Trajectory Approximation for Energy Constrained Aerial Wireless Sensor Platforms

ABC
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

ABC
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

ABC
Starfleet Academy
San Francisco, California 96678-2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—The abstract goes here.**

I. Introduction

Large scale sensor networks have been used for terrestrial [1] and ocean monitoring [2]. Recently sensor platforms have been built to monitor the flying animals as well [3] [4]. For example in case of flying fox monitoring, sensor platforms are attached on these flying animals in form of a collar[4]. The most important goal of monitoring flying foxes is to record the areas they have travelled and their interactions with other species[4]. The major challenge comes from limited battery and mobility behaviour of these animals. They can fly at a speed of 7-8 meters per second. First challenge is to design the framework to offload the trajectory data from the sensor node. These bats can travel really far areas (they can travel hundreds of kilometres in a night) with no connectivity. So putting base stations in known roosting camps and waiting for bats to come there and offload data at that time is only feasible option right now[4]. Now the time between two data offloads can span from couple of days to several months. Two major challenges arise here. One, trajectory data management while these bats are away and second come from the limited data that can be transferred when bat comes in contact with ground station because of limited battery.

In order to perform the monitoring, these platforms are generally equipped with accelerometer, microphone, air pressure, GPS and inertial sensors [4]. The most energy and data intensive sensor among all these is GPS sensor. It is used to track the trajectory of these flying animals. Considering the fact that flying foxes spread the viral disease between animal of other species including humans, trajectory of these animals over a period of time is the most important piece of information from application perspective. Therefore in this paper we focus on GPS sensor and gathering trajectory data of flying foxes. However, we will keep the energy usage of other sensors in consideration while we study the affect of energy on amount of data that can be transferred to base station.

The data from all the sensors is recorded at a variable frequency which is adapted based on the amount of energy left in sensor platform at any particular time. This sensory data is then processed and stored by our resource-limited sensor platform. However very small size of this platform restricts the memory, processing and energy resources on-board. Therefore data processing and management becomes a major challenge in this resource-constrained environment.

The fact that sometimes these sensors will have to save the sensor logs of up to several months introduces a big data management challenge. The biggest challenge in this regard is introduced by limited energy. Furthermore, limited energy means limited amount of sensing, limited processing of that data and then limited time to transfer the collected data. The fact that we can only transfer a limited amount of data when flying animal comes in contact with base station raises the important question of summarising the collected trajectory data. The important fact is that the remaining energy at the time of contact with base station can be variable and is known at the time of contact. Therefore we come to know about the amount of data that can be transferred at the time of contact as well. Another important requirement of these type of applications is that amount of error should be bounded by some approximation error. it means we can not randomly choose subset of data to fit in the limit. Our It means we have to keep this variable data transfer limit in view when designing approximation scheme for trajectory summarisation.

Limited amount of available memory (around 10 KB) means that we can not use off-the-shelf compression algorithms which are not optimised for limited available memory. Therefore, we propose to build application specific approximation scheme for this type of sensor platform.

In order to develop an approximation scheme which summarise the GPS logs to make them fit to the size which can be transferred to the base station, one has to keep in mind the queries which will be applied on these logs. This process will define the error bounds applied by the end user on the summarisation process. Even though we have to keep the error bounds enforced by the application requirements. The energy limitation also poses major restriction on the approximation error. Therefore the approximation error has to be adaptive based on the available energy on the sensor platform and application queries.

In this paper, we present an online approximation scheme which is based on hexagon approximating the sensor logs

with the focus on time based GPS logs. We propose to use hexagons of the size of allowed approximation error as the basic unit of approximation. Then we propose that all the GPS points which fall into same hexagon should be approximated to the centre of the hexagon. In this scheme the choice of hexagons serve a special purpose due to its symmetry towards its neighbours. The distance from the centre of any hexagon to all its neighbouring hexagons is same. This allows us to save only the direction towards next approximating hexagon. In case of hexagon the direction can be saved in a number between 1-6 which can be saved in three bits as compared to typical 8 bytes for a single GPS point.

II. Case Study

A. AWSN-Based Adaptive Frequency Data Gathering from Flying Foxes

Our application case-study is Flying Fox Monitoring system explained in [4]. The major goal of this monitoring system is to deliver the position and activity information. They have developed a platform named camazotz to put on-board the flying foxes in order to gather this position and activity data. They hope to achieve two clear goals from this monitoring platform. 1) They would be able to identify the roosting camps of these animal which are generally in locations which are inaccessible otherwise. 2) The monitoring of their movement pattern in order to get a more detailed knowledge of their foraging choices and interaction with other species. As that is the time when they are more likely to spread the virulent diseases.

Considering the flying foxes roost together in big numbers (40-50K together) and they are quite noisy when roosting[5], simple processing of mic data can be used to get this information. However, in order to achieve the second goal, we need to gather high resolution location and activity data over the whole period of time. The activity data in this particular application includes GPS, accelerometer, air pressure and mic. The data from these sensors can be used to detect the activities like orientation of bat, temperature of the environment, current location of the bat and speed of the bat. In order to accurately detect these activities, we need these platforms to sample data at a reasonably high frequency. Sometimes as high as 1 Hz.

These are the 5 requirements from the sensing platforms developed for this purpose as described in [4].

- Collect regular daytime fixes (with an accuracy of 10m) at camps to identify new camps.
- Collect high-frequency nighttime fixes to monitor movement patterns and landscape use, and doing this with an accuracy of 10m or less using inertial sensors during fine-scale movements.
- Make daytime audio recordings to allow estimation of camp size.
- Operate over long periods, i.e. 12 months, and preferably longer.
- Provide data download capability.

