

Identification of Change in a Dynamic Dot Pattern and its use in the Maintenance of Footprints

Maximillian Philip Dupenois

March 2012

Supervised by Dr Antony Galton and Dr Joviša Žunić

Submitted by Maximillian Philip Dupenois, to the College of Engineering, Mathematics and Physical Sciences as a thesis for the degree of Doctor of Philosophy in Computer Science, March 2012.

This thesis is available for Library use on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

(signature)

Abstract

Examples of spatio-temporal data that can be represented as sets of points (called dot patterns) are pervasive in many applications, for example when tracking herds of migrating animals, ships in busy shipping channels and crowds of people in everyday life. The use of this type of data extends beyond the standard remit of Geographic Information Science (GISc), as classification and optimisation problems can often be visualised in the same manner.

A common task within these fields is the assignment of a region (called a footprint) that is representative of the underlying pattern. The ways in which this footprint can be generated has been the subject of much research with many algorithms having been produced. Much of this research has focused on the dot patterns and footprints as static entities, however for many of the applications the data is prone to change.

This thesis proposes that the footprint need not necessarily be updated each time the dot pattern changes; that the footprint can remain an appropriate representation of the pattern if the amount of change is slight. To ascertain the appropriate times at which to update the footprint, and when to leave it as it is, this thesis introduces the concept of change identifiers as simple measures of change between two dot patterns. Underlying the change identifiers is an in-depth examination of the data inherent in the dot pattern and the creation of descriptors that represent this data.

The experimentation performed by this thesis shows that change identifiers are able to distinguish between different types of change across dot patterns from different sources. In doing so the change identifiers reduce the number of updates of the footprint while maintaining a measurably good representation of the dot pattern.

Without my primary supervisor, Dr Antony Galton, this thesis would not have been possible, so I thank him for all the help and encouragement that he has given throughout my PhD. I would also like to thank Dr Jennifer Evans, my parents and my brothers for their unconditional support during the research and writing of this thesis. Finally I would like to thank any colleague, particularly Dr Jacqueline Christmas, Dr Zena Wood, Dr Andrew Clark, Kent McClymont and David Walker, that has patiently listened and offered input while I have discussed my work with them.

Contents

List of tables	8
List of figures	9
Nomenclature and Abbreviations	14
1 Introduction	15
1.1 Overview	15
1.2 Terminology	19
1.3 Summary of Thesis And the Novelties it Presents	22
2 Background	24
2.1 GIS and Dot Patterns	24
2.2 Data Structures	25
2.3 Spatio-Temporal Data	29
2.3.1 <i>Qualitative Representation of Change</i> [Hornsby and Egenhofer] . . .	29
2.3.2 <i>A Spatio-Temporal Taxonomy for the Representation of Spatial Set Behaviours</i> [Thériault <i>et al.</i>]	30
2.3.3 <i>Granularity in Change Over Time</i> [Stell]	32
2.3.4 <i>Finding REMO - detecting relative motion patterns in geospatial lifelines</i> [Laube <i>et al.</i>]	33
2.3.5 <i>Event-oriented approaches to geographic phenomena</i> [Worboys] . . .	33
2.3.6 <i>Reporting Flock Patterns</i> [Benkert <i>et al.</i>]	34
2.3.7 <i>Designing visual analytics methods for massive collections of move- ment data</i> [Andrienko and Andrienko]	34
2.3.8 <i>Towards a Taxonomy of Movement Patterns</i> [Dodge <i>et al.</i>]	35
2.3.9 <i>Modelling Herds and Their Evolvments from Trajectory Data</i> [Huang <i>et al.</i>]	36
2.3.10 <i>A taxonomy of collective phenomena</i> [Wood and Galton] and <i>Detect- ing and Identifying Collective Phenomena within Movement Data</i> [Wood]	36
2.3.11 <i>A Graph Model for Spatio-temporal Evolution</i> [Del Mondo <i>et al.</i>] . .	38
2.3.12 <i>How fast is a cow? Cross-Scale Analysis of Movement Data</i> [Laube and Purves]	38
2.3.13 Others	39
2.4 Shape	39

2.4.1	<i>Qualitative Spatial Change</i> [Galton]	40
2.4.2	Others	42
2.5	Footprints	42
2.6	Dynamics	50
2.7	Summary	53
3	Dot Patterns	54
3.1	Change and Dot Patterns	54
3.2	Descriptor Classes	54
3.3	Descriptors	59
3.3.1	Position	59
3.3.2	Extent	60
3.3.3	Orientation	60
3.3.4	Connectedness	61
3.3.5	Dimensionality	63
3.3.6	Dispersion	64
3.4	Descriptor Analysis	66
3.5	Summary	74
4	Footprints	76
4.1	What is a footprint?	76
4.1.1	A Troublesome Example	79
4.1.2	Unique Footprints	80
4.1.3	Information Content	81
4.2	Footprint Classification	83
4.2.1	Intrinsic footprint criteria	84
4.2.2	Relational footprint criteria	86
4.3	Using the Footprint Classification	87
4.4	Summary	90
5	Change Identifiers	91
5.1	What is a Change Identifier?	91
5.2	Distance Change Identifiers	94
5.3	Change Identifier Sets	97
5.4	Constructing Change Identifiers	100
5.4.1	Descriptor Identifiers	101
5.4.2	Distance Identifiers	104
5.5	Performance Analysis	105
5.6	Summary	107
6	Methodology	109
6.1	How does the dynamic dot pattern data arrive?	109
6.2	How is the data stored?	110
6.3	How is the footprint algorithm specified?	111
6.4	How are the change identifiers run?	111
6.5	How are the results displayed?	112

6.6	How can the system be tested?	112
6.7	Eliminating Wasteful Processing	117
6.8	Summary	118
7	Results	119
7.1	The Dynamic Dot Patterns	119
7.2	The Algorithms	121
7.3	The Change Identifier Sets	122
7.4	Plotting the experiments	122
7.5	Results	125
7.5.1	Footprints	125
	Results	125
	Discussion	125
7.5.2	Change Identifier Sets Time	130
	Results	130
	Discussion	130
7.5.3	Change Identifier Sets Error	131
	Results	131
	Discussion	132
7.5.4	Time against Error	136
	Results	136
	Discussion	136
7.5.5	Individual Run: NEARESTNEIGHBOURDISTVARIANCE on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull.	141
	Results	141
	Discussion	143
7.5.6	Individual Run: NEARESTNEIGHBOURDISTVARIANCE on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the minimum isothetic bounding box.	145
	Results	145
	Discussion	145
7.5.7	Individual Run: NEARESTNEIGHBOURDISTVARIANCE on the random dynamic dot pattern using the χ -hull.	146
	Results	146
	Discussion	147
7.5.8	Comparison of USERSELECTED and NEARESTNEIGHBOURDISTVARIANCE.	152
	Results	152
	Discussion	152
7.6	Summary	156
8	Summary and Conclusions	157
8.1	Thesis Summary	157
8.2	Conclusions	158
8.3	Future Work	159

8.3.1	Dot Patterns	159
	Dot Pattern Types	159
	Plotting a Dynamic Dot Pattern	162
8.3.2	Footprints	163
	Footprint Types	163
8.3.3	Footprint Algorithm Parameters	163
8.3.4	Change Identifiers	164
8.3.5	Other Fields	165
	Appendices	166
	A Extra Result Graphs	167
	Bibliography	171

List of Tables

4.1	Table showing classification of Fig. 4.14	88
4.2	Algorithm Classification Examples	89
5.1	The patterns arriving at twice the speed it takes a footprint to be computed.	92
5.2	Table showing change identifier complexities.	105
6.1	Difference in approaches for shape similarity measures.	116

List of Figures

1.1	Footprint as a Representative	15
1.2	Time Domain (T)	20
2.1	An example quadtree division that shows the route taken to find a given dot	26
2.2	Examples of the Binary Tree Data Structure	27
2.3	Examples of the 2-3 B-tree Data Structure	28
2.4	A Dot Pattern with Associated 2d-Tree.	29
2.5	A single dot pattern showing disconnected components.	41
2.6	When a convex hull is inappropriate	43
2.7	Images show the difference between α -hull and α -shape. Image from [Edelsbrunner et al., 1983]	44
2.8	Negative α -hull and α -shape. Image from [Edelsbrunner et al., 1983]	44
2.9	Example of Uncertainty Triangles. Image from [Hershberger and Suri, 2003]	51
3.1	Dynamic dot pattern changing in position.	55
3.2	Demonstrating the relative difference in position change of two patterns of different extents.	56
3.3	Dynamic dot pattern changing in extent.	56
3.4	Dynamic dot pattern shearing. The arrow represents the pattern's orientation (as found by Ordinary Least Squares regression).	56
3.5	Dynamic dot pattern changing in orientation.	57
3.6	Dynamic dot pattern changing in connectedness.	57
3.7	Dynamic dot pattern changing in dimension.	57
3.8	Two dot patterns with the same extent, position, orientation, connectedness and dimensionality	58
3.9	OLS (Solid line) VS. PCA (Dashed Line)	62
3.10	Agglomerative Clustering Method	63
3.11	The case when the x and y ordered trees will not locate the nearest neighbour. $\vec{o}\vec{a}$ is 5 units in length while $\vec{o}\vec{b}$ is 6.1 units.	65
3.12	Examples of dot patterns produced using the random generation method	67
3.13	Correlation Heat Map: All Descriptors	68
3.14	Examples of different values for estimated nearest neighbour variance and collinearity	69
3.15	Time taken (nanoseconds) per dynamic dot pattern phase: All Descriptors	70
3.16	Time taken (nanoseconds) per dynamic dot pattern phase: Descriptors of extent	72
3.17	Time taken (nanoseconds) per dynamic dot pattern phase: Descriptors of orientation	72

3.18	Time taken (nanoseconds) per dynamic dot pattern phase: Descriptors of dimensionality	73
3.19	Time taken (nanoseconds) per dynamic dot pattern phase: Descriptors of dispersion	73
3.20	Average time taken (nanoseconds) per dynamic dot pattern phase: All classes	74
3.21	Time taken (nanoseconds) per dynamic dot pattern phase: Fastest non-correlated Descriptors	75
4.1	Example of the convex hull remaining the same despite differences between the dot patterns.	77
4.2	Example of pattern difference decreasing while footprint difference increases.	78
4.3	Example of small difference between dot patterns leading to large difference between footprints created by the Swinging Arm algorithm. The red dot is further to the left in ϕ_{i+1} and, as such, the line length is no longer long enough to reach it.	78
4.4	Example of a dot pattern in which a degenerate line may be desirable. . . .	80
4.5	Figure showing that a footprint algorithm is not injective, both dot patterns a and b map to the same footprint.	81
4.6	Changes in information content for a footprint over a small dot pattern . . .	83
4.7	Connectedness examples	84
4.8	Regular	84
4.9	Polygonal	85
4.10	Jordan Boundary	85
4.11	Simply Connected	85
4.12	Curvature Extrema and Dot Boundary Example	86
4.13	Full Coverage	87
4.14	Footprint type tracking example.	88
5.1	Has the dot pattern at ϕ_1 changed sufficiently for the footprint to be updated?	91
5.2	Figure showing increasing lag in footprint representation	92
5.3	Difference in bounding box area for two dynamic dot patterns.	93
5.4	Figure showing change in effect for a change in position for two different sized patterns	95
5.5	Simplification of prey being chased by a predator	97
5.6	Symmetric area difference between boxes a and b . The shaded area is the difference	105
5.7	Total computation time against aggregate footprint error	107
6.1	Modular Framework Architecture	110
6.2	Hausdorff distance Example.	114
6.3	Hausdorff boundary separation Example.	114
6.4	Symmetric area difference	115
6.5	Example of the inconsistent fashion in which the Hausdorff Distance treats spikes. The black arrow is the Hausdorff distance in both images	115

6.6	Example of the small Hausdorff boundary separation given when one footprint encloses another.	115
7.1	Example of a graph of time taken per timestep.	123
7.2	Example of a graph of symmetric area difference per timestep.	123
7.3	Example of a graph of time against error for multiple change identifier sets.	124
7.4	An example selection of eight timesteps ($\phi_{100} - \phi_{108}$) showing the footprint with change identifiers (solid red) and the footprint without change identifiers (dashed black) on the dynamic dot pattern using a change in extent and cardinality behaviour and the χ -hull algorithm	126
7.5	An example selection of eight timesteps ($\phi_{109} - \phi_{116}$) showing the footprint with change identifiers (solid red) and the footprint without change identifiers (dashed black) on the dynamic dot pattern using a change in extent and cardinality behaviour and the χ -hull algorithm	127
7.6	Average time taken per timestep for each footprint algorithm on each dot pattern type.	128
7.7	Average time taken per timestep for the minimum isothetic bounding box algorithm on each dot pattern type.	129
7.8	Average time taken per timestep for the fastest performing thresholds of each change identifier set on each dot pattern type.	131
7.9	Average proportionate error per timestep for the best performing thresholds of each change identifier set on each dot pattern type.	132
7.10	Graphs showing symmetric area difference against change identifier threshold for the set containing all identifiers for different thresholds.	133
7.11	Graphs showing symmetric area difference against change identifier threshold for the BOUNDINGBOXAREA set identifier for different thresholds.	134
7.12	Graphs showing symmetric area difference against change identifier threshold for the NEARESTNEIGHBOURDISTVARIANCE set identifier for different thresholds.	135
7.13	Graph showing dominance relations.	137
7.14	Time taken against proportionate error per timestep averaged across all dot patterns for the best performing thresholds of each change identifier set. The first plot is of all the change identifiers together; the subsequent plots are the bands of similar symmetric area difference from the original plot. Note the difference in scales along the x axis. (Legend is given in Fig. 7.16)	138
7.15	Time taken against proportionate error totalled for each run averaged across all dot patterns for the best performing thresholds of each change identifier set. The layout is the same as for Fig. 7.14; the first plot shows all of the identifier sets and the subsequent plots are ‘zoomed’ in sections of this initial plot. (Legend is given in Fig. 7.16)	139
7.16	Legend for plots Fig. 7.14 and Fig. 7.15	140
7.17	Time taken against error per timestep averaged across all dot patterns for the best performing thresholds of the identifiers appearing in Band 1	141

7.18	Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull footprint algorithm.	142
7.19	Proportionate error at each timestep for NEARESTNEIGHBOURDISTVARIANCE - 0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull footprint algorithm.	142
7.20	Example of the Problem in Using Symmetric Area Difference as an Error Measure	143
7.21	Proportionate error histograms for NEARESTNEIGHBOURDISTVARIANCE - 0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern	144
7.22	Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the isothetic minimum bounding box footprint algorithm.	145
7.23	Proportionate error at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the isothetic minimum bounding box footprint algorithm.	146
7.24	Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Random dynamic dot pattern using the χ -hull footprint algorithm.	147
7.25	Difference between with and without in time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Random dynamic dot pattern using the χ -hull footprint algorithm.	148
7.26	Proportionate error at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Random dynamic dot pattern using the χ -hull footprint algorithm. The red line indicates the mean proportionate error.	149
7.27	Difference between with and without in time taken to compute at each timestep for FASTESTNODISTANCE-0.25 on the Random dynamic dot pattern using the χ -hull footprint algorithm.	150
7.28	Proportionate error at each timestep for FASTESTNODISTANCE-0.25 on the Random dynamic dot pattern using the χ -hull footprint algorithm. The red line indicates the mean proportionate error.	151
7.29	Time taken against proportionate error per timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, NEARESTNEIGHBOURDISTVARIANCE-0.25 and USERSELECTED-0.5 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull footprint algorithm.	153
7.30	Proportionate error at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, NEARESTNEIGHBOURDISTVARIANCE-0.25 and USERSELECTED-0.5 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull footprint algorithm.	154
7.31	Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, NEARESTNEIGHBOURDISTVARIANCE-0.25 and USERSELECTED-0.5 on the Random dynamic dot pattern using the χ -hull footprint algorithm.	155
8.1	Graph showing Sigmoidal Normalisation Curve $\frac{x}{1+ x }$	160
8.2	Dendrogram for the clustering of dot patterns by the fastest non-correlated descriptor set	161

8.3 Snapshot of the clustering process of dot patterns by the fastest non-correlated descriptor set. 162

A.1 Time taken against error per timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, OPTIMISED[304], USERSELECTED-BOIDS and the user selected change identifier set created for the Extent-Cardinality-Expand-Noisy dynamic dot pattern, now called USERSELECTED-EXTENTCARD using the χ -hull footprint algorithm on the boid like dynamic dot pattern. 167

A.2 Proportionate symmetric area difference at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, OPTIMISED[304], USERSELECTED-BOIDS and the user selected change identifier set created for the Extent-Cardinality-Expand-Noisy dynamic dot pattern, now called USERSELECTED-EXTENTCARD using the χ -hull footprint algorithm on the boid like dynamic dot pattern. 168

A.3 Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, OPTIMISED[304], USERSELECTED-BOIDS and the user selected change identifier set created for the Extent-Cardinality-Expand-Noisy dynamic dot pattern, now called USERSELECTED-EXTENTCARD using the χ -hull footprint algorithm on the boid like dynamic dot pattern. 169

Nomenclature and Abbreviations

Abbreviations and Initialisations

GIS	Geographic Information System.
GISc	Geographic Information Science.
KDS	Kinetic Data Structures.
OLS	Ordinary Least Squares.
PCA	Principal Component Analysis.

Nomenclature

$change_f$	Change identifier f .
DDP_b	Dynamic dot pattern b .
$desc_e$	Descriptor e .
d_d	Dot d from a dot pattern.
DP_c	Dot pattern c .
$foot(DP_c)$	A footprint over DP_c .
$foot(\phi_a)$	A footprint over ϕ_a .
$dist_g$	Distance change identifier g .
n	Number of dots in a pattern.
p	Number of patterns in a dynamic dot pattern.
ϕ_a	Phase a of a dynamic dot pattern.
T	A time domain containing timesteps.
τ	A timestep from a time domain.

1. Introduction

1.1. Overview

Many phenomena, real and abstract, can be represented by sets of point locations. These *dot patterns* occur in many fields including Geographic Information Systems (GISs), classification and optimisation. Dot patterns are a useful abstraction allowing for many types of mathematical analysis to be performed e.g., using statistical analysis methods to see if the spread of leukemia within the vicinity of a nuclear power station is significant. O’Sullivan and Unwin [O’Sullivan and Unwin, 2002, ch. 5.1] detail such analysis while commenting on the 1984 report by Sir Donald Black [Black, 1984].

Often a way of approximating the region that contains a pattern is required, and an areal or volumetric object used for this approximation can be called a *footprint* of the dot pattern. Footprints can be used to reduce the memory space taken up by the pattern, and in some cases they can be interpreted in such a way as to increase the known information about the phenomenon underlying the pattern. For example, consider the case of a location aware application that wishes to identify the boundaries of a city by the locations of buildings sampled from within it ([Moreira and Santos, 2007; Alani et al., 2001]); the boundary locations are not present in the data but can be approximated by a footprint.

In theory any bounded region is a candidate footprint for a given dot pattern, however it is intuitively obvious that some footprints are better than others. Just how appropriate a footprint is for a dot pattern is a clearly context specific problem. However there are some general distinctions that can be made. One of the goals of footprint creation is to make clear the information that the pattern represents; a footprint that only serves to

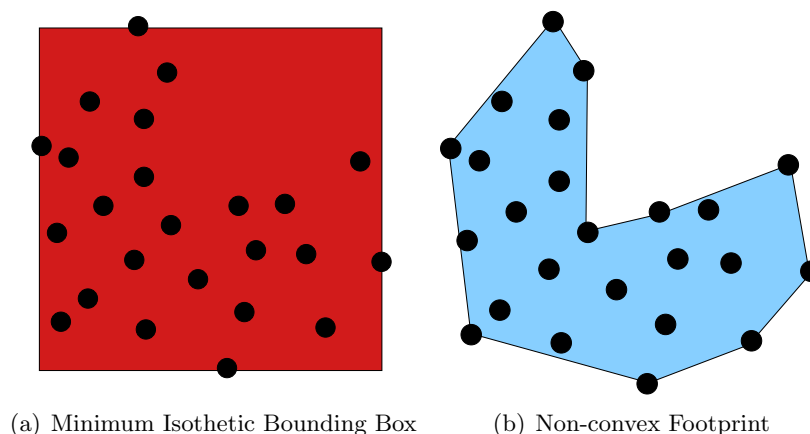


Figure 1.1. Footprint as a Representative

obfuscate is unlikely to be desirable. In Fig. 1.1 we can see two footprints (Fig. 1.1(a) and Fig. 1.1(b)) for the same dot pattern. Fig. 1.1(a) may be a sufficiently ‘good’ representation for the application context but Fig. 1.1(b) provides more information about the spread of the pattern. In using clarity of salient information as a criterion we can state that Fig. 1.1(b) is a ‘better’ representation than Fig. 1.1(a) of the dot pattern. We must be aware, however, that there is generally an inverse relation between the information content of the footprint and its computational complexity; this trade-off is discussed in greater detail in Chapter 4. Current work in the field of footprint generation has not focused strongly on general methods for assessing the quality of a footprint. Most algorithm authors provide a set of requirements on the types of shape they allow (e.g., the boundary must form a Jordan curve) but judge the quality of the footprint on human intuition alone. Galton [Galton, 2008] presents an exploratory set of findings on a study designed to find the mathematical properties which lead to one footprint being intuitively ‘better’ than another. The results show a strong tendency for people to attempt to optimise the trade-off between minimising the area and minimising the perimeter, but Galton also posits that there may be other factors at play such as sinuosity and cultural cues (e.g., similarity to alphabet characters). Dupenois and Galton [Dupenois and Galton, 2009] present a classification of footprints that is intended to be used to delineate algorithms by the footprint types they produce. The classification does not itself make statements on footprint quality but by delineating the footprints the user can choose the algorithm most likely to produce the appropriate footprint types for their application. An extension and discussion of this work is presented in Chapter 4.

The convex hull¹ has existed as a mathematical construction for many years. However one of the earliest papers in which it is used explicitly to find an appropriate region for a point set was by Jarvis in 1973 [Jarvis, 1973]. While not the most efficient algorithm, Jarvis’ paper identified the need for such work in the field of pattern recognition. Ten years after Jarvis’ work the much cited Edelsbrunner *et al.* [Edelsbrunner et al., 1983] presented the α -shape as one of the first region approximations of a point set that could produce concavities. For a set S the convex hull can be considered to be the intersection of all closed half-planes that contain all the points of S . The α -hull is obtained by using closed discs of radius $1/\alpha$ instead of half-planes; the α -shape is derived from this in a straightforward way. Edelsbrunner *et al.* were also working in the field of pattern recognition, one of their given examples being character recognition from sampled images. Footprints have expanded in use across several fields, being used in image processing (e.g., Edelsbrunner *et al.* [Edelsbrunner et al., 1983; Edelsbrunner and Mücke, 1992; Edelsbrunner, 1992]), pattern recognition (e.g., Gofman [Gofman, 1993]) and GIS (e.g., Alani *et al.* [Alani et al., 2001]). The prolific nature of footprint use has given rise to a large, and increasing, number of different algorithms approaching the problem from a variety of viewpoints: The χ -hull [Duckham et al., 2008], for example, uses the Delaunay triangulation of the dot pattern and successively removes edges longer than a threshold length², whereas the k -nearest neighbours algorithm [Moreira and Santos, 2007] builds a footprint by, from some origin dot, iteratively selecting the next dot on the hull from the set of k nearest dots. The

¹Unless otherwise specified we use convex hull to mean minimum convex hull.

²Assuming removal does not invalidate any of the hull requirements stated by Duckham *et al.* .

two algorithms appear to operate from different base concepts, using ‘destructive’ and ‘constructive’ viewpoints respectively.

There is a distinction to be made between the concepts of a hull and a footprint. Footprint is a general term for any candidate region that has been assigned to a dot pattern. Whereas hull is more specific, Klette and Rosenfeld [Klette and Rosenfeld, 2004] define a hull with three requirements, given a hull operator H and set of subsets \mathbf{S} ³:

$$[\mathbf{H1}] \quad \forall M \in \mathbf{S} \quad M \subseteq H(M)$$

$$[\mathbf{H2}] \quad \forall M_1, M_2 \in \mathbf{S} \quad M_1 \subseteq M_2 \text{ implies } H(M_1) \subseteq H(M_2)$$

$$[\mathbf{H3}] \quad \forall M \in \mathbf{S} \quad H(H(M)) \subseteq H(M)$$

Klette and Rosenfeld note that **[H1]** and **[H3]** imply $H(H(M)) = H(M)$ but they do not replace **[H3]** with this equality as they define variations of the hull (pseudohull and near-hull) that have differing combinations of these requirements (along with a fourth). Many of the footprint algorithms in the literature use hull as part of their naming convention (convex-hull, α -hull, χ -hull, etc.) and not all fit with Klette and Rosenfeld’s strict definition (for example the α -hull need not contain all the dots of a pattern). Further, there tends to be an understanding of a hull as being minimal in some respect. A footprint can extend beyond the the dot pattern, with no dot coinciding with its boundary. Although we note that, in general, the more useful footprints tend to be minimal because, as was mentioned earlier, it is wise to avoid footprints that only serve to make the information the application is interested in less clear. For the purposes of this thesis, and to avoid confusion in general, we will take all hulls as footprints but only use hull to describe footprints created by the few algorithms that already use it as part of their naming convention.

The range of different algorithms that can produce footprints that are not the convex hull tend to have some control parameter. Change in this parameter often leads to change in the ‘spikiness’ of the footprint; varying the concavity it presents. The parameter is a necessity because the patterns can vary so greatly that no method without such a parameter can be said to always produce an ‘appropriate’ footprint. With a control parameter the algorithms can be adjusted until their output fits the context specific definition of correctness. This parameterisation often leads to problems when using the algorithms. For many methods the parameter is an abstract measurement; the best value for which is not immediately obvious to a human user. The α -hull, for example, uses $1/\alpha$ as a radius for discs as part of its process; for a human to give a value of α that will produce a context appropriate footprint is largely trial and error. Even the algorithms with more apparent parameters (e.g., length of edge to remove in χ -hull) still often require some adjustment to find the best region the algorithm can produce for the context. When the set of dots is no longer static there is the added complexity that a parameter value that worked well at one timestep may no longer provide a good fit region at a later timestep. Parameterisation is an important factor when discussing footprint algorithms and in the examination of the change identifiers this thesis introduces, consequently it will be revisited in the chapters

³where each element of \mathbf{S} is a set of points

dealing with these topics (Chapters 4 and 5 respectively).

While the footprints for static phenomena have been the centre of much research there has been far less inquiry into how a region may be maintained over a changing collective phenomenon (its members can move, be added or be removed). Examples in which the phenomenon is subject to change are ubiquitous and cover a range of fields, e.g., tracking animals ([Laube and Purves, 2011]); identifying ship movements, to avoid collisions and traffic; understanding the behaviour of crowds in shopping centres ([Ali and Moulin, 2005]); and tracking the populations of optimisation problems across multiple generations. Additional to these is the emergent field of using sensor networks to provide real-time updates in emergency systems about the state of a current situation, for example the spread of toxic gases, wildfires and floods ([Jiang and Worboys, 2008; Jiang et al., 2011]).

Existing dynamic footprint work has looked primarily at using the convex hull (e.g., [Overmars and Leeuwen, 1981; Chiang and Tamassia, 1992; Basch et al., 1997; Hershberger and Suri, 2003; Guibas, 2004; Guibas et al., 2004]). The convex hull is a strictly defined mathematical construct and is uniquely defined for any single pattern. Footprints considered more generally do not have this unique definition on a pattern. Neither do they often have a short, simple mathematical definition; their construction being the product of their algorithm and can not be reduced into a single statement. The non-uniqueness of the algorithms arises from the parameter (as discussed above) that most algorithms have for controlling some aspect of their formation. Current work uses the strong mathematical properties of the convex hull (e.g., [Chiang and Tamassia, 1992; Basch et al., 1997; Guibas, 2004; Guibas et al., 2004]) to create ‘certificates’ that can be checked at each time step for failure. A certificate is a small, easily-computable property of the footprints relation to the dots. In the event of a failure the footprint is updated, either locally at the point of failure or globally. Footprints in general can not have certificates in the same fashion as there are no definite properties to check⁴. However, fortuitously this vagueness allows us to state that if a footprint is a suitable representation at a timestep τ_i , under most conditions of change it will still be appropriate at τ_{i+1} . Instead of checking the footprint in relation to the dots for suitability we can examine the pattern itself to see if sufficient change has occurred such that the footprint must be updated. This checking requires a method of measuring change on dot patterns that is computationally faster than the footprint algorithm used and a way to assign an appropriate change threshold.

There is a body of work that concerns itself with describing change for spatial-temporal entities like dot patterns and this is examined in Chapter 2. The initial difference in our approach to the existing research is in the definition we give the dot patterns. By taking an individual pattern as a mathematical static abstraction its properties can be examined without the complication of change. The differences between these measurements for two different patterns can be used as measures of how much change needs to occur in each property for one pattern to match the other. Taken across a wide range of properties these measurements provide a value for the total change a pattern must undergo to be equivalent to another. Previous work has looked at dot patterns as part of a continuum

⁴When such properties exist they are prescribed by the context and as such can not be relied upon to exist in all contexts, for example an application that requires all dots exist within the footprint.

over which change must be measured. Both approaches lead to similar measurements (for example both are likely to lead to a measure of change in location), however by looking at the static properties first we are able to use measurements from different fields such as statistical analysis and are able to draw distinctions between different measurement types (see Chapter 5).

1.2. Terminology

The sheer scale of work that uses constructs similar in nature to dot patterns means that there is a high probability of confusion when comparing work by different authors. Before we present the definition of dot patterns that is used for this thesis it is important to note that not all existing work treats dot patterns in the same way and we will clarify the differences when they are relevant.

Dot

A *dot* is an $\langle \text{id}, \text{location} \rangle$ pair. It is a representative data point of any phenomenon that can be assigned a location within a space (whether real-world or abstract). The pairwise nature of a *dot* renders it very simple, since it is only when they are grouped and moving does complexity arise. The term *dot*, over the more common *point*, is used for two reasons. Firstly, a dot may have more data associated with it than just a location, i.e., an identifier. Secondly, to draw a distinction between the dot as a member of a dot pattern and a point within the dot pattern, for example the mean centre (centroid) of a pattern is a point that may not coincide with a dot.

As an addendum to the definition of the dot it should be noted that the work presented in this thesis does not make use of the identity attribute; this is to reduce the number of assumptions made about the raw data and will be discussed in Chapter 3. It is included, however, as a convenient way with which to describe differences between the dot patterns, e.g., a particular dot has moved to the left. Future work may involve the creation of descriptors, and change identifiers, that use the dot identity for applications which can guarantee its availability.

Dot Pattern

A set of dots is called a *dot pattern*. It is fully exhausted by these dots and is a mathematical abstract. Its mathematical nature means it is incapable of change.

Dynamic Dot Pattern

As a dot pattern cannot undergo change we need a structure that uses dot patterns to represent the change of the underlying phenomenon. The *dynamic dot pattern* is a function

mapping from time to dot pattern, as shown in Fig. 1.2. The function maps such that $\forall \tau \in T$, $DP(\tau)$ is the dot pattern representative of the underlying phenomenon at time τ . Rather than use $DP(\tau)$ to refer to a dot pattern at a specific time we use the nomenclature *phase* (for which we borrow the wave notation of ϕ). For example the pattern in Fig. 1.2 at timestep τ_0 is the first phase (ϕ_0) of the dynamic dot pattern and the pattern at τ_1 is the second phase (ϕ_1).

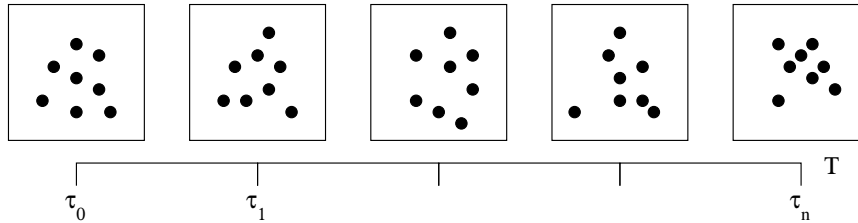


Figure 1.2. Time Domain (T)

The phenomenon that the dynamic dot pattern represents is likely to be changing in a continuous fashion, however the phases are discrete steps. Thus the timesteps are used to provide a discretised time domain for the dynamic dot pattern to map from. There will almost certainly be differences between the timesteps, and therefore between the phases, but for the purposes of the work presented in this thesis we can ignore the ‘in-between’ dot patterns as we are assuming that the *granularity* of the arriving data is appropriate to the application.

For clarity we reiterate that that this thesis is not interested in the change between patterns. Preferring to treat the problem as the difference between two dot patterns. However, as mentioned above, many of the real world examples are of continuant phenomena which do exhibit change. As a result there will be occasions in this thesis where the difference between patterns is discussed in terms of the change between them, conferring a continuant identity on the pattern. This should be treated as a convenient way to visualise the measures and the possible situations in which the differences can occur. It should not be taken as a confirmation that the measures in question can only be used when change has occurred.

Collective

A *collective* is a grouping of objects with some attribute in common⁵. A dot pattern is representative of the locations of the members of a collective at an instant in time. Hence, a collective may have a continuant identity whereas a dot pattern’s identity is given entirely by the locations of its dots and its time value. This means that a dynamic dot pattern for which each phase comes from the same collective is representative of the change the collective undergoes over the given time domain.

⁵Rather trivially (and tautologically) this attribute can be the fact that they have been grouped together

Descriptor

Dot patterns have various mathematical properties that arise from the cardinality and the locations of their member dots; for example the standard deviation from the mean centre. There is an incalculable number of these dot pattern *descriptors* but many will be measuring similar attributes of a pattern in different ways (e.g., bounding box and standard deviation can both be said to be measures of the extent of the pattern). We propose several *classes* of descriptor that are general headings for types of information that can be retrieved from the pattern. Further to the mathematical descriptors these classes can provide qualitative information (e.g., a pattern is large) and such information may be advantageous to a user, particularly in application contexts where response time is important. There are problems faced in using these qualitative statements, primarily as they tend to be context specific, but they do not affect the central preposition of this work so discussion on qualitative assertions are left until the chapter concerning future work.

Change Identifier

The entities that are represented by individual dots in a dot pattern can only change in a three ways: appearance, disappearance and translation⁶ ([Thériault et al., 1999; Huang et al., 2008]). The dot pattern phases within a dynamic dot pattern can, therefore, only differ in the location and the cardinality of their members. The differences can be measured using the descriptors of the dot patterns, and comparison of the extent of these differences is a measure of the change undergone between the phases. Simple entity changes can lead to complex pattern differences and, if change is to be measured in any formal sense, we must be able to measure the difference in such a way that none of the emergent complex behaviours are ignored. Change within a descriptor class represents an emergent complex behaviour from the collective. For example:

Two phases differ in standard deviation in that the deviation at the later time step is greater than in the earlier. Standard deviation is within the class of extent descriptors. We can therefore infer that the collective has undergone the complex behaviour of expansion.

Further to the change identifiers based on descriptors, there are distance measures which may be used to measure the difference between dot patterns. For example, the distance between the mean centres of two patterns.

1.3. Summary of Thesis

⁶The entities could undergo many other types of change, ageing for example, but for our abstraction only the three given apply.

And the Novelties it Presents

The background chapter presents an overview of the existing literature from the fields that frame this thesis. It begins with a brief study on how dot patterns have been represented in traditional Geographic Information Systems (GISs), including an examination of some of the possible data structures that may be used to contain the patterns. The chapter continues with discussions of the literature for the fields of spatio-temporal data, shape description, footprints and footprint algorithms, and dynamic region maintenance.

After the literature review given in Chapter 2 further background is provided by the next two chapters, which discuss dot patterns and footprints (Chapters 3 and 4 respectively). The dot patterns represent the input and footprints the output of the change identifiers, and before discussing change identifiers themselves it is necessary to understand the structures they use. Underpinning the change identifiers is the novel examination of the dot patterns as mathematical abstracts removed from the collective that they represent, by which the mathematical properties (descriptors) of the patterns which describe them can be isolated. Chapter 3 presents a detailed discussion of these descriptors and how those used within this thesis were selected. The descriptors provide a definition of a dot pattern that allows us to examine the myriad types of pattern that exist and, more importantly, compare them. It is this comparison which leads to the concept of change identifiers and the generalised framework in which we use them to measure change across a dynamic dot pattern. The footprints chapter presents a classification of footprint types (first published in [Dupenois and Galton, 2009]) with a discussion on how the classification relates to dynamic dot patterns. The classification is used to show that the footprint algorithms, which tend to be created for generalised fields rather than specific applications, can be paired with applications that best suit the footprint types they produce.

Chapter 5 builds on the foundations laid down by the previous chapters, particularly that of Chapter 3, to formally define the change identifiers; distinguishing between two different types of identifier: one that uses dot pattern descriptors and another that makes use of standard distance measures. The chapter presents the core novelty of the thesis; detailing how the change identifiers can be used as a method by which to measure change in a collective of spatio-temporal entities represented by a dynamic dot pattern. The use of multiple change identifiers is explored within this chapter and methods to combine their measures fairly are discussed. Finally, the change identifiers chapter provides a method for assessing the quality of a change identifier and a change identifier set using the trade-off between time saved and how well the footprint is tracked.

Having given detailed descriptions of the input, output and process components (dot patterns, footprints and change identifiers respectively) of the change identifier framework, Chapter 6 (the methodology chapter) presents the system that combines them. The chapter demonstrates the modular nature of the framework used for the results in this thesis and how it can be employed for testing the identifiers and for real-world applications. The methodology also describes the data structure used for storing the dot patterns and the reasoning behind the choice of format that the dot patterns are expected to arrive

in. The chapter finishes with an examination of some of the ways in which the difference between footprints can be measured, as such measures are important to the way in which the quality of the change identifiers can be assessed.

Since the results span a large number of experiments they include many different (and often large) graphs, consequently the results are separated into their own chapter (Chapter 7). The chapter also provides an explanation for the selection of the dynamic dot patterns and footprint algorithms that are used within the experiments. This differs from the discussion in Chapter 6 which describes the way in which the framework is constructed and the methods by which to use it, whereas the results chapter details the sources used within the experimentation.

The conclusions chapter reiterates the areas in which the change identifiers have performed as expected, as well as those areas in which they can be improved. The conclusions drawn lead to the final section of this thesis (§ 8.3), which concerns itself with the possible ways in which the change identifiers can be taken forward. The future work section begins with an examination on how dot pattern descriptors may be used to define different types of dot pattern and shows some preliminary results using an agglomerative clustering technique to partition the dot pattern classes. The dot pattern classification is followed by a consideration of the use of change identifiers to indicate when the footprint type has changed according to the classification given in Chapter 4. Further to the discussion of footprint types the chapter proposes using change identifiers to dynamically alter the footprint algorithm parameters based on the change the collective is undergoing (as represented by the dynamic dot pattern). Finally the chapter, and the thesis, finishes with some ruminations about where change identifiers may be used and whether or not they could return salient qualitative information.

2. Background

This thesis is situated within, and contributes to, the discussion of two complementary fields. The first is the analysis of spatio-temporal data and the second is the assignment of regions (footprints) to point-based data sets. There are large bodies of work for each and the intersection of these fields has been used in Geographic Information Systems (GISs ¹) as a form of query filtering (e.g., DSAM [Arampatzis et al., 2006]). Most of this work has dealt with data sets that are solely spatial or in which change is glacial in pace (regions of forests, cities, etc.). This thesis focuses on forming a general framework with which to maintain footprints across spatio-temporal data.

To situate this work in the associated research we will provide a quick overview of the nature of dot patterns when considered within GISc and the structures that may be used to contain them, after which approaches to the classification and study of spatio-temporal data will be examined. This will be followed by a general overview of shape descriptions and footprint algorithms. Finally the existing research on dynamic tracking will be looked at. This program of study will provide the necessary background required to further discuss the themes of this thesis.

2.1. GIS and Dot Patterns

O’Sullivan and Unwin [O’Sullivan and Unwin, 2002, ch. 4] give a good description of the treatment of dot patterns (called point patterns) from a geographic standpoint. The chapter begins by noting that point patterns frequently occur in GISs and gives the examples of crime or death hot-spot analysis. The members of a point pattern are termed *events*, and each event represents a single object of interest from the region being studied. O’Sullivan and Unwin also state that a set of events is only a point pattern if it conforms to a number of criteria:

-
1. The pattern should be *mapped on the plane*.
 2. The study area should be *determined objectively*.
 3. The pattern should be an enumeration or *census* of the entities of interest, not a sample.

¹Occasionally there is some confusion as to whether GIS should stand for Geographic Information System or Geographic Information Science. Within this thesis we will be using GIS when we refer to a system and GISc for the science as a whole.

4. There should be *one-to-one correspondence* between objects in the study area and events in the pattern.
5. Event locations must be *proper*. They should not be, for example the centroids of areal units chosen as representative ... They really should represent the point locations of entities that can be sensibly be considered points at the scale of the study.

O'Sullivan and Unwin [O'Sullivan and Unwin, 2002, ch. 4]

Of the requirements only requirement 2 can be generally applied to dot patterns; the wide array of fields in which dot patterns are used means that the other requirements are not always satisfied. For example, requirement 4 assumes a direct mapping between dots and a real-world object and 3 states that a the pattern should not be a sample, however a dot pattern may represent a sampling from a region [Alani et al., 2001] or a set of classification results. The requirements are given despite the differences between dot patterns and point patterns because much of the spatio-temporal literature comes from the GISc field, and we should be aware of the assumptions that the work makes.

O'Sullivan and Unwin examine two approaches to point pattern analysis: point density and point separation. Density measures can be used to show first-order effects while separation is indicative of second-order effects. A first-order effect occurs when the physical location has correlation with the event, for example in a study of the locations of the swans in Hyde Park it is likely that the clustering would occur around the bodies of water. A second-order effect occurs where an event affects the incidence of other events, for example a study of locations of a particular contagious disease would have areas of clustered points as the probability of catching the disease increases with the number of events in the area. The static way in which we examine dot patterns will not be able to draw a distinction between the two types, however the change identifiers may be split along the same delineation. This suggests that a possible avenue for future research is the classification of types of change identifiers and how they can be used to inform a user of the more complex behaviours that the collective is exhibiting; i.e. Is a collective's change representative of a first or second order effect?

2.2. Data Structures

The data structure containing the pattern is important as it can greatly affect the complexity of change measures (e.g., can the extremal points be found in $O(n)$?). Worboys and Duckham [Worboys and Duckham, 2004] provide a useful overview of some of the common structures and their properties.

Grid-based structures are a simple starting point. The underlying concept is the *bucket*, a contiguous memory location, that will contain only points that the grid deems to be

related. The basic grid type is the fixed grid structure, in which the grid partitions the region containing the pattern into equal sized cells and each cell constitutes its own bucket. All points within a specific cell are held in the same place. The problem with this formation occurs when the dot pattern is not uniformly distributed, consequently some cells may be empty while others may be near overflowing. An extension to the fixed grid is the grid file, in which the horizontal and vertical lines making up the cell divisions do not have to be equally spaced. They are placed based on the dot distribution and cells can be divided or amalgamated depending on the amount of free space they contain. The major benefit of the grid file is the ability for it to be easily dynamically updated, however using it to search for specific dots (extremal, median, etc) is not particularly fast. One of the more common uses of grid layouts is in the quadtree, which divides the space up into 4 evenly sized quadrants (buckets) and iterates over each quadrant performing the same space dissection. Fig. 2.1 demonstrates an example of this division and shows the tree that would be used to locate the dot in the space. The quadtree requires both a minimum size for its final quadrants and a bounded space to delineate.

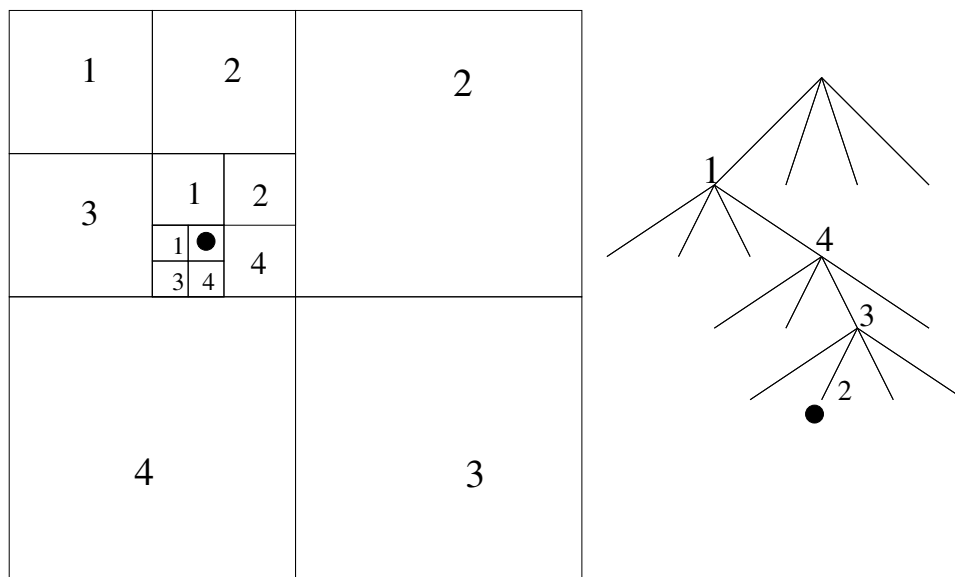
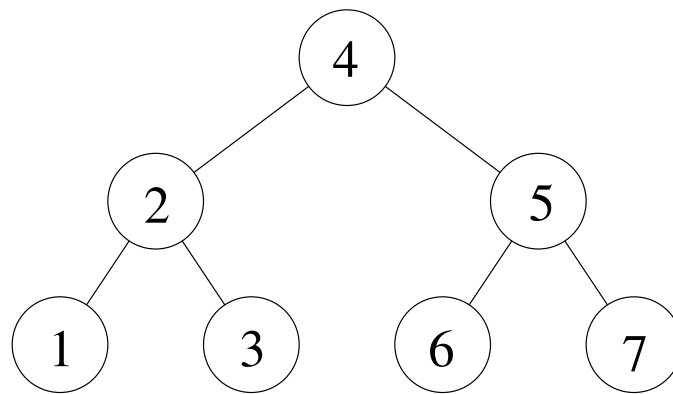


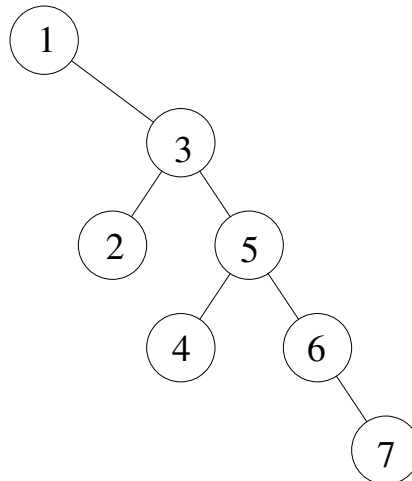
Figure 2.1. An example quadtree division that shows the route taken to find a given dot

Tree structures are quick to both build and search and do not have to mimic the spatial relationships of the dots, the data can be stored in any way that preserves those relationships. Knuth's book on sorting and searching [Knuth, 2007, ch. 6.2.2-3] provides a description of the various methods that can be used to store data in tree structures, the most basic of which is the binary tree. A binary tree has nodes with at most two children and stores the data so that searching on it performs a binary search. The binary tree is fast to search but suffers from difficulties in insertion; it relies on the incoming data to be suitably randomised otherwise it builds unbalanced trees. An unbalanced tree has subtrees from the same node with unequal heights and can result in linear search complexities. Examples of a balanced and unbalanced binary tree can be seen in Fig. 2.2(a) and Fig. 2.2(b) respectively.

To prevent the creation of degenerate trees there have been a number of variations on the



(a) Balanced Binary Tree.



(b) Unbalanced Binary Tree.

Figure 2.2. Examples of the Binary Tree Data Structure

binary tree, further information on these can be found in Knuth [Knuth, 2007], Worboys and Duckham [Worboys and Duckham, 2004], and Berg *et al.* [de Berg et al., 2008]. For the purposes of this thesis we look at the 2-3 B-tree² in which each node of the tree can have, either one data object and two children, or two data objects and three children. This tree is far easier to balance due to a simplified insertion, as demonstrated in Fig. 2.3. At Fig. 2.3(b) the 10 node has space so accepts 15, however at Fig. 2.3(c) it is incapable of accepting 17 so it moves it up to its parent node, as shown in Fig. 2.3(d); as this node now has two elements it must have three children so the bottom-left node is split (Fig. 2.3(e)). This insertion approach gives a *self-balancing* tree so the degenerate branches of a normal binary tree can be avoided.

Guibas and Sedgwick [Guibas and Sedgwick, 1978] showed that the 2-3 B-tree can be refactored into a binary tree called a *red-black* tree³. The ability to mimic the 3-node of a 2-3 B-tree is achieved by having two types of linking edge: red and black, often the nodes will be referred to by the colour of edge that connects to them (i.e. red or black nodes). The black edges are the same as a standard *vertical* node linking in a 2-3 tree. However,

²The source of the B in B-tree is somewhat of a mystery but is commonly attributed to either Bayer (who, along with McCreight, created the B-tree [Bayer and McCreight, 1972]) or Boeing (where Bayer and McCreight were working when they created it).

³They also show how B-trees of higher orders can be converted but for this thesis the description of the 2-3 is sufficient

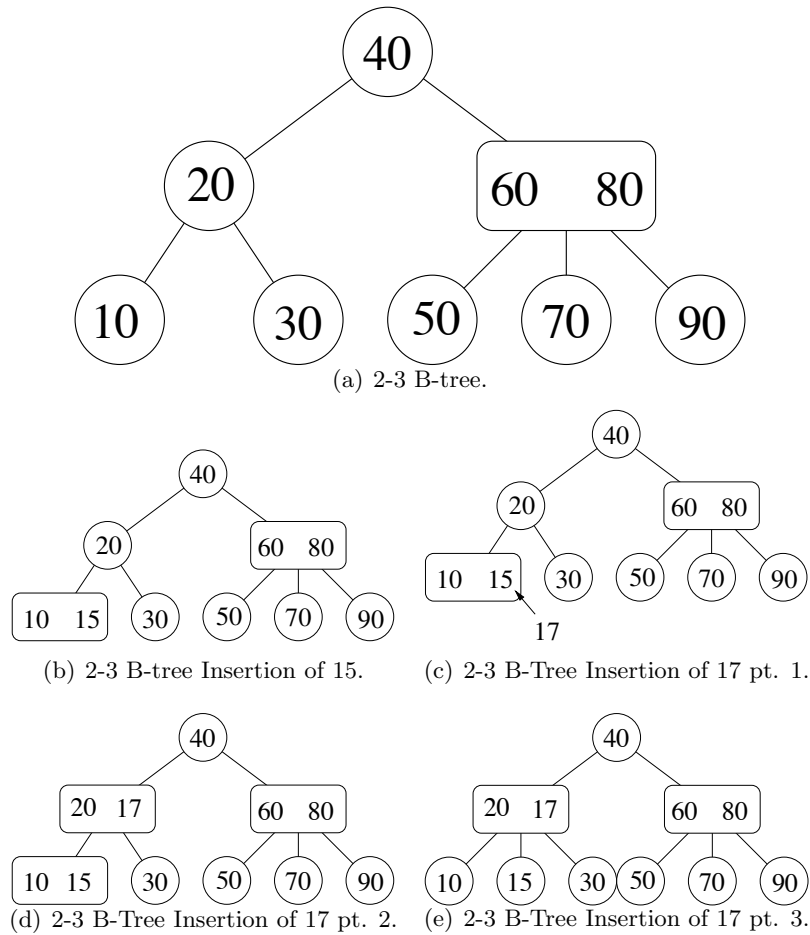


Figure 2.3. Examples of the 2-3 B-tree Data Structure

the red edges are *horizontal* links, that is to say they indicate a link between two nodes that would be concatenated in an equivalent 2-3 structure.

The dot patterns we consider in the thesis are generally in a planar space and are therefore described by two values. The tree structures given above currently only sort on one value, and, while this may be sufficient for the needs of the change identifiers, it is wise to consider ways in which we might better sort the data. Berg *et al.* [de Berg et al., 2008] gives a good treatment of such structures, beginning with the kd-tree. The kd-tree sorts by alternating dimensions, for example for a planar dot pattern the 2d-tree (sometimes called a 2-dimensional kd-tree) splits alternately by the horizontal and the vertical. Fig. 2.4 shows an example in which a dot pattern has been organised into a 2d-tree, the numbers 1–4 represent splits and the letters *a–e* represent dots. The kd-tree is a useful structure as it makes searches within rectangular regions in $O(\sqrt{n} + k)$ time where n is the number of dots and k is the number of dots found in the query. Whether or not we will need to be able to perform such a query with the change identifiers will not be known till we have examined them further in Chapter 5⁴.

The concept behind the general framework is that it remains applicable regardless of application. This generality requires it to be free of too many assumptions about the

⁴Berg *et al.* also describe the range trees which are better suited to rectangular range queries when there are many dots that will be found (i.e. when k is large)

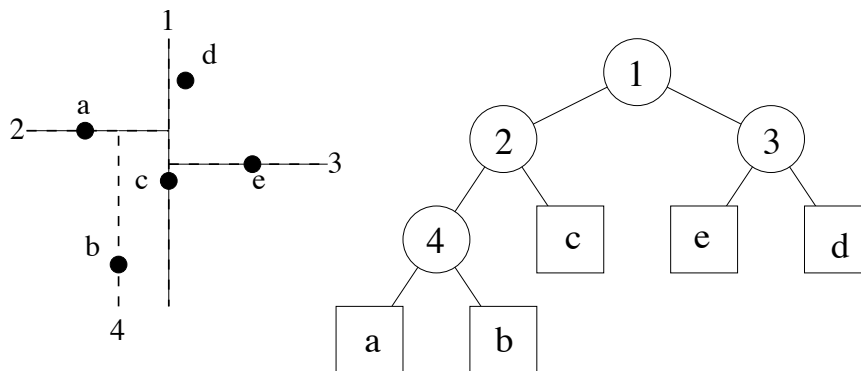


Figure 2.4. A Dot Pattern with Associated 2d-Tree.

data, in particular no assumptions can be made about the format in which the data arrives. Some applications may simply give locations for all dots in each phase whereas others may provide information about only the dots which have moved. Not knowing, or being able to specify, the exact data format means that the data structure will likely have to be rebuilt at each timestep. Some form of balanced tree-structure would make a good preliminary choice as it would be quick to build and easy to search, however without having looked at the possible change identifiers' requirements it is impossible to know which specific structure would be best. Consequently, this will be revisited in the chapter detailing the methodology used for the experimentaion (Chapter 6).

2.3. Spatio-Temporal Data

The existing approaches to spatio-temporal data include both qualitative and quantitative methods. The change identifiers are primarily a quantitative concept but by looking at the qualitative as well as the quantitative we provide a better background for our own research. It should be noted that this is by no means an exhaustive literature review as the field of spatio-temporal research is both extensive and very much alive.

2.3.1. *Qualitative Representation of Change* [Hornsby and Egenhofer]

Hornsby and Egenhofer's paper [Hornsby and Egenhofer, 1997] begins with a definition of objects and a discussion on object identities; drawing attention to the fact that, when the members of a collective can move and change, identity is not always easy to ascertain. This identity ambiguity is of particular importance to their work as the changes that they consider are directly related to an object's identity. The work provides a visual representation for describing different change types as a product of identity operations on objects and an extension of this to describe the change in object composites. Identity and its relation to dot patterns will be addressed in Chapter 3; for now it suffices to state that a dot patterns identity is inextricable from its component dots and the timestep for which it was created. The work in this thesis is primarily concerned with quantitative descriptions of change in which distance is important, as opposed to the topological approach. One

possible avenue for further research may be to look at using change identifiers to indicate topological changes.

