LIU-ITN-TEK-A--15/041--SE

Real-time Wind Direction Filtering for Sailboat Race Tracking

Emil Nielsen

2015-06-11

LIU-ITN-TEK-A--15/041--SE

Real-time Wind Direction Filtering for Sailboat Race Tracking

Examensarbete utfört i Datateknik
vid Tekniska högskolan vid
Linköpings universitet

Emil Nielsen

Handledare Jan Petersson
Examinator Aida Nordman

Norrköping 2015-06-11

Abstract

In this paper, an algorithm that calculates the direction of the wind from the directions of sailors during fleet races is proposed. The algorithm is based on a 1-D spatial convolution and it is named Convolution Based Direction Filtering (CBDF). The CBDF-algorithm is used in the TracTrac race client that broadcasts sailboat races in real-time. The fact that the proposed algorithm is polynomial makes it suitable, to be used as a real-time application inside TracTrac, even for large fleets. More concretely, we show that the time complexity of the CBDF-algorithm is $\mathcal{O}(n^2)$, in the worst-case, where $n > 0$ is the number of boats in competition. It is also shown that in more realistic sailing scenarios, the CBDF-algorithm is in fact a linear algorithm.

Sammanfattning

I denna rapport föreslås en algoritm för att beräkna vindriktningen baserad på riktningarna av seglare under en kappsegling. Algoritmen är baserad på en 1-D spatiell faltning och kallas Convolution Based Direction Filtering (CBDF). CBDF-algoritmen används i TracTracs tävlingsklient som direktsänder kappseglingar i realtid. Det faktum att den föreslagna algoritmen är polynomial gör CBDF-algoritmen passande för att användas i realtid, även för stora kappseglingsflottor. Mer konkret visar vi att tidskomplexiteten för CBDF-algoritmen i värsta fallet är $\mathcal{O}(n^2)$, där $n > 0$ är antalet båtar i kappseglingen. Dessutom påvisar vi att i realistiska scenarion under en kappsegling är CBDF-algoritmen en linjär algoritm.

Contents

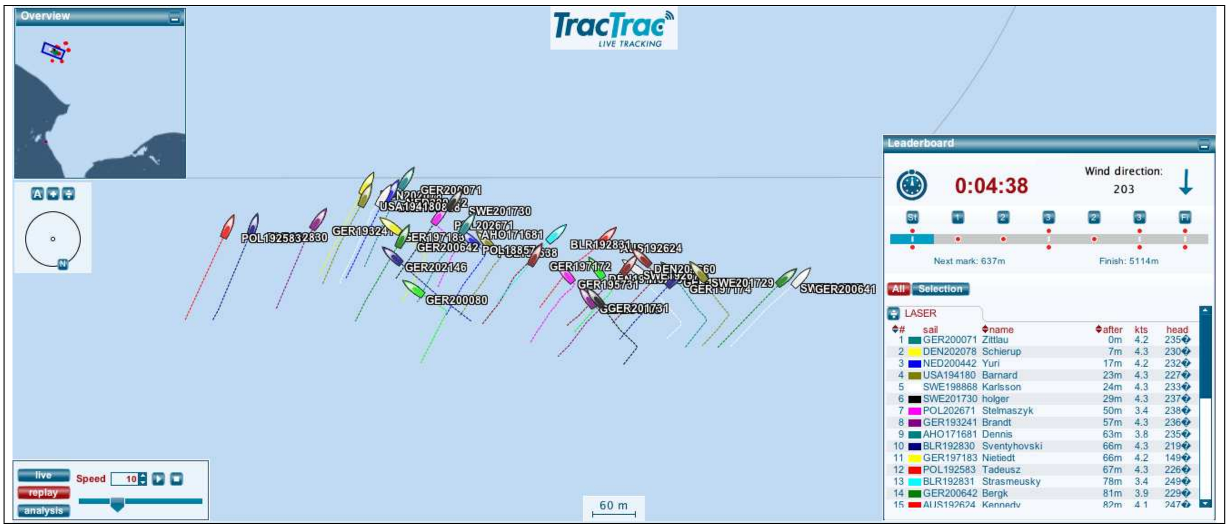
List of Figures

Chapter 1

Introduction

1.1 Motivation

In sailboat fleet racing the action often happens far from the spectators, which makes it hard to follow the race for the audience. During the last decade GPS tracking during competitions has been used to show sailboat racing up close. Figure 1.1 shows a fleet of Laser dinghies. The virtual leader board is shown in the lower right corner in figure 1.1. It shows the current standing in the race. The three rightmost columns in the virtual leader board contain information derived from the GPS unit on each boat. The *After*-column denotes the distance in meters that the boat is behind the leader. The *kts*-column denotes the current speed in knots. The *head*-column denotes the current heading. An overview map showing the distance to shore is shown in the top left corner.



Leaderboard

0:04:38

Wind direction:
203

St 1 2 3 2 3 Fi

Next mark: 637m

Finish: 5114m

All Selection

LASER

#		sail	name	after	kts	head
1	🟩	GER200071	Zittlau	0m	4.2	235�
2	🟨	DEN202078	Schierup	7m	4.3	230�
3	🟦	NED200442	Yuri	17m	4.2	232�
4	🟫	USA194180	Barnard	23m	4.3	227�
5	⬜	SWE198868	Karlsson	24m	4.3	233�
6	⬛	SWE201730	holger	29m	4.3	237�
7	🟪	POL202671	Stelmaszyk	50m	3.4	238�
8	🟪	GER193241	Brandt	57m	4.3	236�
9	🟩	AHO171681	Dennis	63m	3.8	235�
10	🟦	BLR192830	Sventyhovski	66m	4.3	219�
11	🟨	GER197183	Nietiedt	66m	4.2	149�
12	🟥	POL192583	Tadeusz	67m	4.3	226�
13	🟦	BLR192831	Strasmeusky	78m	3.4	249�
14	🟩	GER200642	Bergk	81m	3.9	229�
15	🟥	AUS192624	Kennedy	82m	4.1	247�

Figure 1.1: *A fleet tracked during the 2012 Kieler Woche in the Laser class*

During TracTrac GPS trackings a virtual leader board is calculated and shown in the race viewer. The spectators can see the current standings at all times during the races despite the vast distances of the race course. The direction of the wind can affect the leader board calculations and therefore the calculations become more exact when the wind direction is known.

An approach to the wind direction problem would be to cover the entire competition area with wind vanes (wind direction measurement equipment), but this is not practical or economically viable. Another approach is to use one or two wind vanes at the marks but this will not give an exact picture of the wind throughout the competition area, since the distance between the marks and the boats often is more than 2-3 kilometres and the surrounding landscape (mountains, valleys etc.) have a local effect on the wind direction that wind vanes at the marks would not be able to detect.

This thesis highlights another approach to the problem based on an algorithm that calculates an estimate of the wind direction, given the directions of the boats in a racing fleet. The algorithm and its implementation is the focus of this work. We also present the expert knowledge gained in a discussion with sailing experts and how it is used to modify the algorithm to reflect the specifics of sailing.

The main motivation for this work is to provide information about the wind to the spectators. Information about the wind direction help the spectator understand the decisions of the sailors. Furthermore the information about the wind makes it easier to introduce the sport of sailing to new audiences as commentators can explain the decisions of the sailors as they see the changes in wind direction.

The implementation of the algorithm reported in this paper was done at TracTrac ApS (`http://www.tractrac.com`) using the company's datasets. It is implemented and running as a part of TracTrac's race viewer client.

The next section introduces the reader to the necessary sailing background required for this thesis.

1.2 Background

During this section and throughout the report sailing specific words are used to describe the interactions of the wind with the boats and the interactions of the boats. In appendix A we have provided a list of these words and a short description.

Sailboats are propelled by the wind using the sails on board. With the wind coming from behind the boat, the principles of sailing are simple. You use the sail as a parachute and the boat moves in the direction of the wind. Sailing against the wind is when sailing becomes harder. Figure 1.2 shows Bernoulli's Principle as airflow around a wing or in the case of sailing a sail [1, Chapter 4]:

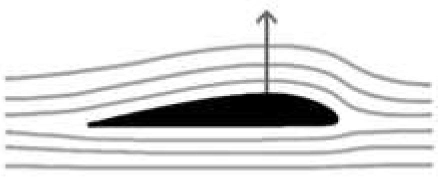


Figure 1.2: *The principle of lift*

In figure 1.2, the grey lines illustrate the flow of wind around the sail. The wind traveling around the outside of the sail (above the sail in figure 1.2) travels faster. Thus, there will be a decrease in pressure on the outside of the sail, as stated in Bernoulli's Principle. This leads to higher pressure on the inside of the sail than on the outside. This difference in pressure creates lift and the boat is moved in the direction of the arrow. In this way, a sailboat can sail approximately in the direction of wind, if the sail is pointed in the right direction. Figure 1.3 shows the different ways of pointing the sail. The red marked area (marked with $\mathcal{A}$) is the no-sail zone, where it is not possible to get the sail at an angle to create the needed lift to move the boat. The two positions closest to the no-sail zone are called close haul or beating the wind (marked with $\mathcal{B}$). This is the optimal direction for sailing against the wind [2, Chapter 2 and 3], [3].

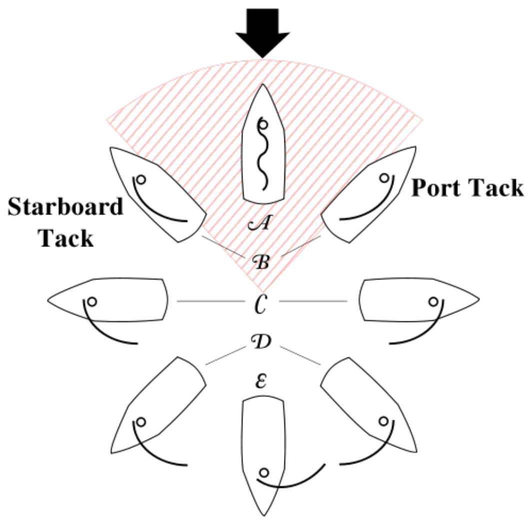


Figure 1.3: *The main positions for the sail on a boat*

Sailing boats cannot move directly against the direction of the wind and therefore the sailors tack. To tack is to turn the bow of a boat (the front) through the wind. This is done by pulling or pushing the rudder and adjusting the sails so the boat will move from $\mathcal{B}$ on one side of the wind to $\mathcal{B}$ on the other side of the wind in figure 1.3.

Tack is also used to describe the two close haul directions and the two tacks are called starboard and port tack. Port tack is where the wind is blowing from the left side of the boat. Starboard tack is where the wind is blowing from the right side of the boat. By tacking the sailor can go to any desired point. Figure 1.4 shows the process of sailing against the wind by tacking, called beating to windward:

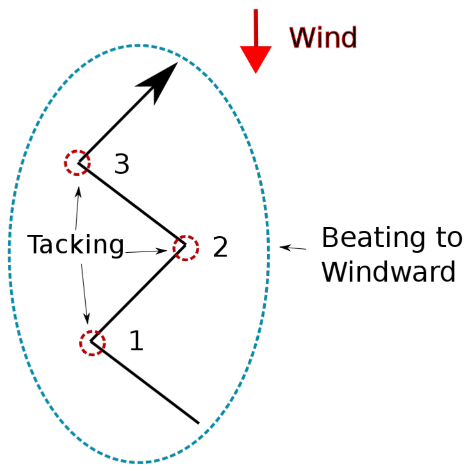


Figure 1.4: *Beating to windward, the process of sailing against the wind by tacking*

The proposed algorithm highlighted in this report is based on the statistics of boats in fleet races. Fleet races are the most common type of sailboat racing and can be seen in the Olympic Games and other competitions. Fleet races consist from five to upwards of 150 boats. The wind direction estimation is based on all or a subset of the directions of the boats. The sailors tracked by TracTrac are all skilled at trimming the sail and therefore the directions of their boats will not vary much from the optimal direction of close haul. Figure 1.5 shows the simplest racing course used in fleet racing. The red circles indicate marks and the black line the route of sailing. The marks correspond to buoys in the water and the objective is to get as quickly as possible through the course abiding to the rules and regulations of sailing [2], [3], and [4].

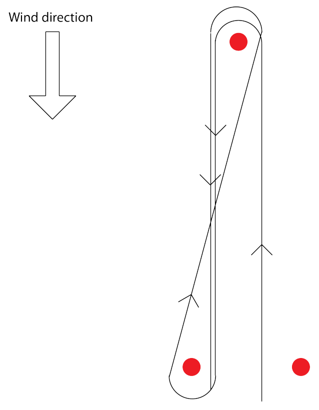


Figure 1.5: *The most simple racing course with two upwind and two downwind legs*

The last topic of this section is the concepts of lay- and leader lines. In figure 1.6, the lay lines are shown as dotted lines. Lay lines are imaginary lines from a mark. When a boat is outside one of the lay lines it can sail directly to the mark and does not need to tack to reach the mark. For instance, the boat marked with C can sail directly to the mark, but boats A and B need to tack at least one more time to get to the mark. The fully drawn line shows where the leading boat is and this line is called the leader line by TracTrac. The leader line rounds off, since outside the lay lines the sailors can sail directly to the mark [2], [3].

