

Master Thesis

COMPUTATIONAL FLUID DYNAMICS FOR SIMULATION OF WIND-TERRAIN INTERACTION IN FLIGHT SIMULATION.

Masterarbeit zur Erlangung des akademischen Grades

„Master of Science“

Verfasserin/Verfasser : **Manuel Dobusch**

Vorgelegt am FH-Masterstudiengang MultiMediaTechnology, Fachhochschule Salzburg

Begutachtet durch:

DI DI Dr. Michael Wimmer (BetreuerIn)

DI Gerlinde Emsenhuber (ZweitgutachterIn)

Salzburg, 31.05.2016

Eidesstattliche Erklärung

Hiermit versichere ich, Manuel Dobusch, geboren am **18.10.1988** in **Linz**, dass ich die Grundsätze wissenschaftlichen Arbeitens nach bestem Wissen und Gewissen eingehalten habe und die vorliegende Masterarbeit von mir selbstständig verfasst wurde. Zur Erstellung wurden von mir keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ich versichere, dass ich die Masterarbeit weder im In- noch Ausland bisher in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der den BegutachterInnen vorgelegten Arbeit übereinstimmt.

Salzburg, am 31.05.2016

Unterschrift

Manuel Dobusch

Personenkennzeichnen

Zusammenfassung

Flugsimulation ist eine wichtige Trainingsmethode für Piloten. Die Simulation von Wind ist ein wichtiger Bestandteil von Flugsimulationen. Die Effekte der Interaktion von Gelände und Wind sind oft nur approximiert. In dieser Arbeit wird eine neue Methode für die Simulation von Wind-Gelände Interaktion vorgestellt die auf Flüssigkeitssimulation basiert. Die Partikel-basierte Methode Smoothed Particle Hydrodynamics (SPH) ist die Grundlage dafür. Um den Rechenaufwand der Windsimulation zu reduzieren wird eine Methode vorgestellt um die Luftpartikel in einem relativ kleinem Bereich zu halten. Details bezüglich der Interaktion von Luftpartikeln mit harten Oberflächen werden diskutiert und eine Methode für diese Interaktion die sich für frei bewegliche Luftvolumen eignet wird vorgestellt. Numerische Stabilität der vorgestellten Methode wird ebenso behandelt. Die vorgestellte Wind Simulation wird bezüglich vertikaler und horizontaler Winddeflektion von Gelände evaluiert. Abschließend werden Limitationen in Rechenaufwand und der Simulation von turbulenter Luft diskutiert.

Schlüsselwörter:

Windsimulation, Flüssigkeitssimulation, Flugsimulation

Abstract

Flight simulation is an important mean of training for flight crews. One factor of flight simulation is the simulation of wind. The interaction of wind and terrain is often only approximated though. Based on computational fluid simulation, a new method is proposed in this work that aims to give a more detailed simulation of wind-terrain interaction. The smoothed particle hydrodynamics (SPH) algorithm comes to use. A way to reduce computational cost of the fluid-based wind simulation by containing it in a relatively small volume is presented as well as the handling of solid boundaries for freely moveable fluid volumes. Numerical stability of the proposed algorithm is addressed. An evaluation in regards to vertical and horizontal deflection of wind by the terrain surface is conducted. Finally limitations of the proposed wind simulation method in regards to performance and simulation of turbulent air are discussed.

Key words:

Wind Simulation, Fluid Simulation, Flight Simulation

Table of Content

1. Introduction	1
1.1. Contribution	2
1.2. Challenges	2
1.3. Related Works	3
1.4. Limitations	4
2. Background	6
2.1. Ridge Lift Definition	6
2.2. Turbulence Definition	7
2.3. Aviation Terms	9
3. Related Work	11
3.1. Weather systems in Flight Simulation	12
3.1.1. FlightGear	12
3.1.2. Flight Simulator X	13
3.1.3. Prepar3D	14
3.1.4. CumulsX	14
3.2. Fluid Simulation	15
3.3. Smoothed Particle Hydrodynamics	19
3.4. Solid Boundary	22
3.5. Open Boundary	25
4. Method	28
4.1. SPH Implementation	30
4.1.1. Density Calculation	31
4.1.2. Pressure Calculation	32
4.1.3. Viscosity Calculation	34
4.1.4. Body Forces and Incompressibility	36
4.2. Solid Boundary Implementation	38

4.3.	Open Boundary Implementation	40
4.4.	Rigid Body Collision	42
4.5.	Lift Force Adjustment	45
4.6.	Numerical Stability	50
4.7.	Performance Optimization	52
4.8.	Wind Force Recovery	53
4.9.	Visualization Techniques	54
4.10.	Validation Techniques	57
4.11.	Software Architecture	59
5.	<i>Evaluation</i>	64
5.1.	Incompressibility Validation	65
5.2.	Terrain Interaction Validation	70
5.2.1.	Horizontal Displacement Validation	71
5.2.2.	Ridge Lift Validation	75
5.3.	Limitations	85
5.3.1.	Performance Evaluation	85
5.3.2.	Turbulence Spectrum Evaluation	90
6.	<i>Summary</i>	94
7.	<i>Conclusion</i>	96
8.	<i>Outlook</i>	99
9.	<i>List of Figures</i>	102
10.	<i>List of Tables</i>	104
11.	<i>List of Abbreviations</i>	105
12.	<i>References</i>	105

1. Introduction

The goal of flight simulation is to recreate the systems and environment of aircrafts for visualization, research or training purposes (Robinson, Mania and Perey 2004). One aspect of flight simulation is the simulation of air currents and its effects on aircrafts. This work aims to find a method on how to improve upon common methods in desktop flight simulators for simulating these currents.

This work focuses on desktop flight simulations run on common PCs. The strength of those programs lies in their portability and affordability. They allow flight crews to train almost anytime and anywhere and are suitable for basic aircraft familiarization and procedure training (Robinson, Mania and Perey 2004). For hobby pilots they are often the only available method for training, so convincingly recreating as many aspects of flying as possible on these programs may be very valuable to flight crews.

Specifically the simulation of air masses at low altitudes is the topic of this work. Special attention lies on airflow above uneven terrain such as hills and mountains. In these environments airflow causes effects like ridge lift, upwind caused by wind hitting obstacles like hill or mountain slopes as well as turbulence. It is very important for pilots to know when and where they occur. Glider pilots need to know where to find upwind and lift. Strong downwind and turbulence is a danger for any airplane. Especially for sports pilots they are a known cause of accidents.

In regard to the mentioned wind effects and the question

Can computational fluid dynamics improve the simulation of low-level wind with respect to wind-terrain interaction for current desktop flight simulation?

a method to simulate wind at low altitudes based on computational fluid dynamics (CFD) is proposed. Air currents below 1000 meters above ground are simulated with attention to terrain interaction. A particle based fluid simulation algorithm, called smoothed particle hydrodynamics (SPH), is used. Said particles carry physical properties of the simulated fluid. The current state of the fluid is estimated by interpolating said properties between particles via weighting kernel functions to account for spatial distribution of particles.

1.1. Contribution

In the course of this work, a wind simulation technique was developed, implemented and tested. The contributions of this work are summed up as

- a wind simulation based on the SPH algorithm.
- a setup that enables an SPH-based fluid simulation to be confined within a small region which is freely movable in space. SPH particles can move within and through this region independently from the region's own movement.
- a method for flow-through boundaries that allows SPH particles to leave the fluid body while maintaining valid pressure conditions within the fluid body.
- a simple method for solid boundaries, providing correct pressure conditions where the fluid interacts with solid objects, that is suitable for the proposed freely movable fluid simulation domain.
- an implementation of the method that focuses on flexibility and modularity to allow for quick prototyping. The algorithm is split into separate operations that can be added and removed with little programming effort. Visualization and validation methods are included as well.

1.2. Challenges

Terrains used in flight simulation are potentially very large, Microsoft's Flight Simulator X for example allows to fly around the whole earth. The area of wind simulation has to be much smaller, otherwise the computational cost would be potentially too high to handle on a common desktop computer. This poses a challenge though since the particles of the SPH algorithm are usually only confined by hard obstacles, like water in a glass. Over an open terrain a SPH fluid would simply disperse.

Other works have addressed similar problems in the past. Federico et al. (2012) provide a 2D basis on which the method to confine the SPH particles within a region of interest (ROI) is developed. This region is surrounded by more particles that do not follow the SPH simulation. They rather follow a predefined flow direction, referred to as global wind direction. They carry the same physical properties as the SPH particles. Once they enter the region of interest, they will be simulated as SPH particles. When an SPH particle leaves this area it becomes one of the non-simulated particles. This allows the SPH particles to move through the region of interest, and even leave

it, without the need to have this region cover the whole terrain. In this work this idea is expanded to 3D and the ROI is made freely movable.

Moving the fluid body around introduces further issues concerning the interaction with solid bodies. When an SPH fluid collides with a solid object, such as the terrain, care has to be taken to ensure correct pressure conditions at the fluid boundary layer. Based on the work of Marrone et al. (2011) immobile particles are created underneath the surface of the terrain or other solid objects. If no particles were present below the surface the fluid particles at the boundary would not experience any pressure force when colliding with it. The reason is that a particle's pressure is calculated based on the number and closeness of surrounding particles.

As mentioned before, terrains in flight simulation can get very large, a vast amount of immobile particles would be needed to cover the entire surface. Instead it is proposed to only use enough to cover the area in contact with the fluid and move those along with the ROI.

1.3. Related Works

Other methods to model wind and wind-terrain interaction exist and are used in desktop flight simulations, current to the date of writing. The proposed method offers advantages over some of those current methods:

- Increased detail
- No manual setup needed
- No pre-processing or additional data needed

Take as an example FlightGear, an open source flight simulator originally released in 1997 and still further developed today. The program simulates ridge lift based on the terrain's shape, wind properties like speed and direction and the aircraft's position. One concrete method available in the simulator is the one proposed by Forster-Lewis (2007). Five probes are placed on the ground at different horizontal distances from the aircraft along the wind direction. The slopes between pairs of probes are calculated. The steepness of those slopes and the distance of the probes to the aircraft determines how strong the up- or downwind is. Wind speed and aircraft altitude are also factored in. With the limited number of terrain sampling points, small surface details are easily overlooked. Furthermore, all sample points are positioned on one

line, parallel to the wind direction. Sideways deflection of wind by slopes that are not exactly perpendicular to the wind direction is neglected.

In Microsoft's Flight Simulator X, at the time of writing the latest in the series, up- and downwind over terrain can be defined manually. The so-called liftboxes can be placed by the scenario designer. Those boxes provide constant lift within the box and do not factor in terrain shape or wind speed. The scenario designer needs to manually setup lift boxes and determine the adequate lift strength. Besides a lack of details this method holds a number of obvious downsides. Manual placement limits the area in which terrain induces up- and downwind can be experienced to whatever the designer can cover in a given time. Once a scenario is created this way, ridge lift will not adjust automatically to a change in wind condition.

