NTNU
Norwegian University of
Science and Technology

Heuristics for Dynamic Delaunay Triangulation

Victor Fielding

Master of Science in Computer Science
Submission date: June 2018
Supervisor: Magnus Lie Hetland, IDI

Norwegian University of Science and Technology
Department of Computer Science

TDT4900 Datateknologi, masteroppgave

Heuristics for Dynamic Delaunay Triangulation

Victor Fielding Supervisor: Magnus Lie Hetland

Spring 2018

Abstract

The Delaunay triangulation is a useful tool for organising the relationship of points. When these points move slightly the triangulation could be maintained by selectively processing regions that no longer have a valid relationship. Multiple methods for detecting these regions have been explored and an approach for triangulating these regions have been made. The approach is heuristic and can be tuned to be arbitrarily precise. It is in the majority of cases more efficient than conventional static triangulation algorithms, due to reusing existing information in the triangulation. Speedups of over 10x have been achieved in some cases. This approach has been tested on a simulation similar to Smoothed-particle hydrodynamics. Applications could be in areas like Robotic Mapping and Navigation systems through traffic.

Sammendrag

Målet med denne forskningen er å forbedre en metode for å vedlikeholde en Delaunay triangulering av punkter i bevegelese. Denne metoden skal utnytte informasjonen i den eksisterende trianguleringen for å spare ressurser.
Forskjellige løsninger blir presentert, inkludert løsninger som bruker heuristikker. Styrker og svakheter ved hver av disse blir sammenlignet og diskutert.

Preface

This thesis was made during the subject TDT4900 - Computer and Information Science, Master Thesis at the Department of Computer Science, under the Faculty of Information Technology and Electrical Engineering at the Norwegian University of Science and Technology.

The idea for this work came from a hobby project of making a physics simulation and the task of making it more efficient. I humbly thank my supervisor Magnus Lie Hetland for his generous help and brilliant conversations.

The work in this paper follows from work done in [Fie17] from the same author. Many foundational concepts will therefore be repeated. The text in Introduction, Chapter 2 Background Theory and Section 5 Classifiers in Chapter 4 are based on earlier work or copied for reader convenience.

Contents

Chapter 1

Introduction

The Delaunay triangulation and Voronoi Diagram are widely known concepts in Computational Geometry. They are tools used in various spatial situations like solving navigational tasks and generating favourable topologies.

The core problems in computational geometry could be classified as: Static problems, Dynamic problems, Geometric query problems and variations of problems. In Static problems, the situation stays constant and there is no consideration of time. Dynamic problems involves finding a solution and maintaining it after each incremental modification of the input data.
A point set triangulation is the result of connecting points by edges, so that no edges cross and that the area between points is fully tiled by triangles. This is a static problem when the points are not moving. Delaunay Triangulation is a popular point set triangulation that is used for a number of tasks, including creating a polygon mesh from point clouds for 3D modelling or for physics simulators. Creating a mesh for 3D modelling is a static problem since the point cloud is not moving or changing. While for physics simulators with moving point masses, it is a dynamic problem. It is, however, possible to frame this problem as a static one, by looking at time intervals so small that there is a negligible amount of change occurring. This would allow the use of static algorithms, but fail to exploit potential benefits of treating the problem as a dynamic one. By considering time scales that incorporate change and previous triangulations, an opportunity for great performance improvement appears. This paper focuses on this dynamic case where points are moving, and seeks to explore various heuristics for efficiently maintaining a Delaunay Triangulation.

This paper considers two dimensional triangulations, however many concepts seem applicable in higher dimensions. Focus has been concentrated on the algorithm and to maximise speedup while maintaining a satisfactory Delaunay triangulation. Perfect precision is not required, but achieving results that are visually indistinguishable from perfect results are required. From previous work, triangulation culling has yielded high speedups for a small drop in accuracy.

This work is motivated by achieving a real time physics simulation. This means that it is desired that the time for each frame is below 40 milliseconds. If such a low frame time is achieved, no perceptible stuttering or delay would be experienced. A user and the simulation could smoothly interact with each other. Allowing for activities like balancing an inverted pendulum.

1.1 Goals and Research Questions

Goal: Explore the capabilities and limits of a heuristic dynamic Delaunay triangulation.
Examine and define relationships between the classifiers and their influence to the validity and speed of the algorithm. Measure various performance values and view them in light of a possible real time usage scenario.

Research Question 1: Can the Voronoi diagram be used for a competitive method to detect invalid neighbourhoods?
After points have moved, determine which triangles have become invalid using the Voronoi diagram. This might prove to have advantages over circumcircle checks due to different mathematical expressions.

Research Question 2: What properties does a heuristic triangulation algorithm based on Bowyer-Watson have?
The Bowyer-Watson algorithm explicitly stores information closely related to the Voronoi diagram and could be more suited for the Voronoi method than the Guibas-Stolfi algorithm.

Research Question 3: What considerations must be made when seeking

to alter the Bowyer-Watson and Guibas-Stolfi algorithms, to incorporate the management of previous neighbourhood information?

These algorithms are static and therefore assumes no preexisting results. Their design incorporates this assumption, but may still contain useful concepts for a dynamic triangulation algorithm.

1.2 Contributions

This paper goes through relevant concepts to triangulation algorithms, constructing a dynamic algorithm from static algorithm and presenting test results.

- Created a Heuristic Dynamic Delaunay Triangulation algorithm where heuristics are easily replaceable.
- Performed investigations of classifiers and heuristics and how they influence the performance.
- Proposed a technique for replacing invalid regions in a large graph with a small valid graph.
- Discovered hard to notice issues related to maintaining a graph where invalid regions are not precisely detected.

Chapter 2

Background Theory

2.1 Physics Simulation

A Physics Simulation is an imitation of a physical system, run by computer software. A desired physical system could commonly involve rigid body dynamics, soft body dynamics and fluid dynamics. These cases involve time varying situations and causal relationships which require some form of collision detection. Applications of Physics simulation include product performance and failure testing, scientific research, computer graphics for entertainment like movies and video games, and real time physics responses in video games. Depending on the application, various feature and performance requirements apply. And a central resource used to meet these requirements is computational power.
In order to meet the requirements set, certain simplifications could be acceptable. Many creative techniques have been made so that the same results of a simulation could be achieved in less time. Real time physics for computer games must be performed very quickly, but is not required to be more accurate than is visually pleasing. Scientific physics simulation must be very precise. Precise enough to relate to the physical systems in the real world, and it is acceptable to wait days for the result. Both use as efficient algorithms as possible as well as managing a trade-off between speed and precision.

2.1.1 Collision detection

A common efficiency obstacle, is to determine which objects in the physics simulation are colliding or not. If there are few objects, the cost is low enough

to check all objects for intersections. This procedure has a computational cost of O(n^2) which will quickly become prohibitive at higher numbers of objects. By using neighbourhood techniques this cost could be drastically reduced. One such technique is based on the Delaunay Triangulation, that defines neighbourhoods. These neighbourhoods have topological properties that are well suited for collision detection. The computation of this Delaunay triangulation can be achieved with the, relatively low, computational cost of O($nlog(n)$). Delaunay triangulation can be used to build meshes in the Finite Element method, Finite Volume method and Boundary Element method used in physics simulations.

2.1.2 Considerations

A simulator computes the state of a system at a specific point in time, by considering one or more previous states. The time interval between states is called a frame, and is often constant and predefined. The constant time interval a simulation utilises is called time step. The time step plays a significant role in determining if a simulation is stable or not. If the simulation aims to model interactions over small time periods, the time step must be small enough to capture the necessary details of these interactions. Any interaction within a frame will be assumed through some model. However, the larger a frame is, the higher the chance the simulator will treat multiple interactions as one interaction. The result of multiple interactions is commonly highly dependant on the sequence of interactions. If this sequence is not captured, the simulation might loose its relevance to a real world system. This error can compound itself over time and could lead to the behaviour of the simulation to diverge from the desired behaviour. A central behaviour of a physical system is that the energy of the system does not increase. Enforcing this behaviour in a simulation is not straightforward, due to the possibility of small errors growing exponentially. These errors can be reduced by reducing the size of the time step. The size of the frames will then be lower and the chances of having multiple interactions within a single frame is reduced. After lowering the time step beyond a certain point, no frames will contain more than one interaction. And further reductions in the time step will not have an affect on the behaviour of the simulation. A simulation is deemed stable if the result does not deviate from the result of equivalent simulations at smaller time steps.

2.2 Delaunay Triangulation

Triangulations are undirected graphs where all edges take part in forming triangles. They can also be seen as subdivisions of a surface into triangles. The Delaunay triangulation has beneficial properties for neighbourhoods and neighbourhood searches.

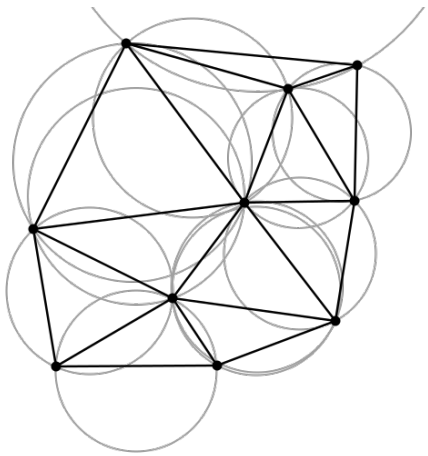
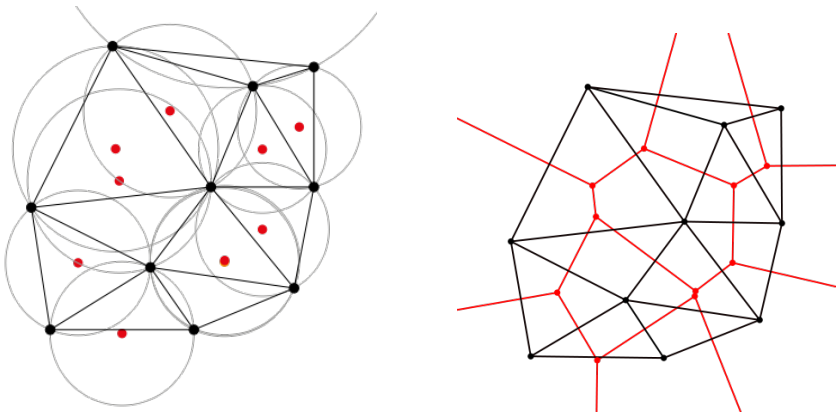


Figure 2.1: A Delaunay triangulation of ten points, with circumcircles shown.