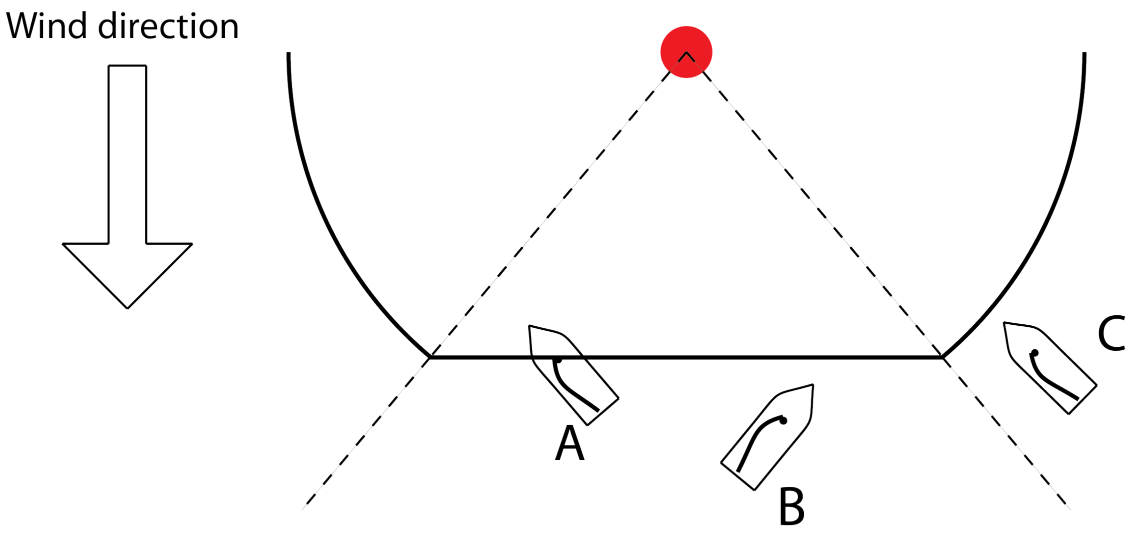


Figure 1.6: *Lay lines (dotted) and leader line (fully drawn)*

1.3 Problem Description

The main goal of the project in this report is to create an algorithm that helps the audience of tracked sailing events understand the decisions and choices of the tracked sailors, by providing information about the wind direction. Furthermore, the proposed algorithm should give a more precise leader board at every time during a race.

During the process of algorithm creation the following problems are addressed in this thesis:

- To analyze the mathematical background and statistics of the boats' directions in fleets.
- To understand the sailing specific principles affecting the direction of the boats.
- To calculate the wind direction from GPS tracking data in real-time.
- To calculate a confidence measure to show the reliability of the wind direction.
- To visualise the wind direction for the end user in the TracTrac race viewer client. (Examples: `http://www.tractrac.com/index.php?page=events&limit=1`)
- To improve the virtual leader board calculations by using the wind estimation.

1.4 Approach

The approach taken to tackle the problems described in section 1.3 is based on four incremental steps. First of all the initial hypothesis is tested: *"Is it possible to calculate a wind direction from a fleet of boats beating the wind?"* This part is done in MatLab and with ideal test datasets. During the last part of the MatLab implementation, non-ideal datasets are used to test the robustness of the algorithm, and to highlight how to solve problems with noise and too little data. Furthermore, a confidence measure is developed to show the viewers the reliability of the wind estimation.

Secondly, the sailing specific considerations are investigated since domain knowledge has a large effect on the solution to the problem presented in this report. A sounding board meeting was conducted with three sailing experts to acquire the sailing specific knowledge needed.

Thirdly, the algorithm is implemented in the Java programming language on TracTrac's live race servers. The complexity of the algorithm is analyzed and the different parts of the algorithm are timed.

The last step is the application of the calculated wind direction. The wind direction is visualized in the race viewer together with the confidence measure. TracTrac's virtual leader board is enhanced by taking the wind direction into account when calculating the ranking of the competitors. The leader line visualisation is also enhanced to take advantage of the calculated wind direction estimation.

1.5 Related Work

GPS-tracking is a new and small business market and there are few GPS-tracking companies world-wide. TracTrac had planned to include calculations of the wind direction for a while prior to the project that forms the basis for this paper. One of TracTrac's competitors tried solving the wind direction problem as a part of the competitors race viewer. Due to the nature of competing business TracTrac only knew that the competitor were working on a solution and that they had not succeeded.

The proposed algorithm of this thesis provides a cost-effective way to display the wind direction to the spectators as the algorithm is based on data that is already available from the tracking units. The proposed algorithm solves an important problem for the tracking companies as the primary goals for these companies is to provide as much information as possible to spectators in sports where the action happens far from the spectators.

The proposed algorithm is only one way to solve the problem. One drawback of calculating the wind direction based on a fleet of boats is that we assume that the sailors always choose the best direction to sail. But due both to the interaction of the wind with other boats and rules and regulations in sailing competitions the sailors cannot always choose the best sailing direction. During the sounding board discussion, described in section 2.2, it was proposed to fit the boats with wind meters to actively measure the wind direction during races. This would involve sending even more data, but could give insight into the wind direction in a fleet of boats. As stated in section 1.3 the main goal of the proposed algorithm is to provide insight into the decisions of the sailors. Therefore a wind direction estimation based on the sailors' observation of the wind is more desirable than a more exact wind direction as TracTrac aims to give insight into the decisions of the sailors.

1.6 Outline

The report is organized as follows: Chapter 2 presents the Convolution Based Direction Filtering (CBDF) algorithm and is organized in four sections. Section 2.1 presents the mathematical model underlying the calculations of the wind direction estimation and forms the basis of the CBDF-algorithm. Section 2.2 describes the recommendation and sailing specific considerations obtained in a sounding board discussion with three sailing experts. We discuss how the expert knowledge is added to the algorithm in order to enhance its functionality. Section 2.3 introduces the real-time algorithm using pseudo code. The time complexity of the proposed algorithm is analyzed. Section 2.4 highlights the applications of the wind direction estimation, this includes visualization and virtual leader board calculations. Chapter 3 discusses the algorithm implementation and the calculated wind direction. Furthermore the validity of the calculated wind direction is discussed with regards to the goals stated in section 1.3. Chapter 4 concludes the report, and points out the limitations of current work, and future work is highlighted.

Chapter 2

Convolution Based Direction Filtering (CBDF)

In this chapter, an algorithm for calculating the current wind direction of a fleet of boats is presented. This chapter follows the four incremental steps described in section 1.4.

2.1 Mathematical Model

As described in the introduction (see section 1.1), when beating the wind, the boats will sail in two distinct direction groups or clusters: starboard and port tack. During the first part of the MatLab implementation the directions data from the tracking units are used to get a first impression of the directions of the boats.

During the MatLab implementation, a dataset with the directions of a fleet of boats was used. The dataset consists of the directions of all the boats for every third second. Figure 2.1 shows one time step's direction data ordered from 1-360 degrees. The two direction clusters are distinct and the figure shows how the typical direction distribution looks. The typical distribution is two clusters where the centers of the two clusters are separated by 30 to 80 degrees based on the type of boat. The directions of the boats are from *"The Swedish Dragon Boat Championship race 2"* and the timestamp is 2011-08-05 09:18:39. This time step's data will be used throughout the evolution of the CBDF-algorithm.

As described above, the two direction clusters are distinct but the challenge is to identify the two clusters and classify "noise" in the data. The "noise" is boats that are not in one of the two clusters. This can happen when boats are tacking and the boats direction is between the two clusters.

The evolution of the CBDF-algorithm was done in four incremental steps and the four approaches will be introduced in the following sections:

- **Maximum-value:** The average of each clusters maximum number of boats is used as the wind estimation
- **Convolution with All-pass Filter:** Convolution with an all-pass filter and the average of the maximum value of each cluster is used as the wind estimation

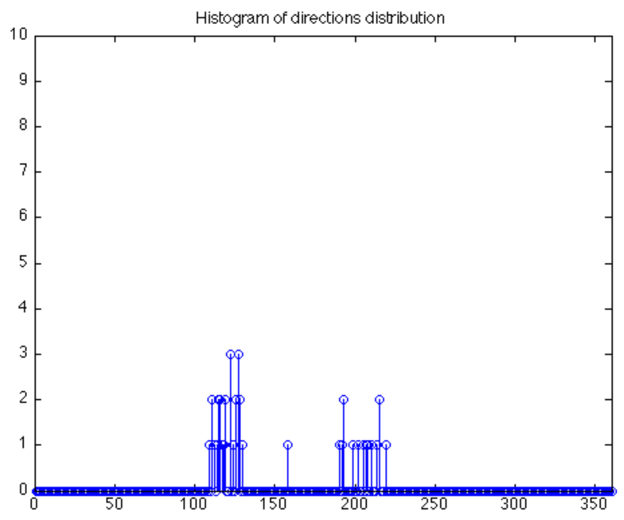


Figure 2.1: *One time step's direction data ordered from 1-360 degrees*

- **K-means:** The k-means algorithm computes the maximum value of the clusters and the average is used as the wind estimation
- **Convolution with Modeled Filter:** Convolution with a filter modeled after the data and the average of the maximum value of each cluster is used as the wind estimation

2.1.1 Maximum-Value

The initial data, as shown in figure 2.1, shows two distinct clusters. To differentiate between the two clusters and the noise, the first approach used was a maximum value approach. The maximum value of the entire dataset is used as the first cluster centroid. The data surrounding the maximum value in a prefixed constant interval of 19 degrees is removed from the dataset. 19 degrees was chosen as 8 degrees on either side of the cluster centroid and was chosen together with TracTrac and based on their sailing experience. The maximum value of the remaining directions is then used as the second cluster centroid. The average of the two cluster centroids is calculated and used as the wind direction estimation.

The maximum-value method gave a good initial confirmation of the hypotheses that a wind direction can be calculated from the direction data of a fleet of boats. But the calculated wind direction varied a lot and was not very stable. The instability of the method depends on the maximum-value moving around in the cluster. In figure 2.1 this can be seen in the right-most cluster, where there is actually two cluster maximum-values. If the maximum-value moves between these to maximum-values for each time step the calculated wind direction varies 5-6 degrees each time step.

2.1.2 Convolution with All-pass Filter

To overcome the weaknesses pointed out for the maximum-value approach, we tested a second approach based on the summation of the vicinity from the ordered data shown in figure

2.1.

Assume that

- i is a possible direction in degrees, within the interval $1 \leq i \leq 360$
- g is the filter function, more concretely an all-pass filter [5] with coefficient 1
- f is the directions distribution as shown in figure 2.1
- k is half the size of the filter

Then, equation 2.1 represents the mathematical formula for the summation of the vicinity.

$$\text{vicinity}(i) = \sum_{j=-k}^{k} g(j) \cdot f(i+j) \qquad \text{, where } 1 \leq i \leq 360 \tag{2.1}$$

The 1-D convolution with an all-pass filter results in an array as shown plotted in figure 2.2:

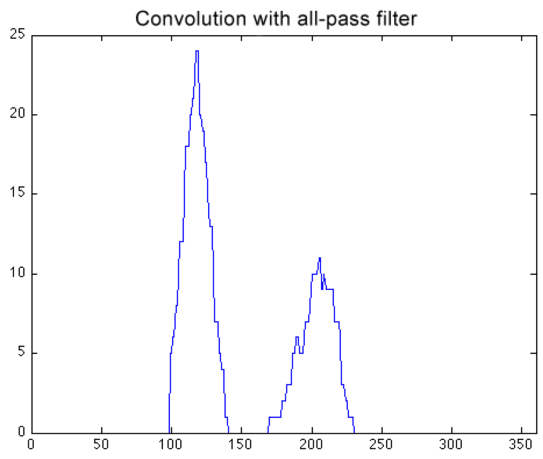


Figure 2.2: *Vicinity summation with two distinct clusters and shown as a continuous function for clarity*

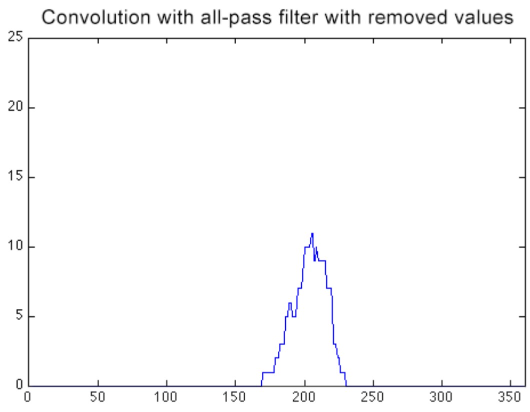


Figure 2.3: *The distribution with data around the first centroid deleted. This shows clearly the second tack*

The maximum-value of the result of the convolution in formula 2.1 is used as the centroid of the first cluster. A new array is saved with the same contents, but the data in the same prefixed constant interval of 19 degrees (the same interval as used in section 2.1.1) around the centroid is removed in the new array. This creates the distribution in figure 2.3, where the maximum-value is used as the centroid for the second group of boats. The two cluster groups are stored as the data in the prefixed constant interval around the two centroids. The clusters are used for plotting (to evaluate the results) and for calculations of the statistical measures used in the confidence calculations described in 2.1.5. The mean of the two centroids are calculated and used as the estimation of the wind direction.

