6 Methodology

Previous chapters have discussed dot patterns, footprints and change identifiers but have not yet detailed exactly how an application using change identifiers would be constructed. This chapter provides a framework in which the change identifiers can operate and shows how the experiments used in this thesis were constructed. The need to formalise the use of the change identifiers arises from the need to answer the questions that have emerged from the examinations of dot patterns, footprints and change identifiers namely:

1. How does the dynamic dot pattern data arrive?
2. How is the data stored?
3. How is the footprint algorithm specified?
4. How are the change identifiers run?
5. How are the results displayed?
6. How can the system be tested?

The *change identifier framework* proposed in this chapter to encompass the running of the identifiers is highly modular (Fig. 6.1) in construction allowing each of the above mentioned concerns to be dealt with individually. As shown in Fig. 6.1 the core engine of the framework requires a change identifier set and a footprint algorithm. The dynamic dot pattern is read by a buffer that 'feeds' a pattern for every timestep to the core; this pattern is processed in accordance with the change identifier set and, if an update is required, a footprint is generated using the footprint algorithm. The core sends a footprint for each timestep to the application layer (if an update has not occurred this is the same as the previous footprint) which then displays the footprint to the user.

6.1 How does the dynamic dot pattern data arrive?

When considering the way in which the dynamic dot pattern data arrives we wished to remove as many assumptions about the data as possible. The buffer (see Fig. 6.1) can be configured to work with different formats but for this thesis we use only what we consider as the bare minimum data configuration, a text file of dot locations at a given timestep with no identity information (and no guarantee that any two files have the same dot at the same position within the file). It could be argued that identity and location (or a movement vector) for the entities that have changed is less information than all of the dot