CumulusX is a plugin for Flight Simulator X that offers automatic calculation of ridge lift, similar to Forster-Lewis' method. It requires a database of slope angles and directions of the simulation's terrain. Therefore some pre-processing and additional data-storage is required to use CumulusX.

1.4. Limitations

The proposed SPH-based wind simulation method has several limitations. The simulation in this work is strictly confined to a relatively small area, e.g. 1 km³ and close to ground, below 1 km. The size of the simulation domain can be increased at the expense of spatial resolution and computation performance. Movement of air beyond the small simulation domain is out of the scope of this work. A steady global wind speed and velocity is assumed as input for the SPH-based method.

Turbulence is not well modelled by the proposed SPH simulation. In reality, wind-terrain interaction creates eddies. Those are experienced by aircraft as turbulence. The SPH method's accuracy in regard of those turbulences is examined, but recreating the phenomenon faithfully is out of scope of this work.

Note that some properties of the atmosphere are omitted in the proposed SPH method. Humidity and air temperature are important to simulate effects like thermal lift or cloud formation. They also influence the performance of aircrafts. However this work focuses on air movement due to pressure forces and interaction of air with a simple triangle mesh terrain.

Even though this method was developed with desktop flight simulation in mind, the necessary performance optimizations to achieve real-time are out of scope. The SPH algorithm is computationally expensive and simple implementations are unlikely to achieve real-time with a useful particle resolution.

2. Background

In this chapter, a brief description of ridge lift and air turbulence is given in 2.1 and 2.2. These phenomena will be referred to repeatedly throughout the paper. Some basic aviation terms that will be used are briefly explained in 2.3.

2.1. Ridge Lift Definition

As stated by Dennis Pagen (1992), ridge lift, also known as slope lift, may occur when air flows over an obstacle. If the obstacle's horizontal expanse perpendicular to the wind direction is great enough, air will flow over it. The air stream is deflected upwards at the obstacle's side that is facing into the wind. Figure 1 displays the flow of air around or over such obstacles.

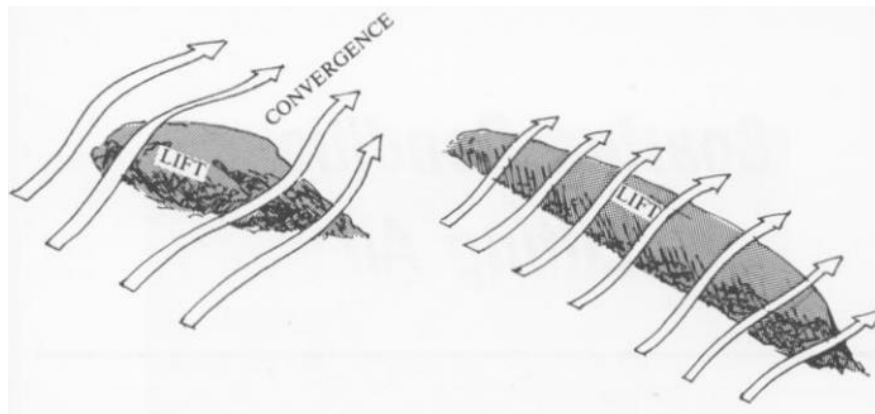


Figure 1: Obstacles deflecting flowing air. The left obstacle is narrow and air flows around. The wide obstacle to the right deflects air upwards, causing ridge lift. Image from Pagen 1992

The intensity of ridge lift depends on various factors like wind speed, wind direction relative to a given slope, steepness of a given slope and surface properties of this slope. Properties of the atmosphere, such as temperature and humidity, play a role as well. Those will, however, not be considered within the scope of this work.

Generally, the higher the wind speed and the steeper the slope is, the stronger will the resulting upwind be. The area in which the upwind expands will also increase with wind speed and slope angle. This area is also referred to as envelope.

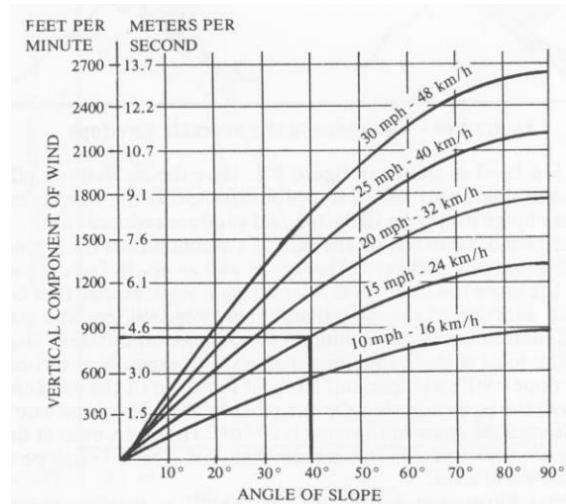


Figure 2: Upwind based on slope angle and wind velocity. The steeper the slope and the stronger the wind, the stronger the resulting upwind will be. Image from Pagen 1992

Figure 2 shows upwind strength for a given slope angle and wind speed, assuming that wind direction is perpendicular to the slope. Note that both meters per second and feet per minute are stated for upwind strength. In aviation in western countries vertical speed and altitudes are commonly stated in feet per minute or feet respectively.

2.2. Turbulence Definition

Pagen (1992, 113) gives the working definition “the random chaotic swirling of air” for the term turbulence. They are caused by eddies in the air, which are also called rotors. When such a swirl passes a point of interest (POI) rapid changes in wind speed and direction are experienced there, as seen in figure 3.

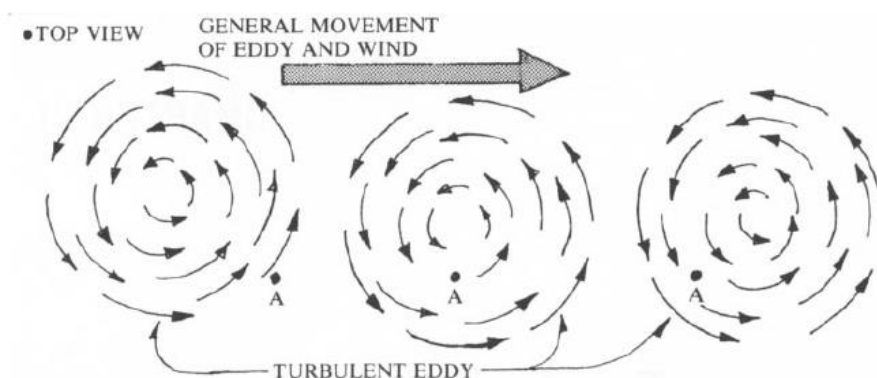


Figure 3: An eddy flowing by point A. At this point rapid changes in wind speed and direction are observed. This is referred to as turbulence. Image from Pagen 1992

Depending on the size of the eddies, an aircraft experiences turbulences differently. Small eddies of few centimeters in diameter cause rapid bumps. Larger eddies, with diameters up to the aircrafts wingspan, may cause control issues and strong bumps. Eddies even larger than that cause sudden ascents, descents and even rolling and yawing. Possible consequences in severe cases are stalling, where the aircraft's wings cease to produce sufficient lift, or pitch overs, where the aircraft is turned upside down (Pagen 1992).

Turbulence may be caused due to different reasons. In this work, mechanically induced turbulence is of interest. When air flows around an obstacle it will cause eddies. Usually those eddies are stationary, remaining at said obstacles. They may however sometimes be blown downstream. The stronger the wind, the bigger and more intense the eddies get, and the further they are carried off downstream, as shown in figure 4 (Pagen 1992).

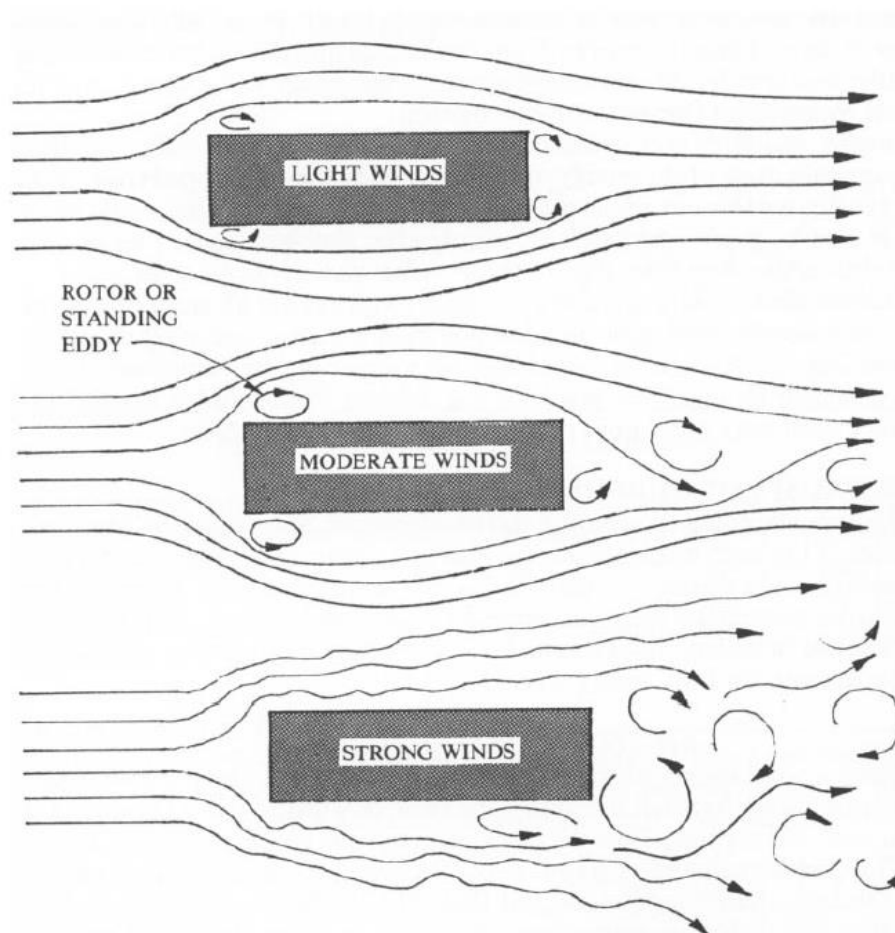


Figure 4: Mechanically induced eddies around an obstacle. From top to bottom wind strength increases, causing the air to become more and more turbulent. Image from Pagen 1992

Other causes for turbulence include thermals and shears. Thermals are warm, ascending air masses. Shears occur on boundaries of air masses moving with different speeds or even different directions. Both, thermals and shears, are out of scope of this work.

2.3. Aviation Terms

As a number of terms common in aviation will be used throughout the paper, an overview with brief explanations is provided here.

Ailerons are the control surfaces on the wings. They roll the aircraft around its longitudinal axis.

