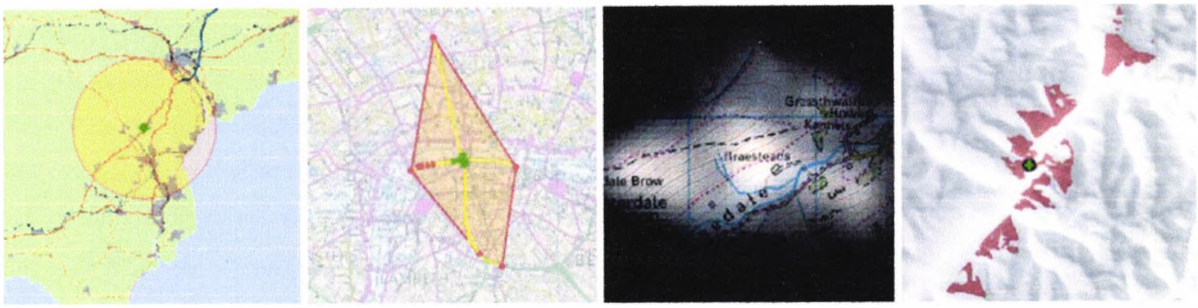

Exploring mobile trajectories:

An investigation of individual spatial behaviour and geographic filters for information retrieval

David Michael Mountain

Appendices

City University, London

Department of Information Science

School of Informatics

December 2005

Appendix 1

User needs evaluation: questions with a spatial reference

The following list of 53 questions are a subset of 90 questions recorded from two user shadowing exercises, conducted as part of the WebPark project in 2001 (Krug et al., 2002).

informaton group	questions	layer	accuracy
fauna	*Are there any marmots here?*	area	5 m
	Where do the animals stay usually?	area	100 m
	Do Ibex occur here?	area	500 m
	Do ibex and chamois occur together?	area	500 m
	Is it possible to see animals in the Ftur valley?	area	500 m
	Is there roe deer up here?	area	500 m
	What tit species do occur here?	area	500 m
	Is there fish in this creek?	line	20 m
	Is this a true ant-hill?	point	5 m
flora	*Was this caused by 'Lothar' or by an avalanche?*	area	20 m
	The little trees may be older as the big ones, isn' t it?	area	20 m
	At which altitude is the timber line? Will we pass it on our way?	area	20 m
	Is this caused by fire?	area	20 m
	Is this grass regularly cut?	area	20 m
	Might this be named an upland moor?	area	20 m
	Are there also berry-bearing bushes?	area	100 m
	Is this a monoculture?	area	100 m
	Why are there only pines here?	area	100 m
	How old is this tree? (three times)	point	5 m
landscape navigation	*Are we at the border of the Swiss National Park now?*	area	20 m
	Are we now leaving the Swiss National Park?	area	20 m
	Is this an artificial lake?	area	100 m
	Is this the lake of Livigno?	area	100 m
	Are we arriving at Alp La Schera?	area	100 m
	Is this a tributary creek of the Inn river?	line	20 m
	What' s the name of this river?	line	20 m
	How will the trail continue?	line	20 m
	Where is the trail leading to Alp La Schera?	line	20 m
	Which mountain is Munt La Schera?	point	20 m
	On what altitude are we now?	point	20 m
	What' s the distance to the turn-off for Munt La Schera?	point	20 m
	Where are we? (twice)	point	20 m
	Are we at the right parking now?	point	100 m
geomorphology	*Is this a moraine? (twice)*	area	20 m
	Is this a frost phenomenon?	area	20 m
	Are these natural hills?	area	100 m
	Is permafrost an important process in this area?	area	100 m
	What kind of rock is this?	area	100 m
	What kind of caves are these? Are they artificial or natural?	point	5 m
	Why is this stone red?	point	5 m
history	*Why was it necessary to cut so many trees here?*	area	100 m
	Did the sumpter-mules also pull carts?	line	5 m
research	*What is this pole for?*	area	5 m
	What is this? (visible installation for research purposes)	area	5 m
SNP	*Is this trail digged out?*	line	5 m
	Who made this and why? (cut tree)	point	5 m
	What is this landmark for?	point	5 m
other	*This doesn't look nature like. What is it?*	point	5 m
	What is this? (mobilephone-antenna)	point	20 m

Appendix 2

Abridged java documentation for the Spatial History Explorer, Dec 2005

SHE: all packages

A full list of all spatial history explorer classes is given below. Only a small number of the most central packages are described in more detail in the following pages. Of the classes that are described in more detail, only summary information is presented for each class. The full java documentation is available on the CD ROM accompanying this thesis, or can be downloaded from http://www.soi.city.ac.uk/~dmm/phd.

edu.cu.gi.dmm.she.activity
edu.cu.gi.dmm.she.attribute
edu.cu.gi.dmm.she.browser
edu.cu.gi.dmm.she.clusters
edu.cu.gi.dmm.she.conversion
edu.cu.gi.dmm.she.envelope
edu.cu.gi.dmm.she.episode
edu.cu.gi.dmm.she.general
edu.cu.gi.dmm.she.geometry
edu.cu.gi.dmm.she.geometry.transformation
edu.cu.gi.dmm.she.grids
edu.cu.gi.dmm.she.image
edu.cu.gi.dmm.she.io
edu.cu.gi.dmm.she.knox
edu.cu.gi.dmm.she.mobileTrajectory
edu.cu.gi.dmm.she.nearestNeighs
edu.cu.gi.dmm.she.pointDensity
edu.cu.gi.dmm.she.prediction
edu.cu.gi.dmm.she.prediction.cwa
edu.cu.gi.dmm.she.screen
edu.cu.gi.dmm.she.screen.backdrop
edu.cu.gi.dmm.she.screen.clusters
edu.cu.gi.dmm.she.screen.color
edu.cu.gi.dmm.she.screen.conversion
edu.cu.gi.dmm.she.screen.frames
edu.cu.gi.dmm.she.screen.grids
edu.cu.gi.dmm.she.screen.interaction
edu.cu.gi.dmm.she.screen.layout
edu.cu.gi.dmm.she.screen.marginalia
edu.cu.gi.dmm.she.screen.mobileTrajectory
edu.cu.gi.dmm.she.screen.panels
edu.cu.gi.dmm.she.screen.polys
edu.cu.gi.dmm.she.screen.prediction

edu.cu.gi.dmm.she.screen.timeGeog
edu.cu.gi.dmm.she.sets
edu.cu.gi.dmm.she.speedModel
edu.cu.gi.dmm.she.srs
edu.cu.gi.dmm.she.stats
edu.cu.gi.dmm.she.subset
edu.cu.gi.dmm.she.timeGeog
edu.cu.gi.dmm.she.util

SHE geometry classes

For the geometry classes, first, more sophisticated point object representations than the java defaults are defined: a point in 3 spatial dimensions, a 3D point with time, and finally PointID (a 3D point with time and a unique identifier). Next a the SpatialHistory class is defined, which stores a mobile trajectory as a list of time-stamped points, with associated motion attributes such as speed, acceleration and heading. Finally exceptions are defined, to handle common errors such as trajectories that consist of only one point, or points in the trajectory that are not in the correct temporal order.

Package edu.cu.gi.dmm.she.geometry

Class Summary	
ModifySpatialHistory	Augments and/or alters the spatial history according to a number of methods.
Point3D	Storage class for a point with three spatial dimensionals Created on 02 Feb 2004
Point3DT	Storage class for a point with three spatial dimensionals and a temporal dimension Created on 02 Feb 2004
PointID	Encapsulates the description of a point 3D location with a timestamp (x,y,z,accuracy,srs,t) Includes a unique (integer) identifier to differentiate this point from others with the same spatial location
SpatialHistory	Stores the data and metadata that comprises a spatial history (a mobile trajectory of points describing an individual's movement through space over time).

Exception Summary	
TemporalTopologyException	Exception when the positions in a SpatialHistory are not in temporal order
TooFewPointsException	Exception when there are two few points to initialise a SpatialHistory

edu.cu.gi.dmm.she.geometry

Class Point3D

```
java.lang.Object
  └java.awt.geom.Point2D
      └java.awt.geom.Point2D.Double
          └edu.cu.gi.dmm.she.geometry.Point3D
```

All Implemented Interfaces:

java.lang.Cloneable

Direct Known Subclasses:

Point3DT

```
public class Point3D
extends java.awt.geom.Point2D.Double
```

Storage class for a point with three spatial dimensions

Version: 1.0
Author: David Mountain

Nested Class Summary

Nested classes/interfaces inherited from class java.awt.geom.Point2D
`java.awt.geom.Point2D.Double, java.awt.geom.Point2D.Float`

Field Summary

`double`	`accuracy` An estimate of accuracy by some measure
`java.lang.String`	`srs` The spatial reference system used
`double`	`z` Stores the z (elevation) value (using some altitude measure) for a 3D point

Fields inherited from class java.awt.geom.Point2D.Double
`x, y`

Constructor Summary

`Point3D(double xIn, double yIn, double zIn, double accuracyIn, java.lang.String srsIn)`
Constructor initializes fields of this class using x, y and z

Method Summary

`boolean`	`equals(Point3D p2)` Compares this Point3D with another Point3D to see if they are the same
`double`	`getAccuracy()` accuracy value
`double`	`getSlope(Point3D p2)`

	Gets the slope between this point and incoming +ve if uphill from incoming to this, -ve if downhill from incoming to this
java.lang.String	getSrs() srs value
double	getZ() z (height) value
double	separation2DMetres(java.awt.geom.Point2D.Double p2) Calculates the two dimensional distance between this Point and incoming (in metres)
double	separation2DNativeCoords(java.awt.geom.Point2D.Double p2) Calculates the two dimensional distance between this position and the incoming (in native coords)
double	separation3DMetres(Point3D p2) Calculates the three dimensional distance between this Point and incoming (in metres) +ve if incoming is above this point, -ve if incoming is below this point
void	setAccuracy(double accuracyIn) set the accuracy value
void	setSrs(java.lang.String srsIn) set the srs value
void	setX(double xIn) set the x value
void	setY(double yIn) set the y value
void	setZ(double zIn) set the z (height) value
java.lang.String	toString() Returns a String that represents the value of this Point3D.

Methods inherited from class java.awt.geom.Point2D.Double
getX, getY, setLocation

Methods inherited from class java.awt.geom.Point2D
clone, distance, distance, distance, distanceSq, distanceSq, distanceSq, equals, hashCode, setLocation

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

edu.cu.gi.dmm.she.geometry

Class Point3DT

```
java.lang.Object
  └java.awt.geom.Point2D
      └java.awt.geom.Point2D.Double
          └edu.cu.gi.dmm.she.geometry.Point3D
              └edu.cu.gi.dmm.she.geometry.Point3DT
```

All Implemented Interfaces:
java.lang.Cloneable

Direct Known Subclasses:
PointID

```
public class Point3DT
extends Point3D
```

Storage class for a point with three spatial dimensionals and a temporal dimension Created on 02 Feb 2004

Version: 1.0
Author: David Mountain

Nested Class Summary

Nested classes/interfaces inherited from class java.awt.geom.Point2D
`java.awt.geom.Point2D.Double, java.awt.geom.Point2D.Float`

Field Summary

`java.util.Date`	`t` Stores the temporal value 3DT point.

Fields inherited from class edu.cu.gi.dmm.she.geometry.Point3D
`accuracy, srs, z`

Fields inherited from class java.awt.geom.Point2D.Double
`x, y`

Constructor Summary

`Point3DT()` Empty java beans constructor
`Point3DT(double xIn, double yIn, double zIn, double accuracyIn, java.lang.String srsIn, java.util.Date tIn)` Constructor initialises fields of this class
`Point3DT(Point3DT pIn)` Constructor initialises fields of this class using another point

Method Summary

`boolean`	`equals(Point3DT p2)`

	Compares this Point3DT with another Point3DT to see if they are the same
java.util.Date	getDate() Gets the timestamp for this point
double	getSpeed(Point3DT p2) Gets the speed between this point3DT and the incoming
void	setDate(java.util.Date tIn) sets the timestamp for this point
java.lang.String	toString() Returns a String that represents the value of this Point3DT.

