A Real Time Localization System for Vehicles Using Terrain-Based Time Series Subsequence Matching

Hongjuan Zhang 1,2, Wenzhuo Li 1,*, Chuang Qian 3 and Bijun Li 1,2

1 State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430072, China; hongjuanzhang@whu.edu.cn (H.Z.); lee@whu.edu.cn (B.L.)
2 Engineering Research Center for Spatio-Temporal Data Smart Acquisition and Application, Wuhan University, Wuhan 430072, China
3 GNSS Research Center, Wuhan University, Wuhan 430072, China; qc_gnss@whu.edu.cn

* Correspondence: alvinlee@whu.edu.cn

Received: 16 July 2020; Accepted: 5 August 2020; Published: 13 August 2020

Abstract: Global Navigation Satellite Systems (GNSSs) are commonly used for positioning vehicles in open areas. Yet a GNSS frequently encounters loss of lock in urban areas. This paper presents a new real-time localization system using measurements from vehicle odometer data and data from an onboard inertial measurement unit (IMU), in the case of lacking GNSS information. A Dead Reckoning model integrates odometer data, IMU angular and velocity data to estimate the rough position of the vehicle. We then use an R-Tree structured reference road map of pitch data to boost spatial search efficiency. An optimized time series subsequence matching method matches the measured pitch data and the stored pitch data in reference road map for more accurate position estimation. The two estimated positions are fused using an extended Kalman filter model for final localization. The proposed localization system was tested for computational complexity with a median runtime of 12 ms, and for positioning accuracy with a median position error of 0.3 m.

Keywords: real-time vehicle localization; time series subsequence matching; integrated localization system; dead reckoning; IMU

1. Introduction

Autonomous vehicles and connected vehicles have emerged in recent years. This development is linked to the rapid development of intelligent transportation systems based on advances in computing speed, and 5G and sensor technologies. One of the key issues in all these systems is the ability to localize a vehicle precisely. Their effectiveness is heavily dependent on lane-level or decimeter-level ego-localization. Furthermore, the localization of the vehicles must be highly reliable to prevent failures of these systems during operation.

Global Navigation Satellite Systems (GNSSs) are commonly used for positioning vehicles. Given a sufficient number of satellites, these systems can achieve centimeter-level precision by taking advantage of Continuously Operating Reference Stations (CORS) in an open area. The BeiDou Navigation Satellite System (BDS) in particular, has been on the rise in recent years and shows great potential in terms of positioning and timing [1]. Combining multiple satellite systems improves the availability of signals, gives operators more access, and increases accuracy [2]. However, GNSS frequently encounters loss of lock in urban areas with high buildings and trees, which is averse to GNSS geometry and visibility, degrades positioning accuracy, and can even lead to positioning failure.

An Inertial Navigation System (INS) is an attractive augmentation of GNSS [3] that can solve this issue. An INS has Dead Reckoning (DR) capability and can provide position, velocity and attitude estimation even when loss of lock occurs in GNSS. At the same time, INS error can be mitigated using...
position and velocity updates from GNSS when in open areas. However, an INS suffers from drifting errors that grow unbounded with respect to operation time, making it difficult for an INS to balance between cost and positioning accuracy. A tactical grade inertial measurement unit (IMU) can achieve high localization accuracy, but is too expensive to be used on a commercial scale [4]. To improve the positioning performance of low cost inertial measurement units (IMUs), in recent years, INSs have been combined with other sensors, like steering encoders [5], wheel tachometers [6] and odometers [7]. These sensors, however, are subject to wheel slips, road skidding and changing wheel diameters, adding uncertainty and accumulated error to positioning results.

Map-based localization using LiDAR [8], and cameras [9] is another common way to deal with GNSS-denied environments. In these methods, a prior feature map is generated from point clouds obtained from LiDAR or images, and then real-time ego-localization is performed using feature-matching techniques. This feature-based principle is effective in environments with abundant features such as indoors, yet, it is not suitable for outdoor environments with sparse features [10]. Moreover, LiDAR and cameras are susceptible to weather phenomena like rain, fog, and snow and light conditions such as heavy shadow, and strong light.

In the map-based category, terrain-based localization has also received attention, as an IMU can measure road grades by vehicle pitch, road bank angles by vehicle rolls, or the road curvature by vehicle yaw rate. In terrain-based localization, terrain information of the driving region is stored in a predetermined map and new measurements of pitch, roll, or yaw angle from an in-vehicle IMU sensor are compared with the predetermined map data to localize the vehicle. Dean et al. [11] proposed a particle filter algorithm to obtain sub-meter longitudinal position accuracy using spatial pitch measurements without a GNSS device. They demonstrated that the low-frequency pitch data is speed-independent so that it is feasible to localize a vehicle by terrain disturbances in pitch. Another main contribution of their work is the use of a particle filter, which finds the optimal position estimation from a non-Gaussian probability distribution via similarities between the pitch measurement and pitch reference in the predetermined pitch map, and is suitable for real-time applications. Li et al. [12] improved this method by considering braking events of vehicles. They proposed an acceleration-considered model to take into account the noise from braking to make the particle filter-based method more robust. Instead of using the vehicle pitch angle, Ahmed et al. [4,13] applied this method using vehicle roll angle estimation using low cost Micro Electromechanical System (MEMS)-IMU sensors whose external acceleration in the lateral direction was compensated before the roll angles were calculated.

Terrain information measured by in-vehicle sensors including pitch, roll, and yaw angles are typical kinds of time-series sequences (TSS). TSS are defined by the explicit information about timing, for example, electric consumption, climate data, and speech; or an ordering of values can be inferred from the data such as handwriting [14,15]. Comparing with other kinds of time-series sequences, terrain information also contains implicit location information. Localizing a vehicle by terrain information becomes a pattern recognition task, specifically, a task about similarity-based pattern querying in time-series databases.

Problems in similarity-based pattern queries in time-series databases include the definition and measurement of similarity in time-series sequences. The Dynamic Time Warping method (DTW) using distance in similarity comparisons is a classic and advanced solution [16–18], which is ubiquitously used in audio/video processing [19,20], gesture recognition [21], handwriting recognition [22], industry [23], astronomy [24,25], medicine [26], geology [27], and finance [28]. In the last decade, dozens of alternative measures have been suggested, like the symbolic aggregate approximation methods [29,30], trend-based time series similarity [31], shape-based time-series similarity [14,32], event-based time-series similarity [33], and interest point methods [34,35]. Rapid developments in time-series matching methods make terrain-based vehicle localization more reliable and practical. Martini et al. [36] used the Pearson-product correlation coefficient as a distance measure, instead of DTW, to compare the road grade measurements with an off-line reference map to find a path in the map corresponding to the location of the maximum correlation coefficient. Vemulapalli et al. [34,37]
proposed a new interest point method with multi-scale extrema features, to extract feature vectors from the pitch information of road regions and created a KD-tree in the preprocessing phase. In the online phase, the feature vectors for a query signal were tested for a match within the feature tree to find vehicle’s position estimation. Laftchiev et al. [38] proposed a linear model based on autoregressive models with an exogenous input to segment the pitch data and compare model output instead of original pitch data, which accelerates the matching process. However, in all these methods, finding a balance between time consumption and accuracy is an elusive goal. Therefore, new approaches must be developed to improve the viability of using advanced time-series subsequence matching methods for vehicle localization. In this work, we present a new real-time localization system that combines INS measurements, odometer data and an advanced DTW method to achieve both accuracy and computation efficiency without recourse to GNSS signals.

