Appendix A

Data Description and RTM Application Specifics

The development of the real-time monitoring system described in this research was made under the auspices of the organization Save the Elephants (STE) based in Kenya. Collection of elephant movement data using Global Positioning System (GPS) tracking collars by STE capable of telemetering data in real-time began in May 2002 with the deployment of the first satellite-based tracking collars. The real-time monitoring (RTM) system was developed, beginning in 2004 over several years and *Geofencing* was the first algorithm implemented in 2007. However, our discussion below and within the manuscript is limited to our second generation system that we established in June, 2012 with the activation of the four RTM algorithms on an Amazon Elastic Compute Cloud (EC2) server and we therefore consider only those datasets being actively received by the RTM system as of June, 2012 and onwards.

1 Collar Description & Type

Movement data is collected from elephants by affixing a tracking unit using a length of belting around the neck of the animal (mean circumference: 260 cm (female), 330 cm (male)) following chemical immobilization by a veterinary doctor. The tracking unit is designed to rest at the crest of the shoulders on the animal's neck while a counter-weight at the opposite side of the belting loop ensures the unit will remain facing skyward.

A typical tracking unit contains a GPS receiver, non-volatile memory for on-board storage of

Table A1. Collar Model Summary.

Manufacturer	Model	Transmission Method	Battery Capacity (Amp/Hr)	Units Deployed
AWT	A \ AM	SMS	91	10
AWT	AG	TCP/IP	143	15
AWT	MT100	TCP/IP	130	62
ST	GL100	TCP/IP	80	5

data, a Very High Frequency (VHF) radio beacon, and either a Global System for Mobile (GSM) or Satellite communications modem for telemetry of recorded positions. Certain models also house a temperature sensor for recording ambient temperatures.

Tracking units for real-time monitoring were purchased from two commercial collar manufacturers: *African Wildlife Tracking* (AWT) based in Pretoria, South Africa (http://www.awt.co.za/) and *Savannah Tracking* (ST) based in Nairobi, Kenya (http://www.savannahtracking.com/). A summary of collar models is presented in Table A1. Using either the GSM or Satellite network infrastructures, data from a unit is telemetered using either the Short Message Service (SMS) or via a data connection using Transmission Control Protocol/Internet Protocol (TCP/IP). The SMS format limits message content to 160 characters, therefore recent collar models have adopted the use of data connections in preference to SMS since transmission is faster, cheaper and there is not a limit to the amount of data that can be sent.

2 Temporal Sampling Regime

Data were most frequently sampled at 1-hour intervals (Table A2). Lower sampling frequencies were sometimes selected for operational reasons, such as where re-collaring operations would be difficult and the lifetime of the unit was selected in preference over more up-to-date information on the animal's whereabouts (although Table A2 suggests that selection of a less frequent sampling interval does not necessarily result in a longer-duration dataset). There are many variables involved

Table A2. A summary of tracking datasets. Two collars were re-deployed and used to collect more than one dataset and therefore a total of 92 collar units have led to 94 datasets. Data collection is ongoing and 78 units are still currently active.

Sampling Frequency (Hours)	Dataset Count	Median Duration (Days)	Max Duration (Days)
1	59	562.41	2409.88
4	31	489.06	2532.96
8	3	1092.35	1175.83
24	1	1079.49	1079.49

that confound establishment of a relationship between sampling frequency and dataset duration. In several cases for example, elephants were shot and killed by poachers for their ivory tusks, thus truncating a potentially longer dataset. Several collar units were also simply dropped by animals, or suffered damage, causing a premature termination in the collection of a tracking dataset. Field operations and conditions can also create delays between the time a collar is manufactured as compared to when it is deployed onto a given animal and therefore the battery is not always at full capacity on deployment. The transmission method - GSM or Satellite - affects transmission power consumption but is severely dependent on the units's ability to obtain a decent connection and the number of attempts made for each transmission. For example, units on the fringes of GSM network coverage would expend more energy transmitting than a unit in close proximity to a GSM tower, but there is very little way to model or predict this beforehand. Weather conditions such as humidity and temperature can also affect the transmission capability of a collar. Vegetative canopy and terrain are also major factors in a collar's ability to transmit data as well as the GPS constellation configuration. Further study is needed to establish the potential relationship between unit battery capacity, transmission method, longevity and dataset quality for different tracking collar models.

