provide many advantages for civil users, in terms of availability, reliability and accuracy. However, severe multipath effects will continue to be a problem in dense urban areas. The work explained in this paper is focused on vehicle positioning. However, the method used can be adapted for almost any Mobile GI Service, such as those described by Jiang and Yao (2006), Li (2006) and Zipf and Jöst (2006). The map matching techniques used to test GPS positioning accuracy are particularly relevant to a trajectory prediction approach for location (Liu & Karimi, 2006).

To date, there have been many attempts to improve the reliability of vehicle positioning through the fusion of observations obtained by the integration of various positioning and navigation instruments. The vast majority of such systems use a GNSS, for absolute positioning, and a variety of other sensors to provide relative positioning. The usual model is using GPS to position a vehicle whenever possible and some form of inertial navigation system (INS) or dead reckoning (DR) system, such as odometer, gyro and compass, to determine a vehicle's position relative to an initial position.

Kealy, Tsakiri, and Stewart (1999) and Ramjattan (now Kealy) and Cross (1995) describe a typical solution, integrating GNSS and DR using a Kalman filtering technique. In this experiment a test route for the system was established in the centre of Perth, Western Australia. The results of this work found that "DGPS/DR solution starts to degrade from 1 m to errors as much as 35 m by the end of a 10 min period" (Kealy et al., 1999). Kalman filtering techniques do have an inherent problem, for vehicle navigation, on road networks, "in terms of stability, computational load, immunity from noise effects and observability" (Chiang, Noureldin, & El-Shiemy, 2002). The performance of the filter is heavily dependent on the models used. The model used is a compromise between a statistical predictive dynamic model and the measurement (observation) model. If too much weight is given to the dynamic model, an overly smooth track is the result, i.e. rapid changes of direction are not recognised quickly enough. If too much weight is given to the measurement model, errors would be construed as sharp changes in direction. Devising the correct model is very difficult, and without a very good model a Kalman filter will deliver the wrong result. Other accounts of using a Kalman filter for multi-sensor vehicle

navigation are given by Stephen and Lachapelle (2000) using GPS and low cost gyro, Petrovello, Cannon, and Lachapelle (2003) providing an informative discussion on levels of integration, and also Mezentsev, Lu, Lachapelle, and Klukas (2002). Hailes (1999) uses Kalman filtering with map matching.

Fei, Qishan, and Zhongkan (2000) describe fuzzy logic techniques as an alternative to Kalman filtering for GPS/INS integration. Furthermore, Mayhew and Kachroo (1998) compare solutions using various configurations of GPS, steering position, odometer, gyroscope, forward accelerometer and map matching, with sensor fusion methods Kalman filtering, rule based and fuzzy logic. Chiang et al. (2002) developed a GPS/INS multi-sensor navigation system that utilises an artificial neural network (ANN) as another alternative to Kalman filtering. Wise-McLain and Murphy (1993) describe GPS and a DR system for tracking.

Over the past 4 years a group of researchers from the GIS Research Centre, School of Computing, University of Glamorgan, have designed, developed and implemented a software application package for researching algorithms and techniques to improve GPS based on map matching for navigation and tracking. This test-bed application, called Map Matched GPS (MMGPS) processes raw GPS output data, from RINEX files, or GPS derived coordinates. It provides linkage to a GIS for access and analysis of appropriate spatial and related attribute data (primarily road and height information). MMGPS identifies the correct road, on which a vehicle is travelling on, and snaps the vehicle position onto that road. Furthermore, MMGPS corrects the derived position using its own computed correction parameters, e.g. Correction Dilution of Precision (CDOP) using history of previous position estimates and road geometry (Blewitt & Taylor, 2002). Various research experiments utilising MMGPS have been conducted and results have been fully described in Taylor, Blewitt, Steup, Corbett, and Car (2001).

Since the main objectives of this work are to determine both the accuracy and reliability of position, of a public transport bus, that can be provided using GPS, odometer and map matching techniques, a new algorithm has been developed that integrates odometer observations with the existing software, now called OMMGPS. In OMMGPS, height information obtained from digital terrain models (DTM) are used to achieve 3D GPS point positions, when only three GPS satellites are visible to the receiver. More importantly, height aiding improves the accuracy of GPS point positions with poor satellite geometry (high PDOP), and when severe signal multipathing occurs (multiple reflected GPS satellite signals).

The developed method uses absolute GPS positioning, map matched, to locate the vehicle on a road centre-line, when GPS is known to be sufficiently accurate. When this is not the case, odometer readings are used to locate the vehicle on a road centre-line. The odometer is calibrated using relative GPS positions, based on map matching criteria, such as the residuals of CDOP, for GPS precision determination (see also Section 2). The accuracy of OMMGPS is a function of the frequency of accurate GPS points, reliable map matching and correct odometer calibration.

