CHANGE: The Use of Change Identifiers to Update Footprints of Dot Patterns in Real Time

Maximillian Dupenois and Antony Galton

College of Engineering, Mathematics and Physical Sciences, University of Exeter, UK
{m.p.dupenois, a.p.galton}@ex.ac.uk

Abstract. hello

1 Introduction

It is possible to represent many real-world phenomena as point objects in a cartesian space. These 'dot patterns' occur in many GIS areas of interest, ranging from plotting the locations of buildings within cities to the the tracking of weather patterns.

Often it is desirable to be able to approximate the region containing the dots, this 'footprint' can be used to reduce the memory space taken up by the pattern or to be interpreted in such a way to increase the known information about the pattern.

There is a large body of work detailing the many processes that can be used to create footprints from dot patterns taken from a broad set of different contexts: geographical information theory [1, 10]; pattern recognition [13, 6]; computer vision [11]; and computational geometry [8] to note just a few. A more detailed examination of these algorithms and others can be found in [7]. These papers all work from the assumption that the patterns are static, however, in reality many of the patterns are intrinsically dynamic. Any flock or crowd will change in location and or membership over time [15]. Previous attempts at tracking moving point aggregrates have tended to be centered around object tracking within videos [5, 2]. Object tracking, however, is attempting to track a fixed shape amongst moving background as opposed to tracking a possibly changing shape with no background noise. Other work is in the same vein as Huang et al. [12] which concernins tracking herds, this is less interested in the footprint of the pattern and focuses on the herd as an abstract looking at on four major possible evolvements: *expand, join, shrink and leave*.

This paper examines the problem of tracking the footprint across a dynamically changing dot pattern. There are some cases in which a real-time update may be required, for example tracking the spread of a forest fire using a dispersed network of sensors. Most footprint algorithms run at $O(n \log n)$ time, where n is the cardinality of the pattern, and as a result running the algorithm for each update could lead to an increasingly out of step representation.

A possible answer is to incrementally update the footprint. Doing this in such a way as to be sure that the footprint at any given step is identical to that which would be achieved by recomputing. This would be similar to the certificate based system proposed in [3]. They, however, are only looking at updates to the convex hull which has

strictly defined mathematical terms. The more general idea of the footprint, which convex hull comes under,does not have such a strict definition and as such does not readily lend itself to this sort of processing.

Instead we make use of the vagueness implicitly within the footprint concept and propose that the footprint may not need to change every timestep. If the footprint was considered appropriate for one instant, it is unlikely that the very next instant will see it rendered as a poor approximation of the region occupied by the pattern. By examining the dot pattern as it relates to the footprint we can judge when the pattern has changed sufficiently (above some predetermined threshold) and recompute the footprint then. There will be a trade-off between the frequency of updates and the accuracy of the footprint, this is examined later in the paper.

Below is directly from ECAI, may need to alter it

Our approach is to use a suite of easily computable *change identifiers*, each with its own threshold. Recomputation of the footprint is triggered when some aggregate value computed from the values returned by the change identifiers exceeds a given threshold. In the simplest form this aggregate value could be a count of how many of the change identifiers individually exceed their thresholds, amounting in effect to a vote amongst the change identifiers. Alternatively, the change identifiers could be ranked in order of importance and a weighted combination of their values compared with some threshold. We investigate the effect of using different sets of change identifiers, and different ways of combining the results returned by them.

2 Dot Patterns

Before examining the process for creating change identifiers in any detail we must first be able to describe the dot patterns with more information than simply listing their coordinates. Without an accurate way of describing their properties there is no reasoned way in which to talk about their change.

When examining the patterns there are four immediate observations we can make. These are 'high-level' descriptors in that they can be used to describe a pattern both mathematically and qualitatively. For example the position of a pattern has an actual value and can be used to describe the pattern to a human.

- *Position* Where in the plane is the pattern situated?
- *Extent* How much of the plane does the pattern fill?
- *Cardinality* How many dots are in the pattern?
- *Density* How does the cardinality fill the extent?

There are two approaches to measuring each of these descriptors depending on how the pattern is viewed. The pattern can be seen as no more than the set of coordinate points; this leads to a set of methods for each descrtiptor using cartesian mathematics. We'll call methods based on this approach *base methods*. Alternatively a 'surrogate' footprint can be given to the pattern and simple shape measures can be used to attain values for the descriptors; called *surrogate methods*. The following subsections detail some of the methods for these measurements concluding with the two that we feel are most appropriate as base and surrogate methods for that descriptor

2.1 Position

The postion is one of the most immediately obvious measures for which noting change is important. However, as the pattern is a distributed set, is not a uniquely defined property. Some of the ways in which it can be measured are:

- Base
  - The mean average position of the dots (the centroid).
  - The median average position of the dots.
- Surrogate
  - The centre point of the isothetic[1] minimum bounding box[2].
  - The centre point of the minimum bounding circle.

Looking first at the base methods. The mean is commonly used as it implicitly takes into account the distribution of the set. However the median has the benefit of disregarding outliers; dots that as so far from the majority of the set that they would drag the centroid towards them. In terms of requirements the median needs an ordered listing of the dots and the ordering used will change the result; ordering by x then y can give a different median to ordering by y then x. The mean does not have this ambiguity nor the extra complexity from sorting the set. The mean is also an intuitive measure as evidenced by how often it is used.