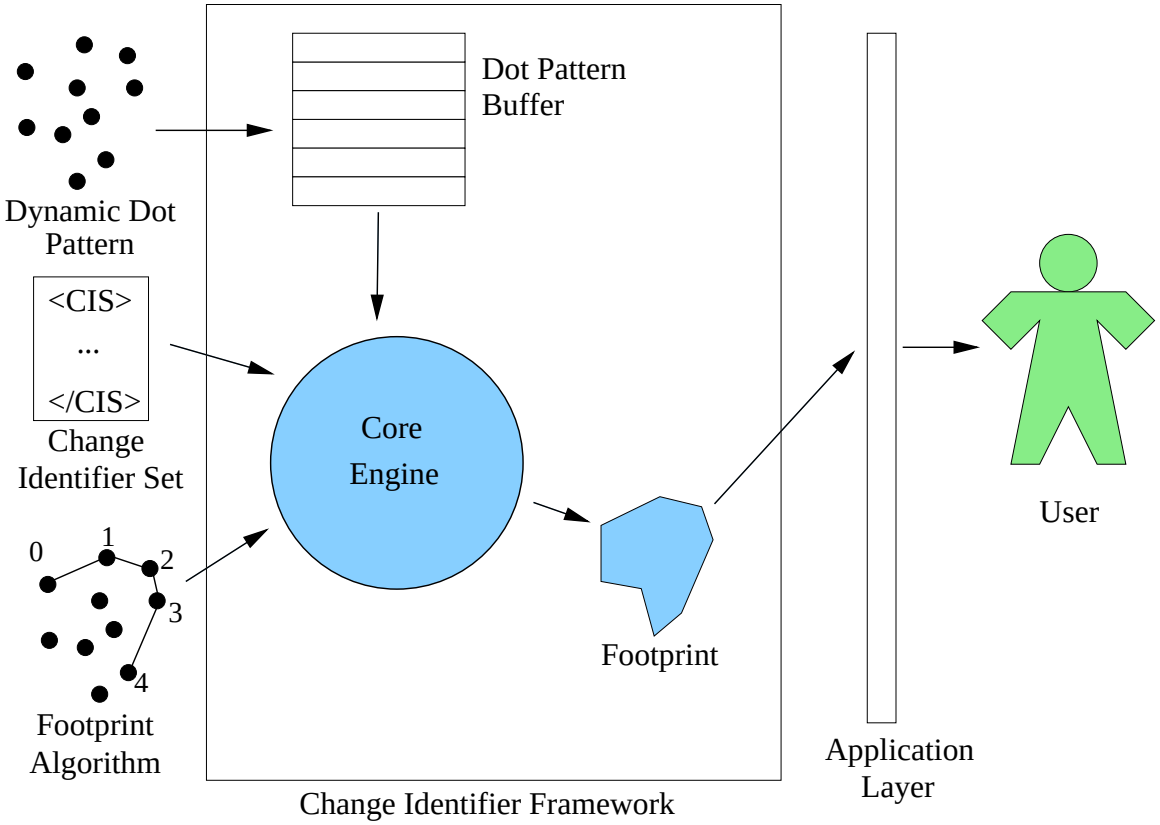


Figure 6.1 Modular Framework Architecture

locations, as it will produce a smaller file, however we took the view that reducing the types of information required would make the framework more generally applicable.

6.2 How is the data stored?

Storage of the dot pattern is a complex problem as it strongly affects the running of the change identifiers. Not having an identity associated with the dots makes performing updates on an existing structure difficult, so ideally the storage format should have a fast construction time. The main aim when considering possible data structures is to provide simple and fast access to the dots that the identifiers request. We can not provide an optimised structure to achieve this because there is no way to know in advance all possible queries that can be made by change identifiers (there being no end to the number of identifiers that can be imagined). Instead we look at the requests that we most commonly come across in the identifiers created in for this thesis; under the assumption that these common queries are likely to be consistently occurrent across the set of all possible identifiers. What we find is that most of the identifiers only wish to sum the values of the location vectors, find the centroid, find extremal points in some dimension or find (estimated) nearest neighbours. This can be achieved by maintaining as many ordered binary trees as dimensions (one for each element in the location vector). These binary trees can be built concurrently as dots from the data file are parsed. We should note that Java (the language the framework is written in) already implements the red-black tree (Guibas

and Sedgewick [32]) in its ordered sets, however even if this was not the case, red-black trees are a suitable data structure for our purposes. As was discussed in the background chapter, the red-black tree is a binary tree (each node has at most two children) with two different types of edge: red and black, and this dichromatic approach allows it to be considered as analogous to a 2-3 B-tree (a tree that can have a up to two values at each node [**?**]) by thinking of the red edges as horizontal links with a black node between them. Its structure allows for easier balancing and a computationally fast insertion time without hindering its search time. As the data structure is rebuilt for each phase, the red-black trees fast insertion and search times make it a sensible choice. The big-O notation of the complexity for a red-black tree is $O(\log n)$ for both insert and search time in normal and worst cases. The footprint algorithms will also benefit from the small search times provided by the data structure so they are by no means being hampered when we compare the time taken of using change identifiers against the time taken to update the footprint at each phase. As a final note on data structures: If the format that the data arrives in changes drastically, the buffer being distinct from the main core of the process renders the process of changing the data structure relatively easy[1].

6.3 How is the footprint algorithm specified?

The footprint algorithm is specified at the initialisation of the program. Which would be straight-forward if not for the footprint selection and parameterisation. The classification given in Chapter 4 will aid the selection of the footprint algorithm as, ideally, the user knows in advance the geometric requirements for the footprint (for example, must it be able to contain cavitities?)[2]. The parameter choice is beyond the purview of this thesis but we discuss in Chapter 10 how the identifiers might be used to help with its selection as the dynamic dot pattern changes and how the dot pattern descriptors might be used to inform the inital choice.

6.4 How are the change identifiers run?

Chapter 5 detailed how the change identifiers are described using the XML specification. The specifications are loaded into the core of the framework and the process that is implemented is shown in Algorithm 1, which works as follows: The incoming data consists of a sequence of dot patterns (e.g., from observations relayed by sensor arrays or from RFID tags attached to a flock of animals). At the beginning of the sequence a footprint $footprint(\phi_0)$ is generated for the dot pattern at phase ϕ_0 and saved as the *stored footprint* SFP_0. The phase ϕ_0 from which it is generated is stored as the *stored dot pattern* (SDP_0).

