**Sparse High-noise GPS Trajectory Data Compression and Recovery based on Compressed Sensing**

Mingjun Zhu1, Guan Yuan1,2\*, Zhixiao Wang1, Yanmei Zhang1, Shixiong Xia1

1. School of Computer Science and Technology, China University of Mining and Technology, No. 1 Daxue Road, Xuzhou, Jiangsu, China, 221116.

2. Jiangsu Key Laboratory of Mine Mechanical and Electrical Equipment, China University of Mining and Technology, No. 1 Daxue Road, Xuzhou, Jiangsu, China, 221116.

\*Corresponding to: Guan Yuan (yuanguan@cumt.edu.cn); Tel.: +86-137-767-87137

**Abstract** With the extensive use of location based devices, trajectories of various kind of moving objects can be collected. As time going on, the amount of trajectory data increases exponentially, which brings (presents)a series of problems in storage, transmission and analysis. Moreover, GPS trajectories are never perfectly accurate and sometimes may have high noise. Therefore, how to overcome these problems becomes an urgent task. In this paper, an adaptive noise filtering, trajectory compression and recovery algorithm based on Compressed Sensing is proposed. Firstly, a noise reduction model is introduced to filter the high noise in the GPS trajectory. Secondly, we obtain the compressed data by an improved compression algorithm. Thirdly, an adaptive GPS trajectory data recovery algorithm is used to restore the original GPS trajectory. Comprehensive experiments on real datasets show that the Compressed Sensing based algorithm can not only get a good noise filtering, but also achieve high compression ratio and recovery performance compared to current algorithms.

**Keywords**: GPS trajectory; high noise; compression; recovery; Compressed Sensing

1. **Introduction**

In recent years, with the rapid growth of GPS-equipped mobile devices, sensor network and wireless communication technologies, various kinds of moving objects can be traced all over the world. The popularity of these devices and technologies has leading(led) to an exponential growth in the amount of trajectory data as time going on. For instance, there are 5000 taxis in a city and each taxi is tracked by sampling its position once every five seconds, so we will overwhelm(cost) two GBs of storage capacity to store a single day trajectory data. These data are the foundations for analyzing activities and patterns for moving objects. However, the enormous volume of data has brought several problems [1], such as: **i.** It is quite expensive and time-consuming to transmit these large amounts of trajectory data. **ii.** It is computationally expensive operations to query and extract useful patterns from these large amounts of trajectory data. **iii.** GPS trajectories are often with much redundant and trivial data that waste storage and cause increased disk I/O time. These issues can be addressed by reducing the size of trajectory data. Therefore, the aim of data compression technique is to decrease the occupied memory space and improves the transmission, storage and processing by reducing data volume without obviously losing information, or by reorganizing data with certain strategies to reduce the redundancy and memory cost.

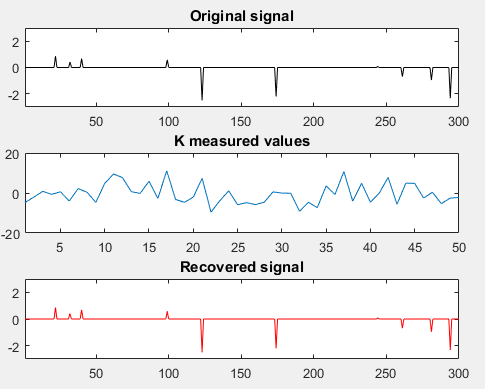
The Nyquist sampling theorem states that if we want the original signal to be recovered from the sampled signal without distortion, the sampling frequency should be greater than twice the maximum frequency of the signal. However, considering that the GPS trajectories are mostly sparse, the Compressed Sensing (CS) can recover sparse signals of far fewer samples [2-4]. The purpose of compressed sensing is to allow us to obtain fewer than the previously required number of samples while still perfectly (or nearly perfectly) recovering the signal. Moreover, for the third issue, various filtering technologies have been applied to smooth the noise, such as mean filter [5], median filter [6], Kalman filter [7], and Particle filter [8].

As discussed above, we propose a sparse high-noise GPS trajectory data compression and recovery based on Compressed Sensing (SGTCR-CS). Experiments on real datasets demonstrate their efficiency and effectiveness, and the results are more practically significant.

A simple example is given in Figure 1 to describe the flow of CS in this article. The processes of CS are shown in Figure 1(a). *x* is the raw signal, and the *y* is the sampled data. We recovery the signal with the sampled data, and evaluate the performance of the approach by compare *x* and the recovered signal *x~*. The specific example is shown in Figure 1(b). The original signal had 300 values, and the we sampled only 50 measurements as the compressed results. As we can see from the Figure, the recovered signal is almost the same with the original signal.



(a)



(b)

**Figure 1.** An example of trajectory compressing based on Compressed sensing. (a) The process of compression and recovery. (b) The compression and recovery of a sparse time-domain signal.

In the example above, the amount of data is very small and the signal is error-free, but in real GPS trajectories, the amount of data is very lager and noise is unavoidable. Therefore, a method of sparse high-noise GPS trajectory data compression and recovery based on Compressed Sensing is proposed in this article to solve these problems.

To summarize, the main contributions of this paper are as follows:

(1) A noise reduction model is employed to smooth the noise and reduce the impact of noise on the trajectory analysis.

(2) An improved Compressed Sensing algorithm which based on sparse transformation is used to compress data with random linear projection measurements.

(3) Original signals are recovered by a reconstruction algorithm based on sparsity adaptive matching pursuit.

(4) Comprehensive experiments on real trajectory dataset have conducted to analysis（analyze） the effectiveness and efficiency of the proposed algorithms.