1.1. Problem Formulation

Previous works attempt to localize a vehicle based on terrain information using either particle filter-based methods or solely time series subsequence (TSS) matching methods, but both have limitations. Particle filter represents the posterior distribution of the state using a set of particles which evolve recursively as new information becomes available [39]. Particle degeneracy is a common problem when implementing a particle filter, which reduces the performance of the particle filter. Previous research on particle filter-based localization [4,11–13] using terrain information addressed particle degeneracy by increasing the particle numbers, which increased the computation burden. Moreover, only observation information at a current moment was used to update the particles. As in [12], pitch or pitch difference measurements have similar trends at different speeds, but at a specific location, the values of measurements at various speeds may be at different orders of magnitude. Therefore, using information from the current moment only is not sufficient to ensure accuracy. Consequently, we choose to use TSS matching methods to find the path of the vehicle and current position using a sliding window of a certain length incorporating historical data.

DTW has been shown to be the most precise measure for time-series data queries in most domains [16,40]. Even though previous localization research [11,34,37,38] has argued that DTW is not computationally efficient, nevertheless authors in [16] proposed an exact DTW algorithm whose sequential search is much faster than most alternatives. In our work, we adopted the DTW algorithm from [16] to do TSS matching on pitch data. Furthermore, localization processes discussed in previous research depend solely on time-series matching, regardless of accuracy, which is not robust as pitch data varies with speed and increases the chance of matching failures.

Another unsolved issue with pitch-based localization is that IMU error is neglected. IMU gyros exhibit drift performances from 0.1°/h at a tactical grade to 100°/h at a typical Micro Electromechanical System (MEMS) grade [41]. A tactical IMU alone can accomplish accurate localization but is way too expensive to be applied broadly on consumer vehicles. A new pitch-based localization system needs to be designed for low grade IMUs.

1.2. Contributions

We designed a new pitch-based localization system, which has several advantages over the methods described in the literature:

1. An advanced DTW algorithm is used and the road map with terrain information is stored in an R-Tree to speed up the online localization process, which makes our system truly real-time;
2. To ensure the accuracy at the same time, an integrated localization model is proposed to fuse in-vehicle multisensory data and time-series data matching results;
3. Our proposed model is insensitive to dynamics of the vehicle velocity while most existing methods require steady speeds;
4. Our system is robust to abrupt changes in single-pitch data since it uses time-series subsequence pitch data instead of single time-point pitch data.

The remainder of this paper is organized as follows. The methods are described in Section 2. Experimental implementation and the results are presented in Section 3. Section 4 provides the conclusions.

2. Methods

In our integrated model, only two in-vehicle sensors are used: an odometer and a low-grade IMU, which are easily installed on almost all vehicles. The model includes an offline mapping and an online localization process. In Figure 1, we illustrate our system in the online localization phase at time step k. Measurements from IMU and odometer are combined by a Dead Reckoning (DR) model to predict the navigation state (latitude, longitude, height, roll, pitch, yaw) of a vehicle. The predicted pitch value is stored in a vector together with all pitch values in previous \( t_d \) seconds, to generate a pitch TSS \( x^k \). Accumulated driving distance in the \( t_d \) seconds is calculated by the odometer measurements, and if it is less than a threshold, the predicted navigation state is returned as the result at the current time step. From a predefined map which includes accurate positions and pitch values of road points, a closest road point to the vehicle can be found based on the vehicle navigation state. Given a search window, a search area in the road map can be found based on the closest road point. Then a series of pitch TSS \( Y^k \) from the map is generated based on the search area. A time series subsequence matching model is applied for \( x^k \) and \( Y^k \) to find the best-fit matching subsequence in \( Y^k \). An estimation model is used to estimate the vehicle position from the matching model. Finally, an integrated model is used to integrate the predicted position from the DR model and the estimated position from the matching model, and return a final vehicle position. Detailed explanations of key models with grey background in Figure 1 are shown in the following text.

![Figure 1. The framework of our online localization system at time step k.](image-url)
2.1. Offline Road Map Generation

A road map including terrain information is generated for the driving region, and all sensors are synchronized before mapping. The vehicle must travel slowly and smoothly to gather a sufficient number of discrete points in the test region, because the higher the resolution of the map, the higher the positioning accuracy. The ith point of the road map is defined as \( p_i = [id, lat_{id}, lon_{id}, h_{id}, \theta_{id}, \Delta s_{id}]^T \), where \( id \) is the index of the point \( i \), the values \( lat_{id}, lon_{id}, h_{id}, \theta_{id} \), and \( \Delta s_{id} \) are its latitude, longitude, height and pitch data, respectively; \( \Delta s_{id} \) is the distance traveling from the last point, as recorded by the odometer.

After data acquisition, the road points are stored in an R-Tree. R-Tree is a dynamic index structure for spatial searches [42], and can index multi-dimensional information such as geographical coordinates in a highly efficient manner. When the point data is organized in an R-Tree, for a point, the nearest neighbor \( p_j \) within a given distance \( r \) can efficiently be queried. Even though we structure the R-Tree by latitude and longitude information, tree nodes include all features of \( p_i \). Nodes that are in the same bounding box are stored in a list and have consecutive index numbers. Therefore, to find a previous neighbor whose index is \( id_{i-1} \), merely requires a search for the previous node in the bounding box, or if \( p_i \) is the first node in the bounding box, a search for the last node in the previous child bounding box.

2.2. Dead Reckoning Model

In our DR model, real-time measurements from odometer and IMU are integrated to provide a position estimation of a vehicle. A low-cost IMU can estimate the position, attitude, and velocity of a vehicle from the raw data: velocity increments as measured by an accelerometer and angular increments as measured by a gyroscope, in the navigation frame. Because of the drift of the accelerometer and gyroscope in the IMU, the velocity and angular increments from the IMU contain errors that cause the navigation results to drift rapidly. In [43], the author elaborated estimation techniques for low-cost inertial navigation; we however, focus on integrating an IMU and odometer, we call the DR model.