Air Speed is the speed of an aircraft relative to the surrounding air. One can further distinguish between indicated and true airspeed. True airspeed is the airplane's actual speed relative to the surrounding air. Indicated airspeed is measured via a tube with a forward facing opening, the so-called pitot tube. The pressure of the air flowing into the pitot tube is the basis for the indicated air speed. With increasing altitude, air density decreases. This causes the pressure in the pitot tube to decrease. Therefore, indicated air speed decreases with increasing altitude, even if true air speed remains the same.

Control Surfaces are the movable parts of an aircraft's wings and stabilizers. They are used to control the aircraft's attitude.

Course is the actual travel direction of an aircraft. It is the sum of the aircraft's velocity vector in heading direction and the wind velocity vector.

Elevator is the control surface on the horizontal stabilizer. It rotates the aircraft around the lateral axis.

Ground Level is the altitude of a point of terrain above sea level. In western countries, in the context of aviation, this is usually given in feet. In this work, however, meters are used.

Ground Speed is the speed of an aircraft relative to the ground. It is the sum of true air speed and wind speed.

Heading is the direction that the airplane is facing. In calm wind conditions, this is the direction in which the aircraft will travel.

Rudder is the control surface of the vertical stabilizer and rotates the aircraft around the vertical axis.

Sea Level, or mean sea level, is the average surface level of the ocean.

Variometer is an instrument that indicates the rate of ascent or descent, usually in feet per minute. The displayed rate of change is derived from changing static air pressure around the aircraft due to altitude change.

Wingspan of an airplane is the distance between the wingtips. For helicopters the rotor diameter is equivalent to the wingspan.

3. Related Work

This work is based upon CFD and explores its usefulness for flight simulation. More specifically, the simulation of air flow using CFD is explored. There are several examples of previous works dealing with similar cases. Lee, Sezer-Uzol, Horn, Long and Lyle (2005) used CFD to simulate the air wakes around ships during helicopter take offs and landings. What they did was somewhat similar to what is done in this work. Although, they did focus on a very specific setting and a small spatial domain, the area directly around a ship. Also, they worked on high fidelity flight simulations, which commonly include specialized hardware and cockpit replications, rather than desktop flight simulations.

Galway (2009) uses CFD to predict wind and turbulence patterns in urban environments. He uses these predictions to enhance the performance of unmanned aerial vehicles (UAVs) in urban environments.

Porté-Agel et al. (2011) make use of large-eddy simulation (LES) to predict airflow at planned wind turbine sites. Their focus is on predicting how the air flow will be influenced by placement of wind turbines and the wakes caused by these turbines. The goal is to predict an optimal configuration of wind turbine sites, so that each turbine can achieve a maximum efficiency. LES is a field of CFD that deals with simulating the large-scale part of turbulent flows (Mathew 2010).

This work was developed with desktop flight simulation in mind. In chapter 3.1, models for turbulence and ridge lift implemented in flight simulations, current to the time of writing, are investigated. Models for both effects, available in FlightGear, Microsoft Flight Simulator X and Prepar3D by Lockheed Martin, are presented. Furthermore CumulusX, a plugin for Flight Simulator X, is discussed. Its purpose is to improve the simulation of ridge lift and thermals to provide better environmental prerequisites for flying gliders.

The SPH algorithm is used as the basic method for simulating fluids in this work. More specifically, wind over a large terrain will be simulated. Fluid simulation in general is based on the Navier-Stokes equations. Those equations describe the movement of bodies of fluid and will be discussed in chapter 3.2. After covering those basics, the SPH algorithm will be discussed in some detail in chapter 3.3, followed by a discussion on techniques for solid and open boundaries in chapters 3.4 and 3.5.

3.1. Weather systems in Flight Simulation

In the following, an overview over some popular desktop flight simulations for civil aviation is given. The weather systems of those simulations are examined to provide the reader with an overview of currently used methods for ridge lift and turbulence simulation or modeling.

3.1.1. FlightGear

FlightGear, as previously mentioned in chapter 1, is an open source flight simulation that was originally released in 1997 (FlightGear Wiki. n.d.). As was briefly mentioned, flight gear does simulate ridge lift by implementing a method proposed by Forster-Lewis (2007).

It was already briefly explained that this method uses terrain altitude at a small number of locations near the aircraft to determine terrain slopes, which are the basis for calculating up- and downwind. Five locations, called probes, are chosen along an axis, hinging on the aircraft, parallel to wind direction. The horizontal distances of those probes are 0, 250, 750, 2000 and -100 meters in wind direction. From those points, five slopes, or in other words the altitude differences between neighboring probes, are calculated. Furthermore, slopes are weighed by distance and lift is adjusted for airplane altitude above ground. At very low altitude ground friction and small obstacles dissipate lift. At high altitudes, above 130 meters, lift dissipates as well. In between the full effect occurs.

This method was implemented to serve as reference for testing ridge lift effects in the proposed SPH-based method. A simple drone, an object without a flight model but just a constant velocity, using this lift model is used as base line. Implementation details are provided in chapter 4.10 and results are provided and discussed in chapter 5.2.

As far as turbulence goes, FlightGear offers a number of implementations for the user to choose from. Several possible turbulence models are implemented within JSBSim, one of two flight models currently offered in the simulation.

JSBSim's default turbulence model takes random values, in a range of zero to one, for each directional axis. This is the basis for the resulting turbulent force vector. The horizontal force components are then reduced, and the overall force is adjusted for ground proximity. If the point of interest is lower than three wingspans over

ground, the force is reduced exponentially with zero force at ground level. Finally the forces are adjusted for aircraft dimensions, namely wingspan and tail length. The other two models are similar with differences in details like influence of aircraft size or atmospheric properties (Berndt et al. 2009).

3.1.2. Flight Simulator X

Flight Simulator X (FSX) is the latest installation in Microsoft's Flight Simulator series as of the time of writing. It was originally released in 2006. The first title of the series was released in 1982. Microsoft released a predecessor to FSX called Microsoft Flight. Development and support for Flight was cancelled in 2012 though, and multiplayer servers were closed in 2014. In the same year a new build of FSX was released, therefore it is considered the more current product.

Vertical air movement in FSX is implemented as so-called thermal descriptions. They have a number of properties, describing potential thermals, including a range of possible lift and turbulence intensity, a range of possible thermal size, and potential influence altitude. Each thermal description is associated with a landscape type. (Microsoft n.d.)

FSX partitions its terrain into squares of 1.2 kilometers side length, each of which is assigned a landscape type. Those types are based on the Olson Land Classification table. This table is based on the work of Olson, Watts and Allison (1983) who created a global database on major world ecosystems. The original use of the table and database was to map global carbon density to different ecosystems. Those ecosystems are grouped in the categories

- Forest and Woodland
- Interrupted Woods
- Mainly Cropped, Residential, Commercial, Park
- Grass and Shrub Complexes
- Tundra and Dessert
- Major Wetlands
- Other Coastal, Aquatic and Miscellaneous

Thermals result from ground getting heated up by the sun. Air over hot ground heats up and rises. Woodland, for example, heats up slower than a patch of sand or a large area of tarmac. Therefore, it makes sense to have different lift definitions for different land classifications.

Ridge lift can be simulated by placing so-called lift boxes. A lift box is a cube with a given side length, position, and orientation. A scalar defines the rate of up- or downwind relative to the horizontal wind speed in the box. So if the scalar is 0.5 and the wind speed is 8 m/s the upwind would be 4 m/s. If the scalar is negative it results in downwind.

Placing lift boxes is done by scenario designers. This means that ridge lift does not automatically result from wind. Designers have to be careful to match placed lift boxes with the wind conditions of the scenario. The up- and downwind has to be positioned manually according to wind direction and ridge orientation and match the lift strength with wind speed.

3.1.3. Prepar3D

Prepar3D, pronounced ‚prepared‘, is a desktop flight simulation by Lockheed Martin. Lockheed Martin is an aircraft developer and manufacturer based in the USA. Prepar3D is based on Microsoft ESP. ESP contains a number of tools to enable simulation of any kind of real world vehicle or object. Like in FSX, in Prepar3D those objects are primarily aircrafts. This means that what was described above in chapter 3.1.2 applies to Prepar3d as well.

3.1.4. CumulusX

CumulusX is an add-on for FSX. It was created to provide scenarios suitable for gliding. To stay airborne, gliders rely on upwards moving air, such as ridge lift and thermals. CumulusX provides automatic generation and placement of thermals according to weather and terrain conditions, as well as generation of ridge lift.

According to the provided documentation of CumulusX, ridge lift is based on the terrain slope, wind speed and wind direction (Lürkens 2015). Two areas, one upwind and one downwind, are used to calculate lift. The size of those areas depends on the altitude of the aircraft above ground. By increasing the area size at high altitudes minor terrain features have less influence. In older versions of the plugin, information on the slope angle of a given point of the terrain is stored in a slope data base, which has to be created based on given terrain data.

Calculated lift is reduced according to altitude layers. The first layer starts at ground level and reaches up to 20 meters above ground. The lift factor is 0 at ground and increases back to nominal value at 20 meters. Beyond 20 meters the lift factor is

reduced with increasing altitude. The rate of lift reduction depends on the steepness of the slopes, encountered at the area of interest. Steep slopes will cause higher rising upwind than shallow slopes.

Thermals are created by placing a large number of inactive ones at random and then analyzing the ground and weather conditions at each thermal. Only those with favorable conditions and relatively close to the aircraft are activated. The thermal parameters are adjusted according to season, terrain slope and surface characteristics. Surfaces which heat up well under sunshine, like sand or tarmac, will cause stronger thermals. Atmospheric conditions may alter or limit thermals. They are generally limited to the altitude of the cloud base layer and appear with different strength and frequency in different seasons of the year.

3.2. Fluid Simulation

There are two basic viewpoints commonly used in fluid simulation, the Eulerian and the Lagrangian viewpoint.

The Eulerian viewpoint is named after the Swiss mathematician Leonhard Euler. Fixed points are placed in space at which properties of the fluid, like density, velocity or temperature are measured. When the fluid is moving those measurements will change over time, as the fluid moves past those fixed points. For example, there could be a volume of warm fluid followed by cold fluid. As they move by a measuring point, the temperature at that point will change from warm to cold. Besides movement of the fluid the measurements may also change because of dissipation or a global change in the fluid. If warm water is poured into a pool of cold water the temperature of the warm water will dissipate until the pool has an even temperature. The temperature may also decrease or increase globally over time, e.g. because of sun light.

The Lagrangian viewpoint is named after the mathematician Joseph-Louis Lagrange. Fluids are treated as sets of particles. One might think of each particle as being one molecule of the fluid, although, those particles may be much bigger than actual molecules. So one should keep in mind that one particle may represent a whole packet of fluid molecules. Besides having a position and velocity, those particles may also have other properties, like density or temperature. Besides collision, particles may interact in other ways like attraction or dissipation.