Methods inherited from class edu.cu.gi.dmm.she.geometry.Point3D
equals, getAccuracy, getSlope, getSrs, getZ, separation2DMetres, separation2DNativeCoords, separation3DMetres, setAccuracy, setSrs, setX, setY, setZ

Methods inherited from class java.awt.geom.Point2D.Double
getX, getY, setLocation

Methods inherited from class java.awt.geom.Point2D
clone, distance, distance, distance, distanceSq, distanceSq, distanceSq, equals, hashCode, setLocation

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

edu.cu.gi.dmm.she.geometry

Class PointID

```
java.lang.Object
  └java.awt.geom.Point2D
      └java.awt.geom.Point2D.Double
          └edu.cu.gi.dmm.she.geometry.Point3D
              └edu.cu.gi.dmm.she.geometry.Point3DT
                  └edu.cu.gi.dmm.she.geometry.PointID
```

All Implemented Interfaces:

java.io.Serializable, java.lang.Cloneable

```
public class PointID
extends Point3DT
implements java.io.Serializable
```

Encapsulates the description of a point 3D location with a timestamp (x,y,z,accuracy,srs,t) Includes a unique (integer) identifier to differentiate this point from others with the same spatial location

Version: 0.3 09 Feb 2004
Author: David Mountain, City Uni London, dmm@soi.city.ac.uk
See Also:
Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class java.awt.geom.Point2D
java.awt.geom.Point2D.Double, java.awt.geom.Point2D.Float

Field Summary

int	id Unique integer identifier to differentiate this point from others with the same spatial location.

Fields inherited from class edu.cu.gi.dmm.she.geometry.Point3DT
t

Fields inherited from class edu.cu.gi.dmm.she.geometry.Point3D
accuracy, srs, z

Fields inherited from class java.awt.geom.Point2D.Double
x, y

Constructor Summary

PointID(int idParam, double xIn, double yIn, double zIn, double accIn, java.lang.String srsIn, java.util.Date tIn) Creates a new point

`PointID(int idParam, Point3DT pointParam)` Creates a new point storing the identifier and latitude/longitude location

Method Summary

`java.lang.String`	`toString()` This method returns the point as a string

Methods inherited from class edu.cu.gi.dmm.she.geometry.Point3DT
`equals, getDate, getSpeed, setDate`

Methods inherited from class edu.cu.gi.dmm.she.geometry.Point3D
`equals, getAccuracy, getSlope, getSrs, getZ, separation2DMetres, separation2DNativeCoords, separation3DMetres, setAccuracy, setSrs, setX, setY, setZ`

Methods inherited from class java.awt.geom.Point2D.Double
`getX, getY, setLocation`

Methods inherited from class java.awt.geom.Point2D
`clone, distance, distance, distance, distanceSq, distanceSq, distanceSq, equals, hashCode, setLocation`

Methods inherited from class java.lang.Object
`getClass, notify, notifyAll, wait, wait, wait`

edu.cu.gi.dmm.she.geometry

Class SpatialHistory

```
java.lang.Object
  └edu.cu.gi.dmm.she.geometry.SpatialHistory
```

```
public class SpatialHistory
extends java.lang.Object
```

Stores the data and metadata that comprises a spatial history (a mobile trajectory of points describing an individual's movement through space over time).

Version:
3.6, 10 Feb 2004

Author:
David Mountain; City Uni, London, dmm@soi.city.ac.uk

Field Summary	
LinearAttribute	**acceleration** LinearAttribute object storing all of the acceleration values corresponding to every position in this spatial history and associated metadata (stored as metres per second per second)
LinearAttribute	**accuracy** LinearAttribute object storing all of the accuracy values (in some units, eg hdop, units, metres) corresponding to every position in this spatial history and associated metadata
LinearAttribute	**altitude** LinearAttribute object storing all of the altitude values corresponding to every position in this spatial history and associated metadata
CircularAttribute	**annualDistn** CircularAttribute object storing the annual distribution as mSecs midnight on 1 Jan.
LinearAttribute[]	**attributes** An array of attribute objects
CircularAttribute	**dailyDistn** CircularAttribute object storing the time of day as msecs since midnight.
LinearAttribute	**distCent** LinearAttribute object storing all of the distance to centroid values corresponding to every position in this spatial history and associated metadata
CircularAttribute	**heading** CircularAttribute object storing all of the heading values corresponding to every position in this spatial history and associated metadata
LinearAttribute	**slope** LinearAttribute object storing all of the slope (from -90 to +90) values corresponding to every position in this spatial history and associated metadata .
LinearAttribute	**speed** LinearAttribute object storing all of the speed values corresponding to every position in this spatial history and

	associated metadata (stored as metres per second)
java.lang.String	_srs store somewhere outside Stores the spatial reference system
LinearAttribute	_time Attribute object storing all of the milliseconds (since 1 Jan 1970) values corresponding to every position in this spatial history and associated metadata
CircularAttribute	_turningAngle CircularAttribute object storing all of the turning values corresponding to every position in this spatial history and associated metadata
double	_walkingSpeedOnFlat LinearAttribute object storing walking speed on flat ground.
CircularAttribute	_weeklyDistn CircularAttribute object storing the weekly distribution as mSecs midnight on Sun/Mon.
LinearAttribute	_x LinearAttribute object storing all of the x values (in some numerical Cartesian coord scheme) corresponding to every position in this spatial history and associated metadata
LinearAttribute	_y LinearAttribute object storing all of the y values (in some numerical Cartesian coord scheme corresponding to every position in this spatial history and associated metadata
Point3DT	firstPoint Oldest position
Point3DT	lastPoint Most recent position
long	meanSamplingRate Stores the mean time interval (in milliseconds) between the collection of temporally adjacent points.
int	nPoints Total number of input points
java.util.List	pointIdList List of points as PointIDs
double	sinuosity Stores sinuosity for the entire set of points.
java.util.Date[]	timestamps Array of time stamps corresponding to the object list.
java.util.Hashtable	trajectoryObjects Hashtable storing all of the other screen objects associated with a single trajectory
java.lang.String	userName Variable holding user information

Constructor Summary

`SpatialHistory(java.lang.String srsIn, java.util.List phList)`
First constructor used in offline mode (when username not known)

SpatialHistory(java.lang.String uname, java.lang.String srsIn, net.eads.sysde.portal.userlocator.IPositionHistory iPosHist) Constructor used when converting for the WebPark portal

Method Summary

double	dateToRatio(java.util.Date dateIn) Method to convert an absolute date/time to a ratio wrt this spatial history (if the date is within the spatial history temporal bounds, the ratio will be between 0 and 1) This method is used for spatializing the temporal values, in this case calculating the x axis of the temporal plot
double	dayOfWeekToRatio(java.util.Date dateIn) Method to convert a day of the week to a ratio (between 0 and 1) This method is used for spatializing the temporal values, in this case calculating day of the week for the y axis of the temporal plot
double	dayOfWeekToRatio(int dayOfWeek) Method to convert a day of the week to a ratio (between 0 and 1) This method is used for spatializing the temporal values, in this case calculating day of the week for the y axis of the temporal plot
long	getDuration() The duration of this history in milliseconds
Java.util.Date	getMeanTime() The mean time for the set
Point3DT	latestBreakpoint() Finds the latest breakpoint based on attribute (speed and direction)
Java.util.List	latestSeries(long millisBreak) Returns the most recent series of points (that are an unbroken temporal sequence)
double	monthToRatio(java.util.Date dateIn) Method to convert a month to a ratio (between 0 and 1) This method is used for spatializing the temporal values, in this case calculating the month for the y axis of the temporal plot
double	monthToRatio(int month) Method to convert a month to a ratio (between 0 and 1) This method is used for spatializing the temporal values, in this case calculating the month for the y axis of the temporal plot
Java.util.Date	ratioToDate(double ratio) Method to convert a ratio into an absolute date/time wrt this spatial history (if the ratio is between 0 and 1, the date will be within the spatial history temporal bounds,) This method is used for reverse spatializing the temporal values, in this case calculating the time associated with a location on the temporal plot
int	ratioToDayOfWeek(double ratio) Method to convert a ratio (between 0 and 1) to a day of the week This method is used for spatializing the temporal values, in this case calculating day of the week for the y axis of the temporal plot
int	ratioToMonth(double ratio) Method to convert a ratio (between 0 and 1) to a month (between 0-11) This method is used for spatializing the temporal values, in this case calculating the month for the y axis of the temporal plot

java.sql.Time	ratioToTime(double ratio) Method to convert a ratio (between 0 and 1) to a time of day This method is used for spatializing the temporal values, in this case calculating time of day for the y axis of the temporal plot
Java.util.List	removeOutliers(float centroidSepTol, float millisTol, float speedTol, float accelTol) Returns a subset of the original positionHistory with outliers removed
double	timeToRatio(java.util.Date dateIn) Method to convert time of day to a ratio (between 0 and 1) This method is used for spatializing the temporal values, in this case calculating time of day for the y axis of the temporal plot
java.lang.String	toShortString() A string rep of this class
java.lang.String	toString() A string rep of this class

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

edu.cu.gi.dmm.she.geometry

Class ModifySpatialHistory

```
java.lang.Object
  └edu.cu.gi.dmm.she.geometry.ModifySpatialHistory
```

```
public class ModifySpatialHistory
extends java.lang.Object
```

Augments and/or alters the spatial history by applying a number of methods.

Version: 0.6, 13 Apr 2004
Author: David Mountain; City Uni, London; dmm@soi.city.ac.uk

Constructor Summary

`ModifySpatialHistory()`

Method Summary

`static SpatialHistory`	`degradeResolution(SpatialHistory spatialHist, double spatialRes, double temporalRes)` Degrades the spatial and temporal resolution
`static SpatialHistory`	`interpolateTemporalGaps(SpatialHistory spatialHist, long minTemporalGap, long maxTemporalGap, double maxSpatialDistance, long temporalResolution)` Interpolates temporal gaps in the spatial history according to the incoming parameters If the specified temporal and spatial limits are not exceeded then points are inserted at the last known location at the specified temporal resolution

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

edu.cu.gi.dmm.she.geometry

Class TemporalTopologyException

```
java.lang.Object
  └─java.lang.Throwable
      └─java.lang.Exception
          └─edu.cu.gi.dmm.she.geometry.TemporalTopologyException
```

All Implemented Interfaces:

java.io.Serializable

```
public class TemporalTopologyException
extends java.lang.Exception
```

Exception when the positions in a SpatialHistory are not in temporal order

Version: 0.1, 21 Jan 2004
Author: David Mountain; City Uni, London, dmm@soi.city.ac.uk
See Also:

Serialized Form

Constructor Summary

`TemporalTopologyException()`

Constructs a new exception with `null` as its detail message.

Method Summary

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

edu.cu.gi.dmm.she.geometry

Class TooFewPointsException

```
java.lang.Object
  └─java.lang.Throwable
      └─java.lang.Exception
          └─edu.cu.gi.dmm.she.geometry.TooFewPointsException
```

All Implemented Interfaces:

java.io.Serializable

```
public class TooFewPointsException
extends java.lang.Exception
```

Exception when there are too few points to initialise a SpatialHistory

Version: 0.1, 21 Jan 2004
Author: David Mountain; City Uni, London, dmm@soi.city.ac.uk
See Also:
Serialized Form

Constructor Summary

`TooFewPointsException()`
Constructs a new exception with `null` as its detail message.