We define the vehicle frame (v-frame) as forward-transversal-down, originating at the vehicle center, the navigation frame (n-frame) as north-east-down, originating at the earth mass center, and body frame (b-frame) expressed as forward-transversal-down, originating at the IMU center. More details about the frames can be seen in [43]. The odometer is installed on the rear wheel, as it is not used for steering. We assume that the vehicle does not have either a cross track or vertical speed but only an along-track speed \( v \) at time step \( k \), as measured by the odometer, so that the speed of the vehicle at the v-frame is \( v_{wheel}^n = [v, 0, 0]^T \). The relationship between the velocity of vehicle at the n-frame \( v_{IMU}^n \) and \( v_{wheel}^n \) can be expressed as:

\[
\begin{align*}
\delta v_{IMU}^n & = \delta v_{wheel}^n - C_{nb}^{bn} (\alpha_{nb} \times) \delta \omega_{wheel}^n, \\
\text{with } v_{wheel}^n & = C_{v}^{bn} \delta \omega_{wheel}^n
\end{align*}
\]

where \( \delta v_{IMU}^n \) and \( \delta v_{wheel}^n \) are the error of \( v_{IMU}^n \) and \( v_{wheel}^n \), respectively. The value \( C_{nb}^{bn} \) and \( C_{v}^{bn} \) express the rotation matrix from the b-frame to the n-frame and from the v-frame to the b-frame, while \( \alpha_{nb} \times \) is the cross-product form of the rotation rate of the b-frame with respect to the n-frame, and \( \delta \omega_{wheel}^n \) is the lever-arm vector of the odometer in the b-frame.

Considering this relationship, the Extended Kalman Filter (EKF) is used to integrate the measurements from the IMU and the odometer. A state vector \( X^k \) is constructed as:

\[
X^k = [\delta \phi_{IMU}^k, (\delta v_{IMU}^n)^T, (\delta P_{IMU}^k)^T, (\delta \epsilon_a^k)^T, (\delta \epsilon_s^k)^T, (\delta \epsilon_g^k)^T, (\delta \omega_{wheel}^n)^T]^T
\]

\[
W^k = [\delta \omega_a, \delta \omega_g]
\]
where superscript $T$ is the transposition, $\delta \omega_{IMU}^{n,k}$ is the error in attitude as estimated by INS using the IMU at the $n$-frame, $\delta \vec{p}_{IMU}^{n,k}$ is the error of position, $\delta a_{b}^{k}$ is the drift error of the accelerometer at the $b$-frame, and the drift error of gyroscope is $\delta \omega_{g}^{k}$, and $\delta \omega_{wheel}^{k}$ is the scale factor error of the odometer. The value $W_{k}$ is the noise in $X_{k}$ including the white noise generated by the specific force from the accelerometer $\omega_{a}^{k}$ whose deviation is the velocity random walk of the accelerometer, and the white noise of the angular rate from the gyroscope $\omega_{g}^{k}$ whose deviation is the angular random walk of the gyroscope.

The EKF state transformation is shown in Equation (4), where $P_{k}$ and $G_{k}$ are coefficient matrix. Then, Equation (4) is discretized in time so that a coarse estimation of state vector $X_{k-1}$ can be derived from $X_{k-1}$ by the identity matrix $I$, $P_{k-1}$, $G_{k-1}$ and $W_{k-1}$:

\[
X_{k-1} = \Phi^{k|k-1}X_{k-1} + G^{k-1}W_{k-1},
\]

with $\Phi^{k|k-1} = (1 + P_{k-1} \Delta t)$

The EKF observation function is constructed as follows:

\[
Z_{k} = H_{k}X_{k} + V_{k}
\]

where $H_{k}$ is coefficient matrix and $V_{k} \sim N(0, R_{k})$ is observation error covariance matrix. From Equation (1) we can get:

\[
Z_{k} = v_{IMU}^{n,k} - \nu_{\text{wheel}}^{n,k} = \delta v_{IMU}^{n,k} - \delta \nu_{\text{wheel}}^{n,k} - C_{b}^{n}(a_{b}^{n,k} \times)_{\text{wheel}} = -\nu_{\text{wheel}}^{n,k} \times \delta \nu_{\text{IMU}}^{n,k} + \nu_{\text{IMU}}^{n,k} - M_{b}^{n} \delta \omega_{\text{wheel}}^{k}
\]

so that:

\[
H_{k} = [-\nu_{\text{wheel}}^{n,k} \times \quad I_{3x3} \quad 0_{3x9} \quad -M_{b}^{n} \quad -C_{b}^{n}(a_{b}^{n,k} \times)]
\]

Afterwards, $X_{k-1}$ is updated by $Z_{k}$ via EKF to get $X_{k}$:

\[
X_{k} = X_{k-1} + K_{k}(Z_{k} - H_{k}X_{k-1})
\]

\[
P_{k} = \Phi^{k|k-1}P_{k-1} - \Phi^{k|k-1}^{T} + Q_{k}
\]

\[
K_{k} = \Phi^{k|k-1}^{T}(H_{k} \Phi^{k|k-1} + R_{k})^{-1}
\]

\[
P_{k} = (I - K_{k}H_{k})P_{k-1}
\]

where $Q_{k}$ is the spectral density matrix, $P_{k}$ the error covariance of $X_{k}$, and $K_{k}$ the Kalman gain of EKF. The estimated error $X_{k}$ is fed back to the navigation state of position, velocity, and attitude in the $n$-frame:

\[
\begin{bmatrix}
P_{k}^{n,k} \\
\nu_{k}^{n,k} \\
C_{b}^{n,k}
\end{bmatrix}
= \begin{bmatrix}
I_{3x3} & 0_{1x3} & 0_{1x3} \\
0_{1x3} & I_{3x3} & 0_{1x3} \\
0_{1x3} & 0_{1x3} & I_{3x3}
\end{bmatrix}
\begin{bmatrix}
P_{IMU}^{n,k} \\
\nu_{IMU}^{n,k} \\
C_{b}^{n,k}
\end{bmatrix}
- \begin{bmatrix}
\delta p_{IMU}^{n,k} \\
\delta \nu_{IMU}^{n,k} \\
\delta \omega_{IMU}^{n,k}
\end{bmatrix}
\]

If the $\nu_{k}^{n,k}$ measured by the odometer drops below a certain predefined threshold, a zero-velocity update (ZUPT) is applied instead of the wheel sensor velocity, $\nu_{\text{wheel}}^{n,k}$. More details about ZUPT implementations can be found in [44].
After estimating the current navigation state, given a predefined time duration $t_d$, we calculate the accumulated distance $d_{k_{td}}^d$ within $t_d$ and the corresponding pitch data $x^k$:

$$d_{k_{td}}^d = \sum_{j=k-t_{td}f}^{k} \Delta s_j, \quad x^k = [\theta_{k-t_{td}f}, \theta_{k-t_{td}f+1}, \ldots, \theta_k]^T \quad (14)$$

$f$ is the sensor frequency and $\Delta s$ is the odometer increment. If $d_{k_{td}}^d$ is less than a predefined distance threshold $d_{\text{thre}}$, then the navigation state estimated by the DR model is assumed to be the final vehicle state. As an illustrative example, consider a vehicle stopping when a traffic light turns red, in this condition $d_{k_{td}}^d$ would be small and the DR model is applied for final navigation state estimation at this time step.