The claim is that to achieve these goals, one needs clever power, sensor and data management techniques. Therefore, they have built the camazotz platform such that it has two on-board small solar panels. These solar panels fulfil the online energy requirements of this sensor platform[4].

The fact that camazotz platform needs to provide data download capability to a ground station poses a requirement that battery should not be completely flat. There should be some energy in the battery when it comes in range of ground station in order to transfer data to ground station. However, the total remaining energy when bat comes in contact with the ground station is variable and only known at that time. This limited energy poses a limitation on how much data can be transferred to the ground station while bat is at roosting camp.

This variable nature of how much data that can be transferred when the bat comes in contact with ground station poses a big data management challenge. The duration between two times when bat comes in contact with a ground station can span from 2 days to several months[4]. It means depending on the adaptive duty cycling the raw data can be as large as tens of Giga Bytes. Considering the energy and time limitation, we need to do some approximation on this large data and select smaller amount of data to send to ground station. The main challenge is that amount of data that can be transferred to base station will only be know when bat comes in contact with ground station.

B. Data Management requirement: Application perspective

Along with strict requirements of approximating the data to fit the size requirements, there are some data requirements posed by the application. These data requirements stems out of these queries:

1) Where At: Given a time value t, we need to answer GPS location of the bat at that time. This particular query poses the challenge that we have to keep time stamp of all the GPS values. It means that while approximating the trajectory, we have to approximate time with best possible resolution which is guided by our current energy budget.

2) When At: Given the GPS location, we need to answer the time t when bat was at that particular time.

3) Intersect: This is a query in which which given the approximated trajectory T, a polygon P and time t_1 and t_2. We have to tell that if P intersect T between time period t_1 and t_2.

C. Error Tolerance

As obvious from queries, the approximation of the sensor data has to be both distance ad time bounded. This requirement makes it a challenging problem. If the error was not to be bounded then virtually there is no problem at all. We could have simply sampled the GPS points at equal distances which fit in the allowed amount of data. There are two factors which influence the threshold which bounds the error of approximation.

1) Application Bounds: There are certain application requirements in every sensor network. These application requirements put a certain restriction on the amount of error that can be tolerated. In our application scenario, application specific

requirements come from the biologists. They know how much error can be tolerated in order to keep the target observations meaningful. We call this application specific requirement as application-wish.

2) Enforced Bounds: The second type of influence on error tolerance threshold comes from the sensor framework requirements. In this case as mentioned in section II-A, there are strict requirements of the sensor platform. The most important factor in this regard is limited energy. This puts the restriction of limited time to transfer data to ground station. This means we have to fit the sensor data into this enforced size. We call it as enforced-error. This enforced-error in our sensor network scenario supersedes the application-wish. This prioritisation decision is based on the following scenarios:

- Even if the application requirement is that we need the data for entire trip with an error of up to 10 meters and 2 seconds of error. And this makes the total amount of data to be far more than what can be transferred considering the amount of battery left in the mote.
- However, if there is a strict need that no matter how much data is transferred, but the error should not be greater than application-wish, then part of the data would be transferred.

III. Problem Formulation

The bat-monitoring application described in section II-A presents the problem of data management of GPS logs. Application periodically logs the GPS data at a variable frequency based on the remaining energy of the platform. As described, this GPS data is transferred to Ground Station when bats come in contact. Instead of ideal requirement of transferring all trajectory data as it is, limited energy in the sensor platform at the time of contact forces the approximation of trajectory information. The application requires that approximation error should be bounded by some ϵ which should be known.
The ϵ, in this case will be guided by the amount of energy left in the sensor mote when it come in contact with the ground station. Therefore the problem basically becomes to design an approximation scheme which allows us to calculate the ϵ to be used in case

IV. Constraints

In order to design Data Management framework, following are the constraints which are forced by the sensor platform.

A. Limited Sensing

Limited Energy on the platform in discussion affects the amount of sensing that can be performed at its core. Table I shows the power consumption of different sensors. This platform has on-board battery of 300 mAh. Now if we perform the sensing according to the duty cycling as in table I, battery would deplete as quickly as one day. It means there is strict need to build an adaptive duty cycling strategy for all the sensors which takes the available battery at any time into account. This adaptive duty cycling means that the GPS data would not come on regular intervals. It would influence the design of approximation technique.

TABLE I
Power consumption at 100% and target duty cycle of camazotz components [4].

Component	Power 100% (mW)	Duty Cycle (%)	Power DC (mW)
GPS	74	3	2.2
Radio	99	2	2
Cpu	13.2	5	0.7
Flash	40	1	0.4
Acc/Mag	2.6	10	0.1
Pressure/Temp	0.1	100	0.1
Mic	3.3	1	0.03

B. Limited Processing

The second major way energy constraint affects is the processing time which can be afforded online. For each sensing operation and later on its partial on-line processing requires CPU time which requires energy as mentioned in table I. Therefore any online processing algorithm can not be more than linear time complex.

C. Limited Time to Transfer Logs

The most important operation these sensor platforms have to perform is to transfer the logs from birds to the base station. This operation will be performed at the time when birds (in this scenario bats) are in their roosting camps.The radio should be completely active during this period. The remaining power in the battery at the time when these bats return to base station would be very limited. In order to transfer each GPS point there are three energy consumption operations which are the cost to read the data from flash, cost to perform the processing to facilitate this operation and then the energy cost of the radio in order to actually transfer the data to base station. Therefore, this particular constraint would guide on how much GPS data can be transferred to ground station and as a result what would the approximation error (ϵ) to be used in approximation scheme. Therefore any approximation scheme used should be able to determine the amount of ϵ to be used to fit gathered GPS trajectories into given amount of data.

D. Limited RAM

Another major challenge is due to limited amount of available memory (in our case 10 KB). This means that we can not use off-the-shelf compression algorithms which are not optimised for limited available memory.