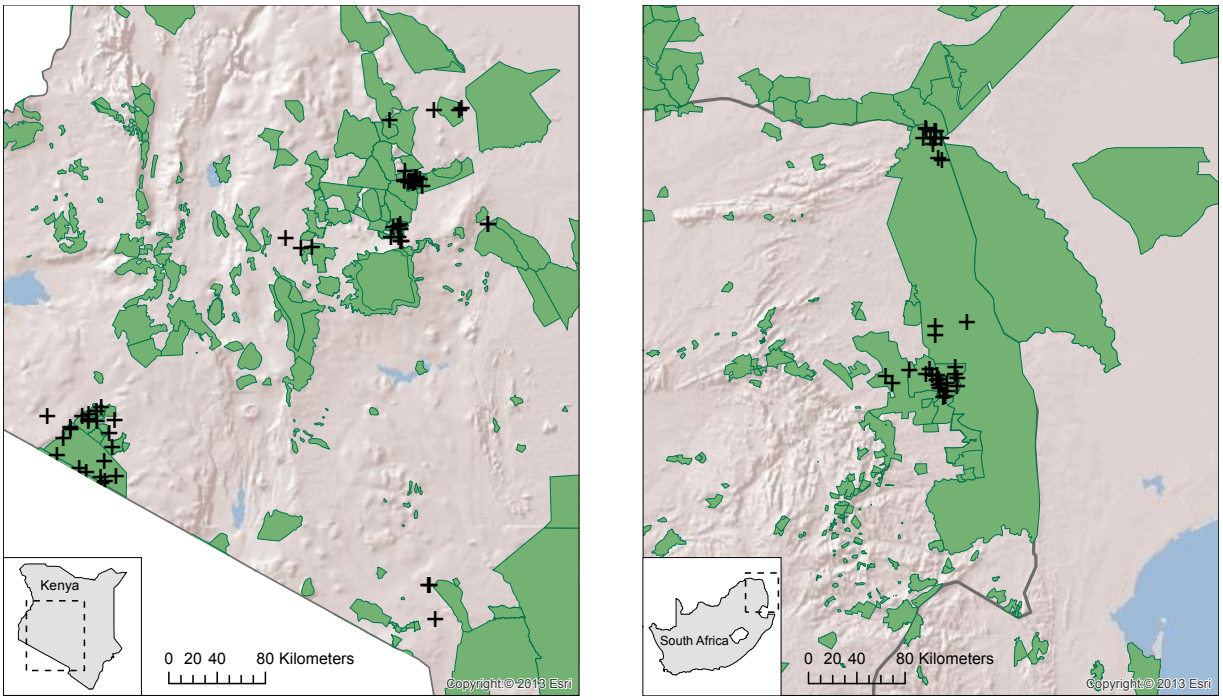

Fig. A1. Collar deployment locations in Kenya and South Africa for collars used in the RTM program. Green polygons demarcate protected areas (source: World Database on Protected Areas (WDPA, 2013)) and elevation hillshade shows terrain (Source: Esri, 2013).

3 Deployment Locations

Collars were deployed onto elephants in both Kenya and South Africa (Figure A1). An approximately even distribution of collars onto males and females was made. Selection of individuals was a mix between choosing individuals known to researchers in the study areas, and randomly selecting animals for collar deployment. Large bulls and females with big tusks have become increasingly favoured for real-time monitoring and selection when deploying collars. Other choices include individuals known to crop-raid frequently, or who are believed to have interesting and far ranging movement.

4 Data Telemetry

Tracking data from AWT collars is telemetered using one of three methods. 'A' and 'AM' units transmit via SMS to a server located in South Africa while AWT 'AG' units telemeter data using

a TCP/IP data connection. Data from all three unit types are stored in a database operated by AWT and made available for download using a Hyper Text Transport Protocol (HTTP) Application Programing Interface (API). Data from AWT Satellite units are telemetered using a data connection via the Inmarsat satellite constellation to a data download center located in Australia. These data are made available via a Simple Object Access Protocol (SOAP) API called the '*Sky-Q*' API.