Standard Ordnance Survey (OS) digital plan and height map products were used for road map matching and height aiding. A number of trips along a bus route in central London – Baker Street, Oxford Street, etc. – were made to test the method. A typical result of map matched GPS positioning is shown in Fig. 1.

The innovative feature of this particular implementation is that map matching is actually used for GPS accuracy determination rather than to identify the correct road the bus is driving on. This is achievable, since predefined bus routes are involved, hence the correct road is always known. The trajectory of a sequence of GPS point positions is compared

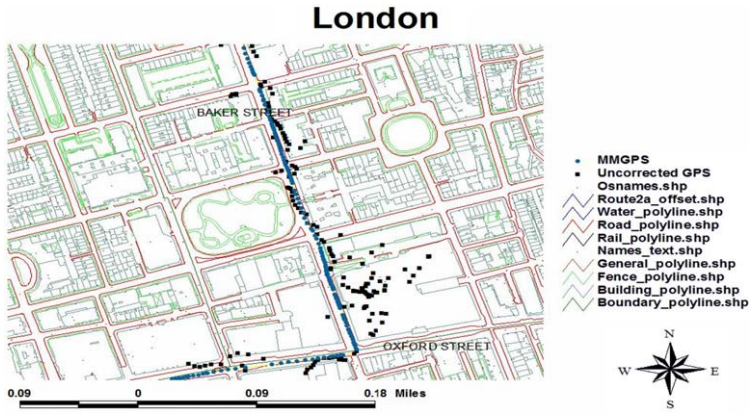

Fig. 1. Map matched GPS positioning with OMMGPS.

with an identified part of the bus route, using the map matching techniques described in this paper. If this comparison meets the map matching criteria, described below, the points are used for odometer calibration and bus positioning. Otherwise, they are discarded.

2. Methodology

The general approach was to use GPS to position the vehicle, and also to calibrate the odometer readings, but only if GPS was available and of sufficient accuracy. At all other times odometer readings were used to position the bus. Odometer positioning was achieved by tracing the distance measured by the odometer along the bus route road centre-line – actually, a 5 m offset centre-line was used, left of direction of travel, see Fig. 1. Map matching techniques are used to improve the GPS positioning accuracy, and to identify grossly inaccurate GPS positions. The previous 10 GPS/odometer positions are used for map matching calculations.

2.1. Map matching

The existing MMGPS software has been adapted to incorporate positioning based on odometer derived distances, when GPS positions are not available. This new version of map matching software, OMMGPS, works in the following way:

1. A GPS observation is read from the GPS RINEX file, and an odometer count is read from the odometer file.
2. A **Raw** vehicle position is computed using all satellites visible to the receiver, above a 15° elevation mask, plus height aiding, where height is obtained from a DTM. This height is interpolated (bilinear) at the vehicle's previous reference (**Ref**) position, i.e. snapped on the road centre-line (Li, Taylor, & Kidner, 2003). This DTM derived height of the receiver is used to provide an extra equation in the least squares pseudorange computation of GPS coordinates, i.e. computation with a minimum of three satellites is possible. For each epoch (instant time of observation) GPS points within 100 m of the road centre-line are considered.

3. An odometer correction is calibrated using odometer count at the current epoch and relative GPS distance travelled. Only if GPS position useable, otherwise this step is skipped for this epoch.
4. Road geometry based on DGPS corrections (from step 9, previous epoch) is added to the **Raw** position to give a corrected (**Cor)** position.
5. This **Cor** position is now snapped to the nearest point on the nearest road centre-line to give the current **Ref** position, and then a DTM height is calculated. That is, a **Ref** position is available that can be used to generate road geometry based on DGPS corrections for use with the next epoch's computed **Raw** position. The resultant **Ref** positions are checked for correctness using tests against map matching criteria (Taylor & Blewitt, 1999, 2000; Taylor et al., 2001); see below.
6. The odometer position is calculated, using previous **Ref** position, as well as the odometer distance.
7. A position error vector is estimated in a formal least squares procedure, in which the Correction Dilution of Precision (CDOP) is computed. This estimate is a map matched correction that provides an autonomous alternative to DGPS, fully described in Blewitt and Taylor (2002). The residuals of this process are used to determine the goodness of fit of the position error vector.
8. Position error vector (step 7) is used to adjust the **Ref** position used for step 9. This provides a long-track correction, especially when CDOP is low (rapid change of road direction).
9. DGPS corrections for each satellite pseudorange are computed using the current **Ref** position. These are retained for future use.

