An Adaptive Algorithm for Compressive Approximation of Trajectory (AACAT) for Delay Tolerant Networks

Rajib Rana[12] Wen Hu [2] Tim Wark[2] Chun Tung Chou[1]

[1] University of New South Wales, Australia
{rajibr,ctchou}@cse.unsw.edu.au
[2] CSIRO ICT Centre, Australia
firstname.lastname@csiro.au

Technical Report
UNSW-CSE-TR-1023
December 2010

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Highly efficient compression provides a promising approach to address the transmission and computation challenges imposed by moving object tracking applications on resource constrained Wireless Sensor Networks (WSNs). In this paper, we propose and design a Compressive Sensing (CS) based trajectory approximation algorithm, *Adaptive Algorithm for Compressive Approximation of Trajectory (AACAT)*, which performs trajectory compression, so as to maximize the information about the trajectory subject to limited bandwidth. Our extensive evaluation using "real" trajectories of three different object groups (animals, pedestrians and vehicles) shows that CS-based trajectory compression reduces up to *30% transmission overheads*, for given information loss bounds, compared to the state-of-the-art trajectory compression algorithms. We implement AACAT on the resource-impoverished sensor nodes, which shows that AACAT achieves high compression performance with very limited resource (computation power and energy) overheads.

1 Introduction

Object tracking is on horizon due to multitude of application scenarios in the present time. The Virtual Fencing (VF) application devised by the CSIRO ICT Center, Australia is one such example, where the locations of animals are controlled, not by a physical fence, but with stimuli (e.g. auditory and mild electric shocks) applied by special devices worn by the animals. Ethical considerations are critical in the VF application and observations regarding the states of each animal must be maintained. Since continuous human observation is infeasible, it is important to return data about the states of the animals and the stimuli applied. The VF application operates in a delay tolerant fashion. The position data of an animal is recorded and stored in a sensor node attached to the animal. There is also a small number of fixed nodes which are connected to the basestations. When animals enter the transmission range of a fixed node, position data from their sensor nodes are uploaded to the basestations. Due to uncoordinated movement of the animal connection time (amount of time a fixed node is connected with the mobile node) is typically unpredictable. Therefore, the amount of data that can be uploaded during a connection is also unpredictable in the VF application.

The VF application poses two fundamental challenges. First, monitoring a large number of animals requires the availability of their complete geographical traces (trajectories), which leads to *transmission challenges* due to limited bandwidth of wireless nodes. Second, limited transmission opportunities and connection time often cause the memory buffer to become full, therefore it becomes necessary to discard data, which leads to data loss.

Besides the VF application, trajectory compression is becoming essential in a number of participatory sensing [2] applications, such as, social networking (user locations need to be continuously uploaded) and traffic conditions monitoring (traffic condition is inferred by analyzing position data uploaded from vehicles) using mobile phones. Although, mobile phones have access to higher bandwidth, expensive cellular communications often restricts the amount of data that can be transmitted.

An efficient trajectory compression algorithm helps to cope with limited bandwidth; therefore, trajectory compression has been extensively studied in the last decades. One class of these compression algorithms [6, 12] is mainly guided by the advances in the field of line simplification and cartographic generalization. The primary disadvantage of these algorithms is the frequent elimination or misrepresentation of important points, such as sharp angles. Another class [1, 10] of algorithms achieves compression via predictions; however, these algorithms generally assume that the mobile objects have a limited number of movement states (e.g moving speed and direction), which may be impractical in the reality.

Recent developments in Compressive Sensing (CS) theory [3] provide an attractive alternative for trajectory compression in WSNs. Given that a trajectory segment $f \in \mathbb{R}^n$ is compressible in a sparsifying domain (e.g. Fourier, Discrete Cosine Transform (DCT) etc.), then a small number of coefficients (which will be denoted by k is this paper), is sufficient to *accurately* represent the trajectory segment. The key idea behind CS is that it takes a small number of projections $m(<< n)$ to *accurately* recover f with high probabilities (projections are typically aggregations of data points of a trajectory segment, we define it in detail

in Section 3). Since, only $m(<< n)$ projections need to be transmitted to the basestations, CS offers efficient use of bandwidth.

The key challenge of applying CS in trajectory compression is that we need to estimate the compressibility, namely the parameter k, of the trajectory in-situ to obtain the most benefits out of CS in practice, because the number of compressive projections (m) is a function of compressibility (k) that changes dynamically with the speed of the mobile objects. Intuitively, when an object is stationary or moves with a constant speed, the compressibility of its trajectory is high and thus it requires a smaller number of projections to recover the trajectory accurately. On the other hand, when the speed of the object changes frequently, the compressibility of the trajectory diminishes and it requires a larger number of projections to recover the trajectory accurately. However, the conventional method of computing compressibility is computationally expensive, and is infeasible for resource constrained WSN nodes. We propose and design a Support Vector Regression (SVR), namely *ϵ-SV regression* [17], based in-situ compressibility estimation technique, to provide an accurate estimation of k based on the speed information of the nodes, and has small computational overheads.

Our key contributions can be summarized as follows:

1. We present an *Adaptive Algorithm for Compressive Approximation of Trajectory (AACAT)*, which adapts the number of CS projections while accurately approximating the trajectory. An ϵ-SV regression based estimation is proposed to adapt the number of CS projections to the object's speed in-situ, which improves the compression performance of CS based on the local speed observations.

2. Our evaluations, based on the trajectories of three types of moving objects, exhibiting large range of speed variations, show that CS-based trajectory compression achieves *30%* better "transmission versus accuracy trade-off", compared to two other state-of-the-art trajectory compression algorithms based on *Kalman filter* and *Spatiotemporal* compression techniques. Our evaluations also show that CS-based trajectory compression is resilient to information loss as it offers promising "loss-distortion trade-off".