### 2.3. Time Series Subsequence Matching Model

After the vehicle navigation state is calculated using the DR model and exploiting the efficiency of an R-Tree spatial search, we calculate the closest point $p_{k_i}^k$ in the road map to the estimated point $p_{k_i}^k$. Given a search window $[-d_w, d_w]$, we find all road points $Y^k$ from $i_{id} - m$ to $i_{id} + n$ in the map whose distances to $p_{k_i}^k$ are within this window:

$$Y^k = [p_{i_{id} - m}, \ldots, p_{i_{id} - 1}, p_{k_i}, p_{i_{id} + 1}, \ldots, p_{i_{id} + n}]^T,$$

with $\text{argmin}(\sum_{j=0}^{m} \Delta s_{id_{id} - j} - d_w)$, \quad (15)

and $\text{argmin}(\sum_{j=0}^{n} \Delta s_{id_{id} + j} - d_w)$

Because of the drift error of an IMU, $p_{k_i}^k$ is the possible vehicle position, so we assume that the accurate position drops in the set of all possible road points, $Y^k$. We use a time series matching method DTW to estimate the optimum position. More details about DTW are in [16]. Algorithm 1 describes the algorithm.

**Algorithm 1: Application of DTW in Localization**

**Initialize Variables:**
Set the distance vector $D^k$ to empty

**Initialize Constant:**
Cutoff frequency for a low-pass filter: $f_c$

**Algorithm Loop:**
FOR $l = i - m; i + n$

Similar to Equation (14), from the road map find a point cluster $B^k$ such that:

$$B^k = [p_{l-m}, \ldots, p_{l-1}, p_l, p_{l+1}, \ldots, p_{l+n}]^T,$$

with $\text{argmin}(\sum_{j=0}^{m} \Delta s_{id_l - j} - d_{k_{id}})$

Get the pitch data vector $y^k = [\theta_{id_{l-m}}, \ldots, \theta_{id_{l-1}}, \theta_{id_l}]$ from $B^k$

Normalize $y^k$ and $x^k$ using Z-Score method

Interpolate or sample $y^k$ to $y^k'$ to have the same length with $x^k$

Low-pass filter $y^k$ and $x^k$ with Fast Fourier Transform (FFT) by $f_c$

Calculate the distance $d_l$ between filtered sequences $y^k'$ and $x^k'$ by DTW

Store $d_l$ in the distance vector $D^k$

END FOR

RETURN $D^k$
Normalization is necessary before executing DTW as $y^k$ and $x^k$ may be on different magnitudes. Moreover, to reduce the noise in the measurement related to velocity, we use FFTW [45] to perform Fast Fourier Transform (FFT) in our work. FFTW is a C subroutine library for computing the discrete Fourier transform in one or more dimensions and is thought to be the fastest Fourier Transform tool among all the open source FFT tools.

2.4. Estimation of Navigation State from the Matching Model

In almost all DTW applications, the best-fit matching subsequence has the smallest distance value. In the pitch-based localization, pitch angle measurements contain bias noise and are affected by vehicle speed. There even might be different vehicles or sensors when generating the road map and processing online localization. Moreover, most parts of a road have slight slopes, so the magnitude of pitch angle measurements is small. Thus, the smallest value in the $D^k$ does not necessarily mean that we can find the best subsequence and the right location. Instead, we choose $n$ minima and choose the one whose position is closest to navigation state estimation from DR, which can constraint time series matching results. This principle makes the final navigation state more robust and precise. We use an algorithm detecting the valleys in data to calculate $M^k$ and $m^k$, shown in Algorithm 2.

**Algorithm 2: Calculation of Minima**

Initialize Constant:
Number of minima: $n$
Length of $x^k$: $l_x = l_t * f$
The smallest index distance between minima: $\text{mid} = \text{ROUND}(l_x/n)$

Algorithm Loop:
Find all valleys $v^k$ (smaller than both the left and right neighbors) in $D^k$ and their corresponding indices $i^k$
Sort $v^k$ by their values in ascending order and get new $v^k$ and its corresponding $i^k$
FOR $\text{ind} = 0$: length of $i^k$
Keep indices that is not in the range $\left[i^k[\text{ind}] - \text{mid}/2, i^k[\text{ind}] + \text{mid}/2\right]$ 
Keep current index
END FOR
Remove these indices from $i^k$
Sort back $i^k$ by their occurrence
Set $M^k = B^k[i^k]$
Calculate the spatial distance between the road points in $M^k$ and $p^k$
Find the smallest spatial distance and the corresponding road point $m^k$ in $M^k$
Return $m^k$

2.5. Integration Model

We obtain the road point $m^k$’s $\text{lat}_{idu}$, $\text{lon}_{idu}$, and $h_{idu}$ from the map. Equation (2) is augmented to:

$$X^k = \begin{bmatrix} (\delta \psi_{IMU}^n)^T, (\delta v_{IMU}^n)^T, (\delta P_{IMU}^n)^T, \\
(\delta P_{IMT}^n)^T, (\delta v_{IMT}^n)^T, (\delta P_{IMT}^n)^T, \\
\delta \kappa_{wheel}^k, (\delta \iota_{wheel}^k)^T \end{bmatrix}^T$$

(16)
including an error $\delta p_{TM}^n$ of position $(lat_{TM}^n, lon_{TM}^n, h_{TM}^n)$ estimated by TSS matching model. Equation (7) is also augmented as:

$$Z^k = \left[ (v_n^k - v_{wheel,n,k}^k)^T (p_{IMU,n,k}^n - p_{TM,n,k}^n)^T \right]^T$$

with $p_{TM,n,k}^n = [lat_{TM,n,k}^n, lon_{TM,n,k}^n, h_{TM,n,k}^n]^T$

$$-v_{wheel,n,k}^k \times \delta \varphi_{IMU,n,k}^n + \delta v_{IMU,n,k}^n - M^k \delta \alpha_{wheel}^k - C^n_b (\omega_{nb}^b \times \iota_{wheel}^b)$$

so $H^k$ becomes:

$$H^k = \begin{bmatrix} -v_{wheel,n,k}^k \times & 0_{3x3} & 0_{3x9} & -M^k & -C^n_b (\omega_{nb}^b \times ) \\ I_{3x3} & 0_{3x6} & I_{3x3} & 0_{5x12} & 0_{12x21} & 0_{3x6} \\ \end{bmatrix}$$

