Smooth, hierarchical competitor clustering using agglomerative hierarchical clustering

Supervisors: Mark Winter, Berend Baas
Responsible Professor: Elmar Eisemann

Christiaan Botha
4802993

A research paper presented for the degree of
Bachelor of Science

Computer Graphics Department
Delft University of Technology
The Netherlands
June 19, 2022

Abstract

Clustering forms a major part of showing different relations between data points. Real-time clustering algorithms can visualise relationships between elements in a 3D environment, provide an analysis of data that is separate from the underlying structure and show how the data changes over time.

This paper analyses whether conventional clustering algorithms can be adapted to real-time dynamic data while remaining stable over time. By implementing an agglomerative hierarchical clustering algorithm combined with an exponential decay smoothing function, this paper tested several different distance functions and compared their resulting clusterings. It then derives a stable distance function for clustering sailboat competitors during a regatta and compared different smoothing values to see the impact on the final result.

The paper shows that an adaptively chosen smoothing value provides the best balance between cluster stability and an intuitive visualisation. This paper concludes this solution can be used and adapted to fit a multitude of applications by changing the distance function and the clustering depth.

1 Introduction

Clustering data is regularly used to analyse large data sets and can provide insight into or a visual representation of real-life data [1], [2]. Clustering algorithms therefore form an important basis for several research areas, such as machine learning, big-data analysis, and bioinformatics[2].

The Sailing+ application is a mobile application that allows users to spectate sailing regattas, or races, in a VR or AR environment. The application can show the field, the route and the competitors in play. A clustering algorithm would allow the race to be analysed in real-time. Clustering the competitors would allow the application to render faraway clusters in lower details, improving runtime, or perform group calculations to gain insight into the effects of certain manoeuvres. Such a clustering algorithm needs to be flexible enough to cluster dynamic data accurately. Furthermore, any functionality that clusters sailboat competitors needs to ensure a certain measure of stability of the clustering timeline, to allow the clustering to evolve with the competitors [3].

There is a plethora of clustering algorithms that will allow for clustering on different scales [2], [4]–[7]. Algorithms such as spectral clustering and mean-shift have to be run multiple times to display the same data with a different cluster sizes. This would not be an issue if the algorithm was run locally for every user. However, to allow the solution to be deployed to a separate machine or server, a hierarchical clustering algorithm would be better suited. This would allow the solution to cluster the data once and display different cluster sizes by looking at different depths of the resulting tree. To fulfil these constraints, this research will use an agglomerative hierarchical clustering algorithm.

Existing implementations of such smoothed hierarchical clusterings, however, run in environments where a clustering needs to be produced at most once per day [3], [8]. This paper will determine whether these existing methods can also be used in an environment that needs to produce a clustering once every frame, or sixty times per second. Furthermore, this paper will present several distance functions that will cluster the competitors differently, allowing for basic data analysis and compares these functions on whether they produce an inherently stable clustering. Lastly, this paper will argue whether extra smoothing of the timeline is necessary in the current context.

This research will focus on the question: "What are efficient methods for smooth, hierarchical clustering of sailboat competitors for multi-scale race visualisation that can provide smooth level-of-detail transitions in the AR/VR environment?" To answer this question, it first needs to be split into two sub-questions: "What are efficient methods for hierarchically clustering sailboat competitors" and "How can a frame-by-frame clustering timeline be altered to provide smooth transitions from one frame to the next?"

Section 2 will explain how this paper answers the research question and which steps have been taken to arrive at that answer. Section 3 will show the results of implementing the steps mentioned in Section 2. Section 4 will then discuss the results and answer the research questions, while reflecting

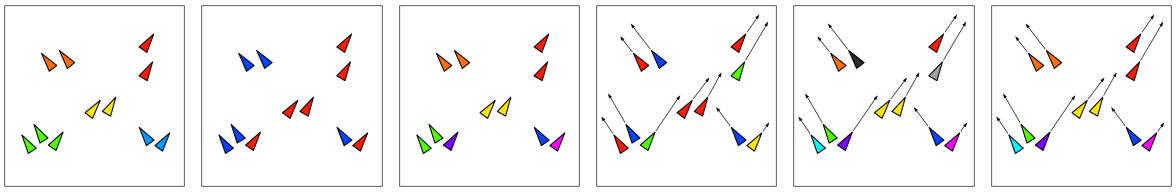
on the assumptions and limitations of the solution. Lastly, Section 5 will provide a summary of the research questions, their answers and the conclusions drawn from said answers.

2 Methodology

The following section explains how the solution clusters sailboat competitors in such a way that the clustering remains smooth over time.

Clustering sailboat competitors

The distance function decides which clusters will be merged, and therefore dictates what any given cluster represents. Given a universe of competitors to cluster $C = \{c_1, c_2, \ldots, c_n\}$, the distance function determines how similar (or dissimilar) two pairs of competitor clusters are, and the features they cluster on will have different impacts on overall smoothness of the timeline.