V. Proposed Solution

This section provides the basic idea of our technique. We propose a k-gon based compression technique which takes the constraints discussed in section IV in account. The following section explains the technique in further detail.

A. Intuition

In order to describe the details of algorithm, we shall assume $k = 6$ when we refer a K-gon. Each hexagon has six neighbours as shown in figure 1. This fact is particularly important in our approximation technique. We start by assuming that

the first GPS point is the centre of a hexagon. The size of the hexagon is based on the amount of error which can be tolerated based on energy in the mote. Then we approximate all the points that lie within boundaries of that hexagon to the centre of this hexagon. However, as soon as we encounter

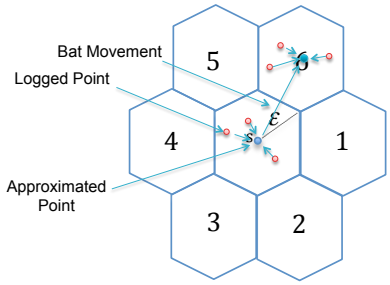

Fig. 1. Hexagon based approximation and sides coding.

a point which is out side of that particular hexagon, we calculate which neighbouring hexagon among the 6 contains this point. Then we approximate that point to the centre of this neighbouring hexagon. We label the neighbouring hexagons as shown in figure 1. This way whenever we have to move from one hexagon to next, we just have to save the label of the next hexagon. It means instead of storing one GPS point in terms of latitude and longitude, we can simply store the direction of neighbouring hexagons. With the start point already known, it is trivial to construct the GPS points from transitions codes of hexagons.

Algorithm 1 basic K-Gon based compression technique.

Input: List of GPS points, Approximation error, KGon type, Distance Type
Output: List of Codes for GPS Point **Notation:**
source: GPS Point List
ϵ: Approximation Error
$\theta(i, j)$: angle between two GPS points i and j
kT: KGon Type
dT: Distance Type
$result$: Coded Values

1: **for all** i in source **do**
2: **if** i = first point **then**
3: currentCentre = i
4: **else**
5: distance = getDistance(currentCentre, i)
6: θ = $\theta(currentCentre, j)$
7: **if** distance $>$ sideLengthAsEpsilon(ϵ, θ, dT) **then**
8: tempCurrent = currentCentre
9: currentCentre = calculateNewCentre(tempCurrent, i, $epsilon$, dT, kT)
10: addCurrentPoint(resultantPoints, tempCurrent, currentCentre, kT)
11: **end if**
12: **end if**
13: **end for**

B. Basic Algorithm

This section describes the details of our algorithm. As described earlier that we tag the bat with sensor framework at some known roosting location. It means the initial location of the bat is known from where it starts the trip. Therefore we consider this point to be the centre of first hexagon. It is shown at line 3 of algorithm 1. Each hexagon looks like the one shown in figure 1 with its neighbours labeled 1-6. In figure 1 the centre point s would be start point of the bat for first hexagon for our approximation. The distance from the centre of the hexagon to the all six edges is equal to ϵ which is basic allowed approximation error. Subsequently, we check if the subsequent point lies within the boundary of hexagon. If it is within the boundary then we approximate the point to s. Otherwise we calculate the angle between s and current point. We use this angle information to identify the neighbouring hexagon in which this new point lies in. Once the neighbouring hexagon has been identified, we calculate the GPS point which lies on the centre of this neighbouring hexagon. And this centre point is used as current hexagon centre for future GPS points and this process is repeated.

1) Angle and Neighbour Calculation: As the algorithm 1 shows that there is a need to calculate the angle between two GPS points in order to identify which neighbouring hexagon this angle points to. Figure 2, shows the how we calculate the angle between the centre of current hexagon and any new point. We have shown it with reference of North and South of the map.

One can see that total angle between two points of a side of the hexagon is 60°. Now as it can be seen from the figure 2, if the angle between the centre of hexagon and new point is between -30° and 30°, then the new pint lies in neighbouring hexagon number 1. And if it lies between 30° and 90° then it lies in neighbouring hexagon labeled as 2 and so on. In these cases the point is approximated to the centre of the corresponding hexagons.

2) Distance Calculation between the centres of neighbouring hexagons: As described in the structure of hexagons in section V-B, the distance from centre to all 6 nodes is allowed error ϵ. But the distance from centre to any point on the side is less than ϵ. Therefore, if we used the ϵ strictly and calculated the neighbouring hexagons, it would calculate it as shown in figure 3. This would leave some distance between the hexagons which would not belong to any of the hexagons. Therefore, the distance to the sides has to be calculated using pythagorean theorem based on the angle between the two points i.e. the centre of current hexagon and centre of the neighbouring hexagon. This calculation is performed by our function sideLengthAsEpsilon(ϵ, θ, dT), used at line 7 of algorithm 1.

3) Distance Type Calculation: If you have a look at our basic algorithm refk-gon-compression, you can see an input of distance type to our compression technique. After we have build our hexagons by using appropriate distances between the hexagon centres, there are still two options for epsilon to do the transition from one hexagon to other. First is that we

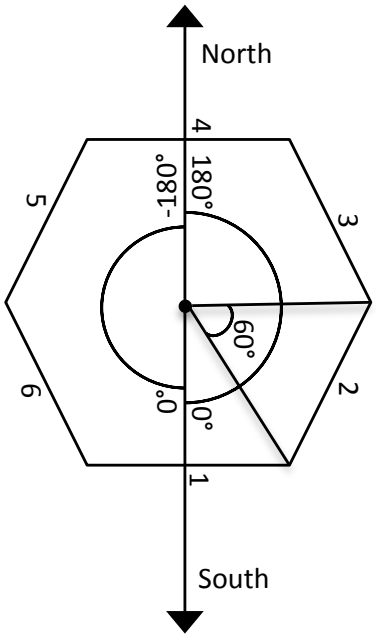

Fig. 2. Angle representation for hexagon sides.