At subsequent time steps, the change identifiers are used to determine whether a new footprint should be computed; this is done by evaluating the extent to which the current phase

[1] Relative to changing the way the change identifiers are run or read in to the core

[2] The data structure may rule out some footprint algorithms that require intensity values or identities but as mentioned earlier it allows the framework to be applicable to more applications.

ϕ_i differs from the previously stored dot pattern SDP_{i-1}. If this value, $eval(\phi_i, \text{SDP}_{i-1}, \text{SFP}_{i-1})$, exceeds some pre-set threshold, then a new footprint $footprint(\phi_i)$ is generated as the new stored footprint SFP_i, and the current phase is used as the new stored dot pattern DP_i. Otherwise, the stored dot pattern and footprint are retained from the previous time step. For any phase ϕ_i, the footprint $footprint(\phi_i)$ that would be computed from it (whether or not this computation actually takes place) will be referred to (admittedly somewhat tendentiously, bearing in mind the non-uniqueness of the footprint) as the *true* footprint for that dot pattern.

Algorithm 1 Process at the Core

```
1: i = 0
2: Input first dot pattern φ_0
3: SFP_0 = footprint(φ_i)
4: SDP_0 = φ_i
5: repeat
6:    i = i + 1
7:    Input φ_i
8:    if eval(φ_i, SDP_{i-1}, SFP_{i-1}) > threshold then
9:       SDP_i = φ_i
10:      SFP_i = footprint(φ_i)
11:   else
12:      SDP_i = SDP_{i-1}
13:      SFP_i = SFP_{i-1}
14:   end if
15: until No more input available
```

6.5 How are the results displayed?

The display of the footprint is handled by the application layer. This is a necessary part of the framework for any real-world application but less so for the experiments performed in this thesis. As such the version of the program used for the experimentation runs on the command line without a Graphical User Interface (GUI). Aside from the user interface, another core difference between a testing environment and a real world application is in the consideration of the length of the dynamic dot pattern. It has been previously stated that the that there should be no restriction imposed on the length of the dynamic dot pattern so the framework must be constructed so that it can, theoretically, be run indefinitely. However any set of test data must come to an end and the length of the dynamic dot pattern must be known so that proper analysis can be performed.

6.6 How can the system be tested?

Over the course of the run on the dynamic dot pattern, the framework can store data that at the conclusion is passed to the test application. For example the length of time taken to process each change identifier, the time taken to process the entire time step and the change identifier that caused an update of the footprint (if any). Immediately after this

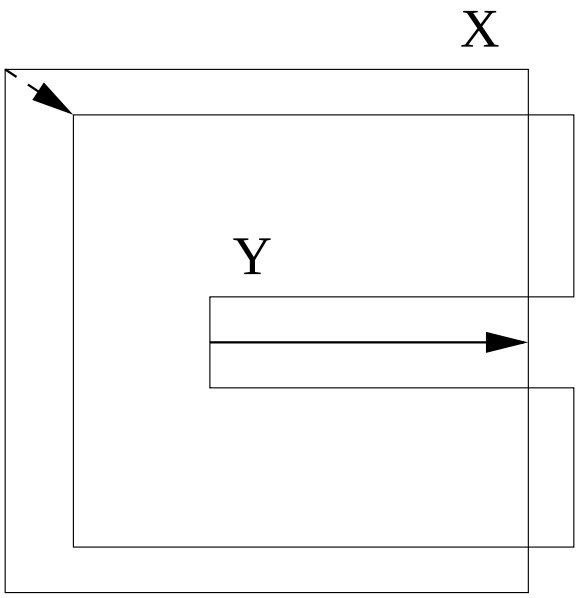


Figure 6.2 Hausdorff Distance Example

run the test application makes a call to the core for it to repeat a run over the dynamic dot pattern updating the footprint at each time step. This provides the above mentioned true footprint for each time step. As described in Chapter 5 we can use the difference between the true footprint and the stored footprint at each time step to get a measure for error. For this measure to be useful it must return a distance of 0 if the true footprint and the stored footprint are identical.

