**Sparse high-noise GPS Trajectory Data Compression and Recovery based on Compressed sensing**

**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 a series of problems in storage, transmission and analysis. Moreover, GPS trajectories are never perfectly accurate and sometime may have high noise. In this paper, an adaptive noise filtering, trajectory compression and recovery method based on Particle Filter(PF) and Compressed Sensing(CS) algorithm is proposed. Firstly, the Particle Filter is used to filter high noise in the GPS trajectory. Secondly, we use Compressed Sensing to compress the GPS Data. Thirdly, we can recover the GPS trajectory via Compressed Sensing. Experiments on real dataset show that: the CS-based method not only gets a good noise filtering but also achieves high compression and recovery performance compared to current algorithms.

**Keywords:** GPS trajectory, high noise, Particle Filter, 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 to an exponential growth in the amount of trajectory data as time going on. For instance, there are 5000 taxis in a city and we track the trajectory of each taxi by sampling its position once every 5 seconds, so we will overwhelm 2 GB of storage capacity to store a single day trajectory data. These data are the foundation for us to analyze activities and patterns for moving objects. However, the enormous volume of data has brought several problems. First, it is quite expensive and time-consuming to transmit these large amounts of data. Second, it is computationally expensive operations to query and extract useful patterns from these large amounts of trajectory data. Third, 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.

Currently, a number of trajectory compression algorithms have been studied to deal with the issues. In many researches, the main idea of line simplification is widely used to reduce the number of trajectory points by introducing a bounded error, which loses some information after compression [2, 3]. This kind of line simplification is mainly derived from the well-known Douglas-Peucker (DP) algorithm [4], which makes use of the divide-and-conquer approach to keep the most important points of a polyline. In order to take both spatial and temporal dimension into account, Meratnia et al. [3] 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. Besides DP algorithm, there are also various trajectory compression algorithm exists in the literature. Each offers a different trade off among compression time, compression ratio, and accuracy. Uniform sampling, which is fast and can archive the specified compression ratio by sampling trajectory at fixed time interval, but introduce large spatial and SED errors. To-Down Time Ratio (TD-TR) algorithm [3], is a variant of DP algorithm with SED instead of spatial error. It’s running time is O(n2). Opening Window (OW) algorithm [5] is an online approximate line simplification algorithm by introducing a slide window. OW algorithm runs with the window anchored at the first point, and gradually 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. Then it executes iteratively until the last point of trajectory is included. The running time of OW algorithm is O(n2). Opening Window Time Ratio (OW-TR) algorithm [3] is an extension to OW algorithm 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 [6] is an efficient compression algorithm that considers not only spatial dimension but also velocity information. DR algorithm firstly marks the start point p0 as the key point, and stores p0 and its velocity in the compressed representation. Then the next point pi is estimated whether it’s location within the SED threshold from p0. If true then continue the next point of pi, else pi 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 in [1]–[5] degrade significantly.

Before we start to use GPS data, it is unavoidable to solve some issues, for example, noise filtering, GPS trajectories are never perfectly accurate, due to the influences of the environment, equipment noise and other factors. Therefore the obtained GPS trajectory data may have a high noise. For this situation, various filtering technologies had been applied to smooth the noise and measurement errors.

Traditional Nyquist sampling requires that the sampling rate be at least twice the highest frequency component of the signal, but compressed sensing(CS) can recover the sparse signals from far fewer samples. Compressed sensing has been used to handle trajectory compression in many works, but most of them do not take consideration of the noise in trajectory data.

In order to solve the problem mentioned above, a compressed sensing(CS) based noisy GPS trajectory compression and recovery algorithm is proposed in this paper. Firstly, we use Particle Filter to smooth the noise, then compressed sensing algorithm compresses the original trajectory, finally, we recover the trajectory.

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

1)Firstly, a CS-based Sparse high-noise GPS trajectory compression and recovery algorithm is proposed and some theoretical aspects for the proposed compression and recovery problems are introduced.

2)Secondly, in order to improve accuracy, the GPS coordinate is converted into Mercator plane coordinate for calculation.