(a) (b) (c) (d) (e) (f)

Figure 1: Images depicting clusterings based on several different distance functions.
a. Physical closeness.
b. Current tack.
c. Closeness and tack.
d. Relative velocity.
e. Closeness, tack and relative velocity.
f. Closeness, tack and distance-to-leader velocity.

Euclidean distance A Euclidean distance function clusters competitors based on physical distance on the regatta field. Clustering competitors on opposite sides of the field prevents a smooth transition between levels of the clustering tree, as not all competitors might be visible when zooming in on a sub-cluster. The distance function

$$d_{euc}(c_i, c_j) \tag{1}$$

returns the Euclidean distance between competitors c_i and c_j. Clusters based on distance are separated by an open space, as shown in Figure 1a. Competitors that are grouped together on the regatta field will also be grouped together in the clustering tree, making the clustering intuitive to look at. However, Figure 1a also shows competitors that are sailing in different directions are still grouped together if they are close at the time of clustering, as shown by the clusters in green and blue. This will result in an unstable clustering, as the two competitors will be moving away from each other over the course of the next few seconds.

Sailboat tack A distance function based on the current tack clusters competitors on the similarity of their current sailing direction. Figure 1b shows that competitors sailing on the same tack will be grouped together, while competitors sailing on different tacks will be in different clusters. By using the Equation

$$d_{dir}(c_i, c_j) = \begin{cases} 0.01 & \text{if } \text{tack}_i = \text{tack}_j \\ 1 & \text{else} \end{cases} \tag{2}$$

where tack_i indicates the current tack, port or starboard, of competitor c_i. Competitors that are on the same tack —or have the wind come over the same side of the ship— will have a smaller distance value than competitors that are on different tacks, clustering them together. A clustering based on tack ensures that competitors that are clustered together will be sailing in a similar direction, meaning that they will still be clustered together after a few seconds. However, competitors sailing in the same direction might be on opposite sides of the field, as shown by the blue competitors that are clustered together in Figure 1b but are in separate clusters in Figure 1a.

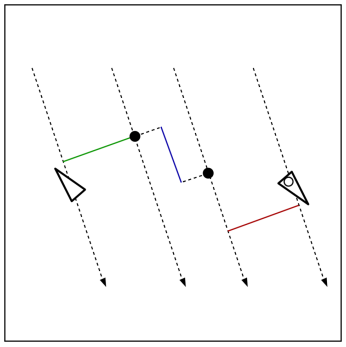


Figure 2: The distance to leader is calculated by adding the green, blue and red lines together. The dotted arrows indicate wind direction, the competitor with the circle is the leader and the black circles are marks.

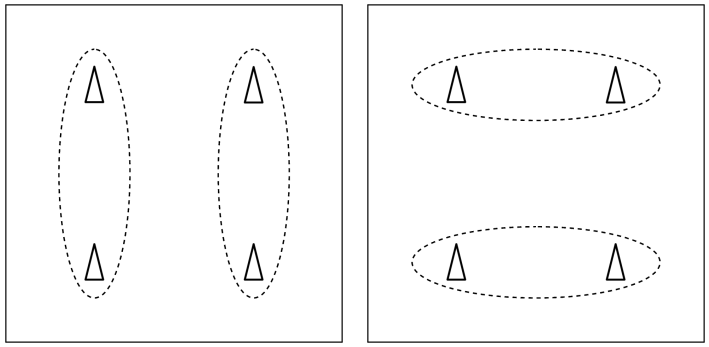
Distance-to-leader velocity A distance function based on the distance-to-leader velocity determines clusters based on whether a competitor is faster than competitors in front of them in the regatta. Firstly, the distance to leader is defined as the sum of multiple-of-90 distances to the wind direction between the competitor c_i and any marks between c_i and the leader plus the distance between the last mark (or c_i if there are no marks) and the leader. Figure 2 shows how the distance to leader is calculated. The green line represents the distance from competitor c_i to the next mark. The blue line represents the distance between marks and the red line represents the distance from the last mark to the leader. The dotted arrows indicate the wind direction and the leader is indicated with a circle. the distance to leader can be defined as

$$\text{dtl}(c_i) = d_w(c_i, \text{nm}_i) + \sum_{m_j=\text{nm}_i}^{\text{nm}_\text{l}} d_w(m_j, m_{j+1}) - d_w(\text{leader}, \text{nm}_\text{l}) \tag{3}$$

where $d_w(\cdot,\cdot)$ returns the point-to-line distance between two object, where one line goes through a specified object and is a multiple-of-90 angle of the wind direction. nm_i returns the next mark m_j from c_i, nm_l is the next mark from the leader. By combining the distance to leader and the relative velocity between competitors, the solution can separate competitors c_i and c_j based on c_i is faster and further back (overtaking), c_i is slower and further back (falling behind) or c_i is just as fast (maintaining distance). Equation 3 can then be used to construct a distance function based on distance-to-leader velocity