2.3.2. A Spatio-Temporal Taxonomy for the Representation of Spatial Set Behaviours [Thériault *et al.*]

Thériault *et al.* [Thériault *et al.*, 1999] present a taxonomy for describing evolutions of sets of geographical entities (SGEs). The authors make clear that there are two simplifications on which their model rests: Firstly that a geographical entity exists in geographical space; and secondly that the size and orientation of the entities is negligible compared to the inter-entity relationships. These simplifications restrict the behaviours that they examine to changes in a point-based framework. The point-nature of an SGE means that, for the purposes of this thesis, we can treat it as largely identical to a dot pattern. Thériault *et al.* are primarily concerned with the manner in which SGEs can change, and to do so they need properties on which the change can be measured. It is these properties that we are interested in examining as classifiers for describing patterns.

Before discussing the measurements that they propose we shall draw attention to an interesting observation that the authors make. There are two different but complementary method types for examining the evolution of entities.

1. Deductive methods; methods based on representations of the spatial entities and their relationships. They note that these methods tend to be based on Euclidean space and/or topological descriptions of space and time. Thus deductive methods form a qualitative approach.
2. Inductive methods; methods based on data analysis often using spatial statistical methods to study properties and distributions of the entities. In contrast to deductive, inductive methods are a quantitative approach.

While laying the groundwork for the taxonomy the authors reason about descriptions of possible complex set behaviour arising from simple entity behaviours, however, like the work presented in this thesis, the majority of Thériault *et al.* 's paper takes a low-level approach; one that is concerned with the measurable changes that take place in the set.

Further to the point-nature they use for the entities of the sets they assign an intensity to each entity indicating the importance the point has within the given context. Such properties have been avoided in the inquiry presented by this thesis for the fact that we allow for dot patterns that arise from contexts in which intensity is not a meaningful concept, for example, a graph of classification data where each data-point is a dot in our pattern.

The taxonomy itself arises from the consideration that the entities can only exhibit very simple changes: appearance, disappearance, translation and intensity change. These changes are manifested within the set as four components of possible change types, although they observe that these four do not necessarily constitute an exhaustive list. It is

these components which bear such close resemblance to our descriptors.

1. Territorial/Spatial Extent

Thériault *et al.* use the phrase territorial extent; however, because of the above-mentioned concerns about applications outside of GISc it makes more sense for this thesis to refer to spatial extent. The extent of the SGE can change via expansion or contraction. To measure the extent the authors suggest using the convex hull of the set. This is assigning a footprint to the set and taking a measurement from that footprint. This topic is re-visited when dot pattern descriptors are outlined, as using a surrogate for the dot pattern has some interesting connotations for the types of measurement that can be made. Extent is clearly a property that will need to be accounted for within the set of descriptors.

2. Spatial Distribution

This component of Thériault *et al.*'s change types includes several sub-properties. Of the four properties of spatial distribution all but the last have direct correlation with some of the descriptors we introduce in Chapter 3.

- a) Centre of Gravity (CG): Indicating the equilibrium point of the distribution. The CG takes into account the intensity of their dots but aside from this difference such a measure is used within the framework presented by this thesis as a descriptor of position.
- b) The Standard Distance (SD): The standard deviation from the CG. Like centre of gravity this quantitative measure has a descriptor counterpart; standard deviation can be used as a measure of the extent of a dot pattern.
- c) The Orientation: Using principal axis extraction the direction in which the set can be said to face can be found. Orientation is one of the classes of descriptors used by this thesis and the principal component's (axis') gradient is one of the descriptors within the orientation class. Thériault *et al.* use both of the returned vectors of the principal axis extraction⁵ to describe the maximum and minimum dispersion of the set. These extrema are then used by the fourth and final property of spatial distribution.
- d) Ellipse of Dispersion: Defined by the two vectors given by the principal axis extraction, this ellipse is used to monitor overall dispersion and density. Thériault *et al.* note that this is an extent measure that, unlike the convex hull, is not sensitive to outliers within the set.

3. Spatial Pattern

Spatial pattern refers to distinguishing between random, clustered and regular patterns. How best to measure this is a topic that we will consider in Chapter 3, for now it can be thought of in terms of degrees of homogeneity across the pattern.

4. Spatial Autocorrelation

⁵Principal axis extraction, or Principal Component Analysis (PCA), in 2-dimensions returns two perpendicular eigenvectors, the one with the highest eigenvalue is the principal axis

The final component concerns the relationships between the entities of the set. It is used to measure how likely entities are to have the same or similar attributes to nearby neighbours. As the patterns we consider have no attributes beyond their location this can be dealt with by the same kind of approach that might be performed for clustering analysis.

The taxonomy presented by Thériault *et al.* shows a strong resemblance to the kind of analysis we wish to perform on dot patterns and certainly has a bearing on the descriptors we have chosen. However it is concerned with changes that can occur to the set and has not been created to describe the pattern as it is at any timestep. Part of our research is to see if other ways of identifying change arise from examining the properties inherent in a static pattern. We have examined Thériault *et al.*'s paper in more detail than we have done for many of the other works as it is so close in nature to the aspect that concerns our research.

2.3.3. *Granularity in Change Over Time* [Stell]

Stell [Stell, 2003] examines the levels of detail at which entities undergoing change can be modeled. Within this thesis it is assumed that the granularity of the phases of the dynamic dot pattern is appropriate to the application context, and hence is not explored further. However when considering granularity Stell defines the nature of the *time domain* over which the data exists, and this is directly applicable to our work. A time domain is a finite set T such that for $\tau, \tau', \tau'' \in T$:

$$\tau \prec \tau' =_{def} \tau < \tau' \wedge \neg \exists \tau'' (\tau < \tau'' < \tau')$$

In effect $x \prec y$ asserts that x and y are adjacent and that x precedes y . We can use this same requirement for the timesteps within the time domains used by our dynamic dot patterns. As each timestep has an associated phase of the dynamic dot pattern it can be stated that a phase ϕ' precedes a phase ϕ'' ($\phi' \prec \phi''$) iff the relationship is also true for their associated timesteps.

$$\phi' \prec \phi'' \iff \tau' \prec \tau''$$

Note that we allow for a ‘live’ system, in which T may be an infinitely large set⁶. To clarify what is meant by a ‘live’ system envision an application monitoring a herd of cows in real time. The change identifier framework sits in an idle state awaiting herd data; processing the dot patterns as they arrive.

Given a time domain T , Stell defines a *dynamic set* over T as a set of objects that evolve over time. He uses dynamic sets to explore the concept of support amongst entities; if entity a supports b then a exists/ed prior to b and the existence of a is necessary for the existence of b . The support relation and the time domain allow Stell to produce

⁶Although, for obvious reasons, all our tests have a finite size and it is possible to make the argument that no real world example is infinite. Despite this, when using the framework it is not known a priori when, if ever, the data will cease to arrive, so assuming infinity leads to a more general framework.

graphs to model the relations between entities over time, and simplify these graphs by the amalgamation of entities or by omission of time steps.

The overview given of Stell's paper is by no means an exhaustive examination of its content. It suffices for this thesis to note the added formalisation it provides for the time domains that support the dynamic dot patterns.

2.3.4. *Finding REMO - detecting relative motion patterns in geospatial lifelines* [Laube *et al.*]

Laube *et al.* [Laube *et al.*, 2004] is one of the core works concerning motion analysis. Laube *et al.* define the concept of *geospatial lifelines* as a series of observations on Moving Point Objects (MPOs) that are a triple of $\langle \text{id}, \text{location}, \text{time} \rangle$; an individual dot within a specific phase of a dynamic dot pattern can be described by the same tuple. The main research interest of this paper is identifying flocking behaviours (and other motion patterns⁷) of MPOs by analysis of spatially constrained RElative MOtions (REMOs). These movement patterns are the same complex behaviours that we have discussed previously as the behaviours of the collective which the dynamic dot pattern represents. One of the issues faced by REMO analysis is the complexity of some of the pattern identification algorithms. It is possible that the change identifiers can provide information on when a movement pattern type has occurred or changed thereby reducing the number of times which the analysis needs to be run.

2.3.5. *Event-oriented approaches to geographic phenomena* [Worboys]

Worboys [Worboys, 2005] gives an alternative view of the concepts behind spatio-temporal data analysis. Worboys states that instead of looking at entities at each timestep the phenomena should be viewed as sets of events. The events in this context are 'happenings' unlike the events as objects described by O'Sullivan and Unwin [O'Sullivan and Unwin, 2002]. While this approach is outside the remit of this thesis it is worth noting that it is only convention which sees us mapping the entities of a collective in our dot patterns.

Worboys, when discussing existing ways of looking at object change, gives a description of the problems with dot pattern identity. Like Hornsby and Egenhofer [Hornsby and Egenhofer, 1997], Worboys notes that identity is not necessarily a fixed concept. Both Worboys and Hornsby & Egenhofer observe that there is an issue when assigning identity to a dot pattern when the membership of the pattern is subject to change⁸. This uncertainty is one of the reasons that the dot patterns, as proposed by this thesis, are independent of a personal identity and why the change identifiers function whether or not identity information is known for the dots.

⁷Note that a motion pattern is different to a dot pattern. To avoid confusion we shall endeavour to avoid using pattern without an appropriate classifier.

⁸An analogy of the identity issue is provided by the philosophical problem presented in the Ship of Theseus

2.3.6. *Reporting Flock Patterns* [Benkert *et al.*]

Benkert *et al.* [Benkert et al., 2007] provide a set of algorithms for identifying movement patterns of the same type defined by Laube *et al.* [Laube et al., 2004]. The algorithms proposed make use of the skip-quadtrees, a data structure that is an extension of a standard region-quadtrees⁹ but that only stores boxes/buckets as leaf nodes if they contain at least one point. This reduces the space required to $O(dn)$ (Where d is the dimensionality of the space the data exists within) at a cost of increasing the time to check if a bucket is empty.

In particular Benkert *et al.* focus on identifying when a group of entities constitutes a flock. They provide two flock definitions, both of which rely on being able to find an encompassing disc of the entities; given that it is a region that represents a dot pattern, we can consider this disc to be a footprint. The definitions differ in the granularity of time intervals at which they require a disc to contain all points; the first requires a disc to be found at all time points whereas the second only requires such a disc at the discrete timesteps¹⁰. They present two algorithms using the skip-quadtrees and these definitions in concert to detect flocks. Within these algorithms they allow for an approximation in their disc radius. There is a conceptual similarity with this thesis in the observation that the footprint need not be exact. It should be noted that theirs is always an over-estimation; Benkert *et al.*'s approximation is performed by adding a positive uncertainty to the radius of the disc.

2.3.7. *Designing visual analytics methods for massive collections of movement data* [Andrienko and Andrienko]

Andrienko and Andrienko's 2007 paper [Andrienko and Andrienko, 2007] presents a set of tools for the visual analysis of large amounts of movement data (so large that it exceeds available computational memory). Within discussion of this topic they raise certain concerns which are pertinent to the research presented in this thesis. To begin with they define movement data as a function that matches a tuple of $\langle \text{entity, time moment} \rangle$ to a point in space (like Laube *et al.* [Laube et al., 2004]). Bearing a resemblance to the approach within this thesis is Andrienko and Andrienko's decision to examine the information that can be gathered from a simplified starting point. They, however, forgo looking at the static pattern and focus on what they call the *derivative movement characteristics* (speed, duration, etc.). These derivative movement characteristics combined over time lead to *individual movement behaviours* (IMBs) which are the more complex movement types indicative of the underlying collective's entity behaviours. These definitions lead to the concept of *momentary collective behaviours* (MCBs) which are the movement characteristics of the set of entities at a time moment. MCBs are close in nature to our descriptors, looking at the spatial and statistical distributions of the entities and their

⁹The same as a standard quadtree but the hyphenated name is presumably to make the distinction between skip and region clearer.

¹⁰They also demonstrate that, if the entities within a set move along straight line segments between consecutive positions, then the flocks produced by both definitions are equivalent

characteristics. Over time MCBs give rise to the *dynamic collective behaviour* (DCB); the overall description of the set's complex behaviour.

Andrienko and Andrienko focus more on the classification of these behaviours and how to identify them than on an in-depth examination of the change the sets may undergo. The reason such a detailed description of their definitions has been given is to provide background to their observation of the influences that can affect the behaviours. These influences are not something that affect the construction of the change identifiers but are of absolute importance to any application that may use them:

- Properties of space (e.g., altitude, accesibility, function, etc.)
- Properties of time (e.g., temporal cycles, duration of daylight, holidays, etc.)
- Properties of entities (e.g., age, movement method, purpose, etc.)
- Various affecting phenomena (e.g., climate, sport, culture, etc.)

Should an application's users be fully aware of these factors it is likely that they will be able to choose change identifiers that check for change in ways that are appropriate to the context.

Further to their discussion on factors that can affect the data, Andrienko and Andrienko note that correlation of two types can affect the DCBs; that by *influence* and that by *structure*. Influence correlation is when one characteristic directly affects another whereas structure correlation occurs when two or more characteristics are combined to form a new more complex characteristic. We examine how correlation affects the dot pattern descriptors (and therefore the change identifiers) in Chapter 3.

While visualisation is not the core of this thesis the change identifiers could add to Andrienko and Andrienko's work by providing a concrete examination of how the patterns can be assessed.

2.3.8. *Towards a Taxonomy of Movement Patterns* [Dodge et al.]

Dodge *et al.* [Dodge et al., 2008] provide a formalised approach to describing the movement of patterns. The authors define the patterns for which they supply a taxonomy as consisting of *Moving Point Objects* (MPOs), which are dimensionless entities, similar to our dots but with movement data associated. They also provide a formalisation of movement that uses three groups of *primitive parameters* and their derivatives. Dodge *et al.*'s full classification is extensive and there is not the space to describe it here in detail, instead we will discuss the fashion in which they have sectioned its levels. The movement patterns are split into *generic* and *behavioural* patterns, generic being the most widely applicable term for a movement type, for example periodicity is exhibited by any MPOs that have periods of movement punctuated by periods of largely static behaviour. Whereas behavioural patterns are far more specific movement types such as migration or fighting. The generic patterns are split into compound and primitive types, primitive for

patterns where only a single movement parameter changes and compound when more than one change occurs. All the given pattern types can apply to individuals or groups. As with previous work focusing on the complex movement behaviours, Dodge *et al.*'s work does not directly affect the change identifiers. It is important, however, to have a good understanding of the behaviours that underly change so that we can be certain that no particular form of change is missed by the identifiers.

2.3.9. *Modelling Herds and Their Evolvments from Trajectory Data* [Huang *et al.*]

Huang *et al.* [Huang et al., 2008] present a set of four evolutions that a herd can undergo: expansion, joining, shrinking and leaving. These evolutions do little to describe the pattern at any single time, however, instead describing the changes in state that the herd has undergone. This is used, primarily, to provide a way to define the identity of a changing herd. One comment they make that is of particular relevance is that quantitative measures can become qualitative if a significant change has occurred. This leads to the problems inherent in trying to describe where the boundaries of significant change are, but may indicate a way in which change identifiers can be best used to provide qualitative information.

2.3.10. *A taxonomy of collective phenomena* [Wood and Galton] and *Detecting and Identifying Collective Phenomena within Movement Data* [Wood]

Wood and Galton [Wood and Galton, 2009] build on previous taxonomies ([Andrienko and Andrienko, 2007; Dodge et al., 2008]) so that the concepts behind a collective are tightly defined. The definition of a collective they provide involves six observations, given these observations they then have a concrete base from which to form the criteria for their classification. Wood [Wood, 2011] extends and uses this classification to identify collectives from within spatio-temporal data. Dynamic dot patterns can be seen as an abstraction of a collective so, while the criteria do not directly apply to dot patterns, it seems prudent to examine them so that we can be confident that no important information is left out by our abstraction. Wood and Galton provide a set of considerations that are used by their classification, but going over each in detail would be irrelevant for this thesis. Instead only those spatial aspects which relate to dot patterns will be covered. Before looking at these considerations we note that, like Worboys [Worboys, 2005] and Hornsby & Egenhofer [Hornsby and Egenhofer, 1997], Wood and Galton discuss whether a collective maintains identity if the cardinality or identity of its members changes.

Location

By convention the location of a dot pattern tends to be taken as a point, often the centroid (mean position) of the pattern (the Centre of Gravity used by Thériault *et al.* [Thériault

et al., 1999]). The collective definition presented by Wood and Galton is different in that it treats the location as an area or volume. For example the location of a class can be said to be the classroom and, from the level of granularity of the class itself, this constitutes a real 3-dimensional space. They suggest that one way of finding this location could be by aggregating the footprints of the collective at each timestep of its existence.

The location classifier is interested in distinguishing between collectives by the fashion in which they change their location. This thesis is more concerned with being able to classify between different types of dot pattern without necessarily knowing the methods by which they might change.

Coherence

Coherence as it appears in [Wood and Galton, 2009] relates to the exhibited behaviour(s) of a collective and can arise from two main sources: *cause* and *purpose*. Causes are split into external and internal sets; the example given by Wood and Galton of an external cause is of Earth's gravity causing raindrops to fall as a collective, and the example they provide of an internal cause is the mutual gravitational pull of a star cluster maintaining the collective of stars. Purposive collectives maintain their collective nature via some goal, again this can be an internal goal assigned by members of the collective or a purpose placed on the collective by some external agent. Wood extends the discussion on coherence in [Wood, 2011] by defining some coherence criteria, of which a group of individuals must satisfy at least one to be considered a collective. It is these coherence criteria which allow Wood to be able to identify spatial collectives from spatio-temporal data.

The change identifiers as used within this thesis are not concerned with the coherence of the dynamic dot pattern. However further work could examine the elements of coherence exhibited by dynamic dot patterns to see if new change identifiers are suggested.

As was defined in the introduction, neither a dot pattern nor a dynamic dot pattern is a collective; the collective may be a pod of whales but the dot pattern representing this is a snapshot of whale positions at an individual moment and the dynamic dot pattern is a sequence of the dot patterns. The classification allows us to provide further distinction between a collective and its representative pattern(s): The collective classification allows for change within its description at different time steps; it can cover a broad time period with complex definitions covering various phases in a collective's life span. For a pattern we are interested in describing it as it is (or was) at the time the information was recorded and how it may differ in type from another pattern. This allows the question of identity for a dynamic dot pattern to be avoided; it can be assumed that phases of a dynamic dot pattern are from the same collective in that they are all in the collective for which the dynamic dot pattern was created to represent. Tracking a constant collective identity over real world entities is more complicated as issues of changing membership arise (also discussed by [Hornsby and Egenhofer, 1997; Worboys, 2005]).

As with the taxonomies provided by Andrienko & Andrienko [Andrienko and Andrienko,

2007] and Dodge *et al.* [Dodge et al., 2008], Wood and Galton’s classification describes the high-level behaviours that may be exhibited by the entities that underly a dynamic dot pattern. It is possible that our change identifiers will indicate these behaviours and the changes between them. Even if this is not possible we would be remiss not to try and understand the possible reasons for the change measured by the identifiers and such analysis may also inform the creation of new identifier types.

2.3.11. *A Graph Model for Spatio-temporal Evolution* [Del Mondo *et al.*]

Del Mondo *et al.* [Del Mondo et al., 2010] discuss the shortcomings in using only space-time paths to describe the events, processes and changes an entity can undergo. They propose using a graph model to map the complex networks that can be formed. To this end they note three relation types that must be modelled:

Spatial – Relation between two entities at the same time

Spatio-temporal – Relation between spaces occupied by entities at differing times.

Filiation – How entities at distinct times relate to each other (i.e. descent or transmission).

Such a model provides a view of a group of entities in a space that does not lose information about their relationships and allows route tracking of entities through different states.

The work is interesting as an examination of visualising change and the collectives underlying dynamic dot patterns can certainly be drawn in such network graphs. The change identifiers may provide information about salient timesteps at which the network graph could (or should) be updated so that for large, and long, dynamic dot patterns the network graph would remain a manageable size.

2.3.12. *How fast is a cow? Cross-Scale Analysis of Movement Data* [Laube and Purves]

Laube and Purves [Laube and Purves, 2011] is an unusual paper. Whereas the other works by Laube mentioned here are examples of movement analysis, this paper looks at the concepts and problems inherent in analysing movement data, akin to the influences in Andrienko and Andrienko’s paper [Andrienko and Andrienko, 2007]. Most of the concerns they raise are not directly related to the general framework we propose as they are application-dependent. However, some are relevant to the test data used for this thesis and some will apply to any application that uses the change identifiers. While there is not the space to discuss each point in detail, a general overview will be given of their concerns (leaving the amusingly alliterative appellations applied by Laube and Purves intact).

Granularity Grief: Any measured parameter (like change identifiers) will be greatly

affected by the granularity of the time domain. The sampling rate at which the data is provided can result in lost or misleading information. This point is similar to Andrienko and Andrienko's *properties of time* influence. It also bears a relation to Stell's consideration of granularity modelling [Stell, 2003].

Slippery Spaces: The space in which the entities exist may not be unconstrained Euclidean space, for example a city has paths, roads, buildings and pre-determined crossing points. This point is similar to Andrienko and Andrienko's *properties of space* influence.

Delusive Dwarfs: Scaling and sampling may provide faulty data. While computationally easier to process, small data sets do not necessarily indicate the behaviours of larger sets. This is something that must be considered when drawing any conclusions from the real-world test data and the generated test patterns used in this thesis.

Baffling Bias: The data source may be biased. For example some types of people are more likely to allow themselves to be tracked than others. Such people may have movement patterns dissimilar to others. Bias bears a similarity to Andrienko and Andrienko's *properties of entities* influence and their *affecting phenomena*; specifically their observation about differing cultures.

Cast-off Context: Often work in the spatio-temporal and GISc fields looks only at the entities in relation to each other and ignores the geographical context in which they are positioned.

Sinful Simulations: When making test data sets it can be difficult to find the appropriate balance between randomness and realistic movement. This is obviously a concern for us within this thesis and we look at the source of the dynamic dot patterns used for testing in the chapters on dot patterns and change identifiers (Chapters 3 and 5 respectively).

2.3.13. Others

There was much work that focuses on spatio-temporal data that was read as background for this thesis has not been detailed above, as it did not add greatly to the discussion of change identifiers. However, any work read will certainly have affected our approach, so should reader wish to pursue further reading they could start with: [Egenhofer and Franzosa, 1991; Bogaert et al., 2006; Bennett et al., 2008; Delafontaine et al., 2011; Laube et al., 2011].

2.4. Shape

To find descriptors for a pattern we must accept that the pattern is an entity, possessing properties emerging from its component dots but not inherent within any individual dot.

Intuitively this can be done by considering a region as a surrogate for the pattern, and this assumption brings with it methods for measuring the descriptors (e.g., area as a measure of extent). We consider any region that describes a dot pattern as a footprint of the pattern. The fact that assigning a region or footprint is so intuitive is almost certainly due to Gestalt perception but it does lead to the question: Is any region more ‘correct’ than another if they are equally intuitive? Before looking at the footprints, some of the ways in which shape itself has been considered are explored. Not only will this provide a good background on how footprint analysis might be approached when change identifier assessment is discussed (Chapters 5 and 6) but some of the methods may apply to dot patterns without the need for a region assignment.

2.4.1. *Qualitative Spatial Change* [Galton]

Galton [Galton, 2000, ch. 4.7.3] provides a comprehensive overview of the possible attributes and relations of spatial regions; specifically: dimension, connectivity, position, location, orientation, size and shape (concerning spatial attributes like sinuosity).

Dimension

Galton splits dimension into two types: strict and apparent. Strict dimension is the classical approach in which an object classified as d -dimensional has no extension into any dimension greater than d . Apparent dimension is a product of having differing levels of granularity with which examine the object. A road is generally considered to be a 1-dimensional line on most maps, a 2-dimensional surface for most users and a 3-dimensional object for a road builder who has to be concerned with the depth as well as breadth and length. Dimensionality of a dot pattern works best when considered as apparent because it is always possible to draw a 1-dimensional curve through the dots. In 2-dimensions, for example, it may be of interest how collinear the arrangement of the dots is.

Connectivity

Connectivity is not a directly applicable term to dot patterns as the dots are, by their nature, disconnected. However it is certainly true that sometimes the dot patterns have areas that appear distinct from others. Fig. 2.5 shows a dot pattern in which part a is separated from part b by a distance that is substantial compared to the inter-dot differences within each part (we call each part a *component* of the dot pattern). Such dot patterns could be said to be disconnected and identifying such separations is intuitively important.

Position

Position is perhaps the most immediately obvious facet of information that can be gained from a dot pattern, as a pattern is represented within spatial dimensions. Galton notes that

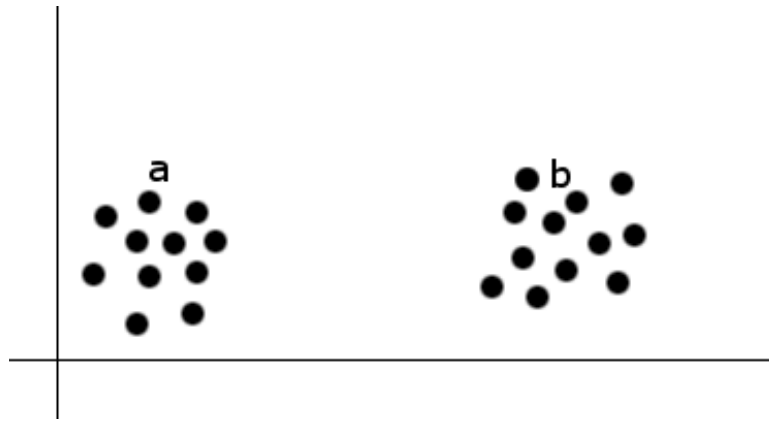


Figure 2.5. A single dot pattern showing disconnected components.

the problem with position is choosing the region of space from which to base the location reference system. As such position is split into two parts: location and orientation. For the purposes of this paper however this is not a relevant concern and Galton's definition of location will not be described in detail; suffice to say it discusses using different reference systems from which to find a 'target' object.

Orientation

Orientation is the direction in which the object can be said to face, or point. Galton describes using a directed line as a 'reference axis' such that the orientation can be found by the direction the axis points. With an axis from which to find the direction the difficulty becomes one of ascertaining how best to specify the direction; for which the book provides several approaches, both quantitative and qualitative. The quantitative methods are variations on units for measuring the line angle and are, as a result, straightforward. The qualitative methods are somewhat more interesting as they can be envisioned in two ways. Firstly by describing the direction in which the axis points to (e.g., north, south, east, west, up, down, left, right, back, forward). And secondly by describing the direction as an observer looking at the object by which sections of the object are visible (the different faces of a cube is the example given in the book). As previously mentioned, this thesis does not focus on the qualitative description of dot patterns, but Galton's quantitative methods will be looked at in greater detail when we discuss descriptor measurement methods (Chapter 3).

Size

Size is a cognitively obvious description of shape. However the fashion in which it is measured is a topic worthy of discussion. Area and volume, for example, are commonly used and often sensible measures but, as Galton points out, they reduce a 2-dimensional and 3-dimensional quantity respectively to a single value. This single value means that information is immediately lost as two different objects of differing dimensions (height/width/breadth) can produce the same result.

The book also describes angular extent as a measure of size but as this requires an observer it does not apply to the context in which we are examining dot patterns.

Shape

Galton uses this section to discuss various ways in which shape can be described including discussions on convex and concave, curvature, symmetry, and the use of natural language. None of these are applicable to the description of a dot pattern but may have use in describing footprints; one of the suggested areas for further work is to produce a more extensive taxonomy for footprint classification so that they may be compared more accurately.

2.4.2. Others

Galton's book is not the only text that details the properties of shape and more quantitative treatments can be found in the field of computer vision such as the work by Žunić and Rosin (e.g., [Rosin, 2000; Žunić and Rosin, 2002, 2003]). However much of this quantitative work requires knowledge of the shapes' boundaries and angles, which is information that requires the footprint to have been computed. Galton's work suffices to give a general overview of the properties of shape that we can use to examine the properties of a dot pattern.

2.5. Footprints

There is a fairly large body of work about the generation of footprints, publications from as early as 1973 ([Jarvis, 1973]) presenting a variety of different algorithms to create representational shapes from dot patterns. Amongst this work there are surprisingly few that examine the footprints created in a comparative fashion. Also conspicuous by its absence is a systematic approach to determining the quality of the produced footprint. Galton [Galton, 2008] makes significant inroads in to both determining how 'good' a footprint is and why this is difficult to judge.

A discussion of the footprint algorithms should probably begin with one of the first to give an efficient algorithm for its computation in 1973. Jarvis [Jarvis, 1973] presented an algorithm, since called the 'Jarvis March', to generate the convex hull of a dot pattern. The convex hull is almost a base level of footprint, its algorithms are generally easily computable and it has distinct mathematical properties. Importantly the convex hull is unique for any particular dot pattern.

The convex hull is not without its problems as a representation.

The point set given in Fig. 2.6(a) could reasonably be interpreted as forming a 'C' shape. The cavity that dictates this shape may be important for the application context (e.g., reconstructing text from samples of a document image) and is lost when the footprint is

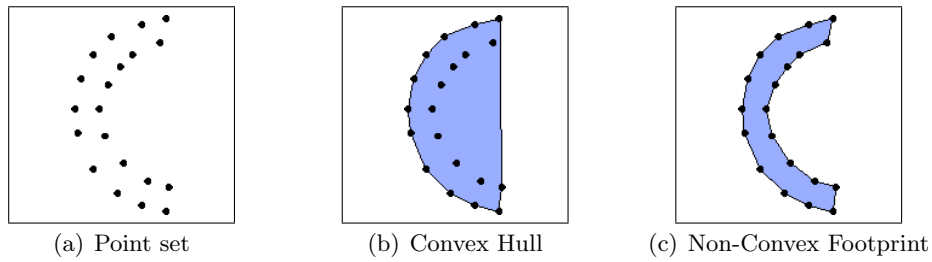


Figure 2.6. When a convex hull is inappropriate

the convex hull Fig. 2.6(b). For the given application Fig. 2.6(c) is a better approximation of the underlying data. An algorithm capable of reaching a better fit representation is a non-trivial problem and one of the earliest, and much-referenced, papers on the subject is by Edelsbrunner *et al.* [Edelsbrunner et al., 1983]. The method produces straight-line graphs called α -shapes, obtained from a generalisation of the convex hull. For a pattern DP the convex hull can be considered to be the intersection of all closed half-planes that contain all the dots of DP . Taking a half-plane to be a closed disc of infinite radius, an α -hull can be defined as the intersection of all closed discs with radius $1/\alpha$ that contain all the points of DP . Using a radius of $1/\alpha$ allows the convex hull to be produced when the arcs are sufficiently straight. If we assume that if $\alpha = 0$ then $1/\alpha = \infty$ ¹¹ and that an arc with an infinite length radius is a line, we can guarantee production of the convex hull when $\alpha = 0$. These assumptions are integrated into the description of the α -hull using the idea of a *generalized disc*. A *generalized disc of radius $1/\alpha$* is defined as a disc of radius $1/\alpha$ if $\alpha > 0$, a halfplane if $\alpha = 0$ and the complement of a disc of radius $-1/\alpha$ if $\alpha < 0$, the α -hull, then, is the intersection of all closed generalized discs of radius $1/\alpha$ that contain all the points of DP .

Before the α -shape can be defined some properties of the hulls need to be noted. A dot d from the pattern DP is α -extreme if there exists a closed generalized disc of radius $1/\alpha$ such that d lies on its boundary and it contains all the points of DP . If two α -extreme points can share the same generalized disc then they are said to be α -neighbours. The α -hull is the intersection of these discs (Fig. 2.7(a)). The α -shape is the straight line graph with vertices at α -extreme points and edges connecting the α -neighbours (Fig. 2.7(b)).

The positive α -shape (where $\alpha > 0$) is clearly a footprint, notable in that it tends to look like an approximation of convex hull save that it is possible for it to not contain all the points Fig. 2.7(b). However the negative α -shape (where $\alpha < 0$) produces far more interesting results as shown in Fig. 2.8(b).

This is one of the earliest steps toward an algorithm that is cognitively more ‘appropriate’ for the dot pattern than the convex hull. There is one more facet introduced by this paper that is of general interest when considering footprints. They make note that as α changes there are only a finite number different α -shapes that can appear¹². These finite shapes are bounded by the convex hull at one extreme and the ‘null’ footprint in which

¹¹ $\alpha \rightarrow 0, 1/\alpha \rightarrow \infty$

¹²Although there can be infinite α -hulls with differing curvature, the α -shapes have straight lines joining the vertices

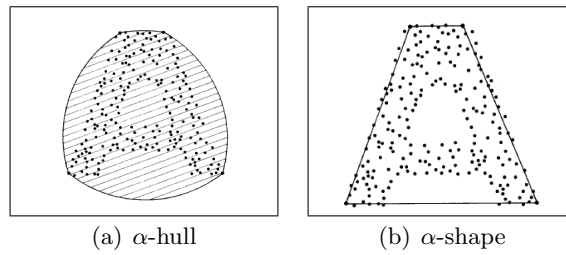


Figure 2.7. Images show the difference between α -hull and α -shape. Image from [Edelsbrunner et al., 1983]

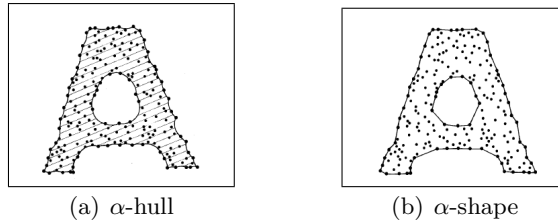


Figure 2.8. Negative α -hull and α -shape. Image from [Edelsbrunner et al., 1983]

no dots are connected¹³. This range of shapes is called the *shape spectrum* ($SP(DP)$), and by generalising their algorithm the $SP(DP)$ can be found in $O(n \log n)$ time. The examination of the literature in this section will concentrate on this type of analysis, however, aside from the papers presented here, it should be noted that such discussion on the nature of the footprints is uncommon. Edelsbrunner *et al.* do not comment on how the inherent properties of the dot pattern affect the shape produced, nor do they present any discussion on how to choose α to produce a specific shape from the shape spectrum.

Since this landmark paper much use has been made of α -shapes, in [Edelsbrunner and Mücke, 1992] Edelsbrunner and Mücke make note of two of the most interesting applications, namely molecular structure mapping and reconstructing a surface from sampled point data. The paper presents an extension of α -shapes into 3-dimensions, but they have also been extended to take into account any intrinsic weighting of the point set in [Edelsbrunner, 1992].

α -shapes require a parameter from which to be formed, as discussed in the introduction this is common amongst all the *non-convex* footprint algorithms. This parameter is required because the idea of the footprint is vaguely defined; any shape that can be said to represent the underlying dot pattern is a valid footprint. The reason that ‘a footprint’ as a concept remains vague is that users of the algorithms have different requirements on the type of shape they need. The parameterization allows control over the detail captured/lost within the footprint therefore allowing for different footprints to be created for different applications.

There are many possible algorithms besides the α -shape that we could examine, however rather than listing them (for a larger study of existing algorithms see [Dupenois and Galton, 2009]) the rest of this section will look for novelties within the literature.

¹³This footprint type will appear in Chapter 4 but we call it the *identity* footprint

Melkemi [Melkemi, 1997] suggested an alternative approach to a single value parameter, such as α used by the α -shape, for their \mathcal{A} -shape¹⁴ algorithm: the parameter \mathcal{A} is a set of dots. The footprint is then constructed from the Voronoi Diagram of the union of the original dot pattern and \mathcal{A} ; the footprint is defined by the outer borders of the cells containing the original dots. The process for choosing \mathcal{A} is not expanded on until [Melkemi and Djebali, 2000] in which it is defined as sampled from the union of two sets:

1. The centres of the Delaunay circles associated with the Delaunay triangulation of the original pattern having radii higher than a threshold $t \geq 0$.
2. For each edge \overrightarrow{pq} of the convex hull of the original pattern, consider the point not belonging to the convex hull of DP and which is the centre of the circle passing through p and q and having sufficiently big radii.

What is meant by ‘sufficiently big’ in the second constraint is not elaborated on; given the nature of the work we can assume it relates to the same threshold as in the first constraint. Melkemi and Djebali [Melkemi and Djebali, 2001] introduce the idea of the weighted \mathcal{A} -shape, this allows the algorithm to deal with dot patterns containing areas of different densities. Each point is given a weight based on the distance between it and its closest neighbour. The set of points \mathcal{A} is found using what they call ‘the power diagram’ of the original pattern, suffice to say that it too uses a threshold value much the same as the unweighted version.

Alani *et al.* [Alani et al., 2001] also use a point set as their input parameter but their paper differs from others in the field in that it is one of the few where the application has directly led to the development of the algorithm. There exist gazetteers (or geographical thesauri) which combine place name data with limited locational information. These systems are used for queries such as requests for all the hotels in a specific area. After noting some of the current constraints on such systems (limited bandwidth, differing search terms to index terms, imprecise or precise matching, etc.) they introduce the *Dynamic Spatial Approximation Method* or DSAM. Much the same as the work performed by Melkemi and Djebali, it uses the union of the original dot pattern and another set of dots to construct the Voronoi Diagram, the footprint being the union of cells containing the original pattern. Unlike the \mathcal{A} -shape, the external points are already in the database as points known not to exist within the query location. In this instance, as the data has already been obtained, there is no need to add the further complexity of sampling. The aim of DSAM is to approximate a known area, the previous algorithms covered are not so specific, and this allows Alani *et al.* to ‘score’ the footprints that DSAM generated. They give three methods with which to evaluate their approximation:

- Total areal error – Gives a basic approximation error.
- Visual error – Gives a measure of how different the shapes are. If we take the false negative error as the areas left out of the approximation and the false positive error to be the areas in the approximation not in the expected area then the visual error

¹⁴In [Melkemi, 1997] it is referred to as the \mathcal{A} -shape but in [Melkemi and Djebali, 2000] and [Melkemi and Djebali, 2001] it is called \mathcal{A} -shape, so we use the most common notation

can be measured as:

$$V = \frac{A_{pp} + A_{np}}{A_o} 100\%$$

Where V is the visual error, A_{pp} is the false positive error, A_{np} is the false negative error and A_o is the original (known) area. A similar technique is used in this thesis; the quality of the change identifiers is assessed by the difference between the footprints created when using them and the footprints created when not.

- Quality of the spatial relationships – Preservation of important spatial relationships of the actual with the approximated area e.g., if a data point is within the actual is it within the approximation.

Interestingly the first two are quantitative measures while the third is qualitative as described, although it is possible to see how it could be made quantitative by defining all the required spatial relationships and finding the percentage that are inconsistent between the approximate and the actual areas. This level of reasoned assessment is notably absent from much of the literature. Although this is undoubtedly because Alani *et al.* have an expected shape to measure against, it seems strange that little has been done to give any form of general scoring to the footprints produced.

α -shape, \mathcal{A} -shape and DSAM are all of complexity $O(n \log n)$. This is common amongst the footprint algorithms, at least in part because they tend to be generalisations or modifications of existing $O(n \log n)$ algorithms (e.g., Delaunay triangulations, Voronoi diagrams and Jarvis March). Aware of this, Chaudhuri *et al.* [Chaudhuri et al., 1997] propose two methods for extracting the ‘perceptual border’ of a dot pattern that have $O(n)$ complexity; the s -shape and the r -shape. The s -shape is generated by laying a grid over the isothetic (axis-aligned) bounding rectangle of the dot pattern. The union of all the grid cells that contain at least one dot gives the s -shape, s being the length of the grid cell sides. Chaudhuri *et al.* note that choosing s is not a simple task and the interesting component of the method is the fashion in which they deal with this problem. First they note that there are a finite number of different s -shapes that can be created for any dot pattern, by defining the sequence $\langle s \rangle$ as:

$$\begin{aligned} \bar{s} &= \sqrt{\frac{A(W)}{n}} \text{ NB:}^{15} \\ s_i &= \bar{s} && \text{when } i = 1 \\ s_i &= \sqrt{\frac{A(H(s_{i-1}))}{n}} && \text{when } i > 1 \end{aligned}$$

W : The minimum isothetic bounding box

$A(x)$: The area of the footprint x

$H(s)$: The footprint (hull) when the grid length is s

¹⁵If the distribution of the dot pattern was completely uniform this would give an optimal value of s .

$\langle s \rangle$ finishes when each grid cell of the footprint contains only one point. This sequence gives rise to the ‘ s -shape spectrum’ of the dot pattern (similar to the work by Edelsbrunner for α -shapes) and can be performed in $O(n)$ time. To choose an appropriate s value from the spectrum they introduce the parameter ε as a measure of disparity within the dot pattern, essentially how uniform the density is across the pattern. s is now chosen from the spectrum by:

$$s = \max \left\{ s_k; \left| \frac{s_{k-1} - s_{k+1}}{s_k} \right| \leq \varepsilon, s_{k-1} \in \langle s \rangle \right\}$$

For the majority of patterns they found that an ε value of 0.3-0.5 was sufficient to achieve a suitable representation. It should be noted that they give little discussion on how the representation is measured and do point out that the ‘perceptual structure’ is not necessarily unique. The s -shape is staircase like and may be considered somewhat crude, whereas the r -shape is a much ‘smoother’ representation. The r -shape algorithm involves placing a disc of radius r over each dot and then joining edges between dots whose discs share a point on the boundary of the union of all of the discs. Like the s -shape it suffers from the difficulty of selecting a suitable value of r . Chaudhuri *et al.* observe this difficulty and proceed to show that, using the s -shape algorithm, a value can be retrieved for r where $r = \sqrt{2s_i}$, combined with the ε -measure of dispersion this gives a $O(n)$ method for producing a footprint with a single parameter and suggested ε value of 0.4. Chaudhuri *et al.* appear to have covered all the major issues (visual salience, complexity and parameter choice) in footprint generation but they do not make clear why they consider any r -shape is more suitable than any other or if it can be judged by anything other than human intervention.

There seems to be a division in types of footprint algorithms appearing, the α -shape, \mathcal{A} -shape and DSAM are all mathematically derived algorithms, they arise from the implementation of easily expressible concepts:

- α -shape – The intersection of all closed discs with radius $1/\alpha$ that contain all the points of DP .
- \mathcal{A} -shape – The union of the cells containing dots of DP from the Voronoi Diagram of $\mathcal{A} \cup DP$.
- DSAM – The union of the cells containing dots of DP from the Voronoi Diagram of $E \cup DP$ where E is a set of dots known to be external to the query area.

However s -shape and r -shape are somewhat different; the basic description of the s -shape seem to be of the same type i.e.

- s -shape – The union of the grid cells of length s containing dots of the pattern.

But the s -shape algorithm, when the iterations used to generate the s -shape spectrum are included, consists of many more steps than this, in fact to properly describe the s -shape is to describe each of the steps taken within its algorithm. The same is true for the r -shape, particularly when taken with the s -shape as a precursor. While this thesis is not meant to be a detailed analysis of the types of algorithms available, it is interesting that there

should be delineating characteristics between the algorithms. Further work in this field could entail examining which type of algorithms can be used to produce which types of footprint.

One of the few works to formally analyse the footprints created by their algorithms is the paper “What is the Region Occupied by a Set of Points?” by Galton and Duckham [Galton and Duckham, 2006]. Galton and Duckham approach the concept of finding an appropriate footprint for a dot pattern by first looking at what is meant by the concept of ‘appropriate’. Before examining the footprint criteria they point out that visual salience is problematic in that human intuition can play a great part in the shapes we see when we look at dot patterns, they note that the notion of gestalt perception almost certainly comes into play. Before describing their criteria for analysis they make one last caveat in that the specific application must decide the relevance of the footprint, by this they mean that whether or not the footprint is a suitable representation is dependent on the application. The nine general criteria they provide are, in fact, questions for which a specific algorithm should give answers in order to be compared to other algorithms to assess suitability for use in a specific application. These criteria are as follows (in which $foot(DP)$ is the footprint producing function $foot$ over the pattern DP):

1. Should every member of DP fall within $foot(DP)$ or are outliers permitted?
2. Should any points of DP be allowed to fall on the boundary of $foot(DP)$ or must they all lie within its interior?
3. Should $foot(DP)$ be topologically regular or can it contain exposed point or line elements?
4. Should $foot(DP)$ be connected or can it have more than one component?
5. Should $foot(DP)$ be polygonal or can its boundary be curved?
6. Should $foot(DP)$ be simple, i.e., its boundary is a Jordan curve or can it have point connections?
7. How big is the largest circular (or other specified) subregion of $foot(DP)$ that contains no elements of DP ?
8. How easily can the method used be generalised to three (or more) dimensions?
9. What is the computational complexity of the algorithm?

The authors note that the criteria can be split into four categories. The questions (1) and (2) focus on the relationship between the footprint and the dot pattern. (3)–(6) describe the nature of the footprint itself. (7) is, in some respect, an indicator of the quality of the footprint, in that reducing the amount of ‘free’ space is important for a visually salient (this is expanded on by Galton in [Galton, 2008]). (8) and (9) are both questions about the nature of the algorithm. They use these criteria to compare three algorithms and the general class of convex hull algorithms. The questions can be split into two categories, questions (1)–(7) are concerned with the state of the footprint whereas (8) and

(9) describe aspects of the algorithm’s nature. We note this as any convex hull algorithm will give identical answers to questions (1)–(7) differing only on (8) and (9). The three algorithms compared are the Swinging Arm, Close Pairs and a Delaunay triangulation based method.

The Delaunay triangulation method is extended into the χ -hull in [Duckham et al., 2008] and is examined in greater detail later in this section. The Swinging Arm method generalises the ‘gift-wrap’ algorithm for constructing convex hulls¹⁶, which is constructed by taking an extremal dot d_0 of DP and a half-line l , l is swung in a clockwise direction about d_0 till it collides with another point of DP , d_1 . l is swung successively from d_i to d_{i+1} till $d_{i+1} = d_0$. The Swinging Arm is identical save that instead of a half-line a line segment of length r is used. Interestingly this change allows that an anti-clockwise direction of spin can change the footprint produced.

The Close Pairs method considers simply joining all point-pairs whose distance is less than or equal to r , then taking the union of all the closed polygons as the footprint. With regard to how Swinging Arm and Close Pairs compare, the authors note that in most cases they are identical, save for criteria (8) and (9). So similar that their produced footprints are identical save for the dot patterns for which the Swinging Arm would generate different results if the direction was changed.

The extension into three dimensions is not particularly obvious for the Swinging Arm, the arm can easily be conceptually thought of as a ‘flap’, but the edge about which to rotate the flap is not pre-determined and would need to be decided on. Close Pairs generalises relatively easily; after including any polygon formed from the joins, any polyhedrons with said polygons for borders are included.

The complexity of both of the algorithms is at least $O(n^2)$ with a worst case of $O(n^3)$ for Swinging Arm and an unknown worst case for Close Pairs. This kind of systematic comparison does not appear prior to this paper and will be looked at in greater detail in Chapter 4. For the moment we note that being able to compare the footprint types and their algorithms can be useful in the assessment of suitability for any specific application.

The final algorithm that will be examined in this section is the χ -hull by Duckham *et al.* [Duckham et al., 2008] (expanding on the Delaunay method presented in Galton and Duckham [Galton and Duckham, 2006]). This paper includes a discussion of the footprint’s properties, and how these are directly tied to the method by which it is created. The method itself is simple to understand; starting with the Delaunay triangulation and successively removing the longest external edge, subject to constraints of maintaining connectedness and regularity, until either some predetermined minimum length is reached, or no more edges can be removed. The authors note that there can be no uniquely ‘optimal’ footprint when the application context is considered to be general, however, like Chaudhuri *et al.*, examine the parameter choice and its effect. There are practical limits on the minimum length l for any triangulation, if it is too large then no lines will be removed and if it is too small too many will be removed, and consequently l can be normalised. Duckham

¹⁶The ‘gift-wrap’ method is a renaming of the Jarvis March mentioned earlier.

et al. propose using this normalised parameter, λp , to find a starting value which should achieve what they call a *characteristic shape* for many, if not all, dot patterns. While they conclude that there is no λp that always produces a ‘good’ characterization, the fact that they spend time considering this is further proof of the desirability of a non-parameterised algorithm.

As previously mentioned, within the field of footprint algorithm generation there is little in the way of hard analysis of the footprints, the algorithms in relation to each other or the patterns. Clearly such work is relevant to the field and much of what has been done has only been performed recently. In 2008 Galton wrote a paper [Galton, 2008], searching for objective criteria for evaluating the acceptability of any proposed footprint in relation to the ‘perceived’ shape of a dot pattern. The paper notes that in most of the published work, “while lip-service is generally paid to the fact that there is no objective definition of such a ‘perceived shape’, little is said about how to verify this, or indeed, about exactly what it means”. Restricting attention to footprints in the form of *polygonal hulls*, simple polygons having vertices selected from the dot pattern, all the other dots being within the interior, the paper presents evidence that while a dot pattern may have several equally acceptable perceived shapes, they all represent optimal or near-optimal compromises between the conflicting goals of simultaneously minimising both the area and the perimeter of the hull.

This work was followed by a paper by this author and Galton [Dupenois and Galton, 2009], suggesting a method for classifying the footprints. Unlike Galton [Galton, 2008] it does not look at their ‘fitness’ but approaches the subject from a desire to be able to describe algorithms by the types of footprints they can create. The paper notes that the context in which the algorithm is being used determines the type of footprint that is satisfactory. With this in mind it proposes a method of using the application specific knowledge to limit the choice of algorithms for any particular user requirement. The classification bears some similarity to the set of criteria proposed by Galton and Duckham [Galton and Duckham, 2006] for evaluating the footprints produced by different algorithms and will be detailed in Chapter 4

2.6. Dynamics

The focus of this thesis is not an examination of footprints or the underlying dot patterns but how the change of the phenomena can be suitably measured. It is therefore prudent to devote some attention to examining the existing approaches to dynamic updates.

The Kinetic Data Structures (KDS), proposed by Basch *et al.* [Basch et al., 1997], are a particularly appropriate starting point because one of the applications they use is that of maintaining convex hulls under movement. The KDS is not a single algorithm, rather it is a system to describe how to create dynamic algorithms. A set of conditions, called certificates, are defined. These certificates are geometric relations that describe the shape to be maintained such that if a certificate is not true (has failed) then the desired shape cannot exist. We can therefore ascertain whether or not the convex hull needs to be

redrawn based purely on whether or not these certificates have failed. Obviously a KDS is only useful if the cost involved in discovering and processing certificate failure is small. They state that the cost is small if it asymptotically of the order of $O(\text{Polylog}(n))$, or $O(n^\epsilon)$, for some small $\epsilon > 0$. A KDS with such small costs is deemed *responsive*. Further to this a KDS is *efficient* if there are very few internal events compared to external events¹⁷, *compact* if it has a near linear number of certificates and *local* if no object participates in too many certificates. The change identifiers can be treated as certificates on the dot pattern, and a framework using them as a form of a KDS. A change identifier KDS should be responsive and, ideally, compact. The terms efficiency and local have no obvious counterpart in a change identifier based KDS as they are specific to certificates on the footprint.

The KDS uses short term motion plans for the objects, these are used to sort the events into queues such that the most likely certificate failures (events) are looked at first. An example on convex hulls is given where it is shown that by checking a set of certificates, all using the rule $\text{ccw}(a, b, c)$ in which a, b and c are points and the relation is true if they form a counter-clockwise triangle, any events whereby the convex hull is no longer valid for the dot pattern can be found. Further to this they provide a more robust method using the dualities of the convex hull and focusing only on the upper envelope. All of this gives an excellent base from which to work but it may not be a directly transferrable approach for footprints. This is because, unlike convex hulls and as discussed earlier, footprints are vaguely defined. Hence, choosing the certificates is no longer a trivial task. We will look at possible requirements that can be made on footprints in Chapter 4.

Hershberger and Suri [Hershberger and Suri, 2003] have a very different idea to Basch *et al.* using a technique called adaptive sampling. By using an approximation of the extrema from the dot set they find a convex hull that approximates the ‘true’ convex hull, with triangles of uncertainty (see Fig. ??) over each line segment. The area of uncertainty is given by the supporting line perpendicular to the direction in which the point was found.

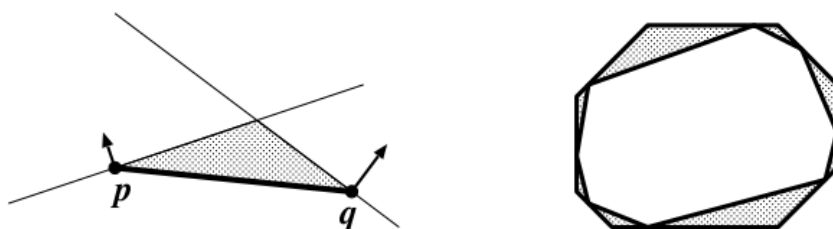


Figure 2.9. Example of Uncertainty Triangles. Image from [Hershberger and Suri, 2003]

While sampling is not a new concept Hershberger and Suri suggest that uniform sampling produces poor quality approximations in low curvature regions. As such they propose an adaptive sampling scheme.

¹⁷*External Event*: Changes the shape. *Internal Event*: Shape stays the same, certificates change.

The method is performed by first uniformly sampling extrema in directions $\frac{2\pi j}{r}$ for $j = 0, \dots, r - 1$ then adding up to r more extrema using their adaptive technique. Given an edge e , $\Theta(e)$ is defined as the minimum angle between the directions the endpoints e were sampled in. $\Theta(e)$ and the proportion of the perimeter that the length of e represents are used to provide a *sample weight* $w(e)$ for e . If $w(e)$ is greater than one then the edge is refined; the extreme point is found in the direction that bisects the angular range defined by e 's endpoints. If the point found is not an endpoint of e then e is replaced by the two new edges of the vertices of e and the newly found point. This greatly reduces the error in the approximation. As has previously been discussed, the approximation of a footprint is central to the concept of the use of change identifiers.

Chiang and Tamassia [Chiang and Tamassia, 1992] present a general review of the field. While it is a little dated (1992) it serves as a good presentation of methods still in use. Unlike Basch *et al.* or Hershberger and Suri, Chiang and Tamassia look at the ways in which the data structure holding the dynamic dot pattern and the footprint is maintained. Chiang and Tamassia examine various forms of binary trees and *fractional cascading*. Using these data structures they approach some general dynamic methods. The examination is quite extensive so we have picked some terms which may be considered when choosing the data structure(s) used to contain the dot patterns.

- **Local rebuilding / Balancing** This is a technique applied to search trees so that they maintain logarithmic height.
- **Partial rebuilding** This rebuilds entire subtrees when they become out of balance.
- **Global Rebuilding** Periodically reconstructs an entire tree, often used with 'weak' updates (like lazy deletion).
- **Lazy Deletion** Does not remove deleted item but marks it as deleted to be dealt with during the reconstruction.
- **Decomposable** A search problem is decomposable 'if for any partition (S', S'') of S the answer to a query on S can be obtained in constant time from the answers to queries in S' and S'' .

Further to the discussion on general dynamic methods considerations Chiang and Tamassia give a list of things we can reasonably expect from any dynamic algorithm for convex hulls:

- find if a given point of DP is on the convex hull $foot_{convex}$ of DP ;
- find if a query point is internal or external to the convex hull $foot_{convex}$ of S ;
- find the tangents to the convex hull $foot_{convex}$ of DP from an external query point;
- find the intersection of the convex hull $foot_{convex}$ of DP with a given query line;
- report the points on the convex hull $foot_{convex}$ of DP .

They note that the set of points DP is updated only by insertions and deletions, so any

point movement should be treated as being removed then added as a new point, which is a concern that we will have to bear in mind when we consider how the data may enter the change identifier framework. Chiang and Tamassia describe a method by Preparata with update and query time of $O(\log v)$ and a report-query time of $O(v)$, where v is the number of vertices currently in the convex hull $foot_{convex}$ of DP . The method only deals with insertions and is therefore not entirely applicable to change identifiers but it does introduce the concept of splitting the footprint into an upper and lower hull, the methods used for the upper are transferable to the lower. This concept seems common, appearing in the next algorithm and in the KDS example.

Overmars and Leeuwen [Overmars and Leeuwen, 1981] present an algorithm for processing what they call fully dynamic hulls which can handle both insertion and deletion. Again this considers splitting the hull into two sets, one for left and one for right. Splitting the hull is a useful technique for speeding up computation but it requires that the two sides are comparable. Footprints considered more generally than the convex hull do not necessarily have the strict mathematical definition that allows us to make such a split.