3)Then, the proposed algorithm is demonstrated based on GPS trajectory data, and shows high accuracy of the proposed algorithm. Extensive simulations show that CS-based trajectory compression and recovery can not only achieve a relatively high compression ratio but also recover almost the whole original trajectory.

4)Finally, simulations show that Particle Filter can enhance the performance of CS-based algorithm.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes our compression motivation and related definitions. In section 4, 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 5. Finally, Section 6 draws conclusions and points out some possible research opportunities.

**2. Related works and definitions**

**A. The classification of trajectory compression algorithms**

The rapid development of various subjects 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. The existing compression methods can be classified into 3 categories, according to their compression ideas.

1) Distance-based trajectory compression

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 [5], 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. In order to overcome this shortcoming, Meratnia et al.(ref.??) 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 [3,6]. 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 [24]. 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 [25].

2) Velocity-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[3]. 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 [3]. 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[26] 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 [18-22]. Schmid and Richter proposed a new and novel representation for trajectories that replaces trajectory data by the form of semantic information in road network [8]. Zheng proposed a new framework, namely paralleled road-network-based trajectory compression, to effectively compress trajectory data under road network constraints [7]. 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.

**B. The Compressed Sensing Algorithm**

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, but CS uses much lower sampling rate and it can recover the original data with little error. In brief, the compressed sensing indicates that as long as 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.

Compression sensing theory mainly consists of three parts:

(1) The sparse representation of the signal;

(2) Design a measurement matrix to reduce the dimension of the original signal x while ensuring the minimum loss of information;

(3) The signal restoration algorithm is designed to recover the original signal of length N without any distortion by M observation.

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 = Φ x is a one-dimensional measurement of length M. The problem of compressed sensing is solving underdetermined equations y = Φ x 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 = Θs.

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

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), discrete wavelet transform (DWT) and so on.

2. The observation matrix of the signal

The observation matrix (also known as the measurement matrix) MxN (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.

In order to ensure that the signal can be reconstructed from the observed value, it needs to satisfy some restrictions: the product of the observed base matrix and the sparse matrix satisfies the RIP property (Restricted Isometry Principle).

Definition. (Restricted Isometry Principle) 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.

(1-δk ) (1+δk )

3. Signal Reconstruction Algorithm

When the matrix Φ satisfies the RIP criterion. Compressive perception 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)

s.t y=ΦΨα

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 under certain conditions and L0 minimum norm is equivalent, can get the same solution. Then the above equation is transformed into the optimization problem under L1 minimum norm

s.t y=ΦΨα

At present, the compressed sensing reconstruction algorithm is divided into two categories:

(1) greedy algorithm, it is through the choice of appropriate atoms and a series of incremental method to achieve the approximation of signal vectors, such algorithms include matching pursuit algorithm, orthogonal matching pursuit algorithm, complementary space matching pursuit algorithm.

(2) convex optimization algorithm, which solves by linear programming through the solution, such algorithms include gradient projection method, basis pursuit method, the minimum angle regression method.

**C. Particle Filter**

The particle filter algorithm is derived from Monte Carlo, which uses the frequency of an event to refer to the probability of the event. Popularly speaking, particle filtering is also able to use some known data to predict future data, but the particle filter is not limited to Gaussian noise, in principle, particle filter can adapt to all non-linear, non-Gaussian system. The particle filter gets its name from the fact that it maintains a set of “particle” that each represent a state estimate. There is a new set of particles generated each time a new measurement becomes available. There are normally hundreds or thousands of particles in the set. Taken together, they represent the probability distribution of possible states. A good introduction to particle filtering is the chapter by Doucet et al. [8], and this section uses their notation.

**3. Trajectory Compression and Recovery Algorithm based on Compressed sensing**

This paper proposes an improved Noise Filtering, Trajectory Compression and Recovery method (NFTCR), which utilizes the Particle filter to filter out the high noise in the GPS trajectory, and combines the compressed sensing algorithm to compress the trajectory, and uses the compressed data to recover the original trajectory.

**A. Definition**

*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 record points in the trajectory.

**B. Sample Data**

In this paper, the GPS trajectory we used was 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 1638 points by GPS logger at the rate of one every five seconds, shown in Figure\_\_.