The main map matching criteria for snapping a **Raw** GPS position to a road centre-line **Ref** GPS position are each of the following, which have to be below a set maximum value:

- **Distance error** (absolute value of the difference between **Raw** distance and **Ref** distance, between the current and previous epochs).
- **Bearing error** (absolute value of the difference between **Raw** bearing and **Ref** bearing, between the current and previous epochs).
- **Residuals of CDOP**.
- **Maximum distance** of **Raw** GPS position from the road centre-line.

The number of satellites visible to the receiver has to be the same for current and previous epochs.

If a **Ref** GPS position passes the check, it is used for positioning the bus and calibrating the odometer correction factor. Otherwise, the position of the bus is derived from the calibrated odometer distance, and the odometer calibration correction factor is not updated. The values used for map matching are obviously open to adjustment and tuning, for different road geometry and environmental scenarios.

2.2. Distance correction factor

The previous section assumes that distances obtained from odometer readings are multiplied by a *correction factor C*, so that the distance supplied to OMMGPS is actually Cd rather than just d. It is not reasonable to assume C is fixed, as different roads and, indeed,

different road conditions on the same road will influence C. For this reason, when GPS and odometer signals are both available, C will be calibrated over a time window by comparing distances travelled, based on odometer readings with those estimated from GPS (i.e. relative distances between two GPS points). If GPS goes offline, the value of C obtained just before the GPS signal is lost (or regarded as unreliable) is used together with the odometer method to estimate location, i.e. the current odometer position is calculated based on the previous GPS or odometer position, and the current odometer reading is multiplied by the correction factor C.

2.2.1. Estimating C

C can be regarded as a correction factor between odometer-based distance estimates as used above, and those obtained from the GPS-based method. At each second t, an odometer distance d_t and GPS-based coordinate estimates (X_t, Y_t, Z_t) are obtained. Here, it is assumed d_t is a cumulative variable, so that the distance travelled between $t - 1$ and t is $d_t - d_{t-1}$. This is called Δd. Also, from the GPS measurements the cumulative distance travelled can be computed using OMMGPS. These distances are called D_t. Similar to the odometer distances, ΔD is defined as $D_t - D_{t-1}$. Thus, a model of the relationship between Δd and ΔD is

$$\Delta D_i = C\,\Delta d_i + \text{error}$$

This model can be calibrated by estimating C using least squares techniques, i.e. C is chosen to minimise the expression

$$\sum_i (\Delta D_i - C\,\Delta d_i)^2$$

It may be verified that, in this case, the estimate for C is

$$C = \frac{\sum_i \Delta D_i \Delta d_i}{\sum_i \Delta d_i^2} \tag{1}$$

However, this assumes that C is a constant correction factor. In reality, C is likely to change, depending on traffic conditions, such as road shape, weather and so on. A more realistic model allows C to vary with time, so that at each time t a distinct C_t is obtained. One approach in this situation is to estimate C according to the same model, i.e. using a 'moving window' least squares estimate. At each time t, only the values for ΔD_i and Δd_i are considered in a time window of k seconds, i.e. only data from times $t - k, t - k + 1, \ldots, t$. At time $t + 1$, data is dropped for time $t - k$ and added for time $t + 1$. Also, a weighting scheme is used in the least squares method, so that the squared errors for data close to t have a higher weighting. This gives an estimation method, which places more emphasis on minimising errors close to time t. In this case, the least squares expression to be minimised is

$$\sum_{i=0..k} w_i (\Delta D_{t-i} - C_t \Delta d_{t-i})^2$$

where w_i is the weight placed on the error at lag i seconds before time t. In this case, C_t is

$$C_t = \frac{\sum_{i=0..k} w_i \Delta D_{t-i} \Delta d_{t-i}}{\sum_{i=0..k} w_i \Delta d_{t-i}^2} \tag{2}$$

2.2.2. Weighting scheme for w_i

Some thought should be given to the weighting scheme for the w_is. Obviously, a time-decay effect is expected, so that $w_0 > w_1 > \cdots > w_k$. One possibility is to choose an exponential fall-off up to w_k, so that $w_k = c^k$ could be chosen, assuming $c < 1$. Note that $w_0 = 1$ can be fixed without loss of generality. In order to obtain the best performance of the tracking algorithm as a whole, it may be experimented with different values of k and c. Thus, the estimate for C_t may be written as

$$C_t = \frac{\sum_{i=0..k} c^i \Delta D_{t-i} \Delta d_{t-i}}{\sum_{i=0..k} c^i \Delta d^2_{t-i}} \tag{3}$$