The rest of the paper is organized as follows. In Section 2, some related works are surveyed and reviewed to address the trajectory compression problem. A noise reduction model is described in Section 3. Section 4 introduces the Compressed Sensing model. In Section 5, the CS-based sparse high-noise GPS trajectory data compression and recovery algorithm is introduced in detail. The numerical results obtained from our analytical models and simulations are compared in Section 6. Finally, Section 7 draws conclusions and points out some possible research opportunities.

**2. Related works**

The rapid development of various subjects(technical support) and the wide usage of Internet provide a great deal of technical supports and a powerful motivation for the rapid development of trajectory data compression technologies. Currently, the traditional trajectory compression methods can be classified into 3 categories, according to their compression ideas.

(1) Shape based line simplification

Many researchers have devoted their talent to compress trajectories by deciding whether the sampling point is reserved based on distance (such as perpendicular distance, Synchronized Euclidean distance and so on) since 1973. In literature [9], Douglas and Peucker proposed an algorithm called Douglas-Peucker (DP) algorithm, which recursively selects the point whose perpendicular distance is greater than given threshold until all points reserved meet the condition. Its advantage is the translation and rotation invariance, namely, when the trajectory and threshold have been given, the compression result is certain. However, there is an apparent drawback about DP algorithm, which only considers spatial information but neglect temporal information in trajectory data. To overcome this shortcoming, Meratnia et al [10]. put forward a top-down time-ratio algorithm (TD-TR) which is a transformation of DP algorithm taking a full consideration of spatiotemporal characteristics by replacing perpendicular distance with SED distance [10,11]. This method has a higher accuracy than DP algorithm and also has the advantage of translation and rotation invariance. Both DP and TD-TR are not suitable for real-time applications, so Jonathan Muckell proposed the Spatial QUalIty Simplification Heuristic (SQUISH) method based on the priority queue data structure, which prioritizes the most important points in a trajectory stream [12]. Three years later, Muckell proposed a new version of SQUISH, called SQUISH-E (Spatial QUalIty Simplification Heuristic - Extended), which has the flexibility of tuning compression with respect to compression ratio and error [13].

(2) Attribute-based trajectory compression

The researches on compressing trajectory data based on velocity are not perfect by now. A famous velocity-based trajectory compression is top-down speed-based algorithm proposed by Meratnia [10]. The algorithm improved the existing compression techniques by exploiting the spatiotemporal information hiding in the time series, which can be made by analyzing the derived speeds at subsequent of the trajectory [10]. It is trivial to implement, but the accuracy is lower than DP and TD-TR algorithm. An online algorithm called Dead Reckoning algorithm proposed by Trajcevski [14] compressed trajectory by estimating the successor point through the current point and its velocity. It has a high execution efficiency for the computational complexity *O*(*n*). And the primary disadvantages are that it tends to achieve lower compression ratios than other techniques introduced in this section and it does not allow users to set the target compression ratio.

(3) Semantic-based trajectory compression

Considering the different environment where objects move, compressing trajectory in road network has attracted many attentions [15-19]. Schmid and Richter proposed a new and novel representation for trajectories that replaces trajectory data by the form of semantic information in road network [20]. Zheng proposed a new framework, namely paralleled road-network-based trajectory compression, to effectively compress trajectory data under road network constraints [21]. PRESS proposed a novel representation for trajectories to separate the spatial representation of a trajectory from the temporal representation and proposed a Hybrid Spatial Compression (HSC) algorithm and error Bounded Temporal Compression (BTC) algorithm to compress the spatial and temporal information of trajectories respectively.

Currently, many researches have been studied to pursuit high efficient and effective algorithms to compressing trajectory data. Among these algorithms, the main idea of line simplification is widely used to reduce the number of sampling points in trajectories by introducing a bounded error. However, this kind of algorithms may lose some important information after compression [10, 22]. The most well-known line simplification algorithm is Douglas-Peucker (DP) [9], which keeps the most important points of a polyline using the divide-and-conquer approach. However, the most drawback of DP is its neglect of temporal information. To take both spatial and temporal dimension into account, Meratnia et al [10]. replace the perpendicular Euclidean distance with Synchronous Euclidean Distance (SED) in DP algorithm, with which, compressed data is confirmed be superiority than the former ones. Beside DP algorithm, there are also various trajectory compression algorithm existing in the literature. Each offers a different trade off among compression time, compression ratio, and accuracy. Uniform sampling can archive the specified compression ratio by sampling trajectory at fixed time interval with few time cost, but it introduces large spatial and SED errors. To-Down Time Ratio (TD-TR) algorithm is a variant of DP algorithm with SED instead of spatial error [10]. It’s running time is *O*(*n2*). Opening Window (OW) algorithm is an online approximate line simplification algorithm by introducing a slide window [23]. OW algorithm runs with the window anchored at the first point, and continuously checks the forthcoming points until the spatial error is greater than the given threshold. The spatial error is the distance of the point to the line segment between the first point and the last point in the window, and it is executed iteratively until the last point of trajectory. The complexity of OW is O(*n2*). Opening Window Time Ratio (OW-TR) algorithm [10] is an extension of OW which takes temporal data into account and uses SED to represent the error. Like OW algorithm, the worst running time of OW-TR is O(*n2*). Dead Reckoning (DR) algorithm [24] is an efficient compression algorithm that considers not only spatial dimension but also velocity information. DR algorithm firstly marks the start point *ps* as the key point, and stores *ps* with its velocity in the compressed representation. Then the next point *pn* is estimated whether its location is within the SED threshold from *ps*. If true then continue the next point of *pn*, else *pn* is marked as the key point and stored to the compressed representation with its velocity. The DR algorithm will execute iteratively to the end of trajectory. The computation complexity of DR algorithm is O(*n*). However, the performance of these algorithms degrades significantly.