2.2.1 Introduction

A Delaunay triangulation of a set of points, is a triangulation such that no point is inside the circumcircle of any triangle. The circumcircle of a triangle is a circle that intersects the three corners.
In figure 2.1, an example of these circumcircles and their corresponding triangles is shown. To start understanding this figure, consider the small triangle at the lower left. By tracing the circle that intersects the three corners, it is possible to see that no other point is within this circle. This is the case for all other triangles, but it might be more difficult to trace their corresponding circles.
In order to construct this triangulation, some choice of three points must be made and then their circumcircles must be checked if other points are contained. If there are other points contained in a circumcircle, then the three chosen points can not be determined to form a triangle. On the other hand, if no other points are within a circumcircle, then a triangle could formed. A triangulation is

complete when all possible triangles are formed. There is, broadly speaking, only one configuration of triangles for a given set of point. However there exist a situation where two or more configurations are tied and all are equally valid. This is when one or more points exactly lie on the periphery of a circumcircle. Then there would be more than three points that are equally distant from the centre of the circumcircle, and any choice of three points would form a valid Delaunay triangle.



(a) The centres of circumcircles shown. (b) Voronoi diagram superimposed.

Figure 2.2: A Delaunay triangulation and Voronoi diagram of ten points

2.2.2 Voronoi Diagram

A Voronoi diagram is a partitioning of a plane into regions. These regions consist of points closest to some seed points. This description results in a space partitioning, where the borders go through the centres of the circumcircles of the Delaunay Triangulation of the seed points. In figure 2.2a, the centres of circumcircles from figure 2.1 is shown in red. In 2.2b the borders between the centres are show in red. This forms regions that are closest to the points in figure 2.1. These points corresponds the the seed points.
Notice that all Voronoi borders are perpendicular to a Delaunay edge. The Voronoi diagram is a dual to the Delaunay triangulation. It contains sufficient information to define a corresponding Delaunay Triangulation, and vice versa.

2.2.3 Properties

A Delaunay Triangulation tends to produce triangles with large surface areas and small circumferences. Another way to say this is that the minimum angle in the set of triangles is maximised. Any given triangulation or graph may not exhibit this Delaunay property for every triangle. But if there are regions which do, then those regions are said to be locally Delaunay. If the entire triangulation or graph exhibits the Delaunay property, then the triangulation is globally Delaunay.

There are some tasks that become simplified by using Delaunay triangulation. Finding the k-nearest neighbours of a point is dramatically more efficient, if the Delaunay Triangulation or a Voronoi Diagram of the points are available. When viewing a Delaunay triangulation as a graph, it has many closely related concepts and graphs. The Gabriel graph, Relative neighbourhood graph and Euclidean Minimum Spanning Tree are all sub-graphs of the Delaunay triangulation. Interestingly, the Euclidean Minimum Spanning Tree is a sub-graph of the Relative neighbourhood graph, which in turn is a sub-graph of the Gabriel graph. [Ber+08]

2.2.4 Movement Invalidation

If a set of points have been processed to construct a Delaunay triangulation, their configuration of edges are said to be valid. If the points move, the edges may or may not be valid. The Voronoi diagram makes this easy to see. In figure 2.2b there are red lines that form Voronoi cells. The length of a line corresponds to how 'close' an edge is to being invalid. Close to the bottom right of figure 2.2b there is a very short line that corresponds to two triangles, with very similar circumcircles. For the four points in these two triangles, the two potential ways to construct triangles are both nearly equally 'good' triangulations.

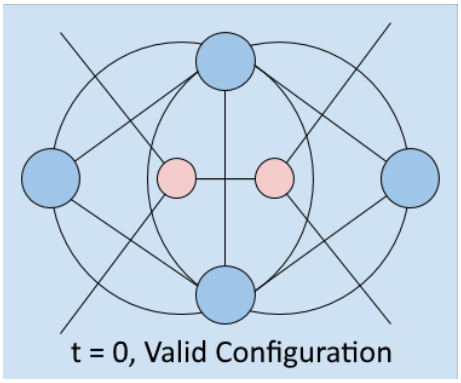
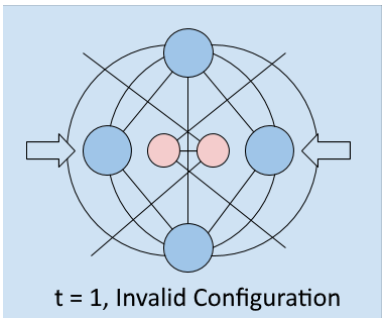
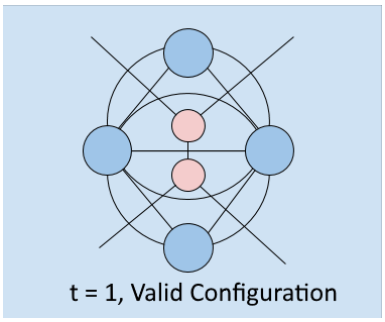


Figure 2.3: A Delaunay triangulation of four points, with circumcircles and circumcentres shown. The time is 0, and the edge configuration is valid.

By considering a basic constellation of 4 points, some valid and invalid configurations of edges, can be observed. The four points will always form two or three triangles, arguably with the exception when all points are on a line. By considering a pair of triangles, some properties could be observed. In figure 2.3, the circumcircles of the two triangles are shown, with their centres coloured in red. These are corners in the Voronoi diagram and have edges going perpendicular across the edges in the Delaunay Triangulation.



(a) Two points have moved closer to the centre. The edge configuration is invalid.



(b) After detecting the invalid configuration, a new valid one has been made.

Figure 2.4: Invalid and valid configuration of edges for four points at time = 1.

In figure 2.4a, the leftmost and rightmost points have moved closer to the centre. So much so that the previous configuration of edges are invalid. The left point is inside the circumcircle of the right triangle, and the right point is inside the circumcircle of the left triangle. Two pairs of Voronoi edges are crossing. In figure 2.4b, a valid configuration of edges has been constructed for the new position of the points. Coincidentally, this image corresponds to a scaling and rotation of the first image.

This scenario clarifies the behaviour of the Voronoi centres. The centres cross each other or 'collides' at the moment when the edge configuration loses its Delaunay property and becomes invalid. In other words, if points are moving, their edge configuration is valid until some Voronoi centres cross and equivalently, a connected circumcircle contains another point.

Chapter 3

Static Algorithms

There are three main paradigms for computing the Delaunay Triangulation: divide and conquer, sweep-line and incremental insertion. All can yield $O(n^2)$ expected time complexity. A divide and conquer algorithm and an incremental insertion algorithm has been explored and modified.

3.1 Bowyer-Watson

This is an incremental $O(n^2)$ algorithm, which explicitly stores triangles. This algorithm is relatively easy to implement and modify. It revolves around the concept of adding points to an existing Delaunay triangulation. After a point has been added, the Delaunay property is maintained. Adding data structures for storing the corresponding Voronoi diagram is easy. Adding support for removing points involves few considerations. [Bow81], [Wat81]

3.1.1 Initialisation

The algorithm needs an existing Delaunay triangulation to function. So before the first point is placed, a trivial triangulation is made in the form of a single temporary triangle. This triangle is large enough to cover all points that will be added. This triangle is called a super triangle.

3.1.2 Addition

All points are added one after the other. This is done by checking which circumcircles contain the new point that is to be added. The corresponding triangles of these circumcircles are removed. This leaves a 'hole' in the triangulation with a polygonal edge encircling the new point. New triangles are made, connecting the new point and each of the polygonal edges. It is ensured that the circumcircles of these new triangles will not contain any points. This could be understood by considering that the new circumcircles are smaller than the old circumcircles. The new circumcircles will contain some area that the old circumcircle did not contain, but this area will not contain any old points. This can be shown through contradiction. If there was a point in that area, they would form a triangle with a circumcircle that would contain the new point. Therefore, this point would be a part of the polygonal edge and never be susceptible to being contained by new circumcircles.

3.1.3 Finalisation

After all points have been processed, the super triangle and all connected triangles could be removed. What remains is a convex hull of a Delaunay triangulation.

3.1.4 Removal

A point can be removed by first noting which triangles it is a part of. Removing the point and these triangles will leave a 'hole' with a polygonal edge. Assuming the surrounding triangles are Delaunay, this 'hole' could be filled by new triangles. By proposing triangles formed by three consecutive points on the polygonal edge, a circumcircle containing no points will be found. This valid triangle reduces the size of the polygonal hole and this process repeats until the hole has been filled.

3.1.5 Applicability for Dynamic triangulation

The classified areas must be retriangulated which requires some considerations. Some retriangulation procedures operate on the assumption that the neighbouring points are Delaunay. This is most of the time not the case. So if a procedure like Bowyer-Watson is going to be used, it would have to sequentially add the invalid points one by one into a triangulation where no other point are invalid.

This could be done by making a second Delaunay triangulation with only the invalid points and their neighbours. And then merging the two triangulations together.
If the invalid particles form patches that are not connected, then they will not interfere with each other and could be computed at the same time, in separate Delaunay triangulations. This will introduce some level of parallelisation.

3.2 Guibas-Stolfi

A Divide and Conquer O(nlog(n)) algorithm, which is parallelisable. This algorithm is relatively hard to implement and modify, but fast. It uses concepts that are less intuitive and its parallel nature makes it harder to bug-test. It is however, widely considered to be the fastest algorithm. [GS85]

3.2.1 Initialise

Sort all points by x coordinate and then y coordinate. The horizontal sorting will be used in the Divide procedure and the vertical sorting will be used in the Merge procedure.

3.2.2 Divide

Recursively split the sorted points vertically, by x coordinate. This creates two sets of close to equal size for each step. This splitting will make a binary tree where the leaf nodes are ordered by x coordinate.

3.2.3 Conquer

When there is only three or two points in a set, make a triangle or edge. These are trivially Delaunay since there are not enough points to make an invalid Delaunay triangulation. This set is now ready to be merged with other sets.

3.2.4 Merge

Traversing the binary tree towards the root will reveal pairs of child nodes. These are ordered so that the left set contains points with lower x coordinates than in the right set. Merging these two sets appears very similar to 'zipping' up a jacket. The two sides are almost interleaved together. The two sets are next to each other, and this procedure makes valid triangles between the two

sets of points. This is done by first choosing the bottom point in each set and make a new edge between them. Then choose the next point in each set that is above the new edge and make a second edge. Circumcircle checks determine which edge is valid. After repeatedly doing this and reaching the top of the two sets, the two sets will be connected and Delaunay. This process continues through the binary tree, starting from the leaf nodes and consumes the tree. Finally there is only one node left, which represents a set of all points. Merged together from progressively larger sets of valid triangulations.