In order to improve the accuracy of the calculation and easy to plot, we transform the GPS coordinates into plane coordinate and the Mercator plane coordinate system is selected, so we converted the latitude/ longitude points to (x, y) in meters. And we can see from the picture, the recorded trajectory shows many spikes due to measurement noise or the equipment noise. Sometimes, the noise of the trajectory is very high, so we manually added some outliers to make the effects of the filtering techniques more obvious. These outliers are marked in Figure\_\_. We will use this data to carry out our experiments.

**C. Particle Filter Model**

Due to sensor noise, measurement error and other factors, the trajectories are not exact. To obtain higher precision data, we can use filtering techniques to smooth the noise of the trajectory. In this paper, we select the Particle filter to filter out noise in trajectories.

As mentioned above, the actual, unknown trajectory is denoted as a sequence of coordinates *Xi=(xi, yi)*T, and the coordinates of the known, measured trajectory is denoted as vectors Zi as

*Zi = Xi + Vi* (1.1)

The *Vi* is the noise vector that drawn from a two-dimensional Gaussian probability density with zero mean and diagonal covariance matrix R, i.e.

Vi ~ N(0, R) R=]

With the diagonal covariance matrix, this is the same as adding random noise from

two different, one-dimensional Gaussian densities to *xi* and *yi* separately, each with

zero mean and standard deviation σ. It is important to note that Equation (1.1) is just

a model for noise from a location sensor. It is not an algorithm, but an approximation

of how the measured sensor values differ from the true ones. For GPS, the Gaussian

noise model above is a reasonable one [7]. In our experiments, we have observed a

standard deviation σ of about ten meters.

The Particle filter uses a measurement model and a dynamic model and makes estimates ˆxi of a sequence of unknown state vectors xi, based on measurements zi.

The particle filter’s measurement model is a probability distribution p(zi|xi) giving

the probability of seeing a measurement given a state vector. This distribution must be provided to the particle filter. This is essentially a model of a noisy sensor which might produce many different possible measurements for a given state.

writing it as

p(Zi | X i)=N((xi, yi)T, Ri)

This says that the measurement is a Gaussian distributed around the actual location

with a covariance matrix Ri from Equation (1.2).

In addition to the measurement model, the other part of the particle filter formulation is the dynamic model, again paralleling the Kalman filter. The dynamic

model is also a probability distribution, which simply gives the distribution over the current state xi given the previous state Xi-1 : p(Xi | Xi-1). The particle filter version is much more general. For instance, it could model the fact that vehicles often slow down when climbing hills and speed up going down hills. It can also take into account road networks or paths through a building to constrain where a trajectory can go. This feature of the particle filter has proved useful in many tracking applications.

It is not necessary to write out the dynamic model. Instead, it is sufficient to sample from it. That is, given a value of Xi-1, we must be able to create samples of Xi that adhere to p(Xi | Xi-1). For our example, we will use the dynamic model which says that the location changes deterministically as a function of the previous location randomly perturbed with Gaussian noise:

*Xi+1=Xi +*

The above is a recipe for generating random samples of xi+1 from xi. The particle filter requires actually generating random numbers, which in this case serve to change the velocity.

Finally, we need an initial distribution of the state vector. For our example, we can say the initial velocity is zero and the initial location is a Gaussian around the first measurement with a covariance matrix Ri from Equation (1.2).

**Measurement Model**

The Particle filter formulation makes a distinction between what is measured and what is estimated, as well as formulating a linear relationship between the two.

As above, we assume that the measurements of the trajectory are taken as noisy

values of x and y:

*Zi=*

Here zi(x) and zi(y) are noisy measurements of the x and y coordinates.

**Dynamic Model**

If the first half of the Particle filter model is measurement, the second half is dynamics. The dynamic model approximates how the state vector xi changes with

time. Like the measurement model, it uses a matrix and added noise:

The particle filter maintains a set of P state vectors, called particles: , j=1…P.

There are several versions of the particle filter, but we will present the Bootstrap Filter from Gordon [12]. The initialization step is to generate P particles from the initial distribution. For our example, these particles would have zero velocity and be clustered around the initial location measurement with a Gaussian distribution as explained above. This is the first instance of how the particle filter requires actually

generating random hypotheses about the state vector. This is different from the

Kalman filter which generates state estimates and uncertainties directly. We will call

these particles .