Because of the deficiencies of the above methods and the problems that exist in trajectory compression, we propose a GPS trajectory compression algorithm based on Compressed Sensing.

Compressed Sensing is a new signal processing theory in the field of signal processing in recent years proposed by D. Donoho (American Academy of Sciences), E. Candes (Ridgelet, Curvelet founder) and Chinese scientists T. Tao (2006 Fields Award winner). In literature [3], they proved that if the signal has sparsity in an orthogonal space, then the signal can be sampled at a lower frequency (well below the Nyquist sampling frequency) and the signal may be reconstructed with high probability. The core concept of Compressed Sensing is to try to reduce the cost of measuring a signal from the principle. For example, a signal contains a thousand data, then in accordance with the traditional signal processing theory, we need to do at least a thousand measurements to complete the recovery of this signal. This is equivalent to that we need a thousand equations to accurately solve a thousand unknowns. But the idea of Compressed sensing is to assume that the signal is sparse on a domain, then you can only do three hundred measurements to complete the recovery of this signal (This is equivalent to that we only need three hundred equations to accurately solve a thousand unknowns).

Moreover, noise is the main negative factor to compress the GPS trajectory. In real world, the multiple electromagnetic interference (EMI) sources are the main problem when we use GPS devices. Therefore, the real GPS trajectories we received may have high noise. So, before we compress the GPS trajectory, we can smooth the noise by applying some filtering techniques, such as Mean and Median Filters, Kalman Filter and Particle Filter.

Mean filter is a simple and intuitive method to reduce noise in images, trajectories and so on. The idea of mean filter is simply to replace each point in the trajectory with the mean value of its several predecessors, including itself in time. One disadvantage of mean filter it that it is influenced by exception value easily. One way to reduce the influence of exception value is to use a medium filter, and the medium filter just replaces the mean of mean filter with a medium. However, if the true value changes suddenly, the estimate from both the mean filter and the medium filter will change gradually.

Kalman filter is widely used and powerful, it can estimate the past, current even the future state of the signal. The basic idea of Kalman Filter is to use the minimum mean square error as the best estimation criterion. The state space model of signal and noise is used to update the estimation of the state variables using the estimated value of the previous time and the observed value of the current time, and the estimation of the current time is obtained value. The algorithm makes an estimate of the minimum mean square error for the signal that needs to be processed according to the established system equation and the observation equation.

For solving a large part of the problem, it is optimal, the most efficient and even the most useful. It has been widely used for more than 30 years, including robot navigation, control, sensor data fusion and even military aspects of radar systems and missile tracking and so on. In recent years has been applied to computer image processing, such as face recognition, image segmentation, image edge detection and so on.

The restrictions of Kalman filter are very harsh. It requires the accuracy of the system model as well as the known of the system error model and the observation error model. To deal with these problems, we choose Particle filter in our paper.

Particle filtering is developed from the late 90s of last century. The basic idea is to describe the probability distribution with random samples and then estimate the actual probability distribution by adjusting the size of each particle and the position of samples based on measurement. The estimated value is the mean of the samples. Particle filter effectively overcomes the shortcomings of Kalman filter. The superiority of Particle filter demonstrated in the nonlinear and non-Gaussian system decides its wide range of applications. Bayesian estimate and Monte Carlo are the basis for Particle filtering algorithm.

Bayesian estimation is the theoretical basis of the particle filter method. It is an estimation method based on the combination of objective information and subjective information. It takes into account both the objective information of the samples and the human subjective factors. Therefore, it performances well even when the sample is abnormal.

The state space model of a dynamic system can be described as:

 (1)

 (2)

Where *f*(.) and *h*(.) are the state transition equation and the observation equations, *xk* is the system state, *yk* is the observation value, *uk* is the process noise, and *vk* is the observation noise.

Bayesian estimate includes prediction and updating. The prediction process predicts the prior probability density of the state using the system model. The update process uses the latest measured value to correct the prior probability density to obtain the posterior probability density.

Monte Carlo method (Monte Carlo method), also known as statistical simulation method, is a kind of very important numerical calculation method guided by probability statistics theory.

When the solution is the probability of a random event, or the expected value of a random variable, the probability of this random event is estimated by the frequency of the occurrence of such an event by some experimental approaches.

Therefore, in the filtering process, when we need to use the probability *P*(*x*), we sample the variable *x*, and demonstrate *P*(*x*) with a large number of samples and their corresponding weights.

**3. Noise reduction model**

Trajectories of moving objects may be noisy due to external environmental factors and equipment error during their position sampling process. Different from outliers, noise causes tremendous impact on trajectory data analysis, such as compression, clustering, classification as well as pattern discovery. Therefore, in order to get rid of noisy data and eliminate the influence of noise, in this section, we proposed a noise reduction model, which can smooth the noise in the measurements.

To formally describe noise reduction model and the trajectory compression and recovery algorithm, trajectory related definitions are given as followings.

**Definition 1.** Trajectory Database (TD). TD (Trajectory Database) denotes trajectory set TD= {*TR1*, *TR2*, …, *TRn*}, and *TRi* is the i-th trajectory. A trajectory is a chronological sequence consisted of multi-dimensional locations, which is denoted by *TRi*= {*P1*, *P2*, …, *Pm*} (1≤i≤n).

