Obstacle avoidance for an autonomous marine robot

Bachelor Thesis

for obtaining the academic degree of

Bachelor of Engineering (B. Eng.)

Silke Schmitt

Supervisor:

Prof. Dr. Fabrice Le Bars
Prof. Dr.-Ing Thomas Latzel

23.09.2013

Universität der Bundeswehr München
Fakultät für Elektrotechnik und Technische Informatik
Wissenschaftliche Einrichtung 4

Neubiberg, September 2013

Academic Honesty Declaration

I hereby declare that I wrote the present bachelor thesis independently and on my own, by exclusive reliance on the tools and literature indicated therein. All text passages that are taken from other papers literally or analogously are indicated with a precise reference.

The thesis has not been submitted to any other examination board.

Neubiberg, 23. September 2013

Silke Schmitt

Abstract

...

Contents

List of Figures

1 Introduction

1.1 Motivation

The championship WRSC 2013 (World Robot Sailing Championship)[1] takes place at ENSTA Bretagne in Brest (France) from 2nd to 6th September 2013. This bachelor thesis arises out of this context and deals with one of the challenges of the WRSC 2013.
The first WRSC took place in the year 2008 in Austria and ever since this event takes place every year in another country. This championship brings together various teams of students from all over the world to compete their autonomous marine systems in contests analysing the ability of their boats in tasks testing endurance, reaction and autonomy. Participating sailboat robots need to be entirely autonomous and may have a length up to 4 m. This year 2013 there are also tasks for the category of motorbaots for the first time. Together with this competition there is always held the International Robotic Sailing Conference (IRSC) to discuss current topics in design and development of autonomous sailing boats.
Autonomous marine vehicles offer varied possibilities of application, for example they can help in observing the marine environment (messurement of temperature, level of salinity etc.) or they can be applied in the fields of defense engineering (e.g. surveillance tasks) as well. Thus there are multiple championships which are engaged in autonomous vehicles.[1]
Compaired to other autonomous vehicles sailing robots do have a clear advantage of low energy consumption which is more and more a very significant point. Though, a sailing robot has to work in a very dynamic environment where it needs quickly to respond to changing conditions. That indicates the need of a well-working control algorithm for steering the sailboat robot.
This bachelor thesis is engaged in the WRSC task for sailboats of avoiding a mobile obstacle with a known position. Therefore the sailboat robot first has to stay in a square with a side of 200 m (defined by GPS coordinates). Within a time of 45 minutes a disruptive boat will enter the square and cross it with a constant speed and heading. Its position, heading and speed will be send by Wifi and XBee. The sailboat robot has to avoid this boat and leave the square, thereby the boat may not come closer than 5 m, the sailboat robot may not leave the square before the other boat enters it and the square needs to be left until 15 minutes after the entrance of the disruptive boat.
This thesis shall realize the control algorithm for obstacle avoidance as it is a fundamental question that needs to be solved.

[1]For example SAUC-E, Student Autonomous Underwater Vehicle Challenge - Europe which since 2010 has been held at the Centre for Maritime Research & Experimentation from the NATO in Italy.

1.2 Thesis Subject

For the participation at this championship the team at ENSTA Bretagne already has developed the hardware of the sailboat robot [1, student report] as well as a big part of the software. Based on this, my work is to realize an algorithm for obstacle avoidance for the sailboat robot. This means the implementation of a control algorithm that takes into account the aiming point to approach to as well as the positions of obstacles that need to be avoided on the path.
This is done in a simulation project that is programmed in C++. The simulation program is developed using the cross-platform complete integrated development environment (IDE) Qt Creator. For graphical representation in the simulation the standard OpenGL is used. The state equations for the sailing boat are given by the paper [2] and are already implemented in the basic project that is provided. Working on its input parameters I can develop the control algorithm to steer to a certain direction considering the target point and points to avoid as well as the conditions of the environment (wind).
Afterwards the working software will be implememted and tested one the real sailboat robot.

1.3 Structure of Thesis

This thesis discribes the developement of an algorithm for an obstacle avoidance on a marine robot platform. Chapter 2 presents principles which serve to provide all fundamental information needed to comprehend this thesis. Chapter 3 describes the software which is used in this thesis. It also contains the explanation and description for the approach that I have choosen and that is followed to solve the problem of an obstacle avoidance control. After this, chapter 4 contains the results of this work. This is the developed C++ project to simulate the sailboat robot with the control algorithm. Chapter 5 evaluates the results based on simulation outputs and live tests. Furthermore, there is given a conclusion.

2 Principles

This chapter explains the principles which are required to understand the subject of this thesis and the implemented algorithms. The first section deals with the theory of sailing. Different wind situations and thus multiple courses of a sailing boat are explained, so that there is a understanding of the possibilities of a sailing boat. The profound physics of a sailing boat are simplified to the part that is essential for this work.

The second section covers the system model of the sailboat robot based on state equations. These differential equations are used to control the sailing robot and therefore this is an important part for a control algorithm.

2.1 Theory of Sailing

2.1.1 The Sailing Boat

A sailing boat consists of a hull (body of the boat), a rudder and a mast with the sail. While the hull is designed to keep the boat at the surface and to stabilize the boat, the rudder and the sail are actors with which the course of the sailing boat is controlled. The mast and the sail are both held in a certain bending in order to develop an aerodynamic force and give them stability.
ENSTA Bretagne works with the autonomous sailing boat VAIMOS (Voilier Autonome Instrumenté pour Mesures Océanographiques de Surface, engl. Autonomous sailing boat with embedded instrumentation for ocean surface measurements). This sailing boat VAIMOS has a length of 3.65m and a weight of 300kg [3]. The sailboat VAIMOS has one mast with a fore sail and a main sail; figure 2.1[1] shows an image of the sailing boat VAIMOS.

2.1.2 Apparent Wind

The sailing boat experiences not the true wind but the *apparent wind*. The true wind is the wind speed relative to the water. From the boat's point of view, the wind seems to be coming from a different direction than the true wind. This is due to the movement of the boat relative to the water. The relationship between the three vectors true wind $\vec{W}$, apparent wind $\vec{V}$ and the boat's velocity $\vec{U}$ is illustrated in figure 2.2.