With a set of particles and i > 0, the first step is “importance sampling”, which

uses the dynamic model p(xi|xi−1) to probabilistically simulate how the particles

change over one time step. For our example, this means invoking Equation (1.19) to create . The tilde (∼) indicates extrapolated values. Note that this involves actually generating random numbers for the velocity update.

The next step computes ”importance weights” for all the particles using the measurement model. The importance weights are

Larger importance weights correspond to particles that are better supported by the measurement. The important weights are then normalized so they sum to one.

The last step in the loop is the “selection step” when a new set of P particles x( j)

is selected at random from the ˜x( j) based on the normalized importance weights.

The probability of selecting an ˜x( j) for the new set of particles is proportional to its

importance weight w˜( j) It is not unusual to select the same ˜x( j) more than once if it has a larger importance weight. This is the last step in the loop, and processing then

returns to the importance sampling step with the new set of particles. While the particles give a distribution of state estimates, one can compute a single estimate with a weighted sum:

Applying the particle filter to our example problem gives the result in Figure 1.12.

We used P = 1000 particles.

**D. Compressed Sensing Model**

In this paper, a total n points are taken into consideration. The data set is symbolized as P = {p1, p2, . . . , pn}. After compressive sensing processing, m points are obtained as a compression version for the overall data set. The compression version can be represented as C = {c1, c2, . . . , cm}.

Note that a position data point is typically represented using two geographic Cartesian coordinates, the x-coordinates (longitude data) and the y-coordinates (latitude data). Since the focused vehicular trajectory data is 2-dimensional for each

sample, strategy proposed in this subsection is based on parallel transmission. That is, the latitude information and longitude information are compressed and recovered separately.

The compression process (without both original sensor measurement errors and compressive sensing errors) can be represented as:

y = Φx = ΦΨα = Aα,

where A = ΦΨ, and there are not any errors in the front measurement process. That is, x = Ψα, where Ψ represents the dictionary matrix, and α is corresponding sparse signal representation of x. Note that x and α are equivalent representation of the trajectory segment, with x in the time domain and α in the Φ domain.

The optimal solution for above under-determined problem can be obtained by solving the following optimization problem:

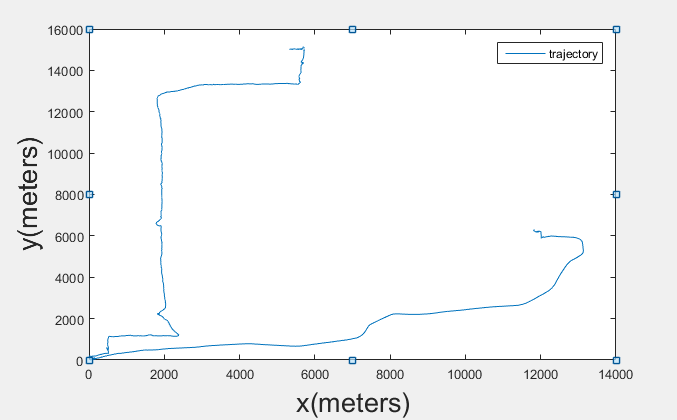
s.t y=ΦΨα

**4. Experiments and analysis**

In this paper, we design and implement a series of experiments to verify the NFTCR.

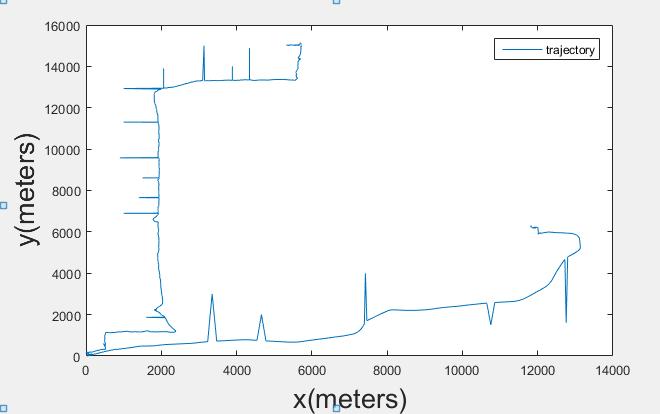
A. Experimental data sets

The experiments use the real human GPS dataset that was collected in Geolife project. These data contain rich attributes including latitude, longitude, time, In this paper, we extract the trajectory which includes 1638 points. For plotting, we converted the latitude/longitude points to (x, y) in meters, shown in Figure 1.