For the surrogate methods the bounding box is simple to compute but is rotation dependant; if the box is not isothetic it could have a different center. The minimum bounding circle is rotation independent but computationally complex. For the purposes of the change identifiers the complexity must be kept low, as such the bounding box seems the most appropriate choice.

Base: The mean average position.
Surrogate: The center of the bounding box.

2.2 Extent

Measuring the extent is a measurement of the size of the pattern[3]; a value that is likely to change over any dynamic pattern. Possible methods of measurement are:

- Base
  - The standard deviation.
  - The variance.
  - The diameter.
- Surrogate
  - The area of a the bounding box.
  - The diagonal of the bounding box.
  - The area of the minimum bounding circle.
  - The diameter of the minimum bounding circle.

[1] Aligned to the xy-axis
[2] Hereafter any reference to the bounding box is the isothetic minimum bounding box
[3] We do not use the term size as it is as easily taken to mean cardinality as it is extent

For the surrogate methods we can ignore the minimum bounding circle for the complexity reasons given in the discussion of the position descriptor. The difference between the area and the diameter as measures is computationally negligible. However, the area more accurately fits the description of extent as it represents an area filled.

The standard deviation[4] and the variance are measures of the same effect; variance is the standard deviation squared. The variance is computationally less expensive but has a squared unit. In most situations the standard deviation is preferred as it gives a result in the same unit as the original data. However we are only interested in measuring the difference between different values and as long as the unit doesn't change we do not need it to be to the same power as the data. Further to this by using variance both the surrogate and the base methods return a squared unit, making comparison somewhat fairer. The diameter is perhaps the most intuitive extent measure from the base methods, however it is computationally expensive. The most efficient methods requiring the computation of the convex hull of the pattern first; making the complexity $O(n \log n)$. As such the variance is the most appropriate base measure.

Base: The variance.
Surrogate: The area of the bounding box.

2.3 Cardinality

Changing membership is clearly indicative that the footprint may need to change. Within this paper we do not make use of it as a change identifier but this is as the test dot patterns have a fixed cardinality. However it is not a useless descriptor and we should examine its methods.

From a base level there's only one way it can really be measured: by actually counting the number of dots. As a surrogate method it is bit more complex. Footprints can often be made up of seperate un-connected components (Swinging Arm [10] and α-shapes [8]) if the dot pattern has areas of different density. A surrogate method for cardinality could involve counting the number of components that make up the surrogate footprint. While this would be an interesting descriptor and quite a useful change descriptor no simple methods for doing so are immediately apparent; so it is unlikely that a simple surrogate method exists.

Base: The number of dots.
Surrogate: Potentially the number of components if a suitable method for computation can be found.

2.4 Density

The density is the amount of dots per unit area of the pattern. As a descriptor it is simply the cardinality divided by the extent. The question is does measuring the density provide any new information considering that it is a combination of two existing descriptors?

[4] positive square root of the mean of the squared deviations from mean; in this case the square root of the mean of the squared distances of each dot from the centroid

As a descriptor it is one of the most immediately obvious ways of describing a pattern and because of its intuitive nature we don't feel we can ignore it without first seeing if it provides different results beyond a combination of both extent and cardinality.

As mentioned above the cardinality in the presented experiments does not change so density works as a measure of extent. If it gives significantly different results to extent we can state that the approach in which it measures change is different enough to allow it to be a valid descriptor.

As the surrogate method for cardinality is unlikely to exist in a computationally simple fashion we use the base method of a cardinality divided by the area of the bounding box.

The density may vary across the set with some areas being sparse and others being very densely populated. In mixed density cases a further descriptor may be to estimate the number of groupings of dots. As for the surrogat method for cardinality. This can be done in a few ways, for example k-nearest neighbour algorithms or kernel density functions. Unfortunately the complexities for these sorts of calculations can be quite high and the time spent would have to be weighed against the potential gain.

Base: The number of dots divided by the variance.
Surrogate: The number of dots divided by the area of the bounding box.

3 Change Identifiers

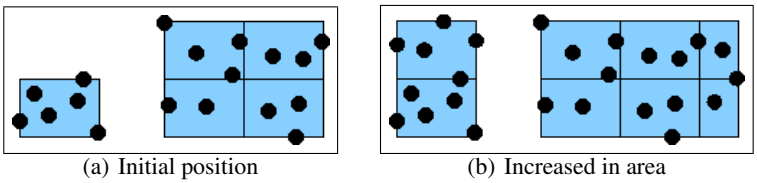
While it is not difficult to think up possible measurements to be used within change identifiers for there to be a strong argument to make use of them their creation needs to approached in an analytical manner. First the requirements must made clear:

- **Req. 1** A change identifier must complete in less time than it takes to compute the footprint with the current algorithm. It is probably less complex than the footprint algorithm but not necessarily so.
- **Req. 2** A change identifier must return a single value indicating how much change has occurred from the previous change causing dot pattern to the current.