(18)

$F^k$ and $G^k$ are changed to the following:

$$F^k = \begin{bmatrix} F^k_{1:9 \times 1:9} & 0_{3x3} & F^k_{1:9 \times 10:21} \\ 0_{12x21} & 0_{3x6} & \end{bmatrix}$$

(19)

$$G^k = \begin{bmatrix} G^k \\ 0_{3x6} \end{bmatrix}$$

(20)

Then, EKF is applied again by repeating Equations (9)–(12), but here $Q^k$ and $P^k$ should also include the error density information and error covariance of $\delta p_{TM}^n$. Finally, we get the integrated navigation state like Equation (13).

3. Experiments

In our work, field experiments are carried out to test our integrated navigation system. Our system and algorithms are implemented in C++ on a Windows10 PC with 3.70 GHz CPU and 16 GB RAM. The test vehicle (see Figure 2a) we use is a Chery Tiggo car equipped with a Novatel SPAN-CPT, which couples GNSS and IMU to provide consistent navigation support. An RTK-GPS (centimeter-level accuracy) is used to gather ground truth positions for evaluation. The IMU components are comprised of a Fiber Optic Gyros (FOG) and a MEMS accelerometer, its performance metrics are shown in Table 1.

| Table 1. SPAN-CPT inertial measurement unit (IMU) sensor specifications. |
|-------------|-----------------|-----------------|
|             | FOG             | MEMS Accelerometer |
| Bias Offset | $\pm 20^\circ$/hr | $\pm 50$ mg      |
| Turn on to turn on bias repeatability (compensated) | $\pm 5^\circ$/hr | $\pm 0.75$ mg |
| In run bias variation, at constant temperature | $1^\circ$/hr @ 1$\sigma$ | $0.25$ mg @ 1$\sigma$ |
| Scale factor error (total) | 1500 ppm, 1$\sigma$ | 4000 ppm, 1$\sigma$ |
| Scale factor linearity | 1000 ppm, 1$\sigma$ |
| Temperature dependent SF variation | 500 ppm, 1$\sigma$ | 1000 ppm, 1$\sigma$ |
| Random walk | $0.0667^\circ$/sqrt Hz @ 1$\sigma$ | $55$ ug/sqrt Hz @ 1$\sigma$ |
| Max input | $\pm 375^\circ$/sec | $\pm 10$ g |

An encoder is used to measure the speed of a vehicle. In our experiments, we travelled twice on a 6.5-km round route (see Figure 2b) on the Huashan Road in Wuhan, China. From Figure 2c, we can see that in our test area there are both sharp and slight changes in height; these various features help test the robustness of our system.

The vehicle travelled along the route (called reference trip) at about 20 km/h to generate the reference map, where 3D positions (latitude, longitude, height), attitudes (roll, pitch, yaw) and the encoder measurements (distances between two adjacent positions) were recorded at 100 Hz. Then, the reference map was stored in an R-Tree as described above in Section 2.1. In the second trip (called measurement trip), the vehicle travelled on the route at various speeds and recorded raw IMU measurements (angle increments, velocity increments), 3D positions, attitudes, and the distances at
100 Hz. The 3D positions and attitudes were used as ground truth for evaluation. Then, the encoder’s distance measurements were converted to velocity and integrated with IMU for the DR model as described above in Section 2.2.

Figure 3 shows the vehicle’s velocity at n-frame in the measurement trip. Around time 3 min and 5 min, the vehicle decelerates to about 0 m/s, stops for a while, and accelerates again. This is because of the traffic lights and the U-turn in the route (see Figure 2b). Unlike most existing studies, which assume that the vehicle travels at a steady speed, the vehicle’s various speed features in our work also help prove the robustness of our navigation system.

Figure 2. The vehicle and the test region. (a) The test vehicle. The wheel encoder is mounted on left rear wheel, the Global Navigation Satellite System (GNSS) antenna is on the roof, and SPAN-CPT is in the mass center of the vehicle. (b) An overhead satellite image of the test region. The mapped route is shown in yellow. To be more specific, two areas in the red frame are zoomed in. The red arrow indicates the starting position and the driving direction. (c) The road height of the test route, and distance is calculated from the starting position.

Figure 3. The vehicle’s velocity in the measurement trip at north (red), east (green), down (blue) directions, and driving direction (black).
To test the repeatability of terrain features of the two trips, pitch values are shown in Figure 4. Note that the pitch values in the measurement trip are the ground truth from the integrated SPAN-CPT and RTK-GPS. Because of the good performance of FOG and centimeter-level accuracy of RTK-GPS, the pitch values in the two trips are almost the same in Figure 4a. In Figure 4b we can see the effect of the vehicle’s stops in the measurement trip, pitch values remain unchanged around time 3 min and 5 min.

![Figure 4. The vehicle pitch values in the reference trip and the measurement trip. (a) Pitch values versus distance, calculated from the starting position in the reference trip (red) and in the measurement trip (black). (b) Pitch values versus travelling time in the measurement trip.](image)

In the experiments, we conduct DR calculations at 100 Hz, DTW matching at 1 Hz, and record the final integrated navigation state at 1 Hz. In our system, there are several predefined parameters, like duration $t_d$, distance threshold $d_{thre1}$ and $d_{thre2}$ in Figure 1. Table 2 shows the predefined parameters in our system and their values in our experiments.

| Parameter | Values | Description |
|-----------|--------|-------------|
| $d_w$     | 10 m   | Search window $[-d_w, d_w]$ |
| $f_c$     | 8 Hz   | Cutoff frequency in the FFT |
| $t_d$     | 2 s    | Duration in which the pitch values are matched |
| $d_{thre1}$ | 5 m   | Distance threshold that the vehicle moves in $t_d$ |
| $d_{thre2}$ | 0.5 m | Distance threshold that the vehicle moves in the 0.01 s, as our system runs with 100 Hz |
| $n$       | 8      | Number of minima |

Figure 5 illustrates the comparison results of the horizontal vehicle position estimation from our integrated model and the pure DR model without being integrated by DTW matching during the measurement trip. We can see that both our system output and the DR model output have similar pattern as the ground truth, but our system bias is much smaller than that of the DR model. In the beginning of the measurement trip in Figure 5e, our system and the DR model have good correspondence with the ground truth. After a while, in Figure 5d the DR model gradually deviates from the ground truth trajectory and in Figure 5b,c it completely deviates from the lane. In Figure 5b there is a U turn where the DR model overshoots the lane during position calculation. This phenomenon
is common in DR, which is caused by lever arm effect and accumulated error in INS and odometer. At the end of the trip, we can see from the Figure 5d,e that the bias of the DR model output decreases and leads to a better correspondence with the ground truth. Our system has a much better performance than the DR model, no matter if on a straight lane or U turn, and our system output keeps good positioning accuracy.