We developed customized software called '*AnimalLink*' using the Microsoft C# programming language and .Net 4.0 Class library and the Environmental Systems Research Institute (Esri) Engine Runtime version 10.1 software components to gather data from multiple tracking unit sources and store them in a single PostgreSQL v8.4 '*AnimalTracking*' database. *AnimalLink* runs on an Amazon Web Services Windows Server 2008 instance hosted in the Amazon EC2 environment (Figure 1 of manuscript). *AnimalLink* uses both the AWT HTTP API and AWT Sky-Q API for collection of AWT tracking unit data. At present, both AWT APIs are queried every three minutes for the most recent collar locations. Data from Savannah Tracking GL-100 units are telemetered directly to our server using a TCP/IP data connection which is continuously monitored by *AnimalLink*.

5 Data Storage

Datasets within the *AnimalTracking* database are identified by a '*chronofile*' number that defines a single collar identification value as being attached to a unique animal for a given period of time. An animal, defined uniquely with a name, may have several chronofiles corresponding to multiple collar deployments on the same animal but never temporally overlapping. GPS records are stored in a single table '*archive_loc*' in their native latitude/longitude format with reference to the WGS84 spheroid. Time information is stored in Universal Coordinated Time (UTC). A database join with a '*TrackingMaster*' table recovers information about the dataset such as the collar ID number, animal name, animal sex, etc. A '*TrackingUser*' table stores details about users of the tracking system including their contact information and *chronofile* access list.

6 Data Analysis

Customized *MovementMonitor* software was developed using the Microsoft C# programming language and .Net 4.0 Class library and the Esri Engine Runtime version 10.1 software components to be able to analyze new data after it is stored within the *AnimalTracking* database. Each of the 4 real-time monitoring algorithms is run in continuous succession for each active chronofile found within the *TrackingMaster* table. Spatial data used in the *Proximity* and *Geofencing* algorithms is stored in an Esri SDE enabled PostgreSQL 8.4 database (*STESpatial*). Each algorithm has a database table (e.g., '*GeofenceMaster*') that stores the algorithm-specific parameters for a particular chronofile. In this way, the algorithms are customizable to each animal being tracked and parameters can be varied as necessary. Below we provide details and pseudocode on each of the four described algorithms: *Proximity*, *Geofencing*, *Movement Rate* and *Immobility*. The algorithms only search movement data within a specified recent temporal window to ensure that any triggered alerts are within a time frame of interest (e.g., within the last 24 hours).

6.1 Proximity

The proximity algorithm uses a set of spatial features (represented as polygons) stored within the *STESpatial* database, to assess the proximity of an animal with each spatial feature of interest. A '*ProximityMaster*' table defines the subset of spatial features that are relevant to a particular animal and the proximity distance threshold value configurable to a particular chronofile.

Algorithm A1: Proximity algorithm pseudocode

```
foreach(animal)
{
        Select animal's LastPosition
        MaxSearchTime=Now-24 hours
        ThresholdProximity=500 meters
        foreach(SpatialFeature)
        {
                Proximity=ShortestDistance(LastPosition,SpatialFeature)
                if(Proximity<ThresholdProximity)
                {
                        SendAlert()
                }
        }
}
```

6.2 Geofencing

The Geofence algorithm determines where an animal's straight-line track (calculated between the LastPosition and the PenultimatePosition) crosses a particular geofence. Geofences are a set of line features stored in the *STESpatial* database. A '*GeofenceMaster*' table defines a subset of geofences that are relevant to a particular animal. Linear interpolation of the break-point between the start time and end time of the animal's track is used to estimate the time of the geofence break.

Algorithm A2: Geofence algorithm pseudocode

```
foreach(animal)
{
        Select animal's LastPosition and PenultimatePosition
        MaxSearchTime=Now-24 hours
        Track=CreateStraightLine(LastPosition, PenultimatePosition)
        foreach(Geofence)
        {
                Crosses=Intersection(Geofence, Track)
                if(Crosses==true)
                {
                        SendAlert()
                }
        }
}
```

6.3 Movement Rate

The movement rate algorithm calculates the temporal sum of the distance traveled by an animal within a set temporal window and compares the value with a distribution of known 'normal' values (i.e., spanning a mix of different but acceptable behavioral modes). An alert is generated if the value falls below what is expected (i.e. a chosen percentile value of the normal-behaviour distribution). A statistically viable sample of movement distances within a normal activity period (e.g., the first two months of movement data) is chosen to establish the distribution of 'normal' movement activity for a given individual. The percentile (e.g., 1%) is calculated by *MovementMonitor* software and this value is stored in a database table '*MovementRateProfiles*'. The period of normal activity can vary between animals and can be re-calculated as more data becomes available. Results of application of the algorithm to a wounded African elephant are provided in Figure 4 of the main article.