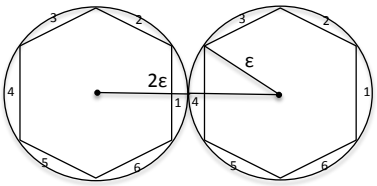

Fig. 3. Neighbours where we use 2xϵ to calculate the distance to the centre of neighbouring hexagon.

use actual epsilon to perform this transition which we refer as Loose Hexagons. Second is where we use the exact distance from the centre of hexagon to the side of the hexagon which we refer as Strict Hexagon.

4) Strict Hexagons: In this case, we find the allowed error using Pythagorus theorem. This error is the distance from centre to the point on the side of hexagon (which side to choose depends on the angle between s and new point). For example in figure 4, s is the current centre and n is new point. The epsilon calculation in case of Strict Hexagon would give the distance from s to p as allowed error. Intuitively, it seems that it is hurting us as we are not taking full advantage of the allowed epsilon in our approximation.

5) Loose Hexagons: In this case instead of using the exact Hexagon and reducing the allowed error, we use full allowed error. Any point which is out of a hexagon, we approximate it to the current hexagon. It means if a point lies in the area which is shared between two hexagons then that point is approximated to most recent one. As you can see in figure 5, most recent hexagon is 1, therefore current centre is s. The new point is n. Although this point lies outside the hexagon. But it lies within the shared area, in this case we would approximate it to s.

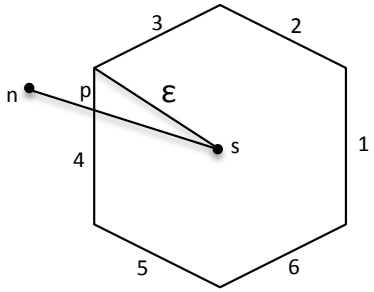

Fig. 4. Strict Hexagon definition.

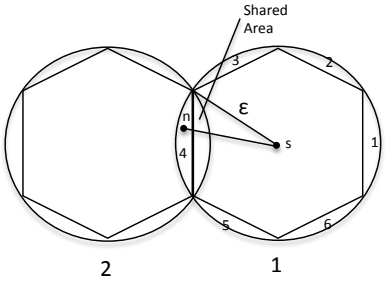

Fig. 5. Loose Hexagon definition.

6) Evaluation of Basic Hexagon compression: We have implemented both variant of our basic algorithm in java. We evaluate its performance against the trajectory approximation algorithm of Douglas-Peuker (DP). We evaluate how much reduction in points happen in case of both approximation techniques. We implemented them on the data taken from a Helicopter which took GPS logs on regular intervals over sydney.

We have changed the error from 10 meters to 1500 meters with a step of 10 meters. The results are shown in figure 6. It shows that performance of our approximation scheme is significantly better. When we compare the two alternatives of our algorithms, it confirms the hypothesis of using the loose hexagons.

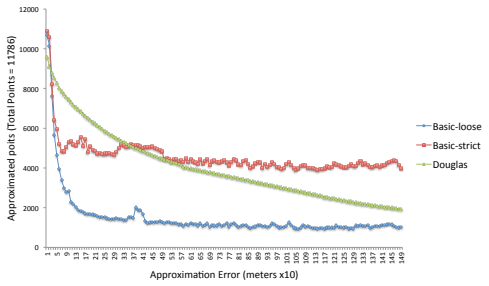

Fig. 6. Number of points after approximation with variable allowed error

However, basic algorithm works when we apply it to applicaiton data which is taken at a regular frequency. however,

when we applied to our bat-monitoring applicaiton it was not the case as mentioned in the following section.

C. Target Application data

As discussed in section II-A, our target application is bat-monitoring. After building the sensor platform, we conducted some data gathering by putting the sensors on the bats themselves. As discussed in section IV-A, the duty cycling of sensor is unavoidable. As a result, there can be a case that the time between two consecutive GPS readings become really big as can be seen in figure 7. In this graph we have ignored really big distance of more than 40k meters when the GPS was off for 9 hours during the day time. One can see that distances range from really small to as big as 10k meters. It suggests that using a fixed approximation error and use our technique would not suffice in this case. Therefore we have to adapt the approximation error based on the distance between the consecutive GPS points.

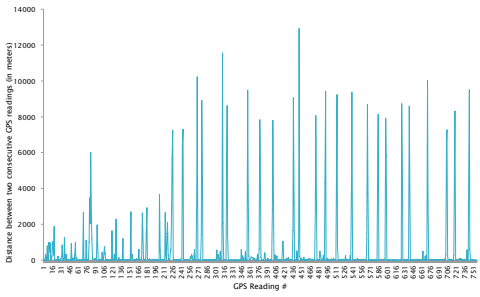

Fig. 7. Distance between two consecutive GPS points in bat-monitoring data.

In the following sections we present three techniques which we developed keeping this consideration in view.

D. Fixed-Bin based approach

The first approach we developed is based on how live GPS data from the actual measurements from the bat looks. If we have a look on figure 7, we can see that distance between two consecutive GPS point oscillates frequently between 0 to 2000 meters. However, the distance goes over 2000 meters occasionally. Therefore it makes sense to store the GPS point as it is whenever the distance between 2 GPS points goes over 2000 meters. However, any point which is less than 2000 meters away from previous centre point of the hexagon, we define the bins and put it in a specific bin.