2.2.3. Implementing the correction factor algorithm

The estimate of C_t needs to be updated each second, where the odometer reading is reliable and the GPS position is available. Providing the GPS is available for k seconds, it may be worked with an iteratively updated 'moving window' estimate. Using Eq. (2), C_t may be written as

$$C_t = \frac{\mathrm{sum1}_t}{\mathrm{sum2}_t} \tag{4}$$

where

$$\begin{cases} \mathrm{sum1}_t = \sum_{i=0..k} c^i \Delta D_{t-i} \Delta d_{t-i} \\ \mathrm{sum2}_t = \sum_{i=0..k} c^i \Delta d^2_{t-i} \end{cases} \tag{5}$$

If a record of the last k values of Δd_t and ΔD_t is available, *sum*1 and *sum*2 may be updated at each second, as well as the estimate C_t. This can be seen in

$$\begin{aligned} \mathrm{sum1}_t &= c\,\mathrm{sum1}_{t-1} + \Delta d_t \Delta D_t - c^{k+1} \Delta d_{t-k} \Delta D_{t-k} \\ \mathrm{sum2}_t &= c\,\mathrm{sum2}_{t-1} + \Delta d_t^2 - c^{k+1} \Delta d^2_{t-k} \end{aligned} \tag{6}$$

Since the algorithm requires only the values of ΔD and Δd at times t and $t - k$ but not those in between, a 'first in first out' (FIFO) of size $k + 1$ is a useful method of handling the information. Unlike the more usual 'last in first out' (LIFO) stack, popping a value from the stack returns the oldest item on the stack rather than the newest. Thus, at each second, the current values of Δd_t and ΔD_t are computed and pushed onto a FIFO stack. For the first k seconds GPS data is online, no values are popped from the stack. However, after k seconds, values of Δd and ΔD are also popped from the stack. Since the oldest values are popped from the stack, and the stack has had items pushed on to it for k seconds, it implies that Δd_{t-k} and ΔD_{t-k} will be popped.

2.2.4. Calibration if GPS data recently online

In the previous section, the method for estimating C_t is used when GPS data is available. If the GPS data is offline, the calibration of C_t cannot take place. Previous section assumed that GPS data is online for a sufficient period of time, so that

- at least k observations are pushed onto the stack, as well as
- the supplied GPS positions 'settled' and are reliable.

Thus, there is a 'run-in' period of l seconds, where the methods in the previous section cannot be applied. Clearly, l cannot be less than k, but a much larger value may be required. This is to be determined by experiment.

How C_t should be estimated in this l-second time interval, and how the position should be estimated? One possibility is to work initially with a global estimate of C to provide the odometer based estimate. As time during the burn-in period passes, the estimate should 'drift' towards the estimate produced using (5). This can be done by combining C_t and C using a weighted average, with the weighting gradually favouring C_t rather than C. The trial method here uses the formula:

$$C_t^0 = q^j C + (1 - q^j) C_t \tag{7}$$

where j is the time into the run-in period, and C_t^0 is the 'combined' estimate of the correction factor, assuming $0 < q < 1$.

Finally, $sum1$ and $sum2$ need to be 'rebuilt' in the first k seconds of this period. That is, for times up to k seconds, the current ΔD_t and Δd_t need to be included into the running mean computation, but ΔD_{t-k} and Δd_{t-k} are not being dropped. Here, it may be written as

$$\begin{cases} \mathrm{sum1}_t = c\,\mathrm{sum1}_{t-1} + \Delta d_t \Delta D_t \\ \mathrm{sum2}_t = c\,\mathrm{sum2}_{t-1} + \Delta d_t^2 \end{cases} \tag{8}$$

The estimate of the global C is updated on an ongoing basis when GPS and odometer data are available. From Eq. (3), it may be noted that

$$C = \frac{\mathrm{gsum1}_t}{\mathrm{gsum2}_t} \tag{9}$$

and $gsum1_t$ and $gsum2_t$ may be updated each second, since

$$\begin{cases} \mathrm{gsum1}_t = \mathrm{gsum1}_{t-1} + \Delta D_t \Delta d_t \\ \mathrm{gsum2}_t = \mathrm{gsum2}_{t-1} + \Delta d_t^2 \end{cases} \tag{10}$$

It may be sensible to scale $gsum1_t$ and $gsum2_t$ to avoid rounding errors, for example a running mean computation such as

$$\begin{cases} \mathrm{gsum1}_t = \frac{n-1}{n}\mathrm{gsum1}_{t-1} + \frac{1}{n}\Delta D_t \Delta d_t \\ \mathrm{gsum2}_t = \frac{n-1}{n}\mathrm{gsum2}_{t-1} + \frac{1}{n}\Delta d_t^2 \end{cases} \tag{11}$$

could be used. At any time both sums are reduced by a factor n, where n is the number of times the updating algorithm is called. This has no effect on the estimate of C as the numerator and denominator are scaled by the same factor, but it stops them from becoming very large, leading to overflow errors.