3. Our *implementation* of AACAT on Fleck 3b platform consisting of an 8-bit microcontroller and 8 kB RAM, demonstrates the resource efficiency of AACAT that makes it a viable algorithm for resource impoverished sensor nodes. Furthermore, we show that AACAT approximates trajectories with high accuracy using empirical study.

The rest of the paper is organized as follows. We start with the description of our datasets in Section 2 followed by the problem formulation and brief introduction to the CS theory in Section 3. To improve the performance of CS in trajectory compression, we introduce an ϵ-SV regression-based technique to estimate k in Section 4, which is followed by the description of proposed AACAT implementation in Section 5. We evaluate AACAT extensively in Section 6 and conclude in Section 8.

2 Datasets

We evaluate the performance of AACAT using three datasets from WSN deployments and field experiments. These three datasets are made up of animal (cow), pedestrian and vehicle trajectories respectively, which will demonstrate the performance of AACAT with a large range of speed variation of mobile nodes. The animal trajectories were collected from a real WSN deployment at the Wivenhoe Dam, where sensor nodes are mounted on the cow collars and locations are sampled by the GPS receivers at 2 Hz. We collected trajectories of *36 cows* for one week. The pedestrian and vehicle datasets were created with the help of *20 staff* and *student members* of the CSIRO ICT center. They were given Nokia N97 mobile phones for a two-week period to use during their daily commute. Annotation was used to differentiate vehicle and pedestrian data. A python script was installed on the phone that recorded the GPS coordinates at 2 Hz. In summary, we collected 75,000 segments of cow trajectories, 1,000 segments of pedestrian trajectories and 5,000 segments of vehicle trajectories, where each segment has $n = 512$ data points.[1]

The ϵ-SV regression method used for estimating k involves a training and a test phase. We divided our datasets into training and testing datasets to use during the training and validation phases, respectively, and ensured that they are independent. For example, the training set for pedestrian was formed using the trajectories within the campus of the CSIRO ICT center and the test set was formed using trajectories outside of the CSIRO ICT center campus. For animals, we used the trajectories of half of the cows as the training set and trajectories from the rest of the cows as the test dataset. Finally, training and test datasets for vehicle were formed using trajectories from separate road segments.

3 Compressive Sensing for Trajectory Compression

We first present the problem that we solve in this paper followed by the basic idea of CS and how to apply CS theory to trajectory compression.

3.1 Problem Formulation

Consider $f \in R^n$ is a trajectory segment containing n consecutive position data points of a moving object. In order to conserve bandwidth, we want to acquire only $m << n$ projections of f, which can be used to accurately reconstruct f at the basestations. Due to limited transmission opportunities and connection time, some of these m projections may further be discarded. Therefore, it is also required that reconstruction performance degrades gracefully with the loss of projections.

[1] We also conducted experiments with smaller segment lengths e.g. 128, 256 and observed similar results.

3.2 Compressive Sensing

The theory of CS provides an attractive solution to the problem of recovering a compressible signal from a few projections. Given the trajectory segment f is an n data point discrete time signal, using an $n \times n$ orthonormal basis matrix $\Psi = [\psi_1|\psi_2|...|\psi_n]$ with the vectors ψ_i as basis vectors, f can be represented as,

$$f = \Sigma_{i=1}^{n} \psi_i \alpha_i \text{ or } f = \Psi\alpha, \tag{3.1}$$

where, α is an $n \times 1$ column vector of weighting coefficients $\alpha_i = \langle f, \psi_i \rangle = \psi_i^T f$ and $.^T$ denotes the transposition. Note that a position data point is typically represented using two geographic Cartesian coordinates, Northing and Easting, where Northing refers to the northward-measured distance (or the y-coordinates) and Easting refers to eastward-measured distance (or the x-coordinates). For simplicity, we consider Northing and Easting separately i.e. f contains either n consecutive Northing or Easting data points. Our approach is to separately recover the Northing and Easting data points of a given trajectory segment to recover the corresponding segment. Note that f and α are equivalent representation of the trajectory segment, with f in the time domain and α in the Ψ domain. Generally, f is compressible when it has a few large and many small coefficients in the Ψ domain. Formally, f is compressible when the reordered entries of its Ψ-coefficients decay like power law; i.e. when we rearrange the sequence of α in decreasing order of magnitude $|\alpha|_{(1)} \geq |\alpha|_{(2)} \geq ... \geq |\alpha|_{(n)}$, for some the $r \geq 1$, the zth largest entry obeys,

$$|\alpha|_{(z)} \leq \text{Const.} z^{-r}. \tag{3.2}$$

Instead of conducting *point-wise* measurements, CS takes $m << n$ *projections* of f, where each projection is an inner product between f and a projection vector ϕ_j as in $y_j = \langle f, \phi_j \rangle$. Forming a $m \times n$ projection matrix Φ using m vectors ϕ_j^T, we can write the projection operation in matrix form, as follows:

$$y = \Phi f = \Phi\Psi\alpha = \Theta\alpha \tag{3.3}$$

Eq. (3.3) represents the typical CS *encoding process*, where $\Theta = \Phi\Psi$.

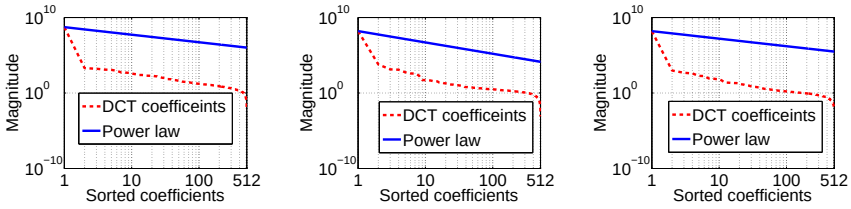
Since $m << n$, the problem of recovering f from y and Θ is ill-conditioned. CS shows that a sufficient condition for a stable solution is that for an arbitrary $3k$ sparse vector v and for some $\delta > 0$, Θ satisfies

$$1 - \delta \leq \frac{||\Theta v||}{||v||} \leq 1 + \delta. \tag{3.4}$$

Condition (3.4) is called Uniform Uncertainty Principle (UUP) [4] or Restricted Isometry Property (RIP). CS theory also suggests mechanisms to generate Θ matrices that satisfy the RIP with high probabilities. For example, if Φ is formed by sampling iid entries from the normal distribution with mean 0 and variance $1/m$, then for an arbitrary orthonormal basis Ψ, $\Theta = \Phi\Psi$ obeys RIP with overwhelming probability, when $m \geq ck \log(\frac{n}{k})$ with c as a small constant [5].