Second the types of change that a pattern can undergo must be identified. For the purpose of this paper we will use the four 'high-level' descriptors given in §2 and measure them using the methods given.

Third the value that is returned must be discussed; change of the dot pattern can not be measured by just the change in the measured descriptor. For the change identifier to be useful it has to be able to represent proportional change. For example, with a change identifier that measures change in extent via a surrogate, a large dot pattern can change by the same area value as a smaller dot pattern (Figure 1) and have less of a proportional change. The smaller bounding box doubles in area compared to the larger box which has a quarter of its original size added to it. If, instead of measuring just the change value, the measurement difference is first divided by the measurement from the previous update causing pattern we have a value that shows the difference in the proportional amount of change that the two patterns in have undergone.

With these concepts clear a formal description of a change identifier can be made:



(a) Initial position (b) Increased in area

Fig. 1. Area Comparison

Formal Description

A change identifier is a method for returning a value representing the change in a descriptor of a dot pattern that is proportional to the value of the descriptor at the last footprint update. It takes less time to run than the footprint algorithm for which it is being used.

Having formally described the concept behind change identifiers we can define a notation for them that eliminates possible abiguity:

DP_t: Dot pattern at time step t. If t is the current time step this is reffered to as the **current dot pattern**
$change_x(\mathrm{DP}_1, \mathrm{DP}_2)$: Change identifier value for descriptor x on dot patterns DP_1 and DP_2
$desc_x(dp)$: Value of descriptor x on dot pattern DP
u: Time at which the footprint was last updated

$$change_x(\mathrm{DP}_t, \mathrm{DP}_u) = \frac{|desc_x(\mathrm{DP}_t) - desc_x(\mathrm{DP}_u)|}{desc_x(\mathrm{DP}_u)}$$

We should be able to create an identifier for each of the dot pattern descriptors; both the base and surrogate methods. The following subsections do this by providing the notation for each descriptor. The name given is to each identifier is in small caps (NAME) and is used as a short-hand symbol to reference that identifer.

3.1 Position:

Note that position is made proportional in a different fashion to the others. The other descriptors return a measurable value while position returns a coordinate. It is not possible to divide by a coordinate so instead the change in position is divided by the appropriate extent (variance for the base and area for the surrogate). The extent is chosen as the change in position should be proportional to the space taken up. For example a movement of 6 units is proportionally greater for a pattern that has a bounding box area of 6 units^2 than it is to one with a bounding box area of 60 units^2.

Base:

Name: MEAN
Key:
$mean(\mathrm{DP})$: Mean average location of all dots in dot pattern DP.
$vari(\mathrm{DP})$: Variance of the dots in dot pattern DP.
p: A dot from a dot pattern.
$|DP_t|$: Number of dots in pattern DP_t (cardinality).
pb: Descriptor for base position.
Notation:

$$mean(\mathrm{DP}_t) = \frac{\sum_{i=0}^{|DP_t|} p_i}{|DP_t|}$$

$$vari(\mathrm{DP}_t) = \frac{\sum_{i=0}^{|DP_t|} (mean(\mathrm{DP}_t) - p_i)^2}{|DP_t|}$$

$$desc_{pb}(\mathrm{DP}_t) = mean(\mathrm{DP}_t)$$

$$change_{pb}(\mathrm{DP}_t, \mathrm{DP}_u) = \frac{|desc_{pb}(\mathrm{DP}_t) - desc_{pb}(\mathrm{DP}_u)|}{vari(\mathrm{DP}_u)}$$

Surrogate:

Name: MBBCENTER
Key:
$mbb(\mathrm{DP})$: Minimum bounding box of dot pattern DP.
$center(r)$: Center of region r.
$||r||$: Area of region r.
ps: Descriptor for surrogate position.
Notation:

$$desc_{ps}(\mathrm{DP}_t) = center(mbb(\mathrm{DP}_t))$$

$$change_{ps}(\mathrm{DP}_t, \mathrm{DP}_u) = \frac{|desc_{ps}(\mathrm{DP}_t) - desc_{ps}(\mathrm{DP}_u)|}{||mbb(\mathrm{DP}_u)||}$$

3.2 Extent:

Base:

Name: VARIANCE
Key:
eb: Descriptor for base extent.
Notation:

$$desc_{eb}(\mathrm{DP}_t) = vari((\mathrm{DP}_t))$$

$$change_{eb}(\mathrm{DP}_t, \mathrm{DP}_u) = \frac{|desc_{eb}(\mathrm{DP}_t) - desc_{eb}(\mathrm{DP}_u)|}{desc_{eb}(\mathrm{DP}_u)}$$

Surrogate:

Name: MBBAREA
Key:

es: Descriptor for surrogate extent.

Notation:

$$desc_{es}(\mathrm{DP}_t) = ||mbb(\mathrm{DP}_u)||$$
$$change_{es}(\mathrm{DP}_t, \mathrm{DP}_u) = \frac{|desc_{es}(\mathrm{DP}_t) - desc_{es}(\mathrm{DP}_u)|}{desc_{es}(\mathrm{DP}_u)}$$

3.3 Cardinality:

Base:

Name: CARDINALITY
Key:

cb: Descriptor for base cardinality.