**Figure 5.** Comparison results of our integrated model and the pure Dead Reckoning (DR) model. Green dots on the map are the ground truth, red dots the results of our integrated model, and yellow dots the results of the DR model. (a) The results for the whole measurement trip. (b–e) the results for 4 specific areas.

Figure 6 shows a boxplot of the vehicle positioning error in the DR model and our system, and a boxplot of the runtime of our system. The data for the boxplot is at 1 Hz for both the DR model and our system. Our system’s median error was about 0.3 m while that of the DR model is about 100 m. Moreover, the median run time of our system is 12 ms, and for all the time steps (cycles) the runtime is less than 100 ms. Therefore, our system meets real-time requirements as the cycle is 1 s.

Figure 7 shows the time series of vehicle state errors in the DR model and our system. In Figure 7a the DR model’s position error is increasing with time in the first half of the trip, after reaching a peak, it keeps decreasing. However, Position Dilution of Precision (PDOP) for the DR model is reflected by the error covariance matrix $P_k$ in Equation (12), increasing over time. The DR model position error, as calculated by the ground truth, decreased unexpectedly in the second half of the trip, possibly due to random values in the EKF. Dilution of Precision (DOP) is not consistent with the error calculated from the ground truth, nor the error calculated from the velocity and attitude. Figure 7b shows the velocity error in the driving direction. The DR model’s velocity error is much larger than that of our system. In Table 1, we can see that the accelerometer in SPAN-CPT is at a MEMS level, its large drift error leads to deviation quickly during operation. However, in Figure 7d–f, errors of pitch, roll and yaw angles of the DR model are slightly larger than those of our system, which is due to the good quality of FOG in SPAN-CPT. As the pitch data is matched in our work, Figure 7c shows the time series of pitch data for the DR model, our system and the ground truth. We can see that both the DR model and our system achieved a similar pattern with the ground truth, and pitch data from our system is slightly closer to the ground truth than the DR model. The similarity and accuracy of the pitch data from our system
benefit our matching process, help find a precise estimation of the vehicle’s position, and lead to a positive constraint on the drift errors of the gyroscope and accelerometer in the integration model. We can see that it is a positive recursive procedure and makes our system more robust.

![Boxplot of vehicle positioning error](image)

**Figure 6.** The error of localization from the pure DR model and our integrated model, compared with the ground truth from RTK-GPS, and the runtime of our integrated model.

These results indicate that DR model is vulnerable to the biases of the gyroscope, accelerometer and odometer. Our system compensates for these biases by integrating position estimation with higher accuracy for improved positioning performance. The FFT and DTW processes in our work are implemented with FFTW and fast DTW, furthermore, we extensively optimize their codes to make them even faster for real-time computing. Our system delivers high performance and efficiency with a median position error of 0.3 m and a median runtime of 12 ms.
4. Discussion

In our system, we use a TSS-matching model in a DTW algorithm to find the possible positions of a vehicle. We choose \( n \) positions where the DTW distance is minima instead of the smallest distance of \( D^k \) in Algorithm 2. Then through the constraint in Algorithm 2, we find the best positioning estimation from \( n \) minima. The smallest DTW distance means that its corresponding subsequence on the reference map matches our calculated pitch subsequence the best. But it may not be the right position. This is reasonable because compared with other applications of TSS matching, amplitudes of pitch values are small, pitch values are vulnerable to disturbances, and for flat road or straight slope the pitch curve may have less features, so subsequences of pitch data at different positions on the reference map may have similar patterns and have similar DTW distances. In this situation, we need an extra constraint, as in Algorithm 2, to find the right positioning estimation. Figure 8 supports our assumption by presenting the position error of using the smallest DTW distance of \( D^k \) instead of minima in our system. We can see that there is a wrong positioning estimation at around 7 min. The error caused by this erroneous positioning estimation accumulates and affects all cycles afterwards.

![Figure 8. Boxplot and time series of the position error of the integrated model using smallest DTW distance.](image)

We also test how these parameters affect the performances of our system by control variates method. A series of tests named Test\( k \), \( k = 1, 2, \ldots, 5 \) are implemented with parameters values shown in Table 3. Parameter values of the control group are from Table 2. The parameter values with gray background are different from the corresponding values in the control group. Note that \( d_{thr2} \) is not tested, because we think it is large enough in our work. Figure 9 shows the position error and runtime of the tests and the control group. We can see that if we increase \( d_{w} \) from 10 to 20 m, the boxplot of the position error slightly changes, but the runtime increases a lot. \( d_{w} \) increases mean that the number of road points \( Y^k \) in Equation (15) increases and the iteration in the loop in Algorithm 1 also increases, which increases time consumption. Even though \( d_{w} \) is increased from 10 to 20 m, considering our system’s positioning error which is less than 2 m, the search window (−10 m, 10 m) is large enough to find the right position. Therefore, Test1 results in a similar boxplot of position error with the control group.
Table 3. Predefined parameters and their values for the five testing experiments and the control group.

| Parameter | $d_w$ | $f_c$ | $t_d$ | $d_{\text{thr1}}$ | $d_{\text{thr2}}$ | $n$ |
|-----------|-------|-------|-------|-------------------|-------------------|-----|
| Test1     | 20 m  | 8 Hz  | 2 s   | 5 m               | 0.5 m             | 8   |
| Test2     | 10 m  | 5 Hz  | 2 s   | 5 m               | 0.5 m             | 8   |
| Test3     | 10 m  | 8 Hz  | 5 s   | 5 m               | 0.5 m             | 8   |
| Test4     | 10 m  | 8 Hz  | 2 s   | 10 m              | 0.5 m             | 4   |
| Test5     | 10 m  | 8 Hz  | 2 s   | 5 m               | 0.5 m             | 8   |
| Control Group | 10 m  | 8 Hz  | 2 s   | 5 m               | 0.5 m             | 8   |

In Test2, the cutoff frequency $f_c$ in the low-pass filter in Algorithm 2 decreases from 8 to 5 Hz; from Figure 9 we can see that the positioning result is slightly worse than the control group, and the running time is almost the same. $f_c$ has no effect on runtime. Lower $f_c$ means a smoother pitch subsequence, but less small features, which is not good for finding the best position estimation.