$$d_{dtl}(c_i, c_j) = \begin{cases} 0.01 & \text{if } \text{dtl}(c_i) > \text{dtl}(c_j) \wedge \text{vel}_i > \text{vel}_j \\ (\text{vel}_i - \text{vel}_j)^2 & \text{else} \end{cases} \tag{4}$$

where vel_i is the current velocity of c_i. By using a distance function based on the distance-to-leader velocity, the solution is able to cluster competitors with similar velocities together, as well as competitors overtaking each other. Where combining the distance function solely on relative velocity would split on any difference in velocity, as in Figure 1e. By combining with the distance to leader, clusters that are approaching each other will be clustered together, while separating competitors whose distance to



(a) Clustering on even frames (b) Clustering on odd frames

Figure 3: How an unstable algorithm clusters competitors on even- and odd-numbered frames.

each other will increase, as seen in Figure 1f.

Combining distance functions To observe the effects of each distance function, the different functions can be combined by first normalising them over their current maximum observed value.

$$d(c_i, c_j) = d'_{euc}(c_i, c_j) + \alpha \cdot d'_{dir}(c_i, c_j) + \beta \cdot d'_{dtl}(c_i, c_j) \tag{5}$$

where α and β are the weight assigned to each distance function and $d'_x(\cdot, \cdot)$ is the normalised counterpart to $d_x(\cdot, \cdot)$, represents the combination of these distance functions. This resulting distance function was then analysed in different scenarios, to determine the effects of different values for α and β on the clustering and the smoothness of the timeline. First by examining the effects of different weight parameters on single instance clusterings. Then by examining the effects on a period of several seconds.

Smoothing the clustering timeline

To prevent an unstable timeline of clusterings due to running a clustering algorithm on dynamic data, it is necessary to implement a method for smoothing clusterings between frames [3], [8]. Take the following, although extreme, scenario as shown in Figure 3: Suppose there are four competitors sailing in a rough two by two grid. On even-numbered frames, the distance between the rows of competitors is ever so slightly smaller than the distance between the columns of competitors (Figure 3a). This means that on even-numbered frames the two competitors in each column will be clustered together. Now on odd-numbered frames, the distance between columns is slightly smaller than the distance between the rows, causing the two competitors in each row to be clustered together. The resulting clustering of a standard clustering algorithm would shift radically between frames, while a more stable clustering where either the competitors in each row or the competitors in each column were clustered together, would still perform arbitrarily close to optimal. A stable clustering timeline means a more stable visualisation of the clusters, or more stable colours when using them to indicate clusters.

To smooth the clustering timeline, the solution constructs a distance matrix M containing the distance between any pair of competitors c_i and c_j. The solution then uses exponential decay to smooth M at time t with the previous matrices. Equation

$$M_{ij,t} = (1 - \delta) \cdot d(c_i, c_j) + \delta \cdot M_{ij,t-1} \tag{6}$$

where δ is a parameter indicating the smoothing value, defines this smoothing function. By increasing the smoothing value, past distance matrices will be more emphasised, leading to a smoother but less accurate clustering.

To find the optimal smoothing value, Chakrabarti, Kumar, and Tomkins [3] proposes a solution that minimises the history cost, while maximising the snapshot quality of every frame. The history

cost is defined as the average of all squared error tree distances between two competitors [3, p. 557]. In other words, the tree distance $\mathrm{td}_T(c_i, c_j)$ between competitor c_i and c_j is defined as the number of edges between c_i and c_j in clustering T. The history cost between two clusterings T, T' is defined as

$$\mathrm{hc}(T, T') = \frac{1}{n} \cdot \sum_{c_i, c_j \in C} (\mathrm{td}_T(c_i, c_j) - \mathrm{td}_{T'}(c_i, c_j))^2 \tag{7}$$

The snapshot quality can then be defined as the history cost between the proposed clustering T_p and the clustering based on the unsmoothed competitor values T_{true}. To find the ideal smoothing value, the solution then tries to minimise the cost of clustering the competitors as proposed in T_p and the differences between T_p and the previous clustering T_{t-1}:

$$\mathrm{cc}(T_p) = \mathrm{hc}(T_p, T_{true}) + \epsilon \cdot \mathrm{hc}(T_p, T_{t-1}) \tag{8}$$

where ϵ is a constant indicating the relative importance of the history cost.

To calculate an adaptive smoothing value, the solution proposes three clusterings T_δ for $\delta = 0$, $\delta = 0.5$ and $\delta = 1$. The solution then calculates the smoothing value by substituting T_p for each T_δ and finding the minimum clustering cost.