2.1.3 K-means

To validate the approach used in 2.1.2 the third approach is a statistical clustering method called k-means [6], [7], [8]. K-means is a multivariate clustering algorithm, that partitions n observations (d-dimensional) into c clusters by minimizing the within-cluster sum of squares. The clusters are initialized with c random seed points within the domain of the observations. The Euclidian distance of each observation to the clusters is calculated and the observations are assigned to the cluster defined by the closest seed point. New cluster centroids are calculated as the mean value of the observations assigned to each cluster. The process of calculating the distance to centroids and assigning observations to clusters is repeated until the difference in the within cluster sum of squares between two repetitions is below a given threshold or a stable solution has been found.

MatLab comes with an implemented version of the k-means algorithm. To compare the convolution with the all-pass filter described in section 2.1.2 with the MatLab k-means implementation, the Matlab k-means implementation is used both on the raw directions data and on the directions in the two clusters after the convolution with the all-pass filter described in section 2.1.2. The mean value of the k-means clusters is used as the wind direction estimation. The two k-means wind direction estimations are compared to the all-pass filter estimations described in section 2.1.2. The wind direction estimations are plotted for each sample of an entire race in figure 2.4:

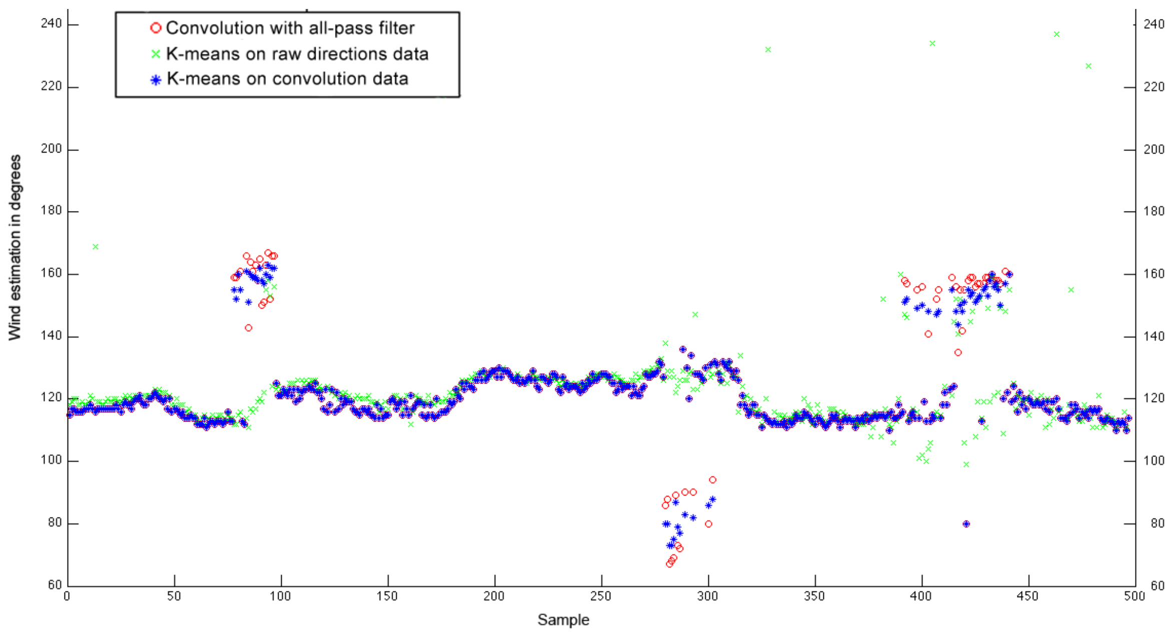


Figure 2.4: *Wind direction estimations for convolution with all-pass filter, k-means on raw directions data and k-means on convolution data*

Figure 2.4 shows that the wind estimation based on the k-means clusters of the convolution data and the wind estimation of the convolution data on its own is very similar and only differs on the three occasions during the race shown in figure 2.4 where the algorithms fails to calculate a stable wind direction (around samples 80-100, 280-310 and 390-450). The estimations from the k-means clustering on the raw directions data varies from the two other as all data points are assigned to a cluster and thus the cluster centroids and the mean values is different from the two other methods. As we want the wind direction to be based only on the boats on the two tacks k-means clustering on the raw direction data is discarded as a suitable method for calculating a wind direction.

Preparing for the real-time Java implementation of the CBDF-algorithm, a Java k-means algorithm optimized for two clusters and few data points was developed. The k-means algorithm is used for large multivariate datasets. The small datasets and two well defined clusters means that the difference in within-cluster sum of squares becomes smaller than the threshold before the actual cluster centroid is found. Due to the similarity of the k-means filtered data and the convolution with the all-pass filter the remaining part of section 2.1 is about modelling the filter function g of equation 2.1 to the data.

2.1.4 Convolution with Modelled Filter

The best case scenario for the data is two well defined clusters surrounding the two close haul directions (see appendix A). The mathematical model described in section 2.1.2 is not optimized for the best case scenario and all directions within the constant interval of 19 degrees, the same interval as in section 2.1.2, contributes equally to the convolution value for each direction. Changing the filter g to one that resembles the best case scenario of two well defined close haul directions should yield better results, as the directions closest to the middle of the filter contributes more to the convolution value than the directions further from the two close haul directions. The only difference between the model described in section 2.1.2 and equation 2.1 and the model described below is the filter function g.

Assume that

- i represents a possible integer direction in degrees, within the interval $1 \leq i \leq 360$
- g is a pyramid-shaped filter function, as shown in figure 2.5
- f is the directions distribution as shown in figure 2.1
- k is half the size of the filter

Then, equation 2.2 represents the convolution used in the CBDF-algorithm:

$$\text{convolution}(i) = \sum_{j=-k}^{k} g(j) \cdot f(i+j) \qquad \text{, where } 1 \leq i \leq 360 \tag{2.2}$$

The first cluster centroid is selected as the maximum value of the array. The convolution values in the 19 degree prefixed constant interval around the cluster centroid are removed and the maximum value of the remaining directions is used as the second cluster centroid. The wind direction estimation is calculated as the mean of the two centroids.

The data set plotted in figure 2.1 is used again and figure 2.6 shows the result of a convolution with a pyramid shaped filter like the one shown in figure 2.5. Figure 2.7 shows the convolution data after the convolution values has been removed in the 19 degree prefixed interval around the first cluster centroid. Comparing figure 2.2 with 2.6 and 2.3 with 2.7 we can conclude the following. First, more distinct maximum values can be seen around 115 and 205 degrees in figure 2.6 than in figure 2.2. In figure 2.2 the convolution value is the same for 3 degrees

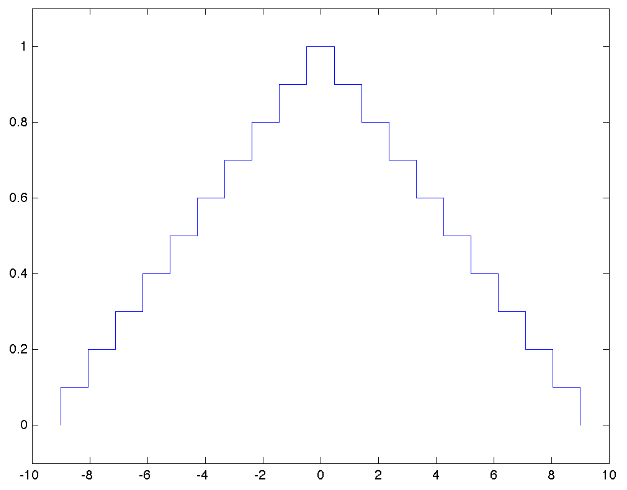


Figure 2.5: *Pyramid shaped filter used in convolution*

in a row at the maximum value around 115 degrees. Second, the convolution with all-pass filter results in many areas where the convolution values are the same for many directions in a row, for instance from 170-175 degrees and from 215-220 degrees in figure 2.2. When the maximum values are selected the convolution values are traversed from 0 to 360 degrees so only the maximum value with the smallest direction value is used. By using a filter that is modelled after the ideal situation these flat areas can be more or less eliminated as can be seen in figures 2.6 and 2.7.

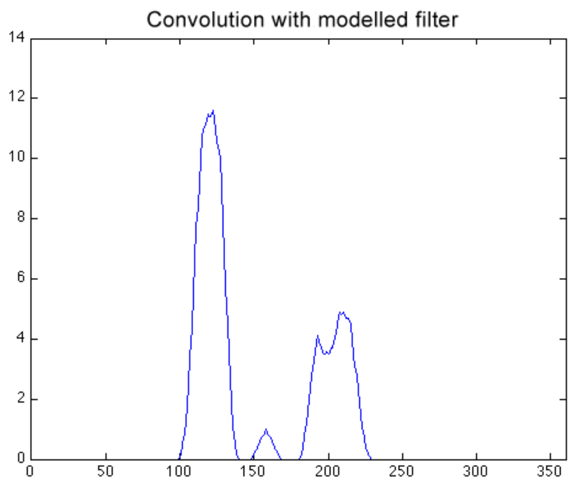


Figure 2.6: *Result of convolution between the directions distribution and pyramid shaped filter*

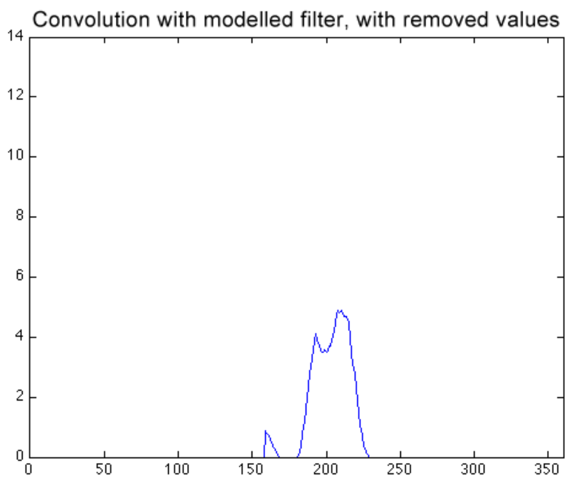


Figure 2.7: *The distribution with deleted data around the first centroid. This shows clearly the second tack*

A bell shaped filter function ($\sigma = 1$) was tested simultaneously as the pyramid shaped filter [9]. Figure 2.8 shows a comparison of the wind direction estimations for an entire race for the two filters. The wind direction estimations for the pyramid shaped filter follows a smoother curve and consists of less noise and fewer outliers. Therefore, the pyramid shaped filter is used in the real-time Java implementation.

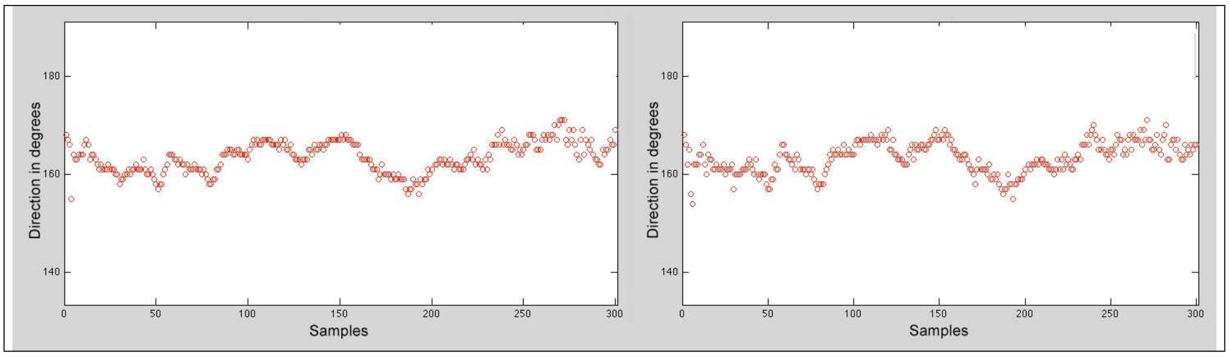


Figure 2.8: *Pyramid shaped filter on the left compared to a bell shaped filter on the right*

The final CBDF-algorithm, as described in section 2.3, is a real-time Java implementation of the algorithm described in this section. The filter function in the CBDF-algorithm is modelled on the knowledge of the two clusters and the behaviour of the experienced sailors. The next section describes an internal verification method that is used to separate good wind direction estimations from estimations where the algorithms fails to calculate a reasonable estimation.