From Figure 1 and Equation (14), we can see that $t_d$ controls the length of the pitch subsequence. In Test3, $t_d$ increases from 2 to 5 s, which means more pitch data are involved in the matching procedure so that the DTW algorithm takes more time. The positioning accuracy is similar to that of the control group, which means that pitch data in the duration of 2 s are enough to find the best position estimation.

In Figure 1, $d_{\text{thr1}}$ is a distance threshold; if the accumulated distance that the vehicle moves within $t_d$ is less than $d_{\text{thr1}}$, the navigation state from the DR model is assumed to be the vehicle’s final state at the current time step. This constraint is for low speed situations, like waiting for traffic lights or obstacle avoidance. In Figure 3, we can see that the vehicle stops twice, and the second stop lasts for about 1 min. During these situations, the vehicle uses the DR model directly to calculate its navigation state. If $d_{\text{thr1}}$ is increased from 5 to 10 m (the situation lasts for longer for the vehicle), our integration model works less, and the vehicle takes the direct output from the DR model in more cycles. From Figure 9, we can see that in Test4 there is a larger position error and the runtime is similar to that of the control group.

From Algorithm 2, we can see that the number $n$ of minima has a direct effect on the choice of possible position estimations in matrix $M^k$. Larger $n$ means more possible estimations using the constraint in Algorithm 2 to obtain the best estimation $m^k$. In Test 5, we decrease $n$ from 8 to 4.
From Figure 9, we can see that 8 minima can produce a more accurate position estimation and $n$ has a very limited effect on runtime.

5. Conclusions

This paper proposed an integrated model for real-time localization. This integrated model deploys a DR model and a DTW algorithm. The DR model calculates the vehicle’s navigation state (position, velocity and attitude) by IMU and odometer. However, the DR model is vulnerable to the drift error of gyroscope and accelerometer in the IMU and the accumulated error of odometer. Using the prediction of the vehicle’s navigation state from the DR model, we take advantage of R-Tree to find the closest road point in the reference terrain map efficiently. Given a search window centering on this road point, we match our calculated pitch data over a period with the pitch data stored in the reference map using DTW algorithm to find the best-fit position estimation. This position estimation is integrated with the DR model by EKF to get final navigation state. The integration constrains IMU drift error and accumulated odometry error for improved positioning accuracy. Field experiments were conducted to test the integrated system, the terrain map covered a route more than 6 km and the measurement trip lasted for more than 10 min. In our work, we used a high-performance MEMS-level IMU sensor which helped the DR system provide good estimations of positions. In the future, we will try low-cost MEMS-level IMU sensors in our system to make our system more practical. Experimental results show that our integrated system can accurately (mean position error: 0.3 m) and efficiently (median runtime: 12 ms) localize the vehicle.

Author Contributions: Conceptualization, H.Z. and W.L.; methodology, H.Z.; software, W.L.; validation, C.Q. and H.Z.; writing H.Z.; supervision, B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1600600, in part by the National Natural Science Foundation of China under Grant 41801377, 41801344 and in part by National Natural Science Foundation of China under Grant U1764262.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wanninger, L.; Beer, S. BeiDou satellite-induced code pseudorange variations: Diagnosis and therapy. *GPS Solut.* 2015, 19, 639–648. [CrossRef]
2. Li, X.; Ge, M.; Dai, X.; Ren, X.; Fritsche, M.; Wickert, J.; Schuh, H. Accuracy and reliability of multi-GNSS real-time precise positioning: GPS, GLONASS, BeiDou, and Galileo. *J. Geod.* 2015, 89, 607–635. [CrossRef]
3. Hide, C.; Moore, T.; Smith, M. Adaptive Kalman filtering for low-cost INS/GPS. *J. Navig.* 2003, 56, 143–152. [CrossRef]
4. Ahmed, H.; Tahir, M. Terrain-based vehicle localization using low cost MEMS-IMU sensors. In Proceedings of the 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring), Nanjing, China, 15–18 May 2016; pp. 1–5.
5. Tin Leung, K.; Whidborne, J.F.; Purdy, D.; Dunoyer, A. A review of ground vehicle dynamic state estimations utilising GPS/INS. *Veh. Syst. Dyn.* 2011, 49, 29–58. [CrossRef]
6. Malvezzi, M.; Vettori, G.; Allotta, B.; Pugi, L.; Ridolfi, A.; Rindi, A. A localization algorithm for railway vehicles based on sensor fusion between tachometers and inertial measurement units. *Proc. Inst. Mech. Eng. Part F J. Rail Rapid Transit* 2014, 228, 431–448. [CrossRef]
7. Georgy, J.; Karamat, T.; Iqbal, U.; Noureldin, A. Enhanced MEMS-IMU/odometer/GPS integration using mixture particle filter. *GPS Solut.* 2011, 15, 239–252. [CrossRef]
8. Levinson, J.; Montemerlo, M.; Thrun, S. Map-based precision vehicle localization in urban environments. *Robot. Sci. Syst.* 2007, 4, 1.
9. Se, S.; Lowe, D.; Little, J. Vision-based mobile robot localization and mapping using scale-invariant features. In Proceedings of the Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164), Seoul, Korea, 21–26 May 2001; pp. 2051–2058.
10. Filipenko, M.; Afanasyev, I. Comparison of various slam systems for mobile robot in an indoor environment. In Proceedings of the 2018 International Conference on Intelligent Systems (IS), Funchal-Madeira, Portugal, 25–27 September 2018; pp. 400–407.

11. Dean, A.; Martini, R.; Brennan, S. Terrain-based road vehicle localisation using particle filters. Veh. Syst. Dyn. 2011, 49, 1209–1223. [CrossRef]

12. Li, T.; Yang, M.; Li, H.; Deng, L.; Wang, C. A terrain-based vehicle localization approach robust to braking. IEEE Trans. Intell. Transp. Syst. 2018, 20, 2923–2932. [CrossRef]

13. Ahmed, H.; Tahir, M.; Ali, K. Terrain based GPS independent lane-level vehicle localization using particle filter and dead reckoning. In Proceedings of the 2016 IEEE 84th Vehicular Technology Conference (VTC-Fall), Montreal, QC, Canada, 18–21 September 2016; pp. 1–5.

14. Paparrizos, J.; Gravano, L. K-shape: Efficient and accurate clustering of time series. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Australia, 31 May–June 2015; pp. 1855–1870.

15. Faloutsos, C.; Ranganathan, M.; Manolopoulos, Y. Fast subsequence matching in time-series databases. In ACM SIGMOD Record; ACM: New York, NY, USA, 1994; Volume 23.