Condition (3.4) ensures that f can be recovered from (3.3); however, since $m << n$, there are many α' that satisfies $y = \Theta\alpha'$. Encouragingly, CS optimization based on ℓ_1 norm

$$\hat{\alpha} = \arg\min_{\alpha'} ||\alpha'||, \text{ s.t. } y = \Theta\alpha' \tag{3.5}$$



(a) Animal trajectory (b) Vehicle trajectory (c) Pedestrian trajectory

Figure 3.1: Validating compressibility of trajectories.

can accurately estimate f (namely producing the k-term approximation by returning its largest k coefficients) with high probabilities, using only $m \geq ck \log(\frac{n}{k})$ projections [4]. Eq. (3.5) is the CS *decoding (reconstruction)* process. Since (3.5) is a convex optimization problem, it can be solved in polynomial time and k-approximation of f (which will be denoted by $\hat{f}$) can be recovered using $\hat{f} = \Psi\hat{\alpha}$.

Given that f is compressible, i.e. k is very small, we have $ck \log(\frac{n}{k}) << n$. Therefore, using the CS theory, we can accurately recover f using only a small number of projections m and achieve high compression ratio $\frac{m}{n}$, since $m << n$.

Consider, we compute $m(= ck \log n)$ projections of f. Due to limited transmission opportunities and connection time, some of these projections (e.g. μ) are discarded and only $\hat{m} = m - \mu$ of them are transmitted to the basestations. Using $\hat{m}$ projections we cannot recover f accurately, rather we can produce $\hat{k} \leq k$ approximation of f with an increase of the reconstruction error. However, in Section 6.1 we will empirically show that the reconstruction error increases gracefully with the increase of μ.

3.3 Trajectory Compressibility

In order to get the most benefit out of CS in practice, we need to find a basis where the trajectory data points are most compressible. We therefore studied the compressibility of trajectories using our large dataset in a number of sparsifying bases including DCT, wavelets (Haar, Daubechies, Symlets) and Fourier. For each type of object, by plotting the representation of a large number of trajectory segments in these sparsifying domains, we observed that out of all the bases, the coefficients in the DCT have the quickest decay. Fig. 3.1 is a representative plot from our study, which shows the decay of the DCT coefficients. Note that, we only show the results for Northing; DCT coefficients of Easting also show similar results. We find that the trajectories are compressible since the DCT coefficients decay like power law as required by (3.2).

From Fig. 3.1 we also observe that the DCT coefficients of the animal trajectories decay slowly compared to other two object trajectories, which indicates that the animal trajectory is less compressible compared to others. Typically, speed of the animals change more frequently compared to pedestrians and vehicles, which causes the corresponding impact.

3.4 Adaptive Compressive Sensing

In CS, the number of projections, m, to approximate a trajectory segment accurately is determined by the number of significant coefficients, k, of the trajectory segment. Typically, k changes dynamically with the speed of the object. Therefore, computing k for every trajectory segment would further improve the compression performance. However, computing k for a trajectory segment involves a transpose of an $n \times n$ matrix ($O(\frac{n^2}{2})$ operations) and then multiplication of the $n \times n$ matrix with an $n \times 1$ vector ($O(n^2)$ operations), which are computationally expensive for resource impoverished wireless sensor nodes. Note that the transpose operation can be saved by storing Ψ^T in the memory, however the multiplication ($\Psi^T f$) will still incur high computational expenses. An efficient estimation of k is therefore required, which accurately estimates k incurring affordable computation cost. In the next section we will propose an ϵ-SV regression based technique to estimate k, which accurately estimates k incurring reasonable computational expenses.

4 In-situ estimation of k

In this section we first show the correlation of the speed of the moving objects with k, then we use ϵ-SV regression to model this correlation.

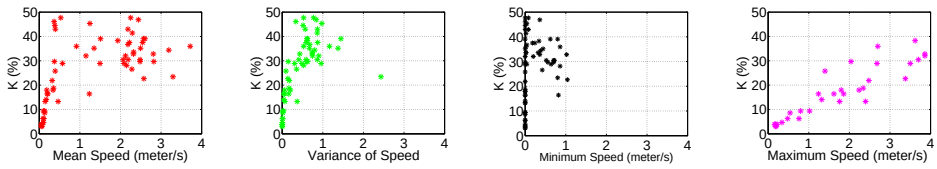
4.1 Correlation between Speed and k

Given $f \in \mathbb{R}^n$ is an n data point Northing or Easting segment of a moving object recorded over time $t_i, (i = 1, ..., n)$, the quantity $\frac{|f_{i+1} - f_i|}{t_{i+1} - t_i}, (i = 1, ..., n-1)$ produces the speed along the corresponding Northing or Easting axis. We aim to estimate k based on the speed of the moving object.

We process the trajectory in the test data set, segment by segment where each segment is of length n. The n data points of a trajectory segment gives $n - 1$ speed readings along both Northing and Easting axes. In order to investigate the correlation of speed with k, for each segment we compute mean, variance, minimum and maximum (we call them *speed candidates* hereafter) of the corresponding $n-1$ speed readings and plot them against the corresponding k values, where we use a value of k that approximates the corresponding trajectory segment (Northing and Easting separately) within a small (one meter) error.