Notation:

$$desc_{cb}(\mathrm{DP}_t) = |DP_t|$$
$$change_{cb}(\mathrm{DP}_t, \mathrm{DP}_u) = \frac{|desc_{cb}(\mathrm{DP}_t) - desc_{cb}(\mathrm{DP}_u)|}{desc_{cb}(\mathrm{DP}_u)}$$

Surrogate: Not Applicable.

3.4 Density:

Base:

Name: DENSITYBYVARIANCE
Key:

db: Descriptor for base density.

Notation:

$$desc_{db}(\mathrm{DP}_t) = \frac{|DP_t|}{vari((\mathrm{DP}_t))}$$
$$change_{db}(\mathrm{DP}_t, \mathrm{DP}_u) = \frac{|desc_{db}(\mathrm{DP}_t) - desc_{db}(\mathrm{DP}_u)|}{desc_{db}(\mathrm{DP}_u)}$$

Surrogate:

Name: DENSITY
Key:
ds: Descriptor for surrogate density.
Notation:

$$desc_{ds}(\mathrm{DP}_t) = \frac{|DP_t|}{||mbb(\mathrm{DP}_u)||}$$
$$change_{ds}(\mathrm{DP}_t, \mathrm{DP}_u) = \frac{|desc_{ds}(\mathrm{DP}_t) - desc_{ds}(\mathrm{DP}_u)|}{desc_{ds}(\mathrm{DP}_u)}$$

This by no means an exhaustive list as there are many other change measures that can be used, for example, the symmetric area difference between the bounding box of the current pattern and the bounding box of the last change causing dot pattern. However identifiers that can catch change in the given descriptors should suffice to show that identifiers can be created to cover the major types of change.

4 Using Change Identifiers

The change identifiers are interesting in of themselves as descriptions of change but need a well defined framework in which to operate. The basic process we implement is shown in Algorithm 4, which works as follows. The incoming data consists of a sequence of dot patterns (which might come from, e.g., observations relayed by sensor arrays). An algorithm for generating footprints from dot patterns is assumed given (we shall refer to this as the **footprint algorithm**), and at the beginning of the sequence a footprint $foot(\mathrm{DP}_0)$ is generated for dot pattern DP_0 and saved as the **stored footprint** SFP_0. The dot pattern DP_0 from which it is generated is stored as the **stored dot pattern** (SDP_0). Earlier the stored pattern is referred to as DP_u; the altered nomenclature is to allow the identify which was the last update causing pattern at any time step (SDP_i).

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 dot pattern DP_i differs from the previously stored dot pattern SDP_{i-1}. If this value, $eval(\mathrm{DP}_i, \mathrm{SDP}_{i-1}, \mathrm{SFP}_{i-1})$, exceeds some pre-set threshold, then a new footprint $foot(\mathrm{DP}_i)$ is generated as the new stored footprint SFP_i, and the current dot pattern 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 dot pattern DP_i, the footprint $foot(DP_i)$ 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.

5 Analysis

To be able to prove that change identifiers are a useable concept we need to be able to measure the 'quality' of the footprint. It should be stressed that we are not commenting

Algorithm 1 Basic Process

1: $i = 0$
2: Input first dot pattern DP_0
3: $\mathrm{SFP}_0 = foot(\mathrm{DP}_0)$
4: $\mathrm{SDP}_0 = \mathrm{DP}_0$
5: **repeat**
6: $\quad i = i + 1$
7: $\quad$ Input DP_i
8: $\quad$ **if** $eval(\mathrm{DP}_i, \mathrm{SDP}_{i-1}, \mathrm{SFP}_{i-1}) > threshold$ **then**
9: $\quad\quad \mathrm{SDP}_i = \mathrm{DP}_i$
10: $\quad\quad \mathrm{SFP}_i = foot(\mathrm{DP}_i)$
11: $\quad$ **else**
12: $\quad\quad \mathrm{SDP}_i = \mathrm{SDP}_{i-1}$
13: $\quad\quad \mathrm{SFP}_i = \mathrm{SFP}_{i-1}$
14: $\quad$ **end if**
15: **until** No more input available

on how well the footprint algorithm can create a footprint that represents the pattern; we assume that the algorithm used was chosen for a reason. The 'quality' we measure is how close the stored footprint is to the true footprint at any given step. Obviously to minimise the difference between the stored and true footprints there would need to be an update at each time step. However, aside from the 'quality' we are also attempting to reduce the length of time taken to run each step's computation. These two objectives are conflicting and as such there is a trade off between them.

So that we can run experiments on the change identifiers we have created a test framework in Java that allows us to feed in a stream of dot patterns with a fixed length. Over the course of a run it produces large amounts of test information, including: the true footprint at each time step; the stored footprint at each time step; the time that would be taken to obtain the true footprint; and the time taken to produce the stored.

Looking first at the process of measuring time across the stream.

- $t_{FP}(i)$ is the time taken to compute the footprint from the dot pattern at step i.
- $t_{CI}(i)$ is the time taken to evaluate the change identifiers at step i.
- $r(i)$ is a Boolean variable, set to 1 if the footprint is in fact recomputed at step i, and zero otherwise.