2.3. Putting it all together

The above algorithms are used on an event-driven basis. Each time a new set of observations is provided, one of three conditions applies:

- GPS and odometer data both available, not during run-in period.
- GPS and odometer data both available, during run-in period.
- Odometer data only available.

The set of actions to be taken in each case are outlined in the algorithms listed in Appendix A.

2.3.1. Alterations to the correction factor algorithm

After the algorithms, described in the previous sections, were implemented, experiments took place in which the algorithms were applied to test data. On the basis of this a number of adjustments were made. In particular, it was found that on occasions the GPS signal was only available for very short periods of time. This occurred either due to a lack of positions provided by the GPS, or because the GPS position provided was rejected as unreliable. As a result of this, there were situations when a GPS location was available for two or three seconds, then unavailable for a similar length of time and so on. This led to a problem with Algorithm 6 in Appendix A. Essentially, each time the GPS signal was available the stack was reset, as were *sum*1, *sum*2 and *j*, effectively 'forgetting' the value of C_t prior to GPS signal cut-out. An associated difficulty was that, if the run-in period was longer than 2 or 3 s, C_t was never being properly updated during these periods of intermittent GPS availability.

This led to a problem, since either the run-in period had to be very short, as did the size of the stack, or very out-of-date values of C_t were used. Neither option was acceptable. In the first instance, C_t was undersmoothed, leading to erratic odometer based estimates (i.e. a lack of precision) – in the second case, C_t does not vary wildly but is biased, since much of the time values closer to the global C would be supplied – this led to a lack of accuracy.

To overcome this problem, it was noted that GPS cut-outs were never very long, so that calibration of C_t prior to the cut-out would still provide useful information. To this end, Algorithm 6 was modified so that the resetting of the stack did not occur when a GPS location became available. The modified algorithm, labelled Algorithm 6m, is listed in Appendix B. Experimentation indicated better performance. For the training data, a run in a period of 10 s was found, as well as a stack size of 10 – together with $c = 0.8$ the best performance was achieved, in terms of offset error to beacons (for detailed discussion of the methodology see Section 4). It was also found at a later stage that a policy of always returning the odometer-based location estimate (even when GPS was available) improved the accuracy of predictions – this is also reflected in Algorithm 6m.

2.4. Height aiding

In OMMGPS height aiding is used throughout to add an extra equation in the least squares approximation computation of GPS position. Height information obtained, using bilinear interpolation, from a digital terrain model (DTM) are used to achieve 3D GPS point positions, when only three GPS satellites are visible to the receiver.

More importantly, height aiding improves the accuracy of GPS point positions with poor satellite geometry (high PDOD), and when severe signal multipathing occurs (multiple

Table 1
Number of GPS positions computed

Trip no.	GPS only	GPS + height aiding
3.4	3834	3834
4	3468	3635
6	3320	3991
9.2	3713	4014

reflected GPS satellite signals). The number of GPS positions computed is nearly always increased by using additional height information, displayed in Table 1. Trip 3.4 is the exception.

3. Implementation

OMMGPS consists of a Dynamic Link Library (DLL), written in C++, together with a GUI for use in ESRI's GIS products ArcView or ArcGIS. The GUI was originally written in ArcView's Avenue and has recently been translated to Visual Basic for use in ArcGIS (Steup & Taylor, 2003). The GIS is used to visualise the results graphically, using background OS mapping.

4. Data processing and results

A number of separate trips along a bus route in central London (Baker Street, Oxford Street, etc.) were made to test the method. During each of these trips, GPS observations using low cost GPS L1 receivers and odometer observations using existing mechanical odometers were taken at each second, on the bus. In total, over 15,000 vehicle positions were computed using OMMGPS. Also, on each trip, the bus recorded the time when it detected a beacon at the beacon's known location, actually to a normal intersect with a 5 m offset centre-line, see Fig. 2.

The positions of the bus at these times were used as the 'true' position of the bus. There are altogether 13 beacons on the bus route, which are used for determining OMMGPS accuracy. Using this data the exact equivalent OMMGPS positions used to calculate distances were obtained by interpolation, applying the OMMGPS positions at the nearest second before and after the beacon detection time. This is simple linear interpolation.