Method Summary

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

SHE attribute classes

The SHE attribute classes can be used to calculate, store and analyse the attributes associated a mobile trajectory. These are predominantly motion attributes such as speed, heading and acceleration. The storage classes (Attribute, LinearAttribute and CircularAttribute) are described in more detail. Classes relating to the other attribute classes relate to calculating these attributes and parameters such as quartiles and have not been described in further detail.

Package edu.cu.gi.dmm.she.attribute

Class Summary	
Attribute	Stores the values for an attribute (as a double array) and associated metadata Only the values and number of values are stored.
CircularAttribute	Stores the values for a 'circular' attribute (as a double array) and associated metadata Should be used for circular attributes (as opposed to linear attributes), where values are measured on an cyclical scale from 0-360 degrees. See Brunsdon 2003, GISRUK proceedings and Cox 2003 for more on the calculation of circular statistics
CreateAttribute	Generates values associated with the attribute classes
DecayFunctions	records different decay functions that can be used when calculating values such as weighted means
LinearAttribute	Stores the values for a 'linear' attribute (as a double array) and associated metadata Should be used for linear attributes (as opposed to circular attributes), where values are measured on an interval/ratio scale and an unambiguous minimum, maximum, mean, standard deviation and quartiles can be identified
Quartiles	Methods for calculating the quartiles associated with an array of values
WeightingParameters	Records different parameters b which points can be weighted, eg distance weighted, time weighted

edu.cu.gi.dmm.she.attribute

Class Attribute

```
java.lang.Object
  └edu.cu.gi.dmm.she.attribute.Attribute
```

Direct Known Subclasses:

CircularAttribute, LinearAttribute

```
public class Attribute
extends java.lang.Object
```

Stores the values for an attribute (as a double array) and associated metadata. Only the values and number of values are stored. For more statistical values, use a subclass LinearAttribute or CircularAttribute TODO create different versions of variable for nominal, ordinal, interval and ratio

Version: 0.4, 20 Oct 2003
Author: David Mountain, City Uni London, dmm@soi.city.ac.uk
See Also:

`edu.cu.gi.dmm.she.attribute.LinearAttribute`, `edu.cu.gi.dmm.she.attribute.CircularAttribute`

Field Summary

`static double`	`NULL_ATTRIBUTE_VALUE` The null value for any attribute

Constructor Summary

`Attribute(double[] values)` Constructor; sets class variables to create an instantiation of this class

Method Summary

`int`	`getTotNoValues()` Returns the number of values for this attribute
`int`	`getTotValidValues()` Returns the number of values for this attribute
`double`	`getValue(int i)` Returns the value of the 'i'th attribute
`double[]`	`getValues()` Returns the values of this attribute

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

edu.cu.gi.dmm.she.attribute

Class LinearAttribute

```
java.lang.Object
  └ edu.cu.gi.dmm.she.attribute.Attribute
      └ edu.cu.gi.dmm.she.attribute.LinearAttribute
```

---

public class **LinearAttribute**
extends Attribute

Stores the values for a 'linear' attribute (as a double array) and associated metadata Should be used for linear attributes (as opposed to circular attributes), where values are measured on an interval/ratio scale and an unambiguous minimum, maximum, mean, standard deviation and quartiles can be identified

Version: 0.2, 11 May 2005
Author: David Mountain, City Uni London, dmm@soi.city.ac.uk

Field Summary

Fields inherited from class edu.cu.gi.dmm.she.attribute.Attribute
NULL_ATTRIBUTE_VALUE

Constructor Summary

LinearAttribute(double[] values) Constructor called when only the values (as an array of doubles) associated with this attribute are known
LinearAttribute(double[] values, double mean, double min, double max, double range, double sDev, double[] quartiles, double iqRange) Constructor called when setting all values manually * @param values Array storing the values for this attribute
LinearAttribute(int nValues, double[] values, double min, double max, double mean, double range, double sDev, double[] quartiles, double iqRange) Constructor called when all the parameters associated with this attribute are known
LinearAttribute(LinearAttribute linAtt) Constructor called when initialising class using another attribute object

Method Summary

double	**getIqRange**() Returns the interquartile range (q3-q1) for this attribute
double	**getMax**() Returns the max value of this attribute
double	**getMean**() Returns the mean value of this attribute
double	**getMin**() Returns the min value of this attribute

double[]	getQuartiles() Returns the quartiles (q0 min, q1, q2 median, q3, q4 max) for this attribute
double	getRange() Returns the range in getValues() of this attribute
double	getSDev() Returns the standard deviation in getValues() of this attribute
void	setSDev(double sDev) Sets the standard deviation of this attribute
java.lang.String	toString() Returns a string representation of this class

Methods inherited from class edu.cu.gi.dmm.she.attribute.Attribute
getTotNoValues, getTotValidValues, getValue, getValues

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

edu.cu.gi.dmm.she.attribute

Class CircularAttribute

```
java.lang.Object
  └ edu.cu.gi.dmm.she.attribute.Attribute
      └ edu.cu.gi.dmm.she.attribute.CircularAttribute
```

```
public class CircularAttribute
extends Attribute
```

Stores the values for a 'circular' attribute (as a double array) and associated metadata Should be used for circular attributes (as opposed to linear attributes), where values are measured on an cyclical scale from (for example) 0-360 degrees. See Brunsdon 2003, GISRUK proceedings and Cox 2003 for more on the calculation of circular statistics

Version: 0.1, 22 Oct 2003
Author: David Mountain, City Uni London, dmm@soi.city.ac.uk

Field Summary

Fields inherited from class edu.cu.gi.dmm.she.attribute.Attribute
`NULL_ATTRIBUTE_VALUE`

Constructor Summary

`CircularAttribute(CircularAttribute circAtt)` Constructor called when initialising class using another attribute object
`CircularAttribute(double[] values, double lowerLimit, double upperLimit)` Constructor called when only the values (as an array of doubles) associated with this attribute are known
`CircularAttribute(double[] values, double mean, double spread, double lowerLimit, double upperLimit)` Constructor called when all the parameters associated with this attribute are known

Method Summary

`double`	`getLowerLimit()`
`double`	`getMean()` Returns the mean value of this attribute
`double`	`getSpread()` Returns the spread value of this circular attribute (a measure of distribution)
`double`	`getUpperLimit()`
`void`	`setSpread(double spread)` Sets the spread value of this attribute
`java.lang.String`	`toString()`

	Returns a string representation of this class

Methods inherited from class edu.cu.gi.dmm.she.attribute.Attribute
getTotNoValues, getTotValidValues, getValue, getValues

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Package edu.cu.gi.dmm.she.grids

The grids package is used for field (raster) based representation. Among other classes in this package is the PredictionGrid class, which is used in for the storage of speed-heading prediction surfaces. The Grid and DoubleGrid classes, which the PredictionGrid class extends, are both also shown. Various other grid classes, including GridFactory which is used to created these raster surfaces, are not shown but can be made available upon request.

Class Summary	
CircularAttributeGrid	Storage class for a 2 dimensional grid coverage that extends Grid and stores a grid of CircularAttributes.
DoubleGrid	Storage class for a 2 dimensional grid coverage that extends Grid and stores a grid of values to double precision
Grid	Stores the spatial dimensions (aka domain) of a (2D) grid coverage; objects of this grid store no cell values but contain the methods for translating between column/row values to the native referencing system.
GridFactory	Class for creating different types of surfaces such as point density surfaces, speed surfaces etc...
LinearAttributeGrid	Storage class for a 2 dimensional grid coverage that extends Grid and stores a grid of LinearAttributes.
PredictionGrid	Stores prediction grid of all locations a moving object is likely to be given an origin and period of time into the future.

edu.cu.gi.dmm.she.grids

Class Grid

```
java.lang.Object
  └edu.cu.gi.dmm.she.grids.Grid
```

Direct Known Subclasses:

CircularAttributeGrid, DoubleGrid, LinearAttributeGrid

```
public class Grid
extends java.lang.Object
```

Stores the spatial dimensions (aka domain) of a (2D) grid coverage; objects of this grid store no cell values but contain the methods for translating between column/row values to the native referencing system. It must be aligned according to some unrectified, 2 dimensional, numeric, linear coordinate schema (eg WGS84, decimal degrees, CH1903+).

Implementing classes store a grid of values (aka range set) where values can be numeric (such as double or int) or other objects (such as color).

The grid below shows the row column indexing for a 5*5 grid ([column,row] or [x,y]).

```
 52.6   [0,4] [1,4] [2,4] [3,4] [4,4]
 52.5   [0,3] [1,3] [2,3] [3,3] [4,3]
 52.4   [0,2] [1,2] [2,2] [3,2] [4,2]
 22.3   [0,1] [1,1] [2,1] [3,1] [4,1]
 52.2   [0,0] [1,0] [2,0] [3,0] [4,0]
         0.6   0.7   0.8   0.9   1.0
```

It follows the Euclidean 2D map coordinates format as opposed to the table/spreadsheet convention.

Version: 5.2, 16 June 2004
Author: David Mountain; City Uni, London; dmm@soi.city.ac.uk
See Also:

GridCreation

Field Summary	
static double	OUT_OF_BOUNDS Indicates that a row/column combination, or an x/y ref, fall outside the grid bounds

Constructor Summary
Grid(com.vividsolutions.jts.geom.Envelope boundsIn, int nColsIn, int nRowsIn, java.lang.String srs) Creates new Grid by setting class variables
Grid(Grid gridIn) Creates new Grid from an existing grid

Method Summary	
com.vividsolutions.jts.geom.Envelope	getBounds()

double	getCellRes()
int	getCol(double xIn) Converts an x axis into a column value; returns the column that this x axis value falls into).
int	getNCols()
int	getNRows()
int	getRow(double yIn)
java.lang.String	getSrs()
double	getX(int colIn) Converts the column value (colIn) into an x axis value such (returns the value for the centre of the column).
double	getY(int rowIn) Converts the row value (rowIn) into a y axis value (returns the value for the centre of the row).
boolean	outOfBounds(int colIn, int rowIn) True of row/col number is out of grid bounds
java.lang.String	toEsriHeaderString() Returns a string representation of this grid's spatial metadata in ESRI Grid format

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

edu.cu.gi.dmm.she.grids

Class DoubleGrid

```
java.lang.Object
  └edu.cu.gi.dmm.she.grids.Grid
      └edu.cu.gi.dmm.she.grids.DoubleGrid
```

Direct Known Subclasses:
PredictionGrid

```
public class DoubleGrid
extends Grid
```

Storage class for a 2 dimensional grid coverage that extends Grid and stores a grid of values to double precision

Version: 2.1, 17 Nov 2003
Author: David Mountain; City Uni, London; dmm@soi.city.ac.uk

Field Summary

LinearAttribute	**attribute** Stores the actual grid values of this grid coverage as an attribute
double[][]	**cellValues** Stores the actual grid values of this grid coverage as doubles in a 2d array

Fields inherited from class edu.cu.gi.dmm.she.grids.Grid
OUT_OF_BOUNDS

Constructor Summary

DoubleGrid(Grid gridDimensionsIn, double[][] cellValuesIn) Creates new DoubleGrid by setting class variables

Method Summary

double	**getCellValue**(int colIn, int rowIn) Returns the attribute value of the specified cell (OUT_OF_BOUNDS, -999, if the cell/column combination exceeds the grid dimensions) Do not confuse this method with getCellValue(double x, double y), which uses real world coordinates as input parameters
double	**getCellValue**(java.awt.geom.Point2D.Double xyIn) Returns the attribute value of the cell that corresponds to the specified x,y location (OUT_OF_BOUNDS, -999, if the cell/column combination exceeds the grid dimensions)
Grid	**getGrid**() Returns the superclass for this DoubleGrid
java.lang.String	**toEsriString**() Returns an ESRI Grid string representation of this clas

Methods inherited from class edu.cu.gi.dmm.she.grids.Grid

getBounds, getCellRes, getCol, getNCols, getNRows, getRow, getSrs, getX, getY, outOfBounds, toEsriHeaderString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

edu.cu.gi.dmm.she.grids

Class PredictionGrid

```
java.lang.Object
  └edu.cu.gi.dmm.she.grids.Grid
      └edu.cu.gi.dmm.she.grids.DoubleGrid
          └edu.cu.gi.dmm.she.grids.PredictionGrid
```

All Implemented Interfaces:

Prediction

Direct Known Subclasses:

AccessibilityGrid, CrwaPredictionGrid

```
public class PredictionGrid
extends DoubleGrid
implements Prediction
```

Stores prediction grid of all locations a moving object is likely to be given an origin and period of time into the future. This class implements prediction, so may predict for an instant of time in the future, or over an interval of time.