1) Bin Thresholds: As mentioned in section V-D, the distances between two consecutive GPS points oscillates between 0 and 2000 meters. Therefore, we have decided to define the bins starting from 8 meters to 2048 meters. Here are all the codes we use for this particular scheme. One can see in table II that fixed bin technique works with 16 codes in total. It means in order to store a GPS point, we need to use 4 bits as compared to 64 bits of storing an actual point. On top of this saving we get the savings for number of points in result of hexagon based approximation.

TABLE II
NUMBERING OF THE SEGMENTS FOR BOTH ROUTES USED IN FIELD TRIALS

Hexagon Codes		Fixed Bin Codes	
Code	Hexagon Movement	Code	Distance limit
0	Same Hexagon	7	8 meters
1	First Hexagon	8	16 meters
2	Second Hexagon	9	32 meters
3	Third Hexagon	10	64 meters
4	Fourth Hexagon	11	128 meters
5	Fifth Hexagon	12	256 meters
6	Sixth Hexagon	13	512 meters
-	-	14	1024 meters
-	-	15	2048 meters

2) Interpolation based technique: In this technique we try to interpolate the intermediate points whenever the distance between current centre and new point is more than ϵ. For instance consider that in figure 8 point s is the current centre of hexagon being used for approximating current point. And next point comes in which is labeled as n with a far bigger distance than ϵ. We interpolate the points in between based on the angle between n and s.

Fig. 8. Case where interpolation is needed.

Let us consider that in this case epsilon is 100 and the distance between s and n is 800 meters. Then we add 8 point at every 100 meters based on the angle between s and n. Then we approximate using the basic technique. This way we only need 7 codes. It means we only need to have 3 bits to save a GPS pint instead of 64 bits for that point.

3) Coded-Interpolation based technique: The problem with simple interpolation scheme is that it adds to many extra points. Therefore in coded-interpolation, instead of adding all the pints we introduce a new code which specifies that next point is farther than ϵ. In this case the new code signifies that next two codes identify the angle between the s and n and $\frac{distance}{\epsilon}$.
Therefore, in this scheme we have 8 codes. 0 identifies that the GPS point is in the same hexagon. 1-6 identify the sides of the hexagon and we use 7 which identifies that next two codes are special codes. First of them is the angle to use to calculate the hexagon side and second is the number with which we have to multiply the ϵ in order to get the distance to be used with angle for the calculation of centre of next hexagon. it means we can really do with three bits in this case.
Here it is important to note that the precision of angle is very important in this case. We tried rounding of the angle with 0 decimal places but it introduces an error which then aggravates and found that accuracy till 2 decimal places worked perfectly.

E. Discussion about influence of Energy Constraint on data transfer

As discussed in section IV-C, there is limited amount of data that can be transferred to ground station. This amount is known as soon as the bat comes in contact with the ground station. Till now we have two types of trajectories, one is full GPS trajectory and other is the one coded with our approximation scheme discussed in above section.
Considering that we have the knowledge of amount of data that can be transferred, we need to know the ϵ which we need to use in order toe fit the whole trajectory data in that amount while keeping the hausdorff distance from original trajectory minimal and better than other approximation schemes in literature. It looks like if we doubled the size of ϵ, the number of hexagon in our approximation would decrease quadratically. However, this is not the case. As it would depend on the shape of the approximated trajectory as well.

F. How to find the appropriate epsilon to approximate the trajectory to fit in required size

The first option is that based on the the amount of data that can be fitted in the given energy budget, choose that many points with equal distances. However there is a basic problem in this approach.
Let us say that there are 8000 GPS points and each of them is stored in 4 bits. It means it takes 12Kb are required to store these points. However, energy budget allows the transfer of 2Kb when this bat comes in contact with ground station at a roost camp. And 2Kb allows 500 points. The total distance covered by bat is 50 kilometres. Now assume that these GPS points are divided in two sets of points A and B based ono distances. And the points in both of these sets are very close to each other and both sets are 30 kilometres far apart from each other.
Using naive approach, we can say in order for all the points to fit in our energy budget of 500 pints, we can choose one point every 100 meters. However, in above scenario we will at most be able to choose 200 points when we could have transferred 500 points in that energy budget. It means that we need a better approach to address this particular challenge.

G. K-Gone based compression and finding the right epsilon

Now considering the proposed KGon-based approximation scheme, we have to find the right epsilon with which to approximate the original GPS trajectory to fit in the allowed data limit imposed by energy budget when sensor comes in contact with ground station. Now the important question is how do we do it. When the sensor comes in contact with the ground station, we get to know the amount of data that can be transferred. Let us say that total size of the original trajectory data is K bytes. The amount of reduction we get by using KGon based approximation scheme with a specific ϵ of k meters depends on multiple factors which includes the shape of actual trajectory, size of k and frequency at which GPS points were taken as well. Therefore finding the right epsilon at the run time looks a hard problem as we have seen that naive approach would not work as explained in section V-F.

However, when the bat comes in contact with the base station it has an approximated trajectory which is created using base allowed epsilon (BAE) using KGON-approximation scheme. Now with this basic approximation we have a base-line information on what is the amount of compression we get by using a particular epsilon with KGon based approximation scheme. Now the point is that is this information of any value when we are looking to find the epsilon to approximate the trajectory with such that it is reduced to the allowed size?

What would be the affect on the compression we get if we used $2 \times$ BAE as epsilon? Would we be able to reduce the number of points 6 times if we used hexagons? The answer again is that it depends on the shape of trajectory. Then the question is what is the worst possible compression if we used double the size of BAE as epsilon. Would it be the case that the new number of points required would be half than original points (if approximated with BAE)? The answer is not necessarily. The reason for that is the case when the distance between two points is more than $2 \times BAE + BAE$ as shown in the figure.