In this "velocity triangle"the true wind is coming from above with the magnitude W. The angle of the true wind w is defined, as it is conventional for sailors, between the direction the true wind is comming from and the direction where the sailing boat is moving to. In the same way the angle of the apparent wind v is defined. The "apparent wind

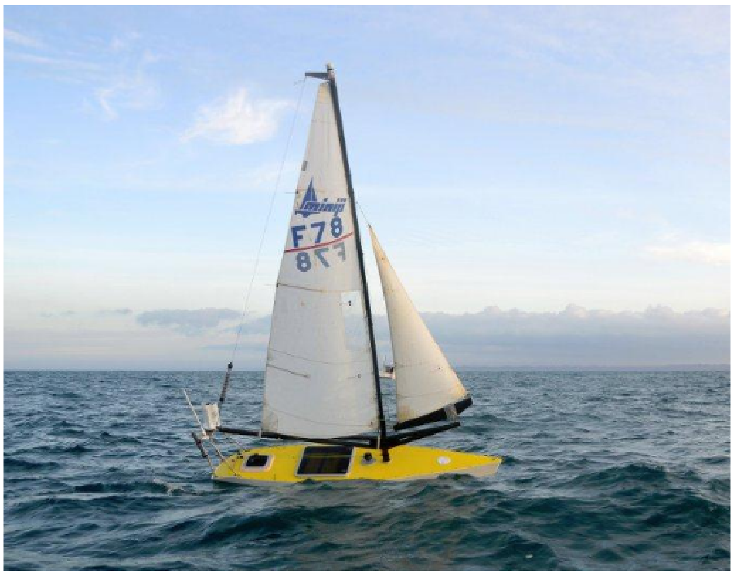


Figure 2.1: Autonomous sailing boat VAIMOS

shift"$(w - v)$ is the angle that the apparent wind differs from the true wind because of the motion of the boat.

This apparent wind is the one that matters on the sailboat. When hereafter the wind is mentioned, it is referred to the apparent wind. This is the wind that has to be used to calculate the forces on the sailingboat.[4, p.40-42]

2.1.3 Forces on a Sailing Boat

The two important forces on a sailing boat are the power of resistance (drag) F_D and the lift F_L. These forces occur on the hull through the flowing water and they occur at the sail as well, caused by the air flow.

The sail of the sailing boat is used for steering either by taking advantage of the drag or of the aerodynamic power that causes the lift. The drag force F_D is simply due to the fact that the wind cannot go through the sail, so the sail poses a resistance to the air. The drag F_D is calculated with the following equation. In this equation c_d is the drag coefficient, ρ is the fluid density, A is the effective area and V is the fluid speed, in this case it is the apparent wind velocity V.

$$F_D = \frac{c_d}{2} \cdot \rho \cdot A \cdot V^2$$

To go upwind the lifting force F_L of the sail matters. A sail works by deflecting air just similar to a wing of an airplane. The air is deflected on the windward side since the air cannot go straight through the sail. On the leeward side of the sail the air is deflected to

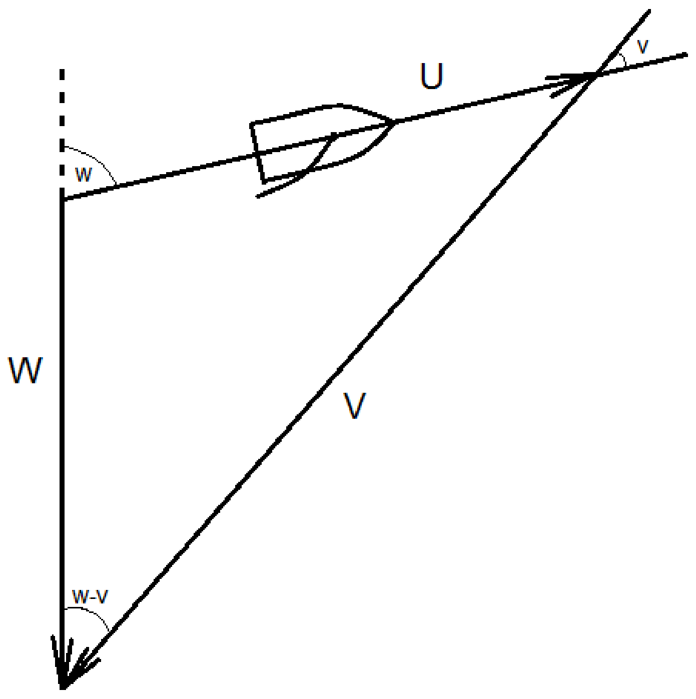


Figure 2.2: A velocity triangle to illustrate the apparent wind.

the sail because of the Coanda effect[1]. On the leeward side arises a low-pressure whereas on the windward side there is high pressure. This difference in pressure causes a force perpendicular to the (apparent) wind. The equation to calculate the lifting force F_L is similar to the resistance force, instead of a drag coefficient there is a lift coefficient c_a which specifies the dynamic lift character of the object (here: the sail).

$$F_L = \frac{c_a}{2} \cdot \rho \cdot A \cdot V^2$$

The hull, with its keel and rudder, poses a resistance against the flowing water. By setting the rudder to a certain angle, this resistance force can be varied in order to steer the boat. The resistance force F_D is always opposite to the boat's speed U. It is calculated with the equation that has just been presented above with reference to the sail. At the hull the fluid is not the air but the water and hence the fluid velocity V is the boat's speed U. Similar to the sail, there is also a lifting force caused by the water. A small sidewards movement of the boat (the result of the wind force) generates a low-pressure on the one side of the boat, and a high pressure on the other side of the keel. As a result there is a lift F_L (hydrodynamic force) perpendicular to the boat's speed (see equation above).
The forces are added to determine the resulting force which is the boat's course. In figure ... an upwind course of a sailing boat is presented. The wind force and water force are in

[1]This is the phenomena of a jet flow tending to be attracted to a nearby surface, even when this surface bends it to another than its initial direction[5].

balance, when the sailing boat has a constant speed U.[4, p. 42]

2.1.4 Sailing Courses

In general a sailing boat is able to drive up to an angle of 45° towards the wind (this can vary a bit depending on the boat type). Hence there is a no-go zone of ±45 to the wind, that is not feasible to move directly to.
To sail with the minimum angle of 45° to the wind is called *sailing on a close reach*. The maximum speed is achieved at a course that is called *beam reach*; this is when the boat is sailing with a right angle to the wind (wind coming from abeam). Is the boat sailing with a greater angle than 90° to the wind, this is reffered to as a *broad reach*.
The direction of the no-go zone can be reached by beating, which means to sail close to the wind with a series of tacks. A tack is the turn of the boat towards the other side, with the boats bow going through the no-go zone. Figure 2.3 shows all terms to define the course of the sailing boat in relation to the wind direction.