Fig. 4.1 summarizes the correlations of different speed candidates with k over all the segments of the training set from animals.[1] For a given speed candidate e.g mean speed, we compute the mean speed of each of the trajectory segments and compute the corresponding k value to approximate the trajectory segment within one meter error. We then plot these mean speeds along x-axis and corresponding k values in y axis in Fig. 4.1(a). In particular, instead of plotting the k values, we plot the ratio between k and the total number of coefficients within a trajectory segment along the y-axis. We observe that *mean*, *variance*

[1] Due to space constraints, in Fig. 4.1 we only depict the correlation for Northing. Similar results are observed for Easting. Furthermore, we only use the animal dataset for this illustration since similar to animal, for the rest two datasets, mean, variance and maximum speed show the best correlation with k.



(a) CC=0.355 (b) CC=0.322 (c) CC=0.008 (d) CC=0.459

Figure 4.1: The correlations of various speed candidates with k. The values of the correlation coefficients (CC) are shown in the caption of corresponding figure.

and *maximum* speed are well correlated with k, because with the increase of these speed candidates, the corresponding values of k increase proportionally. For clarity we also show the value of correlation coefficients (CC) in the caption of the corresponding figure. In the next section we will model the correlations between the three chosen speed candidates (mean, variance and maximum) and k using ϵ-SV regression.

4.2 Modeling Correlation by ϵ-SV Regression

Consider for a given speed candidate s, the training set,$\{(s_1, k_1), ..., (s_L, k_L)\}$, $s \in \mathbb{R}, k \in \mathbb{R}$, has L elements, where s_i is ith value of s and k_i is the corresponding value of k. The ϵ-SV regression determines a function $g(s)$ that has at most ϵ deviation from the actual k_i for all the training data, and at the same time is as flat as possible. The function g is typically computed from

$$g(s) = \langle \omega, s \rangle + b \text{ with } \omega \in \mathbb{R}^n, b \in \mathbb{R}, \tag{4.1}$$

where $\langle ., . \rangle$ denotes the dot product within $\mathbb{R}^n$. In (4.1) *flatness* is achieved by *minimizing* ω. Because of space limit, we only show the case for linear function in (4.1), readers are encouraged to read [17] for the complete description of ϵ-SV regression. One of the main characteristics of ϵ-SV regression is that it is a Quadratic Problem (QP) which has a unique global solution in general.

In Section 4.1, we have shown that three candidates namely mean, variance and maximum values of speed have close correlation with k. Using ϵ-SV regression we further attempt to find the best candidate among them. We use the combinations of these candidates to increase the scope of our search. We first train the ϵ-SV regression process using the mapping of k with different speed candidates of the *training dataset*. In these mappings the values of k were chosen to approximate the corresponding trajectory segment within one meter error.[2] For each speed candidate, the training process computes the correlation function $g(s)$ as in (4.1).

We then use the *test datasets* to estimate k for different candidates by passing the values of the speed candidate (s) to the function $g(s)$ computed during the training process. We also use two different forms of datasets within the

[2]We chose the values of k to approximate Northing and Easting separately within 0.5 meter, therefore the resultant distance error is less than or equals to one meter.

Table 4.1: MEE of k estimation for different speed candidates. The smallest MEE for different forms of the datasets are highlighted in the corresponding column.

Speed candidates	MEE of k estimation			
	animal	vehicle	pedestrian	universal
mean	2.97	1.28	0.19	3.82
variance	**2.74**	2.06	**0.19**	**3.29**
max	3.67	1.05	0.20	4.41
mean, variance	2.78	1.29	0.19	3.47
mean, max	3.44	0.93	0.19	4.35
variance, max	3.10	1.05	0.19	3.95
variance, mean, max	3.25	**0.92**	0.19	4.16

training and validation process. In the first form, we use the data from individual object group (i.e. animal, pedestrian separately), whilst in the second form we combine data from all three object groups to form a "universal" dataset. In particular, we combine the training and test datasets of individual object group to form, respectively, the universal training and test datasets. The underlying idea of using a universal dataset is to investigate the performance of a universal training (training SVR using the combined datasets of different object groups) over individual training (training SVR using individual object group dataset). Note that a universal training can lessen the requirement of using individual training results for estimating k of corresponding object trajectories.

We compute the Mean Estimation Error (MEE) to compare the the performance of estimating k using different speed candidates. If there are total J segments in a trajectory dataset, MEE is computed by $(\frac{1}{J}\sum_{i=1}^{J}(\delta_E^i + \delta_N^i))$, where δ_E^i is the absolute difference between the test and the estimated k for ith trajectory segment (Easting) and δ_N^i is the absolute difference between the test and the estimated k for ith trajectory segment (Northing). MEE for various candidates are summarized in Table 4.1. We observe that for the animal, pedestrian and universal datasets, the variance of speed produces the smallest MEE, but the combination of mean, variance and maximum speed produces the smallest MEE for the vehicle dataset.

In order to use ϵ-SV regression for estimating k in-situ, we create a lookup table (ϵ-SV lookup) based on the ϵ-SV regression training results. The advantage of using a lookup table over running resource-intensive ϵ-SV regression on the nodes is the higher computation efficiency, which is crucial for resource impoverished sensor nodes. The disadvantage of this strategy is relatively lower quality estimation of k, because of the limitation of search spaces of a pre-calculated lookup table can have. We will demonstrate this design trade-off in details in Section 6.

5 AACAT Algorithm

In our current implementation, AACAT has two main threads where Thread 1 stores and compresses the data (generates projections) and Thread 2 opportunistically transmits the projections to the basestations. Memory management is handled via fixed-length linked lists, where we store the position data and the projections in two separate lists called `receiveQUEUE` and `sendQUEUE`, respectively (for Northing and Easting we use separate `sendQUEUE` and `receiveQUEUE`).