2.1.5 Confidence Measure

The confidence measure is used to discriminate between good and bad wind direction estimates. The confidence measure is based on the standard deviations of the two clusters. The two clusters are defined as the boats sailing in a direction contained within the pyramid shaped filter centered around the cluster centroid. The confidence calculation method is only used if the difference between the two cluster centroids is between 60 and 170 degrees. The 60 to 170 degree interval was defined together with the sailing experts at the sounding board meeting described in section 2.2 and based on the knowledge of the different points of sail described in figure 1.3.

Formula 2.3 below defines the confidence measure. The output of the confidence measure is a constant with the value of 5, 3, 1 or -1, the higher the value the better the confidence. The value is used by the TracTrac race client to discriminate between good and bad wind direction estimations. The confidence measure is based on the fraction of boats outside the two clusters and the total number of boats and the standard deviation of the two clusters. The standard deviation is a good measure of how close the boats in each cluster is to the cluster centroid and the smaller the value the closer the boats are to each other.

Assume that

- σ is the standard deviation of the clusters
- n number of boats on the current leg
- p number of boats not in a cluster, $p \leq n$

then equation 2.3 shows the calculation of the confidence measure:

$$\text{confidence} = \begin{cases} 5 : \sigma - p/n < 2.5 \\ 3 : \sigma - p/n < 4.5 \\ 1 : \sigma - p/n < 6.5 \\ -1 : \sigma - p/n \geq 6.5 \end{cases} \tag{2.3}$$

2.2 Adding Expert Knowledge

As a part of the early evaluation, a team of three sailing experts contributed to the project by attending a sounding board discussion. The discussion gave insight into the sailing specific interactions of boats in a fleet. In this section, it is discussed how these sailing specific effects are incorporated into the CBDF-algorithm.

The two main points obtained from the sounding board discussion are described below:

- The turbulence created by sails at the front of the fleet when beating to windward.
- The tactical decisions and rules governing tacks close to marks.

Figure 2.9 shows how sails at the front of the fleet create turbulent winds. The turbulent winds force the boats further down to either tack to get away from the turbulent winds or sail at a bigger angle to the wind [2, pp. 160-167].

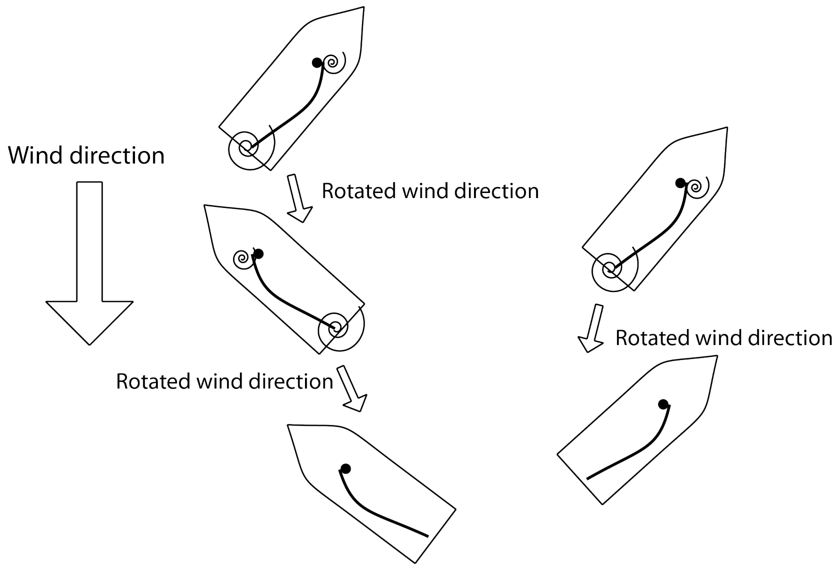


Figure 2.9: *The turbulence in the wind introduced by sails in front. The rotated wind directions are exaggerated for clarity*

The turbulent winds are accounted for by weighting the wind direction of the boats at the back of the field lower than the boats at the front of the field with the function shown in equation 2.4 presented below. Formula 2.4 is based on a recommendation by the sailing experts. The best quarter of the fleet is weighted full (by 1), the two middle quarters are weighted linearly based on the position and the last quarter is weighted by 0.1. The slope of the linear function depends on the number of competitors as the final weight is always between 1 and 0.1.

Assume that

- $rank$ is the ranking of the boat in the previous time step
- $quarter$ is one fourth of the size of the fleet
- $w_1 = 1$ is the weight for the leading quarter of the fleet
- $w_2 = 0.1$ is the weight for the last quarter of the fleet
- $slope = \frac{w_2 - w_1}{3 \cdot quarter - 1 \cdot quarter} = \frac{-0.9}{2 \cdot quarter}$
- $\mathit{offset} = w_2 - slope \cdot quarter = 0.1 - slope \cdot quarter$

then equation 2.4 calculates the weighting for each based on the ranking:

$$weight = \begin{cases} 1, & \text{if } rank < quarter \\ slope \cdot rank + \mathit{offset}, & \text{if } quarter \leq rank \leq 3 \cdot quarter \\ 0.1, & \text{if } rank > 3 \cdot quarter \end{cases} \tag{2.4}$$

The second part of the experts' recommendation, tactical sailing at the marks, is difficult to extract from the regular sailing manoeuvres performed during other parts of the race. At the mark the boats will be close together as all boats need to sail around the same mark. The fact that the boats are close together and sail towards the same mark makes it easier for the audience to see the current standing of the boats in the fleet. The CBDF-algorithm was implemented to help the audience understand sailing decisions. Furthermore the estimated wind direction is an important part of calculating a more accurate leader board in situations where the boats are not in tactical situations like starts and mark roundings. TracTrac and the author decided not to implement any specific handling of these tactical sailing decisions as it is outside the scope of both the project and the problems described in section 1.3.

2.3 Real-Time Convolution Based Direction Filtering

This section describes the real-time algorithm for calculating an estimate of the wind-direction. The algorithm will be used during live competitions and during playbacks after competitions. The constraints introduced by these two applications are a low time-complexity and the ability of the algorithm to receive and process live data as it becomes available during competitions. Firstly, we give an overview of the Java class and their interdependencies. Secondly, we present the algorithm in pseudo code. Lastly, we analyze the time complexity of the algorithm.

2.3.1 Description of the TracTrac Java Client

This section describes briefly the classes and interfaces used by the wind direction estimation algorithm. A subset of the data model for the TracTrac Java client is presented in figure 2.10 as a UML [10, Section 6] class diagram.

1. The `ITimeUpdateListener` interface has the `timeChanged` method invoked every time step. The time step and therefore the frequency of calls to the `timeChanged` method changes based on client processing power and bandwidth.

2. The `BoatRaceData` class implements the `ITimeUpdateListener` interface and has a `WindDirectionEstimator` instance variable.

3. The `WindDirectionEstimator` class implements the `ITimeUpdateListener` interface and has an instance variable `resultItemBoatCompetitorList` with a list of all the competitors in the specific race. The `WindDirectionEstimator` class has three important methods apart from the `timeChanged` method:

   (a) `legSortingAndWeighting`: The competitors are sorted in descending order according to the boat's current leg. The two remaining parts of the algorithm (marked (b) and (c) below) is executed for each leg. The competitors are weighted based on their ranking if the number of boats on a leg exceeds a constant threshold because of the turbulence created by the sails.

   (b) `calculateEstimation`: Loops through all competitors, calculates and stores the direction, leg, and ranking of each competitor.

   (c) `confidenceCalcualtions`: First, the wind estimation is calculated as described in 2.1.4. The confidence measure is then calculated, as described in 2.1.5. The wind direction estimation and confidence values are stored in each `ResultItemBoatCompe` instance.

4. The `ResultListBoat` class is the result list class and it calculates the leader board. It has a `WindDirectionEstimator` instance variable as the wind direction is used in the leader board calculations.

5. The `ResultItemBoatCompetitor` is the class for each boat in the specific race and, besides the two instance variables `windDirectionOnCurrentLeg` and `windConfidenceOnCurrentLeg`m it has the direction and the leg for each boat.

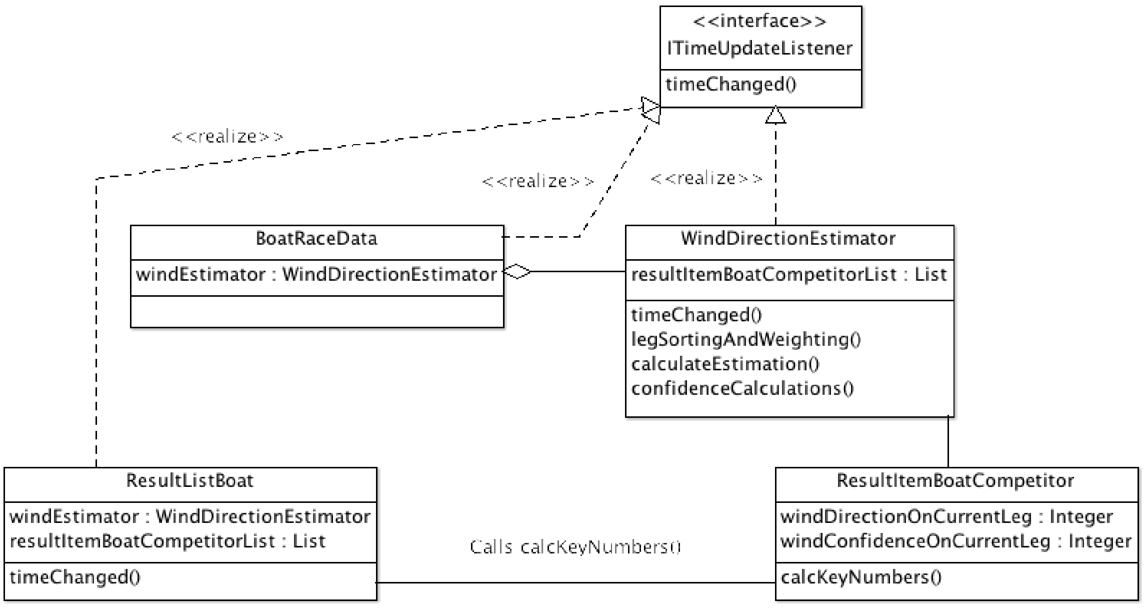


Figure 2.10: *UML class diagram showing the dependencies of the WindDirectionEstimator class*

In the following section the implementation of the three main methods of the `WindDirectionEstimator` class are described with pseudocode and the most interesting parts of the algorithm are explained.

2.3.2 Algorithm Implementation

In this section, the three main steps marked with (a), (b) and (c) described in section 2.3.1 are shown in pseudo code and explained. In the following, the `ResultItemBoatCompetitor` class described in figure 2.10 is denoted as `competitor`. The algorithm described in algorithm 2.1 calculates the sailing direction for each competitor using the equirectangular projection [11, pp. 5-8]. The equirectangular projection is used to transform meridians to vertical lines with equal spacing. The equirectangular projection is used in our particular case to create a coordinate system and calculate the direction of movement for each competitor.

Assume that

- $long$ is the longitude of the GPS data point
- lat is the latitude of the GPS data point
- $lat1$ is the standard parallel
- $earthradius$ is the radius of the earth equal to 6372795 meters
- x is the projected x-coordinate
- y is the projected y-coordinate

then equation 2.5 calculates the projected coordinates:

$$\begin{aligned} x &= long \cdot \cos(lat1) \cdot earthradius \\ y &= lat \cdot earthradius \end{aligned} \tag{2.5}$$

Algorithm 2.1 describes the calculation of each boat's direction. The direction θ is calculated as $\tan\theta = \frac{y_2 - y_1}{x_2 - x_1}$, where x_1, x_2 and y_1, y_2 are the projected coordinates. Algorithm 2.1 get a list of all competitors with GPS-position for both the current and previous time step as input. The output of algorithm 2.1 is a list with the direction of each competitor.

Algorithm 2.1 Direction Calculation

```
 1: Initialize list
 2: for all competitors do competitor
 3:     if competitor has started then
 4:         Get the current GPS position for competitor
 5:         if Current GPS not null then
 6:             Get current time step's GPS coordinates
 7:             Get previous time step's GPS coordinates
 8:             Project GPS coordinates using the equirectangular projection as in 2.5
 9:             Calculate direction $\theta$ of competitor's movement
10:             Store competitor and direction $\theta$ in list
11:         end if
12:     end if
13: end for
```

The output `list` from algorithm 2.1 is used as input to algorithm 2.2 where the competitors are sorted based on their current leg and weighted based on the ranking of the `competitor`. The method is called recursively until there are no competitors left in the `remainingCompetitorsList`. The output of algorithm 2.2 is one or more lists with sorted and weighted competitors that is used as input to algorithm 2.3.