A bus position was available at each second, either computed by GPS or odometer observations. For the whole route, odometer positions are used much more than GPS positions: 70% odometer positions, 30% GPS positions. For the calculation of OMMGPS, position at beacon detection time for the 13 beacons used; four positions were calculated using only GPS, and nine positions were calculated using only odometer. The results

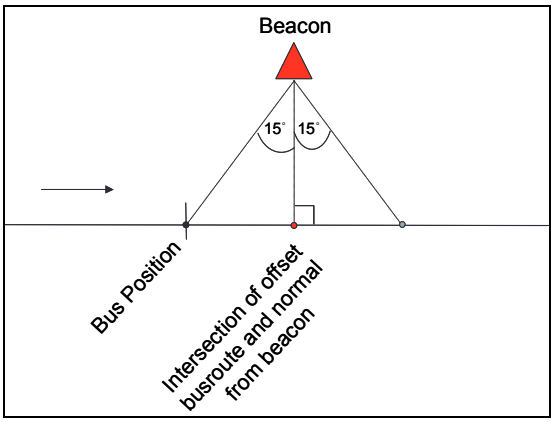

Fig. 2. Beacon detection.

Table 2
OMMGPS statistics for all beacons – for 95% – trip 4

	Error for 100% (m)	Error for 95% (m)
Mean	8.8	5.2
Standard deviation	6.6	3.6
Range	18.7	11.2
Minimum	0.9	0.9
Maximum	19.6	12.2

Table 3
Average errors for all trips

Trip no.	GPS only (m)	OMMGPS (m)	95% cut-off OMMGPS (m)
3.4	27.9	11.3	6.6
4	53.7	8.8	5.2
6	>100	28.3	14.1
9.2	40.7	22.8	14.4

obtained for trip 4 are presented in Table 2. This bus trip is the one, on which the algorithm was developed and tuned, and it shows the excellent potential of the method. The results of the other three trips processed, i.e. trip 3.4, 9.2 and 6, are shown in Table 3.

There is a substantial improvement in the accuracy of bus position using OMMGPS instead of only raw GPS. In the case of trip 4, OMMGPS provides a mean error of 8.8 m compared with 53.7 m for raw GPS without odometer.

5. Conclusions

A new algorithm that integrates odometer observations with the existing MMGPS map matching software was developed and successfully implemented. This new algorithm, called OMMGPS, utilises map matching criteria, not to determine which road a vehicle is on, but to determine GPS position precision, in order to calibrate an odometer with the help of GPS. In OMMGPS, GPS pseudorange observations are combined with odometer positions and DTM height information to provide a vehicle position at '1 s' epochs. Generally, odometer positioning is used much more often to position the vehicle than GPS. Typically, the ratio is 7:3 odometer positions to GPS positions. This predominant use of odometer positioning is due either to GPS not being available or GPS positions being considered to be too inaccurate to use. This lack of GPS positions is due to satellite masking by buildings or the result of severe GPS signal multipath in the urban canyons of central London.

Four bus trips along the same bus route were used to test OMMGPS. The results obtained from these four trips are most encouraging. In total, over 15,000 vehicle positions were computed using OMMGPS. The positions provided by OMMGPS at the time of beacon detection can be considered to be a random sample of the accuracy provided by OMMGPS, compared to the accuracy provided by GPS alone. That is, if a GPS position was available at all, on or near the beacon detection time. The average error of OMMGPS positions, over all vehicle positions, using the random sample of beacon detection times, is 17.8 m overall, and 10.1 m for a 95% cut-off. This compares with an average error for GPS alone of at least 55.6 m overall.

Moreover, the effectiveness of OMMGPS is entirely dependent on the frequency and accuracy of GPS derived positions. The GPS data provided for testing compared very poorly with similar raw L1 pseudorange GPS data collected independently along the same bus routes, albeit using a much more expensive receiver and antenna for the independent test observations. Similarly, receiver coordinates collected using another low cost L1 GPS receiver also provided improved positions, although this was most probably due to smoothing provided by this particular receiver's own navigation filter.

In conclusion, the technique developed in OMMGPS works well, and can be further improved with more superior low cost GPS receiver technology or a more careful attention to its operational application. Due to the fact that the method is based on known routes, it is not only appropriate for bus positioning but also for the use on railways.

Acknowledgements

This work was undertaken as part of a project commissioned by PA Consulting Group and Transport for London. Particular thanks must be given to Dr. Phil White and Kenny Steele, for their help with the work, as well as problem solving and ideas.