The size of the `receiveQUEUE` is chosen so that it can fit data of one trajectory segment. Each segment is identified using the timestamp of the first sample in the segment. Projections attached with the same timestamp are used to reconstruct the corresponding segment in the basestations. Northing and Easting segments are reconstructed separately and then combined to recover the corresponding trajectory segment. We also use separate lookup tables for Northing and Easting, where the values of the speed candidate(s) are rounded to one decimal place.

Thread 1 constantly takes samples from the GPS module and adds the samples to the `receiveQUEUE` when the `receiveQUEUE` is not full. Once the `receiveQUEUE` is full, Thread 1 computes the projection(s) of the samples in the queue, by first determining k using the ϵ-SV lookup table, followed by computing $m = \lceil ck \log(\frac{n}{k}) \rceil$ projections of the samples. We observed $c = 1$ is sufficient for pedestrian and vehicle datasets, but a higher value ($c = 2$) is required for animal dataset. We used iid Gaussian $\mathcal{N}(0, \frac{1}{m})$ numbers as the elements of the projection vector which can be constructed distributedly at the node and the basestations by using the same seed of a pseudo-random generator. The lookup operation selects the value of the speed candidate in the table that has the smallest difference with the speed candidate of the current segment and retrieves the corresponding k value.

If the `sendQUEUE` is not full, projections are appended in the `sendQUEUE`. Otherwise, the projections can be discarded from the `sendQUEUE` randomly because each projection is a linear combination of the samples in a segment. This property offers promising loss-distortion trade-off as in Section 6 we demonstrate that the reconstruction error increases gracefully as we reduce the number of projections. Algorithm 1 shows the pseudocode of Thread 1.

The role of Thread 2 is to detect the network connectivity. Once the network is connected, it attempts to transmit a copy of the element at the head of `sendQUEUE`. If the element is successfully received, it removes the sample at the head of the list that in turns frees up one space in the `sendQUEUE`.

6 Evaluation

We use Average Distance Error (ADE), which gives the distance between the real and reconstructed trajectory.

Consider that $N^i_\omega, 1 \leq i \leq n$ and $E^i_\omega, 1 \leq i \leq n$ are respectively the consecutive Northing and Easting data points of a trajectory segment ω, and $\hat{N}^i_\omega, 1 \leq i \leq n$ and $\hat{E}^i_\omega, 1 \leq i \leq n$ are the reconstruction of $N^i_\omega, 1 \leq i \leq n$ and $E^i_\omega, 1 \leq i \leq n$, respectively. If there are J segments in a trajectory dataset, we compute ADE by,

$$\text{ADE}=\frac{1}{nJ} \sum_{w=1}^{J} \sum_{i=1}^{n} \sqrt{(N^i_\omega - \hat{N}^i_\omega)^2 + (E^i_\omega - \hat{E}^i_\omega)^2}.$$

6.1 Transmission–Accuracy Trade-off of CS

CS achieves promising transmission versus accuracy trade-off, since it requires only a small number of projections to accurately recover a trajectory segment. We compare this trade-off of CS with that of two widely used trajectory compression algorithms: *Kalman Filter (KF)* and *Spatiotemproal* compression algorithm (SPT).

Algorithm 1 Thread 1

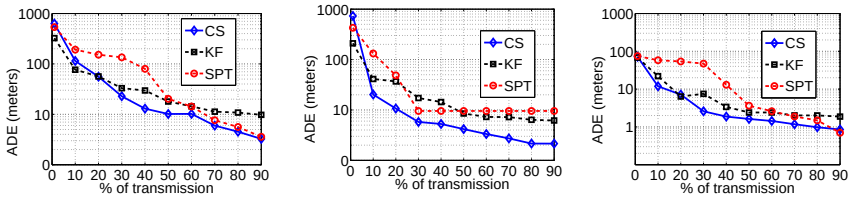
Ensure: Memory for linked list is allocated

```
receiveQUEUE ← list to store incoming position data (GPS samples)
sendQUEUE ← list to store outgoing AACAT projections
while GPS has lock do
  p ← New GPS position
  if receiveQUEUE NOT full then
    push p to receiveQUEUE
  else
    look up k of the segment at receiveQUEUE and compute m = ⌈ck log(n/k)⌉ projections.
    for all projection do
      if sendQUEUE NOT full then
        push one projection in sendQUEUE.
      else
        if Each segment has one projection then
          Randomly delete one segment.
        else
          Randomly delete one projection of a segment having more than one projection.
        end if
      end if
    end for
  end if
end while
```

KF is a stochastic, recursive data filtering algorithm widely used for system state estimation, where state estimation process operates using recursive steps of prediction and correction based on observations. We used a centralized implementation of KF where Northing and Easting were separately modeled as state variables and the predictions were made remotely at the basestations using the data acquired from mobile nodes. In order to make a fair comparison the number of bytes transmitted by CS is the same as the number of bytes transmitted by KF (Note that the number of bytes (4kB) for a projection value is the same as the number of bytes for a Northing or Easting point).

SPT is the improvement proposed by Meratina et al. [12] to the prominent Douglas-Peucker (DP) [7] method. It leverages the Synchronous Euclidean Distance (SED) [13] that is the measurement of the distance between an observed position and its estimated position based on a constant velocity model. For a triplet of points with low SED, the middle point can be removed from the trajectory with small loss of information. Similar to KF, the number of bytes transmitted by SPT is the same as the number of bytes transmitted by CS. Note that in both SPT and KF, when there is a missing data point, we use the last observed data point as the current data point.