As with all the work described in this chapter the above discussion of dynamics is not a complete listing. There is, for example, the work performed by Gold [Gold, 2005] examining the nature of multiple dimensions for dynamic data structures. The field is extensive but is not exhaustively detailed here as the other works do not directly impact on this thesis.

2.7. Summary

The papers we have examined here are by no means the sum of all the papers in the fields represented. We have instead intended to give an overview that shows the related materials and, most importantly, the methods that have been used to analyse dynamic dot patterns, footprints and dynamic footprint assignment.

With regard to the originality of our work, there has been much work on the field of spatio-temporal data and dynamic updates of convex hull. None of the papers that were found in researching for this thesis has presented any idea that is close in nature to our change identifiers. While the change identifiers do have similarities with some existing ideas, the examination of the spatial properties of the dot pattern as a static abstraction combined with the allowance for some ‘error’ when updating footprints (considering any footprint can be seen as an approximation of some abstract ‘true’ or ‘best’ region for the dot pattern) makes them a unique concept.

3. Dot Patterns

The previous chapter was an examination of existing literature that included work that has constructs with the same or related structures to that of dots, dot patterns and dynamic dot patterns. The definitions of the structures that this thesis uses were given in the introduction and this chapter will expand upon these definitions in light of the existing work. In particular we examine the descriptors that can arise from looking at the dot pattern as a static representation.

3.1. Change and Dot Patterns

The abstraction provided by considering the entities as dots means that change as it relates to a dot can only occur in three ways: movement, disappearing or appearing [Thériault et al., 1999; Huang et al., 2008]. These simple actions can lead to complex emergent behaviour in the dynamic dot pattern; for example expansion, dispersion, rotation, etc. These behaviours can be measured by looking at the change in the properties of the patterns between two adjacent (as defined by Stell [Stell, 2003]) phases of a dynamic dot pattern. Expansion, for example, is a measure of increase in extent between two phases and can be determined by the positive change in the standard deviation from the centroid. There is more than one method by which to measure extent (standard deviation, bounding box area, etc.) and change in any of these properties indicates a value for the complex behaviour of expansion (negative or positive). As mentioned in the introduction, these different properties are *descriptors* of a pattern while a general property characterisation such as ‘extent’ is a *class* of descriptor. Using this reasoning, a definition of change for collectives represented by dynamic dot patterns can be given by the difference between two phases in one or more descriptor classes measured by descriptor methods from those classes.

3.2. Descriptor Classes

For the measurements of change to be useful across any given dynamic pattern we need to be able to state, with some confidence, that any of the complex behaviours a collective represented by a dynamic pattern can exhibit will be ‘caught’. The framework this thesis presents should not have a list of exception cases in which it cannot accurately identify change, e.g., ‘the change identifiers will not correctly cause a footprint update if the dynamic dot pattern rotates’. There are numerous, possibly incalculable, different

descriptors for a dot pattern; any statistical, geometric or otherwise calculated value that can be assigned to a pattern is a valid descriptor. Using all possible descriptors is not only a Sisyphean task, it will also probably involve repeated measurements within the same descriptor classes. It follows that we should instead focus on having a descriptor method from each class.

The classes are not, however, a strictly delineating classification – some descriptors straddle the boundaries of multiple classes and some classes are clearly dependent on others; as such the class divisions should be used as a guide rather than a set of requirements. Choosing the classes is not a simple task, there are multiple aspects of the dots which can be measured and the classes selected must be those that could be considered both broad and descriptive. To deduce the appropriate classes from first principles the most basic information of the pattern is investigated, the exploration increases in complexity until we have covered a sufficient range of aspects. The cardinality of a dot pattern is, perhaps, the most immediately apparent datum of information that can be attained. However, there are no alternative methods to counting with which to find the cardinality of a pattern, and cardinality is therefore not so much a class as it is a descriptor.

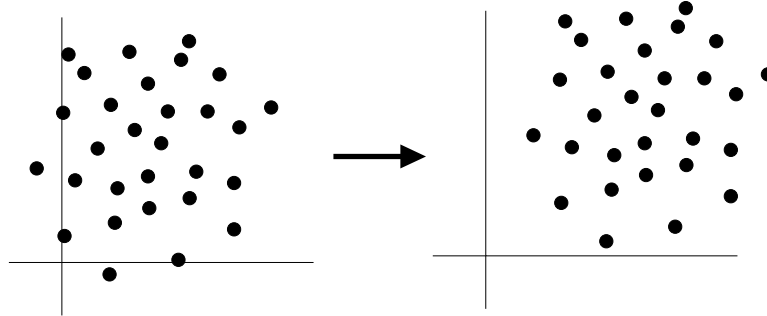


Figure 3.1. Dynamic dot pattern changing in position.

The dots are little more than a location and it seems sensible that the first aspect of a pattern we would need to be able to describe is its *position* (Fig. 3.1). Position can be measured in multiple ways but will tend to return a vector as its value, usually a coordinate location. This observation is important as such multi-part values can lead to issues when normalising. Normalisation will be further discussed later, for now we note that if a pattern with an areal coverage of 1000 units² moves by 1 unit very little change has occurred, if however the areal coverage is only 4 units² such a change is quite large (an example of this is shown in Fig. 3.2¹). Thus, a description of change should be proportionate to the pattern, and for the location change to be proportionate we need to have some value by which to make it so. As position is so greatly affected by areal coverage it has indicated our next class to consider: the size of the pattern. So as not to be confused with the cardinality we shall name this class *extent* (Fig. 3.3).

A pattern's location and extent allow measurements of change by translation and scaling respectively, if we require that we must be able to measure change in at least all the standard affine transformations (i.e., translation, scaling rotation and shearing), and that

¹Although the areal coverage value for the larger pattern is far less than 1000 because of image space constraints.

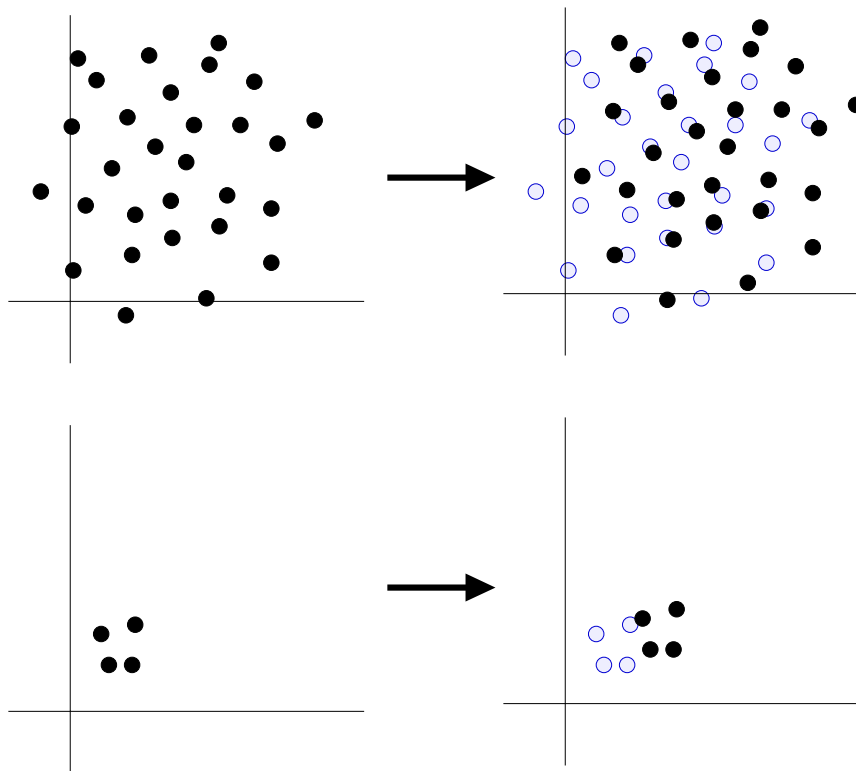


Figure 3.2. Demonstrating the relative difference in position change of two patterns of different extents.

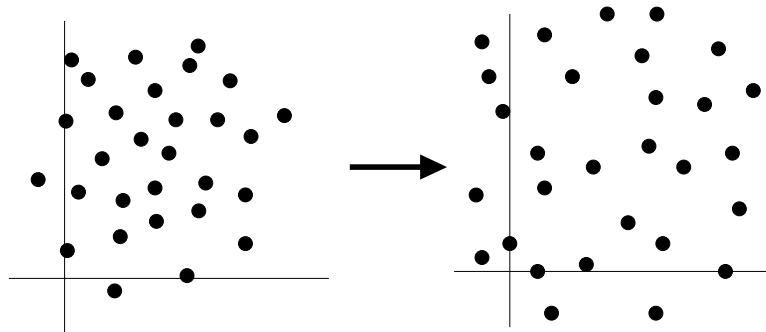


Figure 3.3. Dynamic dot pattern changing in extent.

rotation can occur without change in extent or location, then the third class should be *orientation* (Fig. 3.5). Shearing a pattern will change the orientation of a pattern so does not need to be directly measured; this can be seen demonstrated in Fig. 3.4.

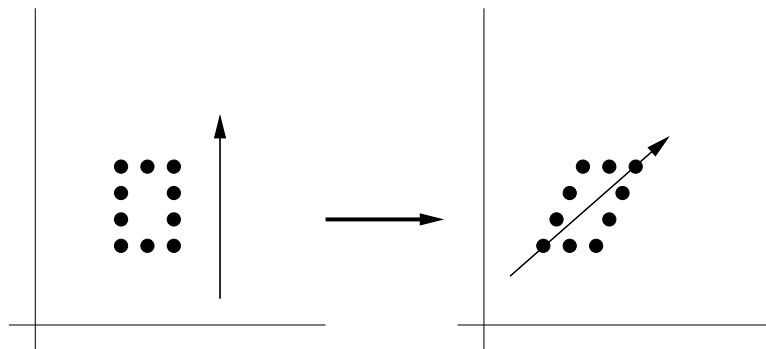


Figure 3.4. Dynamic dot pattern shearing. The arrow represents the pattern's orientation (as found by Ordinary Least Squares regression).

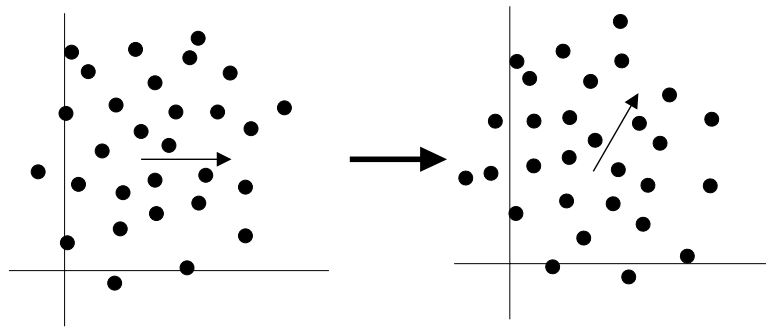


Figure 3.5. Dynamic dot pattern changing in orientation.

The three classes described above (*position*, *extent* and *orientation*) treat the pattern as a region with its own properties, as opposed to a set with no properties beyond its membership. Further traits a region can have are given by the shape descriptions in Galton [Galton, 2000] that were detailed in the background chapter. The two which have not yet been considered are *connectedness* and *dimension*. As was previously noted, Galton's shape descriptions of connectedness makes little sense when applied to dot patterns; the dots are, by their nature, disconnected. However it is often the case that patterns can appear to have areas of differing densities, which could easily be seen as disconnected components of a pattern (Fig. 3.6). Measuring connectedness in this fashion is not straightforward; do we consider the degree of connectedness or do we measure the number of distinct components? If we measure the number of components how do we place a threshold on when we consider a disconnection to have occurred? Connectedness cannot be ignored as too problematic a class, as it is indicative of complex underlying behaviour of the type described by Huang *et al.* [Huang et al., 2008] (merge and split) and we will look at how it may be measured when methods for the descriptors are introduced.

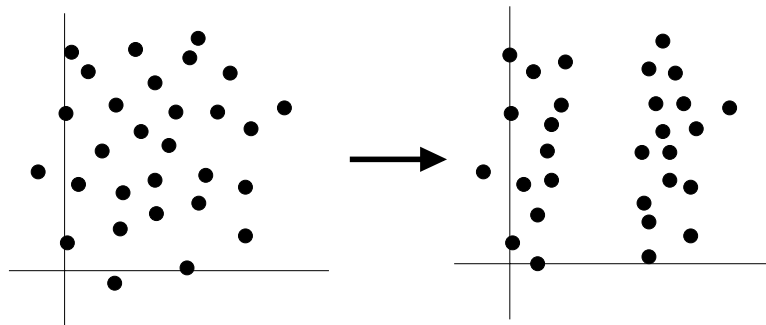


Figure 3.6. Dynamic dot pattern changing in connectedness.

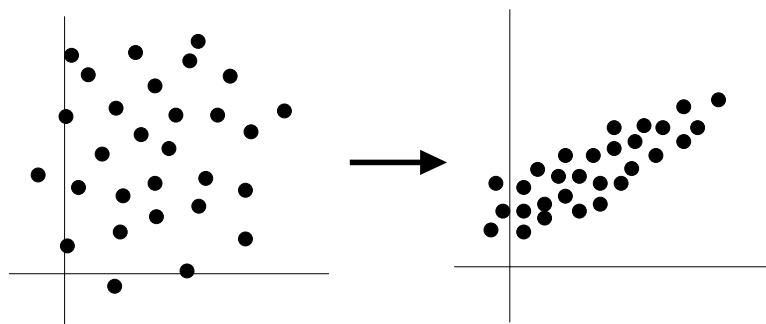


Figure 3.7. Dynamic dot pattern changing in dimension.

As discussed by Galton [Galton, 2000, ch. 4.7.3] Dimension is split into two approaches; strict and apparent. As was mentioned in the background (Chapter 2), strict dimension is not particularly useful for describing dot patterns; it is always possible to draw a curve through all the dots in a pattern and thereby call it 1-dimensional. We are more interested in apparent dimension; the idea that a 2-d pattern can be close enough to linear as to, at a coarse enough granularity, appear 1-dimensional (Fig. 3.7). Change in dimension may not be picked up by identifiers for the previously described classes; thus dimension is the next identifier class. Like connectedness, dimensionality requires a choice between either stating to what degree a pattern is in the dimensions of the space in which it is embedded or measuring the integer value for which dimension appears to represent it. This can be solved by having multiple descriptors; measuring the degree of dimensionality for each dimension the pattern can exhibit and comparing their values. For patterns in a planar space this is not necessary because the pattern can only exhibit apparent dimensionality in 0 or 1 dimensions. An apparent dimensionality of 0 (the pattern is densely clustered around a single point) will be identified by extent descriptors; leaving only a measure of apparent dimensionality 1, or collinearity, to be implemented.

If measures of change of extent, orientation, position, connectedness and dimensionality are implemented, change in the affine transformations and in the herd evolvments of Huang *et al.* [Huang et al., 2008] can be tracked. The current classes all treat the pattern as if it had a surrogate region, so changes which may affect the footprint arising from its collective nature should also be considered. As an example of the distinction consider the two dot patterns shown in Fig. 3.8.

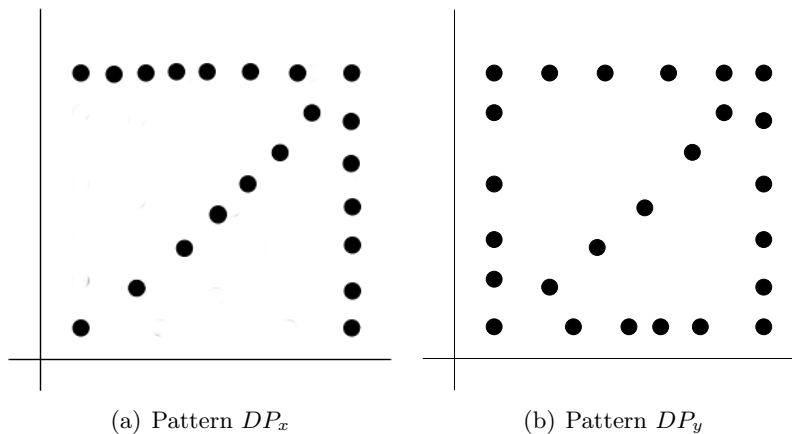


Figure 3.8. Two dot patterns with the same extent, position, orientation, connectedness and dimensionality

Depending on the methods used as descriptors both DP_x (Fig. 3.8(a)) and DP_y (Fig. 3.8(b)) can have the same extent, position, orientation, connectedness and dimensionality. However DP_y has large empty spaces. Measuring cardinality would prevent this but we have already discussed why it is not sufficient as a class of descriptor. Instead we track the *distribution* of the dots within the pattern, this allows descriptors that measure homogeneity, global density and cardinality to be tracked, all of which are likely to show different measures for DP_x and DP_y .

This section has presented list of descriptor classes that cover the aspects of change that

a dynamic dot pattern can undergo. Consequently we can be confident that, if we have a descriptor for each class and a change identifier using each descriptor, we will be able to identify the significant changes in a dynamic dot pattern and use this to inform the update of a footprint representing the dynamic dot pattern.

3.3. Descriptors

With the classes identified we can begin to examine the actual descriptors. Those described below are by no means a complete list but instead give an overview of the descriptors used within this thesis.

3.3.1. Position

To compute a value for position requires a point or space relative to which we can measure it. Even with a clear origin or frame of reference there are a range of different units of measurement for position, either with numerical or with qualitative values; e.g., polar or Cartesian coordinates and compass positions respectively. It makes intuitive sense to use the frame of reference in which the dots themselves are positioned and to use the same unit. However it is not clear how change would be measured for qualitative units without defining quantitative thresholds on them. To avoid the complication of further thresholding we will focus on the numerical measures.

Example Methods:

- **Centroid** – The mean location of all the dots within the pattern.
- **Isothetic Bounding Box Centre** – The centre point of the axis-aligned (isothetic) minimum bounding box of the pattern.
- **Bounding Box Centre** – The centre point of the minimum bounding box of the pattern (non-axis aligned).
- **Minimum Disc Centre** – The centre point of the minimum bounding disc of the pattern.

It is apparent that there is a difference between the first and the last three given descriptors. The former descriptor treats the pattern as a set of points and the latter all apply a *surrogate* footprint to the pattern. Many of the classes have descriptors of each type. The minimum disc and the minimum bounding box (not axis aligned) are more computationally complex than the isothetic minimum bounding box and as a result may be unusable as change identifiers; it not being clear if they actually provide a sufficiently ‘better’ (more accurate) centre than any other measure for the greater computation time they take.

3.3.2. Extent

Unlike position, extent measures tend to be represented by a single value, but this value is not always of the same unit. Some descriptors are linear (e.g., pattern diameter) and others are of the order of the space the pattern inhabits (e.g., area of the bounding box). As we are currently only concerned with measuring the properties of the pattern, the relevance of unit type difference is not yet obvious, however when change identifiers are examined in Chapter 5 the unit type will affect how the values of change are normalised.

Example Methods:

- **Variance from Centroid** – The variance from the mean centroid. We use variance so as to avoid the square roots required by the standard deviation.
- **Bounding Box Area** – The area of the isothetic minimum bounding box.
- **Diameter** – The diameter of the pattern is the distance between the two furthest dots.

The diameter of the pattern is found by locating all the external dots i.e., all the dots that are vertices on the convex hull of the pattern. This may make it too complex to be used when change identifiers are considered, despite its conceptual simplicity. However an approximation can be found by considering the mean between the lengths of the diagonal of the isothetic minimum bounding box² and the longest edge of the isothetic bounding box³. Alternatively a less accurate estimation can be attained using the greatest distance between the dots in the extremal axis aligned dimensions (for a Cartesian planar embedded pattern this would be the dots with the greatest and least x and y values). The bounding box of a pattern requires the extremal dots to be found before computing the vertices of its corners and will therefore take longer to compute than the less accurate estimation. This thesis makes use of the axis-aligned extremal dots distance as its estimated diameter to provide a very computationally fast extent descriptor. To further increase its speed it uses the squared distance instead of the actual distance, preventing the computationally difficult task of the square root. Of note is that all three measurements return a squared unit and this should be considered when comparing them to other descriptors.

3.3.3. Orientation

Orientation is the direction in which the pattern is facing. As the dots do not have an associated direction, unlike position, this cannot be an aggregate of individual values. In fact given the information inherent within the dot pattern a true orientation is impossible as even the ‘line of best fit’ will not tell you in which direction along the line that the pattern points. From a change identifier point of view this is irrelevant, as all we need is a measure which will change as the orientation changes; the measure does not need to describe the orientation exactly, as long as it is linked to it.

²Guaranteed to be no less than the length of the diameter

³Guaranteed to be no greater than the length of the diameter

Example Methods:

- **Gradient of Line of best fit** – There are multiple ways of measuring the line of best fit; however, the ordinary least squares (OLS) method is perhaps the least complex. OLS is a linear regression approach from the field of statistical analysis that minimises the squared vertical distance between a dot from the pattern and the line of best fit. As OLS is a statistical analysis technique it has some properties which do not apply directly to the use of spatial data. Within statistical analysis one of the variables would be expected to be an observed result dependent on another. For example when measuring the growth of children between the ages of 10 and 14, the height is observed data that is dependent on the age. Within a dot pattern all the variables⁴ are independent and, as a result, we do not to take into account error in the observation. With no errors to concern us we can use the simplest form of OLS estimation to find the line $y = \alpha + \beta x$ in which:

$$\beta = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$
$$\alpha = \bar{y} - \beta \bar{x}$$

- **Gradient of the Principal Component** – Found by Principal Component Analysis (PCA). PCA finds the dimensions (components) with the highest variability within the pattern and is commonly used as a dimensionality reduction technique. Formally it transforms the coordinate system the data resides so that, when the data is projected onto it, the distribution across first coordinate has the greatest variance, the second coordinate has the second greatest, etc. To find the Principal Component we find the eigenvector corresponding to the largest eigenvalue of the covariance matrix of the data.

We note that the principal component is not always in the same direction as the line of best fit, as the OLS method minimises only the distance in the y-axis (Fig. 3.9). Both PCA and the OLS linear regression technique are explained in more detail in [Bishop, 2007].

3.3.4. Connectedness

Connectedness is actually a form of distribution measure as its measurements are performed by comparing inter-dot differences. It is, however, one that appears salient enough to warrant its own class. The patterns can often appear to split into separate groups and identifying the change in these groupings is similar to the behavioural evolvments of Huang [Huang et al., 2008] (herd splitting and joining). Connectedness in this fashion can be a discrete measure or a continuous value: the number of distinct groups and how connected the pattern is respectively. The continuous value approach, perhaps, fits better in the distribution class. Thus for the connectedness class we look only at the approach

⁴For a pattern in a Cartesian planar space these would be the x and y coordinates

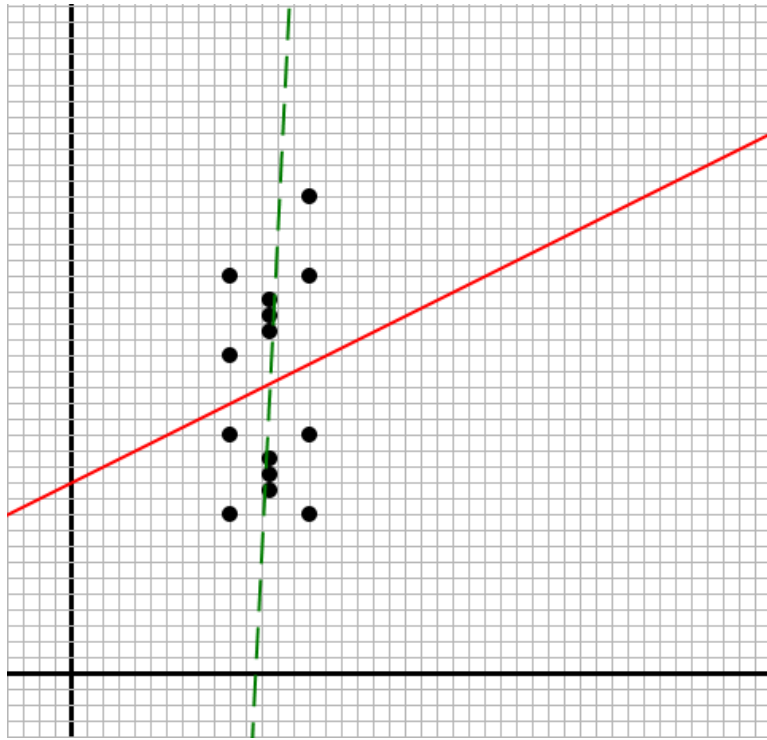


Figure 3.9. OLS (Solid line) VS. PCA (Dashed Line)

that provides an integer value for the number of clusters.

Example Methods:

- **Greatest Jump Agglomerative Clustering** – This is an extension of the agglomerative clustering approach to give a possible value for the number of intuitively identifiable clusters. The extension appears to be novel to this thesis but is an obvious enough technique that we are sure it must have a previous use elsewhere. By running an agglomerative clustering method using, Euclidean distance as its metric, a hierarchy of possible clusterings can be created. Agglomerative clustering iteratively concatenates the dots into clusters by finding the closest distance between any two clusters. Fig. 3.10 shows an example of the agglomerative clustering process. We take note of the first unusually large jump in distance across the run of the clustering. If the jump is greater (by, for example, a multiplication of 2) than the average distance jump then the clustering preceding this jump is likely to contain a saliently identifiable set of clusters. In the example in Fig. 3.10 this would be the jump from step 6 to step 7 (Fig. 3.10(c) to Fig. 3.10(d)). This method is slow but effective at finding the parts of the pattern that we may identify as individual components.
- **K-Means Clustering** – An alternative to the *hierarchical* approach given by the agglomerative clustering method is the *K*-Means approach. Given a number of clusters *K* this minimises the squared distance between each dot and its closest cluster centroid. A full description of this method can be found in [Bishop, 2007] but we assert that it is impractical to use it as a descriptor. While faster than an agglomerative approach it too must use an iterative process to find the best fit *K*

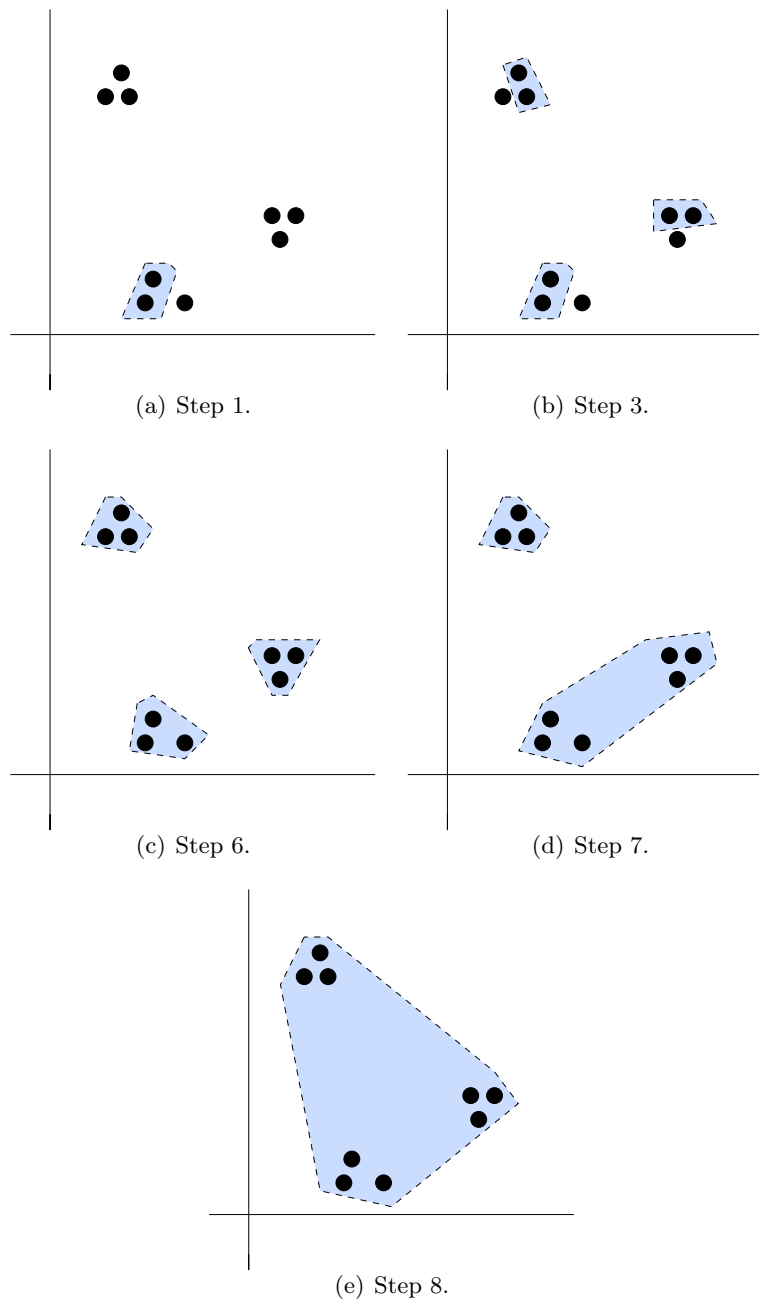


Figure 3.10. Agglomerative Clustering Method

(although there are also a number of different seeding algorithms). have found the results to be less than satisfactory.

The agglomerative clustering method has to perform a large number of computations as it requires the nearest neighbours for each dot to be found (not estimated nearest neighbours as we will consider later) and involves an iterative process comparing cluster distances. As a result it may be infeasible to use as the basis for a change identifier.

3.3.5. Dimensionality

Measuring the appropriate dimensionality can be performed in several ways and, like connectedness, we can envision both discrete and continuous measures: measuring the

apparent dimension the pattern is in and measuring the degree to which a pattern fits a given dimension respectively. Within this work all these measures are assuming a planar space so the dimensionality measures are focused on identifying how linear a pattern is. However most of these are directly extensible into further dimensions. There is also the scope for measures that allow for 1-dimensional non-linear patterns (e.g. a S-shaped configuration), and indeed we suggest in future work that extending the set of descriptors would be a useful endeavour.

Example Methods:

- **Correlation Co-efficient** – Measuring how closely the pattern conforms to its linear estimator the pattern is. The correlation co-efficient can be found using the Pearson co-efficient correlation equation, if $\text{cov}(X, Y)$ is the covariance of X and Y and σ_X is the standard deviation then the Pearson co-efficient correlation is:

$$\frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

If we are using PCA to find the gradient then we already have the covariance matrix for X and Y so may be able to save processing time by using them in conjunction.

- **Principal Component Eigenvalue Difference** – Taking the pattern's eigenvalues and finding how weighted one is over the other. In 2-dimensions we can measure how linear the pattern is by the difference between the principal eigenvalue and the orthogonal eigenvalue divided by their sum. The closer to 1 this value returns the more 1-dimensional the pattern.

3.3.6. Dispersion

This is possibly the most far-reaching of the classes in that it attempts to describe the layout of the pattern: How dense is it? How homogenous is its density? etc. As a result it has a large number of potential descriptors with a range of different complexities. It may be the case that multiple descriptors from this class are required to accurately measure change. If this is the case the class may need to be split into separate sub-classes⁵.

Example Methods:

- **Cardinality** – Simply the number of dots in the pattern.
- **Global Density** – The global density of the pattern: Cardinality divided by an extent measure, usually the bounding box area but variance is an equally valid option.
- **Estimated Nearest Neighbour Distance Variance** – This is a method that returns a value for how clustered a pattern is, and is computed as the variance in distances between estimated nearest neighbours. To avoid the high computational

⁵This relies on such classes being identifiable

complexities suffered when finding nearest neighbours we use the nearest neighbour in the plane aligned dimensions. If the patterns are stored in a data structure that is sorted by all dimensions of the space the pattern is embedded in (e.g., in a Cartesian space a structure sorted by x and y) then this can be estimated in $O(\log n)$ time (assuming a $O(\log n)$ search time). It is not possible to be sure that the actual nearest neighbour has been found, as is demonstrated in Fig. 3.11. Using the estimated nearest neighbour difference, b would be identified as the nearest dot to O while a is its actual closest dot; the dots that surround O in the cardinal directions are closer in x or y coordinates than a and would therefore be adjacent to O in the data structure. For large dot patterns such exceptions will not greatly change the nearest neighbour distance variance so we can accept the estimation that this method produces.

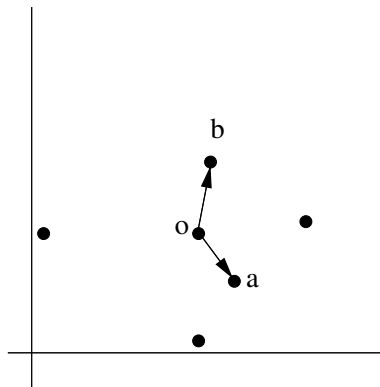


Figure 3.11. The case when the x and y ordered trees will not locate the nearest neighbour. \vec{oa} is 5 units in length while \vec{ob} is 6.1 units.

- **Skewness** – Skewness is a measure of the how uneven the distribution of a pattern is from the mean, often thought of as how much the histogram of the data slants to the left or right. It is given by:

$$s = \frac{\mu_3}{\sigma^3}$$

where $\mu_3 = E[(X - E[X])^3]$

In which σ is the standard deviation and $E[X]$ is the expected value of X , for the purposes of a dot pattern this is equivalent to \bar{X} .

- **Kurtosis** – Kurtosis is the measure of how even the distribution of the pattern is, often thought of as how flat its histogram would be. Kurtosis makes use of the fourth moment about the mean⁶ and is therefore related to skewness which uses the third. It has the equation:

$$k = \frac{\mu_4}{\sigma^4}$$

where $\mu_4 = E[(X - E[X])^4]$

⁶Moments about the mean are values used to describe probability distributions and have the general form $\mu_k = E[(X - E[X])^k]$ for the k^{th} moment about the mean, for example the second moment about the mean is the variance.

3.4. Descriptor Analysis

The descriptors range from simple measures to comparatively complex statistical analysis. For the purposes of this thesis it is necessary for the descriptors, when used within a change identifier framework, to be computable within a reasonably short time. At this point we must clarify that the descriptors are not algorithms, although descriptors often suggest an algorithm. For example, Kurtosis is strictly defined by a mathematical formula but there are numerous ways to implement the formula. It is possible that the more complex descriptors will fail to meet this time constraint (e.g., greatest jump agglomerative clustering), particularly considering that multiple descriptors will likely need to be used to fully explain each pattern. However, it is also known that many of the descriptors will operate within the same classes and, as described above, therefore only a subset of the descriptors may be required. The task is to decide upon a set of descriptors for which each descriptor provides unique information while minimising the time it takes to run the set.

To assess information redundancy a method is required to identify the similarities between the descriptors; using heat maps [Sneath, 1957] of a correlation matrix provides a visualisation of the relationships between descriptors. A heat map is a graph for which each mapped point is shown as a colour from a spectrum, with high values at one end and low values at the other. The heat map shown below uses the standard approach of ‘cold to hot’ colouring, with low negative values being blue and high positive values being red. Given that the map is overlaying a correlation matrix, red indicates a strong positive correlation and blue indicates a strong negative correlation. Before creating the correlation matrix a set of dot patterns on which to run the descriptors is required. For a dynamic dot pattern based on a collective each pattern is related to the others within the set; usually as the dots have a concurrent identity, representing the same entities. This relation means that a correlation matrix of descriptors over a dynamic dot pattern will likely show false correlation. For example if the dynamic dot pattern represents a migrating herd: The herd is traversing a particularly wide and long stretch of their route leaving themselves open to predation. As a result the herd’s extent decreases as the animals huddle together for safety, nevertheless some of the members are caught and devoured. Such a dynamic dot pattern will show strong correlation between cardinality and bounding box area. While these descriptors may well be correlated there is no way of knowing if the correlation is indicative of the behaviour of the underlying collective or the relation between the descriptors. To make sure that such possible false assumptions are avoided an automated method is used to generate sets of unrelated and randomly shaped patterns (while remaining aware of the pitfalls of assumptions based on test data described by Laube [Laube and Purves, 2011] and discussed in the background chapter Chapter 2). The randomly generated patterns can contain obvious concavities and can differ in extent, cardinality, connectedness, dimensionality, distribution and orientation. An example of some of the patterns that can be produced by this method is shown in Fig. 3.12

We must be careful, however, not to assume classhood solely on the basis of a correlation of heat map values. For example the first heat maps produced showed a correlation between the variance of nearest neighbour distance and the area of the bounding box.

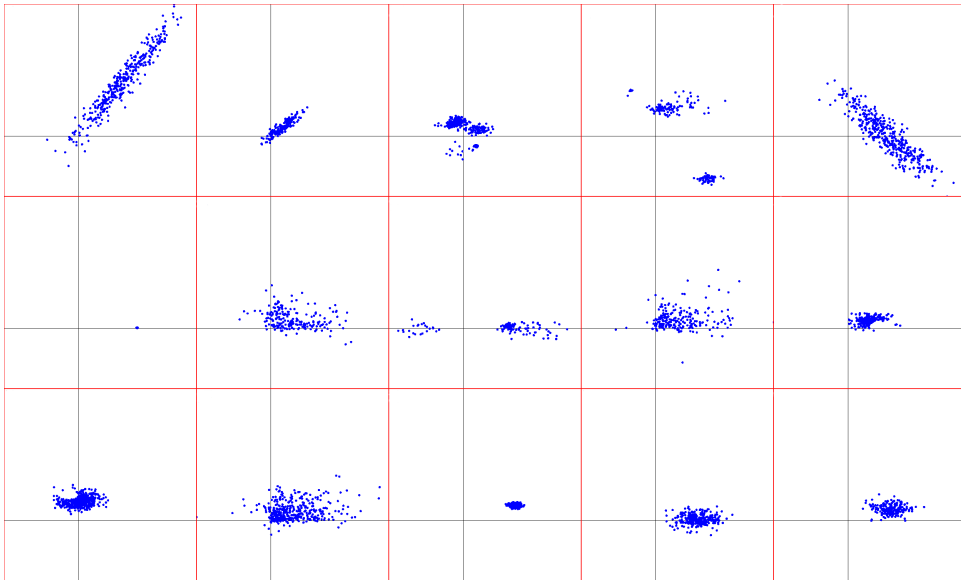


Figure 3.12. Examples of dot patterns produced using the random generation method

This is a logical result as the smaller the area in which the dots are situated the more likely they are to lie close to each other. However this correlation is not one that shows redundant information as the *aim* of the nearest neighbour distance variance is to describe the dispersion of the pattern whereas the area intends to show extent. Instead of finding a correlation indicating concurrent classhood instead a flaw has been found in the nearest neighbour distance variance. To remove this false concurrency the nearest neighbourhood variance needs to be scaled by an extent measure; the important information is not the variance of the distance but the variance in what proportion of the available distance is taken up by a dot and its nearest neighbour. The heat maps, therefore, do not show just which descriptors are superfluous but also those which need to be modified before use.

Position is not included within the heatmap as it is, for patterns in a cartesian planar space, a pair of values and as such cannot be used within the correlation matrix. However even if we could include position within the heatmap it is unlikely to provide any correlations outside its class; all the descriptors that use dot locations⁷, apart from the eigenvalue, difference are relative to the centroid.

Fig. 3.13 shows a map for all the descriptors together. We are primarily interested in uniqueness within each class but by graphing them all together we may find unexpected correlations. The descriptors have been grouped by their respective classes:

Orientation: Principal component vector gradient and gradient of ‘Line of best-fit’ found via the linear regression method of Ordinary Least Squares.

Connectedness: The number of clusters within an agglomerative clustering hierarchy that is found at the stage before the greatest jump in distance when joining the clusters together.

Extent: Area of the bounding box, the estimated diameter squared of the pattern

⁷Of the descriptors we provide, only cardinality does not make use of dot position data.

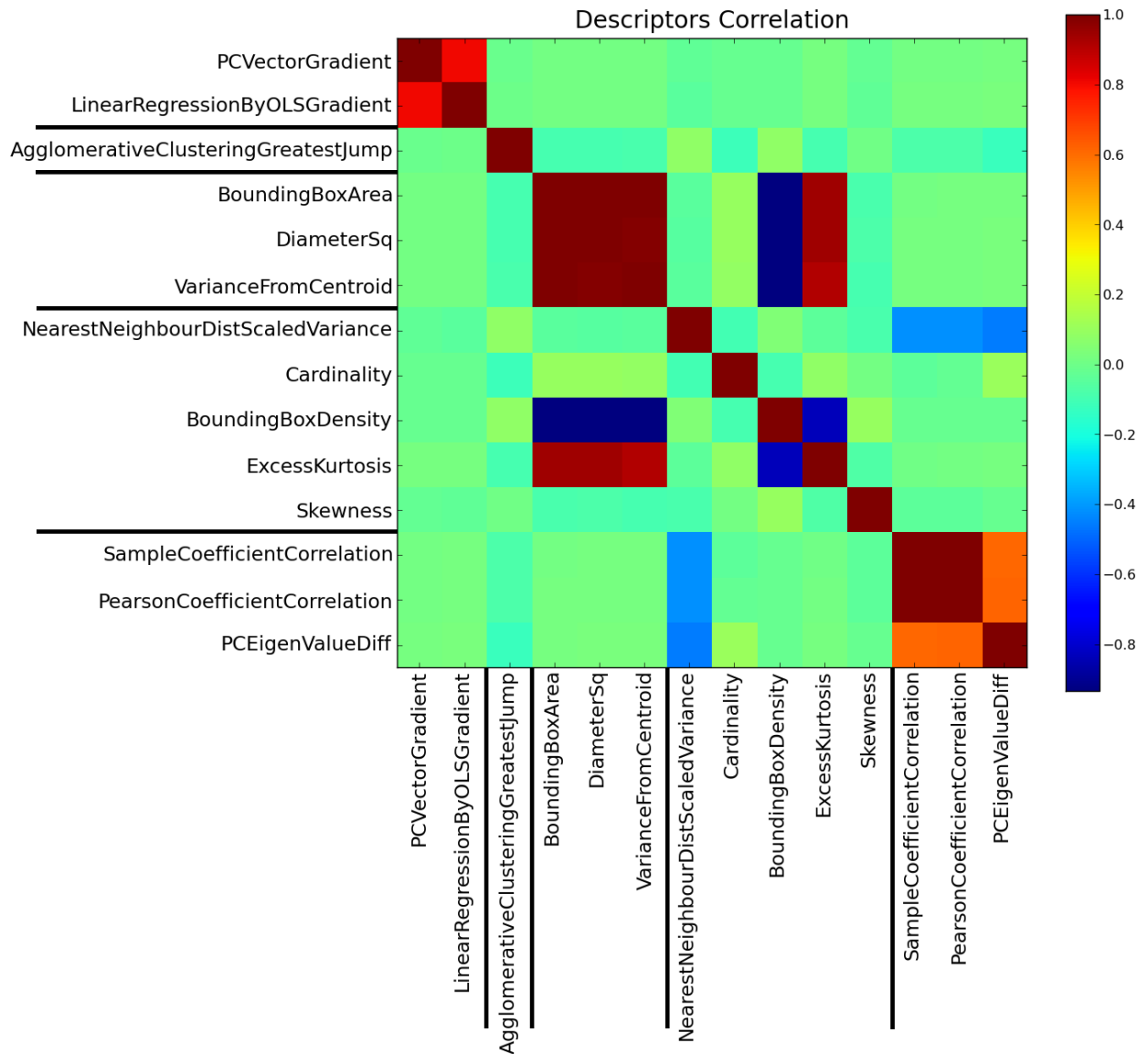


Figure 3.13. Correlation Heat Map: All Descriptors

and the variance from centroid.

Dispersion: The variance in distance between nearest neighbours, the cardinality of the pattern, the density of the pattern within its bounding box, the excess kurtosis and skewness.

Dimensionality: The absolute sample coefficient correlation, the absolute Pearson coefficient correlation and the absolute difference between the eigenvalues divided by their sum.

While above it was shown that the orientation methods can produce very different results (Fig. 3.9) the heatmap demonstrates that they increase and decrease in their values concurrently; leaving the choice of the most appropriate descriptor to be based on which can be performed in the fastest time.

As the agglomerative clustering identification method is currently the only one of its type, and is without strong correlation to any other descriptor there seems little need for further

discussion on its behaviour for the moment.

As might be expected, the descriptors within extent all correlate strongly with each other, so choosing the best extent descriptor will simply be a matter of finding the one that can be computed in the fastest time. Of particular interest in the extent descriptors correlations is the fact that the estimation of the diameter correlates so strongly with the other extent measures – justifying its use as a measure of extent.

The dispersion methods are a ‘mixed bag’, encompassing a large range of descriptors which can provide information about the layout of a pattern, but do not necessarily fit in the more strictly defined classes. Thus, it is not surprising that there is little correlation within this class. The two descriptors that show correlation are density and kurtosis. These also show strong correlation with the extent measures and, with a cursory examination, the fashion in which they are related is revealed. Excess kurtosis is a measure of how flat the distribution is (not strongly clustered around a single point), a pattern with a large extent is less likely to be strongly clustered⁸ as there is more space in which the dots can exist. The density is inversely proportional to the extent because it is the cardinality divided by the area of the bounding box. Consequently, the relationship between kurtosis and density is made clear; they are linked by their relationship to extent. The nearest neighbour descriptor shows a weak negative correlation with the dimensionality descriptors because the estimated nearest neighbour distance variance measures the uniformity of the patterns distribution, and the closer to collinearity a pattern is the less the variance of the distance between nearest neighbours is likely to be (Fig. 3.14(a)). However we can easily conceive of situations in which the nearest neighbour variance is high and the collinearity is high (Fig. 3.14(b)) or alternatively in which both are low (Fig. 3.14(c)). The exception case for high collinearity and high nearest neighbour variance is probably quite rare in real-world situations, however we suspect that the low nearest neighbour variance and low collinearity situation is not. Hence we do not remove the nearest neighbour variance in favour of a dimensionality measure or vice-versa. This does highlight a possible flaw in using heat maps in that they can show correlation that is only indicative of a trend in the test patterns, and as a result we tend to ignore any correlations that are only weakly negative or positive.

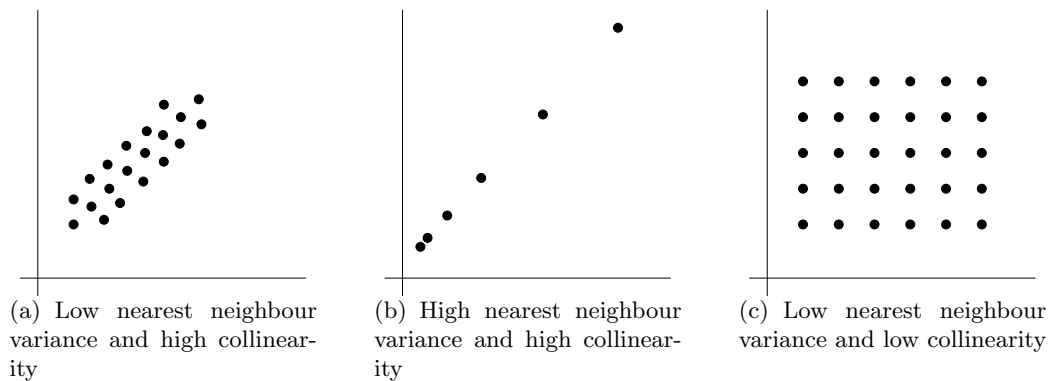
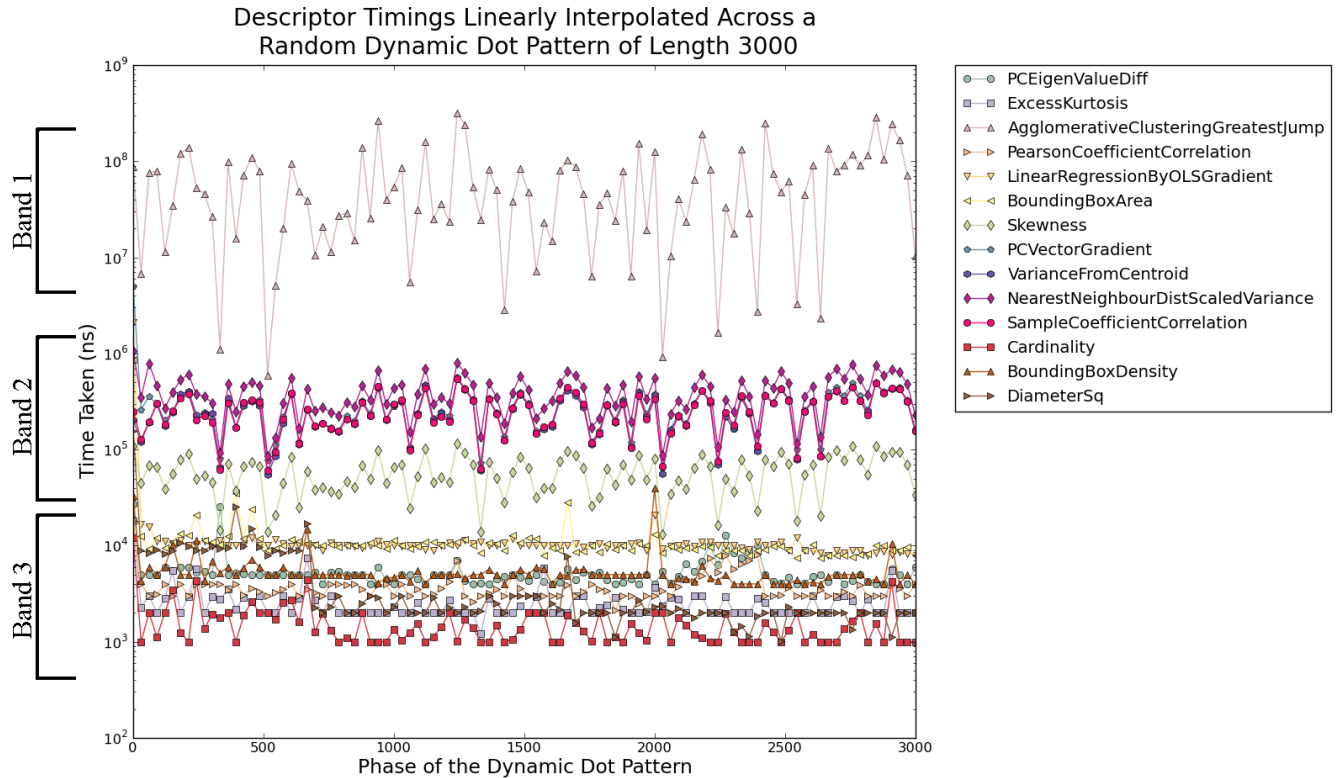


Figure 3.14. Examples of different values for estimated nearest neighbour variance and collinearity

⁸Although it is possible to conceive of a pattern with a high proportion of the dots in the centre and a few outliers increasing the extent.

The dimensionality descriptors all show strong correlation as might be expected. It should be noted that we are using the absolute values for the coefficient correlations and returning a value of 1 when the pattern is completely horizontal or vertical. Without these amendments the coefficient methods can return negative values relating to the slope of the pattern and will return a NaN (Not a Number) value for linear horizontal or vertical patterns. In fact if these amendments are not made then the sample and Pearson descriptors both show correlation with the orientation methods. The eigenvalue difference measure, however, will show how ‘linear’ the pattern is regardless of its orientation.

The kind of analysis performed with the heat map shows the correlation between two descriptors, which is similar to the *influence* correlation considered by Andrienko and Andrienko [Andrienko and Andrienko, 2007]. However it does not provide information similar to the *structure* correlation in which two descriptors would have to interact and correlate with a third. For example, descriptor $desc_1$ does not correlate with $desc_2$ or $desc_3$ but does with $desc_2 \cup desc_3$. This kind of correlation is not easy to identify but we examine how the selection of change identifiers, and thereby their descriptors, in Chapter 7.



With sets of descriptors that do not contain correlated measures the computation time for each set must be examined. Fig. 3.15 is a plot of the time taken in nano-seconds for each descriptor to compute a value for each phase in a randomised dynamic dot pattern set of length 3000 and a maximum pattern cardinality of ≈ 500 dots. For clarity the graphs have been plotted every 10 phases and then linearly interpolated.

There are three distinct ‘bands’ that are apparent in Fig. 3.15. Band 1 is the singleton

containing just the agglomerative clustering descriptor, by far the most computationally expensive of the descriptors as it has a long iterative process. Even so the average completion time for the agglomerative clustering for each timestep is only around five seconds. Five seconds is probably too long a time for it to be used as a change identifier but for a pattern of approximately five hundred dots this is not an unreasonable time in which to find clusters given the computational complexity of the task.

The second band contains the descriptors for average nearest neighbour distance, the sample co-efficient correlation, the variance of the distance from the centroid and the skewness. All of the measures require more than one pass through the pattern: the first to find the mean, standard deviation, etc.; the second to use the values from the first pass. This may account for their taking longer than those in the third band despite many of the third having the same theoretical complexities.

The third band contains the rest of the descriptors. It should be noted that some of these descriptors share information and therefore we cannot say that any descriptor within a band necessarily outperforms another within that same band as their values may depend partially on the order in which they are processed.

The three-part banding indicates the fastest descriptors but the correlations show that some measure the same aspects of the pattern. The 'best' descriptor set to use will have the fastest descriptors with no correlations. The following graphs are plots of time taken against timestep for each of the descriptor classes so that we may find the fastest in each:

Extent: Fig. 3.16.

Orientation: Fig. 3.17.

Connectedness: No figure required as it is a class containing just the agglomerative clustering descriptor.

Dimensionality: Fig. 3.18.

Dispersion: Fig. 3.19.

As has already been discussed, dispersion shows little in the way of correlation apart from the negative correlation of excess kurtosis with the bounding box density, but it has been included for completeness.

Fig. 3.20 shows the average time taken for each class against time steps. This is provided to give a sense of how similar the classes are in computation, apart from the class containing the agglomerative clustering descriptor.