The total computation time over a run of n dot patterns is thus

$$T_{CI} = t_{FP}(0) + \sum_{i=1}^{n} (t_{CI}(i) + r(i) t_{FP}(i)).$$

The value of T_{CI} 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 $1 \leq i \leq n$):

$$T_{\min} = t_{FP}(0) + \sum_{i=1}^{n} t_{CI}(i).$$

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):

$$T_{\max} = t_{FP}(0) + \sum_{i=1}^{n} (t_{CI}(i) + t_{FP}(i)).$$

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

$$T_{NCI} = \sum_{i=0}^{n} t_{FP}(i) = t_{FP}(0) + T_{\max} - T_{\min}.$$

If it is assumed that always $t_{CI}(i) < t_{FP}(i)$ (for if not, there would be little point in using change identifiers) then $T_{\min} < T_{NCI} < T_{\max}$, so the relative size of T_{CI} and T_{NCI} — which provides a measure of the time advantage, if any, gained by using change identifiers — depends on the threshold settings.

Instead of examining the difference between the stored and the true footprints in the context of measuring quality, we inverse the concept and consider the difference as a cost. This allows us to give the goal as a minimisation of both time and cost; thinking of both objectives in the same terms should be less confusing.

To measure this cost, we need a way of quantifying the extent of this mismatch. The difference between two footprints can be measured in various ways, e.g., using Hausdorff distance, or symmetric area difference (see [9, Ch. 7] for a discussion). Here we will use only the symmetric area difference, but the principles described below would apply equally to other measures.

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

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

We use the area of this as a measure of the dissimilarity between two footprints; and since we are only interested in comparisons, not absolute values, we normalise this area by expressing it as a fraction of the area of the 'true' footprint (FP_i) [5]. Thus the aggregate mismatch between the stored footprint and the true footprint over a dot-pattern sequence of length n is given by

$$mismatch = \sum_{i=0}^{n} \frac{||FP_i \Delta SFP_i||}{||FP_i||},$$

If the footprint is recomputed every time, corresponding to total computation time $T_{\max}$, we have $SFP_i = FP_i$ for every i, 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 accuracy and computation time, as indicated in Figure 2, where different choices of change identifier thresholds correspond to different positions on the curve. The optimal setting for the change identifier

[5] We use FP_i instead of $f(DP_i)$ for clarity within the formula

threshold depends on the relative importance attached to the conflicting goals of minimizing both computation time and accumulated footprint error; but in any case no time advantage can be obtained for mismatches below the value m at which $T_{CI} = T_{NCI}$.

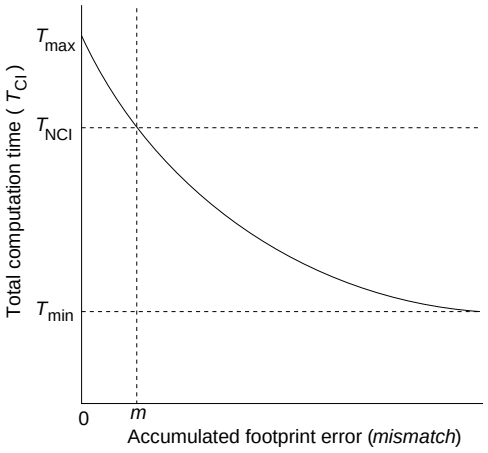


Fig. 2. Total computation time against aggregate footprint error

6 Implementation

One of the important facets of the change identifiers that we've not yet mentioned is the threshold parameter they require. Without this they have no way of actually causing an update. Choosing a suitable parameter is not as difficult as it would first seem. As the values are all proportional choosing the threshold is equivalent to choosing the proportion of change that would be unsuitable for the context. For example an application to measure the spread of a fire by a distributed sensor network might allow for the distribution to change by up to 60% but is going to have a much smaller threshold for scale. Even so there is a certain amount of fine tuning that needs to be done for individual contexts. To aid this the identifiers are specified by an xml document with a an element for the threshold.

Each of the identifiers specified above have been created to service one aspect of change. The framework we use allows us to combine the change identifiers so we can catch varying types of change. The xml document provides this facility and also allows us to control some extra useful parameters. We can control the order in which the identifiers are run; the individual thresholds; a multiplier for each identifier[6]; and a global threshold for the change identifier set.

[6] When using many identifiers one may have greater importance in the application than another

7 Results

We have run tests on streams of 500 dot patterns containing up to 500 dots each[7]. We have implemented a collective motion pattern generator which can use different methods to produce streams of dot patterns. The method that generated the patterns for the current tests makes use of the principles of separation, cohesion and aggregation used to define behaviours in the Boids system of [14]. The footprint algorithm used in the tests displayed here is the χ-hull [6] but tests have also been run on the upper and lower convex hull algorithm as given in [4].

In the experiments the symmetric area difference is given as a percentage. These percentages are plotted against each time step Figure 4 and the area under the graph gives a total areal percentage difference over the course of the run. This value is divided by the number of patterns in the stream to give the percentage symmetric area difference per time step a measure we're calling the 'cost'.