3 Results

Clustering competitors

The resulting clustering of different weighting parameters α and β in Function $d'_{euc}(c_i, c_j) + \alpha \cdot d'_{dir}(c_i, c_j) + \beta \cdot d'_{dtl}(c_i, c_j)$ around a mark are shown in Figure 4. The rows and columns show a change in α and β respectively, with the bottom left corner indicating an α and β of 0 and the top right corner a α of 1 and a β of 0.75. Any values larger did not have a noticeable impact on the clustering and have been subsequently left out. Figure 4 shows higher values of α generally result in a cluster count that is no larger than lower values of α, $\alpha = 1, \beta = 0$ being the only exception. Furthermore, an increase in α results clusterings where more competitors are on the same tack. An increase in β results in a clustering where competitors far away from the mark are clustered with tacking competitors if their velocity is high enough, as seen with the light blue competitor in $\alpha = 0, \beta = 0.25$.

Figure 5 shows how the clusterings of several distance functions change over time. The columns represent distance functions with different weights for the parameters α and β, while the rows are different snapshots taken 4 seconds apart. Column 1 shows that for a Euclidean distance function, the clustering around the mark (red) becomes larger as the density of competitors increases and new clusters are formed as soon as a competitor (in yellow) moves away from the mark in Frame 5 ($t = 5$). Column 2 shows a combination of Euclidean distance and sailboat tack. The cluster around the mark (green) increases in size as t increases, the competitors on different tacks are still separated (orange). Furthermore, the competitor that separated on Frame 5 in Column 1 (yellow) is still clustered in Column 2. Column 3 shows a combination of Euclidean distance and distance-to-leader velocity. This Column shows the greatest change in clusterings over time, as the competitors in each cluster changes every snapshot, where for Column 1 and 2 most of the sailboats remained in the same clustering. Column 4 shows a combination of Euclidean distance, sailboat tack and distance-to-leader velocity. While the red cluster is more consistent in Frame 1 to 4 than in Column three, Frame 5 shows a large change in tree structure with the purple cluster being merged into yellow and the competitors in the red, green and light blue clusters changing cluster. It furthermore clusters competitors yet to pass the mark (yellow in Frame 4) with competitors moving away (yellow cluster) similar to Column 3 and unlike Column 2.

Smoothing the clustering timeline

Apart from the distance function, choosing a appropriate smoothing value also plays a role on the overall smoothness of the clustering timeline. Figure 6 shows the effect different smoothing values