In this paper, the trajectory can be captured as a time-stamped series of location points, denoted as {<*x1*, *y1*, *t1*>, <*x2*, *y2*, *t2*>, …, <*xn*, *yn*, *tn*>} where *xi*, *yi* represent longitude and latitude of the moving object at time *ti* and *n* is the total number of the recorded points in the trajectory.

To obtain higher precision data, we can use filtering techniques to smooth the noise of the trajectory. As we discussed in section 2, among many efficient filter algorithms, Particle filtering algorithm is an efficient and effective noise filter, contributes to its superiority of non-linear, non-Gaussian systems. Therefore, we select the Particle filter to filter out noise in trajectories.

From the Figure 2 we can see that the basic process of noise reduction model includes four stages: particle initialization, state transition, importance sampling and re-sampling.



**Figure 2.** The process of noise reduction model

**3.1 Particle initialization**

As described in section 2, the particle filter algorithm is derived from the Monte Carlo which uses the frequency of occurrence of an event to indicate the probability of the event. In the process of particle filtering, it can be obtained by processing the state of a large number of particles.

Based on the above analysis, the true, error-free trajectory is denoted as a sequence of coordinates *Xi*= (*xi*, *yi*)T, the initialization step is to generate *N* particles  in a rectangular region containing the entire trajectory coordinates, and these particles gather around the initial trajectory point with a Gaussian distribution.

**3.2 State transition**

From the Bayesian estimation theory in section 2 we can see that we need to estimate the posterior probability density in the current environment by the previous priori probability when utilize Particle filter to filter the trajectory and this process is also done by particles. Specifically, the state of each particle in the frame is estimated based on the state of the particles in the previous frame. The process of converting the state of the particles from the previous frame to the state of the current frame is called transmission which can be denoted as follows.

 (3)

 (4)

Formula (3) is the state transition equation, formula (4) is the observational equation, and *ui* and *vi* are Gaussian noise.

According to the state transition equation, each particle gets a prediction particle.

**3.3 Importance sampling**

The transfer phase shifts the position of the particles in the previous frame to get the new position in the current frame. But not all the particles are useful. Thus, at this stage, the particle filter algorithm will calculate the particle weight  according to the distance from the measured value. Each particle needs to calculate its weight, and it needs to be normalized. This stage is a posterior probability update process.

**3.4 Re-sampling**

The particle filter algorithm discards the low-weight particles, allowing the particles with higher weights to produce more particles, which causes the algorithm to converge toward a higher weight. So, in the resampling phase, we resample the particle set , and the resampled particle set is . The state estimate of *X* at time *i* is .

**4.** **Preliminary of improved compression and recovery model**

In this paper, compressed sensing is used to compress the original GPS trajectory. Compressed sensing is widely used in data processing. The traditional Shannon-Nyquist sampling method requires the sampling frequency more than double the highest frequency in the signal(The conventional Shannon’s sampling theorem requires that the sampling rate should not be less than twice the Nyquist sampling rate ), but CS uses much lower sampling rate and it can recover the original data with little error [25-26]. In brief, the compressed sensing indicates that if the signal is compressible or sparse in some transform domain, then the high-dimensional signal can be projected into a low-dimensional space with an observation matrix which is not related to the transform base, finally the original signal can be reconstructed from these small quantities of projections with high probability by solving an optimization problem. It can be shown that such projections contain enough information to reconstruct the signal.

The theory of Compressed Sensing mainly includes three parts, namely the sparse representation of the signal, the observation matrix of and the signal reconstruction algorithm. The content and model of Compressed sensing are shown as follows.

**4.1 The sparse representation of the signal**

The real signal exists in nature is not absolute sparse, but in a transform domain under the approximate sparse, is a compressible signal. Or in theory, any signal is compressible, as long as they can find the corresponding sparse representation of the space, you can effectively compress the sample. The sparsity or compressibility of the signal is an important prerequisite and theoretical basis of compressed sensing.

The significance of sparse representation: Only if the signal is *K*-sparse (and *K*<*M*<<*N*), it is possible to reconstruct the original length *N* signal from *K* larger coefficients when observing *M* observations. That is, when the signal has sparse expansion, you can lose a small coefficient without distortion.

Classical sparse methods include Discrete Cosine Transform (DCT), Fourier Transform (FFT), and Discrete Wavelet Transform (DWT) and so on.

**4.2** **The observation matrix of the signal**

The observation matrix (also known as the measurement matrix) *M*×*N* (*M*<<*N*) is used to observe the original signal of the *N* dimension to obtain the *M*-dimensional observation vector *Y*, and then use the optimization method to reconstruct the *X* from the observation Y with high probability. That is, the original signal *X* is projected onto the observation matrix (observation base) to obtain a new signal *Y*.

The observation matrix is designed to acquire *M* observation values and to reconstruct the signal *X* of length *N* or a sparse coefficient vector equivalent to the sparse basis *Ψ*.

**Definition 2.** Restricted Isometry Principle (RIP). A matrix *Φ* satisfies the RIP of order *k* if there exists a constant *δk* within (0, 1), such that holds for all k-sparse vectors *x*. And it can be denoted as formula (5) as follows.

 (5)

In order to ensure that the signal can be reconstructed from the observed value, in [27], Candes and Tao introduced the restricted isometry principle (RIP) of *Φ* and established its important role in CS, it needs to satisfy some restrictions: the product of the observed base matrix and the sparse matrix satisfies the RIP property.

**4.3 Signal reconstruction algorithm**

When the matrix *Φ* satisfies the RIP criterion. ,(.改为，) Compressed Sensing theory can solve the inverse problem of the above equation by solving the sparse coefficient *s* and then recover the signal *x* of the sparse degree *k* from the measured projection value *y* of the *M* dimension.