Algorithm 2.2 Leg Sorting and Weighting

```
initialize maxLegList
initialize remainingCompetitorsList

Set maxLeg to lowest possible integer value
for all competitors in list do competitor
    if current leg of competitor is greater than maxLeg then
        Set maxLeg as current leg of competitor
    end if
end for

for all competitors in list do competitor
    if current leg of competitor is equal to maxLeg then
        if size of list is greater than threshold then
            Weight competitor based on rank of competitor        ▷ Equation 2.4
            Store direction and weight in maxLegList
        else
            Store direction and weight = 1.0 in maxLegList
        end if
    else
        Store competitor in remainingCompetitorsList
    end if
end for

Call Estimation and Confidence method for competitors in maxLegList

if remainingCompetitorsList is not empty then
    list = remainingCompetitorsList
    Call Leg Sorting and Weighting with list
end if
```

The last part of the CBDF-algorithm is estimation and confidence calculations. The algorithm is described mathematically in section 2.1.4 and programmatically in algorithm 2.3. The two cluster centroids are picked as two maximum values of a 1-D spatial convolution of the distribution of the directions.

Algorithm 2.3 Estimation and Confidence

```
Initialize distribution
Initialize convolution
FS = 8
Initialise filter with size equal to 2·FS+1

for all competitors in maxLegList do competitor
    Get competitor's weight
    Get competitor's direction
    Increment direction in distribution by weight
end for

Initialize sum
for i= 0 → 359 do                                          ▷ For all degrees
    sum is reset to zero
    for j= -FS → FS do                                     ▷ For every filter constant
        sum += distribution(i+j) · filter(j)
    end for
    convolution(i) = sum
end for

Set centroid1 as the maximum value of convolution
Remove data from convolution in interval with size 2·FS+1 around centroid1
Set centroid2 as the maximum value of convolution

Set estimation as mean value of centroid1 and centroid2
Do confidence calculations for the wind estimation         ▷ Described in section 2.1.5

for all competitors in maxLegList do competitor
    Set competitor's windEstimation as estimation
    Set competitor's windConfidence as the calculated confidence
end for
```

In the next section time-complexity analysis is performed for the three parts of the algorithm highlighted above.

2.3.3 Time Complexity Analysis

The time-complexity analysis conducted in this section is based on the model described in [12, Chapter 2] and [13]. The model is a basic computer that executes instructions sequentially. The model consists of a set of simple instructions, like assignment and comparison, and simple mathematical operations, like addition, subtraction, multiplication, and division. It is also assumed that all of these instructions are executed in one time unit despite the difference in actual execution time. It is assumed that the model has infinite memory. This model significantly simplifies the algorithm analysis and gives good insight into the growth rate of the algorithm.

The three parts of the CBDF-algorithm Direction Calculation, Leg Sorting and Weighting, and Estimation and Confidence are called sequentially making the running time of the algorithm a sum of the three parts.

Assume that

- n is the total number of competitors
- $T_{dir}(n)$ is the running time of algorithm 2.1
- $T_{sort}(n)$ is the running time of algorithm 2.2
- $T_{est}(n)$ is running time of algorithm 2.3

then $T_{total}(n)$ denotes the combined running time of CBDF-algorithm

$$T_{total}(n) = T_{dir}(n) + T_{sort}(n) + T_{est}(n) \tag{2.6}$$

Algorithm 2.1: This part of the algorithm loops through all of the $n > 0$ competitors and calculates the direction of travel for each competitor. The direction is calculated using the equirectangular projection described in [11, pp. 5-8] and equation 2.5. Firstly $list$ is initialized. In the model described above all initializations run in constant time, i.e. $\mathcal{O}(1)$. Inside the competitor loop, the GPS-position getter-method of the competitor object is used to get the previous and current time step's GPS coordinates. These values are stored as instance variables of the competitor object and the getter-methods run in constant time. The projection using the equirectangular-projection on line 8 in algorithm 2.1 are multiplications and trigonometric functions all running in constant-time. Calculating the direction from the projected coordinates is done as described in equation 2.5 and involves two subtractions, a division, and the use of the trigonometric function $\tan\theta$. All three functions run in constant time $\mathcal{O}(1)$. The last step of algorithm 2.1, on line 10, stores the calculated values at the end of $list$. Storing values at the end of a list is an operation that runs in constant time. Since the direction of travel can be calculated in constant time for each competitor, the time complexity of algorithm 2.1 is $\mathcal{O}(n)$.

Algorithm 2.2: This part of the CBDF-algorithm is a recursive method. The method separates the competitors into groups for each leg of the race. Lines 1 and 2 initializes two list data structures and as above initializations run in constant time. On line 4 the $maxLeg$ variable is initialized and assigned the value of the lowest possible integer value. Initializations and assignments both run in constant time. On lines 5-9, the competitor list is looped and if the current leg of the competitor is greater than $maxLeg$ then $maxLeg$ is assigned the value of the

competitors current leg. The statements inside the for-loop both run in constant time according to the model described above. Thus, the for-loop on lines 5-9 has a running time of $\mathcal{O}(n)$.

The for-loop on lines 11-22 does the actual division of the competitors into the two groups. The if-statement on lines 12-21 check whether or not the boat is on the maximum leg of the race. If the boat is currently on the maximum leg of the race, then the boat is weighted based on ranking and stored in the $maxLegList$ list. The weighting of the boats is only performed for fleets greater than a threshold, as the wind turbulence described in section 2.2 does not affect small fleets in the same way as big fleets. The weighting of the competitor consists at the most of a multiplication and an addition, and thus runs in constant time. If the competitor is not on the maximum leg of the race the competitor is stored at the end of the $remainingCompetitorsList$, storing at the end of a list runs in constant time.

The last part of the CBDF-algorithm (Algorithm 2.3) is now called for the competitors stored in the $maxLegList$. All if-statements and storing the competitors in the list data structures run in constant time according to the model described above.

The if-statement on lines 26-28 calls recursively algorithm 2.2. The method is called again with the boats not on the maximum leg, i.e. the boats in $remainingCompetitorsList$.

Assume that

- T_{sort} is the running time of algorithm 2.2
- $n > 0$ is the number of competitors
- $p \geq 0$ is the number of competitors not on the maximum leg, $p < n$
- c is constant running times

in the worst-case, where $p = n - 1$, the running time of algorithm 2.2 is

$$\begin{aligned} T_{sort}(n) &= c \cdot n + T_{sort}(p) \\ &= c \cdot (n) + c \cdot (n-1) + c \cdot (n-2) + ... + c \cdot (1) \\ &= c \sum_{i=1}^{n} i \\ &= \mathcal{O}(n^2) \end{aligned} \tag{2.7}$$

Formula 2.7 describes the worst case scenario for algorithm 2.1. This situation occurs when the $n > 0$ boats are on n different legs. This worst case scenario rarely occurs as the number of legs are smaller than the number of boats in almost all the races where TracTrac delivers race tracking. As described in section 1.2 most races consist of four legs and from five to upwards of 150 boats. For the entire first leg all boats are on the same leg. So only the first multiplication $c \cdot (n)$ of line 2 of formula 2.7 remains and the running time of algorithm 2.1 is $\mathcal{O}(n)$. During real races the number of legs are far less than the number of competitors. As described in section 1.2 and figure 1.5 the normal number of legs is four. Thus, the running time of the CBDF-algorithm during a realistic scenarios is $\mathcal{O}(n)$. Formula 2.8 calculates the time-complexity of a worst-case realistic race with four legs and a fourth of the competitors on each leg.

Assume that

- T_{sort} is the running time of algorithm 2.2
- $n > 0$ is the number of competitors
- c is constant running times
- and $\sum\limits_{i=0}^{k} \frac{1}{2^i} = 2 - \frac{1}{2^k}$

then the running time of algorithm 2.2 is

$$\begin{aligned}
T_{sort}(n) &= c \cdot \frac{n}{4} + T_{sort}(\frac{3n}{4}) \\
&= c \sum_{i=1}^{\log n} \frac{n}{2^i} \\
&= c \cdot n \sum_{i=1}^{\log 1} \frac{n}{2^i} \\
&= n \cdot (2 - \frac{2}{2^{\log n}}) \qquad = \mathcal{O}(n)
\end{aligned} \tag{2.8}$$

Algorithm 2.3: The last part of the CBDF-algorithm is the actual convolution and estimation of the wind direction as the mean value of the two clusters. This part of the algorithm can be called multiple times depending on how many legs the competitors are currently on. The initializations on lines 1, 2, 3, 4, 12, and 14 all run in constant time. The assignments and deletion of data on lines 21-23 run in constant time as the size of the interval is constant, the size of convolution is constant and the deletions are done by index. The statement on line 25 is an addition and a division by 2 and runs in constant time.

The confidence calculation involve looping all the boats in each cluster to calculate the standard deviation so the confidence calculations on line 26 runs in linear time.

The for-loop performing the convolution on the wind direction distribution on lines 13-19 has a constant running time, since the size of both for-loops is not dependent on the size of the fleet and that the statements inside the for-loops all run in constant time.

The remaining parts of algorithm 2.3 are the for-loops on lines 6-10 and lines 28-31 respectively. Both for-loops loop through all competitors in the $maxLegList$. The for-loop on lines 6-10 gets the weight and direction from the $maxLegList$ and increments the value of the direction in $distribution$. All these statements run in constant time. The for-loop on lines 28-31 uses the public setter-methods of the competitor object to set the two instance variables $windEstimation$ and $windConfidence$ of the competitor object, both assignments run in constant time.

Assume that

- T_{est} is the running time of algorithm 2.3
- $n > 0$ is the number of competitors

- m is the number of competitors on the maximum leg, $m \leq n$
- c_c is the constant running time of the convolution
- c_i is the constant running time of instantiations
- c are constant running times

then the total running time of algorithm 2.3 is

$$T_{sort}(m) = c_i + c_c + (c \cdot m) + (c \cdot m) + (c \cdot m) = c + \mathcal{O}(m) = \big(m\big) = \mathcal{O}(n) \tag{2.9}$$

The resulting running time of $\mathcal{O}(n)$ is the maximum running time as $m \leq n$ and despite the fact that the competitors can be on multiple legs the sum of competitors will always be n.

Total running time: The worst-case total running time of algorithm is calculated based on the analysis above and equation 2.6:

$$T_{total}(n) = T_{dir}(n) + T_{sort}(n) + T_{est}(n) = \mathcal{O}(n) + \mathcal{O}(n^2) + \mathcal{O}(n) = \mathcal{O}(n^2) \tag{2.10}$$

2.3.4 Performance Measurements

The timing of the algorithm was done on a 2012 MacBook Pro with a 2.3 GHz Intel Core i7 Quad Core processor and 8 GB 1600 MHz DDR3 RAM, running Mac OS X Lion 10.7.5 and Java 7 Update 9. The timing was done with a method call to `System.nanoTime()` which returns the current time in nanoseconds. The method calls are placed at the beginning and end of each part of the algorithm and the difference between the end and start value is used as the running time, i.e. elapsed real time is measured. Table 2.1 shows the average running time for an entire first leg for three different fleet sizes.

Table 2.1: *Timing results for the wind direction estimation algorithm, all values are given in nanoseconds*

Race	Direction Calculation	Leg Sorting and Weighting	Estimation and Confidence	Total
A ($n = 10$)	18300	3400	63600	85300
B ($n = 44$)	24600	3500	64400	92500
C ($n = 140$)	56200	7500	59800	123500

In table 2.1 the A race is the 2012 Kieler 470 Women's Medal Race, the B race is the 2011 Swedish Dragon Championship Race 1, and the C race is the 2012 OK Dinghy World Championship Race 4.

The timing results in table 2.1 does not clearly show the growth rate of the algorithm, but give insights into the actual running time of the algorithm. For the direction calculation step (Algorithm 2.1) the algorithm analysis showed a running time of $\mathcal{O}(n)$. The timing results show an increase in running time when n increases. Accounting for the time used to instantiate the methods and the running time of instantiations and other operations that run in constant time, the $\mathcal{O}(n)$ running time of the Direction Calculation part is confirmed by the timing results.

There is no clear conclusion to the timing results of the Leg Sorting and Weighting part (Algorithm 2.2) of the CBDF-algorithm. The timed situation, all boats on a single leg, has a running time of $\mathcal{O}(n)$, but this is not reflected in the timing results. What can be concluded is that the one part of the algorithm that has a worst case running time of $\mathcal{O}(n^2)$ accounts for approximately 5% of the total running time.