Bridson (2008) provides an intuitive example on how to think of those two methods. One may imagine two ways of doing a weather report. The Lagrangian way

would be to release a weather balloon, which will move with a volume of air and measure change of temperature, humidity and whatever is of interest in that volume. The Eulerian way would be to place a weather station on the ground and measure the change of interesting properties over time at this fixed point.

From a practical point of view, both ways of thinking have advantages and disadvantages. An implementation of the Lagrangian view would simulate fluids as particle systems. This resembles more closely how one might think of fluids in physics. The Eulerian view on the other hand would be implemented as a fixed grid. It is easier to work out spatial derivatives on such a grid. Desbrun, Kanso and Tong (2006) describe how to do so on arbitrarily shaped grid meshes, while Elcot et al. directly apply those methods to fluid simulation. Conservation of mass and boundary conditions on the other hand are solved easier in particle based methods. Lagrangian simulations are also free to move in space, while Eulerian simulations are confined to the space covered by their grid. Choosing one or the other approach comes down to the requirement of the individual use case.

The motion of fluids is described by the partial differential Navier-Stokes equations. The equation describing the acceleration of the fluid, called the momentum equation, is

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u} \quad (3.1)$$

where \vec{u} stands for velocity, ρ stands for the fluid's density and p for pressure. \vec{g} is called the body force and ν is the kinematic viscosity of the fluid.

The different terms of the equation describe different forces that influence the fluid's motion. Bridson (2008) provides an intuitive way of thinking about each term. Imagine any fluid as being made up from small particles. Each particle would have mass m , volume V and velocity \vec{u} . Now the terms of the momentum equation can be seen as forces acting on those particles. To support this way of looking at it, the momentum equation is rewritten as

$$m \frac{D\vec{u}}{Dt} = m\vec{g} - V\nabla p + V\mu \nabla \cdot \nabla \vec{u} \quad (3.2)$$

The term $m\vec{g}$ describes an external force that acts evenly on the whole fluid. A typical example would be gravity. It can also be the sum of multiple forces, like gravity and wind. $V\nabla p$ describes the pressure force acting upon a particle of fluid. ∇p is the gradient of pressure at a given point. Since the gradient of a function has the direction of steepest ascent, and the pressure force would act in direction of lower pres-

sure, the negative of the gradient is used. The particle is supposed to move towards low pressure, so in direction of steepest descent. To approximate integration of the pressure force over the volume of the particle the gradient is multiplied by the particle volume V .

$V\mu\nabla \cdot \nabla \vec{u}$ describes the viscosity of the fluid. Fluids with high viscosity try to resist deformation, while fluids with low viscosity deform quickly. Honey for example has a higher viscosity than water. $\nabla \cdot \nabla$ is the so called Laplacian operator, which measures the difference of a quantity at a point from the average of this quantity around that point. The Laplacian is often alternatively written as ∇^2 . In this case the quantity in question is the velocity \vec{u} . The factor μ is the dynamic viscosity coefficient. As described above in regards to the pressure force, V is used to approximate integration over the particle's volume.

To reduce errors introduced by the approximations done in the above terms, the number of particles in a volume of fluid can be increased. The more particles, the smaller the volume and mass per particle gets. This poses a problem when the number of particles goes to infinity because the volume and mass per particle goes to zero. Dividing by V takes care of this problem. This results in the equation

$$\frac{m}{V} \frac{D\vec{u}}{Dt} = \frac{m}{V} \vec{g} - \nabla p + \mu \nabla \cdot \nabla \vec{u} \quad (3.3)$$

m/V equals the fluid density ρ . By dividing by ρ , defining the kinematic viscosity ν as μ/ρ and rearranging the terms the equation becomes

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u} \quad (3.4)$$

Now the equation is almost back to the initial form seen in equation 3.1. The difference is the first term $\frac{D\vec{u}}{Dt}$. This is called material derivative. To get the rate of change in time t for a value described by a function at a point \vec{x} in space of the form $q(t, \vec{x})$ one has to take the derivative of this function

$$\frac{d}{dt} q(t, \vec{x}) = \frac{\partial q}{\partial t} + \nabla q \cdot \frac{d\vec{x}}{dt} \quad (3.5)$$

$$= \frac{\partial q}{\partial t} + \nabla q \cdot \vec{u} \quad (3.6)$$

$$\equiv \frac{Dq}{Dt} \quad (3.7)$$

Pedley (1997) shows how the momentum equation can be derived from Newton's laws of motion. The first law of motion states that whenever an object changes its

state of motion, a force is applied. The second law, as stated in equation 3.8, shows that the acceleration from said force equals the force divided by the object's mass.

$$F = ma \quad (3.8)$$

The momentum equation gives the momentum m_p for a body of fluid caused by internal and external forces. Momentum is defined as

$$m_p = mv \quad (3.9)$$

Changes in the velocity v are caused by internal and external forces. So equation 3.1 describes how the momentum within a body of fluid changes over time. Note that p is the common symbol for momentum. In the rest of this work p is used for pressure though, so m_p is used instead for momentum.

Many fluids have an almost constant volume. Small changes in density and pressure, and therefore volume, occur in fluids in the form of sound waves. That is true for liquids and gases alike, although it is harder to change the volume of a liquid. Extreme situations, like putting the fluid into a pump, or sonic booms, may change the volume more visibly (Bridson 2008). The change of volume caused by sound waves is negligible when dealing with fluid motion on a macroscopic level (like waves on a sea shore). Other than that, fluids do not change volume under normal circumstance, so for many simulation cases it is acceptable to assume that a fluid is incompressible. This incompressibility can be expressed as

$$\nabla \cdot \vec{u} = 0 \quad (3.10)$$

In literature this is often referred to as the incompressibility constraint.

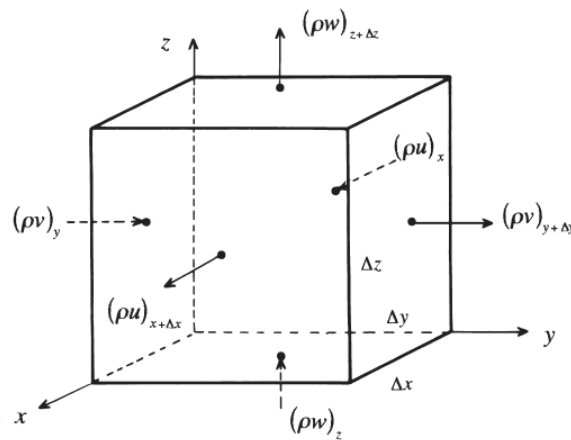


Figure 5: Flow in and out of a rectangular region in space. For a fluid to have constant volume the sum of in- and outflow must be zero. Image from Pedley 1997

To derive this formula, imagine a constant arbitrary rectangular volume in a fluid. The amount of mass within this volume will change with the rate of fluid entering the volume across its surface. Figure 5 illustrates the flow in and out of such a volume. Formally, this is described as

$$\begin{aligned} \Delta x \Delta y \Delta z \frac{\partial \rho}{\partial t} = & \Delta y \Delta z ([\rho u]_x - [\rho u]_{x+\Delta x}) \\ & + \Delta z \Delta x ([\rho v]_y - [\rho v]_{y+\Delta y}) \\ & + \Delta x \Delta y ([\rho w]_z - [\rho w]_{z+\Delta z}) \end{aligned} \quad (3.11)$$

Δx , Δy , Δz are the side lengths in each dimension of the volume. u , v and w are the components of velocity vector \vec{u} . So equation 3.11 describes how much mass is entering the volume at each face. Dividing this by $\Delta x \Delta y \Delta z$ and taking the limit results in

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x}(\rho u) - \frac{\partial}{\partial y}(\rho v) - \frac{\partial}{\partial z}(\rho w) \quad (3.12)$$

In short equation 3.12 can be written as

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \vec{u}) \quad (3.13)$$

since the divergence operator $\nabla \cdot$ is defined as

$$\nabla \cdot \vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \quad (3.14)$$

Equation 3.13 expresses that the density in a given volume increases if divergence in that field is negative. In other words, if the fluid converges on that volume, density increases. It was stated earlier that it is sufficient to assume that the simulated fluid is incompressible, so the density may not change over time. Equation (3.15) expresses that the divergence in any volume of fluid must be zero. If the fluid can not diverge at any given point, its volume can not decrease. Therefore equation 3.15 is called the incompressibility constraint.

$$\nabla \cdot \vec{u} = 0 \quad (3.15)$$

3.3. Smoothed Particle Hydrodynamics

Smoothed particle Hydrodynamics (SPH) is a Lagrangian method that was introduced in 1977 by R. A. Gingold and J. J. Monaghan (1977). It was initially applied to complex problems in astrophysics that lack spherical symmetry. Solving said problems numerically on a grid would have greatly increased complexity. A symmetric

problem would need N grid points on a 1 dimensional grid while a asymmetric problem would need N^3 grid points.

SPH uses parcels of fluid that move according to forces such as pressure, gravity, or magnetic forces. Those parcels will be called particles. The fluid equation only needs to be solved at the positions of said fluid particles. They are distributed randomly according to the fluid's density. To actually get the density at each particle position the smoothed kernel method introduced by Bartlett in 1963 and Paren in 1962 is used. This method can be thought of as an approximation to an integral according to the Monte Carlos procedure. (Gingold and Monaghan 1977)

To interpolate an arbitrary quantity A , the SPH algorithm uses the equation

$$A_I(r) = \int A(r')W(r - r', h)dr' \quad (3.16)$$

where r is the position at which to interpolate and r' is the position from which to interpolate. W is the kernel function and h is the kernels support radius. W weighs the quantity A at distance r' to account for decreasing influence at increasing distance. The interpolation is integrated over the kernel's support radius. To apply this interpolation to a fluid, and ultimately to SPH, the fluid needs to be treated as a set of elements. An element a has mass m_a , density ρ_a and position r_a .

$$A_I(r) = \int \frac{A(r')}{\rho(r')} \rho(r') dr' \quad (3.17)$$

When discretizing the fluid into particles the integral of equation 3.17 is approximated by

$$A_I(r) = \sum_N m_N \frac{A_N}{\rho_N} W(r - r_N, h) \quad (3.18)$$

N is the number of all other particles. Since the kernel function W is 0 beyond h , in practice only the particles closer than h have to be considered. Therefore N will only refer to the particles closer than h , the so-called neighbors.

As an example consider the interpolation of density ρ at point r .

$$\rho(r) = \sum_N m_N W(r - r_N, h) \quad (3.19)$$

In practice, r is the position of a particle. Theoretically it can be a non-particle position as well. Considering this it becomes clearer that, when estimating the density at the particle with position r , N must include this particle as well. Its own mass influences the density at r just like the mass of the neighbours.