We can not use hit and trial based method to find this right epsilon considering limited energy as this process would consume lots of energy. It calls for a need to come up with a way to find the amount of increase in compression we get in worst case by using a specific multiple of BAE. We describe our approach in which we adapt our KGon based approximation scheme to support this kind of information without requiring to do a hit and trial at the run time when bat comes in contact with the base station.

H. Adaptation of KGon Approximation to energy

Therefore we have proposed the adaptation of our approximation scheme to find the epsilon to be used to get that much approximated data to fit in the allowed data limit. We propose to maintain two basic counts when we approximate the original trajectory with BAE in order for us to estimate the worst case savings when we recompress with an epsilon which is a multiple of BAE. First count maintains the number of those points which will be reduced to half when we double the size of epsilon i relative terms. And second count is where we

VI. Evaluation Results

A. Simulation Setup

We perform simulation for all the techniques described in section V. We build our simulation framework in java. This framework takes five types of inputs from the user as command line arguments. These are as follows:

- Epsilon(ϵ): The amount of allowed error
- The choice of algorithm to be used
- The type of KGon to use, it means size of k
- The distance type to be used
- Threshold to be used if it is Fixed Bin based approach

Currently our implementation works for Hexagon and Octagon which means value of k can be either 6 or 8. However, this report does not include the results for $k = 8$. It implements the KGon based functions in our algorithm in two separate classes. In order to perform the comparison, along with implementing compression techniques, we implement the decoding techniques as well.
We use the GPS trajectories gathered from sensors on-board the bat which were collected from initial experiments. We use the variable ϵ. We vary it from 0 to 1500 with the step of 10 meters. We do the comparison of our approximation scheme with DP which is briefly explained below.

B. Douglas-Peuker

Given the trajectory of whole path, DP recursively divides the line. First and last point are always included in the approximated trajectory. Then it takes the first and last point and finds the farthest point from the line made by these two points, let us call it m. If m is closer than allowed ϵ, then DP discards it along with all the pints between this point and the line. Otherwise, if it is farther than ϵ, then trajectory is divided between two trajectories. First has all the points between first point and m and second has all the pints between m and last point. Then recursively, same procedure is applied on these two trajectories.
At the end approximated trajectory is given as output which is subset of the original trajectory.

C. Evaluation Metrics

We use three type of evaluation metrics in order to do the comparison of our techniques with DP. They are detailed as follows:

1) Size of the Trajectory data: First metric is the total size of Approximated trajectory data. We compare the size of of the approximated trajectories by different approximation algorithms.

2) Number of GPS Points: Then we compare the number of points we get after the reduction by our approximation techniques

3) Hausdorff Distance: Last evaluation metric provides the qualitative me sure of the approximation scheme. Hausdorff distance is used to measure the quality of approximation schemes by finding out the distance of approximated trajectory from original trajectory. Researchers have already used it for qualitative analysis of their approximation schemes[6].
Let us say that M is the distance between a point and a line. And the distance $d_M(p, T)$ is the minimum distance between point p and all the line segment in trajectory T. Then Hausdorff M-distance between trajectory T and its approximation T' is defines as:

$$\tilde{D}_M(T, T') = \max_{p \in T} \quad d_M(p, T')$$

D. Results

This section shows the results obtained by approximating the trajectory of bat from different approximation techniques.

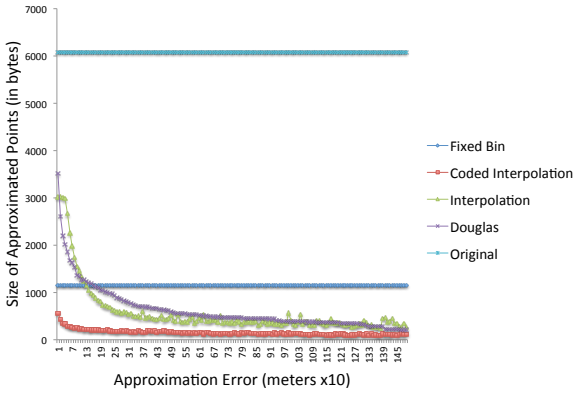

Fig. 9. Total storage taken by approximated trajectories.

1) Storage Space Savings: Figure reffig:storage-size shows the total storage taken by each approximation technique. In this case one can see that when our techniques compared with original size which is 6080 bytes, take considerably less amount of storage. In case of coded interpolation based scheme, it takes 11 times less space than original trajectory with ϵ = 10 meters. When compared to DP at 10 meters approximation error, it takes 6 times lesser space. Here one can see that we have shown only one straight line in case of Fixed Bin based approximation. As explained in section V-D, fixed bin technique is based on the data obtained from the bat. And when we define fixed bins then it does not make sense to change to binning with larger ϵ. This fact limits the adaptivity when we try to use larger base ϵ. This is the major drawback of this technique.
As the ϵ goes over 130 meters, even simple interpolation based technique performs better than DP. The reason for that is lesser intermediate points between points which are far from their neighours. However, Coded Interpolation based scheme performs best. The basic reason is the double advantage we get by both approximating the points in the hexagon to one point and coding of GPS points to hexagon sides. It performs considerably better than simple interpolation based scheme because it does not introduce multiple points in between distant neighbouring points. We can notice that as the size of ϵ increases the total size of approximated trajectories decreases as well. This affects the simple interpolation based scheme the most. Because it decreases the number of points introduced between distant neighbours.