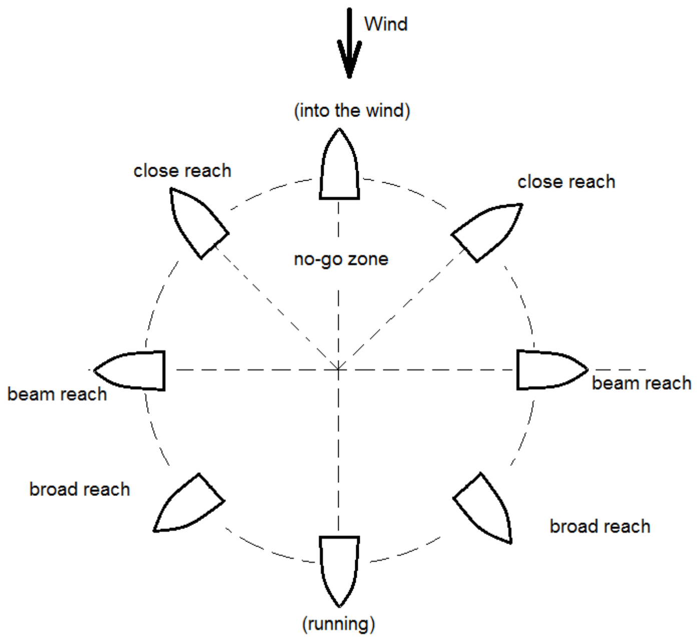


Figure 2.3: Sailing cources with respect to the wind

2.2 System of a Sailing Robot

In order to control the sailing boat, it is necessary to describe the state of the boat with concrete input and output parameters. Therefore the use of state equations for the system is essential. Sailboats are "nonlinear hybrid systems involving strong pertubations such as waves"[2]. There is no exact state equation of a sailing boat existent but as sailing boats

have been steered by humans for a very long time it has been choosen an approach to imitate the strategy of sailors.
The state vector of the sailing boat has the dimension 7 and consists of the following parameters:

- Position coordinates x, y of the center of gravity G of the boat
- orientation θ
- angles δ_s and δ_r of the sail and the rudder
- kinematic coordinates v and ω for the velocity of the center of gravity G and the angular velocity around G

This state vector contains all parameters necessary to capture the state of the boat. Input parameters of the system are the derivatives of the angles δ_s and δ_r. With these input parameters and constant values that are either generally known or can be mesured easily (speed of the wind, dementions of the boat, lift of the rudder and lift of the sail, friction coefficient of the boat on the water, moment of inertia of the boat and drift coefficient) the new state of the boat can be calculated.

In order to calculate the state of the boat physical laws are applied, the calculation proceeds from the force of the wind on the sail and the force of the water on the rudder. To see the concrete (differential) equations and the derivation of these equations I refer to the paper [6, p. 32-33]; here in this section the aim is to clarify that there are state equations on which the regulation is based on. The two input parameters have to be calculated and defined by the control algorithm that will be implemented. [6]

3 Methods

3.1 Initial C++ Project with Basic Control Algorithm

In the provided paper [2] a simple algorithm is presented which controls the steering of the sailing boat in order to move along a defined line from A to B. The provided C++ project for simulation does contain one sailing boat. Managed with a timer event, the state of the sailing boat described by the state equations mentioned in chapter 2.2 is updated continuously. The sailboat is simulated with the line following controller, this is the part generating the input parameters (deratives of the angle of the sail and the rudder) to the sailng boats state. The widget shows the sailing boat moving on the water. The user can turn the camera view with the mouse input and by key input the zoom can be changed. This project displays the initial situation; for the developement of my control algorithm this project has been modified and extended.

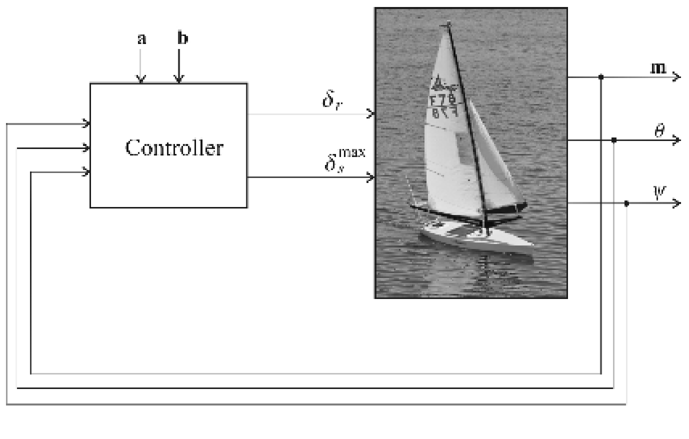


Figure 3.1: Controller of a sailboat robot

Figure 3.1 [2, p.2] illustrates the regulation loop with this line following controller. This is one simple example of a control method using the potential field strategy (vector field). It assumes the boat to have sensors to messure the heading θ (by a compass), the angle of the wind ψ (by a weather vane) and the current boat position m (given by a GPS). As actors operate the angle of the rudder δ_r and the maximum angle of the sail δ_s^{max}. The latter one is the maximum angle of the sail, because the angle of the sail itself cannot be set directly. Yet with the length of the rope the angle of the sail can be limited to δ_s^{max}.

The angle δ_s apllies to:

$$\mid \delta_s \mid \leq \delta_s^{max} \tag{3.1}$$

An important point in this control method is that it has one state varable $q \in \{-1, 1\}$ which implements a hysteresis. By this hysteresis the boat has allways one favorate side to turn to. This is important for an up-wind course, so that the sailing boat will not turn permanently but drive towards one side as long as it is within a certain zone around the desired line. It will not turn before it reachs the side and then do the same towards the other side. This allows the sailing boat to get further on its way but not to be mostly drawn back by the wind due to permanent change of direction.

3.2 Software

3.2.1 Qt Creator

Qt Creator is the name of the integrated development environment (IDE) which is been used to develop applications with the Qt application framework from the company Digia[1]. As a cross-platform application framework Qt enables the user to put the designed application on different systems without code adaption. The Qt Creator serves to design C++ projects, supporting projects with graphical user interfaces (GUIs) with the provided Qt Widgets module. In this work the Qt Creator version 2.0.1 (based on Qt application framework 4.7.0) has been used.

3.2.2 OpenGL

OpenGL (Open Graphics Library) is an application programming interface (API) which is used to render 2D and 3D computer graphics. It is a developement of Silicon Graphics Inc. (released in 1992) and is now managed by the non-profit technology consortium Khronos Group[2]. The OpenGL specification is language-independent as well as plattform-independent and is the most widely used graphics API in industry. It is a low-level API meaning the user has to specify geometry primitives and thereby it leaves a lot of possibilities to the user.

3.3 Chosen Approach of Controlling

An obstacle avoidance strategy should result in a steering of the robot away from disruptive points as collisions have to be prevented. At the same time the robot shall still be approaching its destination. So this is an important task in robot navigation.
There are several obstacle avoidance strategies but as a sailboat robot is a non-linear system operating in a dynamic environment there has to be implemented an efficient and

[1]Reference: http://qt.digia.com/
[2]Reference: http://www.khronos.org/opengl/

simple approach. This is why the potential field method is chosen. The path of the sailing boat robot shall be defined by a vector field. This vector field is constructed upon the goal position (attractive potential) and points which need to be avoided (repulsive potential). The robot is considered as a point which is moving in this potential field; its heading is always the gradient of the field. With this method there is a defined heading of the boat for every possible boat position.
Combining simple vector fields makes it possible to describe a more complex field. The vector field will for this reason be build as a bineary tree. Vectors are either nominal vectors, which point to the destination, or they are caused by a repulsive vector field so that they lead away from a point.