Density estimation is the first step when calculating pressure based forces in fluids. The next step is to calculate the pressure at each particle, and based upon that

the pressure gradient. Assuming no other forces, like gravity or magnetism, the particles will move along the pressure gradient. Pressure p is calculated as

$$p = k\rho \quad (3.20)$$

where k is a gas constant. The influence of temperature on pressure forces can be accounted for by modifying k accordingly. Desbrun and Gascuel (1996) propose an alternative equation that allows to define the density the fluid has when settled

$$p = k(p - p_0) \quad (3.21)$$

where p_0 is said density of the settled fluid. It is called rest density. The pressure force follows the negative gradient of the pressure. The gradient of a function is directed towards the steepest ascent. Pressure however pushes objects towards lower pressure, therefore the negative of the pressure gradient comes to use.

$$f_i^{pressure} = -\nabla p(r_i) \quad (3.22)$$

In terms of integration approximation this becomes

$$f^{pressure} = -\sum_N m_N \frac{p_N}{\rho_N} \nabla W(r - r_N, h) \quad (3.23)$$

Where ∇W is the gradient of the kernel function. In practice, this calculation will most likely result in asymmetric forces. Consider two particles a and b . a uses the pressure at b to calculate the gradient and b uses the pressure at a . If the pressure is not equal at those particles it will result in different forces, violating Newton's third law. An alternative formulation that respects Newton's third law is proposed by Müller, Charypar and Gross (2003)

$$f^{pressure} = -\sum_N m_N \frac{p+p_N}{2\rho_N} \nabla W(r - r_N, h) \quad (3.24)$$

$\frac{p+p_N}{2\rho_N}$ means that the pressure at the particle of interest and the current neighbor are considered. Therefore the calculation results in the same pressure force at both particles.

To evaluate the viscosity term $\mu \nabla^2 v$ of the Navier-Stokes equation at a particle, the velocities at its neighbors are considered. Based on equation 3.18, the viscosity force can be calculated as

$$f^{viscosity} = \mu \sum_N m_N \frac{v_N}{\rho_N} \nabla^2 W(r - r_N, h) \quad (3.25)$$

which may again result in asymmetric forces. This becomes clear when again considering two particles a and b , which may have different velocities. Müller, Charypar and Gross (2003) propose an alternative to this as well. They point out that viscosity forces depend on velocity differences, rather than on absolute velocities.

Considering this, the following alternative to equation 3.25 can be expressed as

$$f^{viscosity} = \mu \sum_N m_N \frac{v_N - v}{\rho_N} \nabla^2 W(r - r_N, h) \quad (3.26)$$

Body forces g are trivial to apply. Their calculation, if there is any necessary, is not part of the fluid simulation. They are applied equally to each particle.

As Gingold and Monaghan (1977) pointed out, kernels are required to have compact support. Beyond the support radius they evaluate to zero. This is a performance requirement, this way particles beyond the support radius can be ignored.

Another requirement for kernel functions is to be continuously derivable twice. Considering equations 3.24 and 3.26 one can see that the gradient $\nabla W(r, h)$ and the Laplacian $\nabla^2 W(r, h)$ are used. The gradient and the Laplacian of a bilinear function $f(x, y)$ are defined as

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \quad (3.27)$$

$$\nabla^2 f(x, y) = \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right) \quad (3.28)$$

Gingold and Monaghan (1977) used a Gaussian kernel in their original paper. In one dimension this kernel has the form

$$W(r, h) = \frac{1}{h\sqrt{\pi}} e^{-(x^2/h^2)} \quad (3.29)$$

Lucy (1977), who in 1977 independently developed a method equivalent to SPH used the kernel

$$W(r, h) = \frac{105}{16\pi h^3} \left(\frac{1-r}{h} \right)^3 \left(\frac{1+3r}{h} \right) \quad (3.30)$$

The Poly6 kernel was designed by Müller, Charypar and Gross (2003) with computational performance in mind. r only appears squared, therefore, taking the square root when calculating the eulerian distance between two particles can be avoided.

$$W(r, h) = \frac{315}{64\pi h^9} (h^2 - r^2)^3 \quad (3.31)$$

3.4. Solid Boundary

Fluids often need to interact with solid objects, such as containers. In a Lagrangian method, such as SPH, it would be easy to simply implement rigid body collision between the fluid particles and the solid objects. However, this causes simulation errors. The calculation of density at particles close to the solid boundary would be erroneous. Since there are no particles present beyond the solid boundary to influence

density estimation at those particles close to the boundary, it will result in an underestimation of density.

A common solution to fix the mentioned issue in density calculation is to use particles on the other side of the solid boundary. One such scheme is presented by Colagrossi and Landrini (2003). They propose a method called ghost particles. At each time step the particles within a layer with thickness of the kernel support radius are mirrored on the opposite side of the solid boundary. The position of the mirroring particles is based on the position of the mirrored particle and the boundary. The distance of the mirrored particle to the boundary is the same as the distance of the mirroring particle to the boundary. The position of the ghost particle is given by

$$r_{iM} = 2r_w - r_i \quad (3.32)$$

r_i being the position of the mirrored particle, r_{iM} being the position of the mirroring particle and r_w the position of the point of the boundary by which to mirror.

Velocity components tangential and normal to the boundary are mirrored in the following way

$$v_{iMn} = 2V_{wn} - v_{in} \quad (3.33)$$

$$v_{iMt} = v_{it} \quad (3.34)$$

where $_n$ denotes the normal velocity components and $_t$ the tangential component to the boundary. So the tangential velocity component of the mirroring particle v_{iMt} is equal to the tangential velocity component of the mirrored particle v_{it} . Therefore, the boundary will behave as a free-slip boundary. There is no friction tangential to the surface. A no-slip version has not been proposed by Colagrossi and Landrini (2003). The normal velocity component v_{iMn} , however, is influenced by the velocity of the solid object V_{wn} .

The pressure of the mirroring particle is simply identical to the pressure of the mirrored particle

$$p_{iM} = p_i \quad (3.35)$$

Marrone et al. (2011) propose an alternative method based on ghost particles, called fixed ghost particles. Other than in the normal ghost particle method, the fixed ghost particles are created only once at the beginning of the simulation at fixed positions. The particles are arranged in a layer beginning just below the surface with a thickness of support kernel radius. To find the positions of particles, the surface is assumed to be a spline discretized into regularly spaced points. Normals and tan-

gents along the surface can therefore be calculated based on said spline, with the normals facing into the solid object. The points defining the discretized spline are duplicated in direction of the normals to define a new spline. This new spline needs to be discretized again into evenly spaced points to repeat the process until enough points are created to fill a layer of kernel support radius thickness. The fixed ghost particles take the position of the discrete points along each spline. Figure 6 illustrates fixed ghost particles along a bend surface.

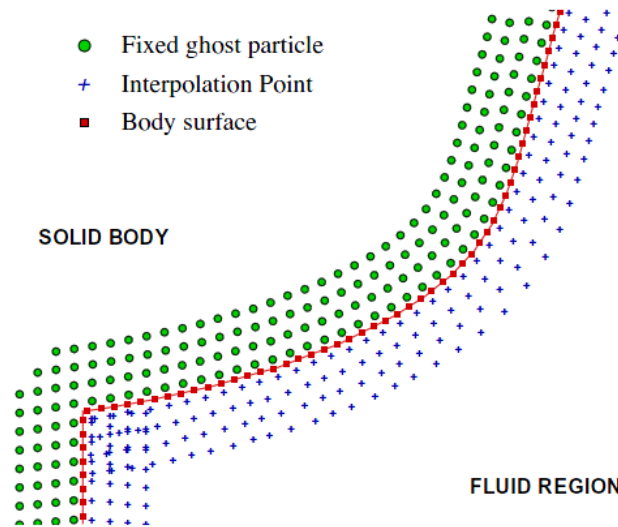


Figure 6: Fixed ghost particles and interpolation points along a surface. The physical properties of the ghosts will be calculated at their corresponding interpolation points which are placed within the fluid region. Image from Marrone et al. 2011

Other than in Landrini's original ghost particle method, the physical properties of the fixed ghost particles are not based on a paired SPH particle. Interpolation points within the SPH domain are used to calculate said properties instead. The positions of those interpolation points are the positions of the fixed ghost particles, mirrored by the solid boundary surface.

Free-slip and no-slip surfaces are both easily realized. For free-slip surfaces, the particle velocity calculated at the ghost's corresponding interpolation point is used as is. No-slip surfaces however are realized by reversing the tangential velocity component. In any case, the normal component of the velocity is reversed.

Both approaches have advantages and disadvantages. The normal ghost particle method is simpler and has no need for artificial interpolation points like the fixed ghost particle method. It potentially needs less ghost particles because they are only created where the fluid actually comes close to the boundary. This might be only a

small fraction of the actual surface of the solid object. Fixed ghost particles on the other hand have the advantage of adding less additional computational work since most of it is done only once, at the beginning of the simulation. It also avoids some potential issues that may arise in the normal ghost particle method when dealing with complex boundary shapes. Sharp edges for instance need to be handled carefully with normal ghost particles. Equation 3.32, which describes how to place ghost particles, might cause problems in such a case. Figure 7 gives an example case. Since the fixed ghost particles are always distributed evenly along the boundary such cases are easier to handle.

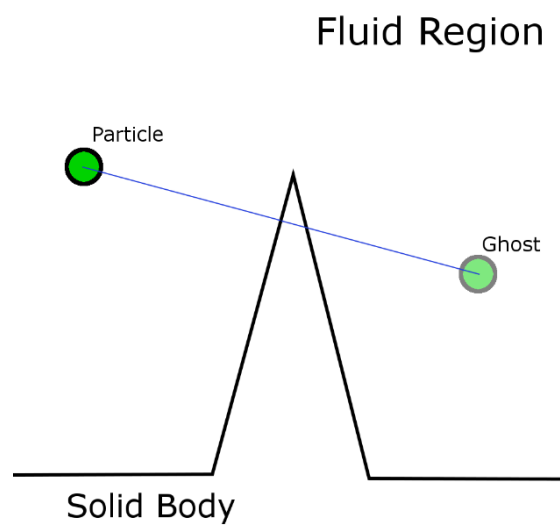


Figure 7: Misplaced ghost particle due to a sharp boundary edge. The ghost particle's position is the position of the original particle sunk into the solid body surface along the surface normal. In this case the ghost erroneously ends up above the surface because of the surface shape.

3.5. Open Boundary

The proposed use case of flight simulation demands for a vast terrain with typically multiple hundreds of square kilometers or more. Simulating airflow over the whole terrain using the SPH method appears unfeasible on current desktop computers. Consider using particles with a diameter of 100 meters. To cover an area with a side length of 10 km with a single layer of particles 10^6 of them would be needed.

To account for this problem, the airflow simulation is constrained to a relatively small area around a point of interest (POI). The method used to achieve this is based on the work of Federico et al. (2012).