With the information about the class speeds known we can create a set of the fastest completing descriptors from each class. This should give a good base set of descriptors on which we can construct change identifiers; completing in a minimum time while avoiding correlating, and therefore surplus, measures. Fig. 3.21 shows two sets using the fastest descriptors both with and without agglomerative clustering. For the change identifiers

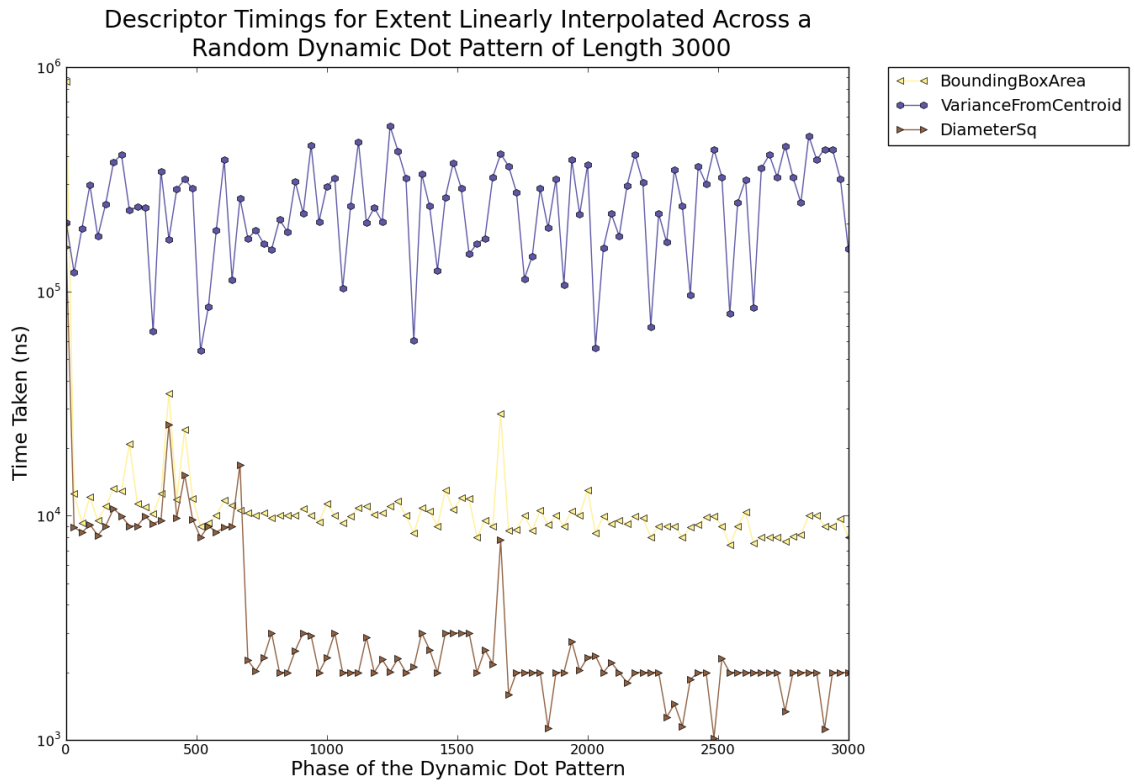


Figure 3.16. Time taken (nanoseconds) per dynamic dot pattern phase: Descriptors of extent

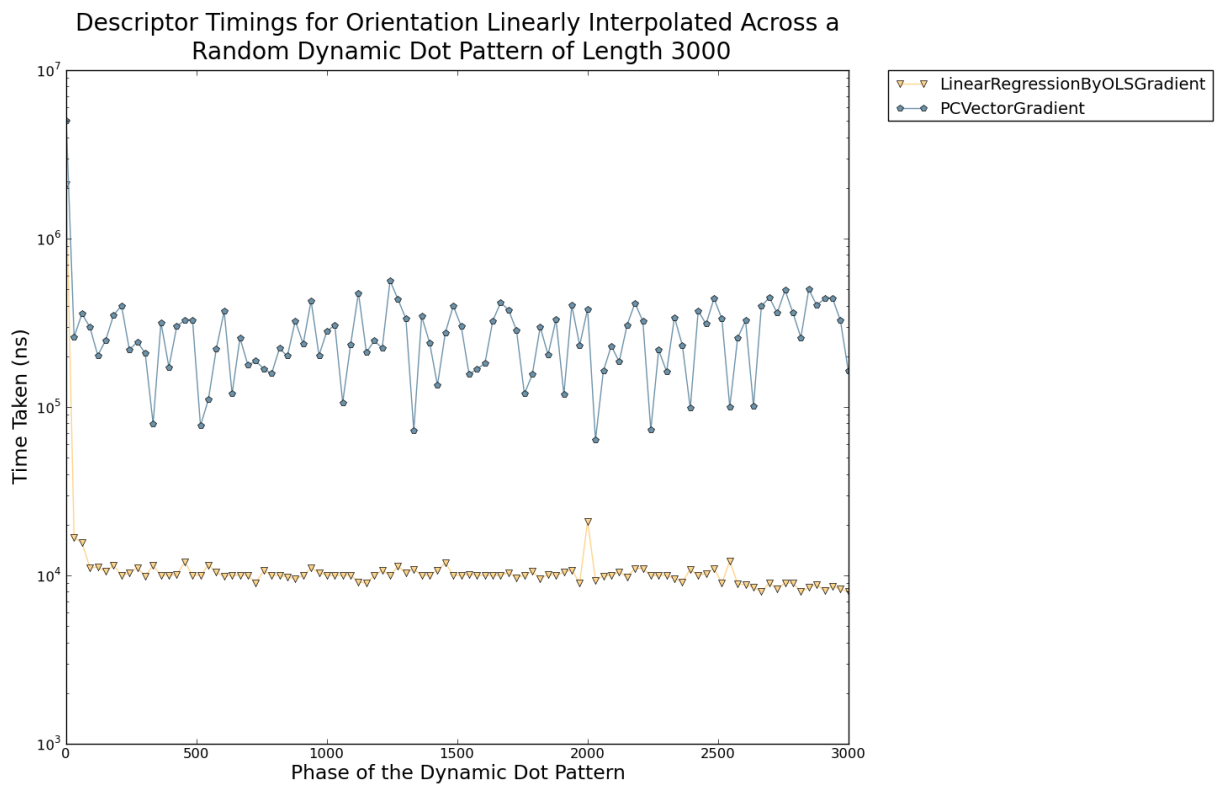


Figure 3.17. Time taken (nanoseconds) per dynamic dot pattern phase: Descriptors of orientation

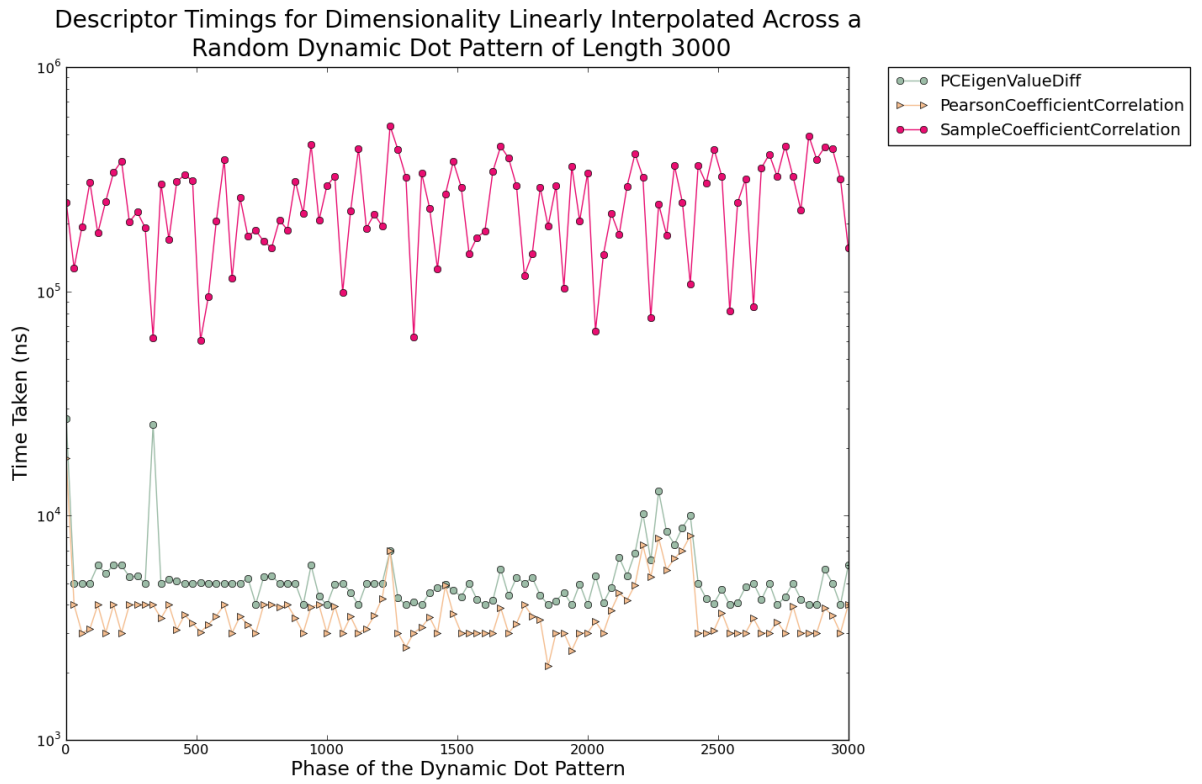


Figure 3.18. Time taken (nanoseconds) per dynamic dot pattern phase: Descriptors of dimensionality

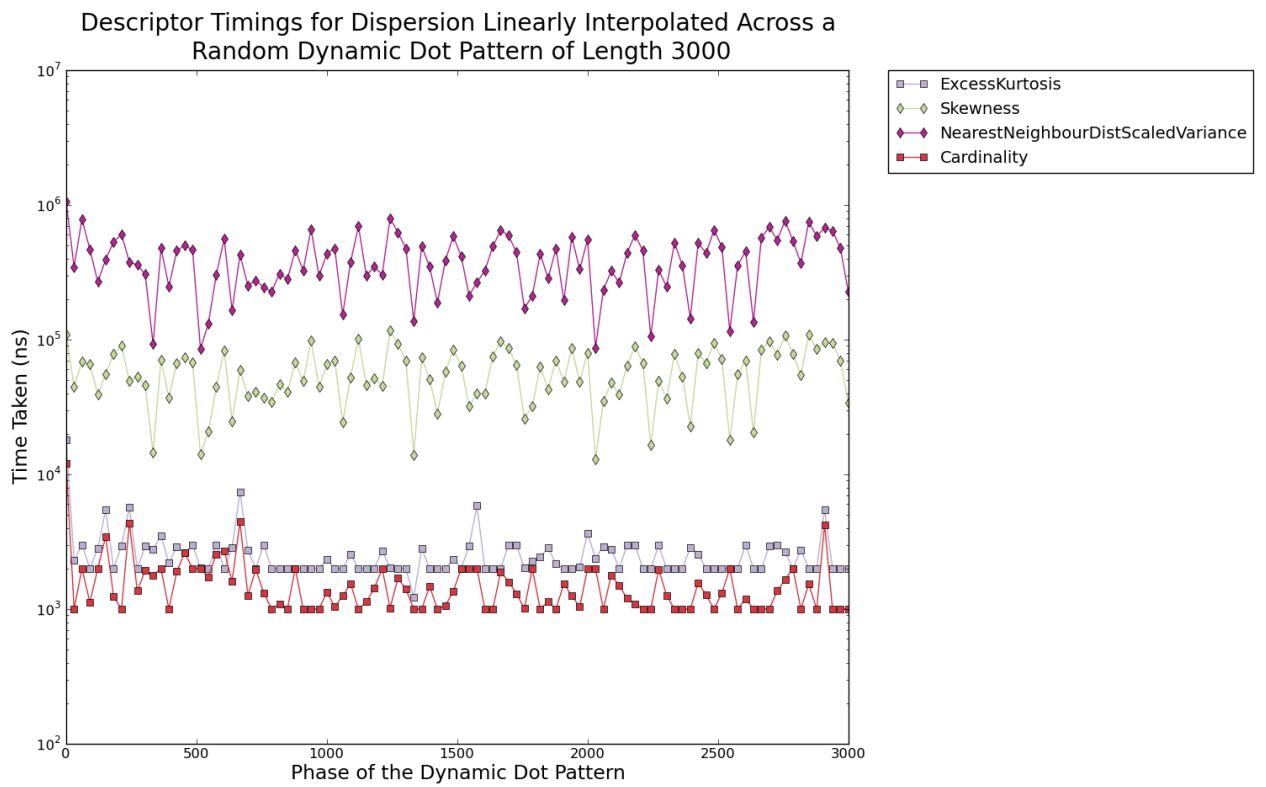


Figure 3.19. Time taken (nanoseconds) per dynamic dot pattern phase: Descriptors of dispersion

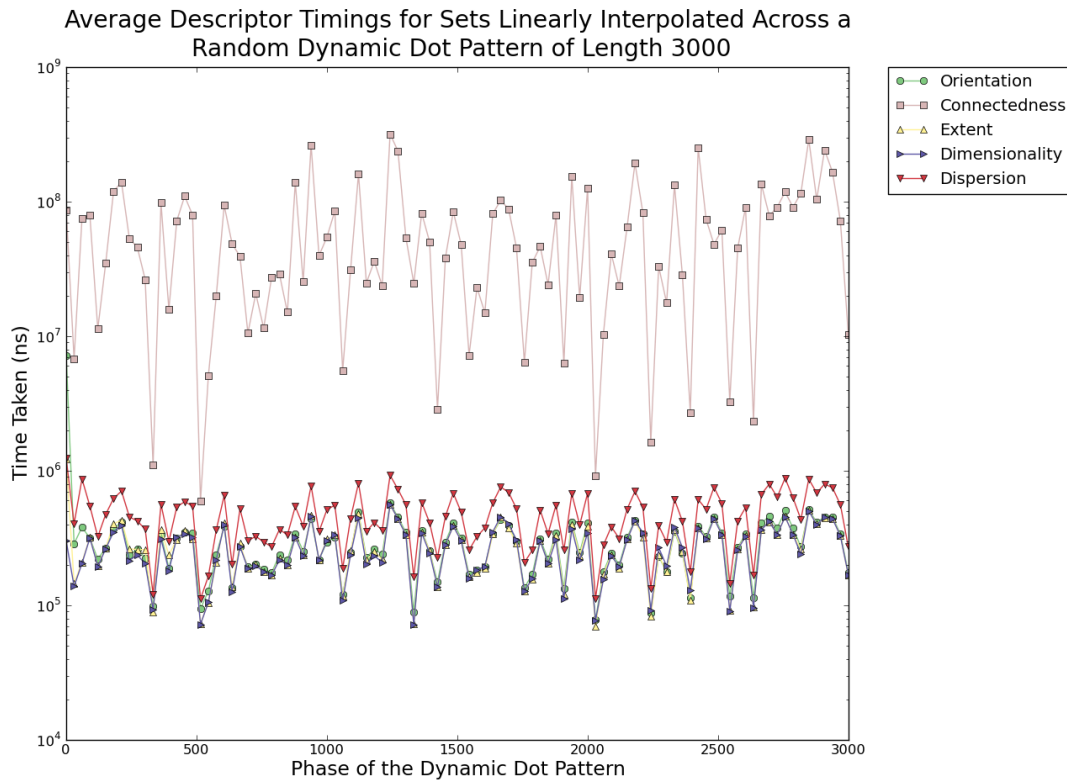


Figure 3.20. Average time taken (nanoseconds) per dynamic dot pattern phase: All classes

we will use only the set without the agglomerative clustering to avoid its high processing time.

3.5. Summary

This chapter has provided a list of descriptor classes and a set of non-correlated descriptors with a representative for each class. The descriptors have been timed and the fastest performing have been identified. It should be noted that speed is not the only indicator of quality for the descriptors. How fast they are to compute does not indicate the level at which they will be able to successfully identify change. To accurately measure this change indication ability requires us to use the descriptors in the change identifier framework. We will examine this in Chapter 5 and show results of experimentation in Chapter 7. We content ourselves here with measuring the facets of the descriptors that exist regardless of change.

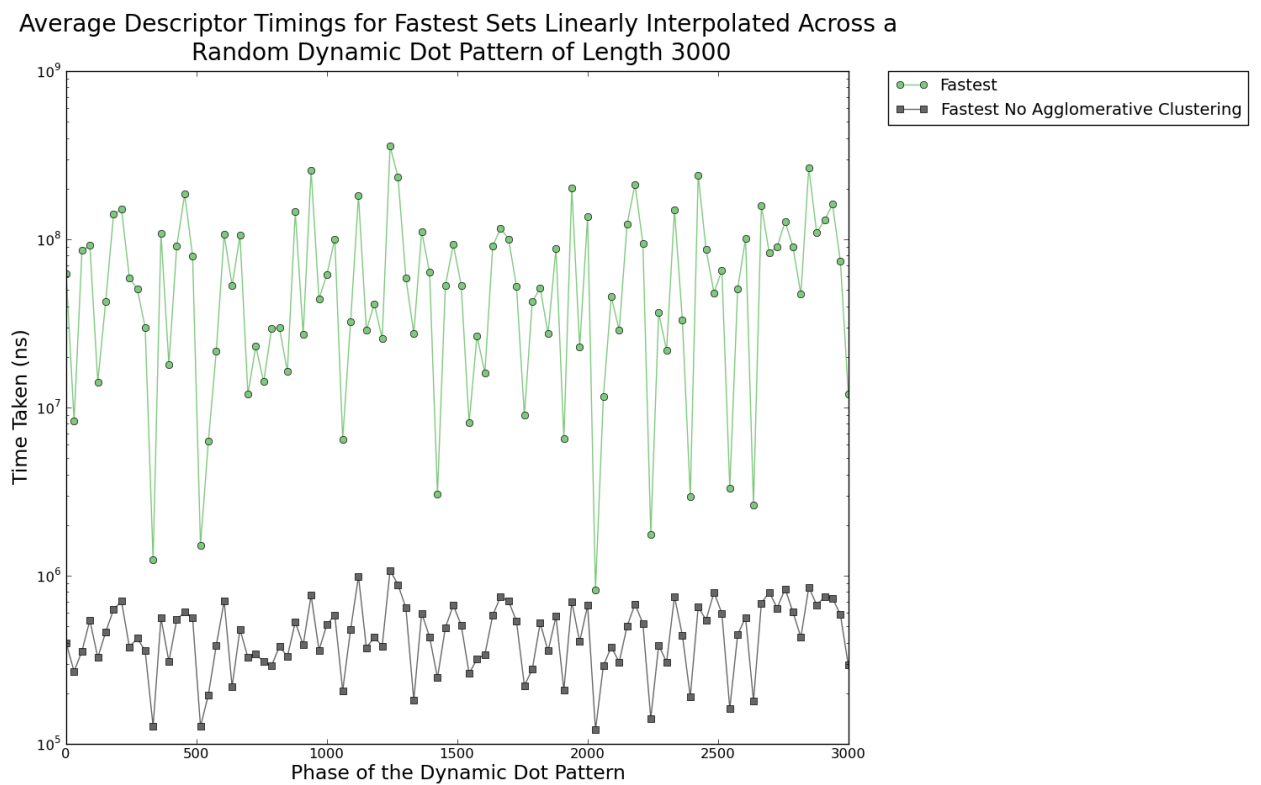


Figure 3.21. Time taken (nanoseconds) per dynamic dot pattern phase: Fastest non-correlated Descriptors

4. Footprints

The example use of change identifiers provided by this thesis is in the reduction of the number of updates of a footprint over a dynamic dot pattern. The previous chapter examined dot patterns and so, before discussing change identifiers, we must be sure we have a clear understanding of the nature of footprints.

The sheer number of algorithms described in the background chapter gives a good indication of the myriad applications in which footprints are required. These applications range from mapping molecular structure [Edelsbrunner and Mücke, 1992] to region approximation for geographic gazetteers [Alani et al., 2001]. Some of the papers make no mention of a specific application; indicating that their algorithm is intended to be applicable to a range of different problems (e.g., Jarvis March [Jarvis, 1973] and χ -hull [Duckham et al., 2008]). Given this ‘broad strokes’ approach taken across the field it is perhaps not surprising that there is a dearth of material that assesses why a footprint may be a ‘good’ footprint for any given application. This chapter provides a discussion on what a footprint is and a classification of the types of footprint that appear in the literature.

4.1. What is a footprint?

The previous chapters in this thesis have avoided formally defining a set of axioms to describe what is and, perhaps more importantly, what is not a footprint. This wariness has arisen from the desire to not exclude any of the shapes created by existing footprint algorithms. However without a clear distinction between footprints and any other entity with which to describe a dot pattern it will be difficult to select a measure of error when considering how well the footprint has been maintained over a dynamic dot pattern.

Difficulties arise from the large number of applications in which spatio-temporal patterns are used as the base on which to generate characterisations of application-interest. With so many different possible requirements how can it be claimed that any interpretation, no matter how unusual, is invalid as a footprint? When considering this question we should bear in mind the driving problem for this thesis: To be able to identify when it would be most appropriate to update a footprint by looking at the differences between two dot patterns. For this to be possible the footprint definition must relate the difference between two dot patterns to the difference between their footprints. It should be noted that for the following discussion, when footprint difference and dot pattern difference, are discussed we assume the existence of some ‘perfect’ difference measure that captures all aspects of

difference. This is so that the examination of the relation of dot pattern differences to footprint difference does not get mired in an analysis of difference measures.

Obvious constraints that can be examined are that of monotonicity and that of continuity of difference values for the dot patterns leading to continuity in the differences between the footprints.

By monotonicity we mean that the footprint difference should be strictly monotonically increasing with the dot pattern difference, and at first glance such a requirement seems sensible; giving assurance to the validity of the use of change identifiers to indicate when to update a footprint. However even this simple specification falls prey to the nebulous nature of the footprint. Consider a convex hull algorithm, should a dot pattern change such that its extremal points in the plane remain unchanged then the convex hull will also remain the same (see Fig. 4.1). Given the number of examples of such algorithms within the literature, and their obvious usefulness in describing the region inhabited by a dot pattern, it cannot be said that convex hull algorithms are not footprint algorithms, therefore we cannot require that the footprint difference is monotonic to the dot pattern difference.

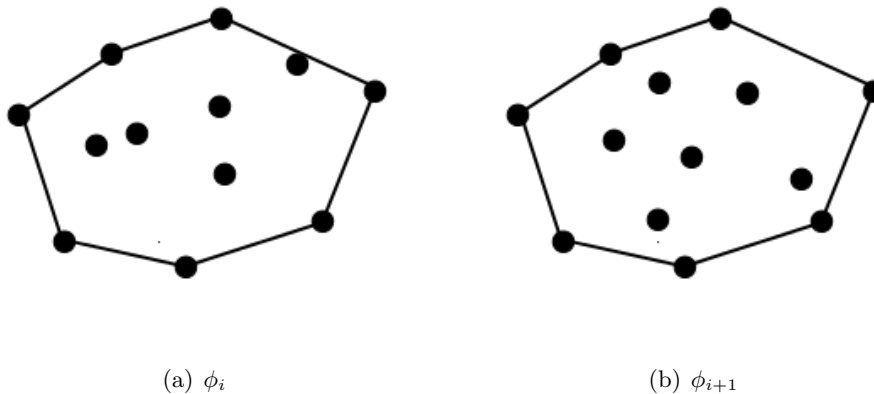


Figure 4.1. Example of the convex hull remaining the same despite differences between the dot patterns.

Since the footprint and dot pattern difference does not increase strictly monotonically a relaxed version of the requirement can be examined: That the difference should never vary inversely. Despite the apparent simplicity of this requirement we can envision a situation in which three subsequent dot patterns decrease in difference while their respective footprints show an increase. An example of this inverse change in difference can be seen in Fig. 4.2.

Having shown that we can not require that footprint and dot pattern difference be monotonic we can instead consider continuity. Continuity is a desirable property because if two adjacent phases of a dynamic dot pattern exhibit only small difference but their footprints exhibit a large one then the change identifiers may not be able to indicate that a footprint update is required. Considering that a dynamic dot pattern consists of discrete phases there is no way to be sure of continuity in dot pattern difference, the closest that can be achieved is consistent small differences relative to the pattern extent. For most of the

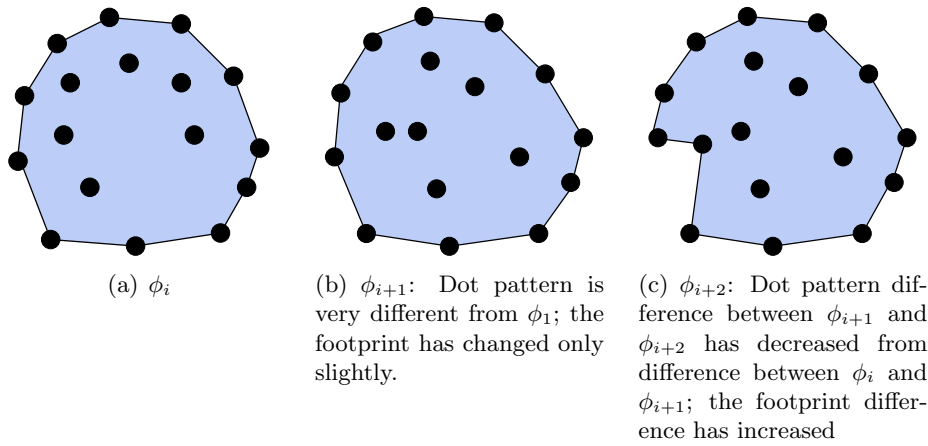


Figure 4.2. Example of pattern difference decreasing while footprint difference increases.

examples considered within this thesis the dynamic dot patterns show only these small differences. However the footprint differences need not be similarly small, the Swinging Arm algorithm [Galton and Duckham, 2006], for example, can create footprints that differ greatly between timesteps when the dot patterns differ only slightly (see Fig. 4.3).

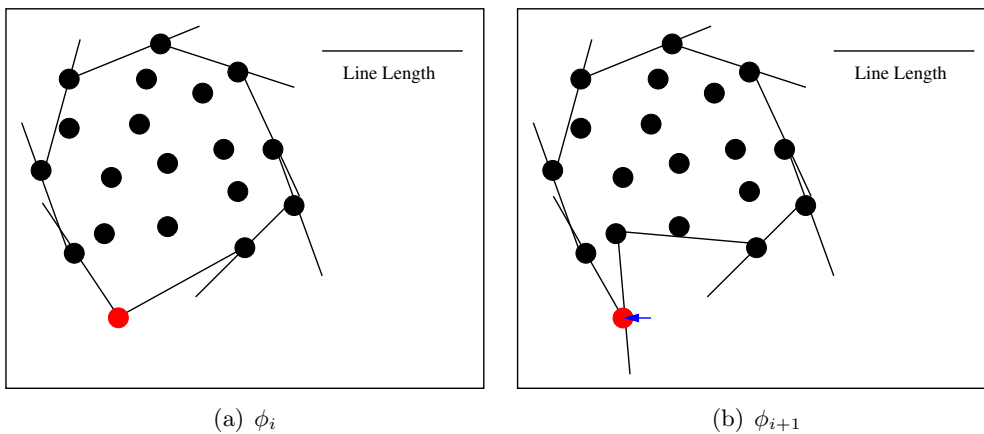


Figure 4.3. Example of small difference between dot patterns leading to large difference between footprints created by the Swinging Arm algorithm. The red dot is further to the left in ϕ_{i+1} and, as such, the line length is no longer long enough to reach it.

A recurrent problem is that different footprint algorithms are sensitive to varying aspects of difference between the footprints. For example a convex hull algorithm is sensitive to differences in pattern extent but ignores the internal dots, whereas the χ -hull can be entirely altered by internal differences. In fact there appears to be a loose hierarchy of footprint types: differences that change a non-convex footprint can leave the convex hull intact, and differences that change a convex-hull can leave the bounding box intact. However if a footprint is allowed outliers (dots outside of the footprint region(s)) then both the bounding box (and therefore the convex hull) can differ while leaving the footprint with outliers unaltered.

So far the discussion has looked at the requirements that can be made on the footprint algorithms against the cumulative differences of the patterns. An alternative approach is use the individual aspects of the patterns to relate a dot pattern to its footprint

- **Scale:** If the dot pattern is magnified then the footprint is magnified by the same factor.
- **Translation:** If the dot pattern is shifted then the footprint is shifted by the same vector.

Unfortunately similar statements cannot be made for rotation or reflection as some of the algorithms are not invariant with respect to these translations when applied to the dot patterns. For example the Swinging Arm algorithm can produce two different footprints for the dot patterns DP and DP' where DP' is a reflection of DP .

4.1.1. A Troublesome Example

There is a type of footprint algorithm which disrupts any attempt at a strict footprint definition. Consider a footprint algorithm that uses the dot pattern as a seed to a stochastic process¹. The footprints produced by such an algorithm will vary wildly for small differences between the dot patterns. We can even place restrictions on the algorithm such that the footprints it produces remain in a fixed position relative to the dot pattern and that its convex hull is proportional to the convex hull of the dot pattern. The algorithm now produces footprints which fit our loose requirements for scale and translation but can still produce vastly different footprints for patterns that differ only slightly. Algorithms such as the above are valid footprint algorithms in the respect that our requirements do not discount them but are clearly not particularly useful characterisations of the dot pattern. It appears that the question of the validity of any particular footprint is fated to rely on a specification that can be only either too stringent or too relaxed.

As mentioned earlier, there are myriad applications in which footprint algorithms are used, and it is only the intention of the users of these algorithms for the result to be a representation of the pattern that makes them footprint algorithms. We would expect that any algorithm could be considered a footprint algorithm if the intention behind its use is to produce an output that provides a ‘useful’ (for that application) description of the dot pattern. However such reasoning becomes dangerously vague; treading close to a tautology and defining a footprint as an entity which is being used as a footprint.

If intention of use is so important to the definition of a footprint algorithm then we may be able to tighten the boundaries of a footprint definition by considering the requirements that need to be in place for this thesis to produce experimental results. We shall state that, for this thesis, footprints are bounded regions; with the caveat that these bounded regions may contain degeneracies. Degeneracies are sections of the footprint with no area (line segments or points) that can take the form of separate components of the footprint or as branches off of a region (see Fig. 4.4)

It may seem strange to allow degeneracies when a ‘pure’ bounded region seems like a more useful requirement. However there are dot patterns, like that shown in Fig. 4.4, for which

¹This example was suggested by Richard Everson.

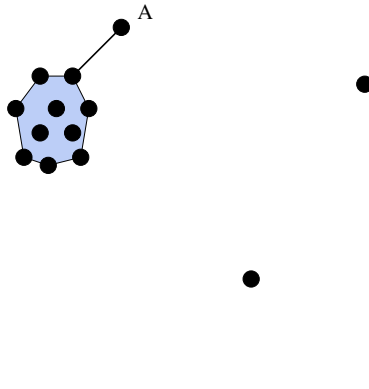


Figure 4.4. Example of a dot pattern in which a degenerate line may be desirable.

a degenerate line may be part of an accurate characterisation of the dot pattern. In such cases algorithms like the Swinging Arm algorithm and the α -shape [Edelsbrunner et al., 1983] can (depending on the parameter used) identify dot A as being close enough to its neighbours so as to be qualitatively different from the outliers, while being far enough from the connected region for the restriction on line length to preclude it making up a part of that region. For an application trying to identify collectives (e.g. movements of groups of people through a shopping centre) such dots may well be important. Arising from this region restriction is the fact that we can ignore internal partitions of the footprint², for example the Delaunay triangulation is not a footprint but the region enclosed within the union of its cells is.

We will also state that, for the change identifiers to be effective, we are looking at only the class of footprint algorithms that produce footprints for which the differences are *globally monotonic* with the dot pattern difference. Globally monotonic allows for small ‘departures’ from monotonicity if the majority of the differences are monotonic. This is a slightly vague term but a threshold cannot be placed on the proportion of allowed departures without it being entirely arbitrary. In essence, the more monotonically increasing the difference for the footprint is with the dot pattern difference the better.

Further to restricting the footprint to a bounded region we also note that the existing literature has a strong tendency to use footprints that are planar, and it would be tendentious to start going too far beyond that which is required by the current applications.

Such a set of considerations, while not particularly strict, allows us to make some assumptions about the type of footprints that will be produced in the experimentation and therefore the analysis that can be performed on them.

4.1.2. Unique Footprints

There are some footprint types for a pattern which are distinctive. Any footprint with a specification that is unique on a particular dot pattern is, at least intuitively, different to a general footprint. Perhaps this is easier understood by the observation that there are

²Although not its cavities.

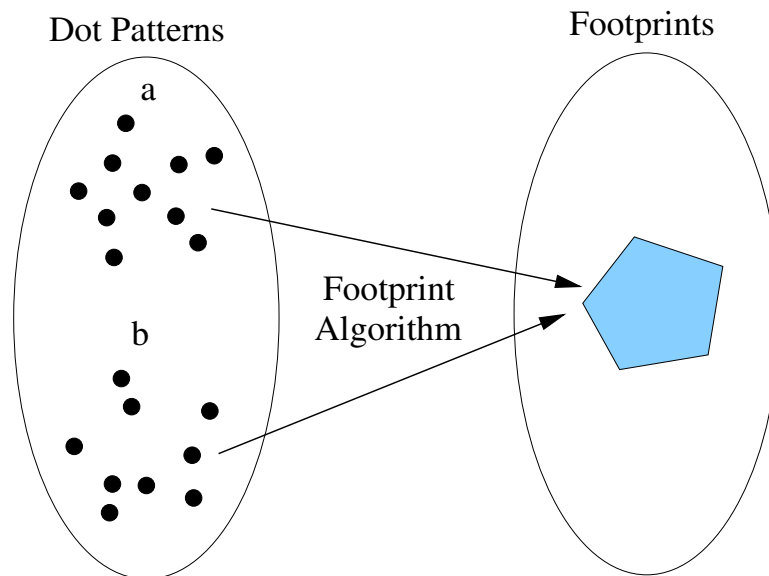


Figure 4.5. Figure showing that a footprint algorithm is not injective, both dot patterns a and b map to the same footprint.

some footprints which are named and the name uniquely defines them for a pattern, for example the convex hull, the minimum bounding disc and the minimum bounding rectangle. We add the *identity* footprint to these as the footprint for which each dot is a distinct component of the footprint with no extent; in effect the dot pattern itself³. It may be that such footprints are only unique in that they fit with some human understanding, for example why are they any different from a footprint named “The α -hull for an α value of 0.1”? Indeed, if we rename such a footprint the 0.1-hull we see that there is no real mathematical distinction. However the 0.1-hull is not an obviously useful footprint, whereas the others commonly appear in the literature (e.g. the convex hull) or are intuitively different (e.g. the identity footprint). Hence human intuition appears to be the key factor in distinguishing between unique and non-unique footprints. Although we also note that the unique footprints we give as examples are all minimally bounding with respect to the dots and that this may play an important part in why we consider them as qualitatively different from the 0.1-hull.

4.1.3. Information Content

As a representation of the pattern the information content of the footprint is a measure of how well it defines that pattern. Note that this says nothing about the quality of the footprint which, as discussed, is difficult to judge, but is more an indication of how well the dot pattern could be re-created given just the footprint. We draw a distinction between the information content of the footprint and that of the dot pattern; while the footprint is created from the pattern, the mappings implied by the footprint algorithms are not injective (Fig. 4.5). There is a loss of information in that, even given the algorithm (and parameter if required) which created it, no footprint uniquely defines a pattern (except

³While a region-less footprint may not be particularly practical it is a useful conception to discuss some footprint properties.

the identity footprint). However, given a specific algorithm with determined parameters a pattern uniquely defines one footprint. Importantly it should be noted that the loss of information is not the same as having less information. The number of bits of data required to draw the footprint is not representative of the ability to retrieve the dot pattern from the footprint and can be greater than that required to draw the dot pattern. Yet, for most sensible examples⁴, how well the dot pattern information can be estimated from a footprint is often related to the number of bits required to draw it. For example a sample of points within the footprint of Fig. 4.6(c) has a better chance of being similar to the actual dot pattern than a sample of points within the footprint of Fig. 4.6(a). While we have been careful to show examples where a footprint specification may require greater memory space than a dot pattern, for dot patterns with areas of high density the data content of the footprint is unlikely to be greater than that of the dots; as the density increases so does the likelihood that fewer dots are on the boundary of the footprint. Some footprint algorithms (e.g., DSAM [Alani et al., 2001]) are created explicitly to produce a representation with a lower memory requirement than the dot pattern. Therefore for most real-world applications in which footprints may need to be found it is likely that footprints with less data content than that of the dot pattern is preferred. Fig. 4.6 shows the differences in information content that different footprints on the same pattern can have.

The complexity of a footprint algorithm is loosely tied to the information in the footprint it produces; as the amount of information in the footprint increases there will be a tendency for the computational complexity of the algorithm to increase. Largely this is due to the need for iterative processes and validity checks. For example the χ -hull is more computationally complex than the Jarvis March/Gift-Wrapping algorithm which is, in turn, more computationally complex than an algorithm for defining the isothetic minimum bounding box. This is not a true relation but a tendency of the algorithms in the literature; some algorithms have low information content but a high complexity, for example an algorithm to find the minimum bounding disc. An algorithm to reproduce a dot pattern need only record the location for each dot, and as such, footprint algorithms are generally far more complex. While it is possible to come up with footprint algorithms less computationally complex than a dot pattern recreation algorithm, this tends to involve having a footprint algorithm ignore the dot pattern (e.g. it simply draws a square with a set length with its top left corner being the first dot it encounters), or by adding wasteful steps to the dot pattern algorithm. As a final point on information and how it applies to footprints, we note that an algorithm to define the identity footprint produces a footprint with identical information to that of the dot pattern (Fig. 4.6(f)) with a computational complexity of $O(n)$; it simply needs to record each dot location. Such an algorithm is bijective (as a dot pattern has only one identity footprint and an identity footprint has only one dot pattern) and therefore unusual amongst the footprint algorithms found in the literature.

⁴It is easy to add extra information to a footprint without increasing the ability to retrieve the dot pattern by adding lines that render the footprint a worse representation

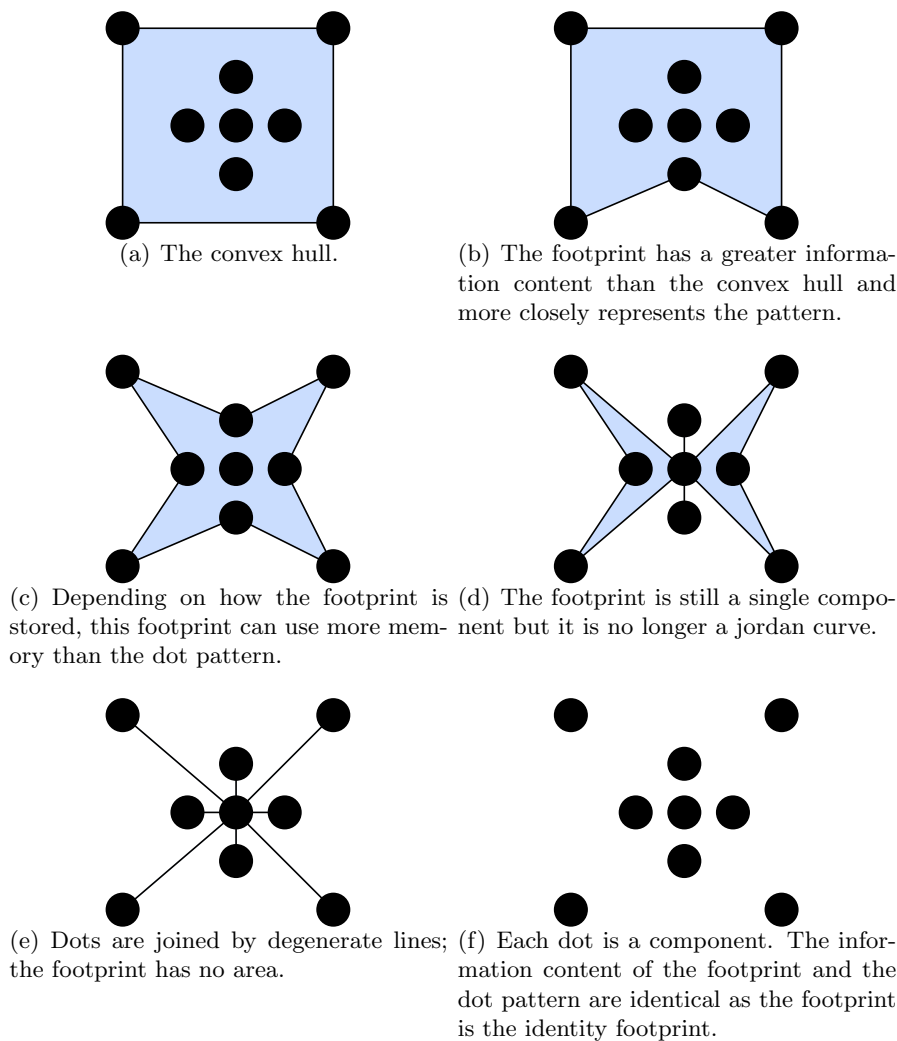


Figure 4.6. Changes in information content for a footprint over a small dot pattern

4.2. Footprint Classification

In the background chapter a summary was given of work performed by Galton and Duckham [Galton and Duckham, 2006] and Galton [Galton, 2008]. These papers provide an insight into some of the aspects that need to be addressed when assessing the quality of an algorithm. Building on the work by Galton and Duckham, this author and Galton produced a classification [Dupenois and Galton, 2009] by which to draw distinctions between different footprint types. The footprints are delineated into classes with the aim of being able to provide reasons why they might, or might not, be suited to a specific application. This section of this chapter will discuss and extend the footprint classification.

The classifications have been split into two sections: *intrinsic*, concerning the footprint independent of the dots, and *relational*, examining the relationship between the dots and the footprint. When the footprint is expected to change, the delineation between intrinsic and relational classifiers becomes more distinct. A change in any of the intrinsic values would indicate a complete change of footprint type. Any relational change is minor and to be expected when using change identifiers, for example whether or not all dots are within the boundary of the footprint is almost certain to change when using a dynamic dot

pattern. It should be noted that the classification is not an exhaustive taxonomy of shape but showcases the features drawn from the common differences between the footprints created by the algorithms in the existing literature.

4.2.1. Intrinsic footprint criteria

[C] Connected: *The footprint consists of a single connected component.*

Figure 4.7 shows examples of connected and disconnected footprints for the same dot pattern. Some algorithms will always generate a single connected component, implicitly assuming that any clustering has been done beforehand, with the algorithm being applied to individual clusters (e.g., Concave Hull [Moreira and Santos, 2007], χ -shape [Duckham et al., 2008]); others can yield footprints with multiple components (e.g., Swinging Arm [Galton and Duckham, 2006]). The desirability or otherwise of multiple components is application-dependent, e.g., if only connected footprints are appropriate, use an algorithm guaranteed to produce such components.

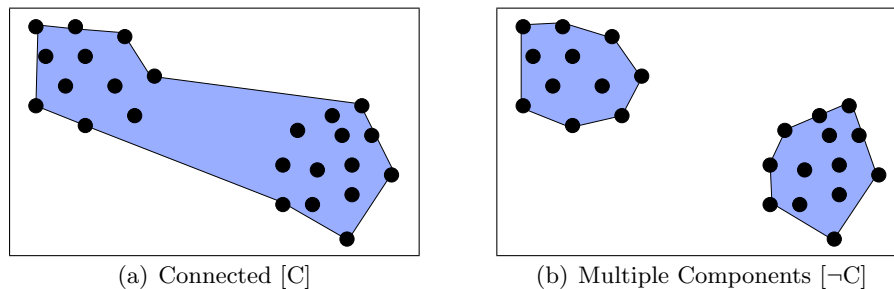


Figure 4.7. Connectedness examples

[R] Regular: *The footprint is topologically regular.*

Assuming the footprint is topologically closed, this criterion tells us whether or not the footprint contains boundary elements that do not bound the footprint's interior, such as the linear 'spike' in Fig. 4.8(c) or the isolated linear component in Fig. 4.8(b).

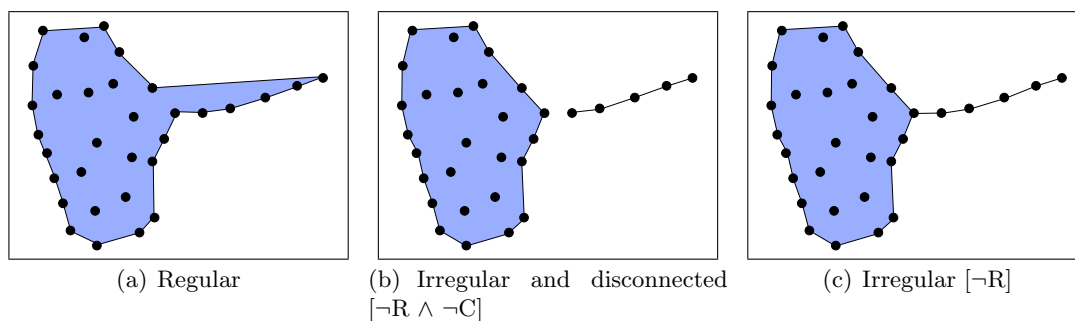


Figure 4.8. Regular

[P] Polygonal: *The boundary of the footprint is made up of only straight lines.*

For a polygonal footprint the boundary is made up entirely of straight line-segments as opposed to curves. (Fig. 4.9).

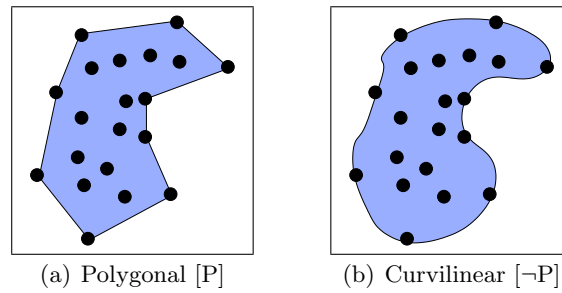


Figure 4.9. Polygonal

[**JC**] Jordan Components: *Each component of the footprint has a Jordan boundary.*

A Jordan boundary is a boundary which is a Jordan curve, i.e., homeomorphic to a circle. Such a boundary does not meet itself, so it is possible to traverse the entire boundary passing through each of its points only once. (Fig. 4.10(a)). In Fig. 4.10(b) the component with a non-Jordan boundary is represented as a ‘bow tie’ shape; of course this is not the only way the Jordan property can fail.⁵

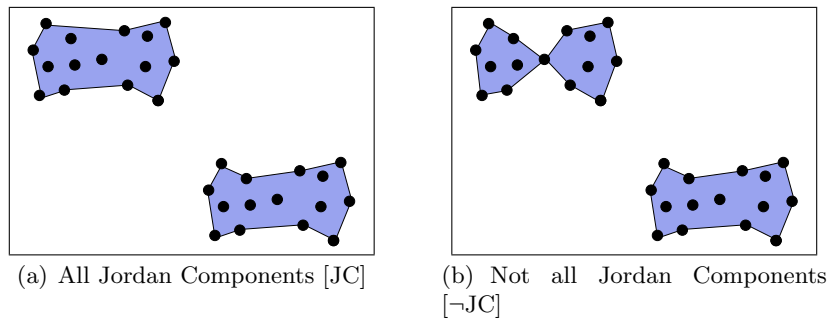


Figure 4.10. Jordan Boundary

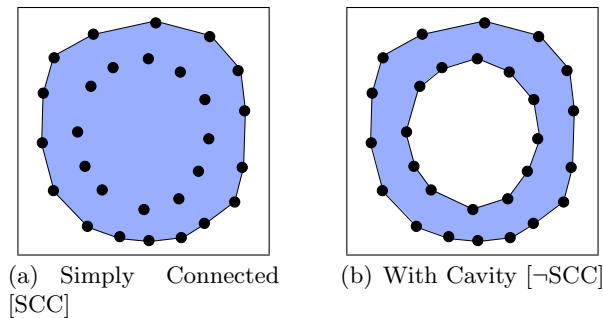


Figure 4.11. Simply Connected

[**SCC**] Simply Connected Components: *Each component of the footprint is simply connected.*

A component that is not simply connected contains a ‘hole’ (Fig. 4.11(b)). In two dimensions this means that the boundary is disconnected, with one of the boundary components facing the ‘outside’, and each other component bounding an internal cavity.⁶

⁵In relation to the ‘bow-tie’ configuration, if the footprint is formed by tracing out its boundary, then the constriction point may be either a *self-intersection*, where the boundary actually crosses itself, or a *pinch point*, where the boundary touches itself without crossing. An intersection or pinch-point may or may not occur on one of the dots; examination of the algorithms suggests that a self-intersection is more likely to occur away from a dot, whereas the opposite is true for a pinch point.

⁶In three dimensions there are more varieties of connectivity to consider, e.g., the distinction between

4.2.2. Relational footprint criteria

[CED] Curvature Extrema at Dots: *All curvature extrema of the footprint boundary coincide with dots.*

Very often a footprint is constructed by tracing its boundary through some or (more rarely) all the dots of the dot pattern. In such cases it is typical for the dots to mark curvature extrema of the outline; this is the normal situation when the outline is polygonal, with the dots at its vertices (Fig. 4.12(a)), and is always found in the case of the convex hull.

Note that this criterion is independent of whether all, some, or none of the dots occur on the boundary (which is given by criteria [ADB] and [NDB] introduced next), as shown by Fig. 4.12, where each value for one criterion can co-occur with each value of the other. However, $[CED] \wedge [NDB]$ (all curvature extrema are dots and all dots are off the boundary) can only be true if the footprint is circular, in which case there are no curvature extrema, so [CED] is true by default.

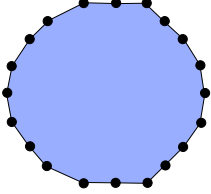
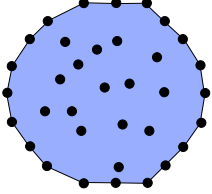
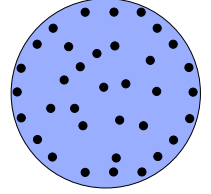
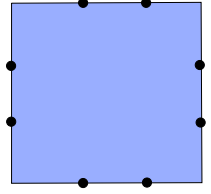
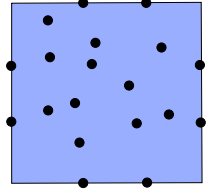
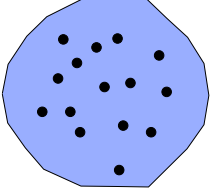
	ADB	$\neg ADB \wedge \neg NDB$	NDB
CED			
$\neg CED$			

Figure 4.12. Curvature Extrema and Dots On/Off Boundary

[ADB] All Dots on Boundary: *All of the dots lie on the boundary of the footprint.*

In general we would not expect footprints to satisfy this criterion, but in some applications the dots are specifically intended to represent boundary points, and in such cases this criterion is appropriate. As mentioned above [ADB] is linked to, but distinct from, whether or not the curvature extrema coincide with dots (Fig. 4.12).

[NDB] No Dots on Boundary: *None of the dots lie on the boundary of the footprint.*

Criteria [ADB] and [NDB] cannot be simultaneously satisfied, thus they are not independent. As with [ADB], [NDB] is linked to, but distinct from, whether or not the curvature extrema coincide with dots (Fig. 4.12) as both [NDB] and [CED] can only be true for a circular footprint. Some algorithms (e.g., the Voronoi-based method of [Alani et al., 2001]) create footprints by amalgamating ‘areas of influence’ surrounding the dots. In such cases the dots typically all lie in the interior of the footprint, and hence off the boundary.

[FC] Full Coverage: *All of the dots are included in the closure of the footprint.* It is

an internal cavity and a perforation. For simplicity (and because the majority of algorithms are in 2-dimensions) we do not discuss these extensions here.

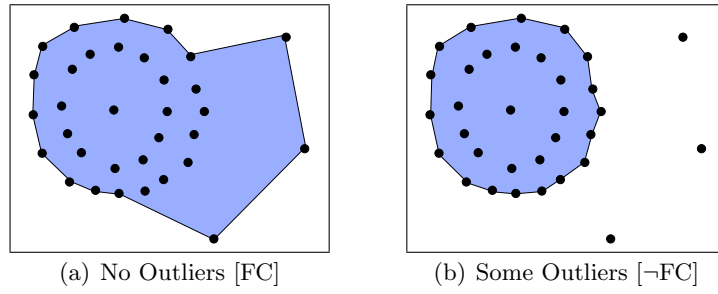


Figure 4.13. Full Coverage

possible that a footprint algorithm may be able to distinguish certain dots from the pattern as ‘noise’, and as such it may wish to exclude them from the footprint. We call such dots *outliers* (Fig. 4.13).

4.3. Using the Footprint Classification

With a set of criteria by which to classify the footprints a nomenclature can be created to easily distinguish between individual footprints. When classifying a footprint the values can be written in the form presented below:

$$\begin{aligned}
 T_x &: \text{Footprint Type (intrinsic) } x \\
 F_\tau &: \text{Footprint at time } \tau \\
 I(v, x) &: \text{Value of intrinsic classifier } v \text{ for type } x \\
 R(v, \tau) &: \text{Value of relational classifier } v \text{ at time } \tau
 \end{aligned}$$

Note that we are using F to denote a footprint instead of *foot* to draw a distinction between the footprint classification and the footprint function.

As discussed earlier, the intrinsic classifiers are the more ‘concrete’ classifiers and the relational are more prone to change with time. This distinction can be used to produce a listing to describe the footprint at a given time:

$$\begin{aligned}
 T_1 &= \{I(C, 1), I(R, 1), I(P, 1), I(JC, 1), I(SCC, 1)\} \\
 F_\tau &= \{T_1, R(CED, 1), R(ADB, 0), R(NDB, 0), R(FC, 1)\}
 \end{aligned}$$

A shorthand representation can be created by requiring that the intrinsic and relational classifiers always appear in the given order $\{C, R, P, JC, SCC\}$ and $\{CED, ADB, NDB, FC\}$ respectively:

$$\begin{aligned}
 T_1 &= \{1, 1, 1, 1, 1\} \\
 F_{\tau_x} &= \{T_1, 1, 0, 0, 1\}
 \end{aligned}$$

The nomenclature allows the tracking of the footprint type over a dynamic dot pattern and an example of this is shown in Fig. 4.14. If $T_1 = \{1, 1, 1, 1, 1\}$ then at dot pattern phase ϕ_0 (Fig. 4.14(a)) the footprint is of type $\{T_1, 1, 0, 0, 1\}$. The full tracking of the footprint classifications of Fig. 4.14 are shown in Table 4.1.

Intrinsic Type	Classification
T_1	$\{1, 1, 1, 1, 1\}$
T_2	$\{0, 1, 1, 1, 1\}$

Fig. Ref.	Phase	Classification
Fig. 4.14(a)	ϕ_0	$\{T_1, 1, 0, 0, 1\}$
Fig. 4.14(b)	ϕ_1	$\{T_1, 0, 0, 0, 0\}$
Fig. 4.14(c)	ϕ_2	$\{T_2, 1, 0, 0, 1\}$

Table 4.1. Table showing classification of Fig. 4.14

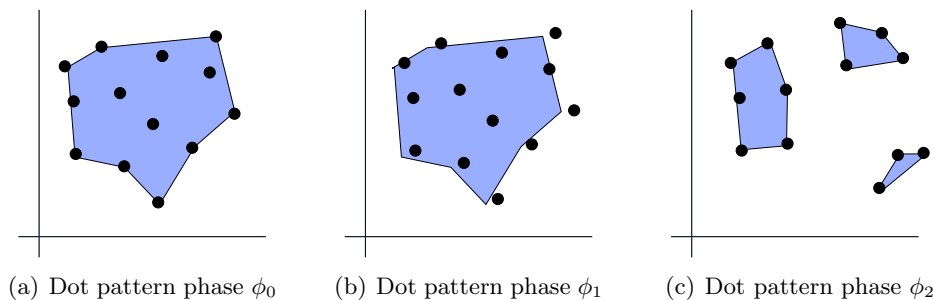


Figure 4.14. Footprint type tracking example.

Before further examining the application of the classification to dynamic dot patterns we look at what is perhaps a more generally useful aspect of the classification. As previously mentioned, there are a large number of algorithms for multiple applications. The algorithms, by virtue of their construction, are limited in the footprints they can produce for any given dot pattern⁷. We can, at least in part, classify the footprint algorithms by the footprints they produce⁸. Instead of the true/false value, used by the footprint classification, the algorithms require a ternary system where -1 , 0 and 1 indicate that classifier never holds true, sometimes holds true and always holds true respectively. We introduce the shorthand of [algorithm] is [property] to indicate that the footprints produced by that algorithm have that property. For example the Swinging Arm algorithm ([Galton and Duckham, 2006]) is always regular ($I(R, 1)$ in the above nomenclature). More accurately the Swinging Arm algorithm is regular except when the dots of the pattern are collinear; the correct value, then, should be 0 . Other algorithms have similar changes in value for particular pattern arrangements and the value system becomes worthless if all the values are 0 so we add two special case values: -1^+ if the classifier is never true for ‘almost all’ patterns, where ‘almost all’ excludes specific and unlikely to occur pattern arrangements (e.g., collinearity, the null pattern⁹ and a pattern with only one dot); and 1^- for when the classifier is always true for ‘almost all’ patterns. Some examples of this nomenclature can

⁷Some algorithms can produce an infinite number of footprints for a given dot pattern (e.g., α -shapes [Edelsbrunner et al., 1983]), however they cannot necessarily produce all possible footprints.

⁸A full classification would describe their complexities, pre-processing requirements and perhaps some description of their running process.

⁹An empty pattern.

be found in Table 4.2¹⁰.