The most straightforward method of decoding is to solve the optimization problem by the L0 norm (0-norm, that is, the number of nonzero elements in the vector *y*) which can be described as formula (6).

 (6)

Thereby obtaining an estimate *s'* of the sparse coefficient *s.* The original signal *x'* = *Ψs'*. Since the solution of the above equation is a NP-hard problem (it is difficult to solve in polynomial time, it is not even possible to verify the reliability of the solution). L1 minimum norm and L0 minimum norm is equivalent under certain conditions, and they can get the same solution. Then the above equation is transformed into the optimization problem under L1 minimum norm which denoted in formula (7).

 (7)

At present, the compressed sensing reconstruction algorithm is divided into two categories which are described as follows. One is Greedy algorithms which achieve the approximation of signal vectors through the choice of appropriate atoms and a series of incremental method, such algorithms include matching pursuit algorithm [28], orthogonal matching pursuit algorithm [29], complementary space matching pursuit algorithm [30]. Convex optimization algorithms which solve problems by linear programming, these algorithms include gradient projection method [31], basis pursuit method [32] and the least angle regression method [33]. In conclusion, the convex optimization algorithm is more accurate than the greedy algorithm, but requires higher computational complexity.

**4.4 Compressing Sensing Model**

Let *x* be a one-dimensional signal of length *N*, with a sparsity of *k* (containing *k* nonzero values), *A* is a two-dimensional matrix of *M*×*N* (*M*<*N*), *y* = *Φx* is a one-dimensional measurement of length *M*. The problem of compressed sensing is solving underdetermined equations *y* = *Φx* to get the original signal *x*, on the basis of the known measurement y and measurement matrix *Φ*. Each row of *Φ* can be thought of as a sensor that multiplies the signal and picks up part of the signal. And this part of the information sufficient to represent the original signal, and can find an algorithm to high-probability restore the original signal.

The general natural signal x itself is not sparse and needs sparse representation on some sparse base, *x* = *Ψs*, *Ψ* is the sparse matrix, *s* is the sparse coefficient (*s* only *K* are nonzero values (*K* << *N*).

The compressed sensing equation is *y* = *Φx* = *ΦΨs* = *As*.

The original measurement matrix Φ is transformed into *A* = *ΦΨ* (called the sensing matrix), and the approximation value *s'* of s is solved. Then the original signal *x'* = *Ψs'*.

**5. Sparse high-noise GPS Trajectory Data Compression and Recovery based on Compressed Sensing**

As discussed above, this paper proposes a Sparse high-noise GPS Trajectory Data Compression and Recovery based on Compressed Sensing (SGTCR-CS). The GPS trajectory is a time-stamped series of location points and we suppose that it contains n points. The data can be denoted by *P* = {*p1*, *p2*, …, *pn*}. The process of the algorithm we proposed is shown in Figure 3 and can be divided into compression algorithm and recovery algorithm.



**Figure 3.** The process of SGTCR-CS

**5.1 The improved GPS Trajectory Data Compression Algorithm**

From Figure 3 we can see that the compression process mainly contains coordinate transformation, trajectory noise filtering and compression. Based on the above analysis, the improved GPS trajectory compression algorithm is provided in Algorithm 1.

**Algorithm 1: The improved GPS Trajectory Data Compression Algorithm**

**Input:** The GPS trajectory *p*

**Output:** The compression result , 

01: =UTM ();//transform the latitude and longitude coordinates into plane coordinates by the Mercator projection

02: Filtering the trajectory noise using noise reduction model:

= GetFilterTrajectory ();//get the filer result of latitude data

03: = GetLatitude();//get the trajectory latitude data

= GetLongitude();//get the trajectory longitude data

04: = GetObervationX();//compute the observation result of latitude

= GetObervationY();//compute the observation result of longitude

**End.**

The original GPS trajectory is a collection of latitude and longitude coordinates. To facilitate the calculation and plotting, we convert latitude and longitude coordinates into plane coordinates of meters by Universal Transverse Mercator Grid System(UTM) [34] in Line 01 of algorithm 1. Since the noise of trajectory is high, Line 02 is used to filter the original trajectory using noise reduction model. The detailed Particle filter is described in algorithm 2.Since the GPS trajectory data is 2-dimentional, the longitude and latitude are compressed and recovered separately. Line 03 is used to obtain the trajectory latitude and longitude data separately. From the above compression and recovery model, we know that the compressed sensing equation is *y* = *Φx*, where *x* is the *N*-dimensional signal and *y* is the *M*-dimensional observed vector. In algorithm 1, Line 04, *Sx* and *Sy* are compression results of the latitude data and longitude data.

**Algorithm 2: Algorithm of Particle filter**

**Input**: The original GPS trajectory with Gaussian noise;

**Output**: The filtered GPS trajectory;

01: **Particle initialization:** At *k*=0 moment, the particle swarm  is generated by the known prior probability , and the weight of all particles is .

02: **Sequential importance sampling**:

03: (1) Sample: At *k* moment, sampling the particles  in the importance distribution function, and the particle set is



04: (2) Weights estimation: After the observed value  is obtained, the estimated value of the importance weight can be denoted as



05: (3) Weights normalization:



06: **Re-sampling:**

07:(1) The particle weights are obtained from **Step 2**, the particles with smaller weight are removed, and the larger particles are copied to obtain the new particle set .

08: (2) The weight of all particles is .

09: **Let** , jump to **02**.