The 'Redrawn' column in Table 7 is simply the number of times the change identifiers caused an update. Unless otherwise specified the identifiers all have a threshold of 0.1 as it is a low enough value to cause change (10%) while not being so low as to guarantee change if the pattern changes little in the descriptor being measured.

Change Identifier	Time W/ CI(ms)	Time W/out CI(ms)	Time Diff.(ms)	Cost(%)	Redrawn
MEAN	10849	196480	185631	86.879	27
MBBCENTER	1924	192333	190409	147.275	4
VARIANCE	2880	191822	188942	120.345	7
MBBAREA	2868	192453	189585	150.191	7
DENSITYBYVARIANCE	2120	192503	190383	133.294	5
DENSITY	2447	191641	189194	149.051	6
ALLBASES	15594	193558	177964	74.522	39
ALLSURROGATES	97348	191911	94563	16.963	250
ALL	98513	194738	96225	16.963	250

The cost seems high for the identifiers by themselves but that is to be expected. The pattern, as mentioned earlier, exhibits boid-like behaviour and as such changes often and chaotically; a single identifier can only account for the change that it measures. The values returned by the combinations of identifiers are far more interesting. Using all the base values (ALLBASES) gave minimal cost decrease for a large time gain against the results of MEAN; which out performed all the other identifiers. However the best results came from the set containing all the surrogate identifiers (ALLSURROGATES) with a time taken of 50.73% of the time taken without and a cost of only 16.963%. Combining the bases and surrogates together ALL wasn't capable of improving on the values and given that the updates are identical it appears as if the base identifiers were overpowered by the surrogates. It may seem strange that the surrogates by themselves only managed a total of 17 footprint updates but when combined caused 250. This is

[7] Larger tests have been run with equivalent results, however the graphs do not scale well and are unsuited for display in this paper

because the set has a global threshold of 0.1 and the values returned by each identifier are summed. If the sum crosses the threshold then an update occurs. Also worthy of note is that if the footprint needed updating at each time step then we would expect updating at every other timestep to produce a cost of 50%. As the actual cost was 16.963% we have shown that a footprint, even on a radically changing pattern, does not neccessarily need to be updated at each step.

As all the experiments were run on the same dot pattern, on the same computer, on the same footprint algorithm and with the same threshold values they can be fairly compared. To do this we produce a graph of cost against time taken for each identifier (Figure 3).

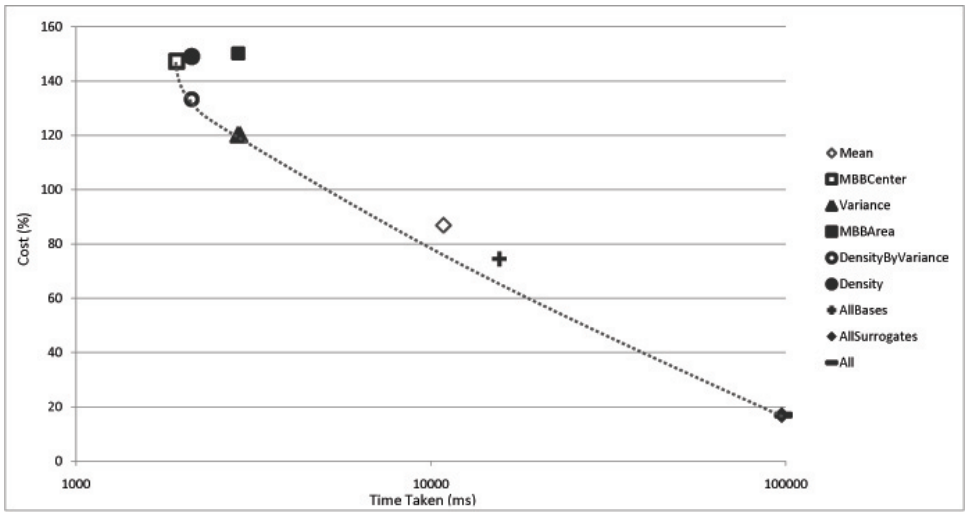


Fig. 3. Plot of identifiers on graph of cost against time taken

The dotted line represents the pareto-front of the identifiers; the identifiers which do no worse for either objective (minimising cost and time taken) than any other but better or equal on at least one. This concept is called dominance and is a way of comparing multi-objective problems. A solution is said to dominate another iff it performs better in all objectives. If it performs worse in all then it is dominated, otherwise the solutions are said to be mutually non-dominating. We should stress that the this graph only represents results for the dot pattern stream the footprint algorithm used so we can not make sweeping generalisations, but we can comment on how they compare in this instance. The identifiers appearing on the pareto-front are MBBCENTER, DENSITYBYVARIANCE, VARIANCE, and ALLSURROGATES. These represent, respectively, position; density; extent; and a combination of all three descriptor types. The fact that these all appear gives weight to the statement that all of the descriptor types are relevant and that they can be combined effectively.

The graph does not have many identifiers on it, but even with the small number of results it has a clear similarity to the expected graph for altering the threshold values Figure 2. This makes sense given that the we can see each solution on a graph with different thresholds as being different change identifiers. With this in mind we can imagine a graph where not only are there multiple identifiers but each set appearing a number of times with varying thresholds.