4 Results

4.1 Simulationproject Sailboat

4.1.1 Design Enhancement

In a first step the provided project “sailboat”, which serves for simulating the sailboat robot, is extended by an improved visualization. For visualizing the sailboat with its environment the standard OpenGL (chapter 3.2.2) is used. This visualization is added by the possibility to move the camera, so that the sailboat robot can be followed and thereby being examined in an easier way than just by turning the camera’s view. Therefor the OpenGL function `gluLookAt()` is used with three parameters defining the 3D-position of the camera. These parameters can be changed by keyboard input.
In the initial version of the class `sailboat` the provided control algorithm of [2] was just partly implemented. In a next step this control algorithm from the paper [2] is completely implemented in the function `Controller()` of the class `sailboat`.
To be able to analyze an obstacle avoidance strategy, the program is extended to handle two sailboats instead of just one. By this each sailboat automatically has one obstacle that has to be avoided.

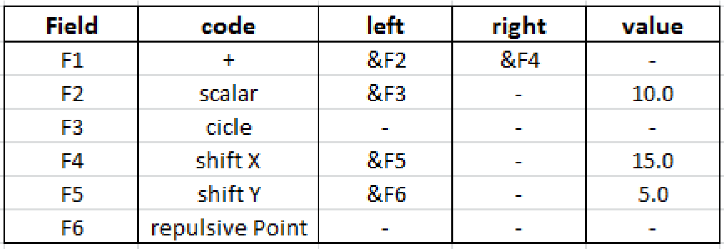
4.1.2 Implementation of Vector Fields

The approach for a control algorithm based on vector fields described in chapter 3.3 is first developed in a separate project `field`. This shall achieve the following points:

- Construction of vector fields based on atomic functions and unical operations
- Graphical illustration of the vector field

After the successful development this control method is integrated into the sailboat project by including the classes explained in the following and binding them to the user interface (UI) as well as making the connection to simulated sailboats.
A vector field is constructed upon the structure of a binary tree. The class `Vectorfield` serves for constructing a vector field; the class `vectorfieldSimu` implements the visualization of the vector field in a 2D figure by means of using OpenGL. The following member variables are the essential variables contained in the class `Vectorfield`:

- `QString* code`



Field	code	left	right	value
F1	+	&F2	&F4	-
F2	scalar	&F3	-	10.0
F3	cicle	-	-	-
F4	shift X	&F5	-	15.0
F5	shift Y	&F6	-	5.0
F6	repulsive Point	-	-	-

Figure 4.1: Example of a vector field F1 displayed in a tabular form.

- `vectorfield* left`
- `vectorfield* right`
- `double value`

The assigned QString to the variable `code` has to be one of a defined number of keywords and either stands for an atomic vector field or for a specific connection of two vector fields (`left` and `right`) respectively a specific connection of one vector field (`left`) with a constant number `value`.
Herewith a complex vector field can be constructed as it is shown in figure 4.1. In this example in figure 4.1 the root node F1 adds the two vector fields F2 and F4. Vector field F2 is an atomic vector field `circle` which is enlarged by the factor ten. Vector field F4 is an atomic vector field `repulsivePoint` which is shifted towards the point $(x, y) = (15.0, 5.0)$.

An equation for a circle regulation [6, p. 134], a line regulation [2, p. 3] as well as (self developed) equations for a vector field to an attractive point, a repulsive vector field guiding away from a line and a repulsive vector field away from a point are implemented as atomic (basical) functions. They are now explained and their graphical plot can be found in the appendix 6.1.
The vector field for a circle regulation leads into a circle with a radius r. The paper [6, p. 134] provides the function for this circle following, which can be seen below. For a point (x,y) this function calculates the tangential force, the vector defined by dx and dy. The parameters v_0, v_r and ρ describe the magnitude of the vectors and how fast the final state, the desired circle line, is approached. For this implementation they are determined to fix values, whereas the radius of the circle is set to a default value of ten but can be differed by the operation `scalar()` (see later in this part).

$$f(x,y) = \begin{pmatrix} dx \\ dy \end{pmatrix} = \frac{1}{\sqrt{x^2+y^2}} \cdot \begin{pmatrix} v_r x \arctan(\rho(r_0 - \sqrt{x^2+y^2})) + v_0 y \\ v_r y \arctan(\rho(r_0 - \sqrt{x^2+y^2})) - v_0 x \end{pmatrix}$$

$$with \quad v_0 = 10, \quad v_r = 10 \quad and \quad \rho = 0.1$$

The line regulation guides the boat along a line from point A to point B. For the implementation of this kind of vector field these two points are chosen to be fix. Without loss of generality they are defined as $A = (-100, 0)$ and $B = (100, 0)$; the line position and orientation may be changed by methods (rotation and shifts) which can be applied upon this atomic vector field. The vector's orientation is parallel to the line, when the boat's position is on the line. Otherwise the angle is inclined towards the line; the greater the distance towards the line[1] the higher the inclination, yet there is a maximum inclination of 45°, so that the boat still keeps going ahead in the direction of the line.

A vector field towards one point (attractive point) is useful to have simply one goal position. The function $f(x)$ to calculate the vectors magnitude has to fulfil the requirements of having a constant limit value for $x \to \infty$ and going through the point $f(0) = 0$. Like this the vector's magnitude is nearly constant, just at the attractive point there is a null vector and nearby the magnitude gets less. The following function fulfils all requirements and is used to calculate the vector's magnitude in the vector field for an attractive point.

$$f(x) = e^{-\frac{1}{x}}$$

For the two repulsive vector fields the vector length is defined by the Gaussian function. The Gaussian function has a maximum $f_{max} = 1$ at the center $(x, y) = (0, 0)$ respectively on the central line $x = 0$ and decreases to zero at infinity.
All nominal vector fields are normalized to return a vector with a magnitude equal ten and all repulsive vector fields are normalized to a maximum vector length equal twenty. So the maximum value of a repulsive vector field is not one but twenty, which does not change the fact, that the magnitude of the vectors is descending to zero at infinity (and sufficient small values are reached at a certain distance). This vector normalization achieves that vectors have an equal impact when added, with the only exception of repulsive vectors near their central point. When the boat gets near the central point of the repulsive vector field, the main task is to depart from there (collision avoidance), so it is reasonable to increase the impact of the repulsive vector fields near there maximum.
As it is an important part of the implemented class `vectorfield`, listing 4.1 shows the method to calculate the repulsive vector field away from the point (0,0). The Gaussian function is streched by the factor of 100 to have an area of influence big enough for sailboats to react.