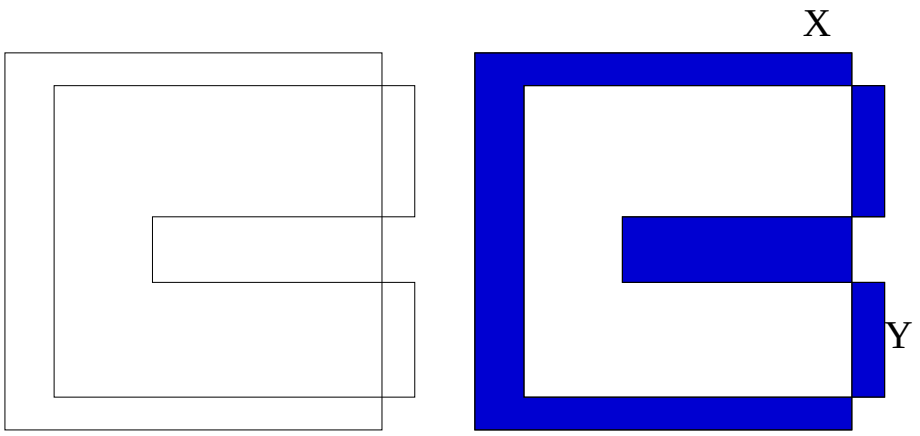
There are a number of different methods with which to ascertain the distance between two regions. Hausdorff distance, Fréchet boundary separations and symmetric area difference are three of the possible metrics that perform the measurement with different approaches ([23, ch. 7.3]). The Hausdorff distance is the greatest distance between a point within a region and the closest point in the an other, Fig. 6.2 shows an example of in which the greatest distance is from footprint Y to footprint X. Hasudorff distance has two variations: the Hausdorff boundary separation and the dual-Hausdorff distance. Hausdorff boundary separation is the Hausdorff distance of the boundaries of the regions and the dual-Hausdorff distance is the greatest of the Hausdorff distance of the two regions and the Hausdorff distance of the closed complements of the two regions.

Fréchet distance requires us to imagine the boundaries of the footprints as paths, then the returned distance is the distance of a line that connects the two paths at any two points. The standard illustration given of this is of a dog and its walker on the separate paths that travel at independant speeds; the Fréchet distance is the minimal length of leash required.

The symmetric area difference between two regions comprises the cumulative area of the parts of each region that do not overlap the other; it is given by

$$R_1 \Delta R_2 = (R_1 \setminus R_2) \cup (R_2 \setminus R_1) = (R_1 \cup R_2) \setminus (R_1 \cap R_2).$$

An example of symmetric area difference is given in Fig. 6.3, the shaded region in Fig. 6.3(a) is the parts of the regions (X and Y) that do not coincide with any part of the other region



(a) Two footprints: X and Y.

(b) The shaded region represents the symmetric area difference between footprints X and Y.

Figure 6.3 Symmetric area difference

$((X \setminus Y) \cup (Y \setminus X))$ and the area of these parts is, therefore, the symmetric area difference of X and Y.

Galton [23, ch. 7.3] provides a comprehensive discussion on these three metrics and how they relate. For now we note that symmetric area difference is the simplest to compute requiring only that the intersections between the vertices be found. It is also an intuitively clear method for measuring the similarity of the footprints, which we consider as regions, because it concerns itself with the contents of the footprint instead of its bounds. Given this simplicty and intuitive nature it is the measure we use for the experimentation performed for this thesis. Future work could examine whether different similarity measures would provide different results in the comparison of change identifiers. Although the author notes that, as long as the area of the symmetric difference provides a good measure of similarity, any difference is likely to be small. This is because the measures will only give different levels of similarity in certain specific cases (e.g., the footprint has a large external spike) and such cases are unlikely to happen consistently across the dynamic dot pattern. In effect the cases where one measure concludes that the footprints as similar and alternative measure does not will probably average out over the run of the dynamic dot pattern.