Version: v2.5, 22 June 2004
Author: David Mountain; City Uni, London; dmm@soi.city.ac.uk
See Also: `edu.cu.gi.dmm.she.cwa.CwaPredictionGrid`, `edu.cu.gi.dmm.she.timeGeog.AccessibilityGrid`

Field Summary

Fields inherited from class edu.cu.gi.dmm.she.grids.DoubleGrid
`attribute`, `cellValues`

Fields inherited from class edu.cu.gi.dmm.she.grids.Grid
`OUT_OF_BOUNDS`

Constructor Summary

`PredictionGrid(Grid gridDimensionsIn, double[][] cellValuesIn, java.awt.geom.Point2D.Double startLoc, java.util.Date startTime, long predictionLength, boolean destinationKnown, java.awt.geom.Point2D.Double actualDestination)` Creates new PredictionGrid by setting class variables.
`PredictionGrid(Grid gridDimensionsIn, double[][] cellValuesIn, java.awt.geom.Point2D.Double startLoc, java.util.Date startTime, long predictionLength, boolean destinationKnown, java.awt.geom.Point2D.Double[] actualDestinations)` Creates new PredictionGrid by setting class variables.
`PredictionGrid(Grid gridDimensionsIn, double[][] cellValuesIn, java.awt.geom.Point2D.Double startLoc, java.util.Date startTime, long predictionLength, int instantInterval)` Creates new PredictionGrid by setting class variables.

Method Summary

Java.awt.geom.Point2D.Double	getActualDestination() Returns the verification point, at an instant in time (if known, else null)
java.awt.geom.Point2D.Double[]	getActualDestinations() Returns the verification points, for an interval of time (if known, else null)
boolean	getDestinationKnown() whether the actual destination is known for this prediction
int	getInstantInterval() Whether the prediction represents where the moving point object is likely to be at an instant in the future (eg exactly 15 minutes from now) or from now until that point in the future, over a temporal interval (from 0-15 minutes from now)
long	getPredictionDuration() The period of time that defines the duration for any prediction
java.awt.geom.Point2D.Double	getStartLocation() The spatial location defining the "starting point" for any prediction; it need not be contained within the grid
java.util.Date	getStartTime() The time at the start of the prediction.
double	getSuccessRate() returns the success rate for this grid (n verification points inside of the grid)
double	getVerificationValue() Returns the verification value.

Methods inherited from class edu.cu.gi.dmm.she.grids.DoubleGrid
getCellValue, getCellValue, getGrid, toEsriString

Methods inherited from class edu.cu.gi.dmm.she.grids.Grid
getBounds, getCellRes, getCol, getNCols, getNRows, getRow, getSrs, getX, getY, outOfBounds, toEsriHeaderString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Package edu.cu.gi.dmm.she.prediction

The prediction package has interfaces for predictions, and classes to store predictions based upon potential path areas (PpaPrediction) and spatial proximity (SpatialBufferPrediction). These type of predictions are described and evaluated extensively in the thesis.

Interface Summary	
Prediction	Interface for any form of prediction This can include point predictions or predictions grids.

Class Summary	
InstantInterval	Keys that record whether a prediction is predicting the future situation at an instant in time, or over an interval of time, in the future.
PpaPrediction	Makes a prediction about future behaviour based upon long-term previous experience
SpatialBufferPrediction	Stores a spatial proximity prediction

edu.cu.gi.dmm.she.prediction

Interface Prediction

All Known Implementing Classes:

AccessibilityGrid, CrwaPrediction, CrwaPredictionGrid, CrwaRandomMultiStepPrediction, CrwaRandomPrediction, PpaPrediction, PredictionGrid, SpatialBufferPrediction

---

`public interface **Prediction**`

Interface for any form of prediction This can include point predictions or predictions grids.

version 0.1, 6 May 2004

author David Mountain, City Uni London, dmm@soi.city.ac.uk

Method Summary

`java.awt.geom.Point2D.Double`	`getActualDestination()` Returns the actual destination (if known, else null)
`java.awt.geom.Point2D.Double[]`	`getActualDestinations()` Returns the actual destination (if known, else null)
`boolean`	`getDestinationKnown()` Returns whether the actual destination is known for this prediction
`int`	`getInstantInterval()` Whether the prediction surface represents where the subject is likely to be at an instant in the future (eg exactly 15 minutes from now - false) or from now until that point in the future, over a temporal interval (from 0-15 minutes from now - true)
`long`	`getPredictionDuration()` Returns the period of time (in millisecs), from _startTime for which the prediction is made, the duration of the prediction
`java.awt.geom.Point2D.Double`	`getStartLocation()` Returns the starting location for the prediction in native coords
`java.util.Date`	`getStartTime()` Returns the starting time for the prediction

edu.cu.gi.dmm.she.prediction
Class PpaPrediction

```
java.lang.Object
  └ edu.cu.gi.dmm.she.timeGeog.PotentialPathArea
      └ edu.cu.gi.dmm.she.prediction.PpaPrediction
```

All Implemented Interfaces:
Prediction

```
public class PpaPrediction
extends PotentialPathArea
implements Prediction
```

Makes a prediction about future behaviour based upon long-term previous experience

Version: V0.1, 06 June 2005
Author: David Mountain; City Uni, London; dmm@soi.city.ac.uk

Field Summary

Fields inherited from class edu.cu.gi.dmm.she.timeGeog.PotentialPathArea
timeBudget, boundaryType, hullPoints, paths, stPathBuffers, targetPoint

Constructor Summary

`PpaPrediction(PotentialPathArea ppa, java.util.Date originTime, long predDurationMSecs)` Creates new PpaPrediction
`PpaPrediction(PotentialPathArea ppa, java.util.Date originTime, java.awt.geom.Point2D.Double actualDestination, long predDurationMSecs)` Creates new PpaPrediction

Method Summary

`java.awt.geom.Point2D.Double`	`getActualDestination()` The actual destination point (eg future position) if known, else null
`boolean`	`getActualDestinationInsideHull()` tf is the prediction inside the hull (false if not known)
`java.awt.geom.Point2D.Double[]`	`getActualDestinations()` returns null, since ppas only predict for an instant in the future at the moment
`boolean`	`getDestinationKnown()` whether the destination point is known or not (must be true for this type)
`int`	`getInstantInterval()` Whether the prediction represents where the moving point object is likely to be at an

	instant in the future (eg exactly 15 minutes from now) or from now until that point in the future, over a temporal interval (from 0-15 minutes from now) @see InstantInterval
long	getPredictionDuration() Returns the point in time (in millisecs) ahead of 'now' (ie in the future) for which the prediction should be made
double	getPredictionEffectiveness() A measure of how effective this prediction is, based upon whether the point is in or out of the radius (1 or 0), and the root of surface area
java.awt.geom.Point2D.Double	getStartLocation() Returns the point of origin (eg present position) in lat/lon
java.util.Date	getStartTime() Returns the time at the start of the prediction
long	getTimeBudget() Returns the time budget, the period of time used to follow each space time path, to define the accessible region

Methods inherited from class edu.cu.gi.dmm.she.timeGeog.PotentialPathArea
getSurfaceArea, toEsriGenerateString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

edu.cu.gi.dmm.she.prediction

Class SpatialBufferPrediction

```
java.lang.Object
  └─edu.cu.gi.dmm.she.prediction.SpatialBufferPrediction
```

All Implemented Interfaces:

Prediction

```
public class SpatialBufferPrediction
extends java.lang.Object
implements Prediction
```

Stores a spatial proximity prediction

Version: 0.1, 25 May 2005
Author: David Mountain, City Uni London, dmm@soi.city.ac.uk

Constructor Summary

`SpatialBufferPrediction(java.awt.geom.Point2D.Double origin, java.util.Date originTime, long predMSecs, double bufferRadius)`   Constructor; sets the class variables for an incoming data set.
`SpatialBufferPrediction(java.awt.geom.Point2D.Double origin, java.util.Date originTime, long predMSecs, double bufferRadius, java.awt.geom.Point2D.Double actualDestination)`   Constructor; sets the class variables for an incoming data set.

Method Summary

`java.awt.geom.Point2D.Double`	`getActualDestination()` The actual destination point (eg future position) if known, else null
`boolean`	`getActualDestinationInsideBuffer()` tf is the prediction inside the buffer (false if not known)
`java.awt.geom.Point2D.Double[]`	`getActualDestinations()` returns null, since ppas only predict for an instant in the future at the moment
`double`	`getBufferRadius()` The radius of the buffer around the spatial origin
`boolean`	`getDestinationKnown()` whether the destination point is known or not (must be true for this type)
`int`	`getInstantInterval()` Whether the prediction represents where the moving point object is likely to be at an instant in the future (eg exactly 15 minutes from now) or from now until that point in the future, over a temporal interval (from 0-15 minutes from now) @see InstantInterval
`long`	`getPredictionDuration()` Returns the point in time (in millisecs) ahead of 'now' (ie in the future) for which the

	prediction should be made
double	getPredictionEffectiveness() A measure of how effective this prediction is, based upon whether the point is in or out of the radius (1 or 0), and the root of surface area
java.awt.geom.Point2D.Double	getStartLocation() Returns the point of origin (eg present position) in lat/lon
java.util.Date	getStartTime() Returns the time at the start of the prediction

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Package edu.cu.gi.dmm.she.prediction.cwa

These classes are used to store speed-heading predictions.

Class Summary	
CrwaPrediction	Takes an origin, speed, time and heading and calculates a destination It is intended to predict a future position given a present position (origin), speed and heading calculated some way
CrwaPredictionGrid	Stores prediction grid of all locations a moving object is likely to be given an origin and period of time into the future.
CrwaRandomMultiStepPrediction	Extends CwaRandomPrediction to be a series if short steps, rather than a single straight step.
CrwaRandomPrediction	Extends CwaPrediction to add random element to the prediction of future locations, based on the the distribution of the speed and heading attributes.

edu.cu.gi.dmm.she.prediction.cwa
Class CrwaPrediction

```
java.lang.Object
  └ edu.cu.gi.dmm.she.prediction.cwa.CrwaPrediction
```

All Implemented Interfaces:
Prediction

Direct Known Subclasses:
CrwaRandomPrediction

```
public class CrwaPrediction
extends java.lang.Object
implements Prediction
```

Takes an origin, speed, time and heading and calculates a destination It is intended to predict a future position given a present position (origin), speed and heading calculated some way

Version: 0.3, 12 May 2005
Author: David Mountain, City Uni London, dmm@soi.city.ac.uk

Field Summary

boolean	determistic Records whether this prediction is deterministic or probabilistic random
boolean	multistep Records whether this prediction is a single or multistep
SpatialHistory	recentBehaviour The set of 'recent' points used to calculate the mean speed and heading, if the first constructor is used

Constructor Summary

CrwaPrediction(java.awt.geom.Point2D.Double origin, java.util.Date originTime, long predMSecs, SpatialHistory recentBehaviour) Constructor; sets the class variables for an incoming data set.