Algorithm A3: Movement Rate algorithm pseudocode

```
foreach(animal)
{
        MaxSearchTime=Now-24 hours
        LocationDataArray=GetGPSData(MaxSearchTime, animal)
        CumulativeDistance=0
        for(int i=1; i < LocationDataArray.Count;  i++)
        {
                TrackSegment=CreateStraightLine(LocationDataArray[i], LocationDataArray[i-1])
                Dist=CalculateDistance(TrackSegment)
                CumulativeDistance+=Dist
        }
        Below=Compare(Cumulativeistance < 1% percentile Value (Normal Activity Distribution))
        if(below==true)
        {
                SendAlert()
        }
}
```

6.4 Immobility

The Immobility algorithm searches for clustering of data-points that fall within a critical radius and that extend beyond a biologically relevant threshold time. The algorithm is based on agglomerative weighted centroid clustering (Legendre and Legendre, 1998) similar to that used by Knopff et al. (2009) and continuously updates a 'Cluster' through the addition of successively acquired positions. The cluster radius is calculated as the mean distance of each point in the cluster from the cluster's centroid. The *MovementMonitor* software references a database table '*ImmobilityProfiles*' to look-up algorithm parameter values for a particular chronofile.

Determining the critical radius value is a trade-off between missing actual clustering of data-points by making the radius too small, or setting the radius too large and triggering false positives (misidentifying a non-cluster as a cluster). The threshold time is species specific and depends greatly on how long one would reasonably expect an animal to remain stationary. For African elephants we estimate this at 5 hours. A within-cluster percentage value is also defined to allow for the occasional positional error and therefore the calculation depends only on a percentage of points occurring within the critical radius.

We tested the performance of our algorithm on six African elephant movement datasets where the animal had been killed by poachers but where the tracking unit continued to function post-

mortality. We ran our test using a month of positions preceding the animal's mortality and twenty-four hours of positions post-mortality. The clustering algorithm was repeatedly run over each dataset by varying the cluster radius value between 1 meter up to 60 meters using a moving window tool and while keeping a threshold time of 5 hours and a within-cluster percentage of 80%. Results are found in Table A3. We found that a cluster radius of 13 meters successfully picked up on all mortality events.

Algorithm A4: Immobility algorithm pseudocode

```
foreach(animal)
{
        Create a new Cluster
        Set Cluster.Time=Now
        Set Cluster.CriticalRadius=CriticalRadius
        Set  ThresholdTime=Now-5 hours
        Set MaxSearchTime=Now-24 hours
        LocationDataArray=GetGPSData(MaxSearchTime, animal)
        for(int i=0; i < LocationDataArray.Count;  i++)
        {
                Cluster.AddLocation(LocationDataArray[i])
                Cluster.ComputeCentre
                numWithin=Cluster.DetermineNumberPointWithinCriticalRadius
                numTotal=Cluster.TotalNumberPositions
                percent=numWithin/numTotal
                if(percent>=CriticalPercentage)
                {
                        SendAlert()
                        Exit Algorithm
                }
        }
}
```

Table A3. The critical radius values sufficient to detect mortality in movement datasets of six African elephants killed by poachers. A month of data prior to, and 24 hours after, the mortality were used for the analysis. The number of false positives triggered by the algorithm is given for each of the critical radius values (e.g., for the animal 'Prunella', the critical radius value needed to detect the true mortality event was 13 meters but the algorithm also generated two false immobility alarms in the prior month when using that radius value).

Elephant	Sex	Critical Radius (Meters)	False Positives (Number)
Chemi Chemi	M	7	0
Kijiji	F	10	0
Marania	M	4	0
Mercury	F	4	0
Prunella	F	13	2
Soboiga	M	3	0

7 Alert Dissemination

Once an algorithm has positively identified an alert condition, the *MovementMonitor* software will query an *Alerts* database (PostgreSQL version 8.4) to determine the users who are subscribed to a particular chronofile/algorithm combination and store details of the alert within the database (Fig. 1 of manuscript). Users can choose to receive alerts via SMS, E-mail or both. Our system currently uses a physical server located in Nairobi, Kenya that has a GSM modem for sending SMS alerts. The *MovementMonitor* software communicates to the Kenya modem using a custom built SMS API we developed using Microsoft Windows Communication Foundation (WCF) technology to disseminate alert SMS messages using custom-built software called '*MessageServer*'. E-mail alerts are also issued by *MovementMonitor* using a C# POP3 class library that sends alerts using the Google *Gmail* E-mail system (Fig. 1 of manuscript). We provide an example alert report generated by the immobility algorithm and received by E-mail in Fig. A2.