In Fig. 6.1 we compare the transmission-accuracy trade-off of CS, KF and SPT. For CS, along x-axis is the percentage of the number of projections to the total number of data points, but for KF and SPT, along x-axis is the percentage ratio of the number of transmitted data points to the total number of data



(a) Animal trajectory (b) Vehicle trajectory (c) Pedestrian trajectory

Figure 6.1: Benchmarking CS using KF and SPT.

points. For all CS, KF and SPT along y-axis is the ADE. We consider an n data point trajectory segment which is comprised of n Northing and n Easting data points. Therefore, the total number of data points within a trajectory segment is $2n$, and the number of transmitted projections or data points is the summation of the number of transmitted Northing and Easting projections or data points, respectively. We observe that, to approximate an animal trajectory segment within 10 m error, both SPT and KF requires approximately 30% more transmissions compared to CS. However, due to higher compressibility, the difference between CS and KF/SPT is smaller for the vehicle and pedestrian trajectories.

Due to delay tolerant transmission and fixed size memory buffer, data loss is very likely to happen in the delay tolerant networks. We illustrate some examples of possible data loss in Fig. 6.2, where for different latencies along the x-axis we compute the percentage of possible transmissions along y-axis. Consider a sampling frequency h Hz and inter-connection interval (since a mobile node has only intermittent connection with the fixed node, the inter-connection interval is the time between 2 connections with the fixed node by a mobile node) ℓ seconds. The total number of projections need to be transmitted during ℓ seconds is: $T = 0.4\ell h$ (we arbitrarily choose 0.4, since from Fig. 6.1, 40% transmission approximates the animal trajectory within 10 m error). Then, for a bandwidth B kbps and connection time τ seconds (the amount of time a mobile node associates with a fixed node), the allowed transmission is $\Delta = \tau B$ kb. The percentage of CS transmission is therefore $\frac{\Delta}{TQ}$, where each projection is Q kb. In Fig 6.1, we use three different connection times, 120, 240 and 360 s, respectively and two different bandwidths, 50 and 128 kbps, respectively. We observe that unless the inter-connection interval is very small (1 or 2 min), even for high bandwidth (e.g. 128 kbps) and connection time (e.g. 360 s), it is impossible to transfer 100% projections.

One key advantage of the CS theory is that it offers promising loss-distortion trade-offs. In Fig. 6.3 we illustrate the loss distortion trade-off of CS based reconstruction. Along the x-axis is the percentage of missing projections and along the y-axis is the corresponding distance error. We first compute the average number of projections (η) required to approximate a trajectory segment within one meter error. Starting from η, we gradually reduce the number of projections, compute the corresponding error, and plot the results in Fig. 6.3. We observe that reconstruction performance degrades gracefully with the loss

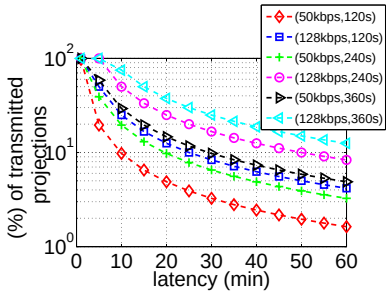


Figure 6.2: Latency versus transmission.

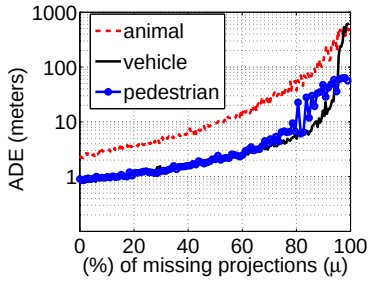


Figure 6.3: Loss distortion tradeoffs.

Table 6.1: ϵ-SV direct versus ϵ-SV lookup estimation.

Method	ADE (meters)					
	animal	animal universal	vehicle	vehicle universal	pedestrian	pedestrian universal
SVR direct	1.99±0.19	3.05±0.21	1.13±0.11	1.99±0.17	1.02±0.02	1.91±0.12
SVR lookup	3.11±1.01	3.91±1.10	2.53±0.87	3.01±1.13	1.08±1.02	2.09±1.99

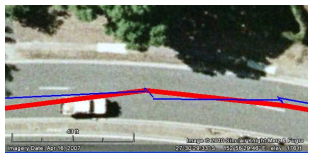
of projections. Such as, for all three object groups, for up to 60% loss of projections, the ADEs are within 10 meters. However, the reconstruction error increases rapidly (especially for animals) beyond that point (recall from Section 3.3 that animal trajectories are the least compressible among the three object groups).

6.2 The Performance of AACAT

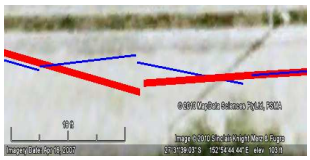
The in-situ estimation of k using ϵ-SV lookup is suitable for the resource constrained sensor nodes; however, the estimation degrades due to the limited search spaces in a precomputed lookup table. In this section, we compute the performance degradation of ϵ-SV lookup over using ϵ-SV regression directly (we call it ϵ-SV direct hereafter). For a given trajectory segment, we estimate k separately using ϵ-SV direct and ϵ-SV lookup and perform reconstruction. The reconstruction results are summarized in Table 6.1. We observe that estimation of k using ϵ-SV lookup has slightly higher approximation errors (the maximum difference is 1.40 meters) compared to ϵ-SV direct. However, the ϵ-SV lookup is significantly computationally efficient (requires only $O(1)$ operations) compared to ϵ-SV direct.

We also compare the reconstruction using universal training result with the reconstruction using individual training result. The comparison results are summarized in Table 6.1. The results in the corresponding column of a object group e.g. animals are produced while both the training and the testing of SVR process use animal dataset. On the other hand the results in the corresponding column of the "object universal", e.g. animal universal are produced by using universal dataset for the training process of the SVR, and the animal dataset for the test process of the SVR. We observe that using the training results of the universal dataset causes insignificant performance degradation, because the maximum

(a) Animal (ADE: 3.5 meters)



(b) Vehicle (ADE: 2.99 meters)



(c) Pedestrian (ADE: 2.19 meters)

Figure 6.4: The approximation of trajectory segments produced by AACAT using ϵ-SV lookup. The thick line shows the real trajectory whereas the thin line shows the corresponding AACAT reconstruction. For better illustration we have whiten the background of the animal trajectory image.

difference in reconstruction error is 1.01 meter In our ϵ-SV lookup table we therefore use the training results of the universal dataset. Fig. 6.4 shows that the approximation of trajectories by ϵ-SV lookup (thin line) is aligned closely (the maximum reconstruction error is 3.5 m) with the ground truth (thick line).