[1] The distance between a point and the line $\vec{AB}$ is calculated by the determinat, which is the orientated area of the parallelogram of two vectors.

```
  void vectorfield::repulsivePoint(double x, double y, double &dx,
     double& dy)
{
    double temp, angleObs;
    temp = hypot(x, y);
    angleObs = atan2(y, x);

    double force = exp(((-1)*pow((x), 2) - pow((y), 2))/10000);
    //division by 100^2 increases the influence radius by the factor
       100
    dx = dx + 20*(force * cos(angleObs));
    dy = dy + 20*(force * sin(angleObs));
}
```

Listing 4.1: Calculation of a repulsive vector field

The vector $\vec{v} = \begin{pmatrix} dx \\ dy \end{pmatrix}$ of this vector field is returned by the usage of reference parameters; its magnitude is calculated with the Gaussian function taking the distance to the centre as the function's input. The vector always shows away from the central point (0,0). On the basis of all atomic vector fields, which have just been presented, more complex vector fields can be build by applying the following listed methods:

- `operator+`
- `rotate()`
- `shiftX()`
- `shiftY()`
- `scalar()`
- `projection()`

With the operator overlaoding method for the operator $+$, this fundamental operation is also specified for the class-type `Vectorfield`. As the name of the method `rotate()` implies, this rotates the vector field by an angle specified with the commited parameter. The method `shiftX()` shifts the vector field in the direction of the x-axis, the method `shiftY()` shifts it in sense of the y-axis. To increase a vector field by a commited factor the method `scalar()` is used.
For a sailboat robot it has to be considered that the direction in a certain angle is impossible to go directly to. This is the direction the wind is coming from (see the previous chapter 2.1.4 about sailing courses). So vectors of a vector field which are in a certain range of angles have to be projected to the next possible angle. This is done by the method `projection()`. The commited parameter `value` is the angle of the wind and all angles

within the following range are mapped to the next possible angle value.

$$[value - \frac{\pi}{4} \cdots value + \frac{\pi}{4}]$$

All these methods that have just been explained do return the new composed vector field as the function return value.

One essential method of the class `vectorfield` is the one to get the vector to a certain point. Therefor the method `eval(double &x, double &y, bool changeAllowed)` is provided. The reference parameters x and y commit the point (x,y) and this function writes the point, where the vector points to, into these variables. The parameter *changeAllowed* is set to *true* when the boat's position is commited and evaluated, for the points to draw the vector field this parameter is set to *false*. This parameter prevends a change of the variable `q` by the vector field illustration; only the position of the boat shall have an impact on the hysteresis variable `q`. In the method `projection()` the hysteresis variable `q`, which is set duing the evaluation of a point, is evaluated. The boat may be within a zone with a distance of up to fifty around the desired trajectory. With the member variable $q \in \{-1, 1\}$ of a vector field the preferred side to turn to is defined. When a projection is necessary, this variable `q` determines the angle, which is chosen. When the boat reachs one side of the zone around the trajectory, the variable `q` changes and the boat will move towards the other side of the zone.
In the vector field for an attractive point the zone is defined to be around the line of the wind direction, crossing the point in the center.
Figures 4.2 and 4.3 give two examples of a composed vector field. They both are composed of two atomic vector fields, one for the desired trajectory and one to avoid the second boat. The last applied method is `projection()` to make sure the sailing boat is just steered towards sailable directions. These two vector fields have been applied on the sailboat robots in the simulation and were tested.

4.1.3 Vector Field Visualization

The vector fields need to be illustrated not just to validate them, but as well to help a better understanding. A simulation run shows one constellation (of the boats' positions) and its behaviour. But there is an infinite number of constellations which can never be all simulated. The illustrated vector field shows a vector for many points and vectors of points inbetween can be estimated, so thereby the boats behaviour in other situation can be assumed.
In a second widget[2], next to the widget which shows the sailboats with their environment, the vector field is illustrated in 2D using OpenGL. With a button the user can switch between the vector fields of the different sailboat robots. A larger arrow displays the direction of the wind, which can be changed by the user with a slidebar. This enables the user to have an influence on the wind condition. As a reference in the 2D vector field

[2]A widget is a part of the GUI that displays information and permits an interaction with the user.

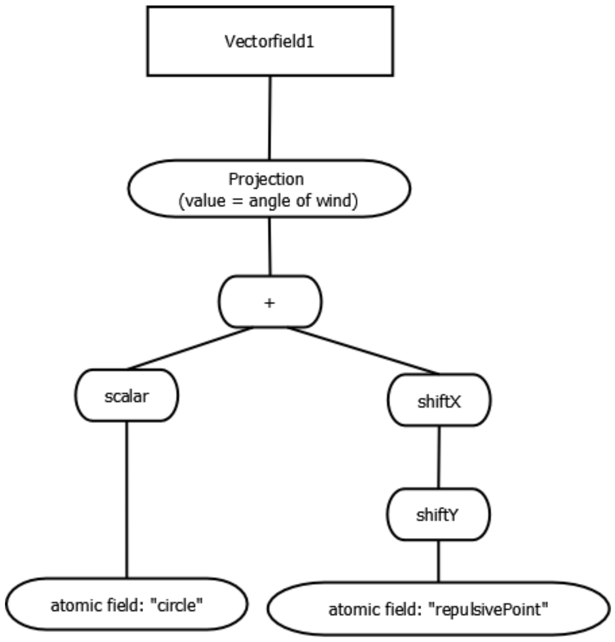


Figure 4.2: The vector field Vectorfield1 constructed as a binary tree.

illustration a green point displays the position (0,0). Boats are seen in the vector field as directed triangles; the boat to which the currently shown vector field belongs to is marked in red. The other boat is a blue triangle.
Figure 4.4 shows two screenshots of the simulation where the visualization can be seen. In the vector field for the line following this line is plotted (in black) together with grey lines showing the zone with a maximum distance of fifty (hysteresis zone).

When the check box “Fan of possible directions”is activated, at each vector there is also a fan. Within the hysteresis zone around the trajectory, this fan reaches from the vector’s direction minus $\frac{\pi}{4}$ to the vector’s direction plus $\frac{\pi}{4}$. Outside this zone, the fan is just onesided as there the variable q can just have one value.