3.2.5 Applicability for Dynamic triangulation

Some procedures of Guibas-Stolfi operate on the assumption that the neighbouring points are Delauney, same as with Bowyer-Watson. The solution to this, from previous work was to create a second Delaunay triangulation of the invalid points and their neighbours and then merge the two triangulations together. This merging procedure follows a different approach than the procedure in Guibas-Stolfi.

Chapter 4

Proposed solution

A solution was previously made that utilised the Guibas-Stolfi algorithm. This will be slightly modified and compared to a new solution that uses the Bowyer-Watson algorithm.

4.1 Algorithm processes

Both algorithms will go through mainly the same steps: Classify Regions, Triangulate Regions, Merge Triangulations. The classifier will determine where resources should be spent. It will determine which points have moved to a position that invalidates their Delaunay neighbourhood. Then both algorithms will triangulate the invalid neighbourhoods. And finally, merge the now valid neighbourhoods with the global triangulation.

4.1.1 Initialisation

The points first need to be triangulated without having any prior neighbourhood information. In this situation, a static algorithm will be the perfect tool. Any Delaunay triangulation algorithm could be chosen, but implementation specifics may have to be taken into consideration. If Bowyer-Watson is chosen in a later step, it would assume the existence of a super triangle. Therefore in this step, it would be necessary to add special points that would serve as the corners of this super triangle.

The first frame will simply use this valid Delaunay triangulation. The next steps will come into play first in the second frame of the simulation.

4.1.2 Classify Regions

After the points have moved, their neighbourhoods may or may not be Delaunay. A classifier will compute each point or region, and attempt to accurately determine their validity. A number of different classifiers have been made to determine this with various degrees of accuracy and computational cost. In the case where no point moves, this will be the only operation that runs. And it will determine the amortised computational cost of this solution.

4.1.3 Triangulate Regions

This approach creates a separate triangulation that is ensured to be valid. The classifier returns a set of points which need to be triangulated. Since the classifier deems them to be invalid, the triangulation method can not assume that their existing neighbours are correct. The existing neighbour information is discarded and the points are simply processed by a static triangulation algorithm. The algorithm of choice should cooperate well with the algorithm chosen in the Initialisation step. The trivial choice would be to choose the same algorithm in both steps. This new triangulation encompasses the points that were classified as invalid which will provide the classified points with valid neighbourhoods.

To do this, a few sets of points are made that serve different purposes. The first set contains the classified points. The second set contains a layer of neighbours to the points in the classified set. The third set contains a single layer of neighbours to the points in the second set. The fourth set contains all the points in the first, second and third sets. The sets form individual layers like an onion. These sets are named the Classified set, Padded set, Border set and Triangulation set respectively. In figure 4.1, the sets are displayed as red for the Classified set, green for the Padded set and cyan for the Border set.

The Classified set is simply what the classifier returns. The Padded set serves to provide additional points for triangulation. The Border set serves to constrain the new neighbourhood of the Classified and Padded sets. The Triangulation set is what the static algorithm takes account of when triangulating the sub-graph.

The Border set constrains the new neighbourhood of the Classified and Padded sets. This is important to avoid separate clusters of classified points to influence the triangulation process of the other. The Border set works in

essence as a convex hull for each cluster of classified points. By using the Border set in this way, it is possible to triangulate all classified points in a single process.

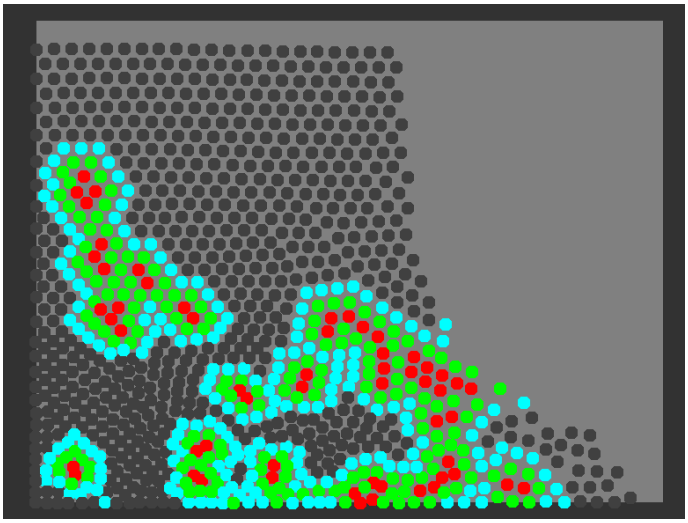


Figure 4.1: The red circles represents classified points with one layer of padding points in green, and a layer of border points in cyan.

4.1.4 Merge Triangulations

After producing a second triangulation, the neighbourhood information must replace the corresponding information in the global triangulation. The second triangulation could be seen as a sub-graph of the valid Delaunay graph of all points. The current global graph contains some invalid neighbourhoods where the sub-graph does not. By substituting with the neighbourhoods in the sub-graph, the global graph will have achieved a region with valid neighbourhoods.

This is only straightforward for the neighbourhoods in the Classified and Padded sets. The procedure for the points in the Border set is more involved. The Border set is the merging line between the global graph and the sub-graph. These points acts as convex hulls for specific clusters of classified points. In figure 4.1 a small cluster can be seen in the bottom right and multiple clusters in the left and bottom middle. The neighbours that the Border points had in the global graph, must still be used. They can keep the new neighbours if they

are in the Padded set or the Classified set, but all other neighbours in this sub-graph are discarded. In other words, the Border points keep their neighbours from before the Triangulation step, except the ones in the Padded set. After the Triangulation step, the new neighbours that are in the Padded set or in the Classified set will be kept.

After this transference of neighbour information, the global graph has achieved the valid neighbourhoods in the sub-graph. The Border points are essential to this process, as they mark the locations where both graphs are assumed to have valid neighbourhoods and are equal. In the sub-graph, they also separate edges going from one cluster to another. Which makes it possible to triangulate all classified particles at the same time. The Border set also manages to behave well with the convex hull of the global graph.

Issues

Merging the triangulations is not a guaranteed process. The classifier might fail to classify an invalid point, which could interfere with the function of the Border set. The Border points mark the locations where both graphs are assumed to have valid neighbourhoods and are equal. The sub-graph is valid, but if the Border points in the global are not valid, then the graphs are not equal at those points. Then it would be unknown which edges are the correct edges to bind the two graphs together with. It may be the case that the correct edge isn't present and a new triangulation is required. This problem could result in an edge missing from the global triangulation.

In the case of an edge missing from the triangulation, there would still be many opportunities for the classifier to classify some of its neighbours. If a point two steps away is classified as invalid, then the points in this edge will be included in a new triangulation. This is not guaranteed and the missing edge could linger in the global graph. However, this problem is mitigated by increasing the precision of the classifier. The precision settings that give high validity values appear to stop this issue from significantly influencing the simulation.

4.2 Classifiers

Both static and dynamic triangulation algorithms needs to determine if a specific triangle or neighbourhood is valid. In the case of the Bowyer-Watson and Guibas-Stolfi algorithms this classification is done by checking circumcircles for intersecting points. In the case of a dynamic algorithm, this could be done by

checking circumcircles too, but only a modest speedup is expected. This is due to the relatively large role this procedure plays in a triangulation algorithm. However a heuristic classifier could approximate the circumcircle classification at a very low computational cost.

4.2.1 Classifier Performance Considerations

The performance of a classifier could be consider in terms of a Confusion Matrix. If the classifier achieves no False Negatives, all invalid neighbourhoods have been detected. If no False Positives are achieved, no computational resources are wasted on triangulation neighbourhoods that are already valid. The classifier hands over classified points to be triangulated. The amount of False Negatives of a classifier says how precise the global triangulation will be, while the amount of False Positives states how efficiently the labour of the Triangulate Regions step will be.

The relationship between a classifier and a triangulation algorithm is similar to a driver and a vehicle. The perfect driver will allow the vehicle to perform most efficiently. However the classifier labours as well as provide labour to the triangulation algorithm. So their combined performance is a sum of the cost of the classifier and the cost of triangulating True and False Positives. This allows the comparison of classifiers by looking at their cost and their ratio of True and False Positives.

The following equations serves to indicate the main contributors to the Total Computational Cost of this approach.

$$TCC(t) = CC(p) + TC(PC(t))$$
$$PC(t) = PPV(TPM(t)) + FDR(TPM(t))$$
$$TCC(t) = CC(p) + TC(PPV(TPM(t)) + FDR(TPM(t)))$$

Where TCC(t): Total Computational Cost of frame t,
p: number of points in the scenario,
CC(p): Classifier Cost function of a graph with p points,
PC(t): Positive Classifications at frame t,
TC(PC(t)): Triangulation Cost of Positive Classifications at frame t,
TPM(t): Turbulent Point Motion at frame t,
PPV(SSPT(t)): Positive Predictive Value function and
FDR(SSPT(t)): False Discovery rate function.

The Total Computational Cost is determined by a sum of the Classifier Cost and the Triangulation Cost which is dependant on the amount of Positive Classifications at frame t. The Classifier Cost function, CC(p), is assumed to be linear and monotonically increase in the number of points. The Triangulation Cost function, TC(PC(t)), is assumed to be O(n*log(n)) on the number of Positive Classified points. The choice of classifier affects the Total Computational Cost, through the Classifier Cost, the Positive Predictive Value and False Discovery rate functions. The classifiers have parameters to reduce or increase their sensitivity. These parameters are assumed to be tuned so the classifier manages to predict most True Positive conditions, which will produce satisfactory triangulations.

The ranking of classifiers depends on the number of points and the Turbulent Motion in a scenario. In scenarios with many points, but low turbulence, a computationally cheap classifier would probably provide the least Total Computational Cost. In scenarios with few points, but high turbulence, a relatively more expensive classifier could be less demanding due to lower numbers of False Positives. It would be unclear which classifier would be best in situations with few points and low turbulence or many points and high turbulence. Determining this ranking of classifiers appears so complex that simply running tests would be the better course of action.

4.2.2 Point distances classification

According to Proposition 1 in [CD08], *"Given a Delaunay triangulation τ, if its vertices move arbitrarily yet inside their safe regions, then τ remains Delaunay."*

A vertex has a region of space it can be in, without invalidating its current configuration of edges. This region is defined by the set of points where the circumcircles of neighbouring triangles does not contain any vertices. This classifier is based on using edge lengths to estimate this region.