16. Rakthanmanon, T.; Campana, B.; Mueen, A.; Batista, G.; Westover, B.; Zhu, Q.; Zakaria, J.; Keogh, E. Searching and mining trillions of time series subsequences under dynamic time warping. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012; pp. 262–269.

17. Echihabi, K.; Zoumpatianos, K.; Palpanas, T.; Benbrahim, H. The lernaean hydra of data series similarity search: An experimental evaluation of the state of the art. Proc. VLDB Endow. 2018, 12, 112–127. [CrossRef]

18. Hsu, C.-J.; Huang, K.-S.; Yang, C.-B.; Guo, Y.-P. Flexible dynamic time warping for time series classification. Procedia Comput. Sci. 2015, 51, 2838–2842. [CrossRef]

19. Müller, M. Analysis and retrieval techniques for motion and music data. In Eurographics (Tutorials); Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; pp. 249–255.

20. Zhang, Y.; Glass, J.R. An inner-product lower-bound estimate for dynamic time warping. In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 22–27 May 2011; pp. 5660–5663.

21. Wobbrock, J.O.; Wilson, A.D.; Li, Y. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In Proceedings of the 20th annual ACM Symposium on User Interface Software and Technology, Newport, RI, USA, 7–10 October 2007; pp. 159–168.

22. Bahlmann, C.; Burkhardt, H. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. IEEE Trans. Pattern Anal. Mach. Intell. 2004, 26, 299–310. [CrossRef] [PubMed]

23. Kim, H.; Ahn, C.R.; Engelhaupt, D.; Lee, S. Application of dynamic time warping to the recognition of mixed equipment activities in cycle time measurement. Autom. Constr. 2018, 87, 225–234. [CrossRef]

24. Rebbapragada, U.; Protopapas, P.; Brodley, C.E.; Alcock, C. Finding anomalous periodic time series. Mach. Learn. 2009, 74, 281–313. [CrossRef]

25. Keogh, E.; Wei, L.; Xi, X.; Vlachos, M.; Lee, S.-H.; Protopapas, P. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures. VLDB J. 2009, 18, 611–630. [CrossRef]

26. Varatharajan, R.; Manogaran, G.; Priyan, M.K.; Sundarasegar, R. Wearable sensor devices for early detection of Alzheimer disease using dynamic time warping algorithm. Clust. Comput. 2018, 21, 681–690. [CrossRef]

27. Zhang, X.L.; Wang, M.; Liu, K.; Xie, J.; Xu, H. Using NDVI time series to diagnose vegetation recovery after major earthquake based on dynamic time warping and lower bound distance. Ecol. Indic. 2018, 94, 52–61. [CrossRef]

28. Nakagawa, K.; Imamura, M.; Yoshida, K. Stock price prediction using k-medoids clustering with indexing dynamic time warping. Electr. Commun. Jpn. 2019, 102, 3–8. [CrossRef]

29. Lin, J.; Keogh, E.; Wei, L.; Lonardi, S. Experiencing SAX: A novel symbolic representation of time series. Data Min. Knowl. Discov. 2007, 15, 107–144. [CrossRef]

30. Sun, Y.; Li, J.; Liu, J.; Sun, B.; Chow, C. An improvement of symbolic aggregate approximation distance measure for time series. Neurocomputing 2014, 138, 189–198. [CrossRef]
31. Abughali, I.K.; Minz, S. Binarizing change for fast trend similarity based clustering of time series data. In Proceedings of the International Conference on Pattern Recognition and Machine Intelligence, Warsaw, Poland, 30 June–3 July 2015; pp. 257–267.

32. Dong, X.-L.; Gu, C.-K.; Wang, Z.-O. Research on shape-based time series similarity measure. In Proceedings of the 2006 International Conference on Machine Learning and Cybernetics, Dalian, China, 13–16 August 2006; pp. 1253–1258.

33. Lara, J.A.; Pérez, A.; Valente, J.P.; Lópe-Illescas, Á. Comparing time series through event clustering. In Proceedings of the 2nd International Workshop on Practical Applications of Computational Biology and Bioinformatics (IWPACBB 2008), Salamanca, Spain, 22–24 October 2008; pp. 1–9.

34. Vemulapalli, P.K.; Monga, V.; Brennan, S.N. Robust extrema features for time-series data analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1464–1479. [CrossRef] [PubMed]

35. Vemulapalli, P.K.; Monga, V.; Brennan, S.N. Optimally robust extrema filters for time series data. In Proceedings of the 2012 American Control Conference (ACC), Montreal, QC, Canada, 27–29 June 2012; pp. 2189–2195.

36. Martini, R.D. GPS/INS Sensing Coordination for Vehicle State Identification and Road Grade Positioning. Ph.D. Thesis, Pennsylvania State University, State College, PA, USA, 2006.

37. Vemulapalli, P.K.; Dean, A.J.; Brennan, S.N. Pitch based vehicle localization using time series subsequence matching with multi-scale extrema features. In Proceedings of the 2011 American Control Conference, San Francisco, CA, USA, 29 June–1 July 2011; pp. 2405–2410.

38. Laftchiev, E.I.; Lagoa, C.M.; Brennan, S.N. Vehicle Localization using in-vehicle pitch data and dynamical models. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 206–220. [CrossRef]

39. Zhang, H.; Qin, S.; Ma, J.; You, H. Using Residual resampling and sensitivity analysis to improve particle filter data assimilation accuracy. *IEEE Geosci. Remote. Sens. Lett.* **2013**, *10*, 1404–1408. [CrossRef]

40. Ding, H.; Trajcevski, G.; Scheuermann, P.; Wang, X.; Keogh, E. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proc. VLDB Endow.* **2008**, *1*, 1542–1552. [CrossRef]

41. Bonnor, N. Principles of GNSS, Inertial, and multisensor integrated navigation systems. *J. Navig.* **2014**, *67*, 191–192. [CrossRef]

42. Guttman, A. *R-Trees: A Dynamic Index Structure for Spatial Searching*; ACM: New York, NY, USA, 1984; Volume 14.

43. Shin, E.-H. Estimation Techniques for Low-Cost Inertial Navigation. *UCGE Rep.* 2005. Available online: https://prism.ucalgary.ca/handle/1880/103387 (accessed on 8 August 2020).

44. Ben, Y.; Yin, G.; Gao, W.; Sun, F. Improved filter estimation method applied in zero velocity update for SINS. In Proceedings of the 2009 International Conference on Mechatronics and Automation, Changchun, China, 9–12 August 2009; pp. 3375–3380.

45. Frigo, M.; Johnson, S.G. The design and implementation of FFTW3. *Proc. IEEE* **2005**, *93*, 216–231. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).