We use the area of this as a measure of the dissimilarity between two footprints; and since we are only interested in comparisons, not absolute values, we normalise this area by expressing it as a fraction of the area of the 'true' footprint ($footprint(\mathrm{DP}_i)$). Thus the aggregate mismatch between the stored footprint and the true footprint over a dot-pattern sequence of length n is given by

$$mismatch = \sum_{i=0}^{n} \frac{||footprint(\mathrm{DP}_i) \Delta\, \mathrm{SFP}_i\,||}{||footprint(\mathrm{DP}_i)||},$$

6.7 Wasteful Processing

As an addendum to considering the methodology we not two ways in which excess computation can be prevented.

Many of the discussed identifiers make use of the same calculations (e.g. bounding box, centroid). It would be wasteful to perform these calculations for each identifier so a data table is attached to each time step in the dynamic dot pattern. The identifiers can query this data table, if a value does not exist then they calculate it and add it to the table for the benefit of any identifier that may require it.

Another, minor, way in which run time can be improved is to reduce the number of times the data is iterated by considering the fashion in which dots enter the application. While for this thesis a complete list of dot positions is assumed at each time step it is also possible that the data is a description of the change for each dot (a list of dot identities with translation vectors and markers to indicate addition or removal). For either of these cases the data will not/can not arrive in the format that the software requires. It must first be translated, in our case to Java point objects. If we perform all the required iterative calculations at this pre-processing stage we slightly slow down the speed at which the patterns are received but can possibly improve the performance of the identifiers by saving them the need to iterate over the pattern.

6.8 Summary

This chapter has provided a framework for the change identifiers to be run within that allows for both real-world application use and for assessment. When considering the assessment of the change identifiers it has discussed three footprint difference measures and shown why symmetric area difference has been used for the experimentation within this thesis.

Finally this chapter has made a note of two possible ways in which the running of the change identifiers can be optimised to reduce wasteful computation.

Bibliography

[1] H. Alani, C. B. Jones, and D. Tudhope. Voronoi-based region approximation for geographical information retrieval with gazetteers. *International Journal of Geographical Information Science*, 15(4):287–306, 2001.

[2] Walid Ali and Bernard Moulin. 2d-3d multiagent geosimulation with knowledge-based agents of customers shopping behavior in a shopping mall. In AnthonyG. Cohn and DavidM. Mark, editors, *Spatial Information Theory*, volume 3693 of *Lecture Notes in Computer Science*, pages 445–458. Springer Berlin Heidelberg, 2005.

[3] Natalia Andrienko and Gennady Andrienko. Designing visual analytics methods for massive collections of movement data. *Cartographica*, 42(2):117–138, 2007.

[4] Avi Arampatzis, Marc van Kreveld, Iris Reinacher, Christopher B. Jones, Subodh Vaid, Paul Clough, Hideo Joho, and Mark Sanderson. Web-based delineation of imprecise regions. In *Computers, Environment and Urban Systems*, volume 30, pages 436–459. Elsevier, 2006.

[5] Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, SODA '97, pages 747–756, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

[6] R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1:173–189, 1972.

[7] Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry - Theory and Applications*, 2007.

[8] Brandon Bennett, Derek R. Magee, Anthony G. Cohn, and David C. Hogg. Enhanced tracking and recognition of moving objects by reasoning about spatio-temporal continuity. *Image and Vision Computing*, 26(1):67–81, January 2008.

[9] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry, Algorithms and Applications*. Springer, third edition, 2008.

[10] D. Black. *Investigation of the possible increased incidence of cancer in West Cumbria: report of the Independent Advisory Group*. H.M.S.O., 1984.

[11] P Bogaert, N Van de Weghe, AG Cohn, F Witlox, and P De Maeyer. Reasoning about moving point objects on networks. In M Raubal, J H Miller, U A Frank, and F Goodchild, editors, *4th International Conference on Geographic Information Science (GIScience 2006)*, 2006.