Algorithm Examples	C	R	P	JC	SCC	CED	ADB	NDB	FC
<i>Jarvis March</i> [Jarvis, 1973]	1	1 ⁻	1	1 ⁻	1	1	0	-1	1
<i>Swing. Arm</i> [Galton and Duckham, 2006]	0	1 ⁻	1	1 ⁻	1	1	0	-1	1
<i>Close Pairs</i> [Galton and Duckham, 2006]	0	1 ⁻	1	1 ⁻	1	1	0	-1	1
α - <i>shape</i> [Edelsbrunner et al., 1983]	0	1 ⁻	1	1 ⁻	0	1	0	-1	0
α - <i>hull</i> [Edelsbrunner et al., 1983]	0	1 ⁻	0	1 ⁻	0	1	0	-1	0
\mathcal{A} - <i>shape</i> [Melkemi and Djebali, 2001]	1	1	1	1	0	-1	-1	1	1
<i>DSAM</i> [Alani et al., 2001]	1	1	1	1	1	-1	-1	1	1
χ - <i>hull</i> [Duckham et al., 2008]	1	1 ⁻	1	1 ⁻	1	1	0	-1	1
<i>s-shape</i> [Chaudhuri et al., 1997]	0	1	1	1	0	0	0	0	1
<i>r-shape</i> [Chaudhuri et al., 1997]	0	1 ⁻	1	1 ⁻	0	1	0	-1	1

Table 4.2. Algorithm Classification Examples

Table 4.2 gives a good indication of the range of different approaches taken by footprint algorithms within the literature. The only two algorithms with identical classifications being the Swinging Arm and Close Pairs algorithms, both of which were created by the same people for the same paper. Of particular interest is the fact that all of the algorithms tend to avoid degeneracies (i.e. are regular) except in extremal cases (such as collinear dots). When we consider the ways in which footprint difference can be measured (Chapter ??) this will be a useful observation.

In classifying the footprints we have given an indication of the range of possible footprint types that can be created. The type of footprint on a dynamic dot pattern can be tracked using the nomenclature provided. While many of the relational criteria would be expected to change (for example, in Fig. 4.14(b) the dots have moved from inside to outside the footprint), if an intrinsic classifier value changes it is an indication that a large change has occurred in the dynamic dot pattern (for example, in Fig. 4.14(c) the footprint is no longer a single connected component). As discussed in Chapter 2, many footprint algorithms require an external parameter (e.g., α in α -shapes [Edelsbrunner and Mücke, 1992], line length in the swinging arm algorithm [Galton and Duckham, 2006], etc.). Most of the papers describing these algorithms come to the conclusion that for their algorithm there is no generally systematic method that can choose an appropriate parameter. Those that do provide such a method involve an iterative construction of the footprint changing the parameter each time until it satisfies some constraints (for example Chaudhuri *et al.*'s *s* and *r*-shapes [Chaudhuri et al., 1997]). In a dynamic dot pattern the changes in the pattern will likely mean that the parameter, even if suitable at the first time step, will need to be changed and this is almost certain if the change has been great enough that the footprint has changed in intrinsic type. We will look at the cost and potential benefit of identifying the change in intrinsic footprint type further in the future work chapter.

¹⁰In which the 1⁻ on regularity and Jordan components refers to patterns with collinear dots.

4.4. Summary

This chapter has provided a detailed look at footprints and how they can be classified. It has discussed the possible uses that such a classification may have and how it can be applied to dynamic dot patterns. Importantly the investigation of footprints, in conjunction with the examination of dot patterns, gives the forthcoming examination of change identifiers a solid foundation. This chapter and Chapter 3 have discussed the outputs and inputs respectively of the framework within which the change identifiers will be expected to operate.

5. Change Identifiers

The thesis has now covered the requisite background information for the change identifiers to be formally introduced and examined. This chapter will present, in greater detail than previously given, the reasoning behind and within the construction of the change identifiers.

5.1. What is a Change Identifier?

Changes in collectives will correspond to changes in the geometric (and statistical) properties of its representative dynamic dot pattern. As discussed earlier it is possible that the change in a dynamic dot pattern between two timesteps is small or even non-existent, for example the phases shown in Fig. 5.1. For such small changes re-computing the footprint at each timestep is unnecessary computation. In the background chapter it was observed that there is current research into the use of spatio-temporal data techniques in emergency situations (wild fires, chemical spills, etc. [Jiang and Worboys, 2008; Jiang et al., 2011]). These emergency situations require a fast and appropriate response, but even when used in non-emergency based applications (such as herd tracking) a fast and appropriate response is desired. If the footprint algorithm takes longer to run than the speed at which the dot patterns arrive then the representation falls behind the actual state of the phenomenon. Table 5.1 shows an example of this when the footprint takes twice as long to run as the time taken for a dot pattern to arrive. The first column is the current time step, the second is the time step that the current footprint is a representation of and the third column shows how far behind the current state the footprint is.

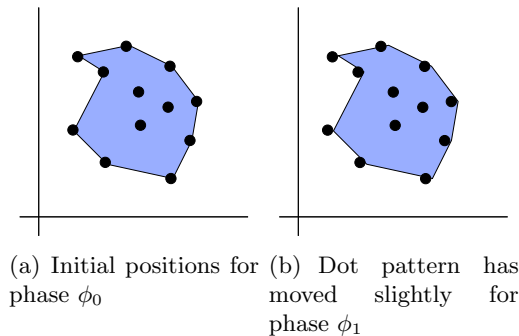


Figure 5.1. Has the dot pattern at ϕ_1 changed sufficiently for the footprint to be updated?

The figure Fig. 5.2 shows how this lag would appear following the same pattern as that given in Table 5.1.

Time Step	Representation	Lag
1	-	-
2	1	1
3	1	2
4	2	2
5	2	3
6	3	3

Table 5.1. The patterns arriving at twice the speed it takes a footprint to be computed.

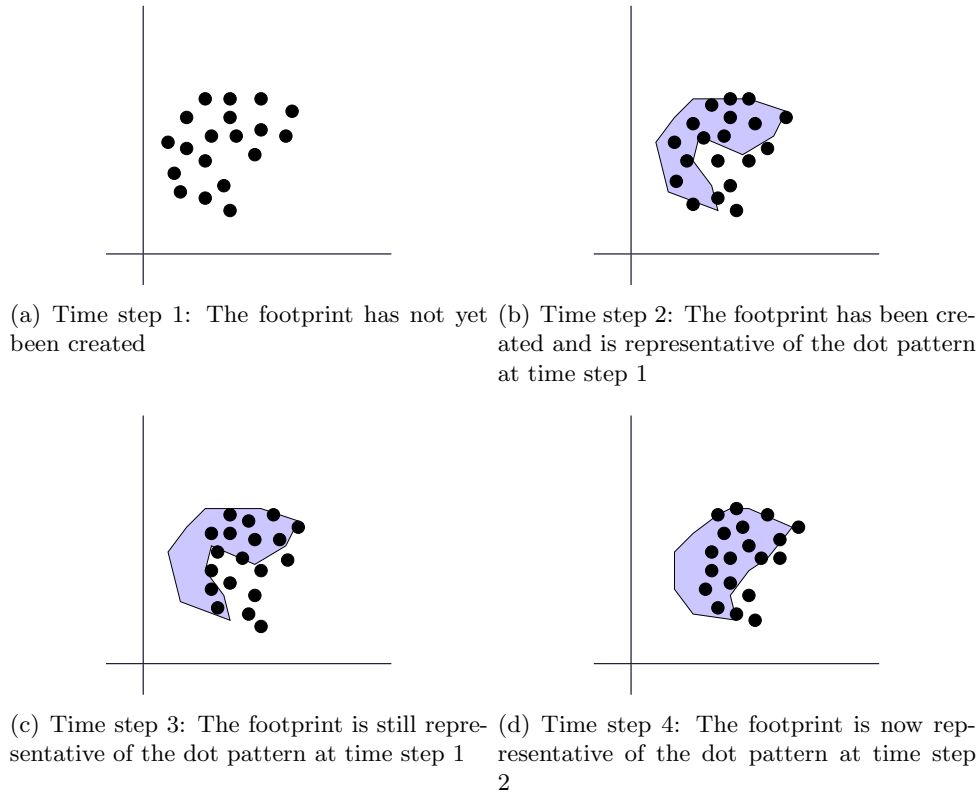


Figure 5.2. Figure showing increasing lag in footprint representation

In any application, when the representation falls behind in this fashion it hinders the user's ability to make decisions about the data. By reducing the number of required footprint updates the change identifiers allow the system to produce accurate footprints faster. For the user, this means that at any given timestep the displayed footprint can be trusted as an acceptable representation.

Often the requirements of an application are not limited to just visualising the region of interest but to use this visualisation to make decisions about how the events underlying the dynamic dot pattern are changing. For example if tracking the spread of a crowd in a shopping centre the user may want to know when clusters are forming. The change identifiers can provide this information without the user having to interpret it from the footprint. We will discuss this further when we consider possible future work (Chapter 8 § 8.3).

Once the method by which to calculate the identifiers has been decided upon, we must provide a cognitively salient way for a threshold to be set. Saliency and its importance has previously arisen within Chapter 2 when footprint algorithms and their parameters were

discussed; the user has to set a parameter for both change identifiers and the footprint algorithms. The difficulty that arises comes from the generality of the expected uses for footprint algorithms and change identifiers; different applications will have different requirements. Parameters are necessary to allow for these different requirements, however they add a layer of complication for a user. For example, given a dot pattern of 1000 dots, ascertaining the correct value of α to use to produce the ideal α -hull for an application trying to find the boundary of a city is not an easy task. In practice such parameters tend to be decided on by trial and error. With change identifiers there is the understanding that time is a constraint, and therefore such trial and error approaches are probably infeasible. By making the threshold setting as cognitively salient as possible the difficulty in choosing an appropriate threshold can be greatly reduced. Ideally the change identifiers should be able to have a threshold set in a way that is intuitive regardless of the application that they are being used in.

The threshold is difficult to set because of the inherent problem in using the difference in property values (i.e. the difference in descriptor values) as the measurement returned by the identifier. This can be demonstrated with a change identifier that measures the area difference of the bounding box. Fig. 5.3 shows two phases (ϕ_1 and ϕ_2) from two

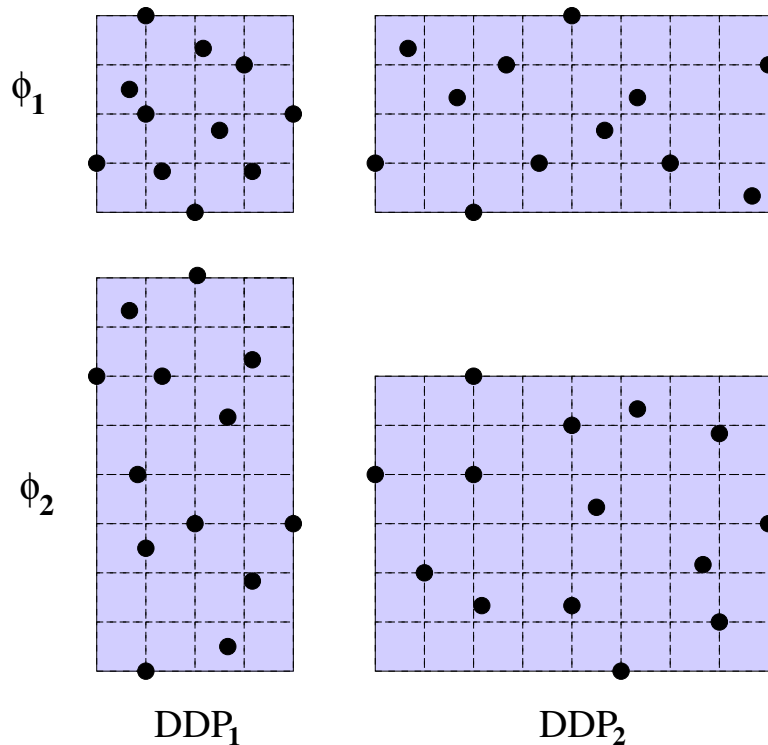


Figure 5.3. Difference in bounding box area for two dynamic dot patterns.

dynamic dot patterns (DDP_1 and DDP_2). The area change from ϕ_1 to ϕ_2 is the same for both (an increase of 16 units²), however DDP_1 has doubled in size whereas DDP_2 has only increased by 50% of its original size. It would not be unfair to state that the change in DDP_1 has a greater impact than the change in DDP_2 . A threshold is required to signal the framework that the footprint must be updated when sufficient change has occurred. Should the threshold be ‘concrete’ (i.e. it is a fixed value and not relative to the dot pattern) then the identifier is not tracking the impact that the change is having

on the dynamic dot pattern. For example, if using the bounding box area, crossing a concrete threshold will represent different impact levels of change because the size of the dot pattern at each phase changes.

The thresholding concern can be satisfied if the requirement that all identifiers return a proportional value is introduced; a value that represents proportional change in the property the identifier measures since the last timestep at which the footprint was updated. The threshold now becomes a percentage value and is, therefore, more cognitively identifiable than a concrete value. For example, in the bounding box example (Fig. 5.3) the user could specify that the footprint is updated when the bounding box area has changed by 100%; triggering an update in DDP_1 but not in DDP_2 . While this still relies on user input we feel that it represents less of a mental leap than intuitively ‘knowing’ by how many units² a pattern’s bounding box area will need to change before its footprint is no longer a suitable representation. The definition of an identifier can now be formalised as:

Definition:

A change identifier is a measurement that compares two phases of a dynamic dot pattern (ϕ_1 and ϕ_2) and returns the difference in a descriptor of the dot patterns expressed as a proportion of the value of that descriptor on the dot pattern ϕ_1 .

For the purposes of this thesis we can add the requirement that the change identifier must be computable in less time than it takes to recompute the footprint. In practice the identifier should ideally have a complexity such that its value can be found in less time than a footprint algorithm with complexity $O(n \log v)$ ¹ takes to produce a footprint, this being the optimal time complexity yet found for a convex hull algorithm. The convex hull time is used as the maximal allowed time taken for computation for an identifier as it is fast compared to the majority of footprint algorithms but not so fast that it is infeasible to compute a change identifier measurement in less time (as opposed to, for example, an isothetic bounding box algorithm).

5.2. Distance Change Identifiers

When considering potential change identifiers one of the most immediately obvious candidates that will need to be measured is change in location. While location maybe an intuitive starting point it has two unique aspects, primarily because we can use it to update the footprint without recomputation; translating the footprint along the same vector that the dynamic dot pattern has moved. The other standard transformations (scaling, rotation and shearing) can also be directly applied to the footprint in a computationally fast time, however identifying change in these transformations is less easy and none of them are as simple to apply to the footprint as the change in location. The other interesting aspect of a location change identifier is that it shows a complication in the use of the

¹Here n is the number of dots and v is the number of dots at the vertices of the footprint.

difference in descriptor values as the change measure. Location is a measurable property of a dot pattern but, instead of a single real-number, it is a vector of dimension equal to the space the pattern resides in. The vector adds complexity as scaling it and checking it against a threshold cannot be done in the same fashion that is provided in the original change identifier definition for two reasons. Firstly it would require the use of a vector as a threshold, as mentioned above we wish to make the choice of thresholds as simple as possible and having thresholds of differing types runs counter to this aim. Secondly, and more importantly, it makes no sense to make the change in position proportional to the value of the position at the earlier phase as this has no bearing on how much the dynamic pattern has changed between the phases. This can be generalised with the statement that any change identifier should be invariant under a change of coordinate system. Taking the

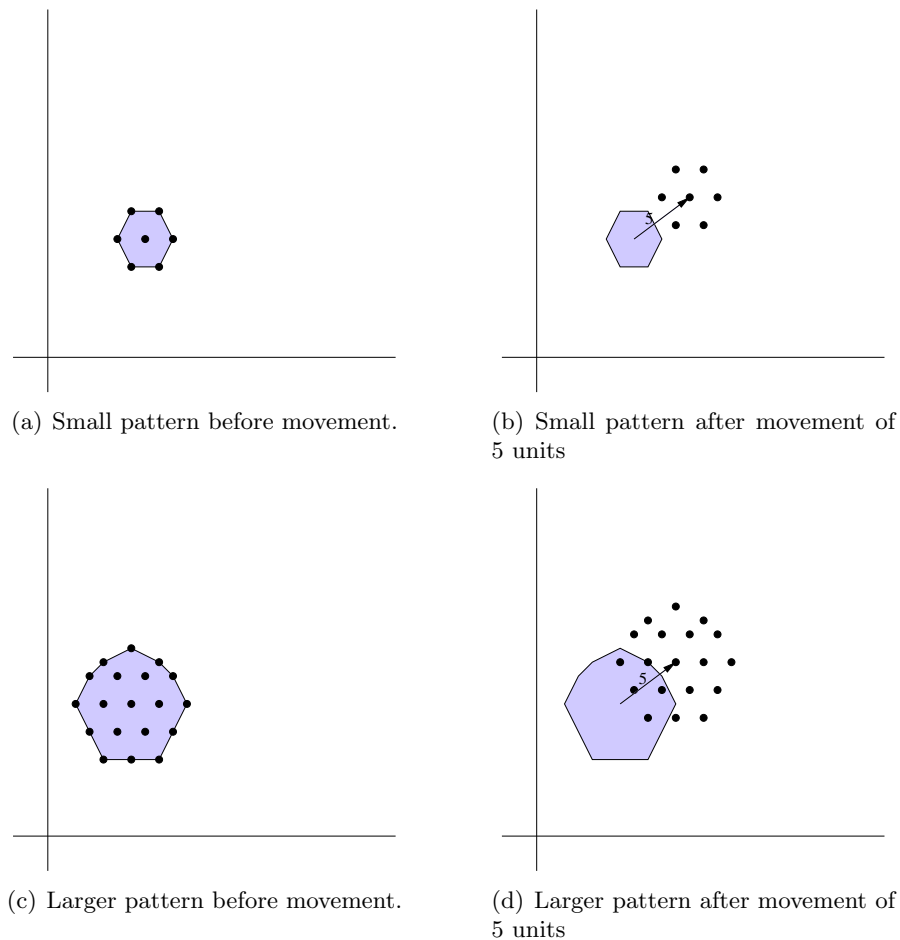


Figure 5.4. Figure showing change in effect for a change in position for two different sized patterns

distance between two locations would be a sensible way to reduce it to a scalar value but doing so loses information about the direction in which the dynamic pattern has moved. A further consideration about the distance measure is that, previously, we have defined a change identifier as returning a value proportional to the value of the property it measures at the earlier of the two dot patterns that it is passed. The distance between the locations of two patterns cannot be scaled in such a way as there is no distance value for any single pattern. Distance can, however, be made proportional to the size of the pattern. Fig. 5.4 shows two dynamic dot patterns at two time steps. Both dynamic patterns move by the same distance but the larger pattern (Fig. 5.4(d)) is still partially within its original foot-

print; the impact the change has on how suitable its footprint is less than that of the smaller pattern (Fig. 5.4(b)). The movement vector could also be made proportional to the extent of the pattern, but it is still a multi-dimensional value. There are several ways in which to approach thresholding the vector:

1. A separate identifier can be created for each of the spatial dimensions.
2. The identifier can return the multi-dimensional value scaled by extent and the framework will check each of the elements of the vector to see if they have exceeded a single value threshold.
3. The framework is set up so that multi-dimensional thresholds are allowed.

(2) and (3) both involve extensions to the framework. Such extension is not necessarily a negative, however both also add complexity when considering combining change values. We will discuss how the identifiers can be combined and why they would be in the section on change identifier sets (§ 5.3). (1) is an approach that avoids the problems of extending the framework or having to change the thresholding methods but it does add an extra identifier for each dimension possibly making it a slower approach. Since both the distance and the vector methods scale by extent neither fit the above definition for a change identifier. To represent this split, we call our original identifier type (identifiers that measure the difference between two descriptors that can be scaled by the measurement of that descriptor) *descriptor* change identifiers. Naming this alternative type of change identifier is a little more difficult, the Euclidean distance is a metric but and therefore calling these measures *metric* change identifiers seems sensible. However we do not wish to invalidate any of the possible measures of difference between two dot patterns and can therefore not assume that all of the axioms of a metric are always obeyed. The only two axioms that it is probably safe to require are that of non-negativity and relaxed version of coincidence:

$$\begin{aligned} dist(x, y) &\geq 0 \\ dist(x, x) &= 0 \text{ instead of } dist(x, y) = 0 \iff x = y \end{aligned}$$

Where *dist* is the distance measure in question. Measures that conform to these axioms are sometimes called *premetrics* but this is not a standard term and as such we shall avoid it. Instead we will name change identifiers using a more complicated measure than the difference between two descriptor values *distance* change identifiers.

Definition:

A distance change identifier is a measurement that compares two phases of a dynamic dot pattern (ϕ_1 and ϕ_2) and returns, for some given distance measure, the distance between the patterns expressed as a proportion of some property measure of ϕ_1 such that the change identifier returns a value indicative of the impact the change has had on the suitability of the footprint.

The descriptor change identifiers can be seen as a special case of distance change identifiers in which the distance measure is the absolute difference between the descriptor values. However they are suitably interesting in their own right that they remain as a distinct type for the examination presented by this thesis.

5.3. Change Identifier Sets

We have previously mentioned that it is unlikely that a single change identifier will be able to ‘catch’ all forms of change. Unless it is known that a dynamic dot pattern will only change in one aspect it is likely that more than one identifier will have to be used. For example, a herd of prey fleeing a predator will change in location, extent, dimensionality and connectivity (Fig. 5.5). At the time step that the connectivity and extent changes (Fig. 5.5(b)), the location and dimensionality may not have changed appreciably. To make sure that the framework does not miss any timesteps at which it should cause a footprint update, identifiers checking for change in all four aspects would be preferable.

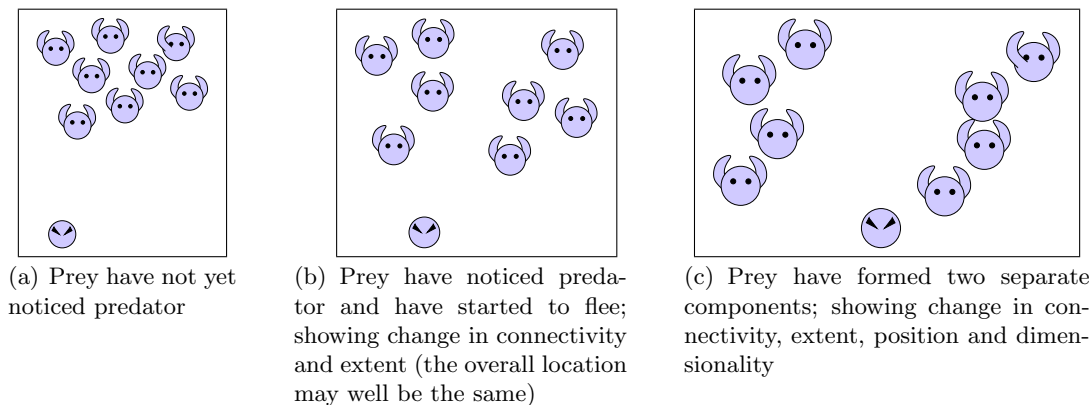


Figure 5.5. Simplification of prey being chased by a predator

The change identifier concept allows for multiple identifiers to be used. No identifier relies on the value returned by any other and as a result they can be run concurrently². Alternatively the identifiers can be ordered by importance and run consecutively. It should be noted that increasing the number of identifiers when running consecutively results in an increase in processing time, while reducing the number of identifiers reduces the amount of change types being checked for. This is not an entirely linear relation as some identifiers may be ‘better’ at checking and/or take longer to compute³.

When combining multiple identifiers we must address the issue that arises from having to assign a sensible threshold for multiple types of change. As discussed earlier, this must be done in such a way that it is obvious to a user what the threshold represents and that the generality of the change identifiers has not been lost. Ideally the user should also be able to indicate that some identifiers are ‘worth’ more than others, i.e., that change in some identifiers is more important to the application than others.

²Although in running experiments using Java it was found that starting a new thread for concurrent identifiers often took longer than any individual identifier took to run.

³When using a single processor the same trade-off relation applies to concurrently run change identifiers.

To allow for flexibility we define a *change identifier set* as a container for multiple identifiers with properties dictating its operation. The change identifier set is a very flexible entity with properties that allow it to be fit to multiple applications. The sets used within this thesis are specified by an XML document that dictates the identifiers and the properties used (Listing 5.1).

```

1 <changeidentifierset name='[set-name]' ver='[version] '>
2   <description>[Description of the set]</description>
3   <threshold>[Total threshold (%)</threshold>
4   <maxFails>[Proportion of identifiers allowed to breach their threshold]</
      maxFails>
5   <concurrent>[Whether or not to run identifiers concurrently]</concurrent>
6   <changeidentifier>
7     <identifier>[Change Identifier Name]</identifier>
8     <priority>[Ordering, lower number = higher priority]</priority>
9     <threshold>[Identifier threshold (%)</threshold>
10    <multiplier>[Custom user defined normalisation value]</multiplier>
11    <updateOnFail>[Whether or not to update if this
12      identifier exceeds its threshold]</updateOnFail>
13  </changeidentifier>
14 </changeidentifierset>

```

Listing 5.1 Change Identifier Set

Each set has associated meta-information (name, version and description). This information is not important to the running of the set but exists to make its identification, and therefore use, simpler. Rather than step through each XML tag, explaining its use and range of values, we will describe some possible user requirements then show an example of how the XML specification would be set up.

In the first example (Listing 5.2), our user has three identifiers ($change_1$, $change_2$, $change_3$) that they need to run in order. If the $change_1$ exceeds a change of 10% the footprint needs to be updated. If $change_1$ does not exceed that threshold then only if both $change_2$ and $change_3$ exceed their thresholds of 10% each should an update occur.

```

1 <changeidentifierset name='use-case-1' ver='0.1b' >
2   <description>Consecutive run of three identifiers</description>
3   <maxFails>0.66</maxFails>
4   <concurrent>>false</concurrent>
5   <changeidentifier>
6     <identifier>change1</identifier>
7     <priority>0</priority>
8     <threshold>0.1</threshold>
9     <updateOnFail>>true</updateOnFail>
10  </changeidentifier>
11  <changeidentifier>
12    <identifier>change2</identifier>
13    <priority>1</priority>
14    <threshold>0.1</threshold>
15    <updateOnFail>>false</updateOnFail>
16  </changeidentifier>
17  <changeidentifier>
18    <identifier>change3</identifier>

```

```

19 <priority>1</priority>
20 <threshold>0.1</threshold>
21 <updateOnFail>>false</updateOnFail>
22 </changeidentifier>
23 </changeidentifierset>

```

Listing 5.2 Change Identifier Set Example 1

User number two (Listing 5.3) has twenty identifiers but has no preference about their running order. The sheer number of identifiers and the unspecified ordering make it preferable that they run concurrently. This user has no thresholds for any individual identifier but needs an update to occur if the total change exceeds 50%.

```

1 <changeidentifierset name='use-case-2' ver='0.1b'>
2 <description>Concurrent run of twenty identifiers</description>
3 <threshold>0.5</threshold>
4 <concurrent>>true</concurrent>
5 <changeidentifier>
6 <identifier>change1</identifier>
7 <updateOnFail>>false</updateOnFail>
8 </changeidentifier>
9 <changeidentifier>
10 <identifier>change2</identifier>
11 <updateOnFail>>false</updateOnFail>
12 </changeidentifier>
13 ...
14 <changeidentifier>
15 <identifier>change20</identifier>
16 <updateOnFail>>false</updateOnFail>
17 </changeidentifier>
18 </changeidentifierset>

```

Listing 5.3 Change Identifier Set Example 2

The requirement given by the second user that the *total* change not exceed 50% is worth further examination. Each identifier returns a value that can be seen as a representative of part of the total change that a dynamic dot pattern is undergoing. To produce a value for total change a method with which to combine these values is required. The combination needs to be done in such a way that we fully justify the value and are not merely creating numbers with no real-world counterpart. The concerns that we have to address when using any combination operator are:

1. Are the values combined in a way that mixes data types or scales?
2. Is undue weighting added to any value?
3. What information is lost when reducing many values to one?

The values are already normalised to percentages, as required by the change identifier definitions; so we may safely ignore (1). (2) raises a little more difficulty, if two identifiers both use descriptors from the same class then change in that class will be represented twice. For example, an identifier which measures the change in the variance from the

centroid and an identifier which measures change in the area of the minimum bounding box will both indicate change if the dynamic dot pattern grows in extent. To alleviate some of this concern we have also allowed for a multiplier to be attached to each identifier. The user can mitigate (or increase) the effect of any particular identifier on the total change value by setting this property. However we strongly suspect that the multiplier would require trial and error to adjust so that the results suit the user's expectations. As discussed earlier in this chapter, this type of variable 'tweaking' is impractical even when the data is static, so with dynamic data it is likely infeasible. Instead we have looked at the measurement types (see descriptor definitions in Chapter 3) in such a way that the identifier selection process can be informed, and thereby overlapping measurements can be avoided. (2) affects any combination operator, assuming the application has cause enough to require a total threshold value then this will have to be taken into account by the user and not the framework. Question (3) is answered comparatively simply: The information lost is the manner in which change has occurred but, as for (2), if the application requires a single total value then this loss must be accepted. The simplest operator, and perhaps the most intuitive, is addition of the percentages. The three questions raise no specific objections to the use of addition, and it is likely that any more complex method will add further confusion about what the value represents as well as increasing the processing time of the change identifier set. Hence, addition is the operator used within this thesis.

5.4. Constructing Change Identifiers

With a formal description in place the construction of some example change identifiers can be considered. We begin by providing the mathematical formula for a change identifier; this is simply the formalisation of its definition.

Descriptor Change Identifier

$$\forall \phi \in DDP_i$$

$$change_x(DDP_i, \tau, u) = \left| \frac{desc_x(\phi_\tau) - desc_x(\phi_u)}{desc_x(\phi_u)} \right|$$

Where x is the descriptor index mapping to an element of the set of all descriptors, $change_x(DDP_i, \tau, u)$ is the change identifier value for the change identifier over dynamic dot pattern i at time τ when the last update time was u and $desc_x(\phi_\tau)$ is the descriptor value for descriptor x on the phase from dynamic dot pattern i at time τ .

Distance Change Identifier

$$\forall \phi \in DDP_i$$

$$change_x(DDP_i, \tau, u) = \left| \frac{dist_x(\phi_\tau, \phi_u)}{desc_{s(x)}(\phi_u)} \right|$$

Where x is the index mapping to an element of the set of all possible distance measures, $dist_x(\phi_\tau, \phi_u)$ is the distance between the phases from dynamic dot pattern

i at times τ and u for the distance measure x and $desc_{s(x)}$ is a *suitable* descriptor for metric x by which to normalise the value.

The list of identifiers presented is by no means complete but is sufficient to provide comparison between different types of identifier and showcase some of their more interesting aspects. We will begin by listing descriptor identifiers as they are often simpler than the metric type. The identifiers are uniquely named and are formatted in small caps (i.e., CHANGEIDENTIFIER) so as to be easily referenceable.

The descriptor for a descriptor identifier must be computed for two different phases: The current phase and the phase at the timestep at which the footprint was last updated. However, the calculation at the update time has already taken place (when it was the current timestep) for all timesteps except the second (the first will always be an update time so there is no point running the change identifiers). Therefore the complexity and time taken of any descriptor identifier are equivalent to those of its descriptor.

5.4.1. Descriptor Identifiers

CARDINALITY

Simple to implement and run, a change identifier that measures change in cardinality is a good initial identifier. Given that this can be found while building the data structure that contains the dot pattern, we can say that, for an identifier to use it, it need not be computed and can be found in constant time.

The rest of the descriptor identifiers will be ordered as they appear in Chapter 3 apart from position which is a distance identifier. As many of the descriptors were already detailed in that chapter they will only be briefly discussed here.

Extent

VARIANCEFROMCENTROID

The centroid is the mean position of all the dots within the pattern and as such can be found in $O(n)$ time.

$$\begin{aligned} \text{centroid}(\phi_\tau) &= d \in \phi_\tau \frac{\sum_{i=1}^n d_i}{n} \\ \text{variance}(\phi_\tau) &= d \in \phi_\tau \frac{\sum_{i=1}^n (d_i - \text{centroid}(\phi_\tau))^2}{n} \end{aligned}$$

Where d_i is a dot from the dynamic dot pattern phase ϕ_τ . Variance has a computational complexity of $O(n)$ and, thus, so does the identifier VARIANCEFROMCENTROID.

BOUNDINGBOXAREA

When the descriptors were examined in Chapter 3 it was noted that this identifier ascribes a surrogate region to the dot pattern: the isothetic minimum bounding box. The bounding box can be found, even on an unordered list, within $O(n)$ time as all that is required is

the extremal points in the cardinal directions of the pattern. The area for the bounding box can be found in constant time so the identifier BOUNDINGBOXAREA can also be run in $O(n)$ time. Should a tree structure be used, such as was detailed in the background chapter, then the time to find extremal points becomes $O(\log n)$.

DIAMETERSQ

The diameter is the distance between the two furthest apart dots in a pattern, and the fastest methods found in the literature to retrieve it check the vertices of the pattern's convex hull. The high complexity renders this identifier impractical to use so the approximation introduced in Chapter 3 is used (the greatest distance between the minimum and maximum dots in the plane). This identifier has a linear computation time if the data structure is an unordered list and a $O(\log n)$ time if the dots are stored in an ordered tree.

Orientation

OLSGRADIENT

The gradient of the line found by the Ordinary Least Squares (OLS) method described in Chapter 3. OLS can be computed in linear time and is therefore well within the change identifier time requirements.

PCVECTORGRADIENT

The gradient of the vector found by Principal Component Analysis (PCA). As explained in Chapter 3, to find the Principal Component the eigenvector corresponding to the largest eigenvalue of the covariance matrix of the data must be found. Finding the eigenvalue for any given matrix is generally a complex task. However the experiments in this thesis, and the majority of the examples in the footprint algorithm and spatio-temporal literature, are planar. The eigenvalues of a matrix are the values of k that satisfy the equation $\det(A - kI) = 0$ (the characteristic equation). For a 2x2 matrix the eigenvalues can be found using the quadratic formula of the characteristic equation and can therefore be found in constant time. Finding the covariance matrix has a linear time complexity, and its construction for a dot pattern in Cartesian planar space is:

$$cov(x, y) = \begin{bmatrix} \frac{\sum_{i=1}^n (x-\bar{x})^2}{n} & \frac{\sum_{i=1}^n (x-\bar{x})(y-\bar{y})}{n} \\ \frac{\sum_{i=1}^n (x-\bar{x})(y-\bar{y})}{n} & \frac{\sum_{i=1}^n (y-\bar{y})^2}{n} \end{bmatrix}$$

Dimensionality

EIGENVALUEDIFF

The greater the difference between the eigenvalues of the covariance matrix the greater the variance in the principal component compared to any other. In effect the greater the eigenvalue difference the closer to collinear the dots of the pattern are. Like PCVECTORGRADIENT, this identifier has a linear time complexity.

PEARSONCORRCOEFF

Pearson correlation coefficient can be found in linear time. This provides a measures

for the interrelation between the coordinates of the dots in the dot pattern; i.e., as x increases to what degree does y change in accordance. The stronger this relation is the closer the dots are to being collinear. If the pattern is collinear and aligned to the axes then, despite the collinearity, there is no correlation between the coordinates. Fortunately `PEARSONCORR` provides a result of NaN (Not a Number) for axis-aligned patterns, therefore allowing them to be identified as distinct from patterns with a low correlation due to low collinearity.

Dispersion

DENSITY

This is the global density of the pattern as defined by the isothetic bounding box. The complexity of finding the isothetic bounding box, as shown above, is at most $O(n)$, and, considering that the cardinality should already be known, this is the maximum complexity of the `DENSITY` identifier.

NEARESTNEIGHBOURDISTVARIANCE

The variance of the distances between each dot and its estimated nearest neighbour (estimated for the reasons given when we examined the nearest neighbour descriptor in Chapter 3). This identifier relies on a sorted data structure to be found in a time less than $O(n^2)$. With such a data structure we can find the estimated nearest neighbour for each dot in $O(\log n)$ time, so the total time to find the variance is $O(n \log n)$. The variance of the nearest neighbour distances indicates uniformity of the spacing of the pattern, so change within it points to a change in the behaviour of the collective, for example a crowd going from calm and well spaced to panicking. The complexity is above the norm for a change identifier but it computes fast because of the limited number of computational steps required for each iteration.

SKEWNESS

Skewness measures the tendency for the dots to be in one direction from the mean. The equation for its computation was given Chapter 3 but is repeated here for clarity:

$$\begin{aligned}\mu_3 &= \text{E}[(\phi - \text{E}[\phi])^3] \\ s &= \frac{\mu_3}{\sigma^3}\end{aligned}$$

Where σ is the standard deviation and $\text{E}[\phi]$ is the expected value of any dot from ϕ , for the purposes of a dot pattern this is equivalent to $\bar{\phi}$. As it requires only the mean and the standard deviation `SKEWNESS` can be computed in linear time.

KURTOSIS

Kurtosis measures the tendency for the dots to be close to the mean. It is related to `SKEWNESS` as a moment about the mean and they both require only the mean and the standard deviation to be computed from iterations over the pattern, so `KURTOSIS` can also be found in linear time.

Miscellaneous

`TIMESTEPCOUNT`

Timestep count is the first identifier of this thesis not to have a counterpart within the dot pattern descriptors. The identifier counts the number of timesteps since the footprint was last updated and once this exceeds a set limit signifies that its threshold is broken. It functions as somewhat of a control; its purpose is to indicate whether just reducing the number of updates without any thought of measuring change is better than updating at each timestep. It serves a second role in that should it perform better than another change identifier it is likely that the ‘beaten’ change identifier is not measuring a useful property. To function, it ‘cheats’ and looks ahead to see how many phases are in the dynamic dot pattern⁴. The threshold is the percentage of phases which are allowed to occur before it forces an update of the footprint.

5.4.2. Distance Identifiers

`CENTROID`

This identifier takes the squared distance between the centroids of the current phase of the pattern (ϕ_c) and the phase at which the footprint was last updated (ϕ_u) and divides this difference by the variance of the pattern at phase ϕ_u . Despite being a very simple identifier there is an interesting point to be made about the units it uses. The variance is the square of the standard deviation whereas the distance is a linear value. To maintain consistency of units either distance squared and variance, or distance and standard deviation should be used. Given that the process of finding the standard deviation and the distance involves first finding the variance and the distance squared, and that finding the square root is a computationally expensive operation (compared to a multiplication, for example), then using variance and distance squared makes more sense than standard deviation and distance.

`BOUNDINGBOXCENTRE`

Bounding box centre is similar to the `CENTROID` identifier, however it takes the distance squared between the centres of the isothetic bounding box of the current phase (ϕ_c) and the phase at which the footprint was last updated (ϕ_u) and divides this value by the bounding box area at ϕ_u . As has been discussed above, the time to find the bounding box is between $O(\log n)$ and n , depending on the data structure, so `BOUNDINGBOXCENTRE` is at least as fast as `CENTROID`, if not faster. Like with the `CENTROID` identifier the units have been kept consistent (they are both square).

`BOUNDINGBOXSYMMETRICAREADIFF`

This identifier takes the symmetric area difference between the isothetic bounding boxes of the current phase (ϕ_c) and the phase at which the footprint was last updated (ϕ_u), and divides this difference by the area of the bounding box at ϕ_c . The symmetric area difference between two bounding boxes is demonstrated in Fig. 5.6, in which c is the

⁴This would not be possible in a live system as the data arrives with no indication of when it will end

overlap of the boxes a and b . The sum of the areas of a and b minus the twice the area of c gives the area of the symmetric area difference, shown as the shaded regions of the figure. The isothetic bounding box is used for the fast time in which it can be found, as described above, and also because it reduces the complexity of finding the symmetric area difference; there are only 4 vertices per box to be checked for containment within the other. Symmetric area difference is looked at in greater detail in the methodology chapter Chapter 6.

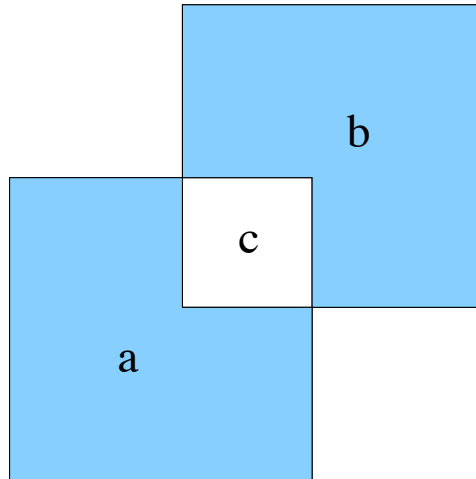


Figure 5.6. Symmetric area difference between boxes a and b . The shaded area is the difference

Identifier	Complexity	Complexity using sorted data struct.
CARDINALITY	$O(c)$	$O(c)$
VARIANCEFROMCENTROID	$O(n)$	$O(n)$
BOUNDINGBOXAREA	$O(n)$	$O(\log n)$
DIAMETERSQ	$O(n)$	$O(\log n)$
OLSGRAIENT	$O(n)$	$O(n)$
PCVECTORGRADIENT	$O(n)$	$O(n)$
EIGENVALUEDIFF	$O(n)$	$O(n)$
PEARSONCORRCEFF	$O(n)$	$O(n)$
DENSITY	$O(n)$	$O(n)$
NEARESTNEIGHBOURDISTVARIANCE	$O(n^2)$	$O(n \log n)$
SKEWNESS	$O(n)$	$O(n)$
KURTOSIS	$O(n)$	$O(n)$
TIMESTEPCOUNT	$O(c)$	$O(c)$
CENTROID	$O(n)$	$O(n)$
BOUNDINGBOXCENTRE	$O(n)$	$O(\log n)$
BOUNDINGBOXSYMMETRICAREADIFF	$O(n)$	$O(\log n)$

Table 5.2. Table showing change identifier complexities.

5.5. Performance Analysis

To establish the fitness of change identifiers for their purpose, we need to be able to measure the ‘quality’ of the footprint. It should be stressed that we are not commenting on how well the footprint algorithm can create a footprint that represents the pattern; we assume that the algorithm used was chosen for its suitability to the application, possibly using

the footprint classification in Chapter 4. The ‘quality’ we measure is how close the stored footprint is to the footprint which would result if it were recomputed from the current dot pattern using the chosen algorithm at any given step. The overall quality for a sequence of dot patterns is obtained by combining the quality values for each step. Our goal is to maximise the quality while minimising the computation time. These are conflicting objectives: to maximise quality is to minimise the difference between the stored and true footprints and this can only be achieved by updating the footprint at every timestep, resulting in a maximal value for the computation time. Conversely, the computation time would be minimised by never updating the footprint, typically resulting in catastrophic loss of quality. We therefore need to seek a middle course which optimises the trade-off between the objectives.

In order to compute the total time taken, we will need to make use of the following quantities:

- $t_{foot}(i)$ is the time taken to compute the footprint from the dot pattern at step i .
- $t_{change}(i)$ is the time taken to evaluate the change identifier(s) at step i .
- $r(i)$ is a Boolean variable, set to 1 if the change identifier(s) evaluated at step i exceed(s) the pre-set threshold, and 0 otherwise.

The footprint has to be computed at least once, namely at the first timestep ($i = 0$). At subsequent timesteps it is only recomputed if the change identifiers return a value above threshold. The total computation time over a run of p dot patterns is thus

$$T_{change} = t_{foot}(0) + \sum_{i=1}^p (t_{change}(i) + r(i)t_{foot}(i)).$$

The value of T_{change} is minimum when the change identifier threshold is set so high that the footprint is never recomputed after the start of the sequence (so $r(i) = 0$ for $1 \leq i \leq p$):

$$T_{\min} = t_{foot}(0) + \sum_{i=1}^p t_{change}(i).$$

It is maximum when the change identifier threshold is set so low that the footprint is recomputed at every time step (so $r(i) = 1$ for all i):

$$T_{\max} = t_{foot}(0) + \sum_{i=1}^p (t_{change}(i) + t_{foot}(i)).$$

If change identifiers are not used at all, and the footprint is recomputed at every timestep, then the total time taken is:

$$T_{NCI} = \sum_{i=1}^p t_{foot}(i) = t_{foot}(0) + T_{\max} - T_{\min}.$$

If it is assumed that always $t_{change}(i) < t_{foot}(i)$ (for if not, there would be little point

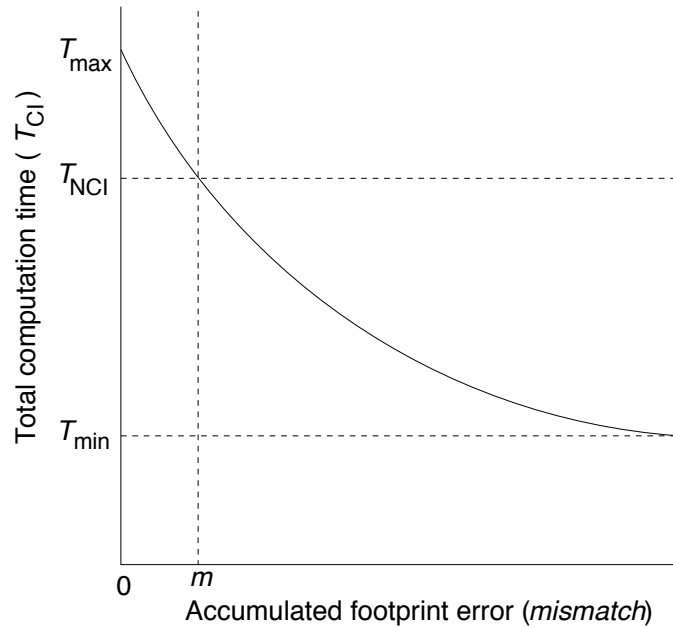


Figure 5.7. Total computation time against aggregate footprint error

in using change identifiers) then $T_{\min} < T_{NCI} < T_{\max}$, so the relative size of T_{change} and T_{NCI} — which provides a measure of the time advantage, if any, gained by using change identifiers — depends on the threshold settings.

It will be convenient in the following chapters to consider an inverse form of the quality measure, which we shall refer to as *error*. Our goal is therefore to seek to minimise both time and error. To measure error, we need a way of quantifying the extent of the mismatch between the stored footprint and the true footprint. The difference between two footprints can be measured in various ways, (e.g., using Hausdorff distance, or symmetric area difference) and these are discussed in Chapter 6 when the difference measure used in this thesis is explained.

If the footprint is recomputed every time, corresponding to total computation time T_{\max} , we have a footprint for every phase, so $mismatch = 0$. At the other extreme, the maximum value of $mismatch$ is obtained when the footprint is never recomputed, corresponding to T_{\min} . There is thus a trade-off between $mismatch$ and computation time, as indicated in Fig. 5.7, where different choices of change identifier thresholds correspond to different positions on the curve. The optimal setting for the change identifier threshold depends on the relative importance attached to the conflicting goals of minimising both computation time and accumulated footprint error; but in any case no time advantage can be obtained for error below the value m at which $T_{CI} = T_{NCI}$.

5.6. Summary

This chapter has provided definitions for two types of change identifier: *descriptor* and *distance* based and used these to construct several example change identifiers. How the change identifiers can be used to measure change has been examined, while paying special

attention to their thresholds and how they may work in concert. The concept of a change identifier set was expanded upon and it was shown how they may be used in applications with differing requirements. Finally this chapter has presented a method by which to assess the change identifiers by the footprints that are produced when using them.

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 each 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

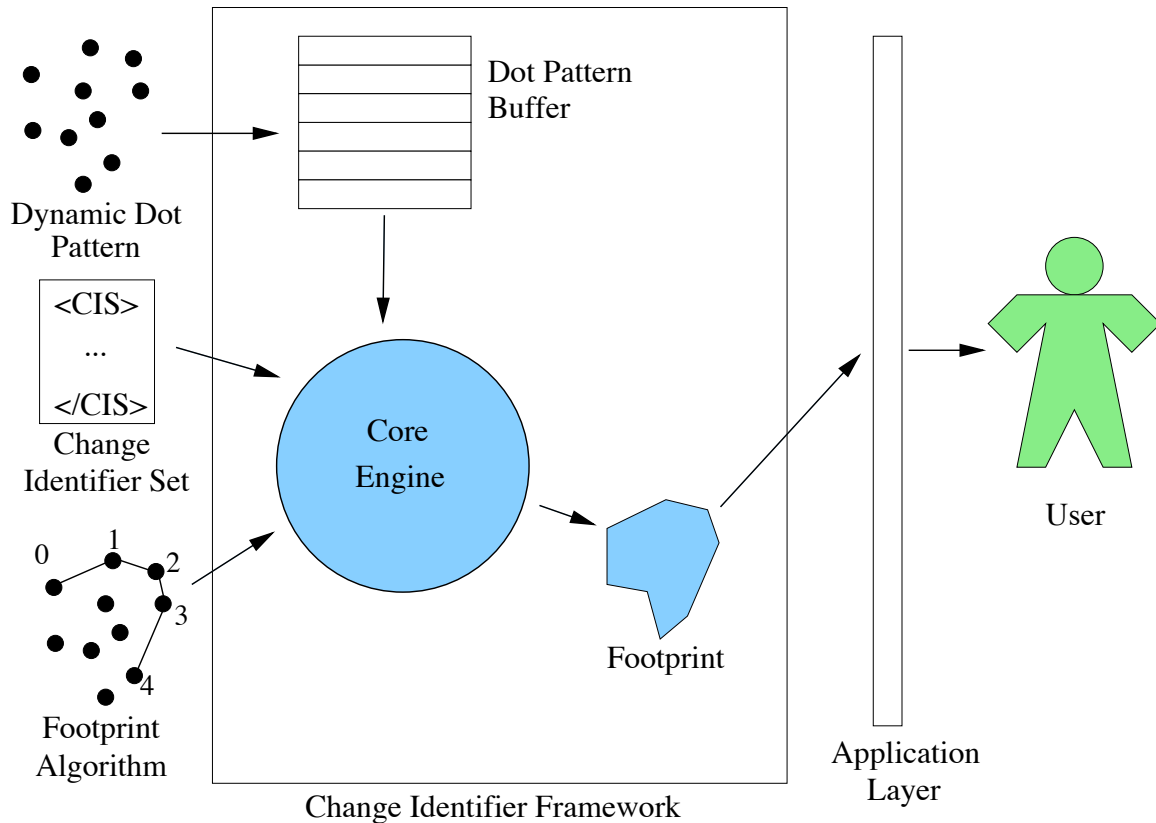


Figure 6.1. Modular Framework Architecture

locations, as it will produce a smaller file, however this assumes that not every entity is likely to change at the same time. 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 cannot 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 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 [Guibas and Sedgewick, 1978]) 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 up to two values at each node [Bishop, 2007]) 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 tree's fast insertion and search times make it a sensible choice. 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 for 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 easy¹.

6.3. How is the footprint algorithm specified?

The footprint algorithm is specified at the initialisation of the program. This would be straightforward 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 cavities?)². The parameter choice is beyond the purview of this thesis but we discuss in Chapter 8 § 8.3 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 initial 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 $foot(\phi_0)$ 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).

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

²The 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.

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 ϕ_i differs from the previously stored dot pattern SDP_{i-1} . If this value, $eval(\phi_i, SDP_{i-1}, SFP_{i-1})$, exceeds some pre-set threshold, then a new footprint $foot(\phi_i)$ 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 $foot(\phi_i)$ 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  $\phi_0$ 
3:  $SFP_0 = foot(\phi_0)$ 
4:  $SDP_0 = \phi_0$ 
5: repeat
6:    $i = i + 1$ 
7:   Input  $\phi_i$ 
8:   if  $eval(\phi_i, SDP_{i-1}, SFP_{i-1}) > threshold$  then
9:      $SDP_i = \phi_i$ 
10:     $SFP_i = foot(\phi_i)$ 
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 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, for each phase

the data stored is the length of time taken to process each change identifier, the time taken to process the timestep (change identifiers and footprint algorithm) and the change identifier that caused an update of the footprint (if any). Immediately after this 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 timestep. 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 by which to ascertain the difference between two regions. Hausdorff distance, Fréchet boundary separation and symmetric area difference are three of the possible similarity measures with different approaches ([Galton, 2000, ch. 7.3]). Hausdorff distance has two variations which we shall also consider: the Hausdorff boundary separation [Alt et al., 1993] and the dual-Hausdorff distance [Davis, 1995]. The rest of this section will involve first a description of how each measure works followed by a discussion on how they relate to the problem of measuring footprint difference. These five measures are by no means the only measures of distance between shapes but they provide a sufficiently distinct range with which to discuss some of the difficulties presented when selecting an appropriate measure for this thesis, and indeed some of the difficulties faced in the field of shape similarity assessment.

The Hausdorff distance is given by the greatest of the Euclidean distances between the closest points from one region to the other. That is, the maximum of the greatest of the distance from any point in polygon X to its closest point in polygon Y and greatest of the distance from any point in polygon Y to its closest point in X , shown by the dashed arrow in Fig. 6.2. The formula for this measure is given below, in which $d(a, b)$ is the Euclidean distance between a and b :

$$d_h(X, Y) = \max(\sup_{j \in Y} \inf_{i \in X} d(i, j), \sup_{i \in X} \inf_{j \in Y} d(j, i))$$

Hausdorff boundary separation is the Hausdorff distance of the boundaries of the regions (in Fig. 6.3 this is the solid line showing the greatest closest distance from Y to X) and the dual-Hausdorff distance is the greater of the Hausdorff distance of the two regions and the Hausdorff distance of the closed complements of the two regions.

The Fréchet distance is the minimal length of line required to connect two paths; this is easier to understand if the Fréchet distance is considered as the minimal length of leash required to connect a dog and its walker travelling on a path each, with the restriction that they cannot go backwards but are allowed to dictate their own speeds. To use the Fréchet distance as a measure of footprint similarity, the boundaries of the footprints must first be refactored as directed paths.

The symmetric area difference between two regions comprises the cumulative area of the

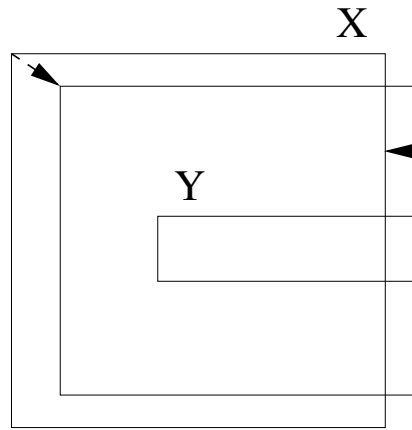


Figure 6.2. Hausdorff distance Example.

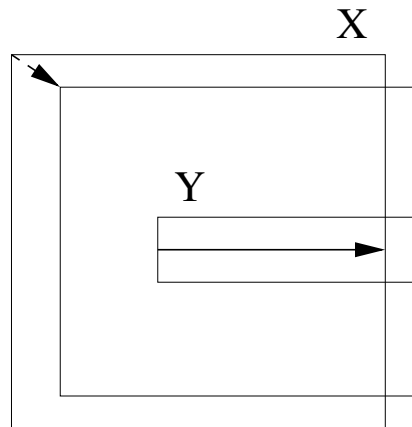


Figure 6.3. Hausdorff boundary separation Example.

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. 6.4, the shaded region in Fig. 6.4(b) is the parts of the regions (X and Y) that do not coincide with any part of the other region ($(X \setminus Y) \cup (Y \setminus X)$) and the area of these parts is the symmetric area difference of X and Y .

The various distance measures have different properties, some of which are beneficial to our application and some which are detrimental. The Hausdorff distance and Hausdorff boundary separation present some particularly interesting aspects for comparison.

As shown in Fig. 6.5, the Hausdorff distance is inconsistent in its treatment of ‘spikes’. Internal spikes are almost entirely ignored whereas external ‘spikes’ can add greatly to the distance. Hausdorff boundary separation, on the other hand, treats both forms of spike with far more equivalence. Hausdorff boundary separation is, instead, adversely affected by one of the footprints ‘wrapping’ the other, as demonstrated in Fig. 6.6.