Appendix A. Outline of algorithms

Algorithm 1 (Estimate location of bus from odometer signal)

```
input: float xlast, ylast, d, xroute[1:n], yroute[1:n]; integer position
currentx    xlast
currenty    ylast
nextx    xroute[position]
nexty    yroute[position]
D1    0
D2    distance3d(currentx, currenty, nextx, nexty)
position    position + 1
currentx    nextx
currenty    nexty
nextx    xroute[position]
nexty    yroute[position]
D1    D2
D2    D1 + distance3d(currentx, currenty, nextx, nexty)
beta    (d □ D1) □ (D2 □ D1)
xestimate    (1 □ beta) × currentx + beta × nextx
yestimate    (1 □ beta) × currenty + beta × nexty
output: position, xlocation, ylocation
```

Algorithm 2 ('distance3d' function used by Algorithm 4). NB. This assumes the existence of a function DTMZ(x, y), which interpolates the z-coordinate of a location (x, y) from a DTM.

```
input: float x1,x2,y1,y2
z1    DTMZ(x1, y1)
z2    DTMZ(x2, y2)
```

d $\sqrt{\{(x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2\}}$
output: d

Algorithm 3 (Update the value of C_t)

input: integer k; float gamma, sum1, sum2, delta.d0, delta.D0, delta.dk, delta.Dk
sum1 gamma × sum1
sum1 sum1 + delta.d0 × delta.D0
sum1 sum1 − power(gamma, k + 1) × delta.dk × delta.Dk
sum2 gamma × sum2
sum2 sum2 + delta.d0 × delta.d0
sum2 sum2 − power(gamma, k + 1) × delta.dk × delta.dk
Ct sum1 − sum2
output: Ct, sum1, sum2

Algorithm 4 (Update the value of C)

input: integer n; float gamma, gsum1, gsum2, delta.d0, delta.D0
factor (n − 1) / n
gsum1 gsum1 × factor + delta.d0 × delta.D0 × (1 − factor)
gsum2 gsum2 × factor + delta.d0 × delta.d0 × (1 − factor)
C gsum1 − gsum2
n n + 1
output: C, gsum1, gsum2, n

Algorithm 5 (Combine C)

input: integer j; float rho, Ct, C
output: (1 − power(rho, j)) × Ct + power(rho, j) × C

Algorithm 6 (Overview of events)

initialise: Set n, sum1, sum2, gsum1, gsum2 to 0
loop
Repeat each reading
if GPS and odometer available, not during run-in **then**
  Compute delta.D0 and delta.d0
  Push delta.D0 and delta.d0 onto FIFO stack
  Pop delta.Dk and delta.dk from FIFO stack
  Update estimate of Ct using Algorithm 4
  Update estimate of C using Algorithm 4
  Return GPS location estimate as current position
end if
if GPS and odometer available, during run-in **then**
  if GPS just became available **then**
    set j, sum1, sum2 to 0
    Reset FIFO Stack

```
    end if
    Obtain current value of C
    if j < k then
            Return odometer based location estimate (Algorithm 4) with correc-
            tion factor C.
            Compute delta.D0 and delta.d0 and push onto FIFO stack
            sum1    sum1 × gamma + delta.D0 × delta.d0
            sum2    sum2 × gamma + delta.d0 × delta.d0
    else
            Compute delta.D0 and delta.d0 and push onto FIFO stack
            Pop delta.Dk and delta.dk from FIFO stack
            Update estimate of C using Algorithm 4
            Update estimate of Ct using Algorithm 4
            Obtain estimate of C^0 using Algorithm 4
            Return odometer based location estimate (Algorithm 4) with correc-
            tion factor C^0.
        end if
end if
if odometer only available then
        Return odometer-based location estimate (Algorithm 4) with correction fac-
        tor of the last C_t before the GPS data went offline.
end if
end loop
```

Appendix B. Modification of Algorithm 6

Algorithm 6m (Overview of events (modified))

```
initialise: Set n, sum1, sum2, gsum1, gsum2 to 0
loop
Repeat each reading
if GPS and odometer available, not during run-in then
    Compute delta.D0 and delta.d0
    Push delta.D0 and delta.d0 onto FIFO stack
    Pop delta.Dk and delta.dk from FIFO stack
    Update estimate of Ct using Algorithm 4
    Update estimate of C using Algorithm 4
    Return odometer-based location estimate as current position
end if
if GPS and odometer available, during run-in then
    Obtain current value of C
    if j < k then
        Return odometer based location estimate (Algorithm 4) with correction
        factor C.
        Compute delta.D0 and delta.d0 and push onto FIFO stack
```

sum1 sum1 × gamma + delta.D0 × delta.d0
sum2 sum2 × gamma + delta.d0 × delta.d0

else

Compute delta.D0 and delta.d0 and push onto FIFO stack
Pop delta.Dk and delta.dk from FIFO stack
Update estimate of C using Algorithm 4
Update estimate of Ct using Algorithm 4
Obtain estimate of C^0 using Algorithm 4
Return odometer based location estimate (Algorithm 4) with correction factor C^0.

end if

end if

if odometer only available **then**

Return odometer-based location estimate (Algorithm 4) with correction factor of the last C_t before the GPS data went offline.

end if

end loop

References

Blewitt, G., & Taylor, G. (2002). Mapping dilution of precision (MDOP) and map matched GPS. *International Journal of Geographical Information Science*, 1365-8816, *16*(1), 55–67.