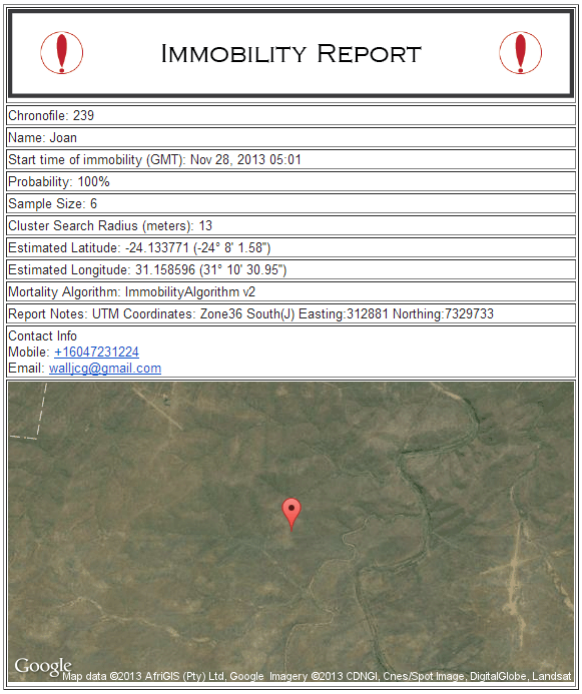

Fig. A2. An example immobility alert received via E-mail. This particular report corresponds to an elephant having dropped its collar which subsequently triggered the alarm.

8 Data Access

User access to tracking and alert data is a key component of the system and can be achieved in several ways. The first is via a Key Hole Markup (KML) tracking data service that is accessed via an HTTP request. The server-side application was built using Microsoft ASP.Net technology hosted within the Microsoft Internet Information Services version 7 (IIS7) framework, and handles incoming HTTP queries from remote clients. KML data can be consumed by a variety of clients, but primarily using the Google Earth geo-browser both for desktop clients and for mobile devices (e.g., Apple iOS iPhones and iPads and Google Android phones and tablets). The KML tracking data API provides a quick way of visualizing data and accessing the latest coordinates and movement of an animal. The API will by default return the most recent 16-days of filtered data. Filtering is accomplished using a cut-off speed and rejecting any GPS points in the animal's movement path that could not be reached using a speed under the threshold value. The API also has the option of turning filtering off to look at all raw locations and for expanding the time window beyond two

weeks. Temporal replay of data is possible in Google Earth using the 'time-slider' functionality; a program feature that was integrated based on Save the Elephants's data (Rebecca Moore: Pers Comm). Similar to the Tracking data API, a second KML Alerts API is available for querying the locations of alerts and displaying their locations and times.

Data is also made available using an API for Esri clients (e.g., ArcMap). The server-side application was built using Microsoft WCF technology and is hosted using IIS7. Desktop clients install the '*STEDownloader*' program which is built using Esri Engine Runtime 10.1 software components, the Microsoft .Net 4.0 class library and the C# programming language. The *STEDownloader* connects to the Esri service using the TCP/IP protocol and updates a locally stored Esri geodatabase with tracking data from the server *AnimalTracking* database. Storage of data within a local geodatabase that resides on the client machine allows for offline access to data as well as retrieval of complete record sets for scientific analysis. A customized database interface within the *STEDownloader* program lets users make advanced selection and filtering operations on stored locations and extract data for further analyses.

Literature cited

Knopff, K. H., A. A. Knopff, M. B. Warren, and M. S. Boyce. 2009. Evaluating Global Positioning System Telemetry Techniques for Estimating Cougar Predation Parameters. Journal of Wildlife Management **73**:586–597.

Legendre, P., and L. Legendre. 1998. Numerical ecology. 2nd edition. Elsevier, Amsterdam ; New York.

WDPA, 2013. World Database on Protected Areas (August, 2013 Edition): Accessed Sept 30, 2013. URL `www.wdpa.org`.