2) Total Number of Approximated Points: In figure reffig:total-points, we plot the number of points obrtained in result of approximation of a trajectory with 760 total GPS points. The analysis is based on the decoded points from our approximation schemes. Immediately one can see that simple interpolation performs really badly. It approximates to more than original points. The reason for that is the intro duction of intermediate points between distant neighbours. However, we can see that coded interpolation gives as much as 62% improvement on the number of points

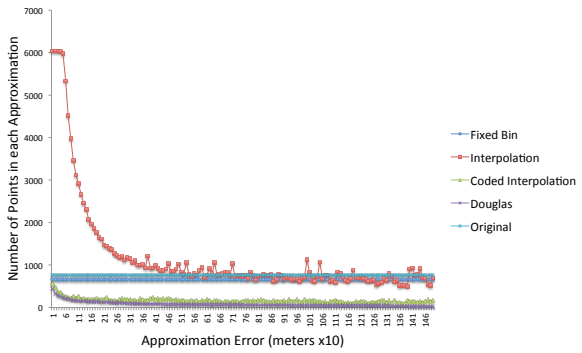

Fig. 10. Total points in approximated trajectories as compared to original trajectory.

at ϵ = 50 meters. Even though we use 6 times less storage in term of space for coded interpolation, our total number of points is not as much asone would think. At ϵ = 50 meters, DP approximates to 232 pionts compared to 292 of Coded Interpolation.

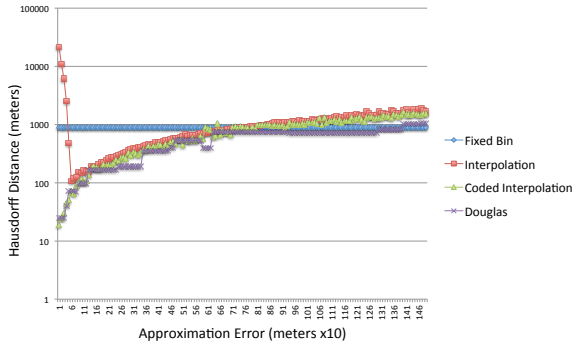

Fig. 11. Hausdorff distance between original and approximated trajectories.

3) Hausdorff Distance: We use hausdorff distance as metric for qualitative analysis of out approximation schemes. We calculate the hausdorff distance between the approximated trajectory with original trajectory. One can see that hausdorff distance between coded interpolation and DP is comparable. With smaller ϵ, hausdorff distance of Coded Interpolation scheme is actually better than DP. The major thing to see is that Hausdorff distance is bounded by approximation error (which is shown at x-axis). This is the desirable property needed from application perspective.

4) Loose Hexagon vs. Strict Hexagon: In this section, we compare the loose and strict hexagon versions of our approximation techniques as described in section V-B. Figure 12 shows the total number oF approximated points by all three approximation schemes. And as discussed earlier the loose hexagon versons performed better. The reason is the full use of allowed approximation error. The difference is quite visible in case of simple interpolation based schme as one would expect. The reason is the introduction of large number of points and hence large number of hexagons and more use of shorter allowed error than allowed. However, if you look

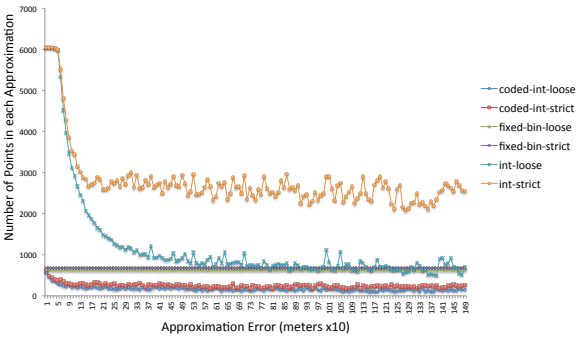

Fig. 12. Total points in approximated trajectories as compared to original trajectory with Loose vs. Strict Hexagons comparison.

at Codded Interpolation scheme that is also affected. Loose Hexagons perform a bit better than strict.

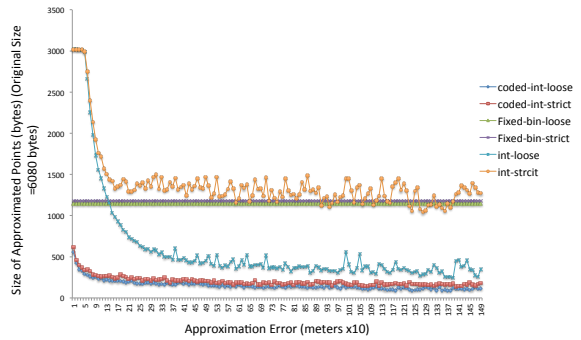

Fig. 13. Total size of approximated trajectories as compared to original trajectory with Loose vs. Strict Hexagons comparison.

We evaluated the size of approximated trajectories by both variants of approximation techniques as well. And we can see that same trend continues in figure 13 as well. It was already quite apparent from the total number of points we got from approximation.

VII. Related Work

The storage limitation suggests that we need to look at algorithms to reduce the amount of information to be stored in addition to applying the standard or new compression techniques. GPS trajectory simplification problem can be mapped to a line simplification problem. The most referenced heuristic for Line Simplification is Douglas-Peuker heuristic[7]. Furthermore problem has been studied in detail in many disciplines like Computational Geometry[8][9], Computer Vision[10] and Graphical Information System[7][11]. However, these line simplification algorithms require equivalent of O(N) memory and frequently require expensive update operations. Database community has been fairly active in the area of query optimisation. For example, Cao et al. have developed a framework to get error-bounded answers of spatio-temporal queries[6]. They define the type of distance functions that are sound to provide error-bounded answers to different queries on spatio-temporal trajectories. Given that the optimisation is based on queries, the

work can not be directly applied to our system. Compression algorithm to compute a set of corset points C whose size is independent of input size N was recently presented[12]. It also suffers from O(N) memory and flash constraints. Sadler et al. have proposed an extension to famous LZW compression scheme for low energy networks. Again they have not kept flash constraints in view.

