Background on Dynamic Algorithms

Max Dupenois

August 6, 2009

When it comes to footprints the majority of the work has centered on their generation. However there are large number of applications which involve motion. As such, it seems that merely generating the shape is not enough in many cases. So, are there mehods which not only produce the footprints but also allow them to be maintained as the dots change position? With this in mind the following is an examination of the methods used in dynamic updates particularly concerning geometric structures.

First the Kinetic Data Structures (KDS), proposed by Basch et al. [1]. This is particularly appropriate because one of the applications they link it to is that of 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 geometric relations that possibly describe the shape 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*. Futher to this a KDS is *efficient* if there are very few internal events compared to external events[1], *compact* if it has a near linear number of certificates and *local* if no object participates in too many certificates.

The KDS uses short term motion plans for the objects, these are used to sort the events into queues such that the mosy 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 $\text{ccw}(a, b, c)$ (where a, b and c are points and the relation is true if they form a counter-clockwise triangle). 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, footprints are vaguely defined. As a result choosing the certificates is no longer a trivial task. Although the classification system we produced may well play some part in defining if the shape is still valid, possibly replacing the certificates.

Hershberger and Suri [3] have a very different idea using 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 over each line segment. There is a little confusion over these areas of uncertainty in that they're supposedly constructed from the supporting lines of the vertices,

[1] *External Event*: Changes the shape of the shape. *Internal Event*: Shape stays the same, certificates change.

however which supporting line (considering there are technically infinite) isn't given. Although from the given diagram it appears as though they mean the supporting line perpindicular to the direction in which the point was found. While sampling is not a new concept Hershberger and Suri [3] suggest that uniform sampling produces poor quality approximations in low curvature regions. As such they propose an adaptive sampling scheme.

The method works thusly, first they uniformly sample extrema in directions $2\pi/r$ for $j = 0, ..., r-1$ then they add up to r more extrema using their adaptive technique. Given an edge e let $\Theta(e)$ be the minimum angle between the directions the endpoints were sampled in. Using this they work out the proportion of the perimeter that is made up by e. If e is a large perimeter contribution then it 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.

This concept is far more easily applicable to the footprint problem although it's interesting to note just how vastly different it is in approach to the KDS idea. It does still give an issue with relating the fact that there is no single footprint the area of uncertainty is much harder to assess.

Chiang and Tamassia [2] present a review of the field. While it is a little dated it still serves as a good presentation of methods still in use. The first part of the paper looks at a different area to the other papers currently looked at. Instead of the updating of the structure or the ways in which it can be approximated they look at the data structure that represents the geometry.

They present a few storage methods; various forms of binary trees and *fractional cascading*. Either of which may be useful to the field of footprints if a suitable way of arranging the data can be created. Next they approach some general dynamization methods. Going through all the methods would be essentially repeating their words but there are a couple of terms worth noting:

- **Local rebuilding / Balancing** Technique applied to search trees 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''.

Jumping forward to Section 7 convex hulls make an appearance. 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 S is on the convex hull H of S;
- find if a query point is internal or external to the convex hull H of S;

- find the tangents to the convex hull H of S from an external query point;
- find the intersection of the convex hull H of S with a given query line;
- report the points on the convex hull H of S.

However they do point out that the set of points S is updated only by insertions and deletions so any point movement should be treated as being removed then added as a new point. This may be a perfectly valid assumption and it does seem to make life simpler in terms of algorithm creation but whether or not a better algorithm could be created with knowledge of point identity may be worth considering.

They describe a method by Preparata with update and query time of $O(\log n)$ and a report-query time of $O(n)$ where n is the number of vertices currently in the convex hull H of S. The method only deals with deletions and is therefore not entirely applicable but it does introduce the concept of splitting it 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.

References

[1] Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. To Appear in Journal of Algorithms, 1997.

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

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