Based on the analysis in noise reduction model, the algorithm of Particle filter is described in Algorithm 2. The main steps of the algorithm are as follows. (1) Particle initialization. During this step, the particle swarm is generated by the known prior probability (Line 01). (2) Sequential importance sampling. We first sample the particles (Line 02), then estimate the weights of these particles (Line 04), and normalize the weights (Line 05) at last. (3) Re-sampling. In this step, we retain particles with larger weight and remove the particles with smaller weight (Lines 06~08), and then repeat the process from Line 02(Line 09).

**5.2. The GPS Trajectory Data Recovery Algorithm**

From the above GPS trajectory data compression algorithm and the compressed sensing model, we obtain the observation value *y*. The compressed sensing equation is *y* = *Φx* = *ΦΨs* = *As*. In this paper, we take the random Gaussian measurement matrix as *Φ*, and use the method of Discrete Cosine Transform (DCT) to construct the sparse matrix *Ψ*. The sensing matrix *A* = *ΦΨ*. Based on the above analysis, the recovery algorithm is described in Algorithm 3.

**Algorithm 3： Trajectory Recovery based on Compressed Sensing**

**Input**: Sensing matrix *A*, observation *y*, step size *sp*;

**Output**: A *K*-sparse approximation of the input signal;

01: **Initialization**:

02: {Trivial initialization}

03: *r0* = *y* {Initial residue}

04: *F0* = {Empty finalist}

05: *I* = *sp* {Size of the finalist in the first stage}

06: *k* = 1 {Iteration index}

07: *j*=1 {Stage index}

08: **repeat**

09: *Sk* = Max (| *A*\**rk-1*|, *I*) {Preliminary test}

10: *Ck* = *Fk-1* *Sk* {Make candidate list}

11: *F* = Max (||, *I*) {Final test}

12: *r* = *y* - {Compute residue}

13: **if** halting condition true **then**

14 quit the iteration;

15: **else if** ||*r*||2 ||*rk-1*||2 **then** {Stage switching}

16: *j* = *j* + 1 {Update the stage index}

17: *I* = *j* *sp* {Update the size of finalist}

18: **else**

19: *Fk* = *F* {Update the finalist}

20: *rk* = *r* {Update the residue}

21: *k* = *k* + 1

22: **end if**

23: **until** halting condition true;

24 **Output**: = {Prediction of non-zero coefficients}

In algorithm 3, the algorithm requires three input parameters, which are sensing matrix *A*, the measurement *y* and the step size *sp*. The algorithm uses an iterative method to obtain the *k*-sparse approximation of the input signal and can be partitioned into four steps. First, the parameters are initialized. The *r* is residual, *k* is the number of iterations and *F* is a collection of iteration (Lines 01 ~ 07). Then, we select the column collection of *A* by the index collection *Ck* (Lines 09~11). Third, we use the least squares method to obtain the approximation (line 24). Finally, the most important step is updating residuals (Lines 12~21). If the r meets the certain conditions, the iteration ends and we can reconstruct the signal with the sparse matrix.

**6. Experiments and analysis**

Moreover, we compare our algorithm with the DP algorithm. The major parameters for the simulation are shown in Table 1.

**Table 1.** Parameter descriptions

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Value Range |
| *τ* | The interval of trajectory observation | 5*s* |
| *N* | The length of the processed sequence | 1600 |
| *n* | The number of Monte Carlo simulations | 100 |
| *p* | The number of particles | 1000 |
| *q* | The *db* of Gaussian noise | 30-40 |
| *m* | The number of noise | 1-100 |

**6.1 Datasets**

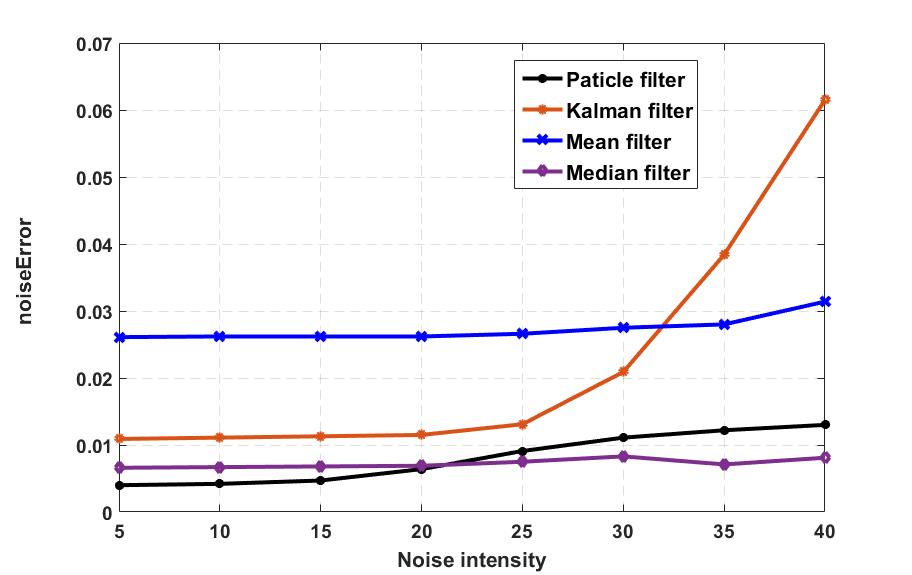
In this paper, the GPS trajectory we used was selected from the GeoLife project which collected by 182 users in a period of over five years. A GPS trajectory of this dataset is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. The trajectory we choose recorded 1000 points by GPS logger at the rate of one every five seconds. To simulate high noise that sometimes appear in GPS trajectories, we added some Gaussian noise manually on the original GPS trajectory. We randomly select m points from the original error-free trajectory and add the Gaussian noise of intensity *q* *db*.