While FAST[13] proposes a generic framework to maintain flash-aware spatial trees. They do not consider the limitation of resources like limited amount of RAM of sensor platform. Nath et al. have proposed an energy efficient data logger for flash-based devices based on amnesic compression[14]. They claim their data logger can be used with any compression algorithm. While they have done well in considering the flash constraints for data logger. However in our view the compression algorithm can not be seen as an independent module. If the compression algorithm itself requires update operation on flash or very large working memory then it is not suitable for lightweight sensor nodes. Another Amnesic Compression based Time Series data summarisation scheme has been proposed by Palpanas et al.[15]. They have proposed an algorithm which bounds the error with a given threshold. However, they have not considered flash constraints for summarisation scheme. Therefore, there is a need to design the compression technique with flash constraints in mind.

VIII. Conclusion and Future Work

We establish that there is a need to approcimate the GPS trajectories recorded on-board an energy constrained sensor platform. Then we proposed and implemented an approximation technique for GPS location data collected from on-board GPS sensor on a flying foxes. We showed that our technique achieves up to 11 times better approximation as compared to origial trajectory in terms of storage size. And when compared with widely used Douglas-Peuker (DP) algorithm, our scheme perfomrs as better as 6 times.

We have implemented the basic form of our aproximation technique and future directions from here include following:

- So far, we have tested our approximation scheme by testing it by varying the error from 10 to 1500 with 10 meters steps. Although it gives good view of how it'll perform in case of different approximation errors. In order to close the loop, we have to build the energy model and start taking its input as allowed ϵ for our approximation technique.
- We have built the approximation scheme for trajectory information. However, in order to answer the time-based queries, there is a need to approximate the time values as well. Therefore, including the timestamps in the approximation is a requirement.
- Along with energy constraint, another important constraint which we did not focus on in this work is flash storage. The sensor platform has two levels of flash constraints. First, it has two different types of flashes which may differ in terms of energy usage for read, write and update. Therefore in addition to approximation, we have to make this decision of which storage to use when. The other flash constraint comes in when the storage completely fills up. Then updating the old data would be crital operation. Therefore, we have to evaluate how our approximation scheme performs in that case and update it in case it does not matches with those requirements.
- We have discussed using different sized KGons for our approximation scheme. However, in this paper we have only included the analysis of hexagons. We plan to try OcatGon and DecaGon in future and see the affects of this on our approximation scheme.
- We plan to implement our technique on the real motes and identify how it works with actual energy usage on-board.

Acknowledgment

The authors would like to thank...

References

[1] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. Sukhatme, "Robomote: enabling mobility in sensor networks," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 404–409, 2005.

[2] I. Vasilescu, K. Kotay, D. Rus, L. Overs, P. Sikka, M. Dunbabin, P. Chen, and P. Corke, "Krill: An exploration in underwater sensor networks," in *Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on*, pp. 151–152, 2005.

[3] D. Anthony, W. P. Bennett, M. C. Vuran, M. B. Dwyer, S. Elbaum, A. Lacy, M. Engels, and W. Wehtje, "Sensing through the continent: towards monitoring migratory birds using cellular sensor networks," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, IPSN '12, (New York, NY, USA), pp. 329–340, ACM, 2012.

[4] R. Jurdak, P. Sommer, B. Kusy, N. Kottege, C. Crossman, A. McKeown, and D. Westcott, "Camazotz: Multimodal activity-based gps sampling," in *Proceedings of the 12th ACM/IEEE conference on Information Processing in Sensor Networks*, IPSN '13, (New York, NY, USA), ACM, 2013.

[5] L. A. Shilton, P. J. Latch, A. D. A. M. Mckeown, P. Pert, and D. A. Westcott, "Landscape-scale redistribution of a highly mobile threatened species, pteropus conspicillatus (chiroptera, pteropodidae), in response to tropical cyclone larry," *Austral Ecology*, vol. 33, pp. 549–561, June 2008.

[6] H. Cao, O. Wolfson, and G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds," *The VLDB Journal*, vol. 15, pp. 211–228, Sept. 2006.

[7] "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*, vol. 10, pp. 112–122, Oct. 1973.

[8] P. K. Agarwal and K. R. Varadarajan, "Efficient algorithms for approximating polygonal chains," *Discrete and Computational Geometry*, vol. 23, no. 2, pp. 273–291, 2000.

[9] W. CHAN and F. CHIN, "Approximation of polygonal curves with minimum number of line segments or minimum error," *International Journal of Computational Geometry and Applications*, vol. 06, no. 01, pp. 59–77, 1996.

[10] J. D. Hobby, "Polygonal approximations that minimize the number of inflections," in *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, (Philadelphia, PA, USA), pp. 93–102, Society for Industrial and Applied Mathematics, 1993.

[11] R. McMaster, "Automated line generalization," in *Cartographica*, pp. 24–2, 1987.

[12] D. Feldman, A. Sugaya, and D. Rus, "An effective coreset compression algorithm for large scale sensor networks," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, IPSN '12, (New York, NY, USA), pp. 257–268, ACM, 2012.

[13] M. Sarwat, M. F. Mokbel, X. Zhou, and S. Nath, "Fast: a generic framework for flash-aware spatial trees," in *Proceedings of the 12th international conference on Advances in spatial and temporal databases*, SSTD'11, (Berlin, Heidelberg), pp. 149–167, Springer-Verlag, 2011.

[14] S. Nath, "Energy efficient sensor data logging with amnesic flash storage," in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, (Washington, DC, USA), pp. 157–168, IEEE Computer Society, 2009.

[15] T. Palpanas, M. Vlachos, E. Keogh, and D. Gunopulos, "Streaming time series summarization using user-defined amnesic functions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, pp. 992 –1006, july 2008.