This region has a complicated shape. However, this area could be approximated by an ensemble of radii emanating from the neighbour points. The radii in figure 4.2 are defined as an inner and outer radius that are $x\%$ shorter and $x\%$ longer than the edge length at triangulation. These radii defines a region where points are assumed to have valid Delaunay neighbours. In other words, if the distance between neighbours is sufficiently different from when they were triangulated, then the neighbours are classified as invalid. The x value is a parameter for the heuristic that adjusts the tolerance for motion. Points will be triangulated more often proportional to their motion and inversely proportional

to the parameter x. By setting x to zero will effectively classify all vertices as invalid and the algorithm will behave as a static one.

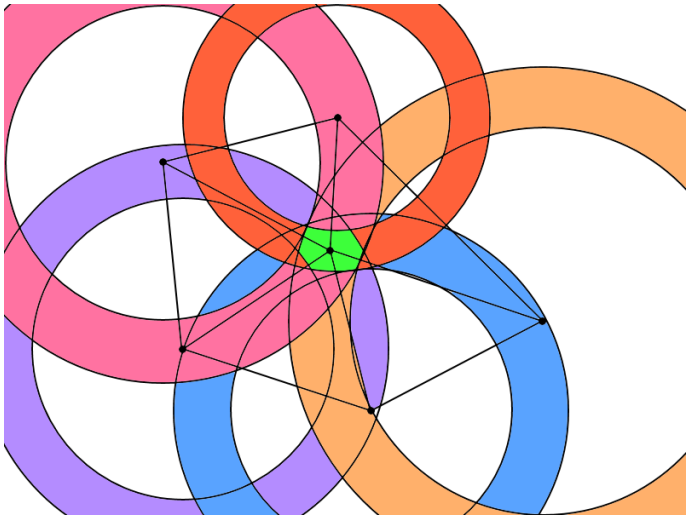


Figure 4.2: The point in the centre lies in a green region defined by circular bands emanating from neighbour points. This green region estimates the positions the middle point can move to without breaking the Delaunay property.

The heuristic works by considering one edge at a time, which corresponds to a single radius of the radii ensemble. If the length of the edge transgress the bounds defined by the x parameter, the two points in the edge will be classified. After all edges around a point have been processed, their combined bounds will approximate the safe region. The time before a classification of an edge occurs can be expressed as follows:

$$Tmin(x, e) = \begin{cases} \dfrac{e.L - e.Lt * x}{e.S}, & \text{if } e.L \geq e.Lt * x \\ 0, & \text{otherwise} \end{cases}$$

$$Tmax(x, e) = \begin{cases} \dfrac{e.Lt * (2 - x) - e.L}{e.S}, & \text{if } e.L \leq e.Lt * (2 - x) \\ 0, & \text{otherwise} \end{cases}$$

$$TT(x,e) = \begin{cases} Min(Tmin(x,e), Tmax(x,e)), & \text{if } Tmin(x,e) \geq 0, Tmax(x,e) \geq 0 \\ Tmin(x,e), & \text{if } Tmin(x,e) \geq 0, Tmax(x,e) < 0 \\ Tmax(x,e), & \text{if } Tmin(x,e) < 0, Tmax(x,e) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Tmin(x, e): Time to minimum length limit of edge e using x,
Tmax(x, e): Time to maximum length limit of edge e using x,
TT(x, e): Time to Triangulation of edge e using x,
x: parameter for the tolerance to motion,
e.L: The Length of edge e,
e.Lt: The Length of edge e after last triangulation,
e.S: The Speed of of which the edge is extending.

The distances between neighbouring points are routinely computed in a physics simulator, which could efficiently be used for classification. This is an optimisation uniquely available for this heuristic, dependant on the application environment.

4.2.3 Random classification

This classifier simply classifies points randomly. A parameter sets the probability for points to be classified as invalid. This probability determines the ratio of valid to invalid points. This ratio is roughly constant over time, regardless of the motion of points. This makes the computational cost stay at a constant value, but does not exploit potential performance gains due to favourable motion.

So if the motion is severe enough, the relevant points might not be correctly classified as invalid before multiple frames have passed, depending on the probability value. The Delaunay validity of all points is expected to drop when points move severely. After they have stopped moving, the points will slowly approach being valid. The computational costs would remain constant.

4.2.4 Inflation aware classification

The Point Distance classifier can be augmented to account for inflation like motions. Consider points that are all moving away from each other like in expanding space or an inflating balloon. The triangulation would behave as if it was scaled up in size and would not change. Such a behaviour would have a constant value of inflation throughout the triangulation. The Point Distance

classifier would continue to return classified points in this case, even if the triangulation is valid.

This could be avoided by taking account of this inflation and classify points, only when an edge increases more than the surrounding edges increases. In other words, classify regions where inflation increases or decreases. This would require more computation to determine, but the increased precision might lower triangulation costs sufficiently.

4.2.5 Circumcircle classification

This is a relatively expensive method, that correctly classifies points in a triangle, if the corresponding circumcircle contains a point. This classifier is not expected to provide a higher speedup than heuristic classifiers, but could be used in comparisons between heuristic classifiers and static algorithms. It is expected to provide a correct validity, so if a heuristic classifier provides a worse speedup than this classifier, then it is strictly dominated and can be discarded. There is no adjustment parameter for this classifier.

4.2.6 Voronoi Circumcenter classification

The Voronoi diagram describes the relationship between points in a way that allows for a succinct expression of valid and invalid positions of circumcircles in a triangulation. Two triangles become invalid when their circumcenters collide. At that point, the circumcircles will contain more than three points. This is shown in figure 2.4a. To detect this, one can iterate through all edges and consider the two triangles it is part of. After triangulation, compute the length of the orthogonal projection of the two triangles' circumcentres on to the edge. The sum of these lengths may either be positive or negative. If the sign changes, the circumcenters have collided and the Voronoi diagram has become invalid. This method of classification is precise, as long as the positions of circumcenters are precise. If the circumcenters are known, the function for determining if they have crossed each other is very short.

$$\overrightarrow{c1}(e) = e.triangle1.circumcenter - e.p1$$
$$\overrightarrow{c2}(e) = e.triangle2.circumcenter - e.p1$$
$$\overrightarrow{e2}(e) = e.p2 - e.p1$$
$$CD(e) = \frac{det(\overrightarrow{c1}(e), \overrightarrow{e2}(e)) - det(\overrightarrow{c2}(e), \overrightarrow{e2}(e))}{|\overrightarrow{e2}(e)|}$$

$$CDS(e) = (\overrightarrow{c1}.x - \overrightarrow{c2}.x) * \overrightarrow{e2}.y - (\overrightarrow{c1}.y - \overrightarrow{c2}.y) * \overrightarrow{e2}.x < 0$$

$$CDS(e) = (\overrightarrow{c1}.x - \overrightarrow{c2}.x) * \overrightarrow{e2}.y < (\overrightarrow{c1}.y - \overrightarrow{c2}.y) * \overrightarrow{e2}.x$$

$\overrightarrow{c1}$(e): The vector from the 'first' point in edge e to the circumcenter of the 'first' neighbouring triangle of edge e,
$\overrightarrow{c2}$(e): The vector from the 'first' point in edge e to the circumcenter of the 'second' neighbouring triangle of edge e,
$\overrightarrow{e2}$(e): The vector from the 'first' to the 'second' point in edge e,
CD(e): The signed distance between circumcenters,
CDS(e): The Circumcenter Distance Sign function.

Chapter 5

Experiments and Results

Determining the ranking of classifiers could be straightforward. However, if they are very closely matched, it may become necessary to compare them on a scenario by scenario basis. The scenarios should then be constructed to differentiate the classifiers by playing to the strengths of some of them.

5.1 Experimental Plan

5.1.1 Performance Metrics

In order to provide a reasonable ranking of classifiers, their most salient performance features must be quantified. One motivating factor for this work was to provide a triangulation algorithm capable of high performance in a real time situation. It is then desirable that a performance metric is capable of reasonably capture the factors which are most important to this situation. Some factors can be extracted from this statement. The real time situation sets an upper bound on the computation time per frame. The high performance statement requests that a lot of usable results can be produced per time. And the variable that pays the price for these desired properties is the Delaunay validity. The requirement that the resulting triangulation is Delaunay has been relaxed to a point where the triangulation is only required to be good enough of a neighbour configuration, that a physics simulation will continue to be stable.

Number of Points

This metric attempts to incorporate properties relevant to real time applications and still remain be easy to communicate. The performance can be measured by the number of points the algorithm can process while still maintaining the desired precision and speed. The desired speed is defined as each frame time must be below 40 milliseconds. The desired precision is that the behaviour of the simulation is not perceptibly different from the behaviour of a precise simulation. This definition includes two performance properties of the algorithm. Frame time and precision has been combined into one metric, the Number of Points. This metric is most relevant when considering real time applications, but may otherwise have limited usefulness.

Delaunay Validity

Delaunay Validity states how close the estimated triangulation is to having the same edges as a Delaunay triangulation of the same points. The closeness is quantified by Matthews Correlation Coefficient. This can be done by treating the estimated triangulation as a statistical classification of correct and incorrect neighbours. A confusion matrix is then available and values for correlation can be computed. The values in Matthews Correlation Coefficient ranges from 1, fully correct classification, 0, random classification, -1 fully incorrect classification. The estimated triangulation is provided by the dynamic algorithm, while the Delaunay triangulation is provided by a static algorithm.

Speedup

This is a relation between the duration of the static and dynamic algorithms. The speedup is quantified by measuring the time of both algorithms on the same scenario, and taking the ratio of their duration. Ratios above 1 indicates that the dynamic algorithm requires less time. Ratios below 1, indicates that the static algorithm requires less time. The time measured does not include other processed from the physics algorithm.

Computers have varying computing power and will complete running the algorithms after different durations. This means that the results of the experiments will vary across computers. This effect of varying computing power is attempted to be mitigated by computing a ratio of two durations. The two durations would be influenced by the same hardware and when compared to each other, will counteract each other. They would not cancel each other out

perfectly, but it is proposed that they would come close. This would make the results more comparable across different hardware, than just the duration alone.