To simulate open channels in 2D, they use two extra sets of particles, one upstream and one downstream of the actual SPH particles. Those two sets move with

the direction of flow, but do not simulate forces in between their particles. The upstream set is referred to as inflow particles, while the downstream set is referred to as outflow particles. Between the three sets, thresholds are defined, referred to as inflow and outflow threshold. Figure 8 shows those particle sets and thresholds. When a particle crosses the inflow threshold, between the inflow set and SPH set, the particle is removed from the inflow set and added to the SPH set. Consequently, this particle is now fully simulated according to SPH. Similarly once a particle crosses the outflow threshold, between the SPH and outflow set, it moves from the SPH to the outflow set. Thereafter it is no longer simulated as an SPH particle.

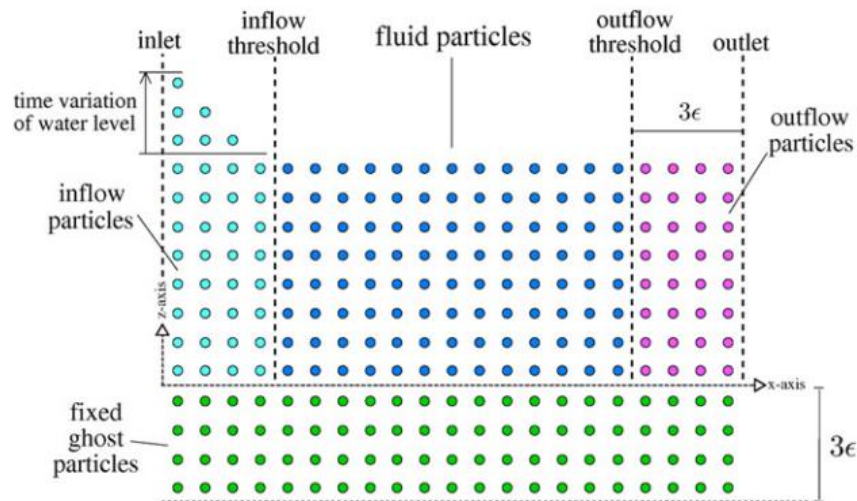


Figure 8: Inflow, SPH and outflow particles with fixed ghost particles underneath. Inflow particles move from left to right regardless of fluid forces. Right of the inflow threshold the SPH simulation and resulting forces takes over. Once an SPH particle crosses the outflow threshold it moves along according to its last assigned velocity by the SPH simulation. Image from Marrone et al. 2012

Upstream, the inflow particles are initially arranged on a regular grid. Velocity and pressure conditions are assigned to them and they move according to the assigned velocity. Once they cross over to the SPH set, their properties change according to the simulation. After crossing over to the outflow set the properties are frozen, SPH simulation ceases and the particles move along according to their last velocity, assigned in the SPH set.

The inflow and outflow sets are necessary to maintain correct hydrostatic pressure at the in- and outflow thresholds. So although the particles in those sets themselves are not simulated, they influence the particles within the SPH set. Without the two sets, the particles at the left and right boundaries of the SPH set would not experience any pressure from the fluid that would be present up- and downstream in a continuous channel. So the main reason for the in- and outflow sets is to get correct den-

sity and pressure values near the in- and outflow threshold of the SPH set. Therefore these two sets need to expand at least as far as the radius of the SPH kernel support radius from the respective thresholds up- and downstream.

4. Method

The main objective of this work is to simulate airflow over terrain. To achieve this, a setup consisting of a polygonal terrain created from a height map and a fluid simulation to simulate airflow over said terrain is used. The final goal is to derive wind forces acting upon objects in the simulation. The simulation is made with aircrafts as objects in mind.

To get the wind forces acting upon the aircraft, velocity vectors will be extracted from the fluid simulation. The boundaries of the fluid simulation are placed so that the aircraft is in the center of it. Force vectors can be interpolated at positions of interest along the aircraft's fuselage based on the fluid simulation's velocity vector field.

To simulate air, the smoothed particle hydrodynamics (SPH) algorithm is used. The fluid, air in this case, is represented as a set of particles. Those particles represent small blobs of fluid and are used as sample points to calculate the fluid's properties, e.g., density and pressure, at their positions. At each simulation step, the density at each particle is estimated based on the position of its neighboring particles within a predefined support radius. The influence of each neighbor is weighed by a kernel function. Based on density, the pressure at each particle can be calculated. The velocity of each particle results from the pressure gradient and possibly other forces, like viscosity and external forces, called body forces. Common examples for body forces are gravity or a current. For details on the algorithm see chapter 3.3. In the following, implementation details are presented.

The terrain for the use case of this work, desktop flight simulation, is potentially many hundreds of square kilometers large. To keep computation costs reasonable, the SPH domain is small in relation to the terrain. Consider using particles with a diameter of ten meters, to have a simulation resolution that is still reasonably high in relation to the size of a typical aircraft with a wingspan between 10 and 100 meters. It would take 10^8 particles to cover the ground of a 100 square kilometer area with just one layer of particles.

To address this, a scheme was developed to keep the SPH domain small in relation to the terrain. Without limiting boundaries the particles would dissipate over the terrain. So boundaries were added. The particles are required to be able to move through their domain freely, and if they reach the boundary, they need to be able to leave. At the same time new particles need to be added to keep the domain full. The

developed scheme satisfies those requirements. It is based on the work of Federico et al. (2012) (see chapter 4.3). As discussed earlier their work deals with open channels in 2D, so the approach had to be adapted to 3D. Instead of two individual additional sets of particles for in- and outflow, only one additional set is used that surrounds the SPH domain in all horizontal directions, not above or below though. Below containment is handled by terrain collision and above, the particles are free to move to account for the shape of the terrain below. The one extra set in this work is referred to as inflow set and it behaves mostly like the inflow set in the work of Federico et al. (2012). An important difference is that when leaving the SPH area, particles are again added to the inflow set since there is no special outflow set. Also different is that when particles leave the inflow domain they are not deleted. Instead they are moved to the opposite side of the inflow domain, and continue traveling downstream from there. For details see chapter 4.3.

Handling of solid boundaries, as it is done in this work, is discussed in chapter 4.2. Some limitations of this method are discussed in chapter 4.4. To overcome them, additional collision handling between terrain and fluid particles is added to the simulation. Details are described in chapter 4.4.

Based on the comparison to a reference ridge lift model, it was found that the proposed SPH-based method overestimates ridge lift forces. Data on this overestimation is given in chapter 4.5. To minimize the resulting error and improve realism of the SPH wind simulation a correction model was developed. The details of this model are given in chapter 4.5 as well.

In chapter 4.6 a method for increasing the numerical stability of the SPH simulation is explained. This method clamps the maximum speed of particles after the SPH forces were applied. Thus explosions of the fluid body are prevented. Besides increasing the simulation stability this addition to the algorithm can increase realism by choosing a meaningful speed limit. Short fluctuations of the average wind speed occur naturally and are called gusts. By setting the speed limit to a value based on natural gusts unrealistic speed fluctuations are prevented.

To increase the implementation's performance some simple measures were taken. Those are described in chapter 4.7. Achieving a real-time framerate is beyond the scope of this work. The goal is to achieve an interactive framerate for the validation process in chapter 5. Chapter 4.8 discusses how wind velocity at a given point within the SPH simulation domain is determined.

Implementation details of the visualization techniques used in this work are discussed in chapter 4.9. Two basic techniques are used, drawing particles directly as spheres and rendering the fluid body as layers of smoke. The first method will be used when it is desired to emphasize movement of individual particles, the second method to emphasize movement of the entire body of fluid.

In chapter 4.10 details of the validation techniques used in chapter 5 are discussed. Simple objects, called drones, are used to compare the SPH method to reference models. Each drone moves according to a given velocity. Additionally each drone is influenced by either the SPH simulated wind or one of the mentioned reference models.

Details on the architecture of the implementation of the SPH-based wind simulation are provided in chapter 4.11. The implementation is split into a number of parts called operations. The operations are executed sequentially and may apply changes to the set particles. A list and description of each implemented operation is provided.

4.1. SPH Implementation

The calculation of physical properties, like density and pressure, in the smoothed particle hydrodynamics algorithm is based on weighed influence of particles on one another. This influence has the form of forces, e.g. pressure and viscosity. To weigh the influence, kernel functions are used. Those kernels have a limited range, the support radius, beyond which the influence of particles is zero. All particles within the support radius of the kernel are considered neighbors. Finding the neighbors is the first action in each simulation time step. In this simulation it is done simply by a brute force search. Each particle's Euclidean (L2) distance to each other particle is calculated. For each particle, a list of indices is created. The indices of other particles that are within the kernel support radius are stored in that list. For all subsequent calculations on the individual particles, only the particles stored in the corresponding neighbor indices list have to be considered.

After determining the neighbors of each particle, the density, the pressure forces and viscosity forces within the fluid are calculated. Chapter 4.1.1 discusses the density calculation in detail. Chapter 4.1.2 deals with the pressure force calculation and chapter 4.1.3 with the viscosity force calculation. Finally, chapter 4.1.4 provides details on the handling of body forces and the enforcement of incompressibility of the fluid.

4.1.1. Density Calculation

After determining the neighbors of each particle, the next action per time step is to calculate the fluid's density at the positions of all particles. This involves the neighbor particles found previously. The implementation follows the method proposed by Solenthaler and Pajarola (2008). They modify the SPH algorithm, which was discussed in chapter 3.3, to account for particles with different properties such as mass or viscosity. The goal of Solenthaler and Pajarola (2008) was to allow for seamless mixing of different fluids. They show that using the standard SPH method creates gaps between fluids with different densities, as shown in figure 9.

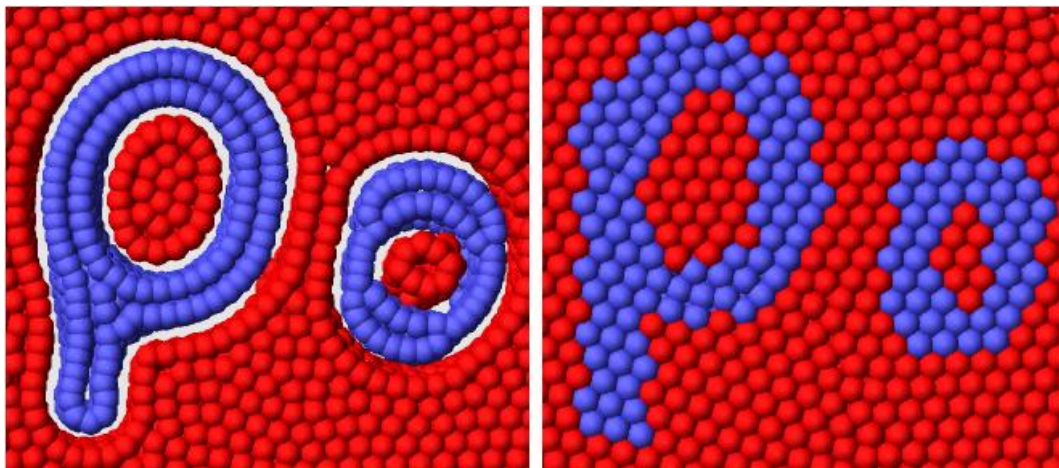


Figure 9: The images show two fluids with different densities. To the left is the result of the standard SPH algorithm, which creates gaps between fluids with different density. To the right is Solenthaler and Pajarola's method, which prevents those gaps. Image from Solenthaler and Pajarola 2008

Their method is used in this work to increase flexibility during development and allow for future extensions. The adapted equations for density calculation will now be discussed.