In order to validate the performance of the dynamic estimation of k, we compare the ϵ-SV lookup based estimation with the alternative approaches of using static mean, maximum and minimum values of k for CS-based reconstruction. The static mean, maximum and minimum values of k are obtained from datasets offline. Hereafter, we name the scenarios best, average and worst cases for using static maximum, mean and minimum values of k from datasets, respectively.

We compute and compare the increased transmission cost and corresponding "probability of successful reconstruction" of best, average, worst cases and AACAT. The probability of the successful reconstruction is defined as the ratio between the number of successfully reconstructed segment(s) (within one meter error) and the total number of segments. A higher probability of successful reconstruction indicates a more accurate reconstruction. For the ease of comparison, we offset the transmission cost and the probability of successful reconstruction results by the results produced by AACAT. Therefore, the results of AACAT is 0, and a positive figure indicates a higher transmission cost or a higher probability of successful reconstruction, whereas a negative number indicates a lower transmission cost or a lower probability of successful reconstruction (see Table 6.2). We observe that the best case and average case scenarios increase the probability of successful reconstruction only by a

Table 6.2: Transmission cost and reconstruction errors trade-off. A positive figure indicates higher transmission cost or reconstruction error compared to AACAT.

	Animal		Vehicle		Pedestrian	
	increased transmission	increased successful reconstruction	increased transmission	increased successful reconstruction	increased transmission	increased successful reconstruction
Best case	0.84	0.010	0.74	0.003	0.39	0.0029
Avg case	0.47	0.010	0.43	0.001	0.21	0.0021
Worst case	-4.71	-0.260	-0.18	-0.130	-0.58	-0.19

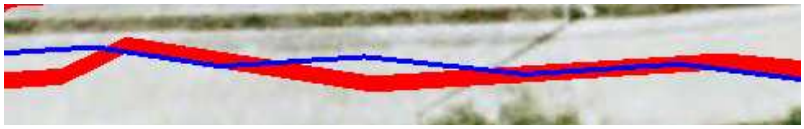


Figure 6.5: The reconstruction performance of AACAT on Fleck 3b platform. The thick line is the ground truth and the thin line is the trajectory approximation produced by AACAT. ADE is 3.67 meters.

small amount (e.g. 0.01) but with significantly more transmission overheads. The worst case scenario reduces the transmission cost but the probability of successful reconstruction is significantly smaller than that of AACAT.

6.3 System Performance

We implemented AACAT on the Fleck 3b nodes, which are in-house sensor platform based on an 8-bit Atmel Amega 1281 microcontroller with 8 kB RAM (similar to Mica family motes) and a 50 kbps Nordic NRF905 transceiver working on ISM 900 band, and evaluated the system performance with field experiments. During the experiments, each subject (human) was given a Fleck to carry around and was asked to walk along a road segment outside our lab. A fixed (connected) node was placed in the middle of the road segment which was one hop away from a basestation placed in the lab.

Two receiveQUEUEs of capacity of 512 data points were declared, one for Northing and the other for Easting. Projected values of Northing and Easting were stored separately in two sendQUEUEs with capacity of 128 projections each. Size of the queues was chosen based on available memory within the 8 kB RAM total on a Fleck 3b. Beacons were broadcast from the fixed node every second. After hearing a beacon from the fixed node, mobile nodes transmitted packets which would be forwarded to the basestations. The reconstruction result of a trajectory segment is shown in Fig. 6.5. The thick line is the ground truth which was recorded by carrying an additional Fleck 3b, recording GPS samples at 2 Hz. It is evident that the reconstruction is very close (ADE is 3.67 meter) to the real trajectory.

The projection and lookup operations are two key operations in AACAT. In Table 6.3, we summarize the memory and energy usage of one ϵ-SV lookup and one projection operations of 512 data points. The mean computation time for both operations is small. For example, it takes only approximately 16.4 ms for a Fleck3b to compute a projection, which is $16.4 \div (256 * 1000) \approx 0.006\%$ of the total time to collect one segment of data. Furthermore, the memory usage

Table 6.3: Memory and energy consumption of AACAT.

Process	current (mA)	time (ms)	energy (mJ)	memory (kB)
projection	8	16.4 ± 0.8955	0.47	1.18
lookup	8	$7.68 \pm .01$	0.18	1.67

(maximum 14% of available RAM) and the energy consumption (maximum 0.47 mJ) of the projection and lookup operations are quite low and are affordable on the resource constrained WSN nodes.

7 Related Work

Approximation of the mobile object trajectory using only partial information collected from sensor nodes has been widely studied in the past. The central themes of the previously proposed algorithms can be classified into two broad categories: prediction and compression.

Prediction can be made in both centralized and distributed fashion. In a centralized prediction technique (e.g. [8]) a basestation based on the information extracted from the movement history of an object, predicts its future movement states, which are then sent to the corresponding sensor nodes. If the prediction do not match with the sensor readings, the sensor nodes correct the basestation by sending their own readings. To save node to base communications of the centralized techniques, dual prediction techniques are proposed in [11, 20] where the predictions take place distributedly at both sensor nodes and basestation, and updates are sent to the basestation only when the prediction error exceeds some given threshold.

Both the centralized and distributed prediction techniques use either individual or group movement history for prediction. The prediction with individual history (e.g. [18, 20]) predicts the movement of an object from its own history. Considering in practice an arbitrary movement trajectory that an object may follow, the simple prediction models in the existing work results in poor prediction performances. The prediction with group history (e.g. [1, 10]) categorizes the objects into groups, and the prediction of a moving object is made based on the history of all objects from the same group. Group history provides richer information about the object movements than the individual history, but, these techniques assume a limited number of movement states (e.g. moving speed and direction) that a moving object can have. However, in practice even within the same group, different objects may demonstrate different movement patterns at different times (e.g., morning, noon and night) and/or with different tasks (e.g., surveillance and disaster response).