Method Summary

void	calcPredictedDestination() Sets the predicted destination point (eg future position) in lat/lon, values.
java.awt.geom.Point2D.Double	getActualDestination() Returns the actual destination (if known, else null)
java.awt.geom.Point2D.Double[]	getActualDestinations() Returns the actual destination (if known, else null)
boolean	getDestinationKnown() Returns whether the actual destination is known for this prediction
double	getHeadingMean() Returns the heading calculated some way (eg the circular mean ove 360 degrees

double	getHeadingSpread()
int	getInstantInterval() returns whether this prediction is predicting for an instant in the future
java.awt.geom.Point2D.Double	getPredictedDestination() The predicted destination point
long	getPredictionDuration() Returns the point in time (in millisecs) ahead of 'now' (ie in t should be made
SpatialHistory	getRecentHistory() Returns the set of 'recent' points used to calculate the mean : constructor is used.
double	getSpeedMean() Returns the speed used for the prediction
double	getSpeedSDev()
java.awt.geom.Point2D.Double	getStartLocation() Returns the point of origin (eg present position) in lat/lon
java.util.Date	getStartTime() Returns the time at the start of the predcition
void	setHeading(double headingMean, double headingSpr Sets the heading used for the prediction
void	setPredictedDestination(java.awt.geom.Point2D.Do Sets the predicted destination point
void	setSpeed(double speedMean, double speedSDev) Sets the speed attribute object

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

edu.cu.gi.dmm.she.prediction.cwa

Class CrwaPredictionGrid

```
java.lang.Object
  └ edu.cu.gi.dmm.she.grids.Grid
      └ edu.cu.gi.dmm.she.grids.DoubleGrid
          └ edu.cu.gi.dmm.she.grids.PredictionGrid
              └ edu.cu.gi.dmm.she.prediction.cwa.CrwaPredictionGrid
```

All Implemented Interfaces:

Prediction

```
public class CrwaPredictionGrid
extends PredictionGrid
```

Stores prediction grid of all locations a moving object is likely to be given an origin and period of time into the future. This format stores all values associated with a Correlated Walk Analysis Monte Carlo Prediction.

Version:

2.2, 14 Nov 2003

Author:

David Mountain; City Uni, London; dmm@soi.city.ac.uk

Field Summary

Fields inherited from class edu.cu.gi.dmm.she.grids.DoubleGrid
`attribute`, `cellValues`

Fields inherited from class edu.cu.gi.dmm.she.grids.Grid
`OUT_OF_BOUNDS`

Constructor Summary

`CrwaPredictionGrid(Grid gridDimensionsIn, double[][] cellValuesIn, java.awt.geom.Point2D.Double origin, java.util.Date originTime, long millisInFuture, SpatialHistory recentBehaviour, long recentBehaviourDuration, boolean destinationKnown, java.awt.geom.Point2D.Double actualDestination)`

Creates new PredictionGrid by setting class variables Used for an instantaneous prediction, when there is just one verification point

`CrwaPredictionGrid(Grid gridDimensionsIn, double[][] cellValuesIn, java.awt.geom.Point2D.Double origin, java.util.Date originTime, long millisInFuture, SpatialHistory recentBehaviour, long recentBehaviourDuration, boolean destinationKnown, java.awt.geom.Point2D.Double[] actualDestinations)`

Creates new PredictionGrid by setting class variables

Method Summary

`SpatialHistory`	`getRecentBehaviour()` Returns the points and attributes defining 'recent behaviour that is used for the predicting future locations'

long	getRecentBehaviourDuration() Returns the period of time into the past for which defines 'recent behaviour', from the first point in the 'recent behaviour' set, to the origin time, in millisecs

Methods inherited from class edu.cu.gi.dmm.she.grids.PredictionGrid
getActualDestination, getActualDestinations, getDestinationKnown, getInstantInterval, getPredictionDuration, getStartLocation, getStartTime, getSuccessRate, getVerificationValue

Methods inherited from class edu.cu.gi.dmm.she.grids.DoubleGrid
getCellValue, getCellValue, getGrid, toEsriString

Methods inherited from class edu.cu.gi.dmm.she.grids.Grid
getBounds, getCellRes, getCol, getNCols, getNRows, getRow, getSrs, getX, getY, outOfBounds, toEsriHeaderString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

edu.cu.gi.dmm.she.prediction.cwa

Class CrwaRandomMultiStepPrediction

```
java.lang.Object
  └ edu.cu.gi.dmm.she.prediction.cwa.CrwaPrediction
      └ edu.cu.gi.dmm.she.prediction.cwa.CrwaRandomPrediction
          └
edu.cu.gi.dmm.she.prediction.cwa.CrwaRandomMultiStepPrediction
```

All Implemented Interfaces:

Prediction

```
public class CrwaRandomMultiStepPrediction
extends CrwaRandomPrediction
```

Extends CwaRandomPrediction to be a series if short steps, rather than a single straight step.

Version: 0.3, 12 May 2005
Author: David Mountain, City Uni London, dmm@soi.city.ac.uk

Field Summary

Fields inherited from class edu.cu.gi.dmm.she.prediction.cwa.CrwaPrediction
_determistic, _multistep, _recentBehaviour

Constructor Summary

`CrwaRandomMultiStepPrediction(java.awt.geom.Point2D.Double origin, java.util.Date originTime, long predMSecs, SpatialHistory recentBehaviour)` Calls super class to get deterministic values, then resets mean and heading with a random element and recalculates the prediction

Method Summary

void	`calcPredictedPath(double meanSpeed, double sdevSpeed, double meanHeading, double sdevHeading)` Calculates the predicted path.
java.util.List	`getStepPoints()`

Methods inherited from class edu.cu.gi.dmm.she.prediction.cwa.CrwaRandomPrediction
randomizeHeading, randomizeSpeed

Methods inherited from class edu.cu.gi.dmm.she.prediction.cwa.CrwaPrediction
calcPredictedDestination, getActualDestination, getActualDestinations, getDestinationKnown, getHeadingMean, getHeadingSpread, getInstantInterval, getPredictedDestination, getPredictionDuration, getRecentHistory, getSpeedMean, getSpeedSDev,

getStartLocation, getStartTime, setHeading, setPredictedDestination, setSpeed

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

edu.cu.gi.dmm.she.prediction.cwa

Class CrwaRandomPrediction

```
java.lang.Object
  └ edu.cu.gi.dmm.she.prediction.cwa.CrwaPrediction
      └ edu.cu.gi.dmm.she.prediction.cwa.CrwaRandomPrediction
```

All Implemented Interfaces:

Prediction

Direct Known Subclasses:

CrwaRandomMultiStepPrediction

```
public class CrwaRandomPrediction
extends CrwaPrediction
```

Extends CwaPrediction to add a random element to the prediction of future locations, based on the distribution of the speed and heading attributes. Whereas CwaPrediction will always return the same prediction for a given set of points (eg based upon the mean speed and heading of those points), CwaRandomPrediction will vary the result according to the distribution of the speed and heading.

Version: 0.3, 11 May 2005
Author: David Mountain, City Uni London, dmm@soi.city.ac.uk

Field Summary

Fields inherited from class edu.cu.gi.dmm.she.prediction.cwa.CrwaPrediction
`_determistic, _multistep, _recentBehaviour`

Constructor Summary

`CrwaRandomPrediction(java.awt.geom.Point2D.Double origin, java.util.Date originTime, long predMSecs, SpatialHistory recentBehaviour)` Constructor; sets the class variables for an incoming data set.

Method Summary

`void`	`randomizeHeading(double mean, double spread)` Resets heading to a value taken from the cumulative probability function
`void`	`randomizeSpeed(double mean, double stDev)` Resets speed to a value taken from the cumulative probability function

Methods inherited from class edu.cu.gi.dmm.she.prediction.cwa.CrwaPrediction
`calcPredictedDestination, getActualDestination, getActualDestinations, getDestinationKnown, getHeadingMean, getHeadingSpread, getInstantInterval, getPredictedDestination, getPredictionDuration, getRecentHistory, getSpeedMean, getSpeedSDev, getStartLocation, getStartTime, setHeading, setPredictedDestination, setSpeed`

Package edu.cu.gi.dmm.she.timeGeog

The time geography classes are used for modeling individual accessibility, and are based upon the ideas of the time geography school. These include the space-time path, potential path area and isochrone surfaces, all discussed extensively in the body of the thesis.

Interface Summary

IIsochrone	Interface for an Isochrone.

Class Summary

AccessibilityGrid	Calculates a surface of values that representing accessiblity from some target location.
IsochroneSurface	Stores the values associated with a IsochroneSurface.
PotentialPathArea	Stores the values associated with a potential path area (calculated elsewhere @see TimeGeogCreation)
PpaBoundaryType	Stores the possible boundary types for a ppa buffer, hull, etc.
SpaceTimePath	Stores a space time path..
TimeGeogFactory	Utility class for creating Time Geog objects

edu.cu.gi.dmm.she.timeGeog

Class SpaceTimePath

```
java.lang.Object
  └edu.cu.gi.dmm.she.sets.PointSet
      └edu.cu.gi.dmm.she.timeGeog.SpaceTimePath
```

All Implemented Interfaces:

java.lang.Iterable, java.util.Collection, java.util.Set

```
public class SpaceTimePath
extends PointSet
```

Stores a space time path - a series of points representing a section of a mobile trajectory defined by an individual moving through space-time. Extends PointSet.... (calculated elsewhere @see TimeGeogCreation)

Author: David Mountain; City Uni, London; dmm@soi.city.ac.uk

Field Summary

java.util.Date	**startTime** The target location for the trace
java.awt.geom.Point2D.Double	**target** The target location for the path
java.util.List	**timeStamps** List of timestamps that correspond to the pointID 2D spatial locations

Fields inherited from class edu.cu.gi.dmm.she.sets.PointSet

centroid, nPoints, pointList

Constructor Summary

SpaceTimePath(java.util.List pointsParam, SpatialHistory parentHist, java.awt.geom.Point2D.Double targetParam)
Creates new path from a set of PointIDs, timestamps and a target location

Method Summary

java.lang.String	**toLongString**() Returns a full string representation of this class
java.lang.String	**toString**() Returns a summary string representation of this class

Methods inherited from class edu.cu.gi.dmm.she.sets.PointSet

add, addAll, clear, contains, containsAll, get, isEmpty, iterator, remove, removeAll, retainAll, size, toArray, toArray, toStringLong

Methods inherited from interface java.util.Set

equals, hashCode

edu.cu.gi.dmm.she.timeGeog

Class PotentialPathArea

```
java.lang.Object
  └edu.cu.gi.dmm.she.timeGeog.PotentialPathArea
```

Direct Known Subclasses:

PpaPrediction

```
public class PotentialPathArea
extends java.lang.Object
```

Stores the values associated with a potential path area (calculated elsewhere @see TimeGeogCreation)

Version: V1.1, 01 June 2004
Author: David Mountain; City Uni, London; dmm@soi.city.ac.uk

Field Summary

long	_timeBudget The time budget to be used when calculating this potential path area.
int	boundaryType Stores whether the boundary type used for this potential path area.
java.awt.geom.Point2D.Double[]	hullPoints The array of points that defines the bounding convex hull, that can be used to represent the region of space accessible given a time budget of timeCutoff, for this ppa This object is only initialized if the boundaryType == PpaBoundaryType.CONVEX_HULL
java.util.List	paths The list of space time paths that defines this ppa
com.vividsolutions.jts.geom.Geometry[]	stPathBuffers An array of polygons that defines the buffers, around each space time path, that can be used to represent the region of space accessible given a time budget of timeCutoff, for this ppa.
java.awt.geom.Point2D.Double	targetPoint The target spatial location (2D) that is the starting position for this ppa

Constructor Summary

PotentialPathArea(java.util.List paths, com.vividsolutions.jts.geom.Geometry[] stPathBuffers, java.awt.geom.Point2D.Double targetPoint, long timeBudget) Creates new PotentialPathArea.
PotentialPathArea(java.util.List paths,

`java.awt.geom.Point2D.Double[] polyPoints,`
`java.awt.geom.Point2D.Double targetPoint, long timeBudget)`
Creates new PotentialPathArea.