The densities are stored in an array with one entry per particle. Initially the self-influence of the particles is calculated. This is done using the kernel value at zero multiplied by the particle's mass. The resulting value is stored in the aforementioned density array. The influence of the neighbor particles is calculated in a similar manner. The kernel value at the neighbor's distance is multiplied by the neighbors mass and added to the density value stored in the density array. Like in Müller, Charypar and Gross (2003), the kernel used for density calculation is the Poly6 kernel.

In 2D, it has the form

$$W_{poly6}(r, h) = \frac{4}{\pi h^8} (h^2 - r^2)^3 \quad (4.1)$$

and in 3D the form

$$W_{poly6}(r, h) = \frac{315}{64\pi h^9} (h^2 - r^2)^3 \quad (4.2)$$

Figure 10 is a plot of the 2D poly 6 kernel.

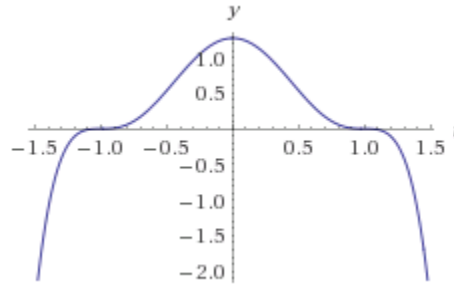


Figure 10: 2D poly6 kernel used for density estimation. The further away a particle is from the kernel's center the less it will affect the fluid's density at the kernel center position

4.1.2. Pressure Calculation

The pressure forces within the body of fluid are calculated based on the fluid's density at each particle. This is based on the densities as well as on the fluid's rest density. The rest density is the predefined density the fluid has when it is completely settled and still. Basically the fluid will try to achieve a state where the density equals the rest density at all particles. Where the density is lower the fluid has negative pressure, where it is higher it has positive pressure, over time those forces will move the particles into a state of equilibrium. Pressure is stored in form of force vectors in an array with one entry per particle. To calculate the force vector for a particle, again the neighbor particles as well as their corresponding densities are used. The neighbors are iterated. The first step of each iteration is to calculate the direction from the particle to the current neighbor. This will be the direction of the pressure force between the two particles. To finally calculate pressure magnitude at the particles, the Tait equation is used, which is

$$\tilde{p} = \frac{k\rho_0}{\gamma} \left(\left(\frac{\tilde{\rho}}{\rho_0} \right)^\gamma - 1 \right) \quad (4.3)$$

where ρ_0 is the fluid's rest density, $\tilde{\rho}$ is the particle's density and k is a gas constant. The factor γ is set to 7 according to Solenthaler and Pajarola (2008). $\tilde{\rho}$ is the number density, as used in their work.

It is calculated as

$$\tilde{\rho} = m\delta \quad (4.4)$$

with δ being the sum of accumulated density kernel values, as given by

$$\delta = \sum_N W(r - r_N, h) \quad (4.5)$$

A kernel function is used to weigh pressure according to distance between the particles. Using calculated pressure values and the kernel value, the pressure force between the particles is given by

$$f^{pressure} = -\sum_N \left(\frac{\tilde{p}}{\delta^2} + \frac{\tilde{p}_N}{\delta_N^2} \right) \nabla W(r - r_N, h) \quad (4.6)$$

A kernel different from the poly6 kernel, used in density calculation, comes to use to prevent the problem of clustering. As it was pointed out earlier by Müller, Charypar and Gross (2003), clustering means that particles that come very close to each other will cease to repel one another. They may get very close to each other, to the extent that they inhabit the same position in space. The reason for this becomes clear when considering that the pressure forces are based on the gradient of the fluid's density. The kernel's gradient is given by its first derivative. Looking at the poly6 kernel's first derivative, shown in figure 11, one can see that when approaching zero it slopes back to zero.

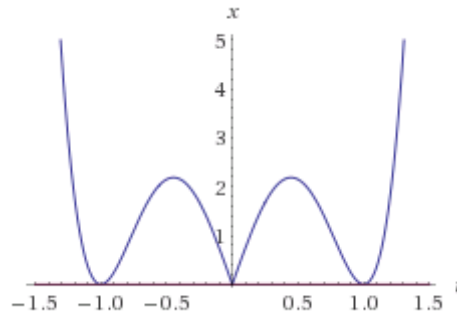


Figure 11: 2D poly 6 kernel gradient. The crevice in the middle causes pressure forces to decrease when particles get very close to one another

Pressure forces resulting from this kernel would decrease once the particles are closer than the curves inflection point. Like in Müller, Charypar and Gross (2003), the kernel used to account for this problem is called spiky kernel. Since it has a spike, it is not a smooth curve at $x = 0$.

The spiky kernel gradient for 2D is given by

$$\nabla W_{spiky}(r, h) = \frac{30}{\pi h^5} (h - r)^2 \quad (4.7)$$

and

$$\nabla W_{spiky}(r, h) = \frac{45}{\pi h^6} (h - r)^2 \quad (4.8)$$

for 3D. As figure 12 shows the gradient of the spike does not decrease towards $x = 0$, therefore its gradient never reverses.

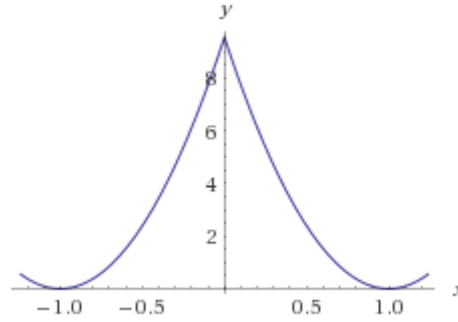


Figure 12: Gradient of 2D spiky kernel. This kernel is suitable to prevent particles from clumping up because it increases constantly towards the center, in other words towards 0 in x

The normalized force direction for the current particle pair multiplied by the pressure force magnitude is added to the particle's entry in the pressure force array.

4.1.3. Viscosity Calculation

Viscosity of a fluid describes how strong the fluid's tendency to resist deformation is. Water would be an example for a fluid with low viscosity, whereas honey has comparably high viscosity. In SPH viscosity is modeled as the tendency of particles to move along with their neighbors. To simulate this the particle's neighbors are again iterated. The velocity difference is calculated. The influence of the neighbor on the particle of interest is based on the two particles' densities and a kernel function to account for increasing distance. This results in

$$f^{viscosity} = \frac{1}{\delta} \sum_N \frac{\mu + \mu_N}{2} \frac{1}{\delta_N} (v_N - v) \nabla^2 W(r - r_N, h) \quad (4.9)$$

The calculated viscosity factor together with the velocity difference results in the viscosity force acting upon the two particles.

Again, not the poly6 kernel is used. Referring to equation 4.9 and the explanation for the use of the spiky kernel in the pressure calculation it becomes obvious why. This time the Laplacian of the kernel is used. As said in chapter 3.2, the Laplacian gives the difference of a quantity at a given point to the average of that quantity

around that point. In this case the difference of a given particle's velocity to the average velocity of particles around it is needed. Looking at the Laplacian of the poly6 and the spiky kernel, as displayed in figure 13 and 14, this would result in reversed forces.

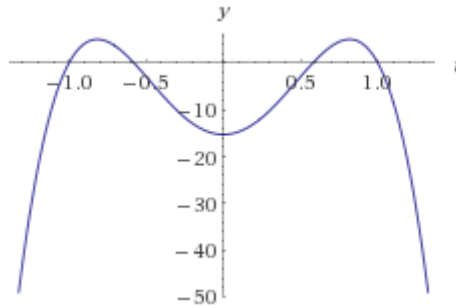


Figure 13: Laplacian of 2D poly6 kernel. The crevice in the middle would cause viscous forces between very close particles to decrease and even reverse

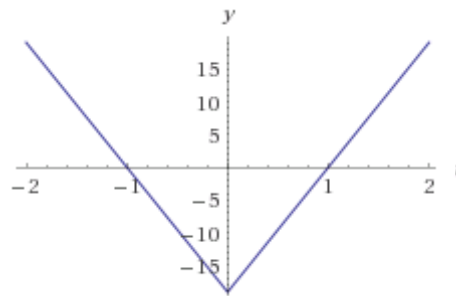


Figure 14: Laplacian of 2D spiky kernel. The crevice in the middle would cause viscous forces between very close particles to decrease and even reverse

Müller, Charypar and Gross (2003) propose an alternative kernel that is also used in this work. Its Laplacian form is

$$\nabla^2 W_{viscosity}(r, h) = \frac{20}{\pi h^5} (h - r) \quad (4.10)$$

in 2D or

$$\nabla^2 W_{viscosity}(r, h) = \frac{45}{\pi h^6} (h - r) \quad (4.11)$$

in 3D. Figure 15 plots the 2D version of the kernel. As can be seen in this plot, viscous forces increase correctly with decreasing distance.

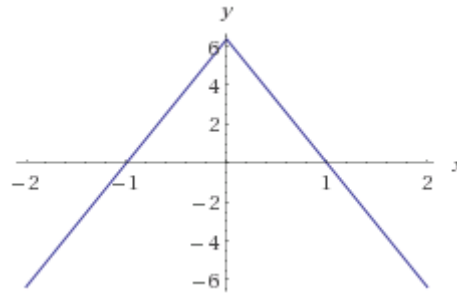


Figure 15: Laplacian of 2D viscosity kernel. Viscous forces calculated with this kernel increase steadily with increasing closeness

4.1.4. Body Forces and Incompressibility

After calculating the pressure and viscosity forces, the particles are ready to evolve accordingly. Some additional external forces are to be considered though. Those forces are commonly referred to as body forces. In this work those forces are gravity and a global wind force. Said global wind force is based on the large scale wind system. Said large scale system is not part of this work, a predefined force vector is used instead. After all those forces are added, the particles can be evolved accordingly. This simply involves modifying all particle positions by each particle velocity, derived from the acceleration resulting from all forces.

SPH may cause high compression in areas of low density. The particles will try to match the rest density, therefore they have to get close to each other. In extreme cases particles might collapse onto the same position. The upper image in figure 16 shows how particles near the bottom and the top get very close to each other to the extent that they overlap. To prevent this behavior this work uses a technique to manipulate the density and pressure calculation at close range. Said close range is the radius of the particles. Since each particle has defined mass and density, it has a resulting radius.