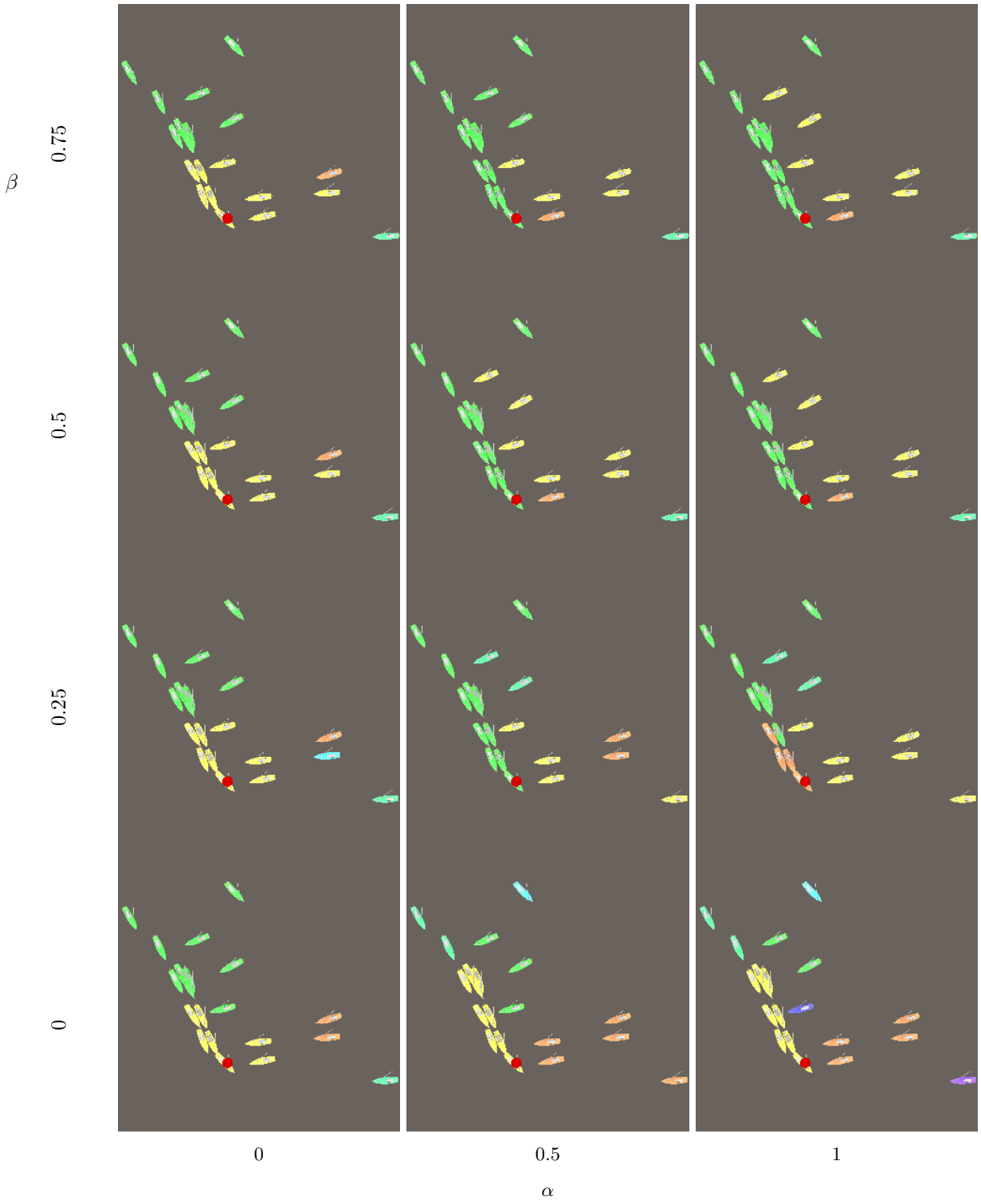


Figure 4: The clusterings resulting from distance functions with different weights for the tack (α) and velocity (β).

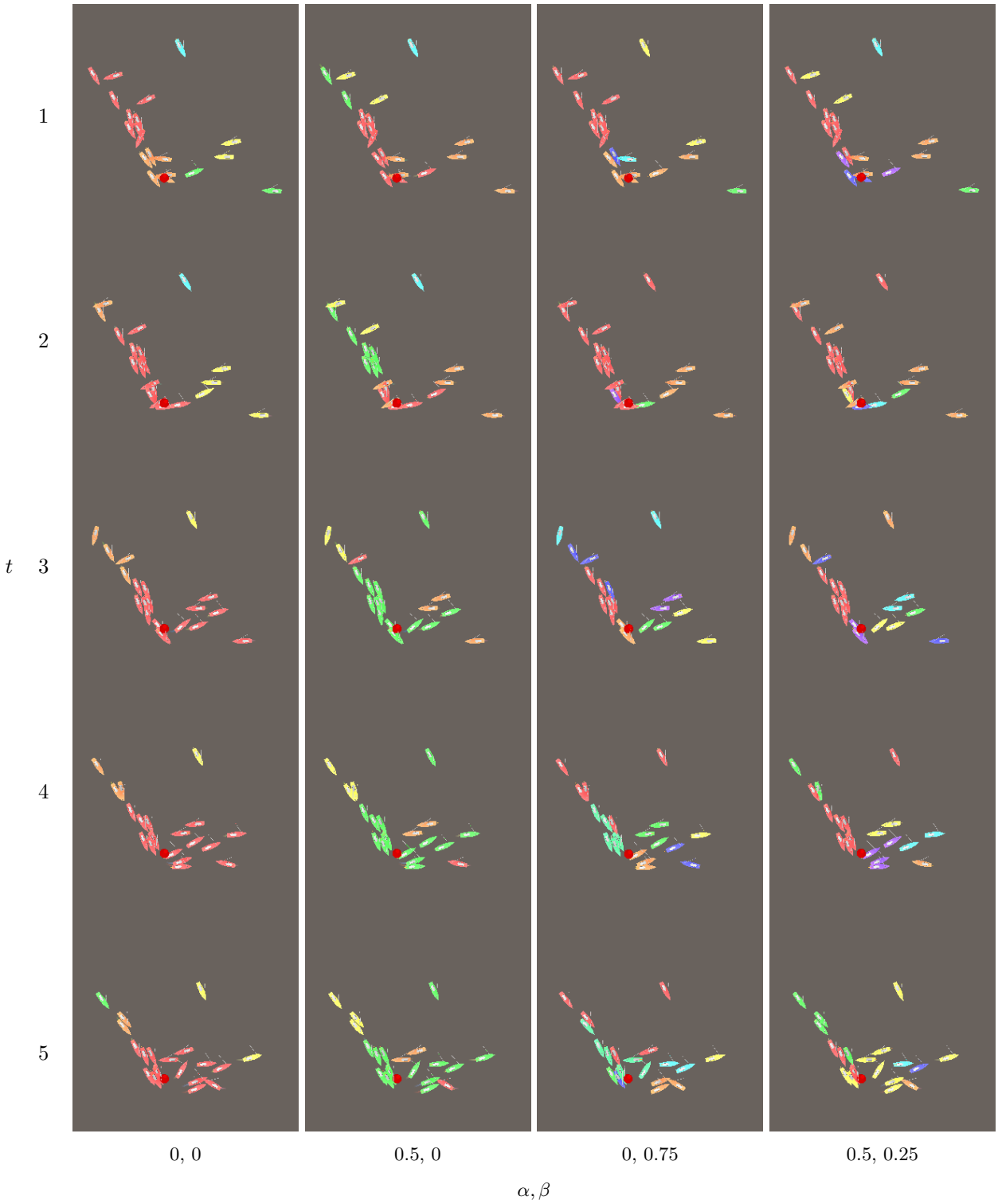


Figure 5: Different values for the weighting parameters α and β impact the stability of the distance function over time.