This ability to compare identifiers is useful when creating new ones: if an identifier consistently dominates all others it may be able to replace them in sets that contain it; thereby reducing the processing time of the set for no increase in the cost.

Further to the tabular information the experiments also graph the results.

Figure 4 shows the time taken for processing at each time step. The dashed line represents the run performed updating at each time step and the solid line represents the run with change identifiers. Each marker (empty discs for updating each time and filled squares for the change identifier run) is at a time step. As can be seen the processing time when using change identifiers consistently well below the processing time when not. For any relatively large set (200 dots upwards) and any complex footprint algorithm[8] we've found this to be the case.

There are several abnormally large spikes for both the run with and the run without. These are because of the measurement unit being small (milliseconds) anytime the computer uses the processor for any other purpose this can make a noticeable difference to the time taken for the experiments. These aberrations affect both with and without equally so cancel each other out when comparing the two time values.

Figure 5 shows the percentage symmetric area difference between the stored and true footprints at each time step. Each marker where the line touches the x-axis is representive of an update. On many of the spikes it can be seen that the lead up to them is more gradual than the drop. This is to be expected as the difference between the stored and true footprint build until one of the change identifiers breaches its threshold causing the footprint to update.

8 Conclusions and Further Work

We proposed the concept of change identifiers as a way of reducing uneccessary updates of the footprint of a dynamically changing dot pattern. By examining the information implicit in the static pattern we defined descriptors that would change in value concurrently with change in the pattern. These descriptors were turned into measures of change from one pattern to the another; the functions for obtaining these measures are called change identifiers. A way of assessing the performance of the identifiers was conceived of by measuring the footprint at any time step with the footprint that would be produced had there been an update (stored footprint against true footprint). The change identifiers were then run in a series of experiments to see if time could be saved while not compromising on accuracy.

The experiments bear out the original hypothesis that large amounts of time can be saved by not updating the footprint at each time step. We have only briefly touched on