**6.2 Noise filtering analysis**

In this section, several filtering algorithms are utilized to process the GPS trajectory data that has been added some Gaussian noise. In this experiment, the original trajectory contains 1000 points, and we add Gaussian noise to 100 of them. The neighborhood size of mean and median filter is 10 and Kalman filter uses the same model of Particle filter proposed in this paper. To evaluate the performance of the filtering process, we adopt the noiseError, which can be denoted as formula (8):

 (8)

where *x* and *y* denote as the original coordinate GPS data, *Fx* and *Fy* represent the corresponding results of the filtering algorithms.

****

**Figure 4.** noiseError performance comparison with four different filtering algorithms

In Figure 4, the comparison is carried out among the original trajectory with different noise intensity. From the Figure 4 we can see that the Particle filter used in this paper has a better performance than the mean filter and Kalman filter. It can be deduced that the mean filter is sensitive to noise and with the increase of noise intensity, the noiseError of Kalman filter increases in exponential form which shows that Kalman filter requires the accuracy of the system model. Comparing the results from Particle filter and median filter, it can be seen that the performance of median filter becomes better than Particle filter when the noise intensity exceeds about 20 *db*. This happens because the median filter simply replaces the point with the median of each point in a neighborhood of this point. So, it is less sensitive to noise and gives a smooth result. But it is just designed to estimate location and ignores some order variables like speed.

**6.3 Accuracy analysis**

In this section, the real GPS trajectory datasets are used to evaluate the SGTCR-CS algorithm proposed in this article. To evaluate the performance of the algorithm, we use the parameter error, which can be denoted as formula (9):

 (9)

where *x* and *y* represent the original coordinates while *x~* and *y~* represent the recovered coordinates.

In Figure 5(a), we compared our algorithm with CS and DP algorithms when the number of noise is 50 and the noise intensity is 30 *db*. From the result of this figure, we can see that the error curve of DP algorithm is relatively stable, while the curves of CS and SGTCR-CS algorithms drop very fast. With the increase of compression rate, the error of CS and SGTCR-CS are less than that of DP algorithm which means that the CS algorithm has a better performance in the recovery of high noise GPS trajectory. Comparing the result of CS and SGTCR-CS, it can be seen that the noise reduction model can enhance the error performance obviously.

From the Figure 5(b) and Figure 5(c), the error of the three algorithms increased with the increase of the noise intensity and the number of noise. From the comparison of the results, it can be seen that when the number of noise and noise intensity increased, the advantage of our SGTCR-CS algorithm became more obviously.

|  |  |
| --- | --- |
|  |  |
| (a) m=50, q=30 | (b) m=50, q=35 |
|  |  |
| (c) m=100, q=30 | (d) m=100, q=35 |

**Figure 5.** Error performance for the high noise GPS trajectory data recovery.

**6.4 Time consuming**

**Table 2.** Comparison of time consuming

|  |  |  |
| --- | --- | --- |
| Compression and Recovery Algorithm | Time cost (*s*) | Compression rate (%) |
| DP | 0.214 | 50 |
| 0.193 | 60 |
| 0.188 | 70 |
| SGTCR-CS | 5.666 | 50 |
| 10.344 | 60 |
| 20.286 | 70 |

From Table 2 we can see that the time consuming of the SGTCR-CS algorithm is much greater than the DP algorithm. The reason for this situation is described as follows.

Simply considering the compression algorithm, the three algorithms are basically the same time consumed. To completely restore the original GPS trajectory data, CS and SGTCR-CS algorithms need to use the greedy algorithm for iterative calculations and the complexity of the algorithms are high. However, the DP algorithm only uses the linear interpolation method to insert the position points directly at the relevant position of the line segment, and it is fast. But DP algorithm does not take into account the integrity of the recovered GPS trajectory data which resulting in that the recovered trajectory is quite different from the original trajectory.

**7. Conclusions**

In this paper, we analyze and design an adaptive trajectory compression and recovery method algorithm for noisy GPS trajectory which based on compressed sensing. Moreover, the algorithm utilizes the noise reduction model to filter out the high noise in the GPS trajectory to enhance the system performance. Extensive simulations show that the SGTCR-CS algorithm proposed in this paper get a good noise filtering and can not only achieve a fairly high compression rate but also acquire a high recovery accuracy.

**Acknowledgements:** This work was supported by the Fundamental Research Funds for the Central Universities, China (with grant of 2015XKMS085).

**References**

[1] Minjie Chen, Mantao Xu, Pasi Franti. Compression of GPS trajectories. In: Proceedings of the 2012 Data Compression Conference (DCC 2012), April 10-12, 2012, Snowbird, Utah, USA, 62-71.

[2] E. J. Candes, J. Romberg and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," IEEE. T. INFORM. THEORY. vol. 52, pp. 489–502, 2006.

[3] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections and universal encoding strategies," IEEE. T. INFORM. THEORY. vol. 52, pp. 5406–5425, 2006.

[4] D. L. Donoho, "Compressing Sensing." IEEE. T. INFORM. THEORY. vol. 52, pp. 1289–1306, 2006.

[5] F. Bin, X. Xinyan, and S. Guozhen, “An efficient mean filter algorithm,” in Complex Medical Engineering (CME), 2011 IEEE/ICME International Conference on, May 2011, pp. 466–470.

[6] T. Huang, G. Yang, and G. Tang, “A fast two-dimensional median filtering algorithm,” Acoustics, Speech and Signal Processing, IEEE Transactions on, vol. 27, no. 1, pp. 13–18, Feb 1979.

[7] R. E. Kalman, “A new approach to linear filtering and prediction problems,” J. Basic Eng, vol. 82, no. 1, pp. 35–45, Mar. 1960.