[12] A. Ray Chaudhuri, B. B. Chaudhuri, and S. K. Parui. A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border. In *Computer Vision and Image Understanding*, volume 68, pages 257–275. Academic Press, 1997.

[13] Yi-Jen Chiang and Roberto Tamassia. Dynamic algorithms in computational geometry. In *Proceedings of the IEEE*, number 9, pages 1412–1434, 1992.

[14] Kalyanmoy Deb. *Multi-Objective Optimization using evolutionary Algorithms*. Wiley, 2001.

[15] Géraldine Del Mondo, John G. Stell, Christophe Claramunt, and Rémy Thibaud. A graph model for spatio-temporal evolution. *Journal of Universal Computer Science*, 16(11):1452–1477, 2010.

[16] Matthias Delafontaine, Anthony G. Cohn, and Nico Van de Weghe. Implementing a qualitative calculus to analyse moving point objects. *Expert Systems with Applications*, 38(5):5187–5196, 2011.

[17] Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. Towards a taxonomy of movement patterns. *Information Visualization*, (7):240–252, 2008.

[18] Matt Duckham, Lars Kulik, Mike Worboys, and Antony Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. In *Pattern Recognition*, volume 41, pages 3224–3236. Elsevier, 2008.

[19] Max Dupenois and Antony Galton. Assigning footprints to dot sets: An analytical survey. In K. S. Hornsby, C. Claramunt, M. Denis, and G. Ligozat, editors, *Spatial Information Theory: Proceedings of the 9th International Conference COSIT 2009*, pages 227–244, Berlin, 2009. Springer.

[20] H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois, 1992.

[21] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. In *ACM Transactions on Graphics*, volume 13, pages 43–72. 1994.

[22] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. In *Computer Vision and Image Understanding*, volume IT-29, pages 551–559. IEEE, 1983.

[23] Antony Galton. *Qualitative Spatial Change*. Oxford University Press, 2000.

[24] Antony Galton. Pareto-optimality of cognitively preferred polygonal hulls for dot patterns. In *Spatial Cognition*, 2008.

[25] Antony Galton and Matt Duckham. What is the region occupied by a set of points? In *GIScience*, 2006.

[26] Yossi Gofman. Outline of a set of points. *Pattern Recognition Letters*, 14(1):31–38, 1993.

[27] Christopher M. Gold. Data structures for dynamic and multidimensional gis. In *4th ISPRS Workshop on Dynamic and Multi-dimensional GIS*, pages 36–41, Pontypridd, Wales, UK, 2005.

[28] D E Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[29] Ronald L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1(4):132–133, 1972.

[30] Leonidas Guibas. Kinetic data structures. In D. Mehta and S. Sahni, editors, *Handbook of Data Structures and Applications*, pages 23–1–23–18. Chapman and Hall/CRC, 2004.

[31] Leonidas Guibas, Menelaos Karaveles, and Daniel Russel. A computational framework for handling motion. In *Proceedings of teh Sixth Workshop on Algorithm Engineering and Experiments*, pages 129–141, 2004.

[32] Leonidas J. Guibas and Robert Sedgewick. A dichromatic framework for balanced trees. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:8–21, 1978.

[33] John Hershberger and Subhash Suri. Convex hulls and related problems in data streams. In *Proceedings of ACM/DIMACS Workshop on Management and Processing of Data Streams*, pages 148–168, 2003.

[34] Kathleen Hornsby and Max J. Egenhofer. Qualitative representation of change. In S. Hirtle and A. Frank, editors, *Spatial Information Theory: A Theoretical Basis for GIS, Proceedings of the International Conference COSIT'97*, pages 15–33. Springer-Verlag, 1997.

[35] Yan Huang, Cai Chen, and Pinliang Dong. Modeling herds and their evolvements from trajectory data. In *GIScience '08: Proceedings of the 5th international conference on Geographic Information Science*, pages 90–105, Berlin, Heidelberg, 2008. Springer-Verlag.