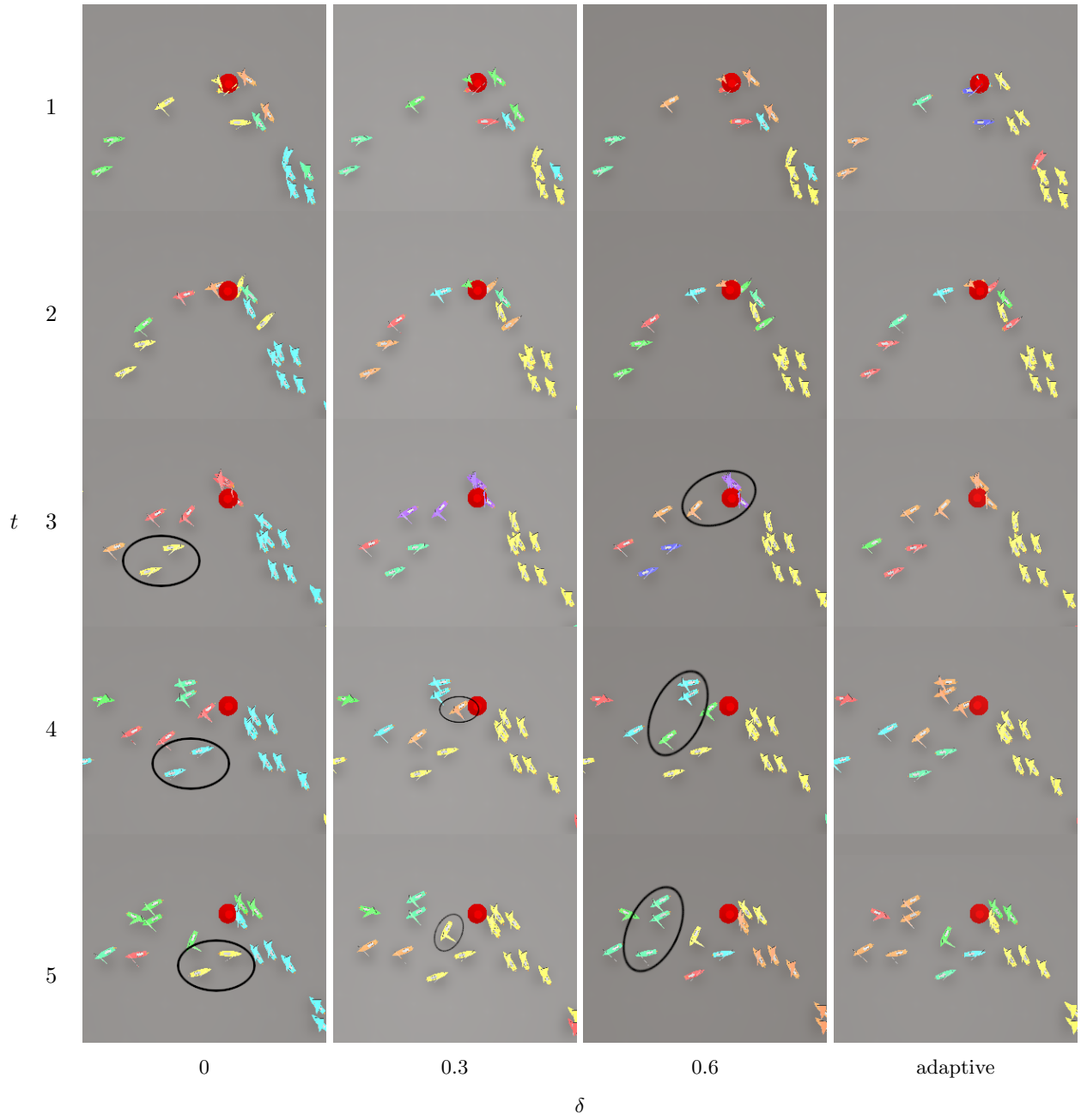


Figure 6: An adaptive smoothing value δ results in a more stable timeline than a low smoothing value, but is more accurate than a high smoothing value. Encircled competitors show unstable clusterings for $\delta = 0, \delta = 0.3$ and a delayed merge for $\delta = 0.6$.

have on the clustering timeline. The first three columns show different static smoothing values. The last column shows an adaptive smoothing value. The rows are snapshots taken of each smoothing value 4 seconds apart and the distance function used weight parameters $\alpha = 1, \beta = 0.5$ to show the result of the smoothing value on an unstable clustering.