Chiang, K., Noureldin, A., & El-Shiemy, N. (2002). Multi-sensor integration using neuron computing for land-vehicle navigation. *GPS Solutions, 6*(4), 209–218.

Fei, P., Qishan, Z., & Zhongkan, L. (2000). The application of map matching method in GPS/INS integrated navigation system. *International Telemetering Conference, USA Instrument Society of America, 36*(2), 728–736.

Hailes, T. A. (1999). Integrating technologies: DGPS Dead Reckoning and map matching. *International Archives of Photogrammetry and Remote Sensing, 32*(2W1), 1.5.1–1.5.8.

Jiang, B., & Yao, X. (2006). Location-based services and GIS in perspective. *Computers, Environment and Urban Systems*, this special issue, doi:10.1016/j.compenvurbsys.2006.02.003.

Kealy, N., Tsakiri, M., & Stewart, M. (1999). Land vehicle navigation in the urban canyon – A Kalman filter solution using integrated GPS, GLONASS and Dead Reckoning. In *Proceedings of ION GPS 99 conference* (pp. 509–518).

Li, C. (2006). User preferences, information transactions and location-based services: A study of urban pedestrian wayfinding. *Computers, Environment and Urban Systems*, this special issue, doi:10.1016/j.compenvurbsys.2006.02.008.

Li, J., Taylor, G., & Kidner, D. (2003). Accuracy and reliability of map matched GPS coordinates: Dependence on terrain model resolution and interpolation algorithm. In *AGILE Conference on geographic information science*, Lyon, France.

Liu, X., & Karimi, H. A. (2006). Location awareness through trajectory prediction. *Computers, Environment and Urban Systems*, this special issue, doi:10.1016/j.compenvurbsys.2006.02.007.

Mayhew, D., & Kachroo, P. (1998). Multi-rate sensor fusion for GPS using Kalman filtering, fuzzy methods and map-matching. In *Proceedings of SPIE conference on sensing and controls with intelligent transportation systems*, Boston, Massachusetts, USA, November 1998 (Vol. 3525, pp. 440–449).

Mezentsev, O., Lu, Y., Lachapelle, G., & Klukas, R. (2002). Vehicle navigation in urban canyons using a high sensitivity GPS receiver augmented with a low cost rate gyro. In *Proceedings of ION GPS 2000 conference*, Portland, OR, USA, September 2002.

Petrovello, M. G., Cannon, M. E., & Lachapelle, G. (2003). Quantifying improvements from the integration of GPS and a tactile grade INS in high accuracy navigation systems. In *Proceedings of ION NTM conference*, Anaheim, CA, USA, January 2003.

Ramjattan, A., & Cross, P. A. (1995). A Kalman filter model for an integrated land vehicle navigation system. *Journal of Navigation, Cambridge University Press, 49*(2), 293–302.

Stephen, J., & Lachapelle, G. (2000). Development of a GNSS-based multi-sensor vehicle navigation system. In *Proceedings of ION NTM conference*, Anaheim, CA, USA, January 2000.

Steup, D., & Taylor, G. (2003). Porting GIS applications between software environments. In *Proceeding of the GIS research UK 2003 11th annual conference*, London (pp. 166–171).

Taylor, G., & Blewitt, G. (1999). Virtual differential GPS and road reduction filtering by map matching. In *Proceedings of ION'99, twelfth international technical meeting of the satellite division of the institute of navigation*, Nashville, USA (pp. 1675–1684).

Taylor, G., & Blewitt, G. (2000). Road reduction filtering using GPS. *Proceedings of 3rd AGILE conference on geographic information science*, Helsinki, Finland (pp. 114–120).

Taylor, G., Blewitt, G., Steup, D., Corbett, S., & Car, A. (2001). Road reduction filtering for GPS–GIS navigation. *Transactions in GIS, 5*(3), 193–207.

Wise-McLain, P., & Murphy, M. D. (1993). A GPS/Dead Reckoning system for tracking and GIS application. In *Proceedings of ION GPS-93, sixth international technical meeting of the satellite division of the institute of navigation*, Salt Lake City, Utah (pp. 1631–1636).

Zipf, A., & Jöst, M. (2006). Implementing adaptive mobile GI services based on ontologies: Examples from pedestrian navigation support. *Computers, Environment and Urban Systems*, this special issue, doi:10.1016/j.compenvurbsys.2006.02.005.