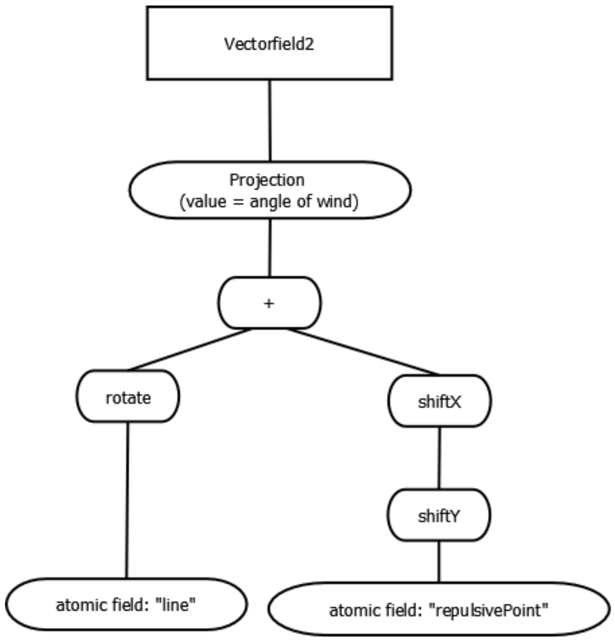
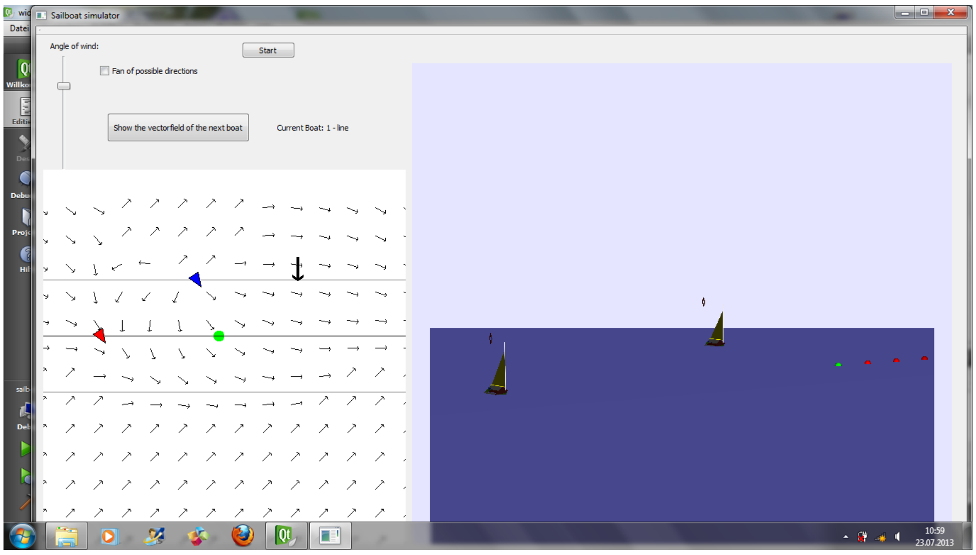
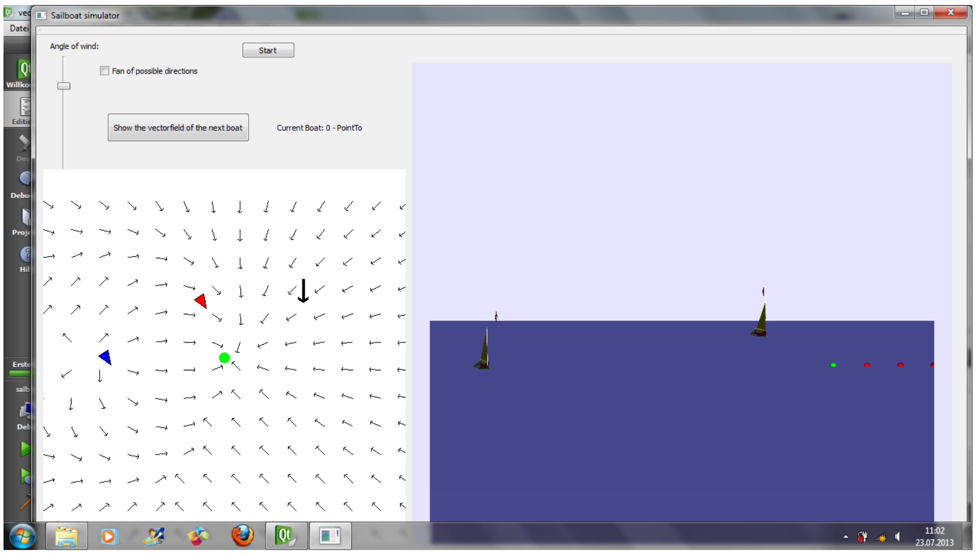

Vectorfield2
Projection (value = angle of wind)
+
rotate
shiftX
shiftY
atomic field: "line"
atomic field: "repulsivePoint"

Figure 4.3: The vector field Vectorfield2 constructed as a binary tree.



(a) Boat 1: Line Follwoing


Sailboat simulator
Angle of wind:
Start
Fan of possible directions
Show the vectorfield of the next boat
Current Boat: 0 - PointTo
11:02
23.07.2013

(b) Boat 2: Attractive Point

Figure 4.4: Sailboat Simulation

4.2 Dynamic Construction of the Boats Vector Field

The ambition is to identify obstacles in the robots environment by sensors (AIS, radar etc.) and to consider these in the vector field for path control. Hereby the number of relevant obstacles can be reduced to obstacles within a certain distance and in a certain angle towards the sailing boat. Obstacles behind the sailing robot which is sailing forwards do not bother the robot and so they shall not influence the vector field.
Other boats are considered as repulsive points, so they are always inserted into the vector field by adding an atomic vector field `repulsivePoint` which is shifted towards the position (x,y) of the detected boat.
...

5 Simulation and Testing

5.1 Simulation

Real live tests for a sailboat robot can be challenging, as the ocean is a dynamic environment. To minimize the probability of failing, but also for economical and time reasons, an important part is simulation.

The three trajectories defined by the implemented atomic vector fields `line`, `circle` and `PointTo`, should be followed as precisely as possible. How well this is done may just be judged, when there is no other disturbing object. Otherwise collision avoidance has top priority and it cannot be made sure that the boat takes the best way.
...

5.2 Live Test

As the sailboat robot VAIMOS is at the time of the test not available, the control algorithm is adapted to a motorboat control. This can simply be done, as the control algorithm using vector fields is a general approach that can be used for the other robots, like the submarine robot and the motorboat robot, as well. The motorboat has two motors, one on the left and one on the right. So coming from the sailboat control, the angle of the rudder δ_r is translated to the two control parameters $u1$ and $u2$, whereas the maximum angle of the sail δ_s^{max} can be ignored. The two control parameters $u1$ and $u2$ may have a value between $[-1, 1]$, so the calculation from angle δ_r with a value between $[-\pi/4, \pi/4]$ follows as seen in these two equations.

$$u1 = \frac{1 + \delta_r * \frac{4}{\pi}}{2}$$
$$u2 = \frac{1 - \delta_r * \frac{4}{\pi}}{2}$$

...