**Fig. 1.** This is the original trajectory recorded by a GPS logger.

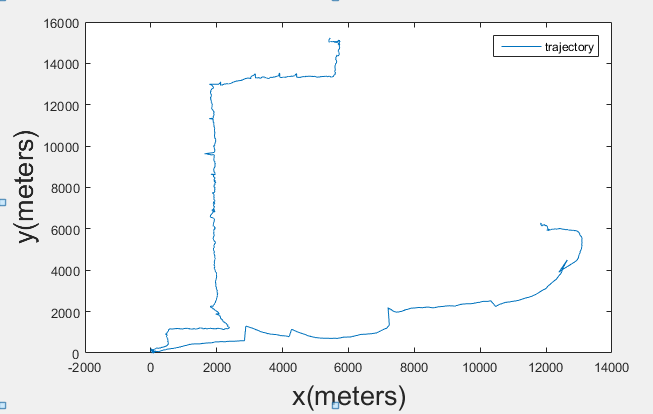
We manually added some outliers to simulate large deviations that sometimes appear in recorded trajectories. These outliers are marked in Figure.2.



**Fig.2.** The outliers were inserted in the original trajectory.

B.Particle Filter result

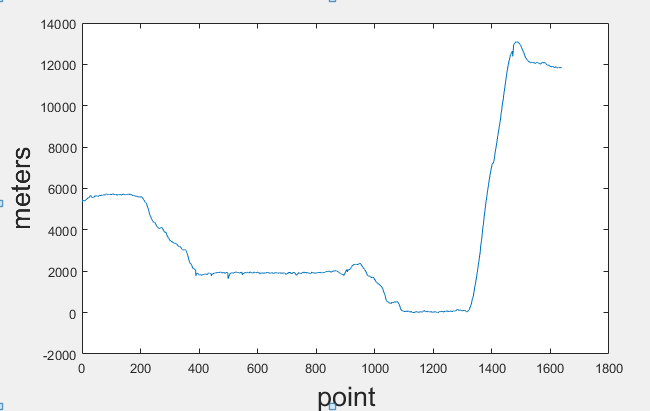
We use Particle Filter to smooth the trajectory, show in fig.3.



**Fig.3.** This is the result of the particle filter.

C. Compressed Sensing Parameter Selection and Results Comparison

As discussed above, Since the focused vehicular trajectory data is 2-dimensional for each sample, strategy is based on parallel transmission. That is, the latitude information and longitude information are compressed and recovered separately, so we select the longitude information to do the experiment, and the latitude information can be handled in the same way. The original longitude information that contains 1638 points is shown in figure 4.



**Fig.4.** This is the longitude information.

The cost of NFTCR mainly depends on the sample vector y, y is the compressed result and the length of y is M, the larger the M is, the more time cost and the less compression ratio is, but the accuracy is higher. The experiment result can be seen from table 1.

**TABLE I. Results of experiments**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Points of compressed trajectory | 100 | 150 | 200 | 250 | 300 |
| Time cost(s) | 0.137105 | 0.230269 | 0.408214 | 0.706762 | 1.141092 |
| Compression ratio (%) | 93.89 | 90.84 | 87.79 | 84.74 | 81.68 |
| Recovery residuals (10-3) | 9.7542 | 6.4506 | 3.3413 | 3.1996 | 2.4155 |

Figure 5, Fig.6, Fig.7, Fig.8, Fig 9 show the compressing trajectory data result where M is in 100, 150, 200, 250, 300. Based on experience, we use DCT to make the trajectory sparse in some domain. The blue curves represent the original trajectory and the red curves represent the recovered trajectory. As can be seen from the pictures, the blue and red curves are mostly coincident. From table 1 we can find that, with the increase of M, the recovery residual of the trajectory is fewer which means the accuracy of the recovery algorithm is higher.