In terms of computation the Hausdorff distance is more complicated than the boundary separation as it takes the region into account. This may be achieved by either treating the shape as a set of simplices ([Alt et al., 2001]) or by sampling from within the shape

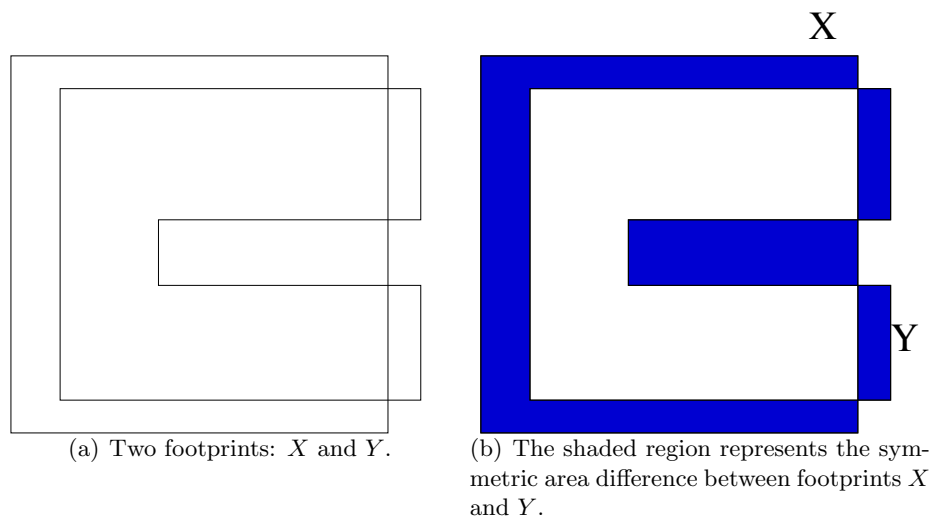


Figure 6.4. Symmetric area difference

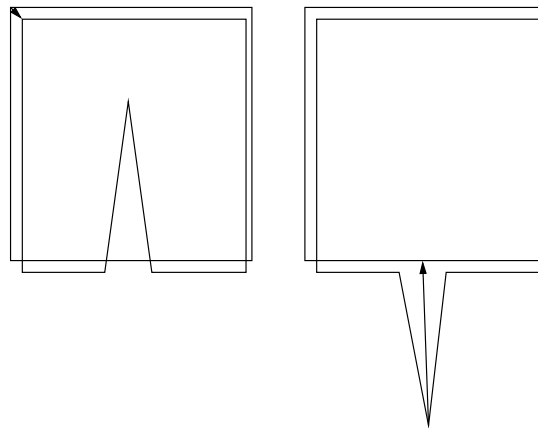


Figure 6.5. Example of the inconsistent fashion in which the Hausdorff Distance treats spikes. The black arrow is the Hausdorff distance in both images

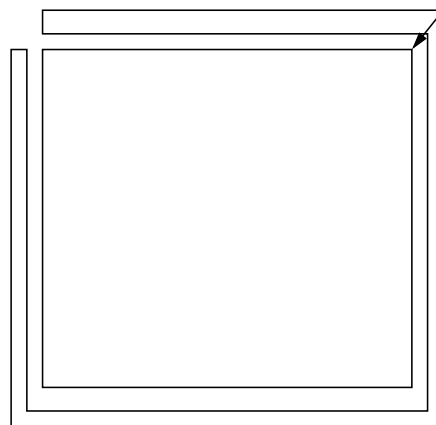


Figure 6.6. Example of the small Hausdorff boundary separation given when one footprint encloses another.

to get an approximation. Either method is made more difficult when the shapes are non-convex and/or they have cavities and, of the two approaches, only sampling can allow for degenerate parts. We note that Alt's paper is not clear on how the simplex approach finds the correct Hausdorff distance if one of the 'ends' of the line from which the distance is taken is within the interior of one of the simplices. Computing the Hausdorff boundary

separation is a little simpler, and an efficient approach was given by Alt *et al.* in 1993 [Alt *et al.*, 1993]. Alt *et al.*'s approach involves computing the Voronoi Diagram of the vertices of the footprint and using a plane sweep algorithm to find the intersections of the other footprint with the edges of the Voronoi Diagram.

Davis [Davis, 1995] gave a description of the dual-Hausdorff distance and how it may be computed. The paper is highly technical, and is concerned with identifying whether mathematical abstractions of objects are similar enough to their real world counterparts for mechanical calculations performed on them to be accurate. It suffices for this thesis to note that assuming the Hausdorff distance can be computed and that there is a sensible limitation placed on the space the footprints inhabit (to provide a boundary for their complements) then dual-Hausdorff provides a more sensitive measure than the Hausdorff distance and does not fall prey to its unfair treatment of spikes.

When considering how the Fréchet distance could be implemented to provide a measure of difference between the two footprints there are some immediate problems. The first is in deciding where the paths for the boundaries start; some footprint algorithms build up their footprint by defining a path, like the Swinging Arm algorithm, others build the edges with no respect to ordering, like the α -shape. Even if a specific extremal point were chosen as the start and end points, for example the upper left hand vertex, it would still not be clear how a cavity or a separate component would be dealt with. Further there is the idiomatic difference in that, despite considering degeneracies, this thesis has mostly considered the footprint as a region, and not as the boundary describing a region. Given the above it would seem that the Fréchet distance is not a good fit for the footprint similarity measure.

The strictures that were chosen when discussing the nature of footprints in Chapter 4 identify two ideologically distinct parts. Initially we describe the footprint as a region instead of a boundary or path. However the discussion also indicated that degenerates could well be desirable components of the footprint, and a degenerate, by definition, cannot be described as a region. (Dis)similarity measures that look only at the boundary of the path will ignore the region it encloses while measures that concern themselves with the encompassed region will be unable to take degeneracy into account.

The five measures can loosely be described as falling into either region measures or boundary measures: There is also a difference in the order of their returned units. The assorted

Region	Boundary
Hausdorff distance	Hausdorff boundary separation
dual-Hausdorff distance	Fréchet distance
Symmetric area difference	

Table 6.1. Difference in approaches for shape similarity measures.

Hausdorff measures and the Fréchet distance all return scalar distances whereas the symmetric area difference returns a measure in the same dimension as the footprints. The Fréchet distance also relies on a 1-dimensional path to exist and would not work with non-planar footprints, for example if the footprints were both spheres.

Within the above discussion there have been clear negatives and positives identified for all of the different measures and these are re-iterated here for clarity. Both Hausdorff distance and Hausdorff boundary separation have types of shapes for which they do not provide accurate measures of distance. The dual-Hausdorff measure is fair but is not straightforward to implement and, like Hausdorff distance, has difficulties when the end point for one of the distance measures exists in the interior of one of the footprints. The symmetric area difference measure has no way of dealing with degeneracies save to ignore them. Fréchet distance requires that a directed path can be decided upon for both footprints, and it is not clear how this would be best achieved, particularly if the footprints contain cavities, multiple components or degeneracies.

We note that, for the purposes of this thesis, symmetric area difference is a good candidate similarity measure. It returns a value in the same dimensional order as the footprints, and therefore loses less information about the difference. It also treats the footprints as regions; fitting in with the requirements given in Chapter 4. It is also comparatively simpler to compute than the other measures as it requires only that the intersections between the edges be found. The only problem is in its treatment (or lack thereof) of degeneracies. It is a difficult decision and this thesis still expresses the view that degeneracies cannot be ignored for the definition of the footprint to be valid. We reconcile this with the observation that all the examples of degeneracies that we have encountered are ‘edge cases’, that is, for most of the applications considered within this thesis we would not expect the footprints to have common or particularly large degeneracies. As such we assume that the symmetric area difference measure is close enough to the ‘true’ dissimilarity between two footprints, and is, therefore, still valid as a measurement of the error produced when computing the footprint using change identifiers.

Since we are only interested in comparisons, not actual values, we normalise the error by expressing it as the symmetric area difference taken as a fraction of the area of the true footprint ($foot(\phi_i)$). Thus the aggregate mismatch between the stored footprint and the true footprint over a dot-pattern sequence of length p is given by

$$mismatch = \sum_{i=1}^p \frac{\|foot(\phi_i)\Delta SFP_i\|}{\|foot(\phi_i)\|},$$

This *mismatch* bears a strong resemblance to the visual error used by Alani *et al.* [Alani *et al.*, 2001]. The similarity has arisen independently and was noticed only after *mismatch* was implemented, but the concurrency adds credence to our choice of error measure.

6.7. Eliminating Wasteful Processing

As an addendum to considering the methodology we note a way 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 phase 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 other identifier that may require it.

6.8. Summary

This chapter has provided a modular framework for the change identifiers to be run within that allows both for real-world application use and for assessment. A data structure to contain the dot patterns has been proposed and rationalised. When considering the assessment of the change identifiers the chapter has explored three footprint difference measures and shown why symmetric area difference has been used for the experimentation within this thesis. Finally this chapter has discussed a method to reduce wasteful computation when running the change identifiers framework.

7. Results

The previous chapters have examined the construction of the change identifiers, how they may be combined and used, and a methodology with which to test them. This chapter discusses the parameters used within the testing, a justification on why they will be sufficient to fairly assess the change identifiers and finally presents the results from the experimentation.

7.1. The Dynamic Dot Patterns

The first hurdle faced when attempting to create a test-bed for change identifiers is in the choice and/or creation of the dynamic patterns. Using just one type of dynamic dot pattern will produce unreliable results, and may lead to overfitting the change identifiers to just that application. While there is a large number of actual and potential applications which generate this data there is a dearth of available examples. Therefore, as well as the few real-world cases, we need a set of example dynamic dot patterns exhibiting a range of behaviours. A pattern generator was constructed to allow for a variety of different behaviour types including the random type used for the dot pattern analysis in Chapter 3. The dynamic dot patterns used as input for the change identifier framework are:

Real World:

- Ship tracking data¹ – Data taken by VHF tracking using the Automatic Identification System (AIS). This data set covers the English Channel for two 24 hour periods in March 2011 and shows an interesting range of movement types² [Mean No. dots: 105, No. phases: 289].
- Running data³ – GPS Data that tracks runners over the Great West Run in Exeter. This set is quite small and because of the way the GPS data is collected operates over very few widely spaced timesteps [Mean No. dots: 9, No. phases: 11].

Generated Simple:

- Translation – The dynamic dot pattern moves but has a fixed distribution from the centre [No. dots: 500, No. phases: 500].

¹Courtesy of David Walker.

²Examples of which can be seen in [Wood, 2011].

³Courtesy of Dr Zena Wood.

- Extent (Translation) – The dynamic dot pattern changes in extent by dot translation [No. dots: 500, No. phases: 500].
- Extent (Cardinality) – The dynamic dot pattern changes in extent by changes in cardinality [Mean No. dots: 500, No. phases: 500].
- Rotation – The pattern is fixed in place but rotates around a centre [No. dots: 500, No. phases: 500].

Generated complex:

- Orbits⁴ – A Pattern that simulates the orbit of six objects around the origin. While an application with this behaviour is unlikely to require a footprint it provides a layer of complexity to the rotation that may occur in other applications [No. dots: 6, No. phases: 121].
- Random – In which no pattern bears any resemblance to the previous pattern. Change occurs often in different ways, almost certainly requiring an update at each step. This acts as a form of control pattern to make sure that extreme changes are always ‘caught’ [Mean No. dots: 275, No. phases: 500].
- Simple with noise – The simple behaviours described above with noise applied to their position or movement. This is to make sure that the identifiers can still perform even with imprecise data [No. dots: as above, No. phases: as above].
- Boid-like behaviour – Reynolds [Reynolds, 1987] identified three rules that an entity can follow which, when observed by a flock of entities, produces complex flocking behaviours:
 1. Alignment – The tendency to travel in the same direction as local flockmates.
 2. Separation – Steering to avoid collision with local flockmates.
 3. Cohesion – The tendency to move toward the average position of local flockmates.

We use a two-dimensional variation of this structure and add the concept of a variable lifespan to the entities. Having a lifespan is not necessarily to account for the possibility of the entities dying but also to allow for possibly incomplete or duplicated data [Mean No. dots: 591, No. phases: 500].

The work performed by Andrienko and Andrienko [Andrienko and Andrienko, 2007], and Laube and Purves [Laube and Purves, 2011] provides a good summation of the dangers inherent in testing across limited types of dot pattern data. The wide range of types of pattern are used so that we can avoid some of the more common pitfalls. We are, however, aware that a large proportion of our data is computer generated rather than representative of real world data, a fact we must bear in mind when drawing our conclusions.

⁴Courtesy of Dr Antony Galton.

7.2. The Algorithms

The aspects of the data produced by running the identifiers in which we are most interested are the ‘error’ presented by the change identifiers and the time taken to run them against the time taken to run just the footprint algorithm. To be sure of a fair representation we need to use more than just one footprint algorithm. We have endeavoured to use algorithms that produce footprints in different fashions so as to give a wide range of results on which to draw our conclusions.

- **Graham Scan** – Graham [Graham, 1972] produced one of the first efficient ($O(n \log n)$) algorithms for finding the convex hull. The paper [Graham, 1972] is a succinct mathematical description of the algorithm, so to avoid repetition we simply note that it expresses all the dots of a pattern in polar coordinates from an origin point known to lie within the pattern, and from this origin successively removes points that cannot be extremal.
- **α -shape** – As described in Chapter 2 the α -shape is the conjunction of all circles of radius $1/\alpha$ that contain either all/none⁵ of the dots of the pattern. Choosing a parameter that would always produce footprints in between the convex hull and the null footprint is not easy (rather tentatively called ‘interesting’ footprints for the sake of brevity). The average estimated nearest neighbour distance⁶ was used as a likely candidate.
- **χ -hull** – The χ -hull [Duckham et al., 2008] relies on the Delaunay triangulation. Using a divide and conquer method this can be found in $O(n \log n)$ time and the χ -hull process then removes lines of length greater than the given parameter when removing them does not ‘break’ the hull. Like α -hull a parameter value needs to be chosen that will produce ‘interesting’ footprints. The parameter also has a direct effect on the process time of the algorithm, the smaller the line length the more lines there will be to check to see if they can be removed. The average estimated nearest neighbour distance provides a value that fits these requirements⁷.
- **Descending Swinging Arm** – The Swinging Arm algorithm [Galton and Duckham, 2006] can be extended to function as an iterative process in which successive arm lengths are tried. The descending version begins with an arm length equal to the diameter of the pattern, a value guaranteed to give the convex hull. Then by choosing the second longest side length of the footprint the arm length decreases until the footprint contains a degenerate line, at which point the footprint previous is selected.
- **Ascending Swinging Arm** – This version of the Swinging Arm algorithm starts with the average nearest neighbour distance and increases in increments of the shortest nearest neighbour distance⁸ In comparison to the descending swinging arm al-

⁵All for positive α and none for negative.

⁶Multiplied by -1 to give a α -shape with concavities.

⁷It should be noted that we are not suggesting that the average nearest neighbour distance always provides a ‘good’ footprint but that it tends to produce an ‘interesting’ one.

⁸Or by a quarter of the average if the shortest is less than that value. This is done so as to reduce to the

gorithm, the ascending version is more likely to produce footprints with multiple components.

- Minimum Isothetic Bounding Box – The dual red-black trees allow us to find the extremal dots in the x and y coordinates in $O(\log n)$ time. It is likely, therefore, that the minimum isothetic bounding box can be computed in less time than some of the change identifiers. It is included to make sure that any positive bias towards change identifiers from the implementation of any of the above algorithms is balanced.

Adding to the earlier discussions on the nature of a footprint we note that from this point on this thesis will be dealing with footprints as defined by by the given algorithms. Whether the change identifiers can generalise to be used for other footprint algorithms is left as an open question. However we can state that many of the algorithms in the literature will produce similar types of footprint⁹ to the algorithms presented here.

7.3. The Change Identifier Sets

As this chapter is looking to make assessments on the general ability of change identifiers to reduce footprint updates, there will need to be a range of change identifier sets. To provide this range several sets are created: A set for each of the identifiers introduced in Chapter 5 as singletons, a set for the change identifiers based on the fastest descriptors from Chapter 3 (i.e., SKEWNESS, KURTOSIS, PEARSONCOEFFCORR, OLSGRADIENT, CARDINALITY and DIAMETERSQ) a set for the fastest descriptor change identifiers in conjunction with the distance measures from Chapter 5, and a set containing all of change identifiers. For each of these twenty sets a version is created with the total thresholds 0.1, 0.25, 0.5, 0.75 and 0.9 giving a total number of change identifier sets used of one hundred. With a hundred different change identifier sets there should be enough scope to see sets that reduce the number of updates for a suitably small trade-off in error.

7.4. Plotting the experiments

Given a set of dynamic dot patterns, a set of footprint algorithms and a collection of change identifier sets we need to decide on an appropriate, defensible procedure with which to test them. Before explaining the procedure we add some useful nomenclature shortcuts: each process of a dynamic dot pattern with a footprint algorithm and a change identifier set is called a *run* of the framework; each run produces a graph of time taken with change identifiers and without against timesteps (Fig. 7.1) and a graph of error against timestep (Fig. 7.2). For brevity we shall refer to the run over the dynamic dot pattern when using change identifiers to signal footprint updates as **with** (i.e., *with* change identifiers) and the run when updating the footprint at each time step as **without** (i.e. *without* change identifiers).

number of iterations.

⁹The footprint type here are those given in the classification in Chapter 4

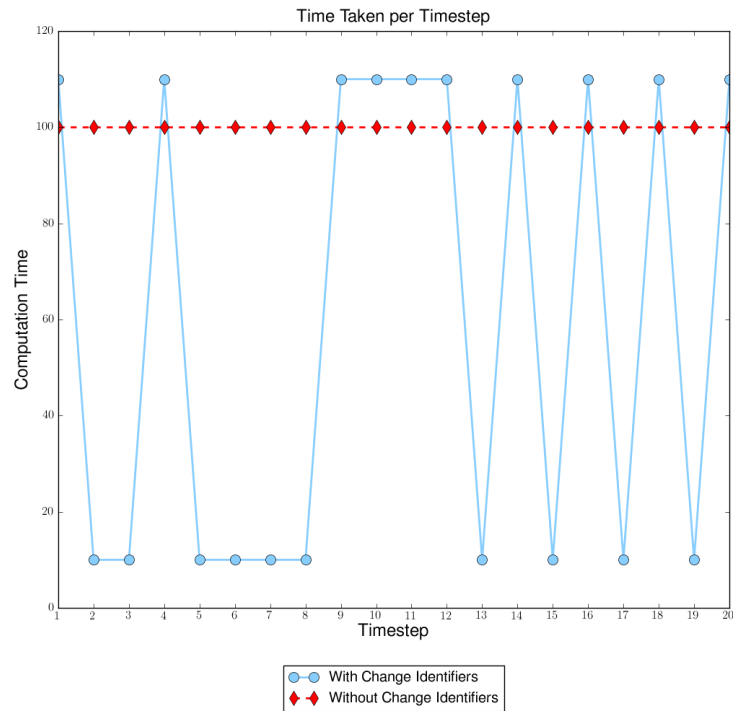


Figure 7.1. Example of a graph of time taken per timestep.

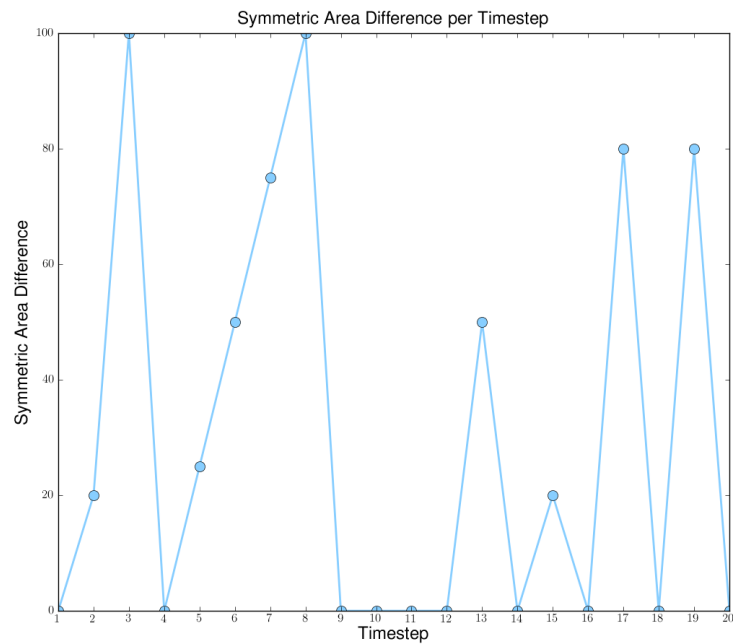


Figure 7.2. Example of a graph of symmetric area difference per timestep.

The graphs shown in Fig. 7.1 and Fig. 7.2 are of the same hypothetical experiment, as evidenced by the correlation of their ‘spikes’. Each update of the footprint, when using change identifiers has a ‘jump’ on the time-taken graph (Fig. 7.1) and a corresponding ‘drop’ to 0 on the area difference graph (Fig. 7.2). The jump-height is the time taken to run both the change identifiers and the footprint algorithm, and the drop to 0 indicates that the footprint from the change identifier run is identical to the true footprint. Fig. 7.2 shows

two types of slope that represent different changes in speed in the dynamic dot pattern: *gradual* and *steep*. The phases in the timestep range 4–8 show a gradual increase in the error, represented by the symmetric area difference, until 9 at which point the thresholds were exceeded on the change identifier set that was being used. The steep slopes between 16–20 are indicative of much faster change than in the 4–8 range as the symmetric area difference increases far more between each timestep. Further to showing the rate of change, the symmetric area difference can also denote different change behaviours in the dynamic dot pattern: *uniform* and *erratic*. Both the ranges 4–8 and 16–20 show uniform change in the dynamic dot pattern. The fast rate of change between 16–20 means that, unlike 4–8, there is no obvious ascension, but, because timestep 19 has the same symmetric area difference as 17, 16–20’s change is likely similarly as uniform as 4–8’s. Whereas between 12 and 16 the symmetric area difference follows the erratic sequence $\langle 0, 50, 0, 20, 0 \rangle$. The graph shows that a symmetric area difference of at least 100 can be reached before an update occurs (timesteps 3 and 8), therefore there must be large changes between 13–14, and 15–16.

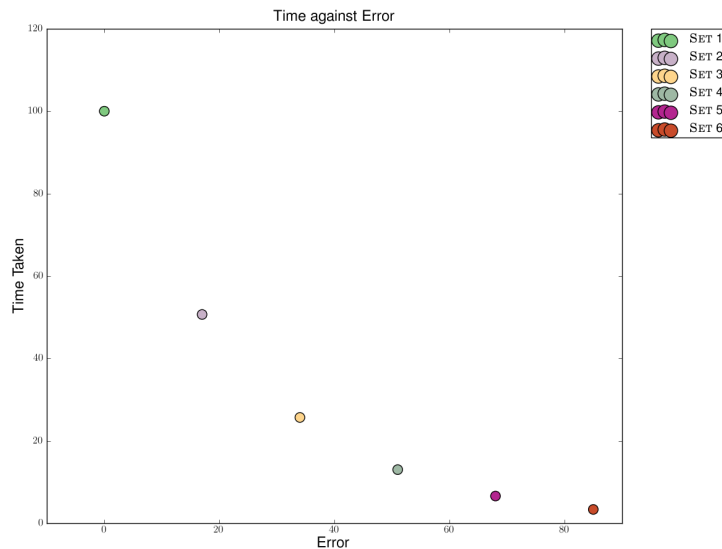


Figure 7.3. Example of a graph of time against error for multiple change identifier sets.

The previously discussed graphs display the effectiveness of a particular set for a specific run. However they are not very useful for comparing identifiers against each other; one set may take longer than another but produce less error. By iterating over the change identifier set collection we also fill in points on a graph of time taken against error Fig. 7.3. Each point on Fig. 7.3 represents the performance of that change identifier set on the given dynamic dot pattern for the given footprint algorithm. Looking at time against error graphs we can instantly see if an identifier is performing better than another at minimising time and/or error.

Altering the algorithm will not change the performance of the identifier sets relative to each other but it will affect the values given to their performance. For example the time differences between **with** and **without** will be far greater for the Ascending Swinging Arm algorithm than for the Graham Scan algorithm. By taking an average performance score

over the set of footprint algorithms we can be sure that we are neither penalising the change identifiers nor being overly biased towards them. The results can be generalised further by taking an average performance score over the set of different dynamic dot pattern types. With these considerations in place we can make confident statements about the performance of a change identifier set in general as well as for a specific dynamic dot pattern type (e.g., A change identifier set may perform particularly well when the pattern changes only by extent but terribly for a dynamic pattern which rotates.)

7.5. Results

There are one hundred identifier sets running over twenty-one dynamic dot patterns using six footprint algorithms, leading to 12,600 individual runs. Each run has values for total time taken, total symmetric area difference (error), average time taken per timestep, average symmetric area difference per timestep and average symmetric area difference as a proportion of the area of the true footprint per timestep. With such a large amount of data it is impractical to plot all the results on a single graph or to plot each run individually, so many of the following graphs show only the best performing thresholds for each set and with their values averaged over the footprint algorithms.

7.5.1. Footprints

Results

Before showing the results of the change identifiers, we present an example of the dot pattern and the footprints both with and without change identifiers for sixteen timesteps of the pattern changing by extent and cardinality (Fig. 7.4 and Fig. 7.5) to give some context to the following charts.

Next the footprint algorithm times are graphed so as to provide context for the times taken when performing **with** runs. Fig. 7.6 shows the average time taken per timestep to run each footprint algorithm for each of the dynamic dot patterns, and Fig. 7.7 shows the same plot but for only the minimum isothetic bounding box algorithm.

Discussion

The α -shape algorithm consistently takes longer to compute than the others; it seems likely that this high computation time is due to the iterative construction process of the algorithm that contains regular distance checks (to see if any dots fall within the discs that are defined by its candidate edges). The expectation would be that the minimum isothetic bounding box, being the computationally least complex, would be the fastest to compute and this is shown in the data. We note that the computation time appears to be almost constant but this is an effect of the logarithmic scaling and disappears when the

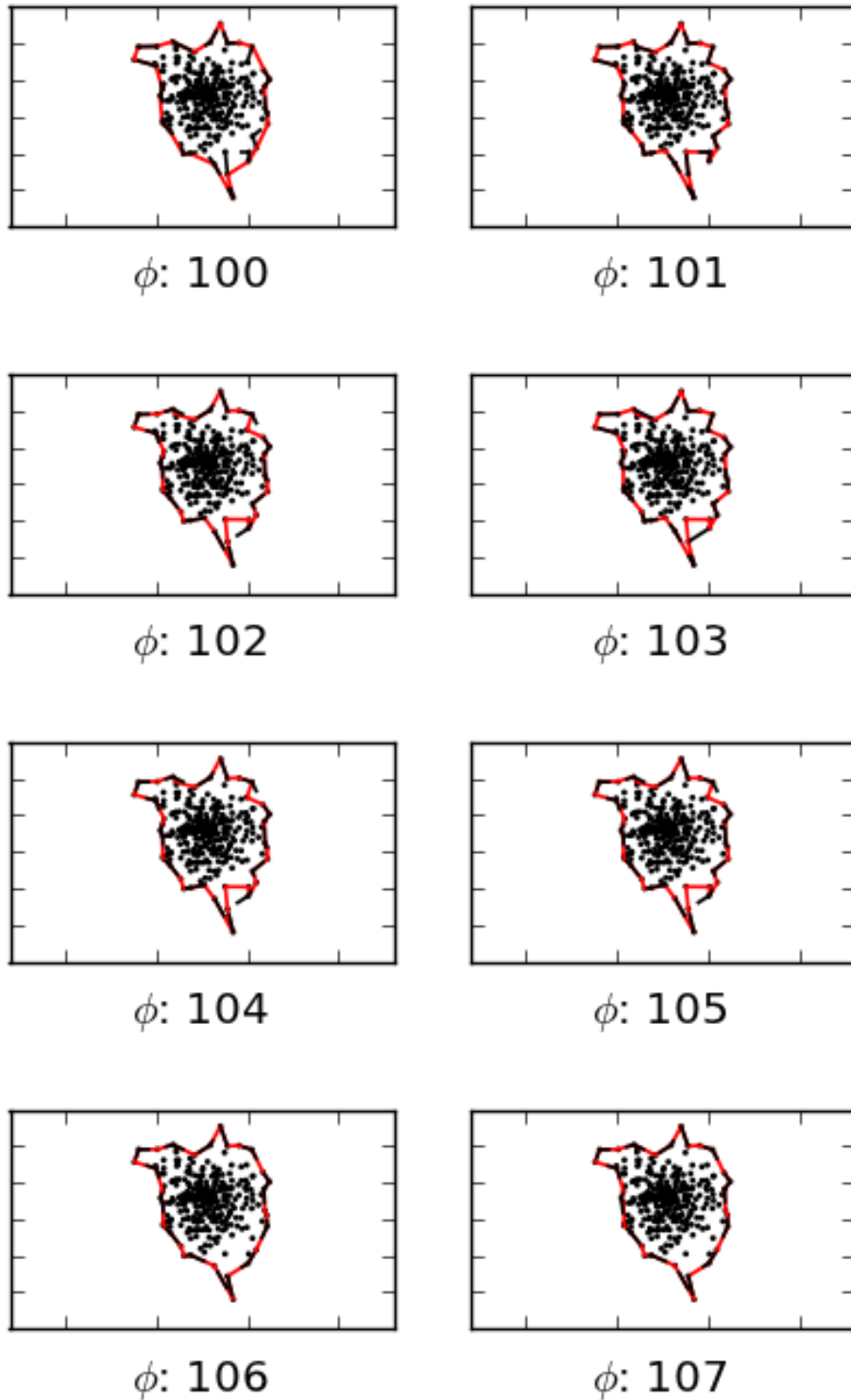


Figure 7.4. An example selection of eight timesteps ($\phi_{100} - \phi_{108}$) showing the footprint with change identifiers (solid red) and the footprint without change identifiers (dashed black) on the dynamic dot pattern using a change in extent and cardinality behaviour and the χ -hull algorithm

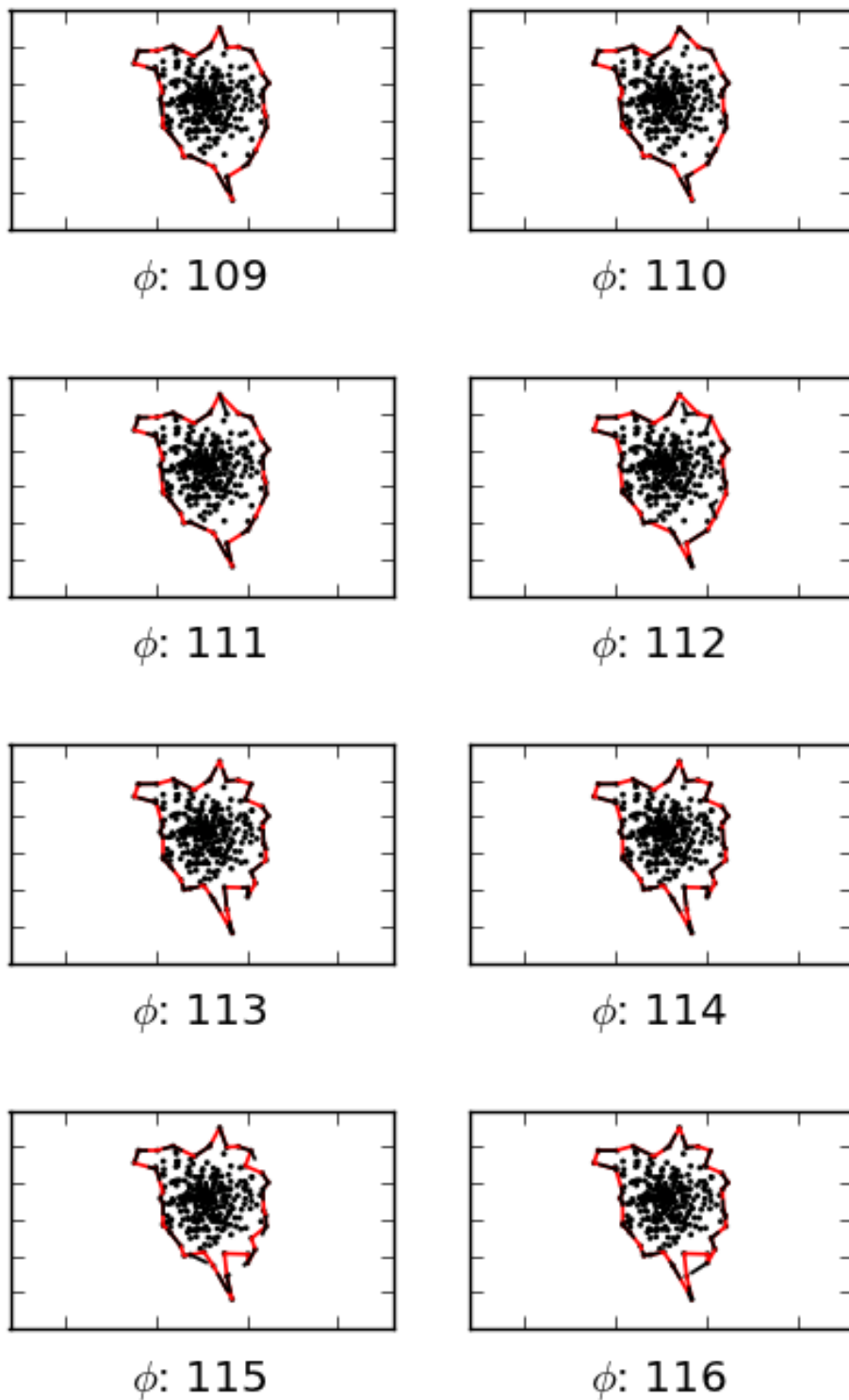


Figure 7.5. An example selection of eight timesteps ($\phi_{109} - \phi_{116}$) showing the footprint with change identifiers (solid red) and the footprint without change identifiers (dashed black) on the dynamic dot pattern using a change in extent and cardinality behaviour and the χ -hull algorithm

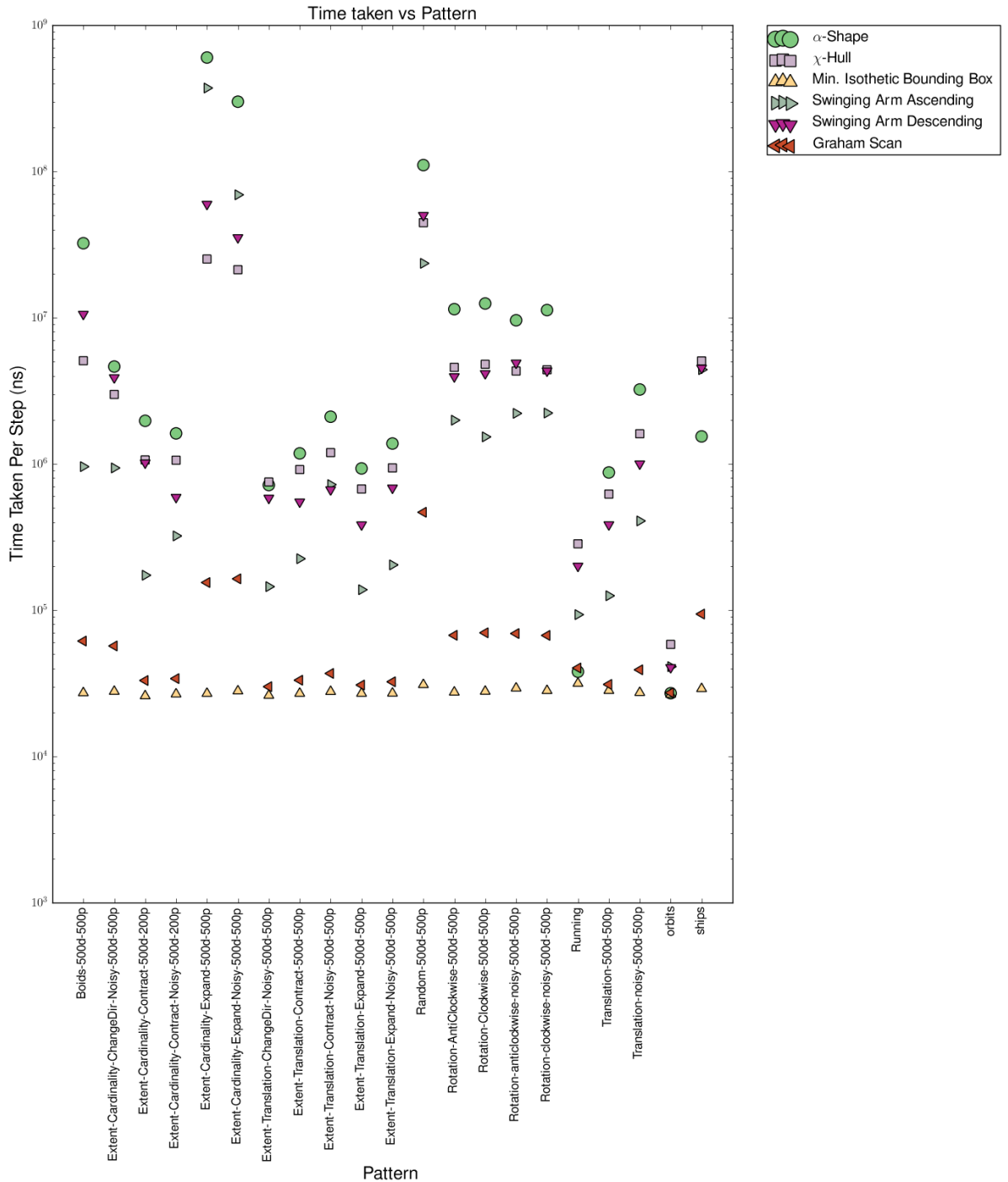


Figure 7.6. Average time taken per timestep for each footprint algorithm on each dot pattern type.

bounding box computation time is looked at on a linear scale (Fig. 7.7). In Fig. 7.7 the bounding box on the running data (the smallest dynamic dot pattern used) appears to be the slowest to compute, however the difference between it and the fastest computation time is 2.53×10^{-6} seconds¹⁰. which is small enough that it is likely due to the processor being

¹⁰Recorded as 2529.06 nanoseconds. The nanosecond timer in the Java framework is its most accurate way of identifying time differences, however whether or not it has nanosecond accuracy is entirely dependant on the capabilities of the processor, thus we can largely trust its accuracy to the millisecond but no further

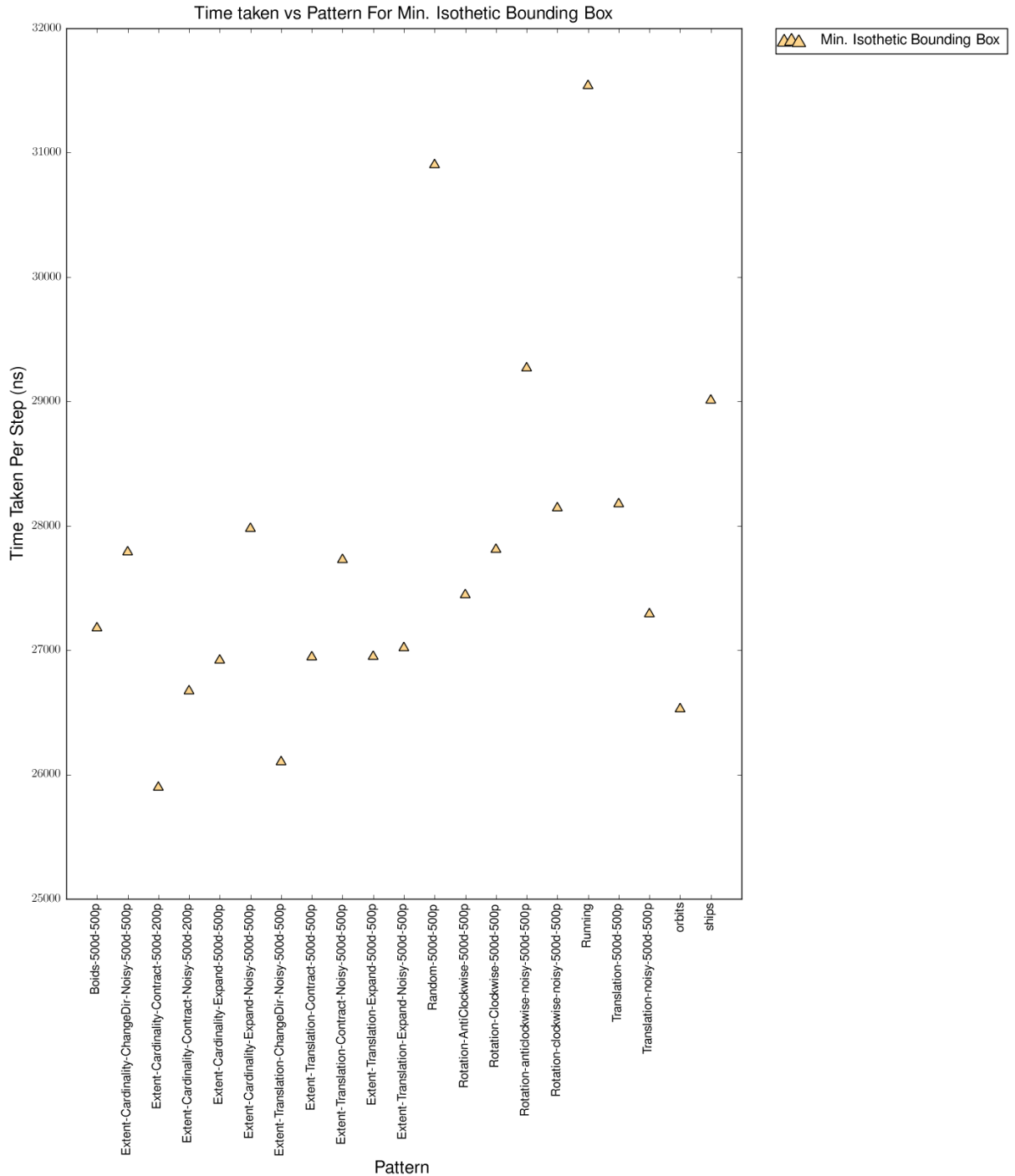


Figure 7.7. Average time taken per timestep for the minimum isothetic bounding box algorithm on each dot pattern type.

used by other tasks momentarily. We tend to ignore such small time differences in our analysis as they are misleading and almost certainly anomalous. This thesis does not focus on the footprint algorithms and their differences, but future work could look at answering such questions as: Why does the Ascending Swinging Arm tend to compute faster than the Descending variant? And why does the χ -hull only perform quicker than the both of the Swinging Arm variants when the dynamic dot pattern has featured expansion by

increasing cardinality?

7.5.2. Change Identifier Sets Time

Results

Fig. 7.8 is a time-taken graph for the change identifier sets in which only the thresholds with the minimum time are plotted for each set; that is, each change identifier set was run with the thresholds $\langle 0.1, 0.25, 0.5, 0.75, 0.9 \rangle$ and only the threshold with the minimum time is shown here. The white diamond indicates the average time taken for a **without** run to be performed on the dynamic dot pattern.

Discussion

The thresholds are all 0.75 or 0.9 and therefore to the high end of the range that was specified. This is expected as the higher the threshold the less likely the change identifiers are to cause an update. The reason they are not all at the 0.9 threshold is that, for those with a 0.75 threshold, the number of footprint updates for both runs using 0.75 and 0.9 are the same and therefore they have the same computation times; the lowest threshold is plotted as a default. Important to note on Fig. 7.8 is that all the change identifier sets, apart from on the running data, take less time than the **without** run. The dynamic dot pattern using the real world running data is very small with only eleven timesteps and nine data points in each pattern, so it is not surprising that the `BOUNDINGBOXSYMMETRICAREADIFF-0.75` and `CENTROID-0.25` identifiers take longer than the mean time of the footprint algorithms, particularly considering that the set of footprint algorithms contain the isothetic minimum bounding box and the Graham Scan.

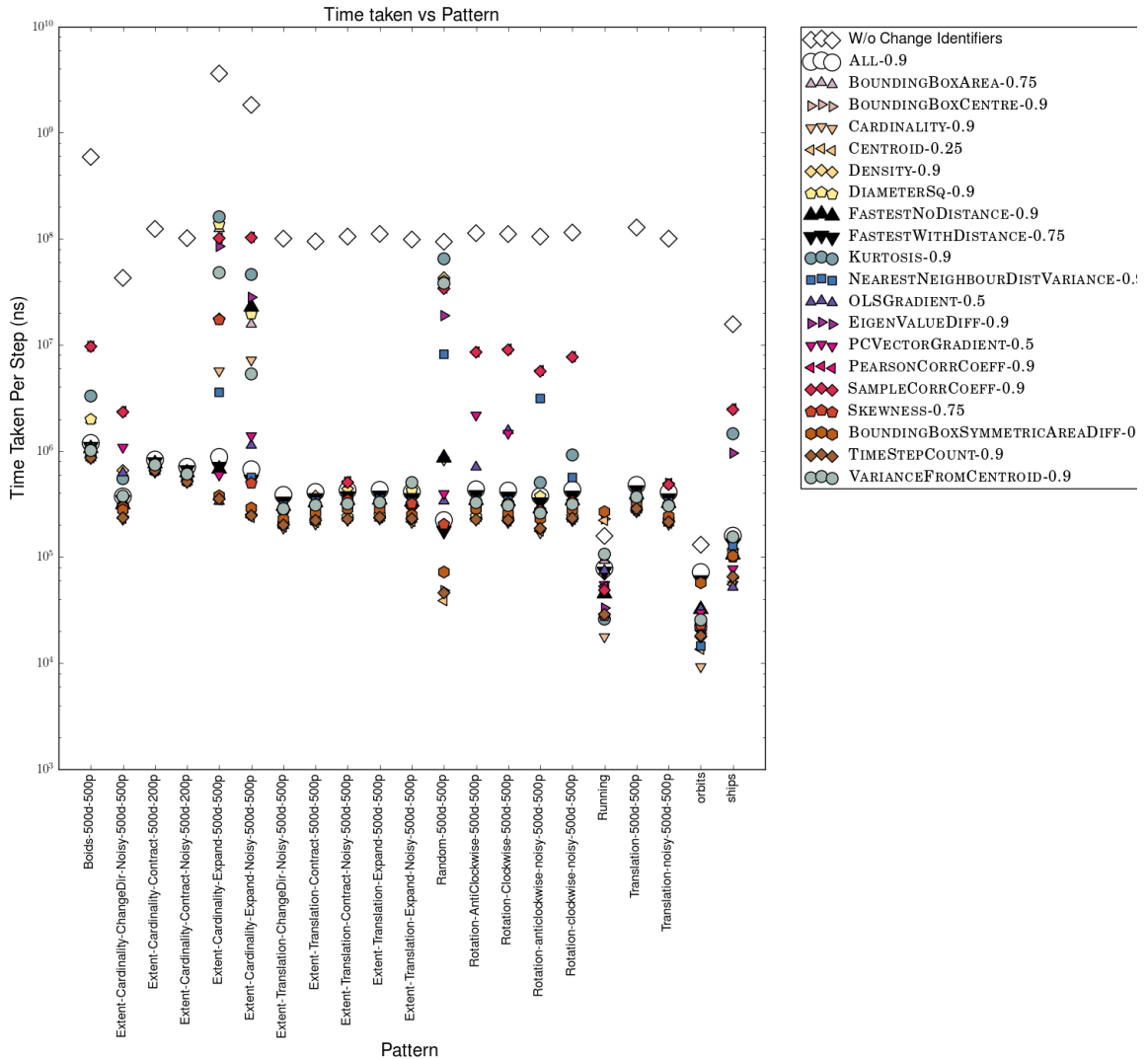


Figure 7.8. Average time taken per timestep for the fastest performing thresholds of each change identifier set on each dot pattern type.

7.5.3. Change Identifier Sets Error

Results

Unlike Fig. 7.8, Fig. 7.9 plots the average proportionate error (found using symmetric area difference and the *mismatch* equation from Chapter 5) per timestep instead of the time taken. The graph shows the thresholds for each identifier set that minimised the error; all of which are 0.1.

The graphs in Fig. 7.10, Fig. 7.11 and Fig. 7.12 show the error for three different change identifier sets (a set containing all identifiers and the singletons of BOUNDINGBOXAREA

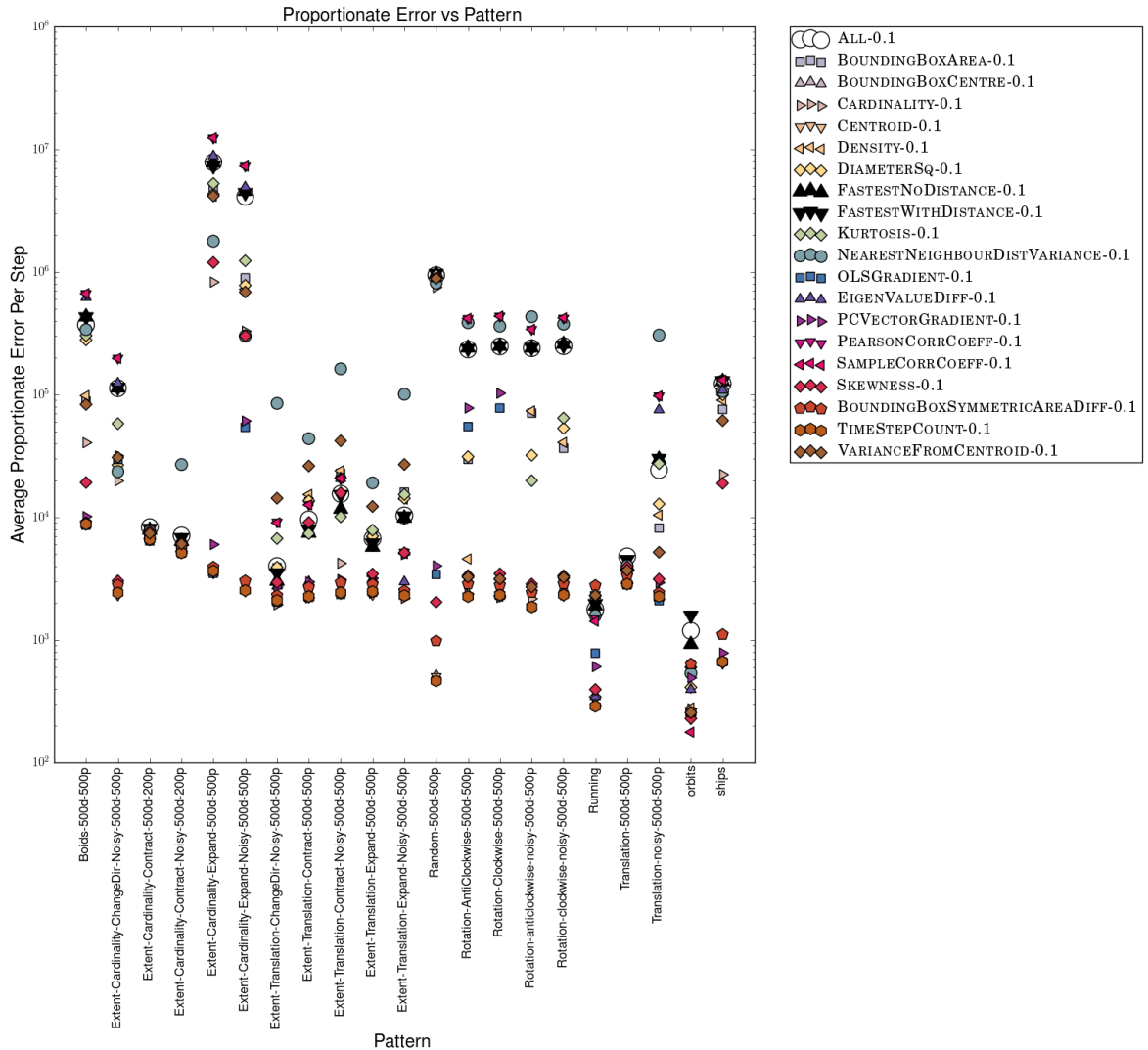
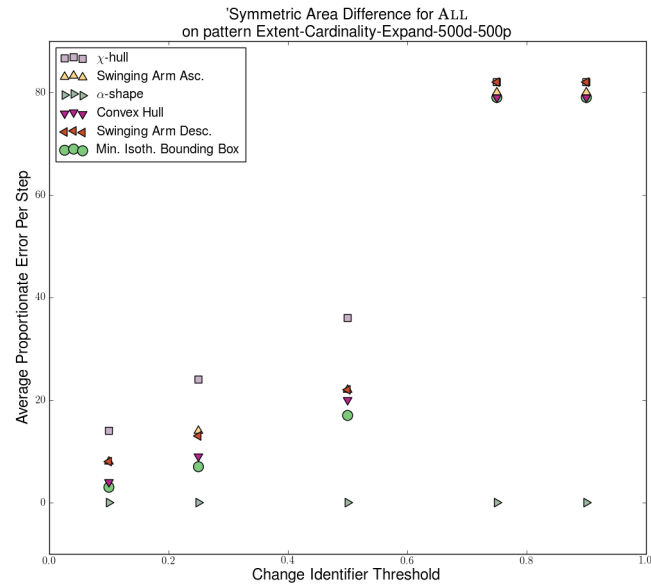


Figure 7.9. Average proportionate error per timestep for the best performing thresholds of each change identifier set on each dot pattern type.

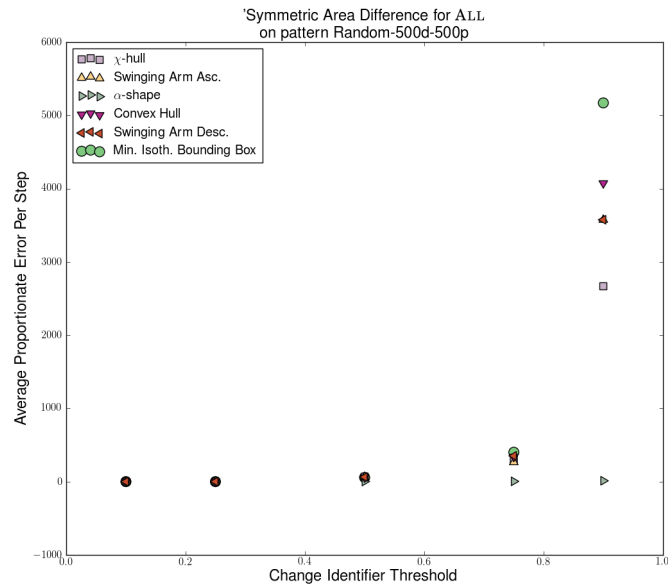
and NEARESTNEIGHBOURDISTVARIANCE respectively) for all algorithms on two different dynamic dot patterns.

Discussion

The lower the threshold the more likely the change identifier set is to cause footprint updates, and the more frequent the footprint updates the less the symmetric area difference between the stored footprint and the true footprint will be. A final point of interest on this graph is that the sets do not appear in the same order for every dynamic dot pattern type indicating the, perhaps intuitive, fact that different dynamic patterns change in different



(a) On the change in extent via cardinality dynamic dot pattern.

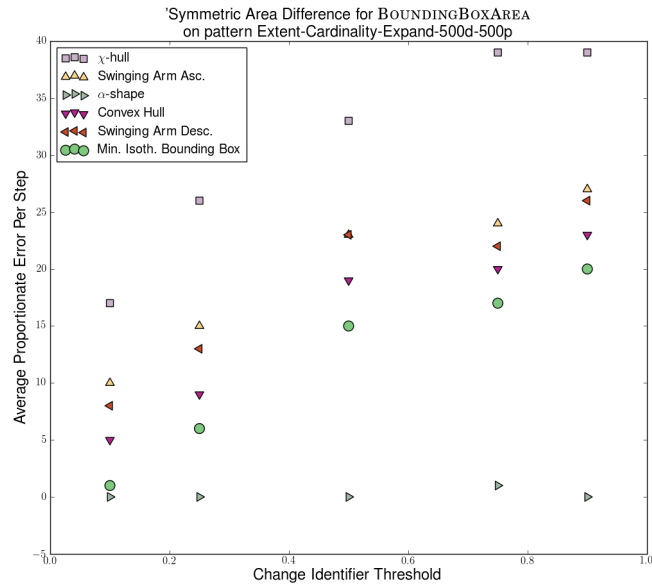


(b) On the random dynamic dot pattern.