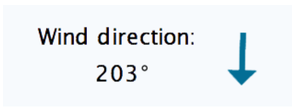
The analysis for the Estimation and Confidence part (Algorithm 2.3) of the CBDF-algorithm showed a $\mathcal{O}(n)$ running time, but this is not reflected in the timing results. The inconclusive results is due to the actual running time of the for-loop on lines 13-19 in algorithm 2.3. The for-loop performs $360 * 19 = 6.840$ multiplications and additions. The for-loop does not depend on the number of boats (n) and takes equally long time for race A, B and C. The decrease in running time from races A on B to the C race is not easy to explain, but we suspect it might be due to other processes running on the computer during the timing of races A and B.

2.4 Applications

This section highlights the applications of the wind direction estimation calculated with the algorithm described in section 2.3.2. The three main applications are a wind rose, the rotation of the leader lines so that the leader line is centered around the wind direction, and leader board improvements.

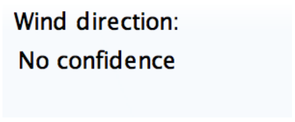
2.4.1 Wind Rose

Figures 2.11 and 2.12 show the wind rose for the situations of where the confidence is above 0 and when it is below 0. The confidence measure is described in section 2.1.5. The wind rose consist of an arrow showing the current wind direction and the wind direction in degrees.



Wind direction:
203°

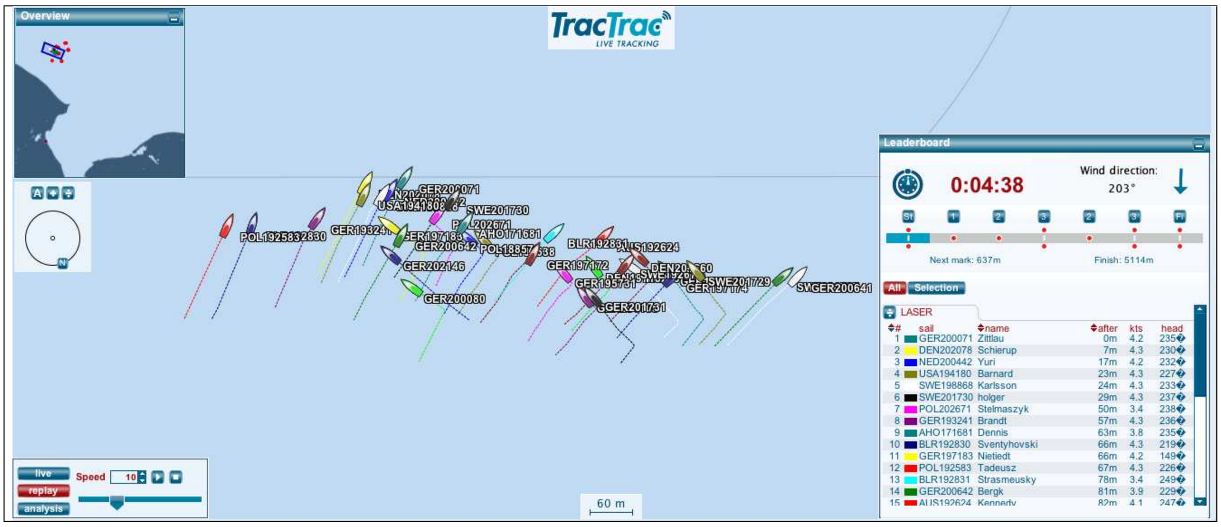
Figure 2.11: *Wind rose when confidence is above zero*



Wind direction:
No confidence

Figure 2.12: *Wind rose when confidence is below zero*

The wind rose is implemented as a part of the leader board and is placed next to the race clock shown in figure 2.13.



Leaderboard

0:04:38

Wind direction:
203°

St 1 2 3 2 3 Fi

Next mark: 637m   Finish: 5114m

All Selection

LASER

#		sail	name	after	kts	head
1	🟩	GER200071	Zittlau	0m	4.2	235�
2	🟨	DEN202078	Schierup	7m	4.3	230�
3	🟦	NED200442	Yuri	17m	4.2	232�
4	🟫	USA194180	Barnard	23m	4.3	227�
5	⬜	SWE198868	Karlsson	24m	4.3	233�
6	⬛	SWE201730	holger	29m	4.3	237�
7	🟪	POL202671	Stelmaszyk	50m	3.4	238�
8	🟪	GER193241	Brandt	57m	4.3	236�
9	🟩	AHO171681	Dennis	63m	3.8	235�
10	🟦	BLR192830	Sventyhovski	66m	4.3	219�
11	🟨	GER197183	Nietiedt	66m	4.2	149�
12	🟥	POL192583	Tadeusz	67m	4.3	226�
13	🟦	BLR192831	Strasmeusky	78m	3.4	249�
14	🟩	GER200642	Bergk	81m	3.9	229�
15	🟥	AUS192624	Kennedy	82m	4.1	247�

Figure 2.13: *A fleet tracked during the 2012 Kieler Woche in the Laser class*

2.4.2 Leader Board Improvement

The TracTrac leader board is shown as a part of figure 1.1 and the leaderboard shows the current standings in the race. Figure 2.14 illustrates one of the sailing specific concepts used to calculate the leaderboard. As the three paths $P1$, $P2$, and $P3$ are all the same length and the boats need to travel equal distances the leader board calculations can be significantly simplified. Firstly, the method used prior to the implementation of CBDF-algorithm is described and, secondly, it is compared to leaderboard calculations after the implementation of the CBDF-algorithm.

The race committee will try to set a course where the upwind legs are exactly the opposite of the wind direction, but this can be hard during wind jumps or wind turns and the limited possibility of changing the course during a race. This results in legs of the race where wind comes at an angle to the upwind legs.

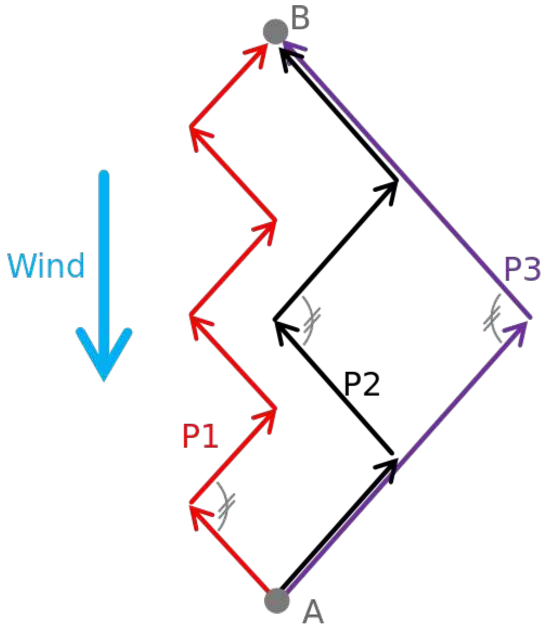


Figure 2.14: *The concept of tacking and distance to the goal of the boat. All the paths have the same length*

The leaderboard calculations are separated into two cases:

1. Boats inside the lay lines
2. Boats outside the lay lines

For case 1, figure 2.16 shows the difference between the Leader Board Calculation before and after the implementation of the CBDF-algorithm. The figure shows three boats marked A, B, and C. Prior to the implementation of the CBDF-algorithm the boats' position were projected on to the line between the two marks of the current leg. In figure 2.16 MP-A, MP-B, and MP-C denote the projected position of the boats. The boats are then sorted based on the projected position and the resulting list is used as the leaderboard. Figure 2.16 show that the current classification is Boat A, Boat B, and Boat C.

After the implementation of the CBDF-algorithm the boats' position is projected on to the calculated wind vector instead of the line between the two marks. WP-A, WP-B, and WP-C denote the projected position of the boats on to the wind vector. The current classification have now changed to Boat C, Boat A, and Boat B. The boat that was last in the classification when the boats' position is projected onto the line between the marks is leading the race when the position is projected onto the wind vector.

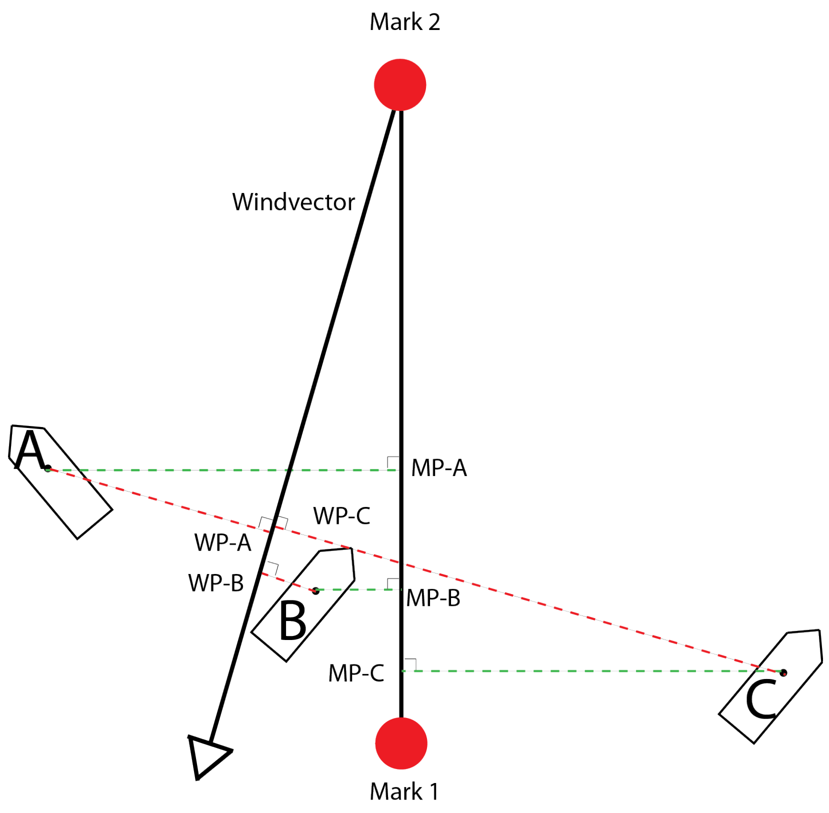


Figure 2.15: *The two different projections: onto the leg direction (green dotted line) and onto the wind direction vector (red dotted line)*

For case 2, outside the lay lines the distance to the next mark is calculated as in equation 2.11 because of the fact that the boats can sail directly to the mark when outside the lay lines.

Assume that

- $\sqrt{dx^2 + dy^2}$ is the euclidean distance from the boat to the mark
- $\cos(\pi \cdot 0.25)$ is the cosine value of 45 degrees

then equation 2.11 calculates the distance to the mark:

$$distance = \sqrt{dx^2 + dy^2} \cdot \cos(\pi \cdot 0.25) \tag{2.11}$$

Figure 2.16 show the situation where the boats are outside the lay lines and can sail directly to the mark.

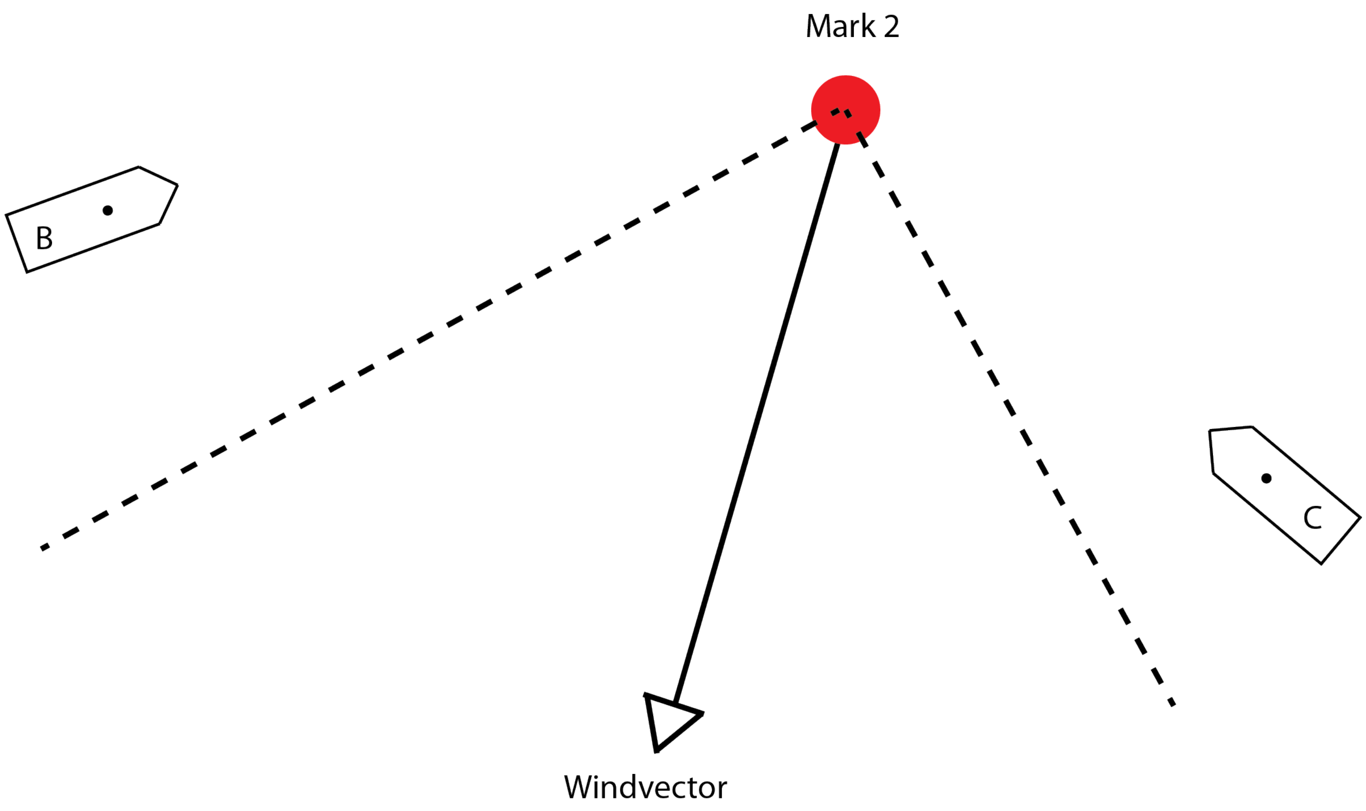


Figure 2.16: *The boats are outside the lay lines and can sail directly to the mark*

2.4.3 Leader Line Visualization

Together with the wind rose the most apparent visual indicator of the wind direction and the current standings is the leader line. Figure 2.17 shows the leader line as it was visualized before the implementation of the CBDF-algorithm.