[8] Doucet, A., Freitas, N., Gordon, N.: An Introduction to Sequential Monte Carlo Methods. In:

Sequential Monte Carlo Methods in Practice., pp. 3–13. Springer: New York (2001)

[9] Douglas H David, Peucker K Thomas. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization, 1973, 10(2): 112-122.

[10] Xiaoying Liu. Trajectory data compression via spatial-temporal properties. Master Degree Thesis, Hong Kong University of Science and Technology, 2014, Hong Kong, China.

[11] Jianjun Liu, Kun Zhao, Philipp Sommer, et al. Bounded quadrant system: error-bounded trajectory compression on the go. In: Proceedings of the 31th IEEE International Conference on Data Engineering (ICDE 2015), March 31-April 4, 2015, Seoul, Korea, 987-998.

[12] Guangwen Liu, Masayuki Iwai, Kaoru Sezaki. An online method for trajectory simplification under uncertainty of gps. IPSJ Transactions on Databases, 2013, 6(3): 40-49.

[13] Wei Pan, Chunlong Yao, Xu Li, Lan Shen. An online compression algorithm for positioning data acquisition. Informatica, 2014, 38(4):339-346.

[14] Goce Trajcevski, Hu Cao, Peter Scheuermann, et al. On-line data reduction and the quality of history in moving objects databases. In: Proceedings of the 5th ACM international workshop on Data Engineering for Wireless and Mobile Access (MobiDE 2006). June 25, 2006, Chicago, Illinois, USA, 19-26.

[15] Falko Schmid, Kai-Florian Richter, Patrick Laube. Semantic trajectory compression. In: Proceedings of the 11the International Symposium on Spatial and Temporal Databases (SSTD 2009). July 8-10, 2009, Aalborg, Denmark, 411-416.

[16] Richter K F, Schmid F, Laube P. Semantic trajectory compression: Representing urban movement in a nutshell. Journal of Spatial Information Science, 2015 (4): 3-30.

[17] Shenzhu Feng, Jian Xu Ming, Xu Ning Zheng, et al. EHSTC: an enhanced method for semantic trajectory compression. In: Proceedings of the 4th ACM SIGSPATIAL International Workshop on GeoStreaming (IWGS 2013). November 5, 2013, Orlando, Florida, USA, 43-49.

[18] Jonathan Muckell, Jeong-Hyon Hwang, Vikram Patil, et al. SQUISH: an online approach for GPS trajectory compression. In: Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications (COM. Geo 2011). Article number: 13, 1-8.

[19] Jonathan Muckell, Paul W, Olsen Jr., Jeong-Hyon Hwang, et al. Compression of trajectory data: a comprehensive evaluation and new approach. GeoInformatica, 2014, 18(3): 435-460.

[20] Cheng Long, Raymond Chi-Wing Wong, H. V. Jagadish. Direction-preserving trajectory simplification. Proceedings of the VLDB Endowment, 2013, 6(10): 949-960.

[21] Birnbaum J, Meng H C, Hwang J H, et al. Similarity-based compression of GPS trajectory data. In: Proceedings of the 4nd International Conference on Computing for Geospatial Research & Applications (COM. Geo 2013), July 22-24, 2013, San Jose, CA, USA, 92-95.

[22] Nirvana Meratnia, Rolf A. de By. Spatiotemporal compression techniques for moving point objects. In: Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004), March 14-18, 2004, Crete, Greece, 765-782.

[23] Michalis Potamias, Kostas Patroumpas, Timos Sellis. Sampling trajectory streams with spatiotemporal criteria. In: Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM 2006). July 3-5, 2006, Vienna, Austria, 275-284.

[24] Jianjun Liu, Kun Zhao, Philipp Sommer, et al. Bounded quadrant system: error-bounded trajectory compression on the go. In: Proceedings of the 31th IEEE International Conference on Data Engineering (ICDE 2015), March 31-April 4, 2015, Seoul, Korea, 987-998.

[25] D. Donoho, “Compressed sensing,” IEEE Trans. Inf. Theory, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[26] J. Haupt, W. Bajwa, M. Rabbat, and R. Nowak, “Compressed sensing for networked data,” IEEE Signal Process. Mag., vol. 25, no. 2, pp. 92–101, Mar. 2008.

[27] E. Cand′es and T. Tao, “Decoding by linear programming,” IEEE Trans. Inf. Theory, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[28] S Mallat, Z Zhang. Matchingpursuit with time-frequency dictionaries[J]. IEEE Transactions on SignalProcessing, 1993, 41(12): 3397-3415.

[29] Joel A. Tropp and Anna C. Gilbert. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit[J]. IEEETransactions on Information Theory, VOL. 53, NO. 12, DECEMBER 2007.

[30] Chen S S, Donoho D L,Saunders M A.[Atomic decomposition by basis pursuit](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.4272&rep=rep1&type=pdf)[J]. SIAM review, 2001, 43(1): 129-159.

[31] Rosen J B. The Gradient Projection Method for Nonlinear Programming. Part II. Nonlinear Constraints[J]. Journal of the Society for Industrial & Applied Mathematics, 1961, 9(4):514-532.

[32] Malgouyres F, Zeng T. A Predual Proximal Point Algorithm Solving a Non Negative Basis Pursuit Denoising Model[J]. International Journal of Computer Vision, 2009, 83(3):294-311.

[33] Alfons A, Croux C, Gelper S. Robust groupwise least angle regression[J]. Computational Statistics & Data Analysis, 2016, 93(C):421-435.

[34] Zhang Y, Deng F, Chen Z, et al. UTM-CM: A Practical Control Mechanism Solution for UTM System[J]. 2010, 1:86-90.