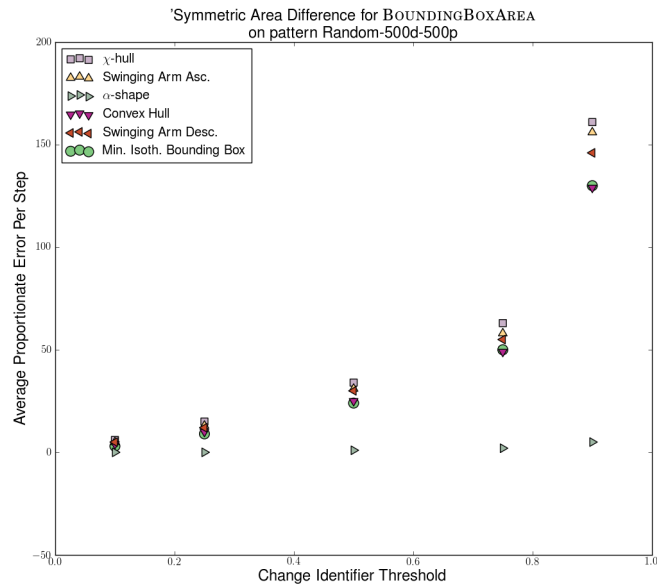
Figure 7.10. Graphs showing symmetric area difference against change identifier threshold for the set containing all identifiers for different thresholds.

ways and that an identifier that can catch one type of change will not necessarily catch another.

The error graphs for change identifiers with different thresholds look much as would be expected, for the pattern that represents a phenomenon changing in extent the error (as measured by proportional symmetric area difference) increases as the threshold increases. The random pattern has given an odd result for the `NEARESTNEIGHBOURDISTVARIANCE` set (Fig. 7.12), the fact that the threshold of 0.9 gives less error than 0.75 appears anomalous. However if we consider that there is a timestep τ_v within the pattern after which



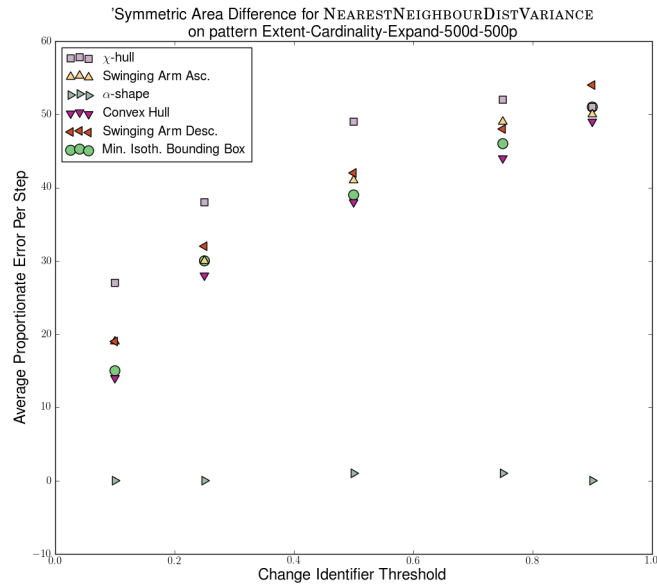
(a) On the change in extent via cardinality dynamic dot pattern.



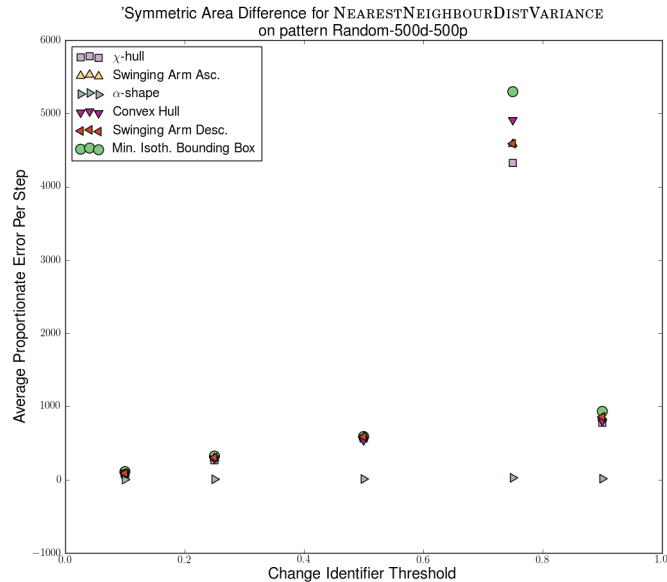
(b) On the random dynamic dot pattern.

Figure 7.11. Graphs showing symmetric area difference against change identifier threshold for the BOUNDINGBOXAREA set identifier for different thresholds.

the pattern varies wildly but not in the nearest neighbour distance variance then we can imagine that the threshold of 0.75 causes an update at τ_v , but that the threshold is now too high for it to 'catch' the change during this period of wild difference. The 0.9 threshold 'misses' the update at τ_v but the cumulative change in nearest neighbour distance variance allows it to update in the middle of the period of wild change. Such a situation could well lead to a higher threshold producing less error than one lower. This is one of the reasons that multiple change identifiers are recommended for use in the set. Also of interest is that the α -hull seems to only vary very slightly if it all. It is likely that the parameter chosen for the α -hull over these runs is producing footprints of 'poor'



(a) On the change in extent via cardinality dynamic dot pattern.



(b) On the random dynamic dot pattern.

Figure 7.12. Graphs showing symmetric area difference against change identifier threshold for the NEARESTNEIGHBOURDISTVARIANCE set identifier for different thresholds.

quality, that is there may be degenerate parts reducing the region that can be considered as a footprint. With regard to the variation in the change identifier performance based on the footprint algorithm used, excluding the α -hull for the above reasons, the graphs give the indication that the footprint algorithms can have different sensitivities for different types of change. For example the χ -hull and the minimum isothetic bounding box always appear to be at alternate ends, what one is sensitive to, the other isn't. The given graphs are only of two pattern types and as such there is not enough data to make strong claims about the sensitivities of the algorithms, and neither is it particularly important to the discussion on whether change identifiers work for any given footprint algorithm. It is,

however, and interesting avenue for future work.

7.5.4. Time against Error

Results

The error and time against dot patterns graphs have confirmed that the thresholds of the change identifier sets affect their computation times and their error values in the expected fashion. That is, decreasing the threshold increases the time taken for a decrease in the error and vice versa. However, as we discussed earlier, such graphs do not allow for an assessment of change identifiers that takes into account both their time taken and their error, and therefore how the sets compare to each other. Figures Fig. 7.14 and Fig. 7.15 show the average time taken per timestep against the average error per timestep, and the total time taken for the run against the total error for the run respectively for the best performing change identifier set thresholds. Best performing in this case are the sets with thresholds which perform better than others in the same family¹¹ on at least one of the two objectives: reducing the time taken and reducing the error. This form of comparison uses the idea of dominance, a concept used within the field of Multi-Objective Optimisation. The following statements use Fig. 7.13 as an example of dominance relations in which the aim is for both objectives to be minimised.

- Strong Dominance: One solution outperforms another on all objectives. So in Fig. 7.13 $\forall x \in O a_x < b_x$ in which O is the set of all objectives.
- Weak Dominance: One solution out performs another on at least one objective and is no worse on any other. $\exists x \in O d_x < b_x \wedge \forall x \in O d_x \leq b_x$.
- Mutual Non-Dominance: One solution performs better than another on at least one objective and worse on at least one objective. $\exists x \in O \exists y \in O a_x < d_x \wedge a_y > d_y$.

To fully explain the figure: $\langle a, d, e \rangle$, $\langle a, c \rangle$, $\langle b, e \rangle$, and $\langle b, c \rangle$ are mutually non-dominating tuples, b is dominated by a and weakly dominated by d , and c is dominated by d and e . Given this description of dominance it can be stated that the change identifiers that will interest us are those that are mutually non-dominating.

The values in the graphs represent the average result over all dynamic dot patterns and all footprint algorithms, and the blue line in each graph indicates the time taken to complete a **without** run (this **without** run is analogous to the the T_{NCI} line from the trade-off graph Fig. 5.7 in Chapter 5).

Discussion

The change identifier sets have formed definite bands in both graphs. The bands have been enlarged in the subgraphs of each figure so that they may be better examined. Why

¹¹That is, sets that have the same identifiers

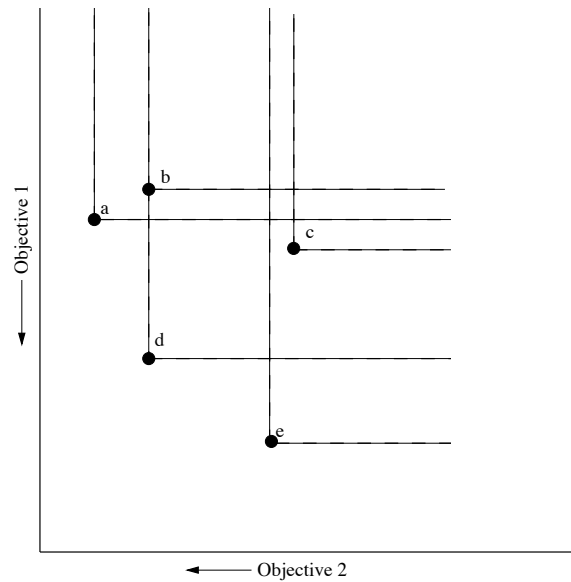


Figure 7.13. Graph showing dominance relations.

the banding should be so distinct is not obvious, however it may be explained if there are set timesteps in the dynamic dot patterns at which not updating causes large increases in the symmetric area difference; the banding would indicate the different granularities at which sets can identify these steps.

The only significant difference between Fig. 7.14 and Fig. 7.15 is that the figure showing totals (Fig. 7.15) has concatenated Bands 3 and 4 from the figure showing averages per timestep (Fig. 7.14) into a single Band 3; this is likely an artifact of the scaling differences between total and average. On both figures Band 2 has a good example of the time-error trade-off curve that was discussed in Chapter 5, in fact all the bands have an indication of this curve but none so strongly realised as in the second band.

All of the identifier sets are well below the computation time when running **without** so we focus on the first band, in which the identifiers have minimised the symmetric area difference. The identifier sets appearing in Band 1 have been re-plotted (with all their best performing threshold values) onto Fig. 7.17¹². We note that once again there is an indication of the time-error trade-off curve appearing. The thresholds of each identifier place them where we would expect on the curve with the lower thresholds to the top-left (more footprint updates so less error and higher computation time) and the higher thresholds to the bottom-right (less footprint updates so greater error but lower computation time). This placement on the curve adds to the evidence given by the separated time and error graphs to bolster the credence of the statement that the time-error curve can be traversed by altering the threshold. This is important because, as previously noted, the effectiveness of the change identifiers will be context specific. Once suitable identifiers have been chosen for an application a user will need to be able to choose a threshold which provides a time-error ratio that is acceptable for that application.

A final note on the time against error graphs is how well the NEARESTNEIGHBOURDIST-

¹²The graphs are only of the average timestep results and not the total results to reduce clutter as both are very similar; differing in only the scales

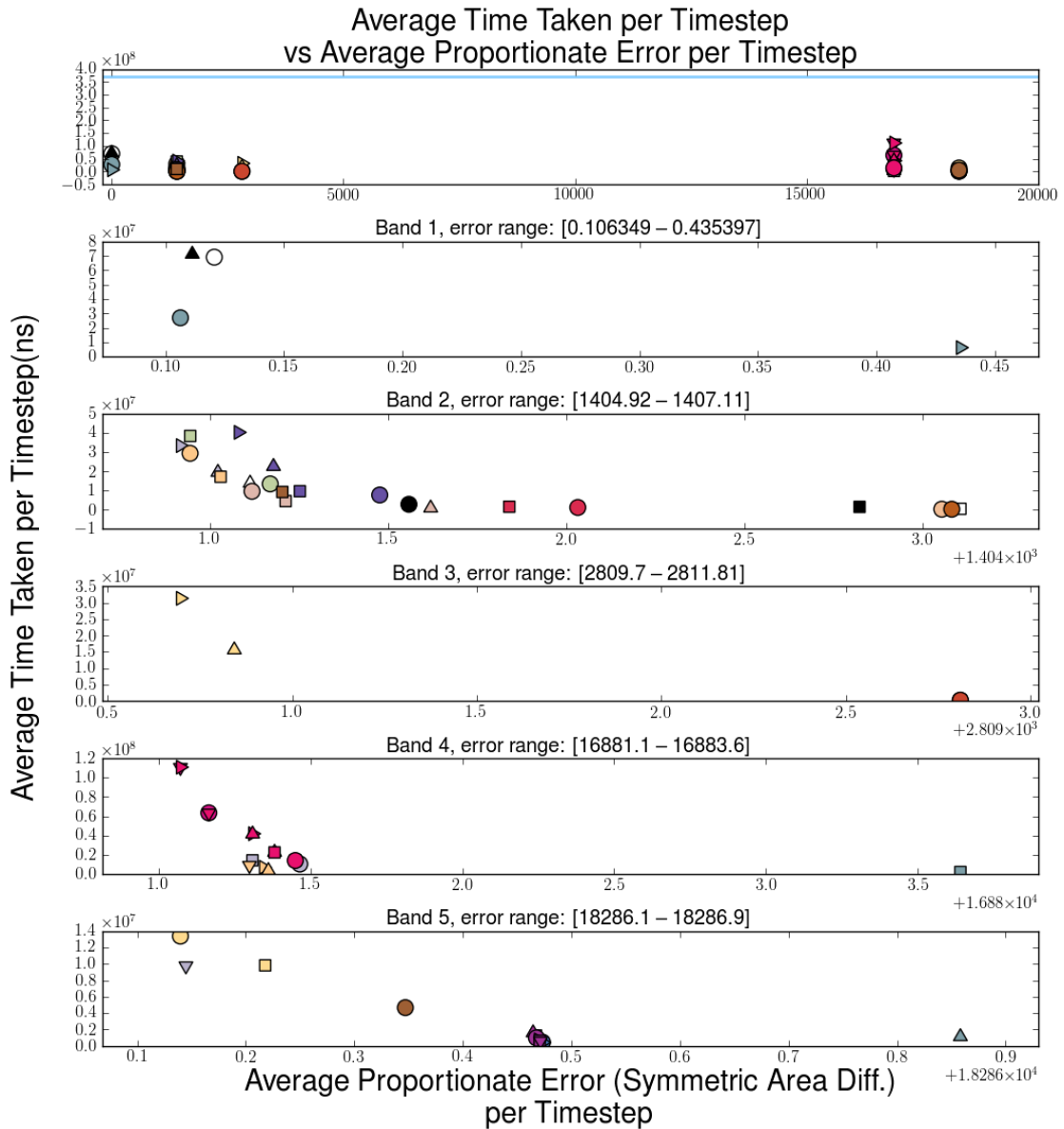


Figure 7.14. Time taken against proportionate error per timestep averaged across all dot patterns for the best performing thresholds of each change identifier set. The first plot is of all the change identifiers together; the subsequent plots are the bands of similar symmetric area difference from the original plot. Note the difference in scales along the x axis. (Legend is given in Fig. 7.16)

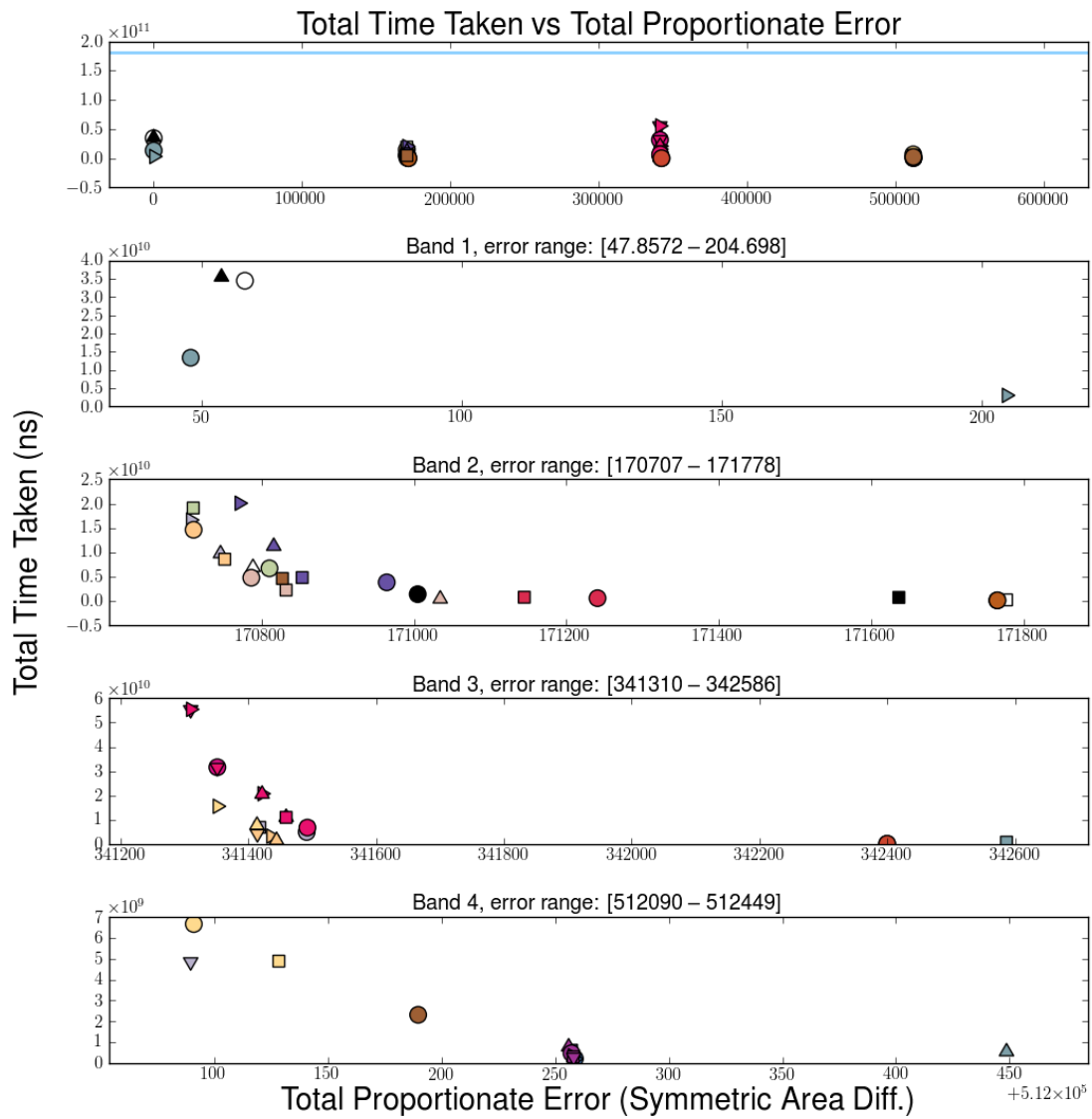


Figure 7.15. Time taken against proportionate error totalled for each run averaged across all dot patterns for the best performing thresholds of each change identifier set. The layout is the same as for Fig. 7.14; the first plot shows all of the identifier sets and the subsequent plots are ‘zoomed’ in sections of this initial plot. (Legend is given in Fig. 7.16)

○○	ALL-0.1
△△	ALL-0.5
□□	ALL-0.9
▷▷	BOUNDINGBOXAREA-0.1
▲▲	BOUNDINGBOXAREA-0.25
▣▣	BOUNDINGBOXAREA-0.5
▽▽	BOUNDINGBOXAREA-0.75
●●	BOUNDINGBOXAREA-0.9
●●	BOUNDINGBOXCENTRE-0.25
▣▣	BOUNDINGBOXCENTRE-0.9
●●	CARDINALITY-0.1
▣▣	CARDINALITY-0.25
▲▲	CARDINALITY-0.9
●●	CENTROID-0.25
●●	DENSITY-0.1
▣▣	DENSITY-0.25
▽▽	DENSITY-0.5
▷▷	DENSITY-0.75
▲▲	DENSITY-0.9
▷▷	DIAMETERSq-0.1
▲▲	DIAMETERSq-0.5
●●	DIAMETERSq-0.75
▣▣	DIAMETERSq-0.9
▲▲	FASTESTNODISTANCE-0.1
●●	FASTESTNODISTANCE-0.75
■	FASTESTNODISTANCE-0.9
▽▽	FASTESTWITHDISTANCE-0.75
▣▣	KURTOSIS-0.1
●●	KURTOSIS-0.9
●●	NEARESTNEIGHBOURDISTVARIANCE-0.1
▷▷	NEARESTNEIGHBOURDISTVARIANCE-0.25
▣▣	NEARESTNEIGHBOURDISTVARIANCE-0.75
▲▲	NEARESTNEIGHBOURDISTVARIANCE-0.9
▣▣	OLSGRADIENT-0.1
▽▽	OLSGRADIENT-0.25
▲▲	OLSGRADIENT-0.5
●●	OLSGRADIENT-0.75
▷▷	OLSGRADIENT-0.9
▷▷	EIGENVALUEDIFF-0.25
▲▲	EIGENVALUEDIFF-0.5
▣▣	EIGENVALUEDIFF-0.75
●●	EIGENVALUEDIFF-0.9
▲▲	PCVECTORGRADIENT-0.1
●●	PCVECTORGRADIENT-0.25
▷▷	PCVECTORGRADIENT-0.5
▣▣	PCVECTORGRADIENT-0.75
▽▽	PCVECTORGRADIENT-0.9
▽▽	PEARSONCORRcoeff-0.1
●●	PEARSONCORRcoeff-0.25
▷▷	PEARSONCORRcoeff-0.5
▲▲	PEARSONCORRcoeff-0.75
▣▣	PEARSONCORRcoeff-0.9
▷▷	SAMPLECORRcoeff-0.1
▽▽	SAMPLECORRcoeff-0.25
▲▲	SAMPLECORRcoeff-0.5
▣▣	SAMPLECORRcoeff-0.75
●●	SAMPLECORRcoeff-0.9
▣▣	SKWENESS-0.75
●●	SKWENESS-0.9
●●	BOUNDINGBOXSYMMETRICAREADIFF-0.25
●●	TIMESTEPCOUNT-0.9
▣▣	VARIANCEFROMCENTROID-0.75
●●	VARIANCEFROMCENTROID-0.9
—	W/o Change Identifiers

Figure 7.16. Legend for plots Fig. 7.14 and Fig. 7.15

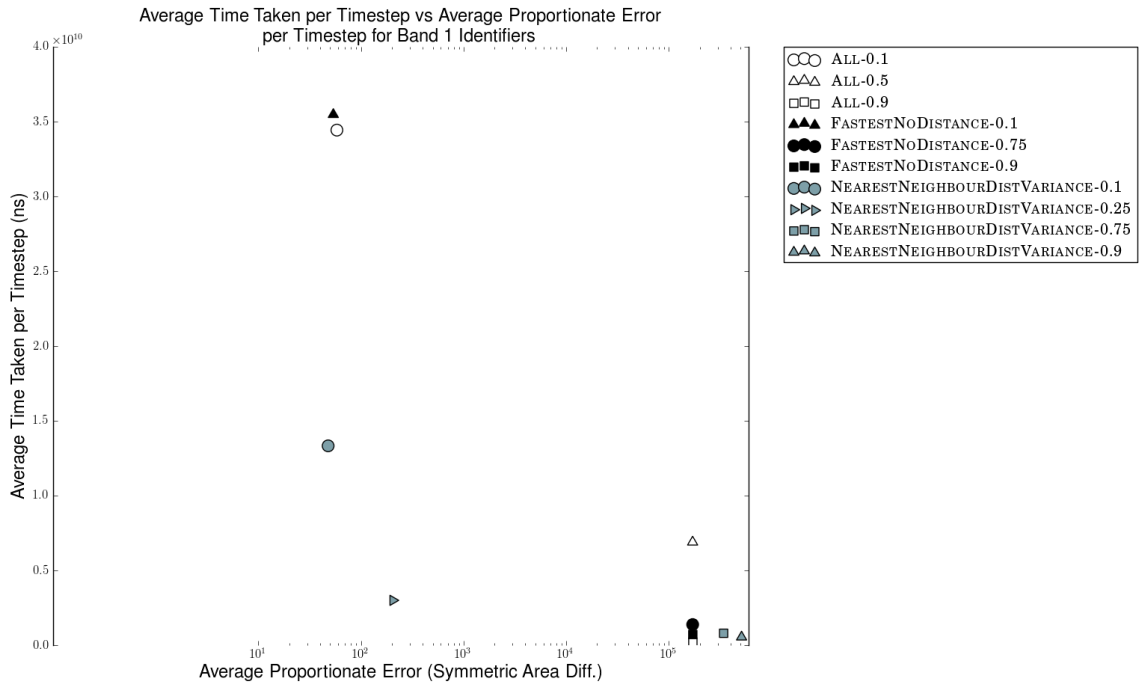


Figure 7.17. Time taken against error per timestep averaged across all dot patterns for the best performing thresholds of the identifiers appearing in Band 1

VARIANCE identifier (estimated nearest neighbour distance variance) has performed, with its lowest threshold value producing the least error score. This does not mean that NEARESTNEIGHBOURDISTVARIANCE is a ‘better’ identifier than any other necessarily but, given the averaging across the twenty-one different types of dynamic dot pattern, suggests that it is a very generally applicable change identifier.

7.5.5. Individual Run: NEARESTNEIGHBOURDISTVARIANCE on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull.

The previous results have all focused on the results averaged across the dot patterns and/or the footprint algorithms for multiple change identifier sets. In this section we will examine the results from runs of an individual identifier set.

Results

Of the identifier sets the NEARESTNEIGHBOURDISTVARIANCE-0.1 performed well across the averaged runs and so we will look primarily at the results of tests for which it was the change identifier set. On figures Fig. 7.8 and Fig. 7.9 the dynamic dot pattern that expanded in extent via change in cardinality with noise presented the greatest variance in time and error so it is the pattern used for the runs shown in Fig. 7.18 and Fig. 7.19. These runs use the χ -hull, as it had the median performance on that particular dynamic dot pattern.

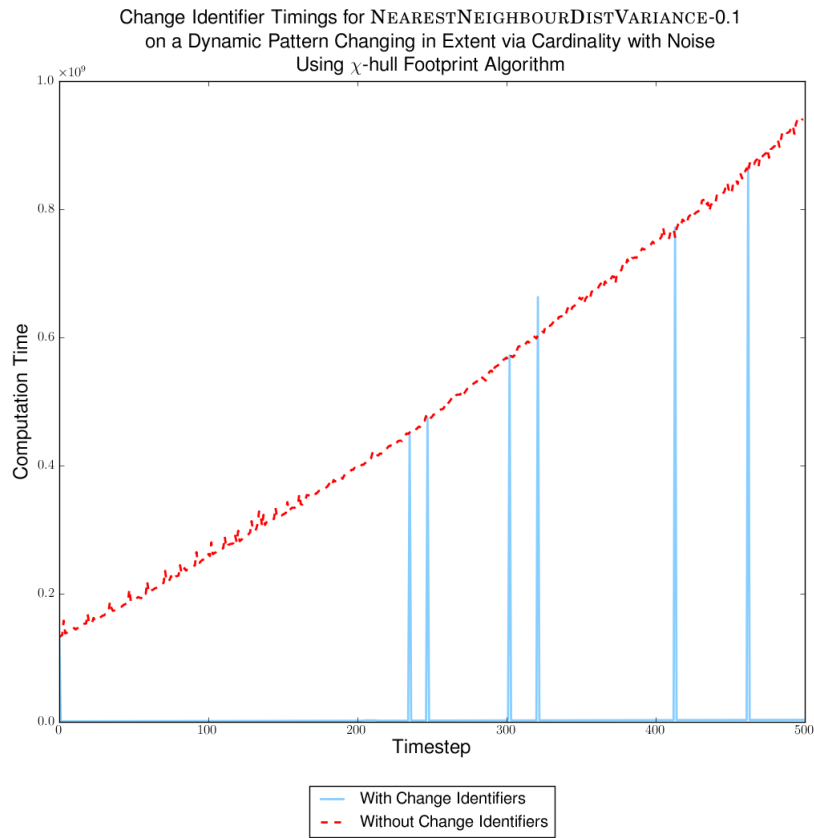


Figure 7.18. Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull footprint algorithm.

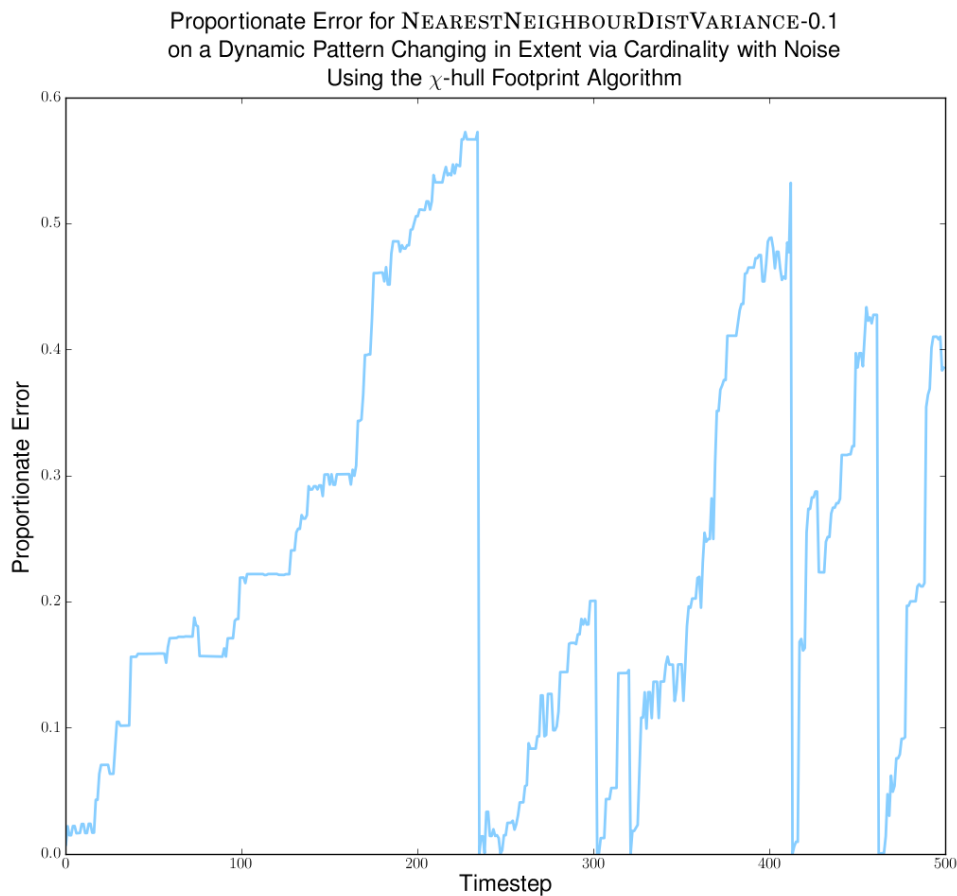


Figure 7.19. Proportionate error at each timestep for NEARESTNEIGHBOURDISTVARIANCE - 0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull footprint algorithm.

Discussion

Fig. 7.18 shows that, even as the computation time for the pattern increases with the cardinality, the change measured by the given set can still be calculated within far less time than the algorithm takes to run. Fig. 7.19 displays the error at each step found by the symmetric area difference between the current and true footprint as a fraction of the true footprint's area. To clarify why the symmetric area difference itself cannot be used as a value for error, we note that a small symmetric area difference corresponds with a similar stored footprint to true footprint, but that this gives no indication about where the distinctions between small and large symmetric area difference values are made. Fig. 7.20 shows an example of this effect; both box pairs A and B have the same symmetric area difference, however the box b_2 could well be considered a better fit to b_1 than a_2 is to a_1 . To be able to draw conclusions about how well the footprint has been maintained the equation for *mismatch* from Chapter 5 is used in which the symmetric area difference at each timestep is taken as a proportion of the true footprint. The maximum proportionate error that is reached in Fig. 7.19 is 0.57 (or 57%) which may seem quite a high value but the mean is less than half that at 0.24. Fig. 7.21(a) shows the same data as Fig. 7.19 but replotted as a histogram; the vertical axis shows the proportion of timesteps from the given dynamic pattern that have error in the range given by each bar. The second histogram (Fig. 7.21(b)) demonstrates that the NEARESTNEIGHBOURDISTVARIANCE-0.1 set provides an error less than 0.24 for just less than 60% of the timesteps of the dynamic dot pattern; this is far more acceptable a score than the maximum error would initially lead us to believe. While the value for an acceptable proportionate symmetric area difference is application specific, this result confirms the initial hypothesis that it is possible to maintain a footprint representation of the dot pattern whilst not updating the footprint at each timestep.

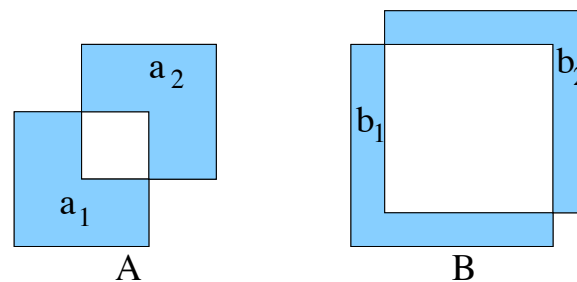
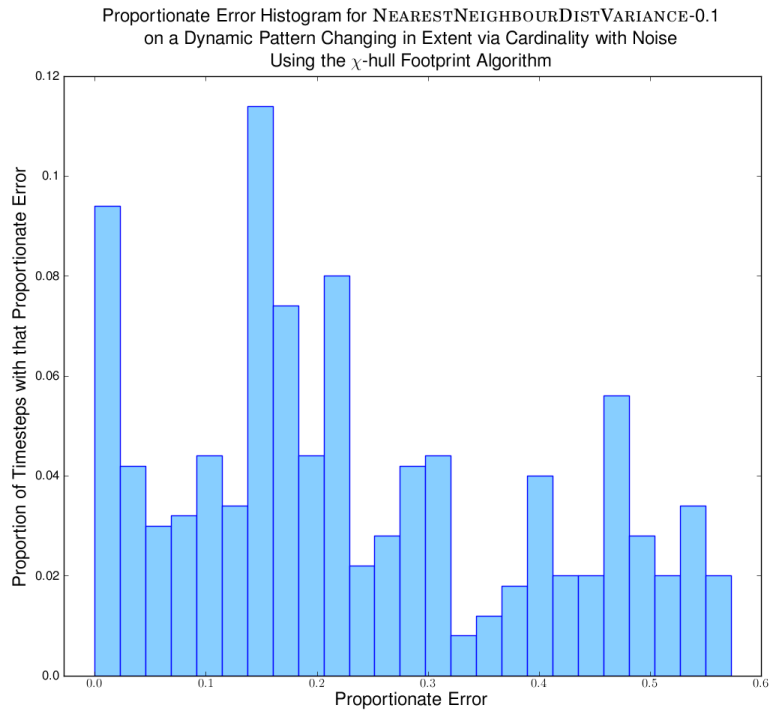
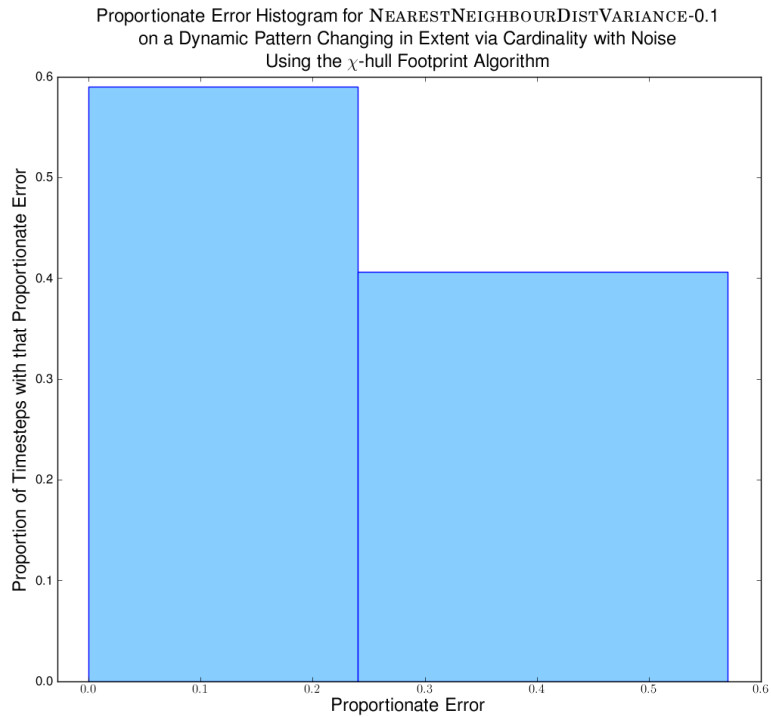


Figure 7.20. Example of the Problem in Using Symmetric Area Difference as an Error Measure



(a) Proportionate error histogram for NEARESTNEIGHBOURDISTVARIANCE – 0.1 on the Extent-Cardinality-Expand-Noise dynamic dot pattern using the χ -hull footprint algorithm.



(b) Proportionate error histogram for NEARESTNEIGHBOURDISTVARIANCE – 0.1 on the Extent-Cardinality-Expand-Noise dynamic dot pattern using the χ -hull footprint algorithm showing the proportion of timesteps below 0.24 proportionate error.

Figure 7.21. Proportionate error histograms for NEARESTNEIGHBOURDISTVARIANCE – 0.1 on the Extent-Cardinality-Expand-Noise dynamic dot pattern

7.5.6. Individual Run: NEARESTNEIGHBOURDISTVARIANCE on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the minimum isothetic bounding box.

Results

To be certain that the choice of the χ -hull is not unfairly weighting the graphs in the change identifier set's favour, tests were run using the same dynamic dot pattern but with the minimum isothetic bounding box as the footprint algorithm (Fig. 7.22 and Fig. 7.23).

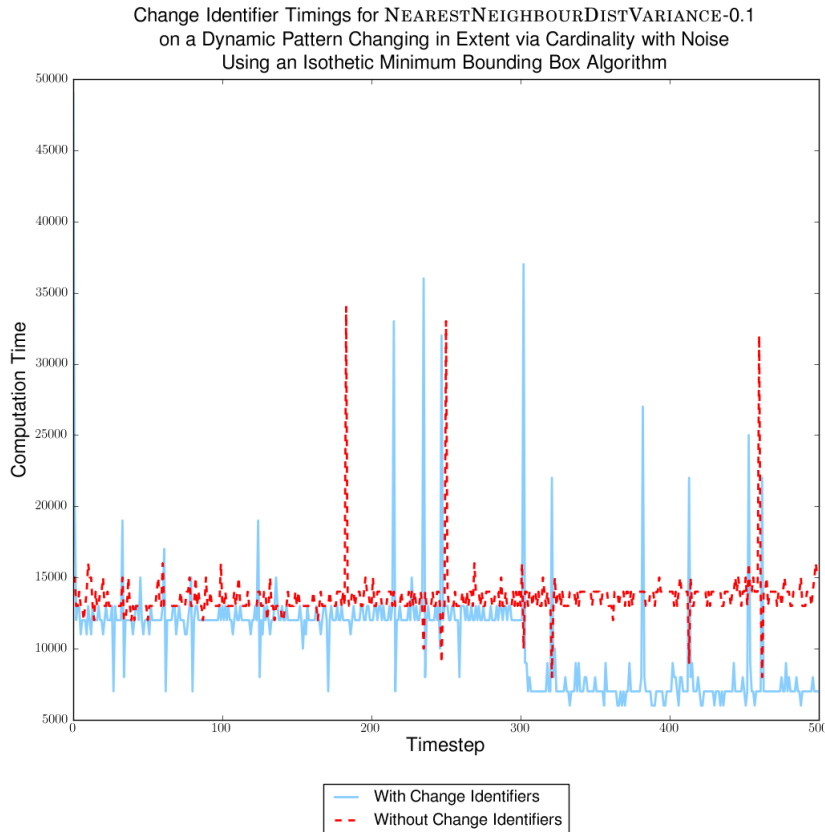


Figure 7.22. Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the isothetic minimum bounding box footprint algorithm.

Discussion

Even with the low complexity of the bounding box algorithm Fig. 7.22 demonstrates that some change identifiers can still operate in less time. The ‘spikes’ for the **without** run on Fig. 7.22 are indicative of the misleading data that can arise when running experiments sensitive to small increments of time on a machine not solely dedicated to this purpose; occasionally the processor will give precedence to other tasks. These anomalous data points are one of the problems avoided by basing our overall conclusions on the average results across multiple runs. Fig. 7.23 matches Fig. 7.19 but is somewhat more ‘blocky’ as would be expected when dealing with symmetric area differences on the minimum bounding box.

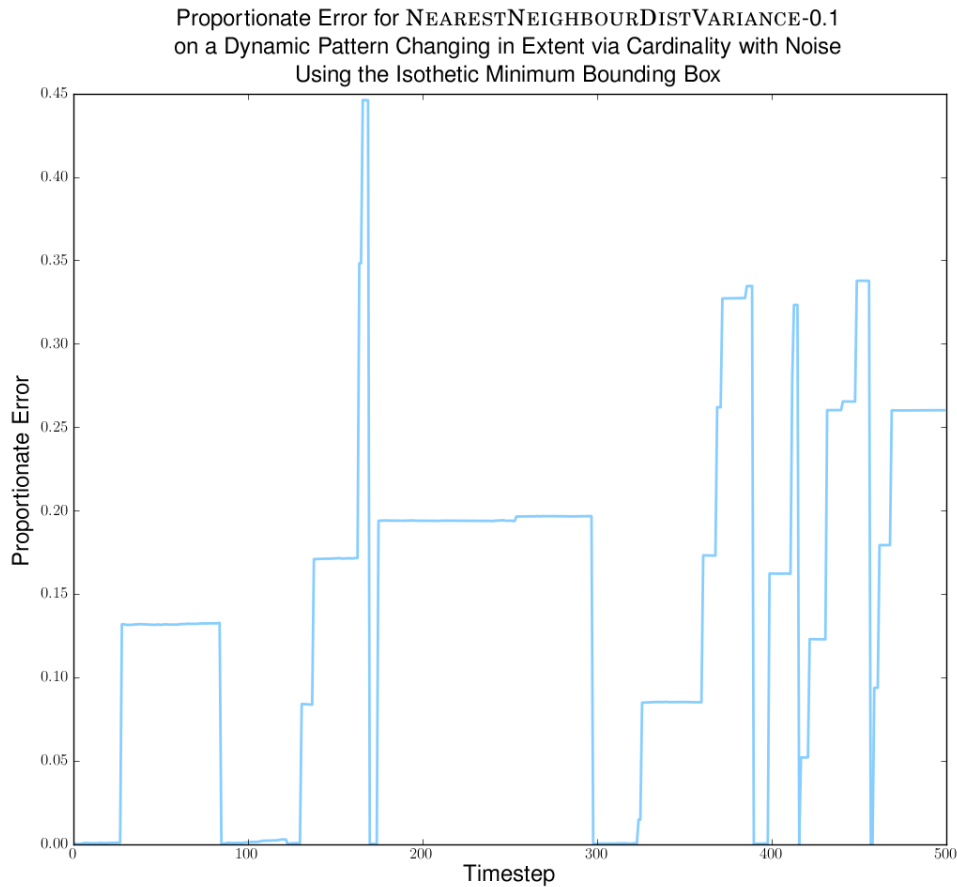


Figure 7.23. Proportionate error at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the isothetic minimum bounding box footprint algorithm.

7.5.7. Individual Run: NEARESTNEIGHBOURDISTVARIANCE on the random dynamic dot pattern using the χ -hull.

Results

The dynamic dot pattern used in the previous tests has a relation between successive phases. While it is noisy, the pattern changes in a consistent fashion and this is evidenced in the steady increase of symmetric area difference shown in Fig. 7.19 and Fig. 7.23. Within the set of dynamic dot patterns used for this thesis there is a random pattern in which no phase need bear any relation to the previous. We would not expect change identifiers to function particularly well for a pattern that requires such regular updates. However, should the dynamic dot pattern have enough brief respites, in which the difference between two phases is not great, it will be possible for change identifiers to save some computational time. Fig. 7.24 shows the time comparison for **with** and **without** runs on the random dynamic dot pattern using the χ -hull footprint algorithm and is, due to the many footprint updates required, decidedly cramped. Fig. 7.25 shows the times taken for the same experiment as the difference between the **without** and **with** runs. Fig. 7.26 shows the proportionate symmetric area when running **with**. The graph has been cropped because the scaling made it difficult to see the timesteps at which updates occurred; the difference between the symmetric area differences ranging between several

orders of magnitude.

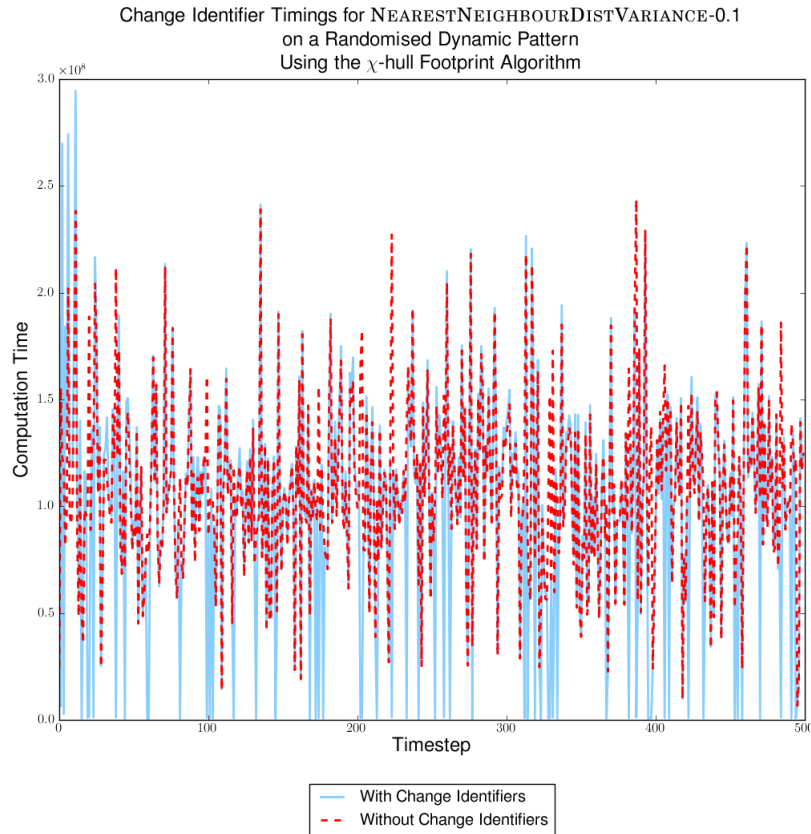


Figure 7.24. Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Random dynamic dot pattern using the χ -hull footprint algorithm.

Discussion

The line on Fig. 7.25 stays mostly above the 0 point on the y -axis, and we can therefore state that, even if it is only by a small amount, the change identifiers are still saving time. With regard to the error Fig. 7.26 shows that the mean proportional symmetric area difference was 0.70 but the maximum was 169; this is far too large an error to be acceptable to most, if not all, applications. However such randomised dynamic dot patterns are unlikely to occur in the real-world and can be mitigated against by using sets with identifiers for each descriptor class, such as the FASTESTNODISTANCE. Fig. 7.27 and Fig. 7.28 show graphs for a test using FASTESTNODISTANCE with thresholds of all its identifiers set at 0.25 to prevent constant updates. The set still manages to have an average computation time less than that of the **without** run: a mean time of around 0.007 seconds per timestep and a total time saved of 3.3 seconds. The proportionate error has a maximum of 5.7 but an average of only 0.13, thus we feel confident stating that, despite the high maximum, the average error is well within the bounds of acceptability for most applications. This adds further weight to our hypothesis that a change identifier set can be used, even on highly variable data, to reduce overall computation time for low values of footprint error.

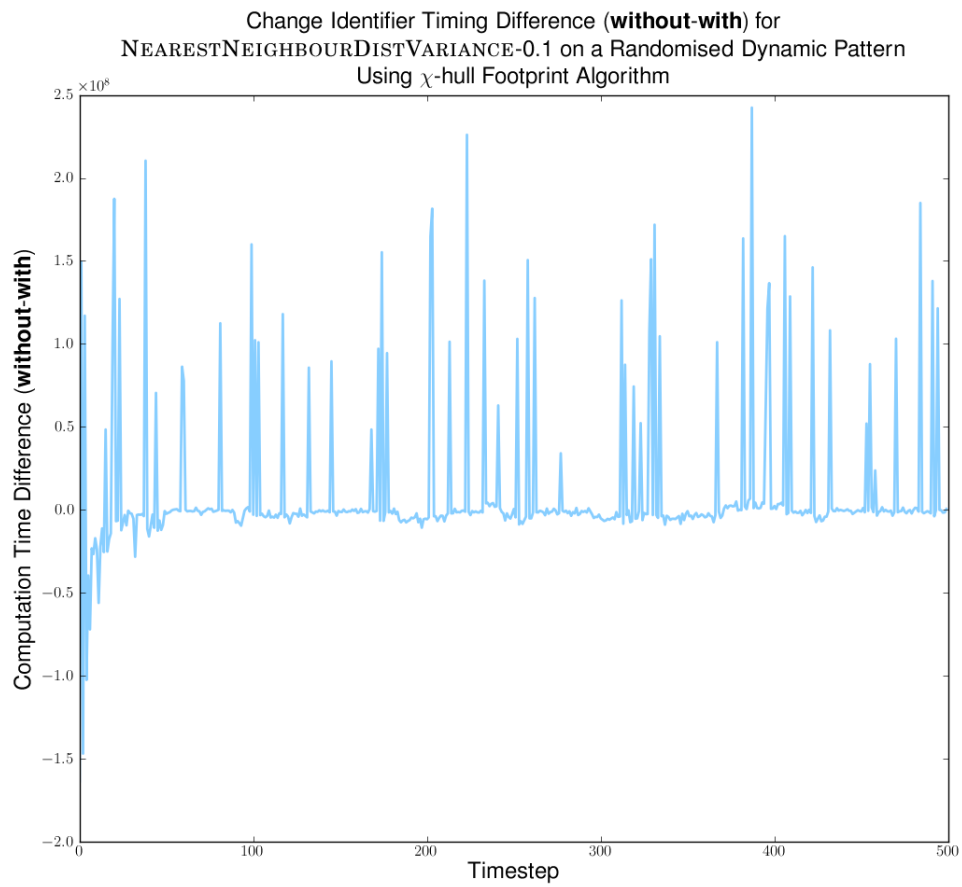


Figure 7.25. Difference between **with** and **without** in time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Random dynamic dot pattern using the χ -hull footprint algorithm.

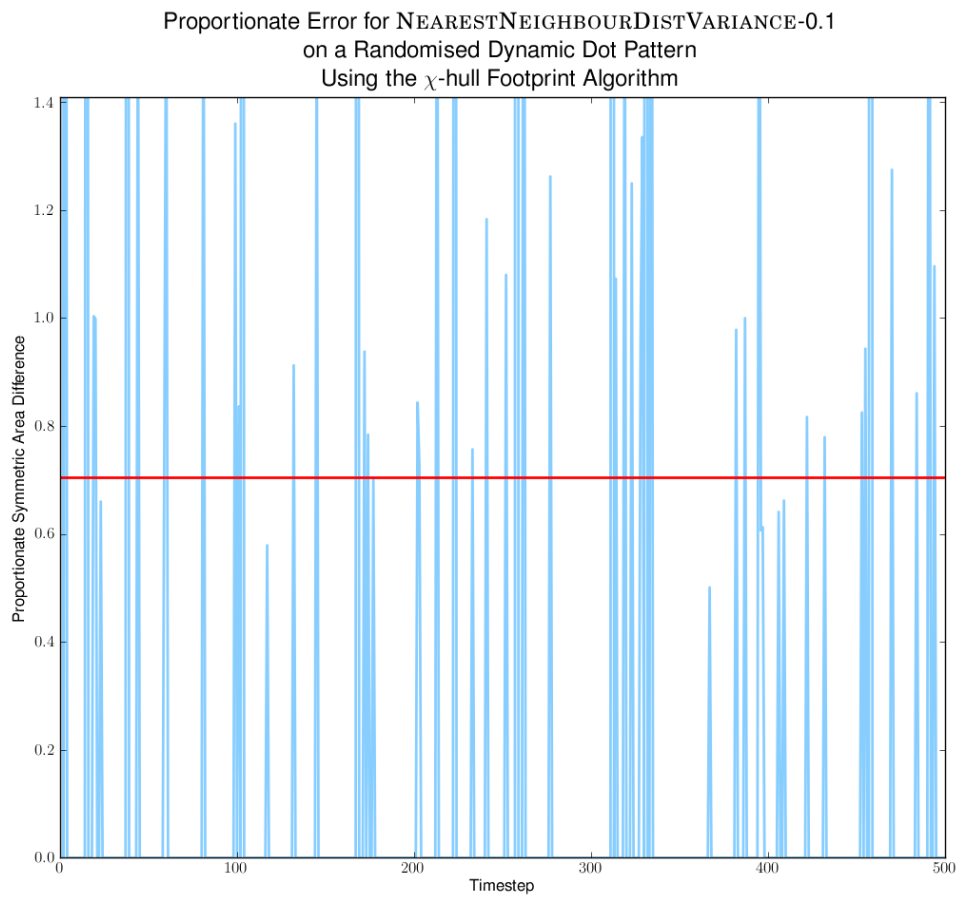


Figure 7.26. Proportionate error at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1 on the Random dynamic dot pattern using the χ -hull footprint algorithm. The red line indicates the mean proportionate error.

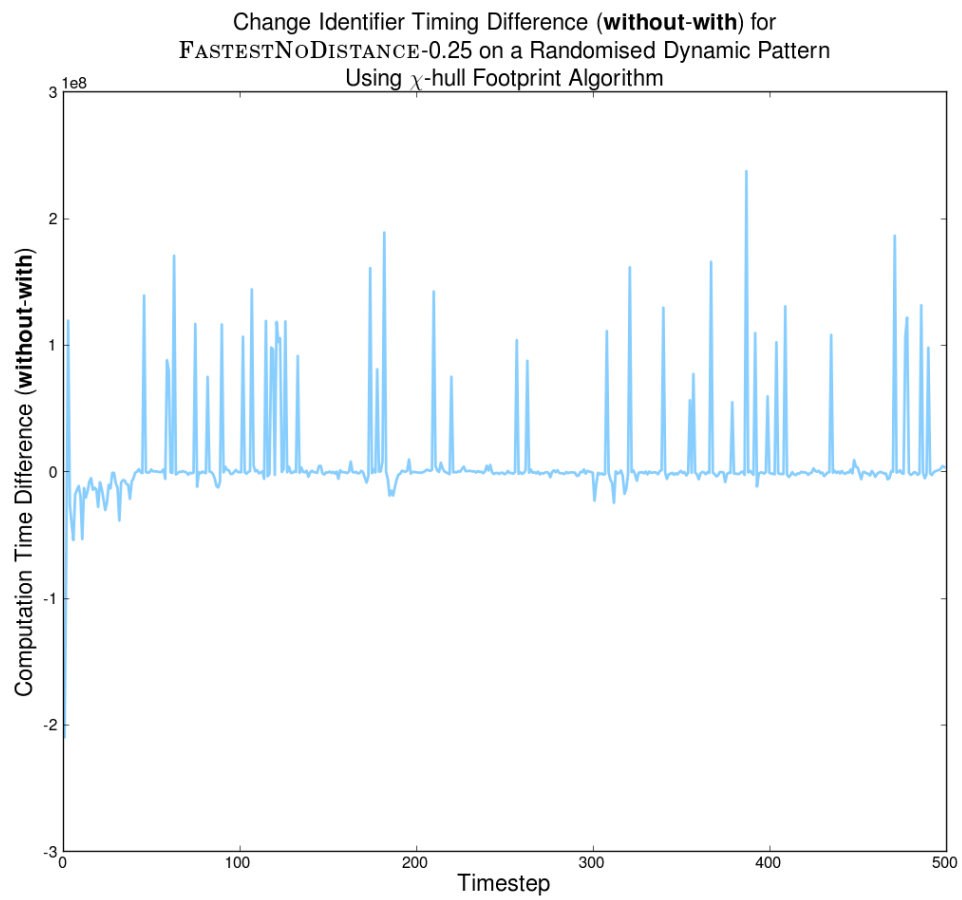


Figure 7.27. Difference between **with** and **without** in time taken to compute at each timestep for FASTESTNODISTANCE-0.25 on the Random dynamic dot pattern using the χ -hull footprint algorithm.