The race shown in figure 2.17 is the 2012 Kieler Woche 49er Gold Race 11. During this race the wind was not blowing in the leg direction. In the current leader line visualization and leader board calculations the boats on the port tack is far behind the boats on starboard tack.

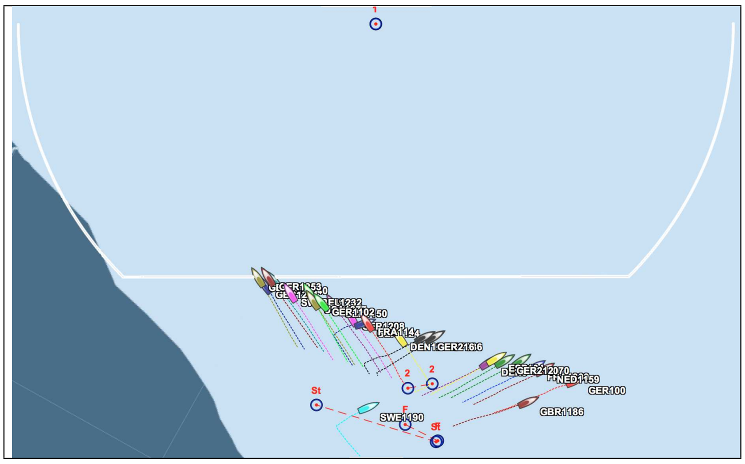


Figure 2.17: *A fleet during the 2012 Kieler Woche 49er Gold Race 11, two minutes into the race. The wind direction estimation algorithm is not used in the leader line visualisation*

With the knowledge of the wind direction the leader line can be centered and rotated around the wind vector. Comparing figures 2.17 and 2.18, the boats on port tack are now closer to leader line.

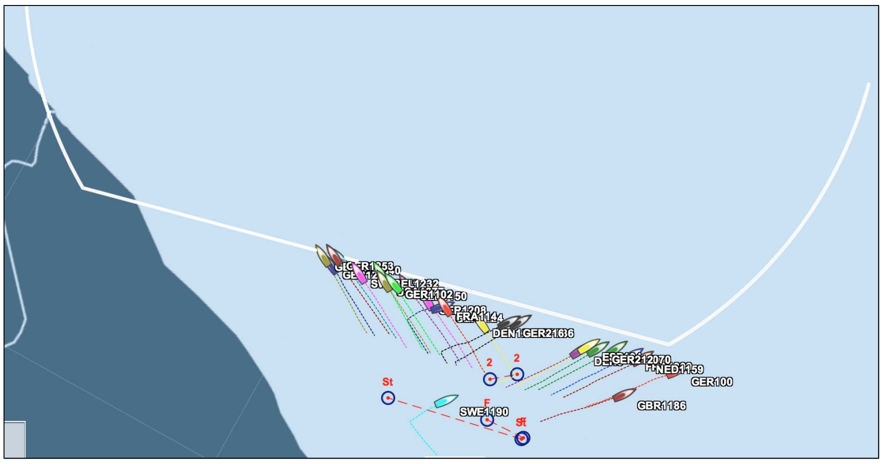


Figure 2.18: *A fleet during the 2012 Kieler Woche 49er Gold Race 11, two minutes into the race. The wind direction estimation algorithm is used in the leader line visualisation*

The calculations and drawing of the leader line are done in a similar fashion as the leader board calculations, but only for the leading boat. The leading boat's position is projected onto the wind vector and the center position of the leader line is the leading boat's projected position. Then, on both sides of the wind vector, a line with the length of the remaining euclidian distance to the next mark is drawn perpendicular to the wind vector. The sides of the leader line are drawn as two 45 degree circle arcs with center in the mark the boats are sailing towards. Figure 2.19 shows the projection onto the wind vector. If the leading boat is outside the lay lines the projection and remaining distance is calculated in the same way as in section 2.4.2 and equation 2.11.

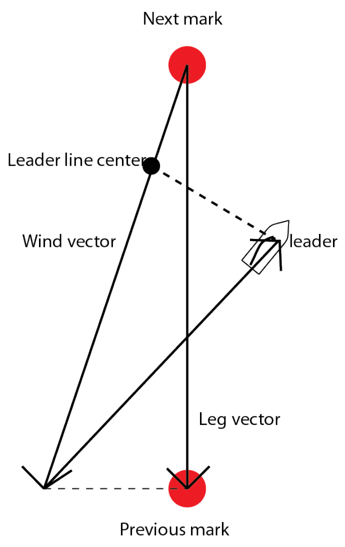


Figure 2.19: *Projection of the leading boat's position onto the wind vector*

The implementation running on TracTrac's servers are damped using previous time steps' wind vectors to make it more pleasing to look at, as jumps in the leader line are disturbing and visually dominant. The same damping is implemented for the leader board calculations to make the calculations similar. The damping is done by a weighted average of the last four samples.

Chapter 3

Discussion

In this chapter, the results from the implementation of the wind direction algorithm will be discussed. The chapter is divided into two parts: Section 3.1 is about the algorithm implementation and the results from the complexity calculations and section 3.2 discusses the applications implemented using the CBDF-algorithm.

3.1 Algorithm Implementation

The nature of the application's context and how the algorithm is used in the TracTrac race client requires that the CBDF-algorithm is a real-time algorithm. All calculations in the TracTrac client are performed 1-8 times per second depending on the playback speed and computer performance on the client side. During live GPS tracking the wind direction estimation is calculated at a maximum 8 times per second, giving a total of 125 ms per time step. As shown in table 2.1, the total running time of the worst case example that exists in TracTrac's tracking archive, Race C ($n = 140$), is $123500ns = 0.1235ms$. Even with a fleet several times larger than Race C the running time of the algorithm will not exceed the total running time of each time step. Race C is one of the biggest races TracTrac has tracked and races do not exceed 200 boats.

The algorithm analysis performed as a part of the evaluation of the algorithm, and highlighted in section 2.3.3, uses the algorithm analysis model described in [12, Chapter 2] and [13]. This is the model normally used for calculating an algorithm's growth rate and give a good indication of how the running time of the algorithm increases as the input size $n > 0$ grows. The time-complexity analysis performed in section 2.3.3 show a worst-case running time for the CBDF-algorithm of $\mathcal{O}(n^2)$. However, this quadratic case corresponds to a race with one boat on each leg. During realistic races with normal conditions the running time of the algorithm is $\mathcal{O}(n)$ as shown in the algorithm analysis in formula 2.8. Looking at the actual measured running time of the CBDF-algorithm, highlighted in section 2.3.4, the part of the algorithm, Leg Sorting and Weighting (algorithm 2.2) that has the worst analyzed running time takes up only around 5% of the actual running time. For the size of fleets that TracTrac tracks it is constant running time parts of the algorithm that takes up the most time of the actual measured running time.

As stated in section 1.3, the main goal of the wind direction estimation algorithm is not to calculate a precise wind direction. The resulting wind direction is an estimation of the wind based on the directions of skilful sailors. The computed wind direction can be used to give the audience a clearer picture of what is actually happening and understand the decisions taken by the sailors. The sounding board meeting conducted as a part of the project at TracTrac was used to validate that the resulting wind direction can be used for the specific purposes highlighted in section 2.4, and also discussed in 3.2. The three sounding board members are skilful sailors and two of the members are employed at a world leading sailmaking company. They have a great understanding of how the wind behaves in a fleet. After the presentation of the applications of the CBDF-algorithm given by the author, and the discussion that followed the presentation, the sounding board agreed that the CBDF-algorithm and the applications would give the audience a clearer picture of decisions made during sailing competitions.

3.2 Applications

The CBDF-algorithm has two major applications:

- To visualise the wind direction for the end user in the TracTrac race viewer client.
- To improve the virtual leader board calculations by using the wind calculations.

The wind rose is a simple way to illustrate for the audience in which direction the wind is blowing. The specific application done for this project is simple, but shows the wind direction to the audience which is the main goal of the wind rose. One problem exists for the wind rose implementation, when the boats are tacking downwind. Depending on the boat type, some boats tack on downwind legs. However but the CBDF-algorithm always assumes that the wind direction is blowing between the boats. The algorithm does not know anything about the direction of the course and the wind. Thus the CBDF-algorithm will always calculate an estimate. Based on the confidence calculations, the estimation might then be used. This means that if the boats' directions form two well formed cluster where the standard deviation is low enough for the confidence calculation to return a high enough value the wind rose shows a wind direction that is 180 degrees wrong. Restructuring the TracTrac client to take the leg direction into account is outside the scope of this paper, but section 4.1 suggests a solution to this problem.

As stated in section 1.3, the goal of the wind direction algorithm is to give the audience a more precise leader board at any time during the race. In an effort to validate improvements of the leader board, the ranking of each competitor for each time step is stored. Then every 80th sample (10 seconds) is used as a reference point and the ranking of that sample is subtracted from each of the preceding 79 samples. The absolute value of the preceding 79 samples is used to calculate an average of the change in ranking for each competitor. The average for each competitor is then combined to a single average describing the average change throughout the fleet, for every 80th sample. Figure 3.1 shows the change in ranking on the first leg of the 2012 Kieler 49er Gold Race 11. During this leg, the wind is stable but blowing from an angle of around 17 degrees compared to the race course:

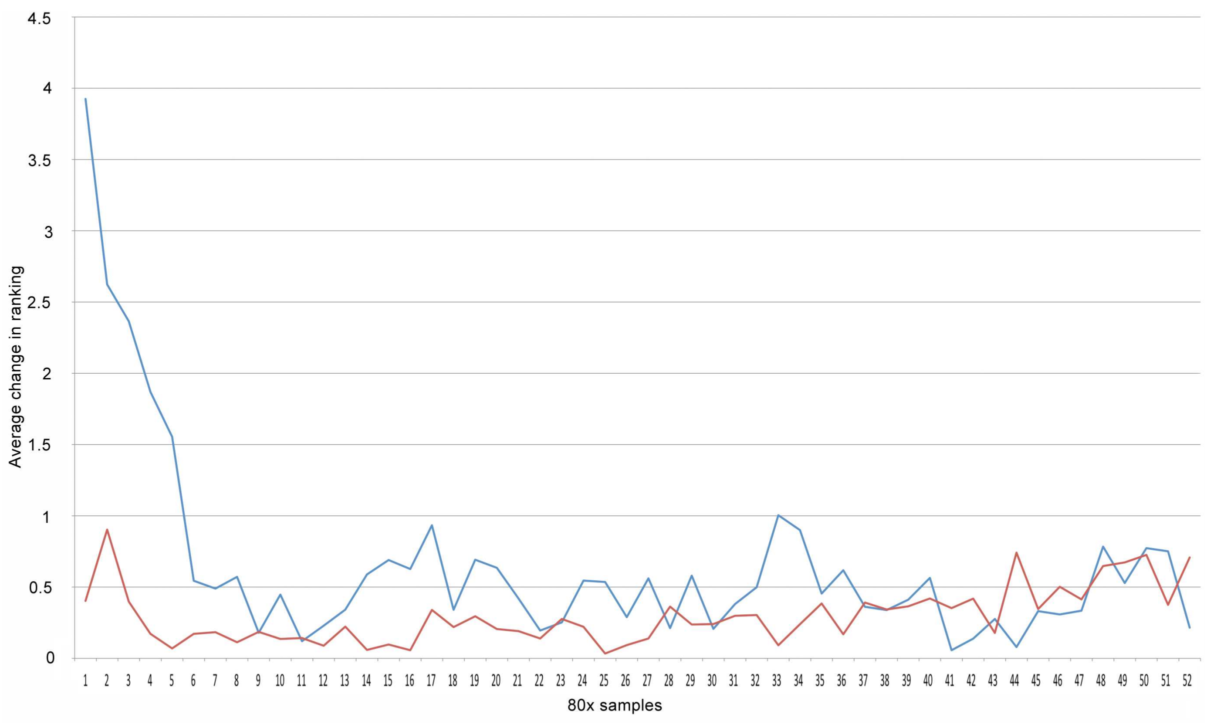


Figure 3.1: *Comparison of ranking change. Shown in the figure is the change in ranking before and after the wind direction algorithm is implemented*

Figure 3.1 does not confirm that the wind direction algorithm gives the audience a more precise and constant leader board. Although the no-wind line increases at the end, this is definitely not evidence that the wind direction estimation algorithm has given a more stable or precise leader board. When designing the validation method the author overlooked the fact that the boats would be a lot closer in the leader board. It is not necessarily a bad thing that the boats' ranking change a lot as the objective of the CBDF-algorithm is to give a more correct leaderboard. Figures 2.17 and 2.18 show that the boats in the lower right hand corner are closer to the leader line, i.e. has a better ranking in the leader board, when the wind estimation algorithm is used. This leads to a lot of small changes in position that leads to a higher average for the wind direction estimation algorithm results. To try to eliminate these small changes in position, for the purpose of validation, the playback speed in the TracTrac client is increased from real-time to five times real-time. Figure 3.2 shows the result of this increase in playback speed. The same procedure as above is used and every 80$^{\text{th}}$ sample is used as an offset and is subtracted from the preceding 79 samples. The absolute value is calculated and an average is stored.