5.1.2 Scenarios

A collection of scenarios have been made to differentiate the performances of the classifiers. They display different types of point motion that some classifiers may be better suited to capture. The position of the points are initialised with a very small randomisation. This prevents identical runs of the algorithm. Every classifier, except the Random classifier, are deterministic, and will provide the same output from the same input. There might be random events during the simulation of the scenarios, so a collection of 10 trials will be run for each combination of classifier and scenario. The results from those 10 trials will be averaged and presented as containing negligible amounts of noise caused by processes unrelated to the simulation.

When measuring Delaunay validity and Speedup, the scenarios will be run simultaneously with a dynamic and static triangulation algorithm. In this case, the scenarios will be initialised with a fixed number of points that are stated below. When measuring for the maximum Number of Points, only the dynamic triangulation algorithm will be run. The scenarios support being initialised with different Number of Points. This will be the main parameter that determines computational intensity and subsequently the maximum Number of Points a classifier can support. The Number of Points will be increased for each successive run of the scenario until, the frame duration constraint is met.

Dam Break scenario

A dam break scenario is a very simplified situation of a containment failure in a hydropower dam. It involves a block of water spilling into a square container and settling into calm waters. The points in the scenario represents water particles of a constant mass and size. This scenario is meant to give results that are representative for a common simulation situation.

This scenario will be initialised with 1008 points for the Validity and Speedup metrics.

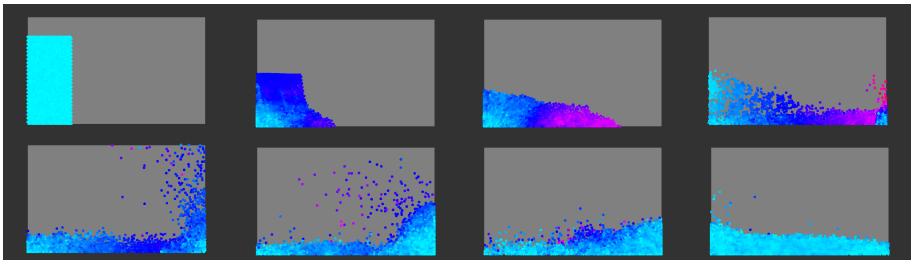


Figure 5.1: The Dam Break scenario starts with a block of water which, due to gravity, will spill out into the container and eventually settle. The colours represent the speed of each point mass. Cyan, blue and pink represents no speed, slow speed and fast speed respectively. Time progresses from top left going row by row.

Spinning scenario

This scenario is meant to specifically test a heuristic's ability to classify points on the hull of collections of points. Points on hulls have vastly differing distances to neighbouring points. Some neighbours lie inside the hull and are very close, while some neighbours are far away, probably on other hulls. Heuristics might be poorly equipped to handle this wide range of distances. Two hard squares of the same size are placed adjacent to each other, that both slowly rotates clockwise.

This scenario will be initialised with 800 points for the Validity and Speedup metrics.

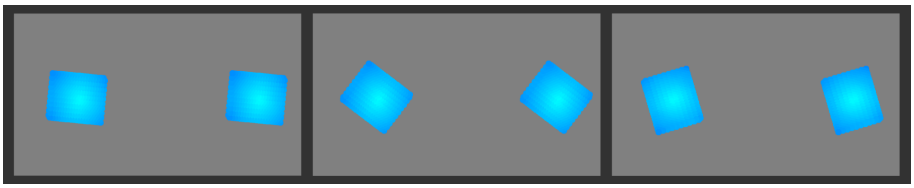


Figure 5.2: The Spinning scenario is presented as a slideshow of images going row by row, starting from top left. The colours represent the speed of each point mass. Cyan, blue and pink represents no speed, slow speed and fast speed respectively.

Explosion scenario

Two hard squares move towards each other with a high velocity. The impact shatters them completely, spraying their pieces in all directions. This scenario is meant to test the classifiers on a computational intense situation. Specifically the case where many point masses are colliding and intermingling, but also moving away from each other. An explosion will quickly develop into a state where points maintain their neighbours, but move away from them. This can be observed by considering that the explosion will cause the points to move away at high, but varying speeds. After some time, the higher speed points will overtake the lower speed points and eventually the points will become sorted by speed. They will also increase their distance to each other and their relative motion will diminish as a result. The situation is similar to cosmic inflation.

This scenario will be initialised with 968 points for the Validity and Speedup metrics.

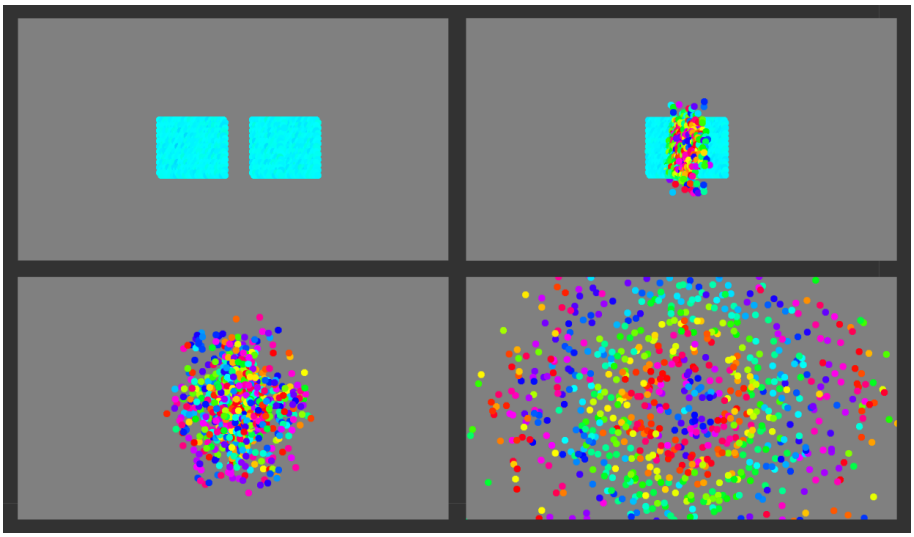


Figure 5.3: The Explosion scenario is presented as a slideshow of images going row by row, starting from top left. The colours represent the speed of each point mass. Cyan, blue and pink represents no speed, slow speed and fast speed respectively.

Static scenario

The scenario features a hard square of point masses which do not perceptibly move. There will be some minuscule shivering but is expected to not be sufficient to trigger classifications from the heuristics. This scenario is meant to show the minimum computational demands of a classifier. If no points are classified as invalid, then there will be no triangulations and all of the computational effort is spent on the classifier.

This scenario will be initialised with 1024 points for the Validity and Speedup metrics.

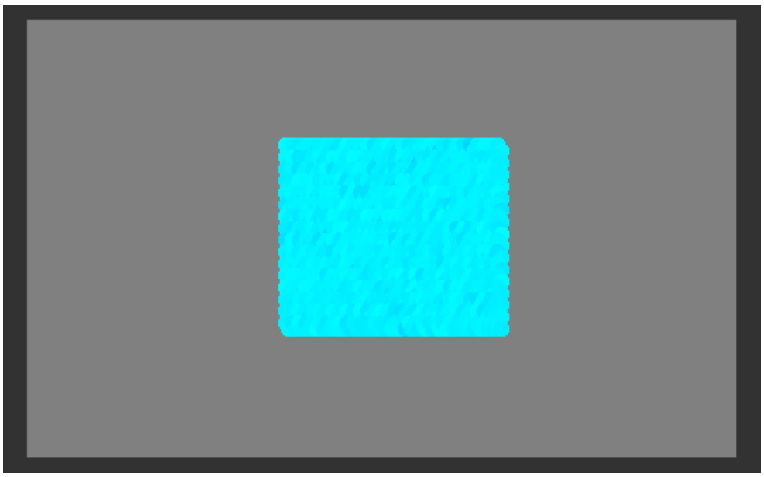


Figure 5.4: The Static scenario is presented as a slideshow of images going row by row, starting from top left. The colours represent the speed of each point mass. Cyan, blue and pink represents no speed, slow speed and fast speed respectively.

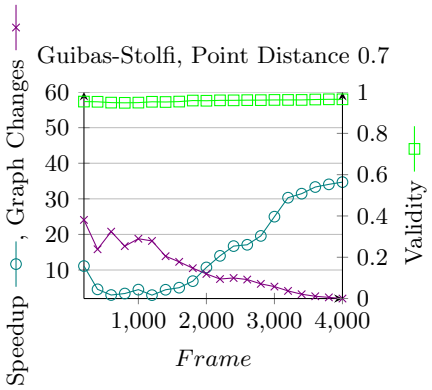
5.2 Experimental Setup

For each configuration of classifier, scenario, algorithm, performance metric and parameter settings the tests will be performed multiple times and the average data of those tests will be presented. These tests will be simulated in one continuous session to mitigate any slowdowns caused by initialisation and random events. The scenarios are initialised with an existing Delaunay triangulation.

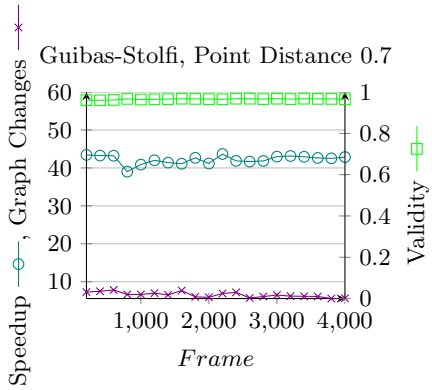
5.3 Experimental Results

Results are separated into classifiers and then into scenarios. Each scenario has a characteristic motion of points and corresponding changes to the graph. The measurements come in Delaunay Validity values over time, Speedup values over time and maximum Number of Points.