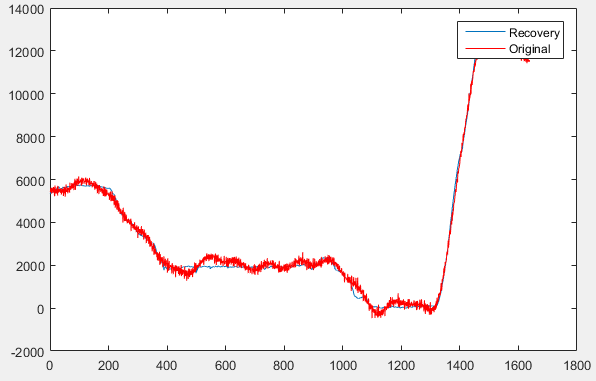
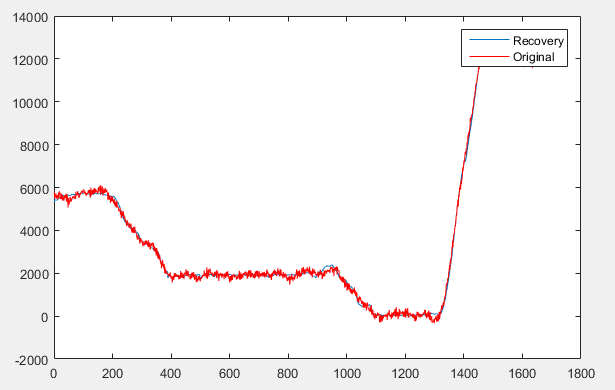
 

Fig.5. M=100 Fig.6. M=150

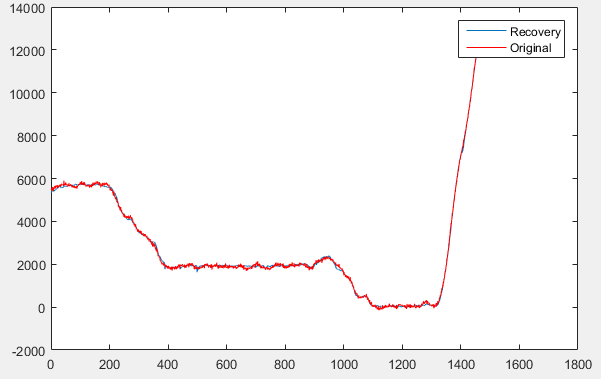
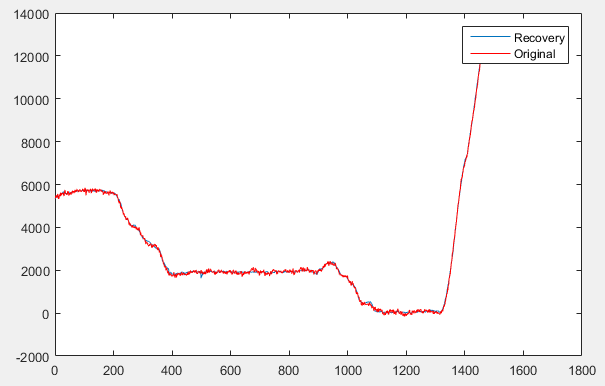
 

Fig.7. M=200 Fig.8. M=250

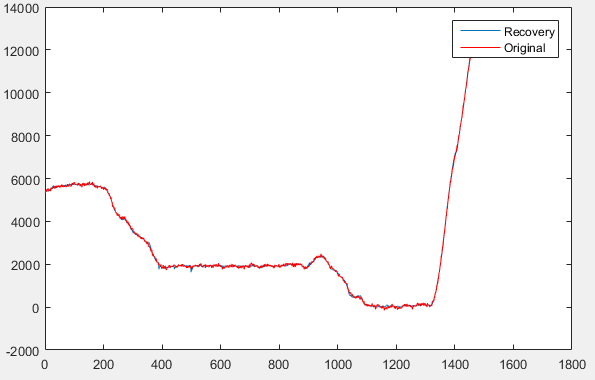


Fig.9. M=300

**5.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 Particle filter to filter out the high noise in the GPS trajectory to enhance the system performance. Extensive simulations show that the CS-based method 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.

In future work, we will devoted to the optimization of the algorithm, so we can achieve higher compression rate and cost less time.