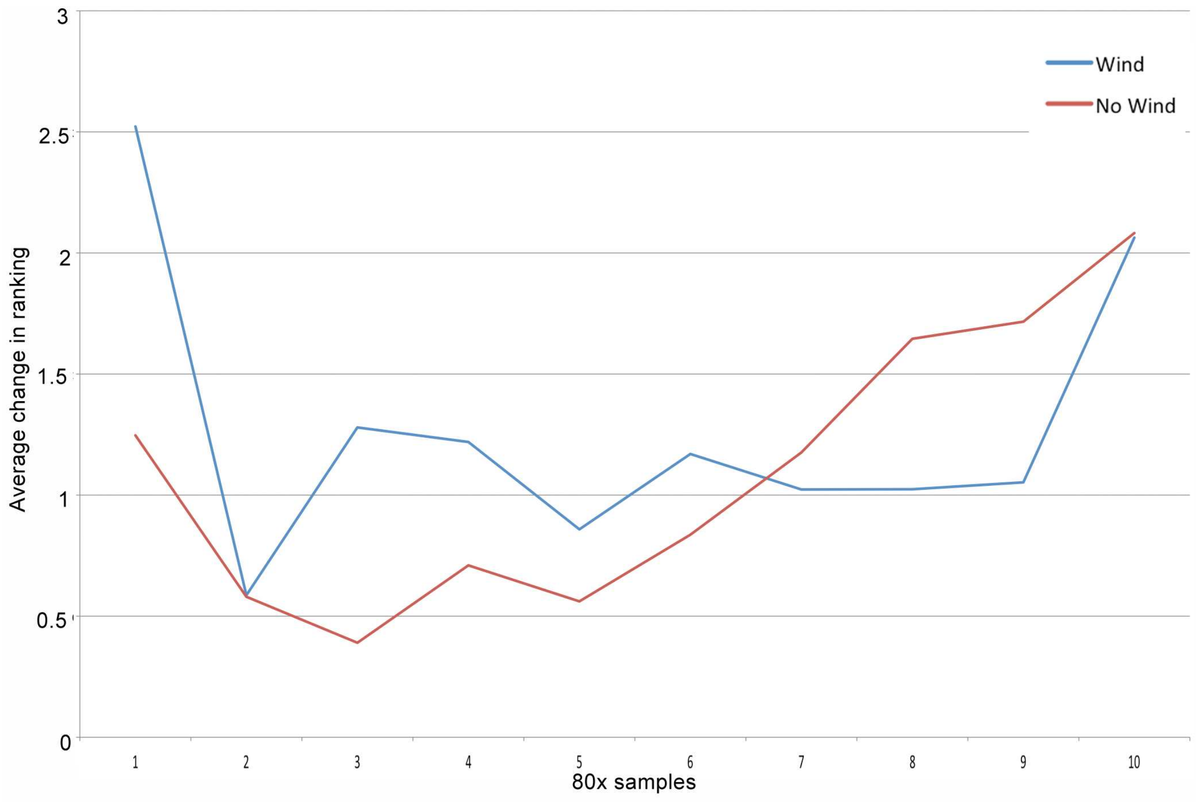


Figure 3.2: *Comparison of ranking change. The playback speed is changed to 5 times real-time*

The increase in playback speed gave a clearer view of the average change in position as the small changes in position are eliminated. The red line in figure 3.2 shows the average change in ranking for the calculation of the leader board without the wind direction estimation and the blue line shows the wind direction estimation calculated leader board. It is clear that the red line increases from around halfway through the race and that the blue line is more stable during this part of the race.

The last application of the CBDF-algorithm is the calculation and visualisation of the leader line. The leader line is the most important visual aid to show the current standings and wind direction. Because distances are huge during sailboat competitions, the fleet is often spread out on the entire course and far from each other. However the leader line can give an instant overview of the current standings. Prior to the implementation of the CBDF-algorithm, the leader line was always centered around the line between the two marks and the straight part of the leader line perpendicular to the leg direction. With the straight part of the leader line centered around the wind vector and perpendicular to the wind direction, the leader line gives a more precise and correct picture of the current standings. The validation of the more precise leader lines are similar to the leader board calculation as the two are calculated similarly.

Chapter 4

Conclusion

In this paper an algorithm is proposed that calculates a wind direction estimation from the directions data of an entire fleet of boats during a fleet race. The main goal of the algorithm and the derived wind direction is to highlight the current wind direction for the audience and to help the audience understand the wind specific decisions made by the sailors.

Firstly, several mathematical solutions are presented and evaluated in MatLab. A convolution based solution was chosen and a pyramid shaped filter was evaluated as the best filter to be used in the convolution. The CBDF-algorithm is based on the convolution based solution and implemented in the Java programming language on TracTracs live race servers. The application of the CBDF-algorithm dictates that the algorithm shall be a real-time algorithm. The timing of the algorithm in section 2.3.4 and the analysis in section 2.3.3 confirms that the algorithm is suitable to be used in real time applications. Normal fleets range from five to upwards of 150 boats, but the proposed algorithm handles fleet sizes several times larger, since the algorithm is a polynomial algorithm.

Secondly, the implemented applications highlight how the wind direction can be utilised as a way for the audience to understand some of the intricate decisions that the sailors make during a fleet race. The leader board calculations and the leader line visualisation are integral parts of the TracTrac broadcasts and section 3.2 confirms, despite problems with the validation method, that the leader board calculations give a more exact leader board at an earlier time during the race.

Thirdly, the sounding board meeting conducted during the project at TracTrac and described in section 2.2 confirmed that the applications of the CBDF-algorithm helps the spectator understand the wind specific decisions of the sailors.

Lastly, as mentioned in section 1.5 there are few GPS-tracking companies worldwide and none of TracTrac's competitors have solved the problem. The CBDF-algorithm and the applications presented in this paper is a cost-effective way to provide the spectators with extra information as all the input data of the CBDF-algorithm is already available from the GPS-tracking units. The goal of the project at TracTrac was to provide more information to the spectator from the data already available and not to calculate a scientifically correct wind direction. First of all the GPS position data that the directions are calculated from are accurate as long as it is compared to the data from the other tracking units in the fleet, but is inaccurate when compared to fixed points. TracTrac's provider of tracking units promises an accuracy of 10 meters, but different tracking units have the same inaccuracy. TracTrac have tested this extensively and the inaccuracy when compared to other boats is less than 1 meter.

4.1 Future Work

The first recommendation for future work is based on the break down of the CBDF-algorithm described in section 3.2, where the CBDF-algorithm estimates a wind direction that is 180 degrees wrong because some boat types tack on downwind legs. Creating a model of the entire course based on the leg directions and data provided in race information about upwind and downwind legs would make it possible to show a correct wind direction even for downwind legs.

Lay lines provide an extra dimension of information to the spectator. When TracTrac calculates the leader board it is assumed that the lay lines are lines rotated 45 degrees on either side of the wind vector. However the difference in degrees between the two tacks is equal to the difference between the lay. The difference in degrees between the lay lines is based on boat type and wind strength and basing the leader board calculation on the difference in degrees between the two tacks would make the leader board calculation even more accurate.

Acknowledgements

First of all I would like to thank TracTrac, and especially the three founders Jesper Grooss, Chris Terkelsen and Jakob Ødum, for their time and commitment during the development of the algorithm. I would also like to thank Erik Tingleff Larsen from Weibel Scientific and Henrik Søderlund and Mads Christensen from North Sails for contributing to a good sounding board discussion on the specifics of sailing and how the algorithm could be optimized based on knowledge of fleet sailing.

All figures are either produced by the author or used under the Creative Commons license.

Appendix A

Sailing Specific Word List

Bow: Front of the boat.

Stern: Back of the boat.

Starboard: Right side of the boat looking at the bow.

Port: Left side of the boat looking at the bow.

Rudder: Device used to steer the boat.

Tack: Turn the boat through the wind. The bow will pass from one side of the wind direction to the other.

Starboard tack: When the wind comes from the starboard side of the boat (right).

Port tack: When the wind comes from the port side of the boat (left).

Beating to windward: The process of moving against the wind.

Trimming: Adjusting the sail and rudder.

Lay lines: Imaginary lines from the goal of the sailboat extending out in the water. When a boat is on the outside of the lay lines it can sail directly to the goal and does not need to tack.

Downwind: Sailing with the wind coming from the stern of the boat.

Upwind: Sailing with the wind coming from the bow of the boat.

Buoy: Floating mark in the water.

Leg: From one buoy to the next buoy.

Fleet: The boats taking part in a race.

Close haul: The optimal direction for beating to windward.

For a more extensive description of the terms above, the reader is referred to [14].

Bibliography

[1] Flandro GA, McMahon HM, Roach RL. Basic Aerodynamics: Incompressible Flow [e-book]. Cambridge University Press; 2011. Available from: MyiLibrary `http://lib.myilibrary.com?ID=338383`.

[2] Smith B, Evans J, Manley P. The Sailing Bible: The Complete Guide for All Sailors from Novice to Experienced Skipper. A&C Black; 2009.

[3] Isler JJ, Isler P. Sailing For Dummies. John Wiley & Sons; 2006.

[4] ISAF. The Racing Rules of Sailing;. `http://www.sailing.org/tools/documents/RRS20092012with2010changes-[8222].pdf`, Date: 2012-08-30. Homepage.

[5] Kamen EW, Heck BS. Fundamentals of signals and systems using MATLAB. Prentice-Hall; 1997.

[6] Hartigan JA. Clustering Algorithms. Wiley Series in Probability and Mathematical Statistics. Books on Demand; 1975. Available from: `http://books.google.se/books?id=v3fRAAAACAAJ`.

[7] MacQueen JB. Some Methods for Classification and Analysis of Multivariate Observations. Defense Technical Information Center; 1966. Available from: `http://books.google.se/books?id=lpfftgAACAAJ`.

[8] Seber GAF. Multivariate Observations. Wiley Series in Probability and Statistics. Wiley; 2004. Available from: `http://books.google.se/books?id=4PCa-OIL34QC`.

[9] Blom G. Sannolikhetsteori och statistikteori med tillämpningar. Studentlitteratur; 2005. Available from: `http://books.google.se/books?id=fbWXMwAACAAJ`.

[10] Pfleeger SL, Atlee JM. Software Engineering: Theory and Practice. Prentice Hall; 2010. Available from: `http://books.google.se/books?id=7zbSZ54JG1wC`.

[11] Snyder JP. Flattening the Earth: Two Thousand Years of Map Projections. University of Chicago Press; 1997. Available from: `http://books.google.dk/books?id=0UzjTJ4w9yEC`.

[12] Weiss MA. Data Structures and Algorithm Analysis in Java:International Edition. Pearson Education; 2006.

[13] Knuth DE. The Art of Computer Programming. vol. 1, Fundamental Algorithms. Reading, Massachusetts: Addison-Wesley; 1997.

[14] Smyth WH, Belcher E. The Sailor's Word-book: An Alphabetical Digest of Nautical Terms, Including Some More Especially Military and Scientific as Well as Archaisms of Early Voyagers, Etc. Blackie and Son; 1867.