5.3 Conclusion

The objective of this project was to develop a control algorithm which will prevent the sailboat from a collision with an obstacle with a known position.

The control algorithm with composed vector fields takes these obstacles as repulsive vector fields, which are added to the nominal vector field, into account. It has the clear advantage of many complex trajectories that can be implemented. So this is a method not to implement just one kind of trajectory (as a line-following algorithm) but it generalizes this to follow any trajectory, that can be build out of the atomic vectorfields with the given operations on them.
The test results have shown, as desired, that with another obstacle nearby the influence of the repulsive avoidance vector is very high. This guarantees that there will be no collision, which is the primary goal. In some constellations this may lead the boat a bit off the desired trajectory, but when the other obstacle is at a greater distance again the vector field will lead the boat back closer to the ideal path.
...

6 Appendix

6.1 Basic Vector Fields

These figures show the basic vector fields. There are no axes with a scale in these figures, because they show the general vector fields which can be changed in position and size but the form would always stay the same.

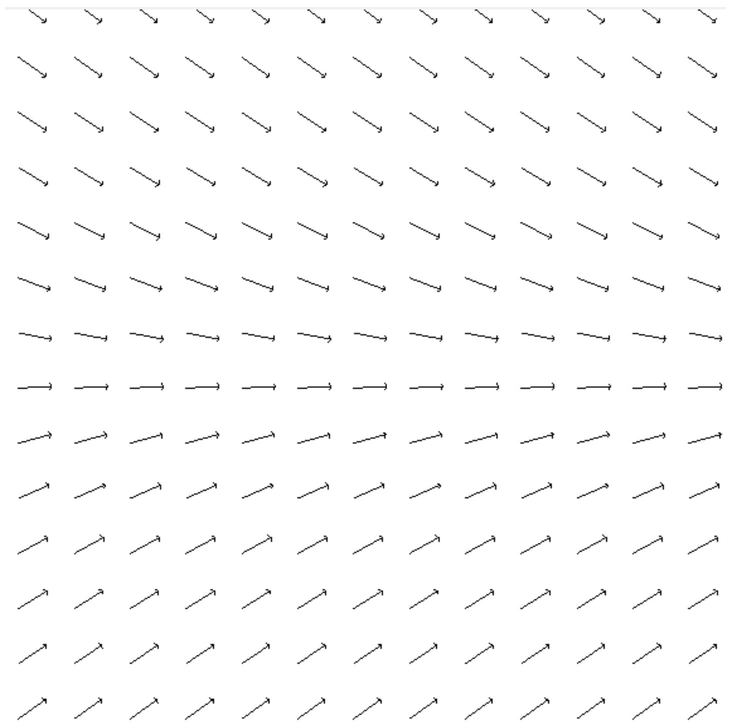


Figure 6.1: Atomic Vector Field - Line Following

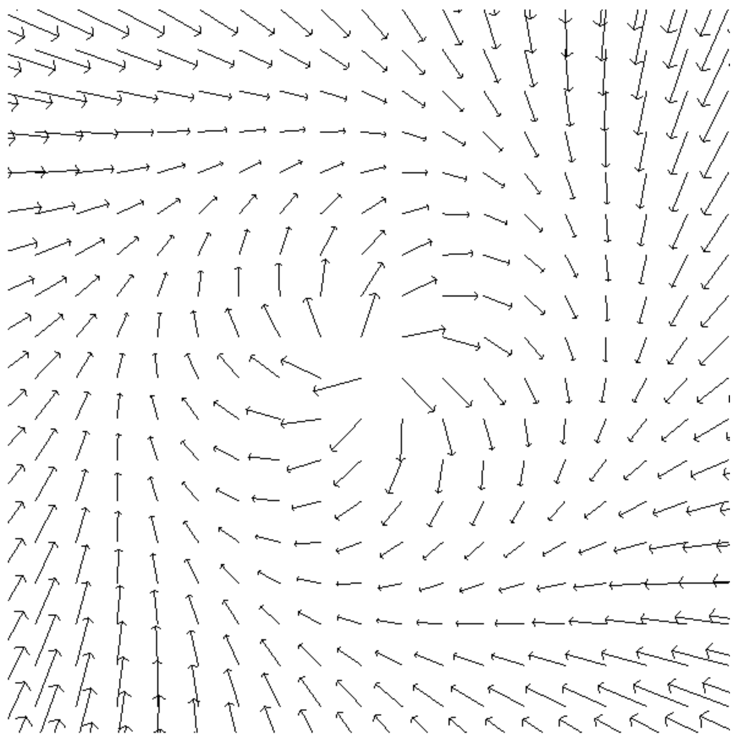


Figure 6.2: Atomic Vector Field - Circle Following

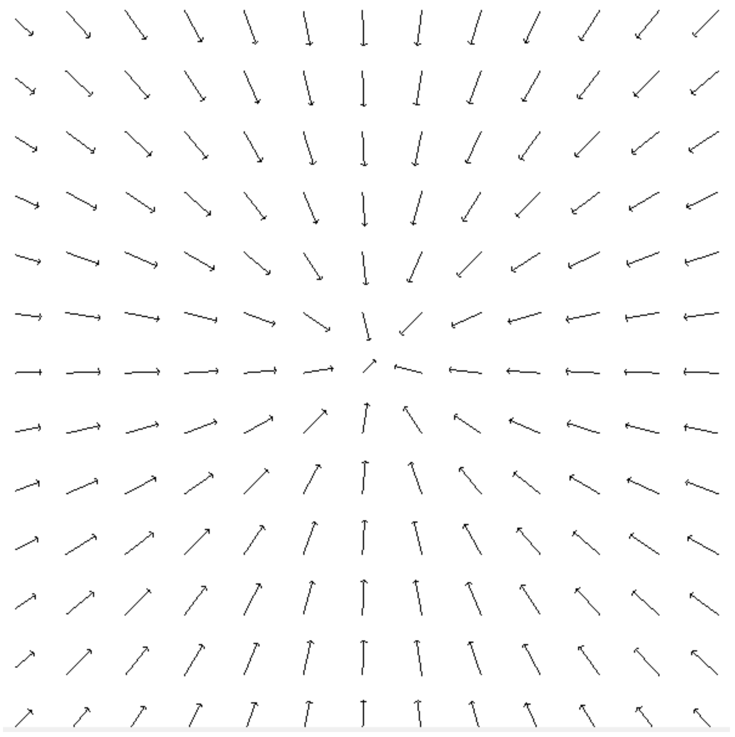


Figure 6.3: Atomic Vector Field - Attractive Point

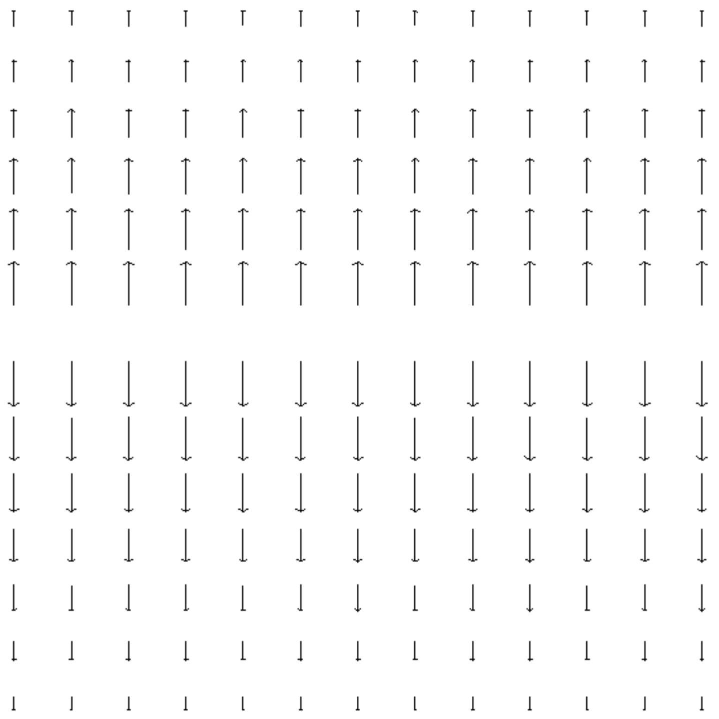


Figure 6.4: Atomic Vector Field - Repulsive Line

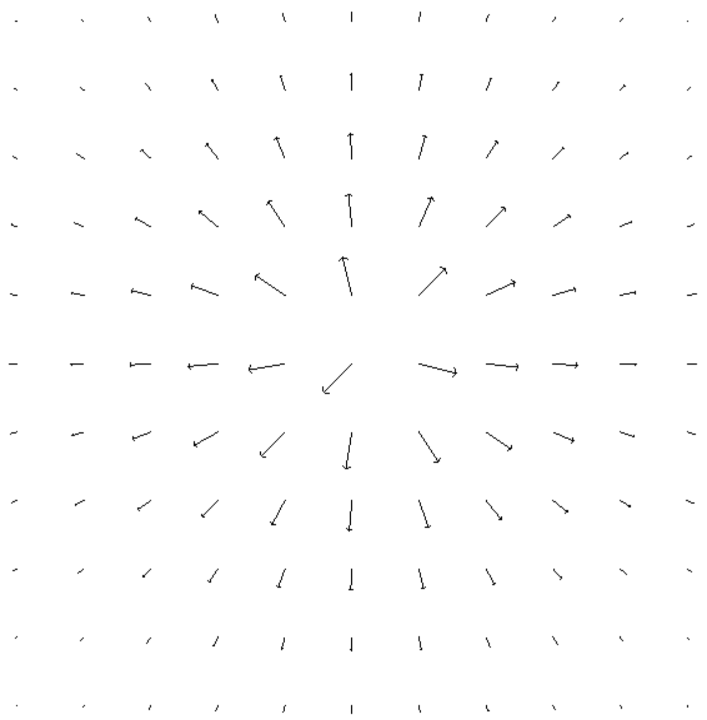
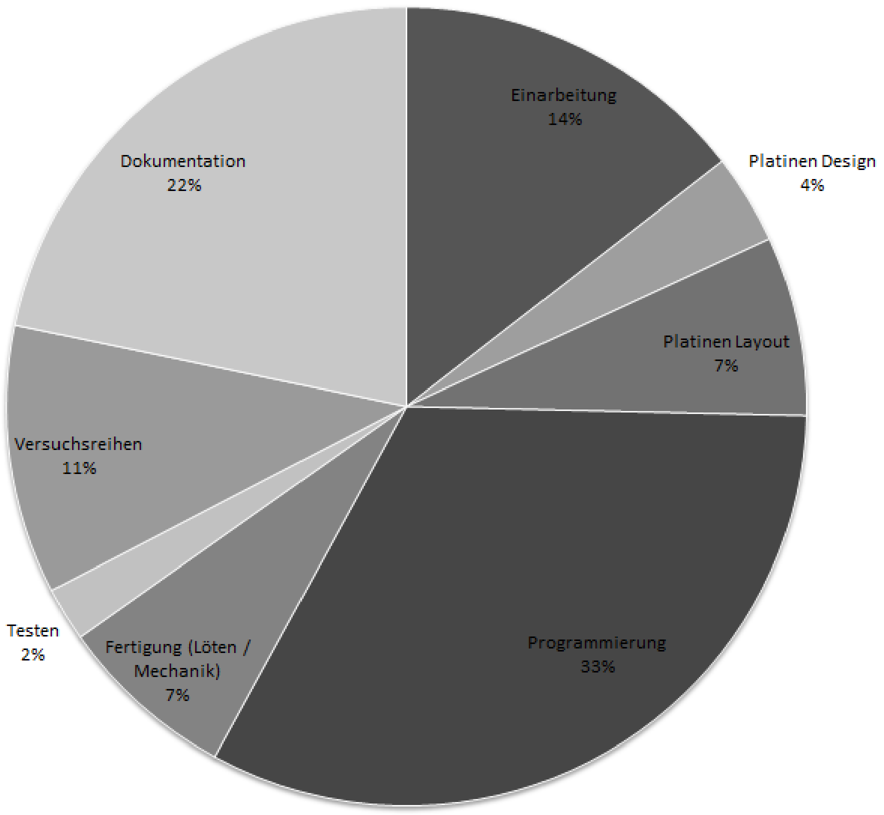


Figure 6.5: Atomic Vector Field - Repulsive Point

6.2 Time Schedule of this Thesis



Bibliography

[1] T. Berthomier, F. Nicolas, V. Leblic, P.-E. Dalidec, and K. Zhang, “Projet RASV: World Robotic Sailing Championship,” tech. rep., ENSTA Bretagne, Jan. 2013.

[2] L. Jaulin and F. Le Bars, “A simple controller for line following of sailboats,” tech. rep., ENSTA Bretagne, 2012.

[3] I. français de recherche pour l’exploitation de la mer, “Le robot voilier intelligent VAIMOS,” Jan. 2012.

[4] J. Kimball, *Physics of Sailing.* Taylor and Francis Group, LLC, 2010.

[5] S. . C. T. Ltd, “Coanda Effect.”

[6] L. Jaulin, “Commande par espace d’état,” tech. rep., ENSTA Bretagne, Nov. 2012.