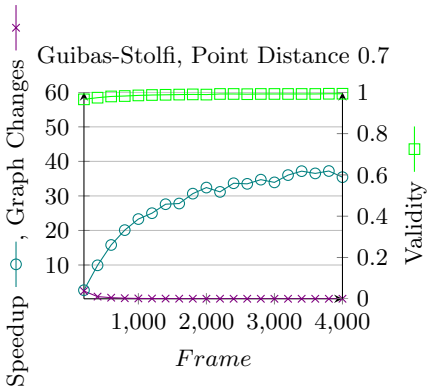
5.3.1 Point distance classifier



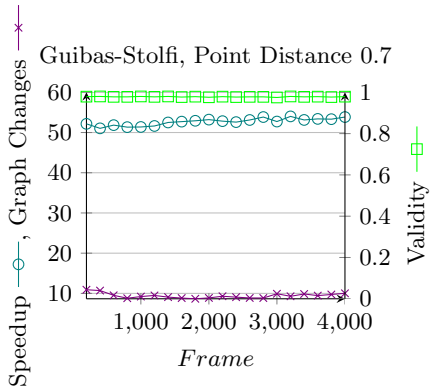
(a) Performance achieved on Dam Break



(b) Performance achieved on Spinning



(c) Performance achieved on Explosion

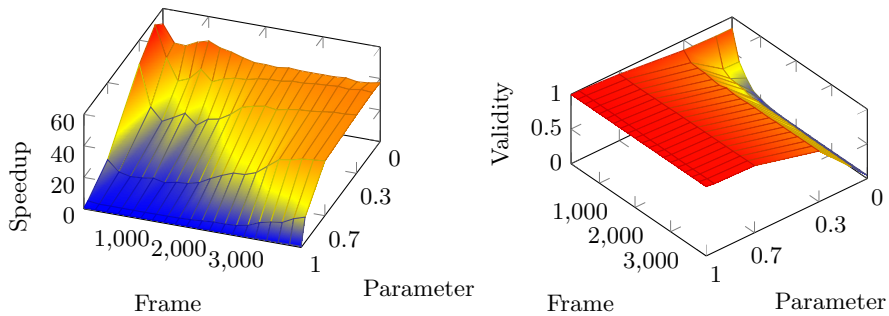


(d) Performance achieved on Explosion

Figure 5.5: The Point Distance classifier with a parameter value of 0.7 achieved a validity value very close to one on all four scenarios. The Speedup seems to approach somewhere in the range [35, 55].

For the Dam Break scenario, the speedup curve seems to be inversely related to the curve for the changes in the graph. The speedup curves for the Spinning and Static scenarios are roughly constant at a value close to 40 and 50 respectively.

The speedup curve for the Explosion scenario increases a lot in the beginning of the simulation and then converges to a value close to 40. The validity for all scenarios are very close to one, throughout the simulations.

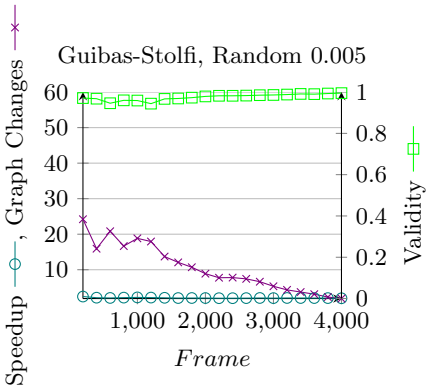


(a) Speedup achieved on Dam Break (b) Validity achieved on Dam Break

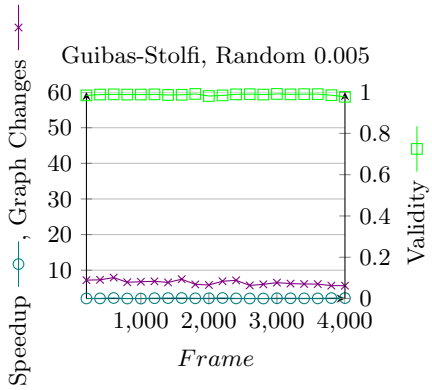
Figure 5.6: Performance of the Point Distance classifier on the Dam Break scenario. 5.6a, the speedup is most reduced at the area around frame 1000 and parameter value 1. 5.6b, the motion of points does not influence the validity of the algorithm, but the parameter setting does. Giving high validity values above parameter value 0.3.

The Dam Break scenario underwent more tests for different parameter values of the Point Distance heuristics. The data from these tests were combined into 3D plots. The Speedup is low around frame 1000 and on higher parameter settings. Parameter values between 0.3 and 0.7 yields both high speedup and high validity. Values above 0.7 yields higher validity, but seeing as the speedup declines fast, gives diminishing returns. Values below 0.3 yields higher speedup, but seeing as the validity declines fast, gives diminishing returns as well. The range [0.3, 0.7] is where the classifier is most efficient.

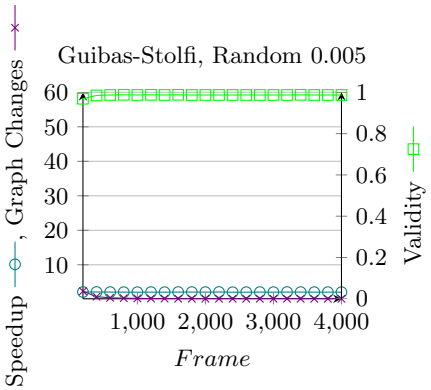
5.3.2 Random classifier



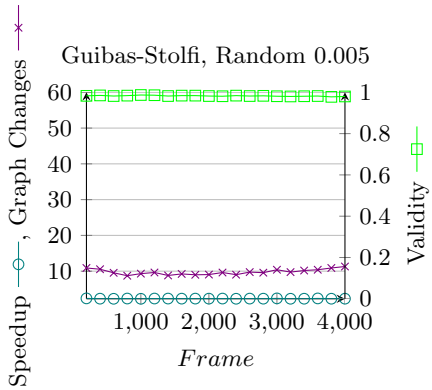
(a) Performance achieved on Dam Break



(b) Performance achieved on Spinning

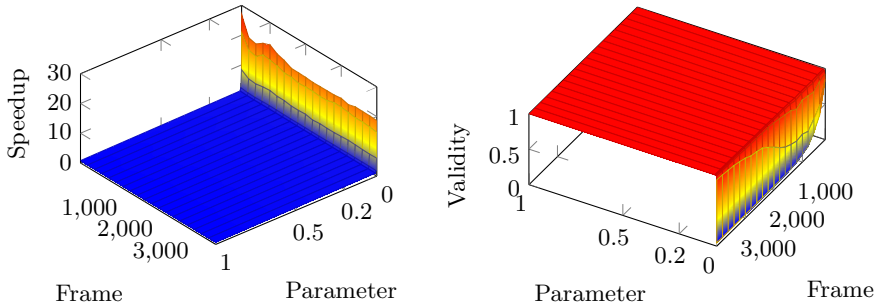


(c) Performance achieved on Explosion



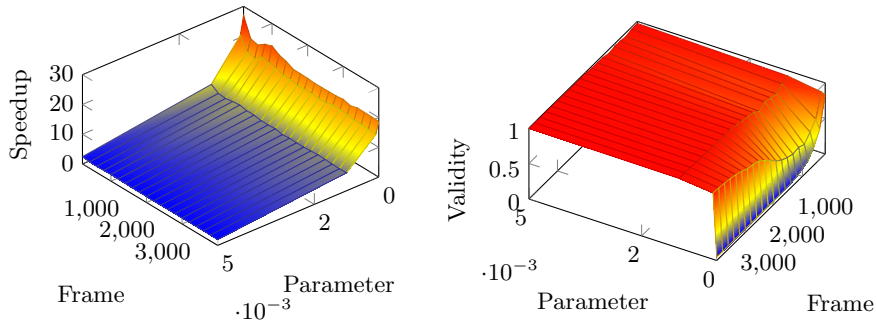
(d) Performance achieved on Explosion

Figure 5.7: The performance of the Random classifier appears mostly constant over all scenarios. The Validity on the four scenarios is very close to one, with some minor deviations around areas where the graph is changing a lot. The Speedup seems to be a constant close to one.



(a) Speedup achieved on Dam Break (b) Validity achieved on Dam Break

Figure 5.8: Performance of the Random classifier on the Dam Break scenario. This classifier appears to classify all points as invalid unless very low parameter values are set.

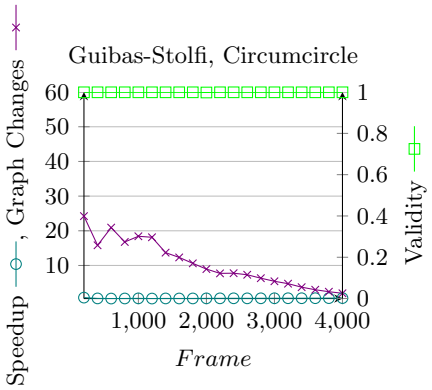


(a) Speedup achieved on Dam Break (b) Validity achieved on Dam Break

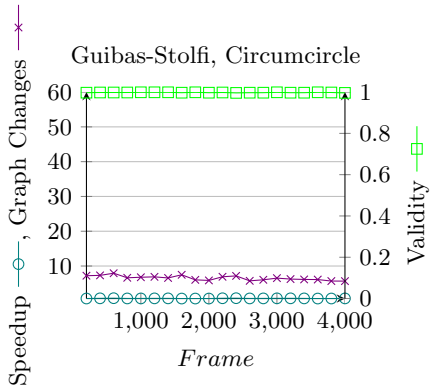
Figure 5.9: Performance of the Random classifier on the Dam Break scenario. This is the same data as in figure 5.8, but disregards parameter values above 0.005 for easier reading.

The Speedup is mostly constant for each parameter value, but varies most at parameter value zero and where there are a lot of graph changes in the simulation. The Validity appears to drop when there is a lot of graph changes and seems to slowly approach one afterwards. The rate at which it approaches one increases with the parameter value.