`PotentialPathArea``(PotentialPathArea ppa)` Creates new PotentialPathArea from an existing Potential Path Area.

Method Summary

`double`	**`getSurfaceArea`**`()` the surface area of the hull
`java.lang.String`	**`toEsriGenerateString`**`()` Exports the spatial boundary of this PotentialPathArea as an ESRI GENERATE string generate format is

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

edu.cu.gi.dmm.she.timeGeog

Class PpaBoundaryType

```
java.lang.Object
  └─edu.cu.gi.dmm.she.timeGeog.PpaBoundaryType
```

```
public class PpaBoundaryType
extends java.lang.Object
```

Stores the possible boundary types for a ppa buffer, hull, etc.

Version:
1.0, 22 Jun 2005

Author:
David Mountain; City Uni, London; dmm@soi.city.ac.uk

Field Summary

static int	BUFFER Indicates that the boundary types is a buffer hull
static int	CONVEX_HULL Indicates that the boundary types is a convex hull
static int	UNKNOWN_PARAMETER

Constructor Summary

PpaBoundaryType()

Method Summary

static int	getKeyFromLabel(java.lang.String label) returns the string label associated with a boundary type
static java.lang.String	getLabelFromKey(int key) returns the string label associated with a boundary type

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

edu.cu.gi.dmm.she.timeGeog
Interface IIsochrone

`public interface IIsochrone`

Interface for an Isochrone. Has polyline spatial geometry, a single temporal attribute and spatial origin location.

Version:
    V0.1, 22 June 2004

Author:
    David Mountain; City Uni, London; dmm@soi.city.ac.uk

Method Summary

com.vividsolutions.jts.geom.LineString	getIsoline()
com.vividsolutions.jts.geom.Point	getOrigin()
long	getTimeBudget()
java.lang.String	toEsriGenerateString()
java.lang.String	toString()

edu.cu.gi.dmm.she.timeGeog

Class IsochroneSurface

```
java.lang.Object
  └─edu.cu.gi.dmm.she.timeGeog.IsochroneSurface
```

```
public class IsochroneSurface
extends java.lang.Object
```

Stores the values associated with a IsochroneSurface. This is a series of potential path areas for the same location, that form a graded surface of how long it takes to reach different points. Calculated elsewhere @see TimeGeogCreation

Version:
V1.0, 10 Mar 2004

Author:
David Mountain; City Uni, London; dmm@soi.city.ac.uk

Field Summary

int	nPPAs The number of potential path areas that form this IsochroneSurface
java.util.List	potentialPathAreas The list of potential path areas that defines this IsochroneSurface
java.awt.geom.Point2D.Double	targetPoint The target spatial location (2D) that is the starting position for this ppa

Constructor Summary

`IsochroneSurface(java.util.List ppas, java.awt.geom.Point2D.Double targetPointParam)`
Creates new IsochroneSurface

Method Summary

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

edu.cu.gi.dmm.she.timeGeog

Class AccessibilityGrid

```
java.lang.Object
  └edu.cu.gi.dmm.she.grids.Grid
      └edu.cu.gi.dmm.she.grids.DoubleGrid
          └edu.cu.gi.dmm.she.grids.PredictionGrid
              └edu.cu.gi.dmm.she.timeGeog.AccessibilityGrid
```

All Implemented Interfaces:

Prediction

```
public class AccessibilityGrid
extends PredictionGrid
```

Calculates a surface of values that representing accessiblity from some target location. Each cell represents the time taken to travel from the startLocation. It is usually calculated by inductive analysis of previous behaviour. It can represent the fastest traversal, slowest traversal or mean of all traversals within the search kernel for each cell in the grid. It is a subclass of PredictionGrid and adds no new methods or fields.

Version: 0.2, 17 Feb 2004

Field Summary

`int`	**kernelSearchParameter** A parameter that controls what value to search for within the searcg kernel, and hence how "free" an agent is to move when constructing this surface.
`static int`	**FASTEST** Indicates that each cell in this accessiblity grid stores the fastest recorded travel time from the target to within the search buffer around each cell
`static int`	**MEAN** Indicates that each cell in this accessiblity grid stores the mean of all recorded travel times from the target to within the search buffer around each cell
`static int`	**SLOWEST** Indicates that each cell in this accessiblity grid stores the slowest recorded travel time from the target to within the search buffer around each cell

Fields inherited from class edu.cu.gi.dmm.she.grids.DoubleGrid
`attribute, cellValues`

Fields inherited from class edu.cu.gi.dmm.she.grids.Grid
`OUT_OF_BOUNDS`

Constructor Summary

AccessibilityGrid(Grid gridDimensionsIn, double[][] cellValuesIn, java.awt.geom.Point2D.Double spatialFocus,

```
java.util.Date startTime, long maxTravelTime,
int kernelSearchParamIn)
```

Creates new AccessibilityGrid by setting class variables

Method Summary

Methods inherited from class edu.cu.gi.dmm.she.grids.PredictionGrid

`getActualDestination, getActualDestinations, getDestinationKnown, getInstantInterval, getPredictionDuration, getStartLocation, getStartTime, getSuccessRate, getVerificationValue`

Methods inherited from class edu.cu.gi.dmm.she.grids.DoubleGrid

`getCellValue, getCellValue, getGrid, toEsriString`

Methods inherited from class edu.cu.gi.dmm.she.grids.Grid

`getBounds, getCellRes, getCol, getNCols, getNRows, getRow, getSrs, getX, getY, outOfBounds, toEsriHeaderString`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Appendix 3

Results from prediction surface testing

A1 Walking scenario

A1.1 Short-term prediction (walking, t+10min)

A1.1.1 Speed-heading predictions

Table A1.1: Median verification value (vv) - Walking scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.00	0.62	1.45	2.32	9.43	9.98	9.83	7.70
Linear	0.00	0.92	1.62	2.51	9.48	9.94	10 68	6.86
power (1.5)	0.00	0.21	1.06	1.79	6.16	9.05	9.18	**10.76**
power (2)	0.00	0.00	0.06	0.23	1.80	3.07	3.09	3 69
power (3)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mean	0.00	0.35	0.84	1.37	5.38	6.41	6.56	5.80

Table A1.2: Median surface area (sa) - Walking scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	1.8E+05	6.7E+05	1.1E+06	1.6E+06	1.3E+08	2.4E+08	**2.4E+08**	8.9E+07
linear	1.0E+05	5.7E+05	9.4E+05	1.5E+06	3.1E+07	1.7E+08	1.7E+08	5.5E+07
power (1.5)	1.3E+05	3.3E+05	4.5E+05	7.2E+05	4.4E+06	1.3E+07	1.4E+07	2.0E+07
power (2)	9.0E+04	1.8E+05	2.2E+05	3.2E+05	9.5E+05	1.6E+06	1.6E+06	1.8E+06
power (3)	**4.2E+04**	6.0E+04	6.4E+04	6.7E+04	9.8E+04	9.3E+04	9.9E+04	9.7E+04
mean	1.1E+05	3.6E+05	5.5E+05	8.6E+05	3.3E+07	8.4E+07	8.6E+07	3.3E+07

Table A1.3: Median prediction surface effectiveness (pse) - Walking scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.0E+00	8.4E-07	8.6E-07	9.2E-07	3.0E-08	4.5E-08	4.6E-08	8.3E-08
linear	0.0E+00	9.5E-07	1.2E-06	9.3E-07	5.0E-08	6.8E-08	7.0E-08	1.1E-07
power (1.5)	0.0E+00	3.8E-07	1.2E-06	1.1E-06	6.9E-07	5.8E-07	5.8E-07	4.8E-07
power (2)	0.0E+00	0.0E+00	1.7E-07	3.9E-07	1.0E-06	**1.5E-06**	1.4E-06	1.4E-06
power (3)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
mean	0.0E+00	4.3E-07	6.8E-07	6.7E-07	3.6E-07	4.3E-07	4.1E-07	4.2E-07

Table A1.4: Success rate (sr) - Walking scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.56	0.73	0.77	0.87	0.68	**1.00**	**1.00**	**1.00**
linear	0.45	0.72	0.81	0.86	0.67	**1.00**	**1.00**	0.98
power (1.5)	0.48	0.65	0.75	0.85	0.97	**1.00**	**1.00**	**1.00**
power (2)	0.36	0.53	0.61	0.65	0.81	0.92	0.92	0.93
power (3)	**0.27**	0.31	0.31	0.36	0.42	0.42	0.42	0.43
mean	0.43	0.59	0.65	0.72	0.71	0.87	0.87	0.87

A1.1.2 Spatial proximity predictions

Table A1.5: Median verification value (vv) - Walking scenario, short-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.00	0.00	1.38	1.38	1.38	1.38	1.38	1.38	1.38	1.38	1.38
linear	0.00	0.00	0.60	1.77	2.36	2.94	3.14	3.51	3.66	3.89	3.89
power (1.5)	0.00	0.00	0.50	0.93	1.45	2.71	3.64	7.82	12.16	40.66	40.66
power (2)	0.00	0.00	0.24	0.56	1.00	2.30	3.40	9.44	17.00	84 98	84.98
power (3)	0.00	0.00	0.03	0.10	0.23	0.80	1.44	6.66	16.08	**179.81**	**179.81**
linear offset	0.00	0.00	1.13	1.45	1.53	1.43	1.33	1.17	1.10	1.00	1.00
mean	0.00	0 00	0.65	1.03	1.32	1.93	2.39	5.00	8.56	51.95	51.95

Table A1.6: Median surface area (sa) - Walking scenario, short-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
linear	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (1.5)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (2)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (3)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
linear offset	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
mean	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08

Table A1.7: Median prediction surface effectiveness (pse) - Walking scenario, short-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.0E+00	0.0E+00	**1.3E-06**	5.8E-07	3.2E-07	1.4E-07	8.1E-08	3.6E-08	1.3E-08	5.8E-09	3.2E-09
linear	0.0E+00	0.0E+00	5.6E-07	7.4E-07	5.5E-07	3.1E-07	1.8E-07	9.2E-08	3.4E-08	1.6E-08	9.1E-09
power (1.5)	0.0E+00	0.0E+00	4.7E-07	3.9E-07	3.4E-07	2.8E-07	2.1E-07	2.0E-07	1.1E-07	1.7E-07	9.5E-08
power (2)	0.0E+00	0.0E+00	2.3E-07	2.3E-07	2.3E-07	2.4E-07	2.0E-07	2.5E-07	1.6E-07	3.5E-07	2.0E-07
power (3)	0.0E+00	0.0E+00	2.6E-08	4.0E-08	5.4E-08	8.3E-08	8.4E-08	1.7E-07	1.5E-07	7.5E-07	4.2E-07
linear offset	0.0E+00	0.0E+00	1.1E-06	6.1E-07	3.6E-07	1.5E-07	7.8E-08	3.0E-08	1.0E-08	4.2E-09	2.3E-09
mean	0.0E+00	0.0E+00	6.1E-07	4.3E-07	3.1E-07	2.0E-07	1.4E-07	1.3E-07	8.0E-08	2.2E-07	1.2E-07