Compression algorithm proposed in [19] performs recursive segmentation of the trajectory, until a trajectory segment can be modeled with an interpolation function with a small error. Compression is achieved by only transmitting the relevant parameters of the interpolation function. However, compression performance of [19] has so far been evaluated using simulated trajectories without considering the resource (computation power, energy and bandwidth) usage efficiency, which is crucial in tiny embedded sensor nodes.

In [16] authors propose a low-energy adaptation of the lossless compression

algorithm (LZW) for WSN, however, a likely scenario in the VF application is that in order to cope with limited bandwidth a large amount of data may need to be discarded. Therefore, a lossless compression algorithm is not a good fit. Furthermore, the authors do not exploit temporal correlations of the data to achieve compression, which can be explored to achieve better compression ratio.

Theoretical results provided in [9] show that CS is not an efficient compression technique while applied with ordinary quantization, however our empirical results show that CS provides reasonably good compression. In particular, we show that CS provides better compression compared to two other state-of-the-art trajectory compression techniques. Furthermore, in number of other papers [14, 15] it is shown that CS provides promising compression in WSNs.

8 Conclusion

In this paper we present a trajectory approximation technique, called AACAT, which utilizes the embedded redundancy of trajectory data using the emerging theory of Compressive Sensing (CS). Furthermore, AACAT introduces a ϵ-SV regression-based in-situ compressibility estimation technique to adapt the number of required projections to trajectory data compressibility dynamically that improves the performance of CS trajectory compression. Our evaluation by three different trajectory datasets, collected by sensor nodes carried by animals, pedestrians and vehicles, shows that for a reasonable approximation accuracy, CS-based compression reduces 30% transmission overhead compared to the stat-of-the-art trajectory compression techniques driven by the classical Kalman Filter and Douglus-Peckur algorithms. Our evaluation also shows that CS-based trajectory compression is loss-resilient since the reconstrctuion error increases gracefuly with the loss of projections. Finally, an end-to-end system, which is implemented on resource-impoverished sensor nodes with 8-bit microcontrollers and 8 kB RAM, demonstrates that AACAT achieves high compression performance with very little resource (computation power and energy) overhead.

Bibliography

[1] Javed Aslam, Zack Butler, Florin Constantin, Valentino Crespi, George Cybenko, and Daniela Rus. Tracking a moving object with a binary sensor network. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 150–161, New York, NY, USA, 2003. ACM.

[2] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *In: Workshop on World-Sensor-Web (WSW06): Mobile Device Centric Sensor Networks and Applications*, pages 117–134, 2006.

[3] E. Candés. Compressive sensing. In *Proc. of the Int. Congress of Mathematics*, 2006.

[4] Emmanuel J. Candès and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[5] Emmanuel J. Candes and Michael B. Wakin. An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]. *IEEE Signal Processing Magazine*, 25(2):21–30, March 2008.

[6] Hu Cao, Ouri Wolfson, and Goce Trajcevski. Spatio-temporal data reduction with deterministic error bounds. In *DIALM-POMC '03: Proceedings of the 2003 joint workshop on Foundations of mobile computing*, pages 33–42, New York, NY, USA, 2003. ACM.

[7] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10:112–122, 1973.

[8] Samir Goel and Tomasz Imielinski. Prediction-based monitoring in sensor networks: taking lessons from mpeg. *SIGCOMM Comput. Commun. Rev.*, 31(5):82–98, 2001.

[9] V. K. Goyal, A. K. Fletcher, and S. Rangan. Compressive sampling and lossy compression. *Signal Processing Magazine, IEEE*, 25(2):48–56, March 2008.

[10] Garrick Ing and Mark J. Coates. Parallel particle filters for tracking in wireless sensor networks. In *Proceedings of the Signal Processing Advances in Wireless Communications*, NY, USA, 2005.

[11] Ankur Jain, Edward Y. Chang, and Yuan-Fang Wang. Adaptive stream resource management using kalman filters. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 11–22, New York, NY, USA, 2004. ACM.

[12] Nirvana Meratnia and Rolf A. de By. Spatiotemporal compression techniques for moving point objects. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT)*, pages 765–782, 2004.

[13] Michalis Potamias, Kostas Patroumpas, and Timos Sellis. Sampling trajectory streams with spatiotemporal criteria. In *SSDBM '06: Proceedings of the 18th International Conference on Scientific and Statistical Database Management*, pages 275–284, Washington, DC, USA, 2006. IEEE Computer Society.

[14] Giorgio Quer et al. On the interplay between routing and signal representation for compressive sensing in wireless sensor networks. In *Information Theory and Applications Workshop (ITA)*, San Diego, USA, January 2007.

[15] Rajib Kumar Rana, Chun Tung Chou, Salil S. Kanhere, Nirupama Bulusu, and Wen Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *IPSN '10: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 105–116, New York, NY, USA, 2010. ACM.

[16] Christopher M. Sadler and Margaret Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 265–278, New York, NY, USA, 2006. ACM.

[17] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[18] Yingqi Xu and Wang-Chien Lee. On localized prediction for power efficient object tracking in sensor networks. In *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 434, Washington, DC, USA, 2003. IEEE Computer Society.

[19] Yingqi Xu and Wang-Chien Lee. Dttc: Delay-tolerant trajectory compression for object tracking sensor networks. In *Proceedings of the IEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC,06)*, pages 436–445, 2006.

[20] Yingqi Xu, Julian Winter, and Wang-Chien Lee. Dual prediction-based reporting for object tracking sensor networks. In *MobiQuitous*, pages 154–163, 2004.