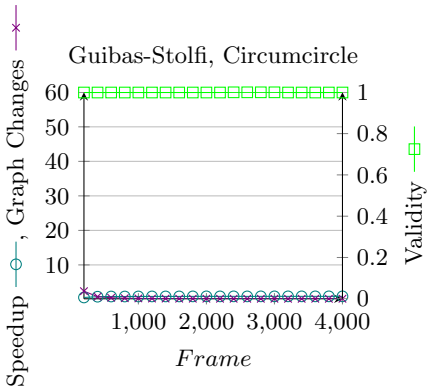
5.3.3 Circumcircle classifier



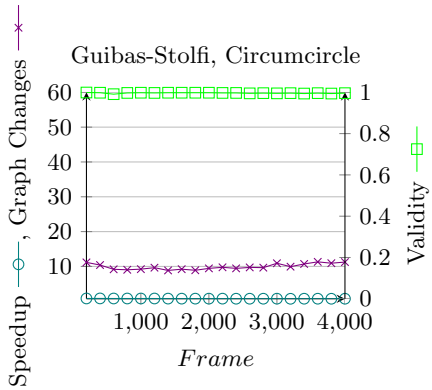
(a) Performance achieved on Dam Break



(b) Performance achieved on Spinning



(c) Performance achieved on Explosion

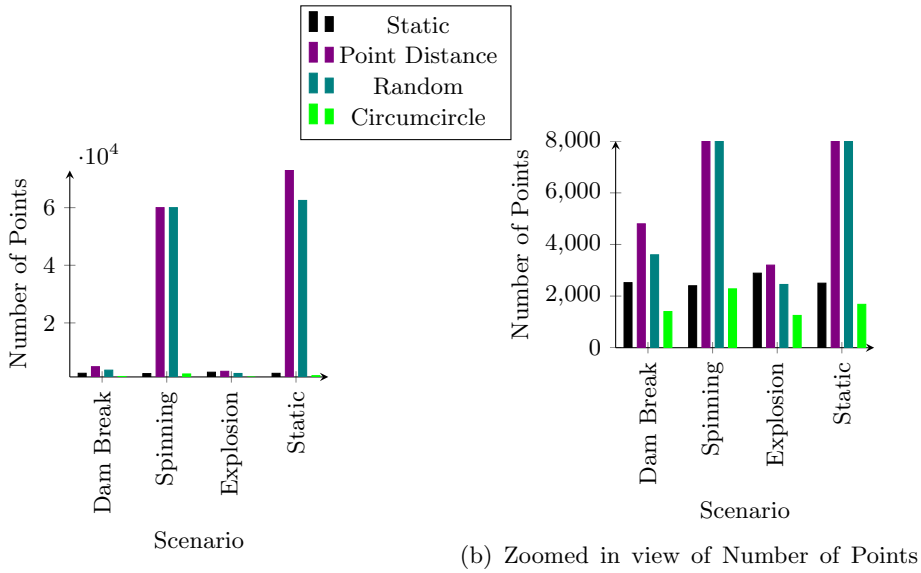


(d) Performance achieved on Explosion

Figure 5.10: The Validity of the Circumcircle classifier on the four scenarios is very close to one. The Speedup seems to be a constant close to one.

This classifier achieves a validity value of almost always 1, for all scenarios. However its speedup is almost always in the range [0.5, 0.9].

5.3.4 Maximum number of points



(a) Number of Points achieved

(b) Zoomed in view of Number of Points achieved

Figure 5.11: The Classifiers achieved a wide range of numbers of points. This varies according to their classifier cost and to the characteristics of the scenarios. The Point Distance and Random classifiers achieved, by a large margin, the highest number of points on the Spinning and Static scenarios. All other cases had values that were relatively clustered together.

Chapter 6

Evaluation and Discussion

6.1 Evaluation

6.1.1 Point Distance classifier

The Point Distance classifier displayed relatively smooth curves for both Delaunay validity and for Speedup. Maximum and minimum Validity values stayed very close for precision settings down to 0.25. While Speedup values varied more.

Different regions of precision settings appear from looking at the Point Distance classifier. At precision values lower than 0.3, the algorithm starts to fail to produce consistent validity values. Precision values higher than 0.3 and lower than 0.9, seem to provide high validity and high Speedup values. Precision values higher than 0.9 seem to provide diminishing returns.

6.1.2 Random classifier

The Random classifier displayed an abrupt bend in both curves at parameter value 0.0025. The parameter value corresponds to the probability for single points to be classified and triangulated. However, during a triangulation, the surrounding points will also be triangulated. When many enough of these triangulated neighbourhoods overlap, any increase in the parameter will hardly alter the number of points that gets triangulated each frame. The relationship between the parameter and the number of particles triangulated is non-linear.

Overall this classifier provides negligible speedup until large drops in Delaunay validity occurs.

6.2 Discussion

6.2.1 Measurement errors

The Delaunay Validity score involves many computational operations. This might introduce rounding errors that pushes the scores away from tidy numbers. The circumcircle classifier should theoretically yield a Validity score of 1 always, but a value of 0.999 was commonly reported.

About half of the measurements for the triangulation capability for the number of points, were averaged over less than 10 trials. This means that noise could have influenced the results more than the other measurements. This reduced precision in the tests were caused by the increased time cost of simulating a large number of points. There are procedures in the physics simulation that have a higher computational complexity than the triangulation algorithms, which will determine the cost of performing tests when the number of points is high enough. The time measurements did not encompass the physics procedures in the simulation and did not show signs of misrepresenting the triangulation duration.

6.2.2 Padded set

The proposed algorithm uses a number of sets for the triangulation step. The Padded set and the Border set surround the Classified set like layers in an onion. In chapter 4 it was stated that the Padded set is composed of neighbours one step away from the classified points. However, this could be increased to n steps if it is required. The Padded set takes a parameter n, for the number of layers. This is due to the nature of some classifiers and to the expected speeds in the scenario. All scenarios in this paper do not require a n higher than 1. Therefore it remained at 1 in conjunction with the statement in chapter 4. If the speed of a point is higher than one neighbourhood each frame, then it would be required to have an n of more than one. This is because the triangulation neighbourhood must envelop the position of a speeding point. If n, were too low, the speeding point would outrun the triangulation area. The parameter n, could be seen as analogous to the speed of sound through different mediums. However in this case it would be the speed of triangulation through different neighbourhood sizes. If points move faster than this speed, they would continuously have outdated

neighbourhoods. If they had no method of reducing their speed, they would be separated from other points indefinitely.

6.2.3 Goal clarifications

The goal of keeping the frame time below 40 milliseconds is an arbitrary restriction. What applies as a 'real time interaction' is a subjective quality and may have more dependencies. It does however, serve to clearly describe a limit where a performance metric could be defined from. This limits which regions desirable values lie within and focuses work efforts.
The goal also requests visually indistinguishable results. By visually indistinguishable results, it is meant to classify results as *convergent* or *divergent*. This is not a strict term, but attempts to describe a consistency of behaviour. Meaning that the simulation resolves situations in the same way as a simulation with higher precision. It is allowed that a simulation drifts due to the system being sensitive to initial conditions. However, at some point the precision can not go any lower without the simulation 'exploding'. Examples include macroscopic point masses tunnelling through other point masses or resting blocks of matter tear themselves into small pieces. These behaviours may be due to delays in receiving up to date neighbourhood information or spring instabilities. Determining if a simulation is *divergent* is not harder than determining if something stable has become volatile.

6.2.4 Dam Break initialisation

When testing for the maximum number of points on the Dam Break scenario, some issues had to be handled. The area where points could move within was increased together with the number of points. The ratios between the height and width of both the points and the area were kept roughly constant. This means that the points, that formed the column of water, would have an increasingly higher centre of mass. This would translate into a higher vertical velocity after a point had fallen some distance. As the height increased, so did the pressures at the bottom of the water column. At some point the pressures increased beyond the capabilities of the physics algorithm that led to visibly altered behaviour of the simulation. This change in behaviour was judged to be caused, or at least influenced, by transgressing the limits of the physics algorithm and not solely due to errors in the classifier. By releasing the constraint of keeping the width to height ratio fixed, the height could be kept below a certain level and the characteristic change in behaviour dispersed.

This adaptation could arguably introduce problems with the data in form of reduced comparability between classifiers. However it comes down to the interpretation of making a scenario scalable. If scaling a scenario in size involves increasing the sizes of all widths and heights, this would square areas and the mass of the collection of points. If a collection of points were doubled in width and height, the mass would be increased with a factor of four and the height of the centre of mass would be doubled. This would also increase the starting energy of the system with a factor of eight.

An alternative method for scaling a scenario comes in the form of not increasing distances, but reducing the mass and radius of points. Then additional points could be introduced to fill the new space and lost mass. The mass and start energy of the system would remain the same over a varying number of points. This however, has an issue connected to the mechanics of the material the points make up. The stiffness of a simulated body, relies on the parameter values for the points as well as the number of points over a set distance. The simulation divides time into discrete frames, where forces are computed at the start of the frame. The points respond to these forces by achieving new positions at the end of the frame. The forces of the next frame are determined by the positions of the points. This pattern repeats with one set of new forces and new positions each frame. This procedure sets restrictions on the speed of sound through simulated bodies. Giving a knock to one end of a rod will send a pressure wave through the simulated points. Importantly, there is one set of new forces and new positions each frame. Resulting in the pressure wave progressing only one point each frame. The speed of the wave will not exceed the distance between points over the time of a frame. The time interval of a frame could be changed to compensate, but it would have repercussions into other aspects of the simulation.

A trade off, of sorts, was chosen. The sizes of widths and heights were increased, but height was constrained when erroneous behaviour occurred. These behaviours were characteristic of transgressing the limits of the physics algorithm.

6.2.5 Time step size

The physics simulator has a setting for the size of the time step. This influences how far points move before the next frame is computed. A static triangulation algorithm will compute every point on every frame, but a dynamic one will ideally compute a fraction of the points on not all frames. This is because the classifiers tend to classify points for triangulation in relation to motion and time,

while the static algorithm triangulates points in relation to frames only.

The speedup will be influenced by the size of the time step. A smaller time step will make the dynamic algorithm triangulate points on a lower proportion of the frames. At very small time steps, the probability that there will be invalid points is asymptotically almost surely zero. While continuing to lower the time step, the speedup will converge to the ratio between the classifier cost and the static algorithm cost.

$$SU(n, \Delta t) = \frac{SC(n)}{DC(n, \Delta t)}$$

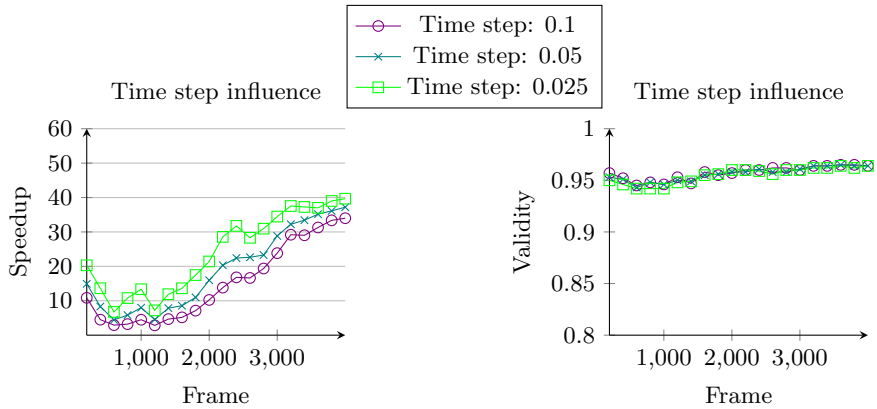
$$DC(n, \Delta t) = CC(n) + TC(PC(\Delta t))$$

$$\lim_{\Delta t \to 0} SU(n, \Delta t) = \frac{SC(n)}{CC(n)}$$

n: number of points,
Δt: Time step size,
SU(n, Δt): Speedup with n number of points with time step size Δt,
SC(n): Static algorithm cost function with n number of points,
DC(n, Δt): Dynamic algorithm cost with n number of points and time step size Δt,
CC(n): Classifier Cost of n number of points,
PC(Δt): Expected Positive Classifications with time step size Δt,
TC(PC(Δt): Dynamic Triangulation Cost of Positive Classifications.