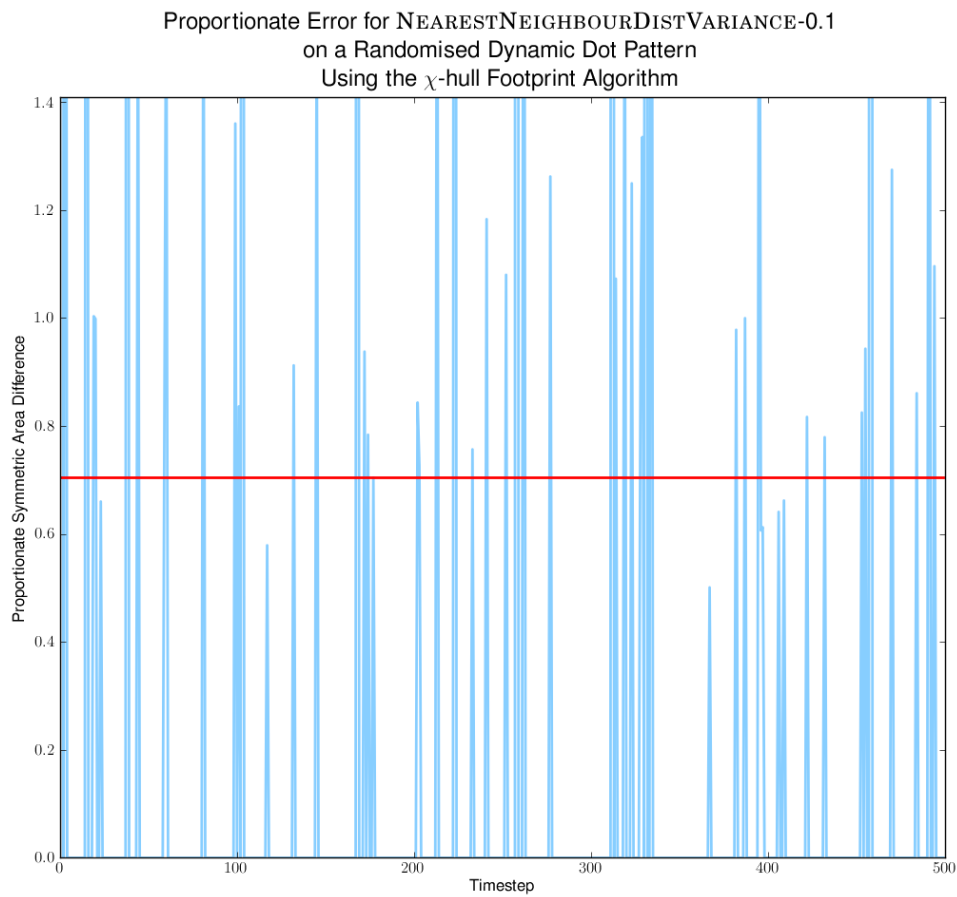


Figure 7.28. Proportionate error at each timestep for FASTESTNODISTANCE-0.25 on the Random dynamic dot pattern using the χ -hull footprint algorithm. The red line indicates the mean proportionate error.

The NEARESTNEIGHBOURDISTVARIANCE-0.1 set's average performance is reasonable¹³ on the extent via cardinality dynamic dot pattern, but by using the knowledge that we have about the nature of a specific dynamic dot pattern and the classes of other change identifiers we can define a set that reduces the symmetric area difference further without increasing the time taken greatly. For the final result of this chapter we define the user selected set USERSELECTED with the specification given in Listing 7.1. The identifier set has two identifiers: a measure of extent (DIAMETERSQ) and distribution (CARDINALITY) and a proportionate max allowed fails of 0.5. The threshold is set at 0.5 so that should any of the identifiers have their thresholds breached the footprint will be updated. The identifiers that have been chosen were selected to match the pattern which increases its extent by increasing the cardinality. The dynamic dot pattern will, therefore, directly affect the change identifier classes of distribution and extent.

```

1 <changeidentifierset name="UserSelected -0.5" ver="0.1">
2   <description>User selected set</description>
3   <maxFails>0.5</maxFails>
4   <concurrent>>false</concurrent>
5   <changeidentifier>
6     <identifier>DiameterSq</identifier>
7     <priority>0</priority>
8     <threshold>0.1</threshold>
9   </changeidentifier>
10  <changeidentifier>
11    <identifier>Cardinality</identifier>
12    <priority>0</priority>
13    <threshold>0.1</threshold>
14  </changeidentifier>
15 </changeidentifierset>

```

Listing 7.1 User Defined Change Identifier Set

7.5.8. Comparison of USERSELECTED and NEARESTNEIGHBOURDISTVARIANCE.

Results

The result of running USERSELECTED over the noisy extent dynamic dot pattern is shown in Fig. 7.29 plotted against two variants of NEARESTNEIGHBOURDISTVARIANCE. The graphs in Fig. 7.30 and Fig. 7.31 show the proportionate error and time taken for each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, NEARESTNEIGHBOURDISTVARIANCE-0.25 and USERSELECTED-0.5.

Discussion

Looking at Fig. 7.29 it can be seen that the USERSELECTED set has clearly increased the accuracy of the footprint tracking for a very small increase in time (3.71×10^{-4} seconds,

¹³Albeit with some, depending on the application, unreasonable maximum errors.

Average Time Taken per Timestep vs Average Proportionate Error per Timestep Showing a User Defined Set on a Dynamic Pattern Changing in Extent via Cardinality with Noise Using the χ -hull Algorithm

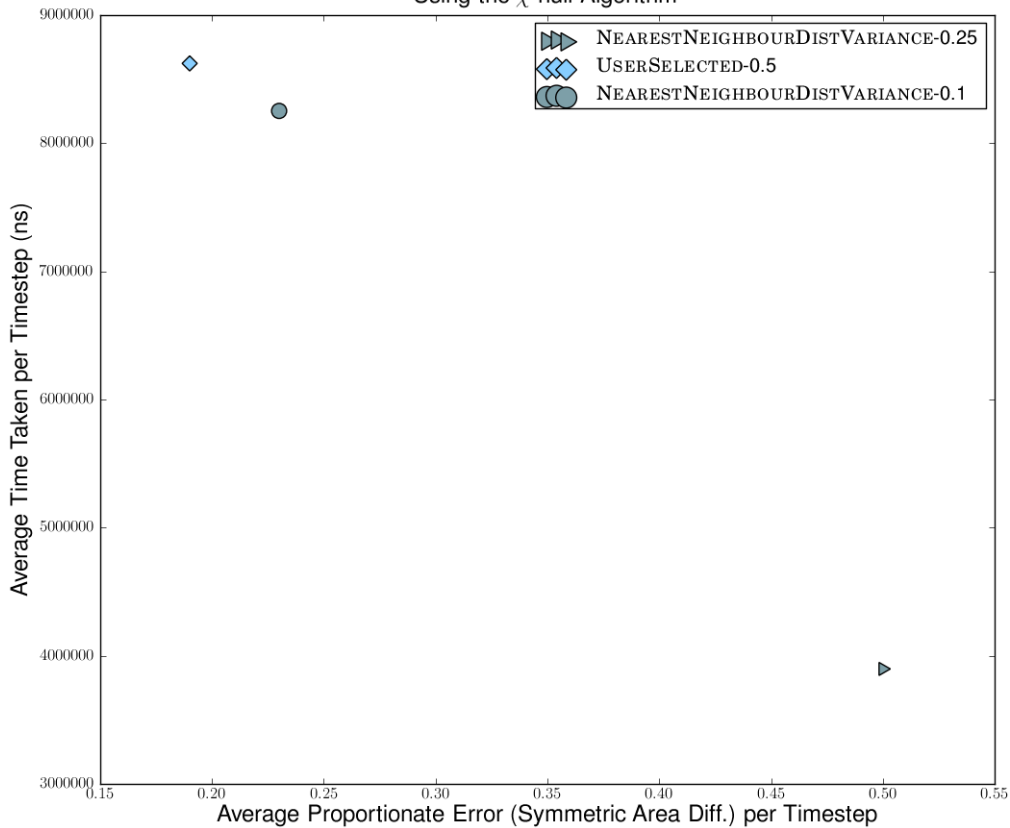


Figure 7.29. Time taken against proportionate error per timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, NEARESTNEIGHBOURDISTVARIANCE-0.25 and USERSELECTED-0.5 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull footprint algorithm.

which is small enough to be considered negligible¹⁴). Fig. 7.30 shows that the USERSELECTED set is well below both of the NEARESTNEIGHBOURDISTVARIANCE sets for nearly half of the timesteps. At timestep 235 the NEARESTNEIGHBOURDISTVARIANCE-0.1 set starts to break its threshold, from this point onwards the proportionate symmetric area difference of the USERSELECTED set and NEARESTNEIGHBOURDISTVARIANCE-0.1 is a lot more balanced. The NEARESTNEIGHBOURDISTVARIANCE-0.1 set appears to update 8 times¹⁵ on Fig. 7.30 and only 7 times on Fig. 7.31 but at timestep 247 the proportionate symmetric area difference is 1.63×10^{-10} , which is so low as to appear to touch the axis on the graph. On Fig. 7.31, around the 300th timestep, we can see that the USERSELECTED set and NEARESTNEIGHBOURDISTVARIANCE-0.1 have similar computation times for similarly sized sets, leading to the conclusion that the USERSELECTED set only has a higher overall computation time because it updates more often. The graphs demonstrate that, with knowledge about the dynamic dot pattern, we can select intuitively sensible change identifier sets that will perform well at reducing computation time for controllable levels of error.

¹⁴Given our previous discussion on the accuracy of measuring small time increments on a computer not dedicated to the purpose of running the change identifier tests.

¹⁵Recall that there is always an update on the initial timestep

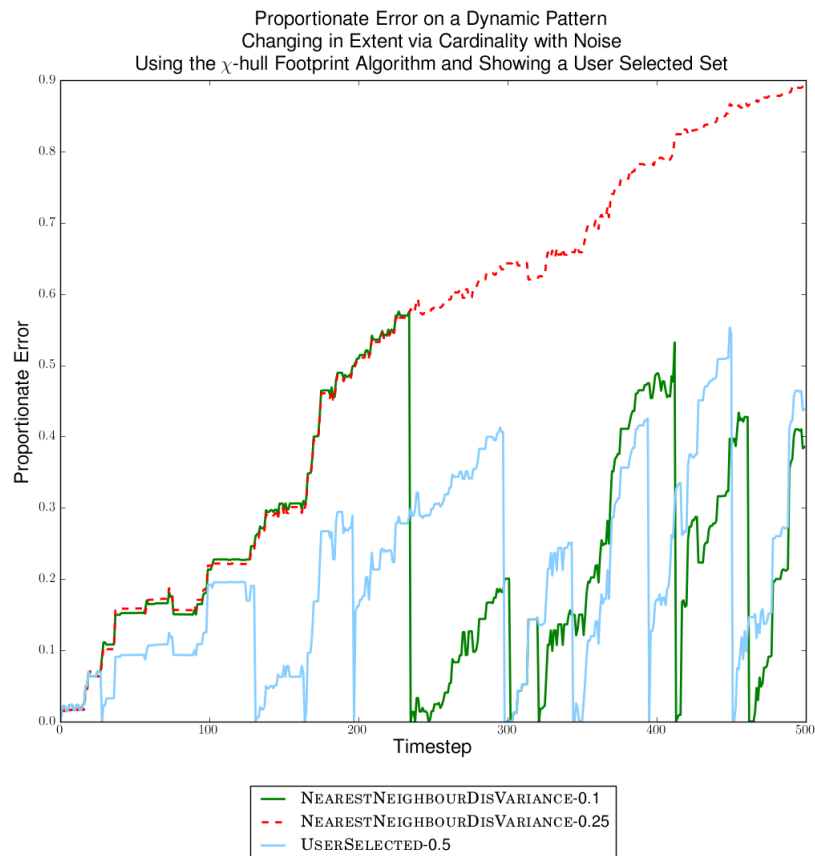


Figure 7.30. Proportionate error at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, NEARESTNEIGHBOURDISTVARIANCE-0.25 and USERSELECTED-0.5 on the Extent-Cardinality-Expand-Noisy dynamic dot pattern using the χ -hull footprint algorithm.

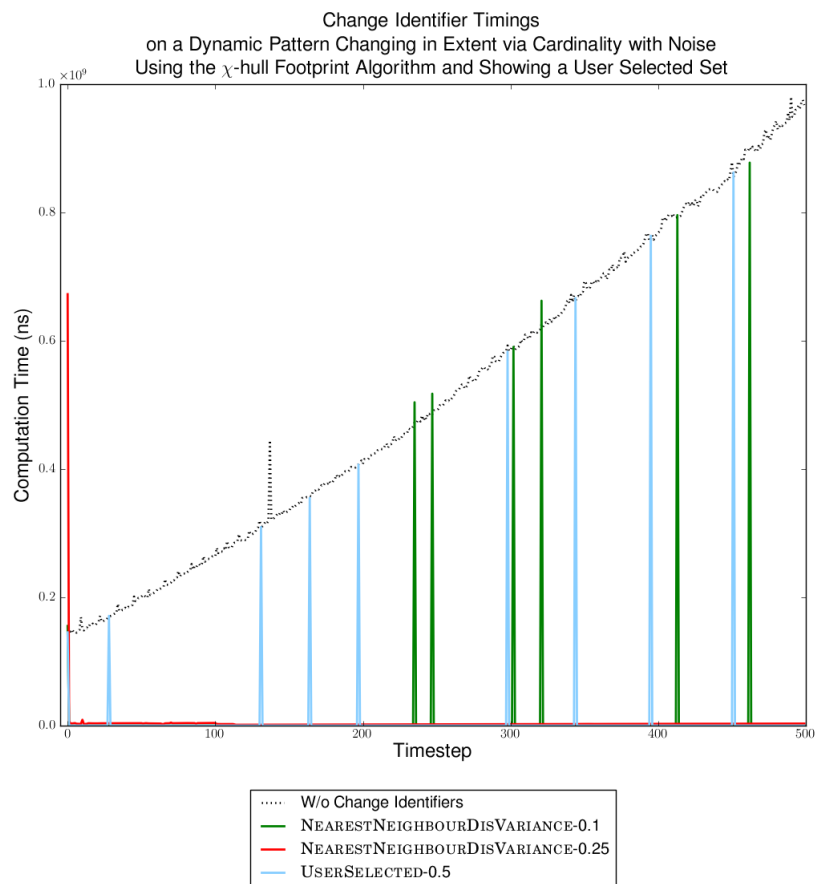


Figure 7.31. Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, NEARESTNEIGHBOURDISTVARIANCE-0.25 and USERSELECTED-0.5 on the Random dynamic dot pattern using the χ -hull footprint algorithm.

7.6. Summary

This chapter has shown that the change identifiers can decrease the number of updates needed to maintain a suitable footprint over a dynamic dot pattern. The graphs presented have demonstrated the reduction in time taken needed for computation per timestep, and they have also provided evidence that the error score for the stored footprint, when measured for similarity against the true footprint, is quantifiably low. While there are many more graphs that could be added to this chapter, for example those that show individual runs for some of the other dynamic dot patterns and change identifiers, we are conscious that the current graphs already extend beyond the text and that it is becoming difficult to follow them in accordance with their discussions. Further to clarify considerations, more graphs will not show new aspects of the analysis but simply reinforce the points already made. Additional graphs of results from tests using the `USERSELECTED` set created in this chapter and some other sets, including a set created for Boid-like dynamic patterns, can be found in the appendices.

The chapter has also indicated that some change identifier sets perform measurably better than others and that, with some thought, it is possible to define change identifier sets that perform well for given dynamic dot patterns. The sets found by the original limited enumerative approach (all singleton sets with multiple thresholds) are, at best, adequate and don't compare favourably to those created by user selection.

8. Summary and Conclusions

8.1. Thesis Summary

The goal of this thesis has been to show that the use of change identifiers will reduce the time taken to maintain a footprint over a dynamic dot pattern while introducing an acceptable level of error.

The inquiry began with an investigation of the dot patterns. Our goal was, not merely to provide background to the change identifiers, but to see if measurements on the dot patterns could provide useful information in its own right. This exploration of the dot patterns led to the identification and analysis of the dot pattern descriptors. There is a wealth of information present in the individual patterns and the descriptors are partial measures of this information. It was found that, not only, could the descriptors provide a stable base for the change identifiers but that they may be able give a classification structure for the dot patterns. A preliminary examination of how this classification might be constructed is demonstrated in § 8.3.

Footprints have a large scope of operation, appearing in different forms across a range of fields. Before looking at the change identifiers this thesis devoted a chapter (Chapter 4) to the investigation of the types of footprint that are commonly produced by the footprint algorithms in the literature and proposed a classification (as an extension of the work performed in [Dupenois and Galton, 2009]) based on this investigation. The chapter also looked at how the footprint type may be affected as it is updated over a dynamic dot pattern; further discussion on which can be found within the future work chapter.

Having discussed the underlying aspects of the proposed problem the thesis turned its attention to the change identifiers. Chapter 5 presented the change identifiers used within this thesis and proposed a method by which to combine them into sets measuring multiple different types of change. It also introduced an assessment approach based on comparing the stored footprint the change identifier set presented at any particular dot pattern phase with the ‘true’ footprint, i.e. the footprint that would have been created had the algorithm been run for the phase.

Chapter 6 gave a description of the more practical issues faced when planning the change identifier experiments. The chapter paid particular attention to the measure of footprint similarity that would be used to ascertain the error of a run with change identifiers. While a conclusion was reached to use symmetric area difference there is scope for future work involving other footprint similarity measures, possibly arising from considering other ways

of defining the requirements on a footprint.

The results of the investigation were shown in Chapter 7, which also detailed the range of dynamic dot patterns and algorithms that were used to provide a fair appraisal of the worth of change identifier sets. Chapter 7 showed that it was possible to reduce (greatly in some instances) the number of footprint updates while maintaining a symmetric area difference that was low proportional to the area of the ‘true’ footprint. The chapter also demonstrated that some identifier sets can out-perform others for specific patterns and that it is possible to create sets to do so using knowledge about the dynamic dot pattern’s nature. Finally the results indicated that some identifiers may be generally more applicable than others.

Choosing the appropriate set for any given application is the main difficulty that may arise when using them. For some applications it may be easy to know in advance how change is most likely to occur and therefore which identifiers to use, however there are some applications in which the change can occur in multiple and unpredictable ways. In such erratic cases a set of identifiers is needed that can identify a mixed range of change types while still taking less time to compute than the footprint algorithm the application is using.

8.2. Conclusions

This thesis provides an investigation into a novel approach for tracking a footprint across spatio-temporal data. The evidence it presents demonstrates that when using change identifiers to indicate the appropriate times at which to update the footprint the average computation time taken per timestep can be greatly reduced for small increases in footprint error. Given the already vague notion of a footprint, this error can be rationalised as allowable approximation of the region that contains the dot pattern at any given timestep. The relationship between the time taken and error has been shown to follow a Pareto-curve on which decreases in error tend to lead to an increase in time taken and vice versa. The traversal along this curve is controllable by the altering of a proportionate threshold on the identifiers. Selection of identifiers appropriate to a context and their thresholds have been made intuitive by the use of considering each identifier as a measure of a specific type of change and their threshold as a percentage of allowed change in the aspect that they measure. We conclude that the change identifiers are a useful approach to the examination of dynamic dot patterns and that they have scope for use beyond that presented here.

Further to the use of change identifiers as a way of reducing the number of updates other uses they might have are explored. This exploration leads to the conclusion that the information supplied by the change identifiers could well be useful in its own right, perhaps even by-passing the need to produce a footprint in many cases (for example when it only needs to be known if the extent is increasing). This will be expanded upon in § 8.3.

8.3. Future Work

The examination presented in this thesis has focused primarily on the ability of change identifiers to reduce the number of required updates whilst tracking the footprint across a dynamic dot pattern. However it has also touched, albeit lightly, on other areas of research (mostly still within the field of spatio-temporal entities) in which they may provide benefit. This section is a look at these unexplored change identifier attributes with some preliminary thoughts and observations.

8.3.1. Dot Patterns

Dot Pattern Types

The descriptors presented in Chapter 3 provide a way of distinguishing different dot patterns, and it may be possible to use these differences to delineate between classes of dot patterns. Should there be intuitively distinct, and plausibly useful, classes that can be used to provide this sort of dot pattern taxonomy then the identifiers can be used to notify the user when a pattern switches from one type to another. The immediate difficulty faced by any taxonomy is in avoiding entirely arbitrary delineations between the pattern types, so to provide a defensible set of distinctions the values of the descriptors are conceptualised as vectors denoting a point in the ‘descriptor space’. The Euclidean distance between these points provides a similarity measure and, with a sample set of randomised patterns, a clustering method (such as the agglomerative clustering method used as a descriptor) can be used to sort them into progressively larger clusters of similar patterns. By looking at the dendrogram¹ of this clustering a ‘cut’ can be made at the various levels allowing different clusterings of varying granularity to be selected.

Before any form of clustering can take place the descriptor values need to be normalised; without this step one descriptor can contribute more to the distance than another. A simple normalisation approach can be performed by taking the maximum and minimum values for a descriptor across the set of randomised patterns and scaling these to a range of -1 to 1 . Descriptors that can result in values of infinity (gradient of principal component for example) are normalised across a sigmoidal function curve as shown in Fig. 8.1.

Should the clustering analysis provide a set of clear descriptor divisions then the next task will be to decide how best to classify a new pattern without having to re-run the clustering process. A possible approach is to settle on some ‘archetypal patterns’ for each of the delineations. The values of each archetype provide a central point in the descriptor space for their respective types. These points can be used to form a Voronoi division of the space and a pattern is therefore of the type whose archetype it is closest to. Such an approach allows a pattern from outside the test set of patterns used for clustering to be classified (the *training set*), but it requires that the descriptor values are normalised. The normalisation currently proposed relies on some maximum and minimum values for the

¹A graph with a tree-like structure showing the concatenation of clusters in the order they appear.

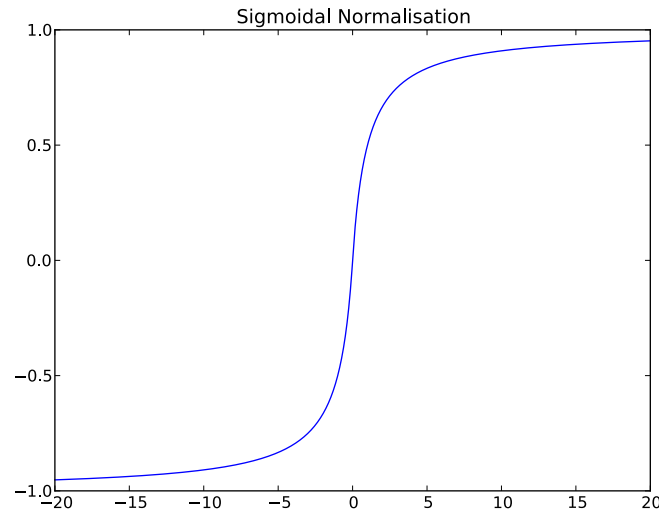


Figure 8.1. Graph showing Sigmoidal Normalisation Curve $\frac{x}{1+|x|}$.

descriptors (except those normalised on the sigmoidal curve), if the pattern to be classified (the *candidate* pattern) has values beyond the range of the maxima and minima of the training set then the clustering analysis will need to be re-run. To provide a classification that never requires the analysis to be re-run requires a training set that contains all possible extremal values for a descriptor; in effect every value would need to be normalised on an asymptotic curve like the sigmoidal approach discussed above. Even if the candidate's values are within the range of the training set, the classification of the candidate pattern is only relative to the training set. A classification that can delineate between the different types of dot pattern presented when tracking a herd's movement may be unsuitable for the set of dot patterns presented by the buildings within cities. A better approach would have sample patterns from the field in which we wish to classify the dot patterns and find the archetypes specific to that field.

Using an agglomerative clustering method we have clustered a randomised dynamic dot pattern of length 15 by the sigmoid normalised values of the fastest non-correlated descriptor set (given in Chapter 3) to give some preliminary indications as to whether or not such 'archetypal patterns' could be found. Only 15 patterns have been used as the graphs and associated image displays of larger sets are not particularly intelligible in a printed document.

The dendrogram in Fig. 8.2 shows a few large 'jumps' towards the end as the distance between the pattern clusters increases. At the cut off point indicated on the figure there are 3 clusters $\langle 5, 6 \rangle$, $\langle 14, 1, 7 \rangle$ and $\langle 0, 2 \rightarrow 4, 8 \rightarrow 13 \rangle$, this cut-off was chosen as both 'legs' of the branch joining $\langle 14, 1, 7 \rangle$ to $\langle 0, 2 \rightarrow 4, 8 \rightarrow 13 \rangle$ are of a significant length compared to the average 'jump'. The patterns defined by this division of the clustering can be seen in Fig. 8.3.

The class divisions at this clustering level are not necessarily those that a human would choose, but some of the reasons why the delineations have been made can be rationalised. For example, in the first cluster, [5] and [6] are both mostly collinear patterns with similar

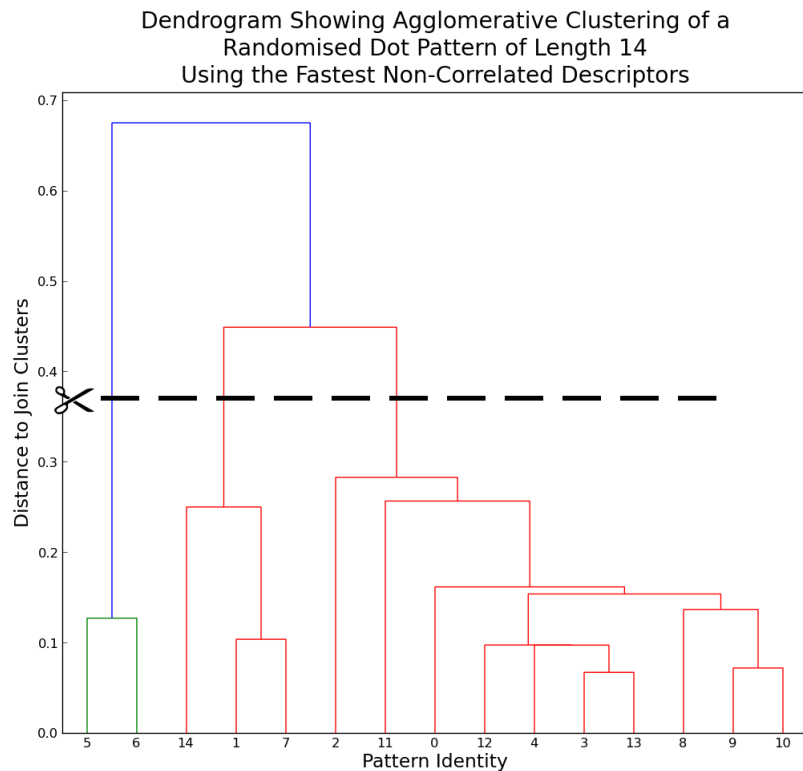


Figure 8.2. Dendrogram for the clustering of dot patterns by the fastest non-correlated descriptor set

orientation, while in the second cluster [7], [14] and [1] also have similar orientation but in the opposite direction to cluster one. However not all the clusterings are as easily understood, for example why are [8] and [2] not connected until the join just before the cut-off shown in Fig. 8.2 despite both being very dense and comparatively small patterns? There are several situations that these non-intuitive clusterings may arise from:

- While the descriptors do not directly correlate, there may be tripartite correlation (as described in Chapter 3 and by Andrienko and Andrienko [Andrienko and Andrienko, 2007]) which unfairly weights some aspects of the clustering.
- The range of dot patterns used in the clustering is not wide enough and some descriptors are represented in a ‘stronger’ form than others. If the differences in extent are small compared to the differences in orientation then the orientation will have a larger weighting in the classification.
- The differences in the parameters that are not visually obvious outweigh the others.

Much more research needs to be performed before such a classification can be used, in particular the question of how the classification is assessed would need to be answered. For example, would the classification be better if the ‘archetypal’ dot pattern types/classes are those that are intuitive to a human? Answering such questions would require an in-depth analysis of the types of information required in applications using dot patterns and, given the range of fields in which such patterns appear, will likely have context specific answers.

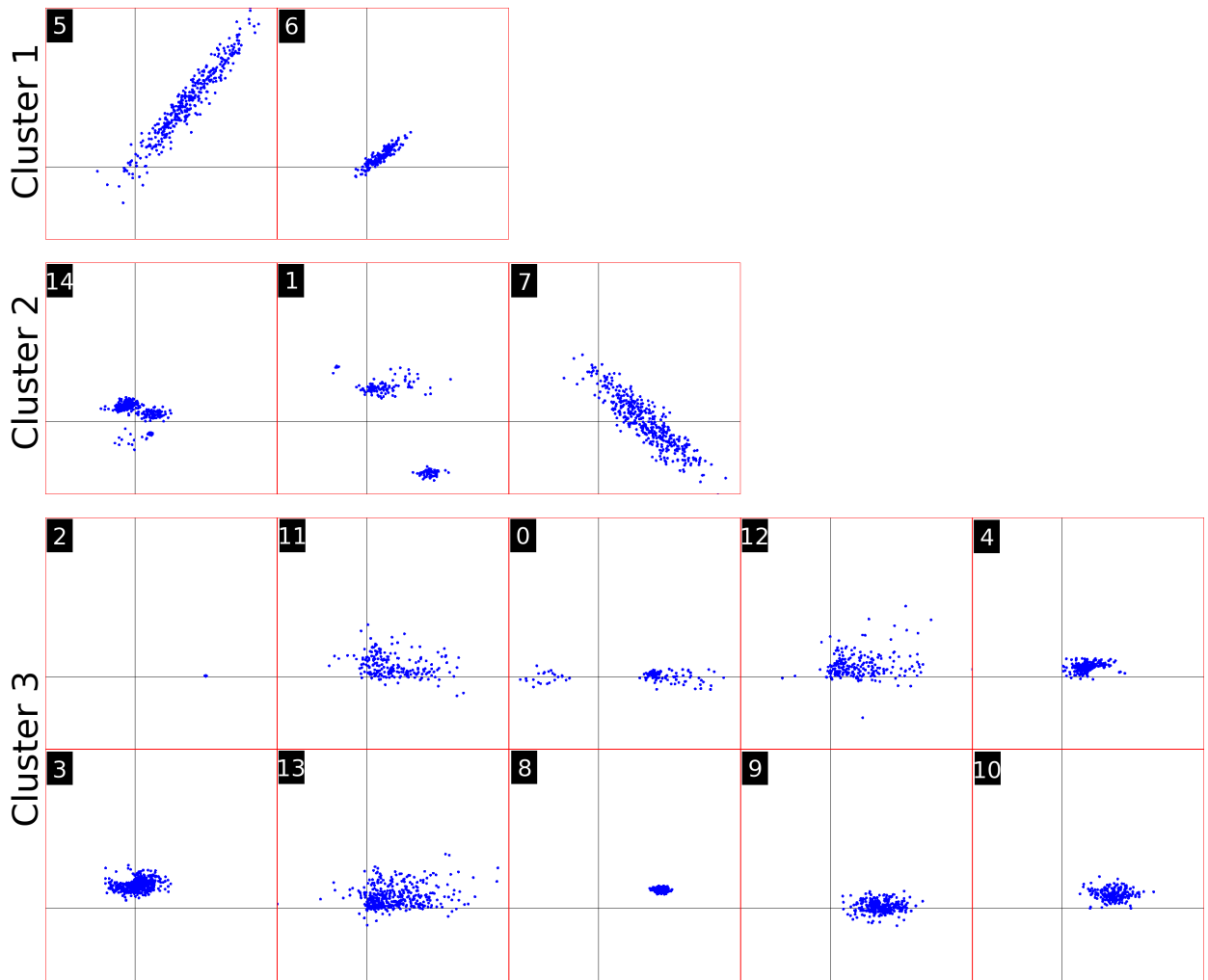


Figure 8.3. Snapshot of the clustering process of dot patterns by the fastest non-correlated descriptor set.

Should definitive classes arise from the agglomerative clustering process it will then be possible to find a best fit class for any given dot pattern. The dynamic dot pattern can be seen as a traversal across the ‘class’ space of dot pattern types. While we may expect a specific dynamic dot pattern to not move too far from its initial class it is certainly possible that it will migrate gradually towards other classes. For example a herd of wildebeest² being chased by a predator may go from a single spread out grouping to several dense fleeing packs.

Plotting a Dynamic Dot Pattern

A further abstraction on the descriptor concept is to imagine the dot patterns as existing in a multi-dimensional ‘descriptor space’, with each descriptor providing a coordinate axis. When mapped onto this space each dot pattern forms a single dot, with each of its descriptor values specifying a position on the appropriate axis. Plotting all of the phases of a single dynamic dot pattern produces a new dot pattern, which represents the dynamic dot pattern by the range of descriptor values it has undergone. The footprint of this

²One of three possible plurals for wildebeest: wildebeest, wildebeests and wildebai

resultant dot pattern describes the bounds of the dynamic dot pattern with respect to the descriptors and is, therefore, representative of the limits of the change of the underlying collective³. If time is added to the axes of the descriptor space then, instead of producing a footprint, the dynamic dot pattern traces a path through its descriptor values. Using the descriptor space to produce a footprint or a path for a dynamic dot pattern may provide a new and interesting approach to classifying the collectives by their behaviour.

8.3.2. Footprints

Footprint Types

We can examine the footprint at each time step at which it is updated to ascertain its intrinsic type within the classification presented in Chapter 4. However, unlike the dot pattern types suggested above, the calculations are outside the scope of the change identifiers – which only examine the dot patterns. Instead the calculations must be added as a new module to the framework; increasing the time taken for computation at each time step that the footprint is updated. Preferably this increase in time will not void the gain made in using change identifiers. The state of all of the intrinsic footprint criteria can be found by iterating over the edges of the footprint, and are therefore of at least time complexity $O(v)$ in which v is the number of vertices of the footprint. This increase in calculation cost may be allowable by imagining that the time saved by the change identifiers can be ‘spent’ on other tasks, as long as the system as a whole can still provide a more up-to-date footprint than attempting to update at each timestep. An alternative way in which to consider the cost-benefit trade-off is that the time lost while performing the footprint classification can be allowed if the average calculation time per timestep when using identifiers and footprint classification is less than the average calculation time per timestep when updating the footprint at each phase.

Being able to indicate to a user when the footprint type changes is of benefit for the same reason that the dot pattern type change notification would be; it provides further information about the nature of the change that the collective is undergoing. In addition to the footprint state, we may also be able to use this information to indicate when the footprint algorithms parameter is no longer suitable.

8.3.3. Footprint Algorithm Parameters

In the background chapter, it was noted that most footprint algorithms need a user defined parameter. The parameter requirement led to the discussion within the results chapter of the reasoning behind the algorithm parameter selection used in our experimentation, which aimed primarily to provide fair testing conditions. There is still much work to be done on the informed selection of the footprint algorithm parameters; the work presented in this thesis on this area being very much inchoate. The use of the footprint classification

³Assuming we have descriptors that accurately capture the state of the collective.

may well help with identifying when the parameter needs to change but does not provide a starting value. Using the footprint classification to select parameters is further hamstrung by the, possibly erroneous, assumptions that the parameter needs to be changed when the footprint type changes and that this is the only point at which it need change.

To attain a better understanding of the parameter and footprint relation will likely require an in-depth examination of the interplay between the dot pattern types and the footprint algorithms. The footprint algorithm classification (Chapter 4) will need to be extended to include a description of the nature of the parameter. For example if the algorithm uses the parameter as a threshold on the edge length (Swinging Arm [Galton and Duckham, 2006], χ -hull [Duckham et al., 2008]) it will require a different starting parameter than that of an algorithm which requires the number of neighbours to compare at each iteration (K-Nearest Neighbours [Moreira and Santos, 2007]).

8.3.4. Change Identifiers

There is further research that can be performed on the information provided by identifiers beyond using them to update footprints. The identifiers provide immediate information about the fashion in which the dynamic dot pattern is changing, and this information leads to a description of the complex behaviours the underlying collective phenomenon of the pattern is undergoing. The changes themselves tend to be small and are increments or decrements in quantitative values but this thesis has previously suggested that it may be possible to provide qualitative information directly rather than having a human observer interpret the results. It may be that, for some applications, providing information about the expansion of a dynamic dot pattern via a status update (e.g. “The phenomenon is expanding by 10% every 20 seconds”) is more useful than showing the expansion of a footprint. Change identifiers can be used to bypass the footprint entirely by relaying only the useful change information; reducing the amount of assessment needed to be performed by the user.

It should also be noted that the change identifiers presented in this thesis do not exhaust the range of possible identifiers, and there are almost certainly other useful measurements to be added to the set we provide. They could also be further examined by looking for three-way correlation of they type suggested by Andrienko and Andrienko in their structure consideration [Andrienko and Andrienko, 2007] and to see if the difference between the first and second order effects proposed by O’Sullivan and Unwin can be identified [O’Sullivan and Unwin, 2002].

The conclusions given within this thesis are given with confidence borne from the range of tests that were performed. However future work using change identifiers could certainly include testing over more real world data, ideally this would involve an industrial case study.

8.3.5. Other Fields

The work in this thesis has primarily focused on examples that are spatio-temporal in nature; situating them firmly in the area of GISc. However, as mentioned in the introduction, there are other fields which use data that can be visualised as dynamic dot patterns, for example the movement through the solution space of an optimisation problem or a set of changing entities in some classification space. It would be interesting to see if the application of both footprints and change identifiers to these fields provides new insights into their behaviours.

Appendices

A. Extra Result Graphs

Average Time Taken per Timestep vs Average Proportionate Error per Timestep Showing a User Defined Set on a Dynamic Pattern With Boid-like Behaviour Using the χ -hull Algorithm

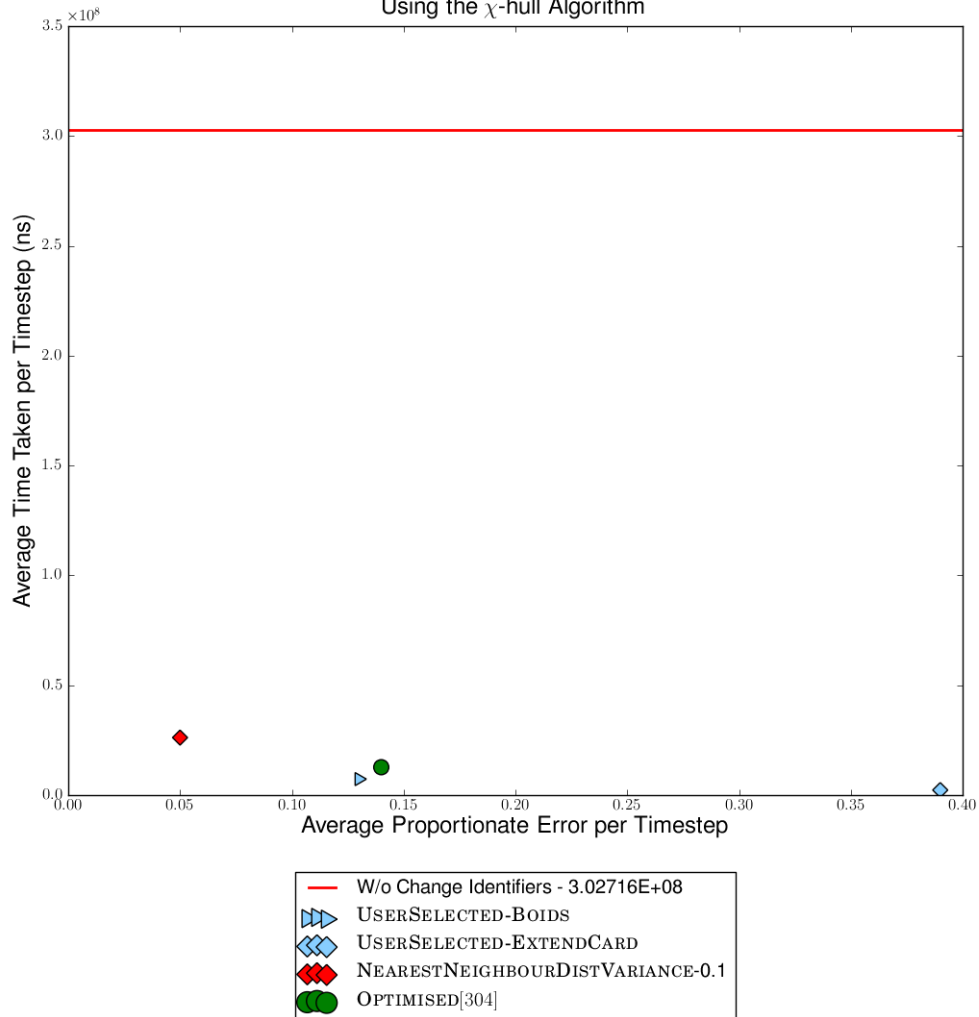


Figure A.1. Time taken against error per timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, OPTIMISED[304], USERSELECTED-BOIDS and the user selected change identifier set created for the Extent-Cardinality-Expand-Noisy dynamic dot pattern, now called USERSELECTED-EXTENDCARD using the χ -hull footprint algorithm on the boid like dynamic dot pattern.

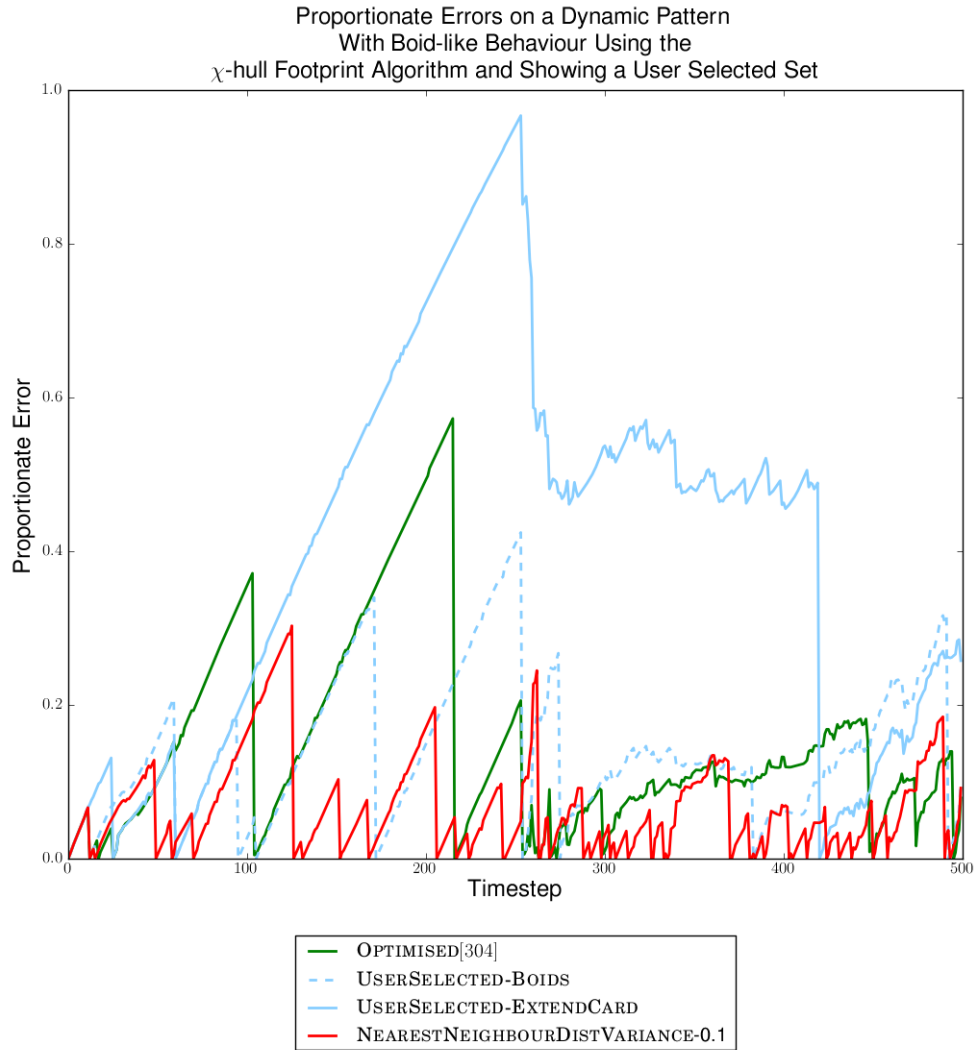


Figure A.2. Proportionate symmetric area difference at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, OPTIMISED[304], USERSELECTED-BOIDS and the user selected change identifier set created for the Extent-Cardinality-Expand-Noisy dynamic dot pattern, now called USERSELECTED-EXTENTCARD using the χ -hull footprint algorithm on the boid like dynamic dot pattern.

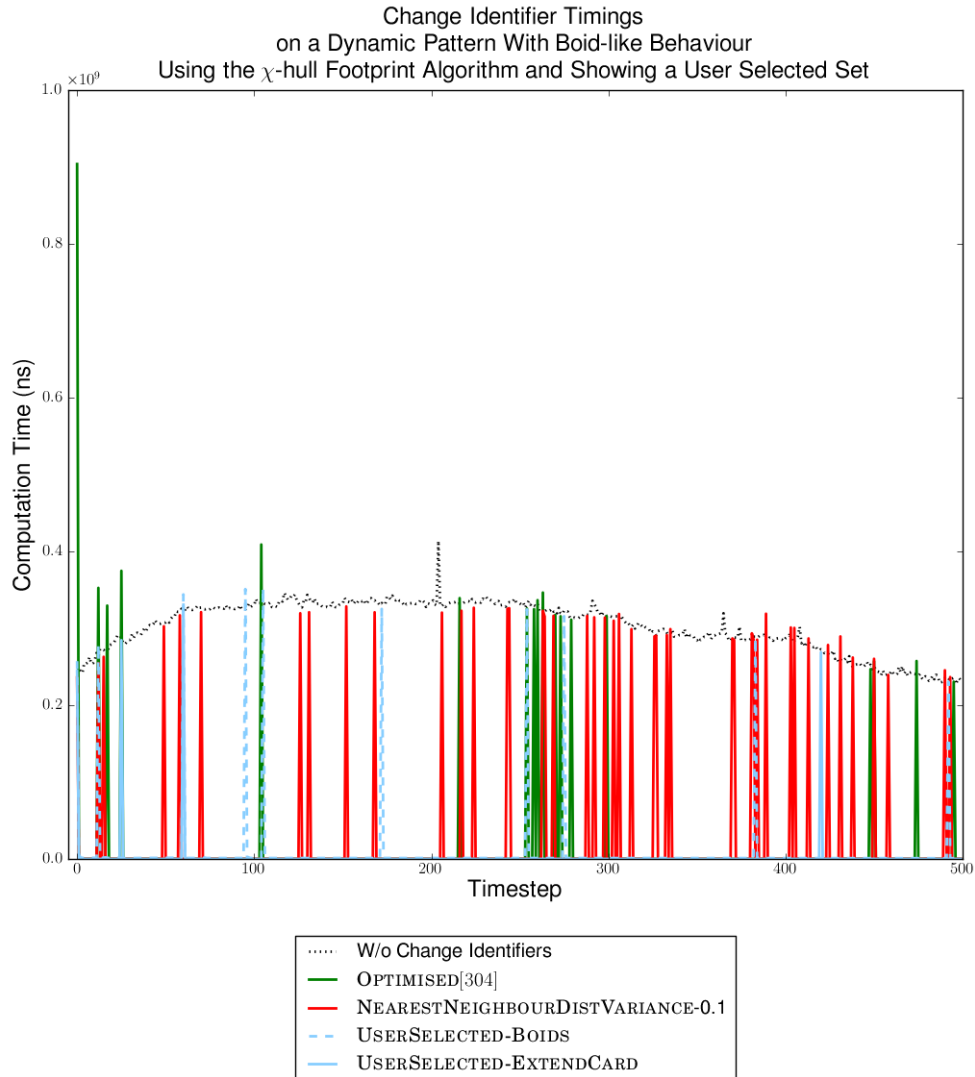


Figure A.3. Time taken to compute at each timestep for NEARESTNEIGHBOURDISTVARIANCE-0.1, OPTIMISED[304], USERSELECTED-BOIDS and the user selected change identifier set created for the Extent-Cardinality-Expand-Noisy dynamic dot pattern, now called USERSELECTED-EXTENDCARD using the χ -hull footprint algorithm on the boid like dynamic dot pattern.

```
1 <changeidentifierset name="UserSelected-Boids" ver="1.0">
2   <description>User selected set for dynamic dot patterns with boid-like
3     behaviour</description>
4   <maxFails>0.33</maxFails>
5   <concurrent>>false</concurrent>
6   <changeidentifier>
7     <identifier>DiameterSq</identifier>
8     <priority>0</priority>
9     <threshold>0.1</threshold>
10  </changeidentifier>
11  <changeidentifier>
12    <identifier>Centroid</identifier>
13    <priority>1</priority>
14    <threshold>0.1</threshold>
15  </changeidentifier>
16  <changeidentifier>
17    <identifier>NearestNeighbourDistVariance</identifier>
18    <priority>2</priority>
19    <threshold>0.1</threshold>
20 </changeidentifier>
</changeidentifierset>
```

Listing A.1 User Defined Change Identifier Set For Boid-Like Dynamic Dot Pattern

Bibliography

- 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.
- W. Ali and B. Moulin. 2d-3d multiagent geosimulation with knowledge-based agents of customers shopping behavior in a shopping mall. In A. G. Cohn and D. M. Mark, editors, *Spatial Information Theory*, volume 3693 of *Lecture Notes in Computer Science*, pages 445–458. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28964-7.
- H. Alt, B. Behrends, and Blömer. Approximate matching of polygonal shapes. Technical Report B96-03, Institut für Informatik, Fachbereich Mathematik, Freie Universität Berlin, 1993.
- H. Alt, P. Bra, M. Godau, C. Knauer, and C. Wenk. Computing the hausdorff distance of geometric patterns and shapes. Technical report, Institut für Informatik, Fachbereich Mathematik, Freie Universität Berlin, 2001.
- N. Andrienko and G. Andrienko. Designing visual analytics methods for massive collections of movement data. *Cartographica*, 42(2):117–138, 2007.
- A. Arampatzis, M. van Kreveld, I. Reinacher, C. B. Jones, S. Vaid, P. Clough, H. Joho, and M. Sanderson. Web-based delineation of imprecise regions. In *Computers, Environment and Urban Systems*, volume 30, pages 436–459. Elsevier, 2006.
- J. Basch, L. J. Guibas, and J. 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. ISBN 0-89871-390-0.
- R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1:173–189, 1972.
- M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. *Computational Geometry - Theory and Applications*, 2007.
- B. Bennett, D. R. Magee, A. G. Cohn, and D. 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.
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

- D. Black. *Investigation of the possible increased incidence of cancer in West Cumbria: report of the Independent Advisory Group*. H.M.S.O., 1984.
- P. Bogaert, N. van de Weghe, A.G. 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.
- A. R. 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.
- Y. Chiang and R. Tamassia. Dynamic algorithms in computational geometry. In *Proceedings of the IEEE*, volume 80, pages 1412–1434, 1992.
- E. Davis. Approximations of shape and configuration space. Technical Report 703, Department of Computer Science, New York University, 1995.
- M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry, Algorithms and Applications*. Springer, third edition, 2008.
- G. Del Mondo, J. G. Stell, C. Claramunt, and R. Thibaud. A graph model for spatio-temporal evolution. *Journal of Universal Computer Science*, 16(11):1452–1477, 2010.
- M. Delafontaine, A. G. Cohn, and N. van de Weghe. Implementing a qualitative calculus to analyse moving point objects. *Expert Systems with Applications*, 38(5):5187–5196, 2011.
- S. Dodge, R. Weibel, and A. Lautenschitz. Towards a taxonomy of movement patterns. *Information Visualization*, 7(3-4):240–252, 2008.
- M. Duckham, L. Kulik, M. Worboys, and A. 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.
- M. Dupenois and A. 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.
- H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois, 1992.
- H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. In *Proceedings of the 1992 workshop on Volume visualization, VVS '92*, pages 75–82. ACM, 1992.
- H. Edelsbrunner, D. G. Kirkpatrick, and R. 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.

- M. Egenhofer and R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.
- A. Galton. *Qualitative Spatial Change*. Oxford University Press, 2000.
- A. Galton. Pareto-optimality of cognitively preferred polygonal hulls for dot patterns. In *Spatial Cognition*, 2008.
- A. Galton and M. Duckham. What is the region occupied by a set of points? In *GIScience*, 2006.
- Y. Gofman. Outline of a set of points. *Pattern Recognition Letters*, 14(1):31–38, 1993.
- C. 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.
- R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1(4):132–133, 1972.
- L. 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.
- L. Guibas, M. Karaveles, and D. Russel. A computational framework for handling motion. In *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments*, pages 129–141, 2004.
- L. J. Guibas and R. Sedgwick. A dichromatic framework for balanced trees. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:8–21, 1978.
- J. Hershberger and S. 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.
- K. Hornsby and M. 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.
- Y. Huang, C. Chen, and P. Dong. Modelling herds and their evolvments from trajectory data. In *GIScience '08: Proceedings of the 5th international conference on Geographic Information Science*, pages 90–105, Berlin, Heidelberg, 2008. Springer-Verlag.
- 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.
- J. Jiang and M. 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. ISBN 978-1-60558-323-5.

- J. Jiang, M. Worboys, and S. Nittel. Qualitative change detection using sensor networks based on connectivity information. *GeoInformatica*, 15:305–328, 2011. ISSN 1384-6175.
- R. Klette and A. Rosenfeld. *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Morgan Kaufmann, 2004.
- D. Knuth. *The Art Of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, 2nd edition, 2007.
- P. Laube and R. S. Purves. How fast is a cow? cross-scale analysis of movement data. *Transactions in GIS*, 15(3):401–418, 2011.
- 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 Data Handling*, pages 2011–214. Springer, 2004.
- P. Laube, M. Duckham, and M. Palaniswami. Deferred decentralized movement pattern mining for geosensor networks. *International Journal of Geographical Information Science*, 25(2):273–292, 2011.
- M. 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.
- M. Melkemi and M. Djebali. Computing the shape of a planar points set. *Pattern Recognition*, 33(9):1423 – 1436, 2000.
- M. Melkemi and M. Djebali. Weighted \mathcal{A} -shape: a descriptor of the shape of a point set. *Pattern Recognition*, 34(6):1159 – 1170, 2001.
- A. Moreira and M. Y. 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.
- D. O’Sullivan and D. J. Unwin. *Geographic Information Analysis*. Wiley, November 2002. ISBN 0471211761.
- M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Science*, 23(2):166–204, 1981.
- C. 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. ISBN 0-89791-227-6.
- 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.
- P. H. Sneath. The application of computers to taxonomy. *J. Gen. Microbiol.*, 17:201–226, Aug 1957.

- J. 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.
- M. Thériault, C. Claramunt, and P. Villeneuve. A spatio-temporal taxonomy for the representation of spatial set behaviours. In M. Bhlen, C. Jensen, and M. Scholl, editors, *Spatio-Temporal Database Management*, volume 1678 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg, 1999.
- J. Žunić and P. L. Rosin. A convexity measurement for polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:173–182, 2002.
- J. Žunić and P. L. Rosin. Rectilinearity measurements for polygons. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1193 – 1200, September 2003.
- Z. Wood. *Detecting and Identifying Collective Phenomena within Movement Data*. PhD thesis, University of Exeter, 2011.
- Z. Wood and A. Galton. A taxonomy of collective phenomena. *Applied Ontology*, 4: 267–292, August 2009. ISSN 1570-5838.
- M. Worboys. Event-oriented approaches to geographic phenomena. *International Journal of Geographical Information Science*, 19:1–28, 2005.
- M. Worboys and M. Duckham. *GIS: A Computing Perspective*, chapter 6.4 Point Object Structures, pages 240 – 248. CRC Press, 2nd edition, 2004.