Table A1.8: Success rate (sr) - Walking scenario, short-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.04	0.32	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
linear	0.04	0.32	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
power (1.5)	0.04	0.32	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
power (2)	0.04	0.32	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
power (3)	0.04	0.32	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
linear offset	0.04	0.32	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
mean	0.04	0.32	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

A1.2 Long-term prediction (walking , t+60min)

A1.2.1 Speed-heading predictions

Table A1.9: Median verification value (vv) - Walking scenario, long-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.00	0.00	0.47	1.47	9.26	10.27	10.00	8.46
linear	0.00	0.00	0.56	1.30	9.32	10.43	**10.53**	7.82
power (1.5)	0.00	0.00	0.00	0.58	5.02	8.60	8.95	10.12
power (2)	0.00	0.00	0.00	0.00	0.21	1.26	1.34	1.45
power (3)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mean	0.00	0.00	0.20	0.67	4.76	6.11	6.16	5.57

Table A1.10: Median surface area (sa) - Walking scenario, long-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	6.3E+06	2.6E+07	3.8E+07	5.9E+07	8.5E+09	**9.3E+09**	9.0E+09	3.8E+09
linear	4.1E+06	2.1E+07	3.3E+07	5.7E+07	2.9E+09	6.6E+09	6.6E+09	2.4E+09
power (1.5)	4.7E+06	1.2E+07	1.8E+07	2.9E+07	3.1E+08	5.4E+08	5.2E+08	7.5E+08
power (2)	3.5E+06	6.9E+06	8.7E+06	1.2E+07	4.6E+07	6.2E+07	6.2E+07	7.0E+07
power (3)	**1.7E+06**	2.6E+06	2.5E+06	2.7E+06	3.9E+06	3.7E+06	3.7E+06	4.0E+06
mean	4.0E+06	1.4E+07	2.0E+07	3.2E+07	2.3E+09	3.3E+09	3.2E+09	1.4E+09

Table A1.11: Median prediction surface effectiveness (pse) - Walking scenario, long-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.0E+00	0.0E+00	9.4E-09	3.9E-09	9.0E-10	1.2E-09	1.2E-09	2.3E-09
linear	0.0E+00	0.0E+00	1.4E-08	8.1E-09	1.5E-09	1.8E-09	1.7E-09	3.0E-09
power (1.5)	0.0E+00	0.0E+00	0.0E+00	1.2E-08	1.1E-08	1.2E-08	1.2E-08	1.0E-08
power (2)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	6.8E-09	1.8E-08	1.8E-08	**1.9E-08**
power (3)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
mean	0.0E+00	0.0E+00	4.7E-09	4.7E-09	4.1E-09	6.6E-09	6.7E-09	7.0E-09

Table A1.12: Success rate (sr) - Walking scenario, long-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.36	0.67	0.71	0.75	0.79	**1.00**	**1.00**	**1.00**
linear	0.24	0.63	0.71	0.76	0.76	0.99	0.99	0.98
power (1.5)	0.28	0.50	0.61	0.69	0.85	0.93	0.93	0.94
power (2)	0.21	0.35	0.38	0.48	0.64	0.77	0.76	0.76
power (3)	**0.14**	0.15	0.15	0.20	0.25	0.27	0.26	0.25
mean	0.25	0.46	0.51	0.58	0.66	0.79	0.79	0.79

A1.2.2 Spatial proximity predictions

Table A1.13: Median verification value (vv) - Walking scenario, long-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.38	1.38	1.38	1.38
linear	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96	2.13	2.72	3.07
power (1.5)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.59	1.21	2.07	**3.25**
power (2)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30	0.78	1.60	2.93
power (3)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.16	0.47	1.15
linear offset	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.33	1.61	1.46	1.33
mean	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.77	1.21	1.62	2.19

Table A1.14: Median surface area (sa) - Walking scenario, long-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
linear	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (1.5)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (2)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (3)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
linear offset	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
mean	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08

Table A1.15: Median prediction surface effectiveness (pse) - Walking scenario, long-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	**3.6E-08**	1.3E-08	5.8E-09	3.2E-09
linear	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.5E-08	2.0E-08	1.1E-08	7.2E-09
power (1.5)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	1.5E-08	1.1E-08	8.6E-09	7.6E-09
power (2)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	7.9E-09	7.3E-09	6.7E-09	6.9E-09
power (3)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	1.0E-09	1.5E-09	1.9E-09	2.7E-09
linear offset	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0 0E+00	0.0E+00	0.0E+00	3.5E-08	1.5E-08	6.1E-09	3.1E-09
mean	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.0E-08	1.1E-08	6.7E-09	5.1E-09

Table A1.16: Success rate (sr) - Walking scenario, long-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.00	0.00	0.01	0.08	0.16	0.26	0.68	0.95	0.97	0.99	1.00
linear	0.00	0.00	0.01	0.08	0.16	0.26	0.68	0.95	0.97	0.99	1.00
power (1.5)	0.00	0.00	0.01	0.08	0.16	0.26	0.68	0.95	0.97	0.99	1.00
power (2)	0.00	0.00	0.01	0.08	0.16	0.26	0.68	0.95	0.97	0.99	1.00
power (3)	0.00	0.00	0.01	0.08	0.16	0.26	0.68	0.95	0.97	0.99	1.00
linear offset	0.00	0.00	0.01	0.08	0.16	0.26	0.68	0.95	0.97	0.99	1.00
mean	0.00	0.00	0.01	0.08	0.16	0.26	0.68	0.95	0.97	0.99	1.00

A2 Driving scenario

A2.1 Short-term prediction (driving, t+10min)

A2.1.1 Speed-heading predictions

Table A1.17: Median verification value (vv) - Driving scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.00	0.45	1.00	1.25	1.20	0.55	0.00	0.00
linear	0.00	0.00	0.84	1.64	**1.78**	1.17	1.07	0.06
power (1.5)	0.00	0.00	0.03	0.31	0.65	0.68	0.90	0.72
power (2)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
power (3)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mean	0.00	0.09	0.37	0.64	0.73	0.48	0.39	0.16

Table A1.18: Median surface area (sa) - Driving scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	3.8E+07	2.4E+08	3.8E+08	5.2E+08	6.1E+08	6.4E+08	6.1E+08	4.6E+08
linear	1.4E+07	1.6E+08	2.6E+08	4.7E+08	6.0E+08	6.3E+08	**7.1E+08**	6.3E+08
power (1.5)	2.2E+07	1.1E+08	1.3E+08	1.7E+08	1.8E+08	2.1E+08	2.2E+08	2.1E+08
power (2)	1.6E+07	5.7E+07	6.5E+07	7.4E+07	7.7E+07	7.8E+07	7.7E+07	8.2E+07
power (3)	**8.8E+06**	2.0E+07	2.2E+07	2.3E+07	2.3E+07	2.4E+07	2.5E+07	2.5E+07
mean	2.0E+07	1.2E+08	1.7E+08	2.5E+08	3.0E+08	3.2E+08	3.3E+08	2.8E+08

Table A1.19: Median prediction surface effectiveness (pse) - Driving scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.0E+00	1.9E-09	2.9E-09	2.4E-09	2.1E-09	8.5E-10	0.0E+00	0.0E+00
linear	0.0E+00	0.0E+00	2.6E-09	**3.4E-09**	2.6E-09	1.8E-09	1.7E-09	9.4E-11
power (1.5)	0.0E+00	0.0E+00	3.4E-10	1.5E-09	2.3E-09	2.7E-09	**3.4E-09**	2.6E-09
power (2)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
power (3)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
mean	0.0E+00	3.8E-10	1.2E-09	1.5E-09	1.4E-09	1.1E-09	1.0E-09	5.4E-10

Table A1.20: Success rate (sr) - Driving scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.36	0.80	0.88	0.88	0.89	0.68	0.49	0.41
linear	0.20	0.60	0.80	**0.91**	0.89	0.87	0.89	0.63
power (1.5)	0.26	0.56	0.63	0.69	0.77	0.76	0.77	0.78
power (2)	0.23	0.41	0.45	0.49	0.52	0.52	0.48	0.53
power (3)	**0.17**	0.25	0.23	0.23	0.24	0.25	0.26	0.28
mean	0.24	0.52	0.60	0.64	0.66	0.62	0.58	0.53

A2.1.2 Spatial proximity predictions

Table A1.21: Median verification value (vv) - Driving scenario, short-term prediction, spatial proximity approach

Buffer radius	250	500	1000	5000	10000	15000	25000	50000	100000	250000
no decay	0.00	0.00	0.00	0.00	0.00	1.38	1.38	1.38	1.38	1.38
linear	0.00	0.00	0.00	0.00	0.00	0.47	1.90	2.94	3.51	3.89
power (1.5)	0.00	0.00	0.00	0.00	0.00	0.48	1.02	2.71	7.82	40.66
power (2)	0.00	0.00	0.00	0.00	0.00	0.23	0.62	2.30	9.44	84.98
power (3)	0.00	0.00	0.00	0.00	0.00	0.02	0.11	0.80	6.66	**179.81**
linear offset	0.00	0.00	0.00	0.00	0.00	1.13	1.61	1.43	1.17	1.00
mean	0.00	0.00	0.00	0.00	0.00	0.62	1.11	1.93	5.00	51.95

Table A1.22: Median surface area (sa) - Driving scenario, short-term prediction, spatial proximity approach

Buffer radius	250	500	1000	5000	10000	15000	25000	50000	100000	250000
no decay	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
linear	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
power (1.5)	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
power (2)	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
power (3)	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
linear offset	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
mean	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11

Table A1.23: Median prediction surface effectiveness (pse) - Driving scenario, short-term prediction, spatial proximity approach

Buffer radius	250	500	1000	5000	10000	15000	25000	50000	100000	250000
no decay	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	**1.4E-09**	5.2E-10	1.3E-10	3.2E-11	5.2E-12
linear	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	4.8E-10	7.1E-10	2.8E-10	8.2E-11	1.5E-11
power (1.5)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	5.0E-10	3.8E-10	2.5E-10	1.8E-10	1.5E-10
power (2)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.4E-10	2.3E-10	2.2E-10	2.2E-10	3.2E-10
power (3)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E-11	4.2E-11	7.5E-11	1.6E-10	6.7E-10
linear offset	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	1.2E-09	6.1E-10	1.3E-10	2.7E-11	3.8E-12
mean	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	6.4E-10	4.2E-10	1.8E-10	1.2E-10	2.0E-10

Table A1.24: Success rate (sr) - Driving scenario, short-term prediction, spatial proximity approach

Buffer radius	250	500	1000	5000	10000	15000	25000	50000	100000	250000
no decay	0.00	0.00	0.00	0.06	0.37	0.84	1.00	1.00	1.00	1.00
linear	0.00	0.00	0.00	0.06	0.37	0.84	1.00	1.00	1.00	1.00
power (1.5)	0.00	0.00	0.00	0.06	0.37	0.84	1.00	1.00	1.00	1.00
power (2)	0.00	0.00	0.00	0.06	0.37	0.84	1.00	1.00	1.00	1.00
power (3)	0.00	0.00	0.00	0.06	0.37	0.84	1.00	1.00	1.00	1.00
linear offset	0.00	0.00	0.00	0.06	0.37	0.84	1.00	1.00	1.00	1.00
mean	0.00	0.00	0.00	0.06	0.37	0.84	1.00	1.00	1.00	1.00

A2.2 Long-term prediction (driving, t+60min)

A2.2.1 Speed-heading predictions

Table A1.25: Median verification value (vv) - Driving scenario, long-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.00	0.00	0.35	0.53	1.80	2.09	0.93	0.00
linear	0.00	0.00	0.10	0.44	1.68	1.13	2.30	1.20
power (1.5)	0.00	0.00	0.35	0.46	2.08	1.99	0.91	0.00
power (2)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
power (3)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mean	0.00	0.00	0.16	0.29	1.11	1.04	0.83	0.24