[8] As opposed to a simple one like the minimum bounding box

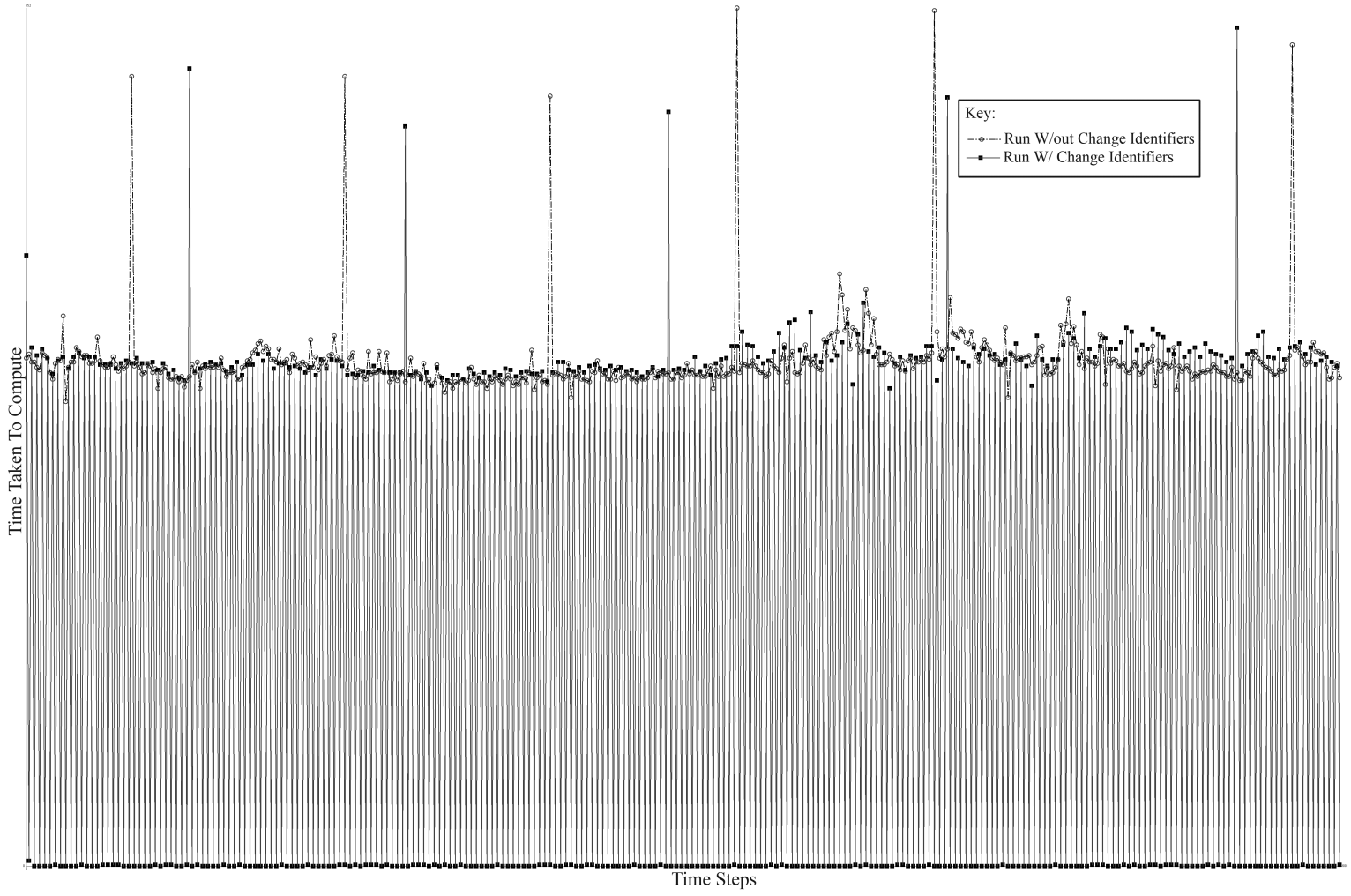


Fig. 4. Graph of Time Taken against Time Steps for ALLSURROGATES

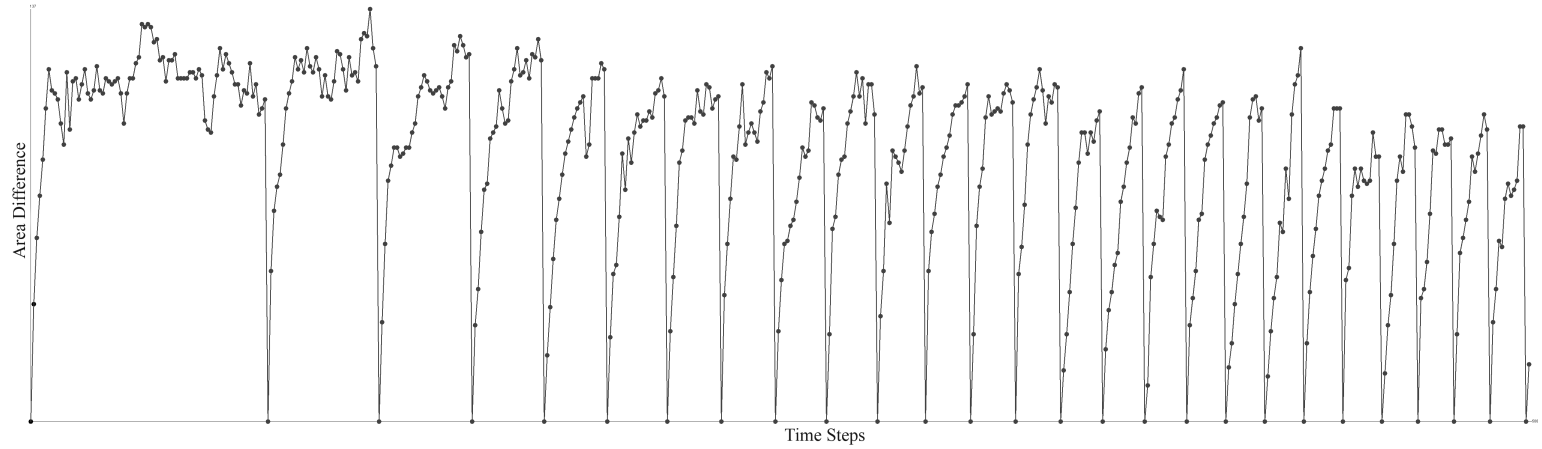


Fig. 5. Graph of Symmetric Area Difference against Time Steps for MEAN

what value a reasonable cost would be; we suspect that this is context specific but there may be a way of ascertaining an average optimum trade-off by looking at the issue as an optimisation problem. That more time saved does lead to a greater cost indicates a multi-objective problem and we have made some forays into running evolutionary algorithm experiments to see if we can optimise the choice of identifiers and their thresholds. This experimentation is too early in development to be discussed in great detail for this paper but the preliminary results seem promising.

There are many other types of identifiers that can be created, for example: symmetric area difference of the minimum bounding box; proportion of dots outside the boundary of the stored footprint; and even a random identifier that at any time step has a given percentage chance of forcing an update. We intend to perform a larger set of tests to see if further information can be gained from these other identifiers.

With regard to types of dot pattern, [15] proposes several types of collective movement. Sets of dot pattern streams that for each of these types would allow a test to be created for the identifiers such that, depending on their performance, they could be said to be useful in all situations, only in some specific cases, or for none. We also wish to apply the system to some real-world examples.

As mentioned above the identifiers have been tested on a convex hull algorithm with similar results. However there are many more non-convex algorithms that can be tested against. For completeness the α-shape from [8] and the swinging-arm algorithm from [10] will be implemented.

Other accuracy measures, e.g., Hausdorff distance, will also be implemented, and it will be interesting to see how they relate to each other.

Bibliography

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

[2] Shai Avidan. Ensemble tracking. In *In CVPR*, pages 494–501, 2005.

[3] Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, SODA '97, pages 747–756, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics. ISBN 0-89871-390-0.

[4] Mark Berg, Otfried Cheong, Marc Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, April 2008.

[5] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 25 (5):564–577, 2003.

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

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

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

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

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

[11] Gautam Garai and B. B. Chaudhuri. A split and merge procedure for polygonal border detection of dot pattern. In *Image and Vision Computing*, volume 17, pages 75–82. Elsevier, 1999.

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

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

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

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