[36] R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. In *Information Processing Letters*, volume 2, pages 18–21. North-Holland Publishing Company, 1973.

[37] Jixiang Jiang and Michael Worboys. Detecting basic topological changes in sensor networks by local aggregation. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, GIS '08, pages 4:1–4:10, New York, NY, USA, 2008. ACM.

[38] Jixiang Jiang, Michael Worboys, and Silvia Nittel. Qualitative change detection using sensor networks based on connectivity information. *GeoInformatica*, 15:305–328, 2011.

[39] R Klette and A Rosenfeld. *Digital Geometry: Geometric Methods for Digital Picture Analysis.* Morgan Kaufmann, 2004.

[40] Donald Knuth. *The Art Of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, 2nd edition, 2007.

[41] P. Laube, M. Van Kreveld, and S. Imfeld. Finding remo - detecting relative motion patterns in geospatial lifelines. In P. F. Fisher, editor, *Developments in Spatial Data Handling: Proceedings of the 11th International Symposium on Spatial ata Handling*, pages 201–214. Springer, 2004.

[42] Patrick Laube, Matt Duckham, and Marimuthu Palaniswami. Deferred decentralized movement pattern mining for geosensor networks. *International Journal of Geographical Information Science*, 25(2):273–292, 2011.

[43] Patrick Laube and Ross S. Purves. How fast is a cow? cross-scale analysis of movement data. *Transactions in GIS*, 15(3):401–418, 2011.

[44] Mahmoud Melkemi. $\mathcal{A}$-shapes of a finite point set. In *Proceedings of the thirteenth annual symposium on Computational geometry*, SCG '97, pages 367–369. ACM, 1997.

[45] Mahmoud Melkemi and Mourad Djebali. Computing the shape of a planar points set. *Pattern Recognition*, 33(9):1423 – 1436, 2000.

[46] Mahmoud Melkemi and Mourad Djebali. Weighted $\mathscr{A}$-shape: a descriptor of the shape of a point set. *Pattern Recognition*, 34(6):1159 – 1170, 2001.

[47] Adriano Moreira and Maribel Yasmina Santos. Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. In *International Conference on Computer Graphics Theory and Applications GRAPP*, 2007.

[48] David O'Sullivan and David J. Unwin. *Geographic Information Analysis*. Wiley, November 2002.

[49] Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Science*, 23(2):166–204, 1981.

[50] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA, 1987. ACM.

[51] P.L. Rosin. Measuring shape: ellipticity, rectangularity, and triangularity. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, pages 952 –955 vol.1, 2000.

[52] John G. Stell. Granularity in change over time. In M. Duckham, M. Goodchild, and M Worboys, editors, *Foundations of Geographic Information Science*, chapter 6, pages 95 – 115. Taylor and Francis, 2003.

[53] Marius Thériault, Christophe Claramunt, and Paul Villeneuve. A spatio-temporal taxonomy for the representation of spatial set behaviours. In Michael Bhlen, Christian Jensen, and Michel Scholl, editors, *Spatio-Temporal Database Management*, volume

1678 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg, 1999.

[54] Zena Wood. *Detecting and Identifying Collective Phenomena within Movement Data.* PhD thesis, University of Exeter, 2011.

[55] Zena Wood and Antony Galton. A taxonomy of collective phenomena. *Applied Ontology*, 4:267–292, August 2009.

[56] Michael Worboys. Event-oriented approaches to geographic phenomena. *International Journal of Geographical Information Science*, 19:1–28, 2005.

[57] Michael Worboys and Matt Duckham. *GIS: A Computing Perspective*, chapter 6.4 Point Object Structures, pages 240 – 248. CRC Press, 2nd edition, 2004.

[58] J. Zunic and P.L. Rosin. A convexity measurement for polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:173–182, 2002.

[59] J. Zunic and P.L. Rosin. Rectilinearity measurements for polygons. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1193 – 1200, September 2003.