Column 1 demonstrates how competitors are clustered around a mark without any smoothing over time. The instability of the clusterings is shown by the encircled competitors. These competitors are clustered together with the light blue cluster only momentarily (Frame 4), while splitting again directly after (Frame 5).

Column 2 shows how a static smoothing value of $\delta = 0.3$ improves stability, as the same two competitors encircled in Column 1 now remain clustered with the yellow group. Column 2 does show that a competitor that has passed a mark can still be clustered with competitors moving towards the mark, as the encircled competitor in Frame 4 is clustered with the yellow competitors in Frame 5.

Column 3 shows the clustering over time when the smoothing value is increased to $\delta = 0.6$. Frame 4 shows that the competitor encircled in Column 2 is no longer clustered with the orange group. Column 3 does show the delay between the encircled competitors passing the mark in Frame 3 and being clustered together in Frame 5.

Lastly, Column 4 shows how the solution clusters competitors over time around a mark, using the adaptive smoothing function. The Figure shows a large cluster (in yellow) and several smaller clusters (in orange, blue and red) approaching the mark. Frame 3 and 4 show how the highlighted competitors in Column 3 remain clustered together as soon as they pass the mark in Frame 3 rather than once the timeline has stabilised in Frame 5.

4 Discussion

The timelines in Figure 5 show that distance-to-leader velocity is not an effective distance function to increase cluster stability. The larger cluster sizes in Figure 4 shows that both distance functions decrease overall competitor distance by increasing cluster sizes for the same tree depth, but due to the volatility of the wind direction, the distance from a competitor to the leader (Equation 3) changes. Changing the distance to leader means that the distance-to-leader velocity (Equation 4) function also changes. Furthermore, a change in wind direction affects both the distance to leader as well as the velocity of a competitor itself. This means that a timeline containing the distance-to-leader velocity is inherently unstable.

Combining the distance function with the tack does increase clustering stability. Instead of competitor density being the only clustering factor, the sailboat tack separates groups sailing in different directions, such as groups approaching the mark and groups moving away from the mark. As soon as the competitors are sailing in the same direction, the sailboat tack functions similar to only Euclidean distance. A clustering based on tack is effectively split into two clusters, as the distance d_{dir} (Equation 2) between any two competitors is either 0.01 or 1. Any subdivision of the clusters is based solely on the order the clustering algorithm finds these minimum values, rather than the distance function. Increasing the value of α beyond 1 therefore has no extra influence on the clustering, as Euclidean distance between competitors will still the deciding factor between competitors on the same tack. A value of $\alpha = 0.5$ therefore provides a trade-off between the spacial stability of Euclidean distance and the stability over time of the sailboat tack.

Smoothing the distance matrix allows the solution to smooth inconsistencies left by the distance function. Where no or a low smoothing value is more accurate to the data, it is also more unstable, as small changes lead to a different clustering almost immediately. A high smoothing value is more consistent, but at the cost of the snapshot quality leading to a less accurate representation. Figure 6 shows that an adaptive smoothing value is able to trade between the fast and accurate clustering of a low smoothing value as well as the consistency if a high smoothing value.

Currently, only three "proposed" clusterings are generated before choosing the smoothing parameter. A larger number of proposed clusterings would increase the accuracy of the smoothing parameter

at the cost of performance. A more accurate smoothing parameter would decrease the history cost over the timeline, resulting in a smoother clustering over time.

The solution assumes a small number of competitors, at most twenty to thirty, when clustering on a mobile device. Given a device with more computational power, the solution would be able to cluster a larger number of competitors in real-time. The expected performance of the solution is $O(n^3)$, but future versions can improve this performance to potentially $O(n^2)$, also allowing for a larger number of competitors [9]–[11].

There is a small time-frame during the start of the regatta where, due to the high similarity and uncertainty of direction competitors will take, clustering becomes unstable. As soon as the competitors decide on a direction, the clustering stabilises again. However, future research into a more distance function over more features, such as whether a team is more likely to choose an upwind or downwind start, could solve this issue.

The solution allows users to modify the clustering via several parameters. The first choice is the cluster size by changing the depth parameter, showing more or fewer competitors per cluster. Second, the weight of the history cost can be chosen to place more emphasis on previous clusterings and smoothness, rather than accuracy relative to current data. In the future, a machine learning algorithm could be developed that, given the number of competitors, the course structure and similar regattas is able to recommend values for these parameters.