Table A1.26: Median surface area (sa) - Driving scenario, long-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	1.2E+09	7.3E+09	1.1E+10	1.5E+10	2.1E+10	2.1E+10	1.9E+10	8.5E+09
linear	5.6E+08	5.5E+09	8.6E+09	1.2E+10	2.0E+10	2.1E+10	**2.1E+10**	2.1E+10
power (1.5)	1.3E+09	6.8E+09	1.0E+10	1.5E+10	2.1E+10	2.1E+10	1.8E+10	9.4E+09
power (2)	5.9E+08	1.7E+09	2.2E+09	2.4E+09	2.4E+09	2.5E+09	2.4E+09	2.5E+09
power (3)	**3.8E+08**	6.6E+08	7.9E+08	8.1E+08	8.4E+08	7.2E+08	7.4E+08	7.9E+08
mean	8.1E+08	4.4E+09	6.5E+09	9.1E+09	1.3E+10	1.3E+10	1.2E+10	8.4E+09

Table A1.27: Median prediction surface effectiveness (pse) - Driving scenario, long-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.0E+00	0.0E+00	3.2E-11	3.6E-11	8.6E-11	8.6E-11	4.2E-11	0.0E+00
linear	0.0E+00	0.0E+00	1.0E-11	4.1E-11	7.3E-11	5.5E-11	8.6E-11	4.3E-11
power (1.5)	0.0E+00	0.0E+00	2.7E-11	3.1E-11	**9.3E-11**	8.9E-11	3.6E-11	0.0E+00
power (2)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
power (3)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
mean	0.0E+00	0.0E+00	1.4E-11	2.2E-11	5.0E-11	4.6E-11	3.3E-11	8.7E-12

Table A1.28: Success rate (sr) - Driving scenario, long-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.20	0.55	0.79	0.80	0.80	0.59	0.58	0.41
linear	0.08	0.46	0.71	0.80	**0.82**	0.79	0.80	0.58
power (1.5)	0.19	0.53	0.76	0.79	0.80	0.59	0.58	0.42
power (2)	0.07	0.21	0.25	0.27	0.26	0.28	0.24	0.30
power (3)	0.04	0.07	0.08	0.11	0.10	0.09	0.09	0.12
mean	0.12	0.36	0.52	0.55	0.56	0.47	0.46	0.37

A2.2.2 Spatial proximity predictions

Table A1.29: Median verification value (vv) - Driving scenario, long-term prediction, spatial proximity approach

Buffer radius	250	500	1000	5000	10000	15000	25000	50000	100000	250000
no decay	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.38	1.38
linear	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.49	3.14
power (1.5)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.79	**3.64**
power (2)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44	3.40
power (3)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	1.44
linear offset	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.53	1.33
mean	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	2.39

Table A1.30: Median surface area (sa) - Driving scenario, long-term prediction, spatial proximity approach

Buffer radius	250	500	1000	5000	10000	15000	25000	50000	100000	250000
no decay	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
linear	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
power (1.5)	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
power (2)	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
power (3)	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
linear offset	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11
mean	2.7E+05	1.1E+06	4.3E+06	1.1E+08	4.3E+08	9.6E+08	2.7E+09	1.1E+10	4.3E+10	2.7E+11

Table A1.31: Median prediction surface effectiveness (pse) - Driving scenario, long-term prediction, spatial proximity approach

Buffer radius	250	500	1000	5000	10000	15000	25000	50000	100000	250000
no decay	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.2E-11	5.2E-12
linear	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	**3.5E-11**	1.2E-11
power (1.5)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	1.8E-11	1.4E-11
power (2)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	1.0E-11	1.3E-11
power (3)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	1.6E-12	5.4E-12
linear offset	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.6E-11	5.0E-12
mean	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.2E-11	9.0E-12

Table A1.32: Success rate (sr) - Driving scenario, long-term prediction, spatial proximity approach

Buffer radius	250	500	1000	5000	10000	15000	25000	50000	100000	250000
no decay	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.29	1.00	1.00
linear	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.29	1.00	1.00
power (1.5)	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.29	1.00	1.00
power (2)	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.29	1.00	1.00
power (3)	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.29	1.00	1.00
linear offset	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.29	1.00	1.00
mean	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.29	1.00	1.00

A3 Daily migration scenario

A3.1 Short-term prediction (daily migration, t+10min)

A3.1.1 Speed-heading predictions

Table A1.33: Median verification value (vv) – Daily migration scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	0.48	1.78	2.46	2.08	2.01	1.62	0.00	0.00
linear	0.01	2.69	2.54	2.12	1.35	1.67	1.01	0.00
power (1.5)	0.00	2.58	2.98	2.32	2.64	3.21	2.80	1.49
power (2)	0.15	1.28	1.52	1.76	1.55	1.94	1.29	1.15
power (3)	0.00	0.00	0.00	0.00	0.05	0.00	0.08	0.00
mean	0.13	1.66	1.90	1.66	1.52	1.69	1.04	0.53

Table A1.34: Median surface area (sa) – Daily migration scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	8.3E+06	3.3E+07	2.8E+07	2.7E+07	2.7E+07	2.8E+07	1.4E+07	**1.9E+06**
linear	3.6E+06	9.8E+06	1.3E+07	1.0E+07	9.7E+06	8.6E+06	1.0E+07	2.6E+06
power (1.5)	7.1E+06	1.6E+07	1.5E+07	1.7E+07	1.5E+07	1.5E+07	1.2E+07	9.7E+06
power (2)	5.3E+06	1.0E+07	6.9E+06	7.7E+06	7.9E+06	8.3E+06	7.2E+06	7.4E+06
power (3)	2.7E+06	**3.9E+06**	3.8E+06	3.2E+06	3.3E+06	3.8E+06	3.9E+06	3.2E+06
mean	5.4E+06	1.5E+07	1.3E+07	1.3E+07	1.3E+07	1.3E+07	9.5E+06	5.0E+06

Table A1.35: Median prediction surface effectiveness (pse) – Daily migration scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	1.1E-07	6.1E-08	7.0E-08	8.3E-08	6.3E-08	8.1E-08	6.5E-09	**0.0E+00**
linear	2.1E-08	1.0E-07	1.1E-07	1.1E-07	9.5E-08	1.5E-07	5.7E-08	**0.0E+00**
power (1.5)	6.6E-08	1.5E-07	1.3E-07	1.4E-07	1.7E-07	1.8E-07	1.6E-07	5.7E-08
power (2)	5.7E-08	1.6E-07	1.6E-07	1.4E-07	**1.8E-07**	1.4E-07	1.4E-07	6.0E-08
power (3)	**0.0E+00**	9.4E-09	**0.0E+00**	1.1E-08	1.5E-08	2.1E-08	2.2E-08	4.0E-10
mean	5.2E-08	9.7E-08	9.4E-08	9.8E-08	1.0E-07	1.1E-07	7.7E-08	2.3E-08

Table A1.36: Success rate (sr) – Daily migration scenario, short-term prediction, speed-heading approach

'recent' period (mins)	5	15	30	60	180	360	720	1440
no decay	**0.83**	0.78	**0.83**	**0.83**	**0.83**	**0.83**	0.50	0.05
linear	0.56	0.78	0.78	0.78	0.78	0.78	0.78	0.16
power (1.5)	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.68
power (2)	0.72	0.67	0.72	0.67	0.67	0.67	0.67	0.68
power (3)	0.50	0.61	0.56	0.56	0.56	0.56	0.67	0.58
mean	0.66	0.70	0.71	0.70	0.70	0.70	0.66	0.43

A3.1.2 Spatial proximity predictions

Table A1.37: Median verification value (vv) – Daily migration scenario, short-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.38	1.38	1.38	1.38
linear	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.03	2.28	2.88	**3.14**
power (1.5)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.55	0.11	0.07
power (2)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.13	0.01	0.01
power (3)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
linear offset	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.38	1.71	1.45	1.33
mean	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	1.01	0.97	0.99

Table A1.38: Median surface area (sa) – Daily migration scenario, short-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
linear	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (1.5)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (2)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
power (3)	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
linear offset	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08
mean	4.3E+04	2.7E+05	1.1E+06	2.4E+06	4.3E+06	9.6E+06	1.7E+07	3.8E+07	1.1E+08	2.4E+08	4.3E+08

Table A1.39: Median prediction surface effectiveness (pse) – Daily migration scenario, short-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	**3.6E-08**	1.3E-08	5.8E-09	3.2E-09
linear	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E-08	2.1E-08	1.2E-08	7.4E-09
power (1.5)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.1E-09	5.1E-09	4.4E-10	1.7E-10
power (2)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E-10	1.2E-09	6.0E-11	2.0E-11
power (3)	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.4E-12	4.0E-11	6.7E-13	1.7E-13
linear offset	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.6E-08	1.6E-08	6.1E-09	3.1E-09
mean	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	1.7E-08	9.5E-09	4.1E-09	2.3E-09

Table A1.40: Success rate (sr) – Daily migration scenario, short-term prediction, spatial proximity approach

Buffer radius	100	250	500	750	1000	1500	2000	3000	5000	7500	10000
no decay	0.05	0.05	0.05	0.05	0.05	0.16	0.42	1.00	1.00	1.00	1.00
linear	0.05	0.05	0.05	0.05	0.05	0.16	0.42	1.00	1.00	1.00	1.00
power (1.5)	0.05	0.05	0.05	0.05	0.05	0.16	0.42	1.00	1.00	1.00	1.00
power (2)	0.05	0.05	0.05	0.05	0.05	0.16	0.42	1.00	1.00	1.00	1.00
power (3)	0.05	0.05	0.05	0.05	0.05	0.16	0.42	1.00	1.00	1.00	1.00
linear offset	0.05	0.05	0.05	0.05	0.05	0.16	0.42	1.00	1.00	1.00	1.00
mean	0.05	0.05	0.05	0.05	0.05	0.16	0.42	1.00	1.00	1.00	1.00

A3.1.3 Temporal proximity predictions

Table A1.41: Median verification value (vv) – Daily migration scenario, short-term prediction, temporal proximity approach

Time budget	5	10	15	20	25	30
buffer (250m)	0.00	0.00	**1.00**	**1.00**	**1.00**	**1.00**
convex hull	**0.00**	0.00	0.00	1.00	1.00	1.00
mean	0.00	0.00	1.00	1.00	1.00	1.00

Table A1.42: Mean surface area (sa) – Daily migration scenario, short-term prediction, temporal proximity approach

Time budget	5	10	15	20	25	30
buffer (250m)	4.6E+05	6.4E+05	8.0E+05	8.0E+05	8.4E+05	1.1E+06
convex hull	**3.7E+05**	1.1E+06	2.0E+06	3.2E+06	3.9E+06	**4.6E+06**
mean	4.1E+05	8.6E+05	1.4E+06	2.0E+06	2.4E+06	2.8E+06

Table A1.43: Mean prediction surface effectiveness (pse) – Daily migration scenario, short-term prediction, temporal proximity approach

Time budget	5	10	15	20	25	30
buffer (250m)	0.0E+00	0.0E+00	**1.3E-06**	**1.3E-06**	1.2E-06	9.5E-07
convex hull	0.0E+00	0.0E+00	0.0E+00	3.1E-07	2.5E-07	2.2E-07
mean	0.0E+00	0.0E+00	6.3E-07	7.8E-07	7.3E-07	5.8E-07

Table A1.44: Success rate (sr) – Daily migration scenario, short-term prediction, temporal proximity approach

Time budget	5	10	15	20	25	30
buffer (150m)	0.09	0.45	**0.91**	**0.91**	**0.91**	**0.91**
convex hull	0.00	0.18	0.45	0.64	0.64	0.73
mean	0.05	0.32	0.68	0.77	0.77	0.82