The Positive Classifications is assumed to increase monotonically with Δt and approach zero when Δt approaches zero. The Dynamic Triangulation Cost is assumed to increase monotonically with Positive Classifications and approach zero when Positive Classifications approaches zero.

If the classifier cost is low, this relationship will potentially give misleadingly good results by displaying large values for speedup. However if the time step is increased, this situation will be less leveraged. At some point, the simulation will become unstable due to the time step being too large. The largest stable time step will be least 'favourable' for the dynamic algorithm. And arguably this is the situation where both algorithms are at their breaking points and their performance are most comparable.



(a) Validity with varying time steps

(b) Speedup with varying time steps

Figure 6.1: Performance of the Point Distance classifier on the Dam Break scenario with varying time steps. The validity remains unaffected by varying the time step. The speedup increases slightly when time step decreases.

6.2.6 Visual observations

The Point Distance classifier has very good results overall, but showed low ability to classify invalid neighbours on the Spinning scenario. The blocks will rotate and show some sides to each other at different times. The points on the surfaces will then eventually have invalid neighbourhoods. The classifier is based on detecting a change in length of the distance between neighbours. If the distance between neighbours stay within a threshold range, they will not be classified as invalid. This threshold range is defined as a percentage of the distance the neighbours had after triangulation. If two clusters of points are far enough apart, so that the threshold range is larger than their radius, no point will be classified if they are rotating. This weakness of the classifier could be argued to be acceptable given the specificity of the situation. If these clusters were to approach each other, the classifier would classify neighbours as invalid more frequently.

When viewing these situations with Voronoi cells, it becomes clear that the rotating cluster will barely alter the cell shape of surrounding points. This indicates that the distance from points in the cluster to neighbouring points do not change much. And this will not trigger a classification from the Point Distance classifier.

The neighbouring points of the cluster will define Voronoi cells that don't change much and encircles the cluster's Voronoi cells. This situation seems to be very applicable to clustering schemes that provides a hierarchical Voronoi diagram.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Maintaining a Delaunay Triangulation with dramatic speedups have been achieved by accepting some reduction of precision. The amount of precision could be set before triangulation and will determine how much the triangulation strays from a Delaunay triangulation.
This system can be seen as a generalisation of Delaunay triangulation, where a parameter can alter the behaviour from Delaunay Triangulation and towards a direction where the speedup increases.

The classifier and the merging procedure effectively performs triangulation culling. The omission of triangulating some areas is the main cause of speedup. The classifier serves to increase the amount of neighbourhoods that can be omitted. Thus leveraging the amount of computation that could be saved.

7.1.1 Addressing goals and research questions

In light of these results it can be stated that the goal of producing an efficient algorithm for heuristic dynamic Delaunay triangulation has been met. The number of points in a scenario can be much increased with a close to negligible reduction in Delaunay validity.

Research Question 1 sought to explore the potential for using the Voronoi diagram as a grounds for making a competitive classifier. An efficient equation for determining an invalid configuration of edges has been made. It assumes that the position of the circumcircles is known beforehand. For this equation to be competitive against other classifiers, a heuristic for finding the coordinates of circumcircles must be devised. Heuristics for this was not achieved in time, however appears to be a promising approach.

Research Question 3 sought to achieve a description of the obstacles involved in creating a heuristic dynamic Delaunay triangulation algorithm by using static algorithms as a basis. The heuristic algorithm had to incorporate the existing information contained in the triangulation to inform the configuration of the triangulation in the following frame. The task of extracting this information in a usable format was achieved with classifiers. This information was then used efficiently with a static triangulation algorithm. Depositing the results in the new triangulation was done with a merging procedure. Some obstacles materialised while creating the classifiers and merging procedure. These were described and ways around them were formed.

Research Question 2 sought to determine if the Bowyer-Watson algorithm was better suited than the Guibas-Stolfi algorithm as a base for creating a heuristic dynamic Delaunay triangulation algorithm. After developing an approach for this algorithm that allowed for easily adding or changing classifier, it became clear that the approach is also applicable to many types of triangulation algorithms. Bowyer-Watson does explicitly store triangles witch makes it easier to implement some types of classifiers. But it is hard to say that this is easier than augmenting the Guibas-Stolfi algorithm to store triangles as well.

Research Question 2: What properties does a heuristic triangulation algorithm based on Bowyer-Watson have?
The Bowyer-Watson algorithm explicitly stores information closely related to the Voronoi diagram and could be more suited for the Voronoi method than the Guibas-Stolfi algorithm.

7.2 Future Work

7.2.1 Classifiers

The Point Distance classifier is a heuristic that provides a lot of benefits over other classifiers. but it has some false positives and a few false negatives. When the classifier is set to a high precision setting, there are dramatic amounts of false positives. Perhaps a refinement procedure could consider the positives returned from the Point Distance classifier and remove some false positives. If this could be done in an efficient way, higher precision settings will be more viable and efficient.
The Point Distance classifier takes only distance into account and thus may provide similar results if applied to higher dimensions of Delaunay Triangulations.

The duration of the dynamic algorithm will vary according to the behaviour of the scenario. This allows for the situation where the speedup can be both above and below 1 during a scenario. In these cases it would be harder to determine which type of algorithm would be the better choice. A 'hybrid' algorithm could potentially achieve best of both worlds by switching between a static and dynamic algorithm. It would however require a reasonably good estimate of the speedup between the dynamic and static algorithm, while attempting to run only one each frame.

7.2.2 Subgraph Merging

Efforts were made to find an alternative method to deal with triangulating classified points. The method that is used in this paper does not use the existing neighbourhood information.
The first attempt was an approach that looked at individual points and tasked them with maintaining the Delaunay property of their neighbourhood. They were supposed to detect errors, correct them and notify neighbours of the new information they had determined. This however developed into a difficult task dealing with concurrency issues and points being triangulated multiple times. In short, this approach became complex and showed indication of not being computationally efficient. The Guibas-Stolfi algorithm appears to have many properties that are clearly better.
The second attempt dealt with salvaging previous information from invalid areas. This proved difficult because some, but not all, procedures assume that the neighbourhood is Delaunay. And the order in which these procedures are

performed may gracefully result in a Delaunay neighbourhood. Consider a triangulation that has valid and invalid neighbourhoods. There are borders that separate regions where the assumption that the neighbourhood is valid, will hold or not hold. And this border may move after some key points have been processed. A potential for an alternative triangulation method could arise if these key points could be identified.

7.2.3 Voronoi Edge lengths Classifier

By considering figure 2.2b, some interesting observations about the centre Voronoi cell can be made. The cell has six edges that corresponds to Delaunay edges. If the vertex in the cell, moves away from an edge in the cell, the length of the edge decreases. When the length is about to go negative, the edge configuration should change in order to remain Delaunay. The length of a Voronoi edge, gives some information about how far the vertex can move away from the Voronoi edge, before one of the vertex's edges become invalid. The Voronoi edges that goes radially outwards from the cell gives different information. Their lengths will decrease when the vertex moves towards them. Together the Voronoi edges sets bounds on the vertex's positions in 12 directions. If the lengths of Voronoi edges were easily accessible, then they could be used for a classifier with promising precision.

7.2.4 Voronoi Circumcenter classification

After triangulation, compute the length of the orthogonal projection of the two triangles' circumcentres on to the edge. The sum of these lengths may either be positive or negative. If the sign changes, the circumcenters have collided and the Voronoi diagram has become invalid. There may be a faster computation that only yields information about the sign of the sum and disregards magnitude.

The classifier assumes that the positions of circumcenters are already known. However there is a great potential for speedup through estimating these positions. A heuristic for determining the positions of circumcenters have not been made. However it has been observed that if the motion of the three points in a triangle can be, in part, be described by a translation, then the motion of the circumcircle can also be, in part, be described by the same translation.

7.2.5 Nearly Delaunay triangulation

Consider cases where two neighbouring triangles have nearly the same circumcircles. An example can be seen in the lower right of figure 2.2b. With a small movement of a point, the edge might become invalid. This could be seen as that the two ways of placing the edge is nearly equally good.
If both edges are allowed to be present in the graph, it would cease to be a Delaunay graph. However this would reduce the rate of invalidating events when points move or oscillate. This loosening of constraints may allow faster static and dynamic triangulation algorithms. It will however, require some definition or parameter for how much an edge can stray from the Delaunay definition.

7.2.6 Static triangulations for key frames

In figure 5.6b it can be seen that the validity remains close to one for some frames, before it converges to a level that is maintainable for the classifier. If a static algorithm triangulates at a frame before this convergence level, it would set the validity to one, and the dynamic algorithm would mitigate the decay of the validity. This approach would allow using a cheaper, less precise classifier since a static algorithm would stop the validity from declining past a certain level. However determining the validity of a triangulation requires information of the Delaunay triangulation of the points. Heuristics could estimate this by considering the motion of points, but a trivial method would be to regularly use a static triangulation algorithm.

This approach does not seem promising, due to the low cost and high precision of dynamic triangulation and that it seems to converge towards the Delaunay triangulation of the points.

Bibliography

[Ber+08] Mark de Berg et al. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, Heidelberg, 2008. ISBN: 978-3-540-77974-2. DOI: https://doi.org/10.1007/978-3-540-77974-2.

[Bow81] Adrian Bowyer. "Computing Dirichlet tessellations". In: *The Computer Journal* 24 (2 1981), pp. 162–166. DOI: https://doi.org/10.1093/comjnl/24.2.162.

[CD08] Pedro Machado Manhães de Castro and Olivier Devillers. *Delaunay Triangulations for Moving Points*. RR-6750. INRIA, 2008. URL: https://hal.inria.fr/inria-00344053.

[Fie17] Victor Fielding. *Heuristic Delaunay Triangulation*. 2017.

[GS85] Leonidas Guibas and Jorge Stolfi. "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams". In: *ACM Transactions on Graphics (TOG)* 4 (2 1985), pp. 74–123. DOI: http://dx.doi.org/10.1145/282918.282923.

[Wat81] David F. Watson. "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytypes". In: *The Computer Journal* 24 (2 1981), pp. 167–172. DOI: https://doi.org/10.1093/comjnl/24.2.167.