5 Conclusion

This research aimed to implement and test whether conventional hierarchical clustering methods can be used in an environment where a cluster needs to be produced sixty times per second. The research further aimed to resolve the issues arising from continuously clustering dynamic data, such as sailboat competitors. Based on a qualitative analysis of several distance functions and their smoothness over a few seconds, it can be concluded that a combination of Euclidean distance and tack result in the most stable distance function. Choosing the smoothing value between two clusterings based on the differences between the two also showed to be the best trade-off between the more stable high smoothing value and the accuracy of a low or no smoothing value.

The presented solution can be applied in contexts other than the Sailing+ application by exchanging the distance function and possibly changing the weight of the history cost. Given a suitable distance function, the solution can cluster elements in an image (sequence), which can then be used to display relations between these elements. Another application would be displaying faraway objects with a lower level of detail. The clusters could be tested against their distance to the camera. Based on the level in the tree, the level of detail could be reduced to increase rendering performance or reduce image noise.

Where previous algorithms only produced at most a clustering once per day, the solution is able to run in a real-time environment while simultaneously maintaining smoothness and accuracy. Due to the hierarchical nature of the solution, it can easily be deployed to a distributed system and each client can analyse the data at a different depth of the tree. Lastly, the distance function is easily replaceable to allow for easy analysis of different aspects of the data.

In conclusion, a real-time solution of a clustering algorithm will opens new doors for data analysis. The feasibility of such a solution is a step towards new research possibilities and more powerful visualisation systems.

References

[1] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann, "Interactive lenses for visualization: An extended survey", *Computer Graphics Forum*, vol. 36, no. 6, pp. 173–200, 2017. DOI: https://doi.org/10.1111/cgf.12871. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12871. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12871.

[2] V. Cohen-addad, V. Kanade, F. Mallmann-trenn, and C. Mathieu, "Hierarchical clustering: Objective functions and algorithms", *J. ACM*, vol. 66, no. 4, Jun. 2019, ISSN: 0004-5411. DOI: 10.1145/3321386. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/3321386.

[3] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering", in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06, Philadelphia, PA, USA: Association for Computing Machinery, 2006, pp. 554–560, ISBN: 1595933395. DOI: 10.1145/1150402.1150467. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/1150402.1150467.

[4] S. Theodoridis and K. Koutroumbas, "Chapter 13 - clustering algorithms ii: Hierarchical algorithms", in *Pattern Recognition (Fourth Edition)*, S. Theodoridis and K. Koutroumbas, Eds., Fourth Edition, Boston: Academic Press, 2009, pp. 653–700, ISBN: 978-1-59749-272-0. DOI: https://doi.org/10.1016/B978-1-59749-272-0.50015-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781597492720500153.

[5] R. Xu and D. C. Wunsch, "Chapter 3. hierarchical clustering", in *Clustering*. IEEE Press, 2009, pp. 31–46.

[6] K. S. Xu, M. Kliger, and A. O. Hero III, "Adaptive evolutionary clustering", *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 304–336, Mar. 2014. DOI: 10.1007/s10618-012-0302-x.

[7] H. Bezerra, E. Eisemann, X. Décoret, and J. Thollot, "3d dynamic grouping for guided stylization", in *Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR '08, Annecy, France: Association for Computing Machinery, 2008, pp. 89–95, ISBN: 9781605581507. DOI: 10.1145/1377980.1377998. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/1377980.1377998.

[8] C. C. Aggarwal, P. S. Yu, J. Han, and J. Wang, "A framework for clustering evolving data streams", *Proceedings 2003 VLDB Conference*, pp. 81–92, 2003. DOI: 10.1016/b978-012722442-8/50016-1.

[9] T. Kurita, "An efficient agglomerative clustering algorithm using a heap", *Pattern Recognition*, vol. 24, no. 3, pp. 205–209, Jan. 1991. DOI: 10.1016/0031-3203(91)90062-a. [Online]. Available: https://doi.org/10.1016/0031-3203(91)90062-a.

[10] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms", *Comput. J.*, vol. 26, pp. 354–359, Nov. 1983. DOI: 10.1093/comjnl/26.4.354.

[11] ——, "Complexities of hierarchic clustering algorithms: State of the art", *Computational Statistics Quarterly*, vol. 1, Jan. 1984.

A Responsible Research

I have taken several steps to ensure academic integrity of the paper. To ensure reproducibility, this paper includes a full description of the process taken to arrive at the results shown in the paper. Furthermore, I have maintained a separate log writing down the thought process behind the steps taken. This log can, at request, be published if anyone wishes to read it, by sending an email to `J.C.Botha@student.tudelft.nl` or by contacting the university.

All data used in this paper was already included in the Sailing+ application and was obtained in cooperation with a sailing organisation and their competitors. The results and conclusions were solely based on the Figures and Equations shown in the Methodology and Results and are anonymous, so that they may be observed without any implicit bias.

Over the course of doing this research, the paper was submitted several times for review. These reviews were able to point out flaws and inconsistencies throughout the writing process and have either been addressed or accentuated as not to hide these concerns.