The kernel value at distances closer than the radius is changed by adding a second kernel during density and pressure calculation. Those kernels will be referred to as near density and near pressure kernel. These kernels are the same as the ones discussed above for density and pressure force but their support radius is the radius of the particle. The lower image of figure 16 shows the same scene as the upper image with this change applied.

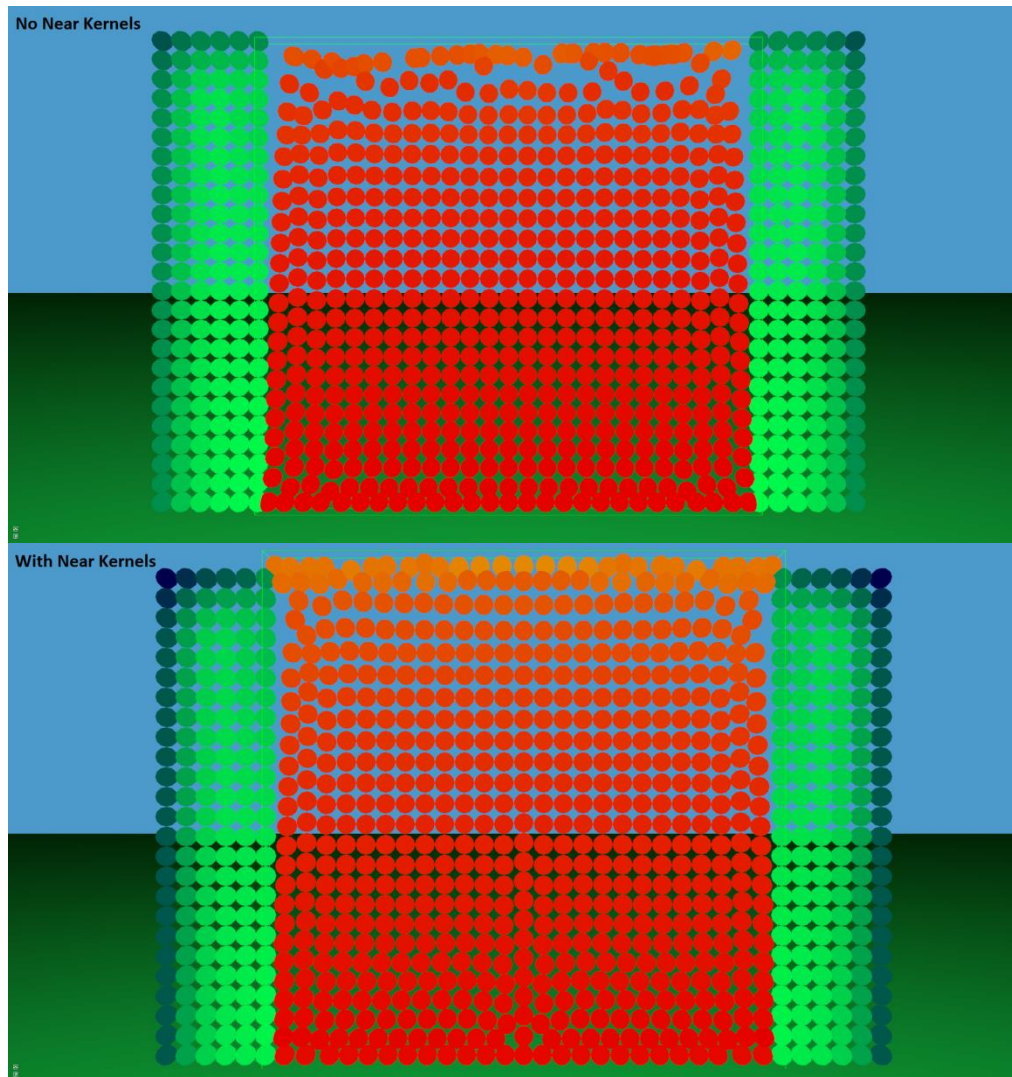


Figure 16: Particle distribution after 270 frames. Red particles are SPH particles, green particles are boundary particles. Color encodes density. The deeper the red or brighter the green the higher the density. The first image shows the simulation without near density kernel, the second image shows it with near density kernels. Particles in the second image are more evenly distributed and overlap is mostly avoided

The forces resulting from density and pressure calculation will increase much faster once particles penetrate each other. As figure 16 shows this prevents particles from bunching up and overlapping as described above. Therefore it helps maintaining the fluid's volume.

4.2. Solid Boundary Implementation

As discussed in chapter 3.4 solid boundaries need to be handled carefully. Simple mechanical collision between SPH particles and solid objects would cause erroneous density calculations near those objects.

This issue is accounted for based on the fixed ghost particles method as proposed by Marrone et al. (2011). For a detailed summary see chapter 3.4. Surfaces are modelled by placing fixed particles, the ghost particles, below the terrain surface. The ghost particles are placed with regular distances in layers along the surface. The first layer is just below the surface. More layers below are added. The lowest layer is as far below as the magnitude of the support kernel radius of the SPH simulation. This ensures that SPH particles touching the surface will still have a kernel full of neighbors below the surface, therefore will experience pressure from below. Other than in the ghost particle method by Marrone et al. (2011), the physical properties of the ghost particles are calculated at each ghost particle's own position. As discussed in chapter 3.4, Marrone et al. (2011) use points inside the fluid, so-called interpolation points, to calculate density and resulting pressure forces of the ghost particles. During testing of the used simplified ghost particle method, more even density conditions were achieved without the use of interpolation points.

For this work the ghost particle method was modified to account for potentially vast terrain sizes. To avoid having to create a very large number of ghost particles, only for the area below the inflow- and SPH area ghost particles are created. The area in which ghost particles are created reaches as far as the kernel support radius length below the terrain. Initially all particles are created along a regular grid. Upon updating the ghost particles are moved along with the inflow- and SPH particles. All the ghost particles maintain their relative altitude to the terrain directly above, with the top most layer of ghost particles being directly below the terrain surface. Other than that, like in the work of Marone et al. (2011), they do not move at all relative to the simulation domain. This way, like in the original fixed ghost particles method, additional computations after the initial setup are avoided. Some flexibility to change position is added. This flexibility is important because at creation time it is unknown what the shape of the surface for the ghost particles will be.

This method poses a problem when dealing with slopes. The ghost particle layer is not oriented on the boundary surface normal. Therefore gaps tangential to the boundary surface between ghost particles can form. This can result in an unfaithful

surface recreation, causing unwanted friction along the surface due to particles getting stuck in gaps. Since the ghost particles are created on an axis-aligned regular grid, those gaps are worst on 45° slopes. Figure 17 shows the ghost particles along a horizontal and a sloped plane, with visibly bigger gaps tangential to the plane underneath the sloped one.

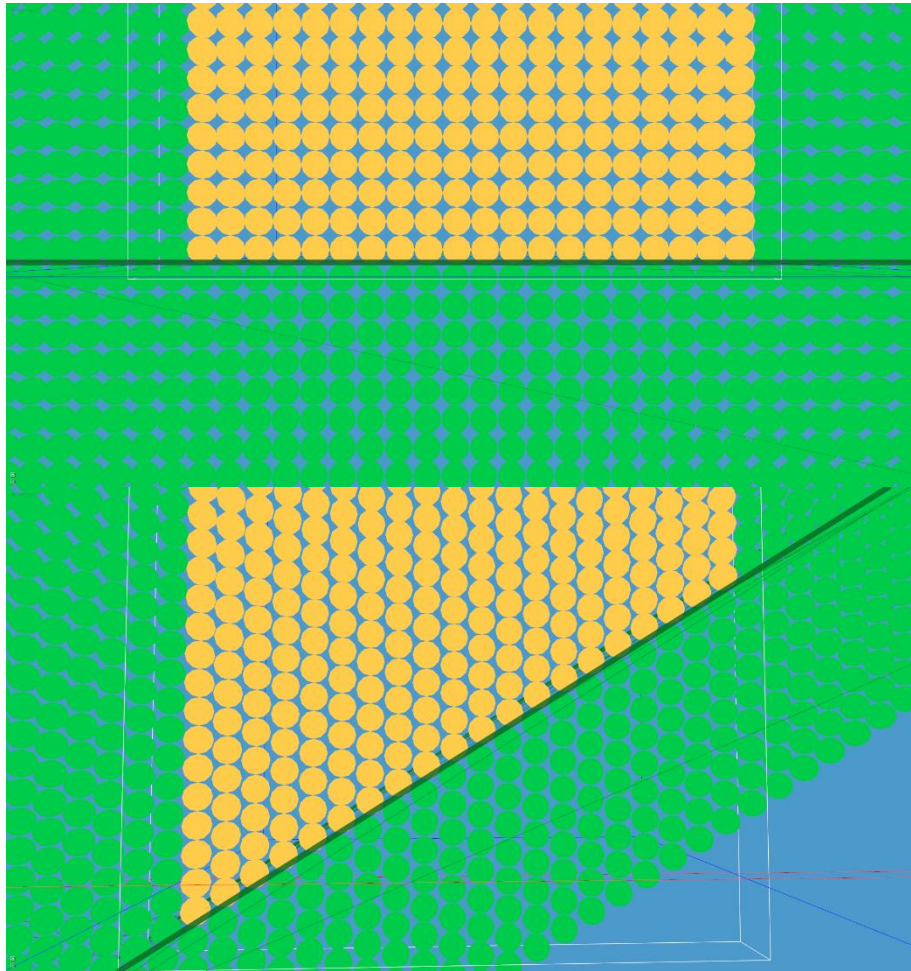


Figure 17: Distribution of ghost particles below the terrain surface (thick green line). The first image shows a horizontal plane. The second image shows a plane sloping upwards from left to right. While the ghost particles on the horizontal plane are closely spaced the ghost particles below the sloped surface have gaps in-between tangential to the surface

To account for this problem, an additional polygonal representation of the boundary surface is used. Simple rigid body collision detection and response between this surface and the SPH boundaries is applied to keep the SPH particles strictly above the surface. The ghost particles are directly below this surface, so the density and pressure calculations are influenced as little as possible by this addition. SPH particles are reliably prevented from moving into a ghost particle gap, therefore canceling out the unwanted friction.

4.3. Open Boundary Implementation

As mentioned earlier the airflow simulation used in this work is contained in a domain relatively small compared to the size of the terrain. This is done in respect to limited computation resources on current desktop PCs. The method used is based on the work of Federico et al. (2012) (see chapter 3.5). Their work is limited to 2D, so it had to be expanded to 3D to be used in this work.

The basic concept of having a spatially limited SPH particle set surrounded by particles that do themselves not follow SPH forces is maintained. Other than having an in- and outflow set though there is only one additional set, referred to as the inflow set. The inflow set fills an area around the SPH simulated area referred to as inflow area. As proposed by Federico et al. (2012), the physical properties in the inflow set are frozen and they move along a globally defined wind velocity. The movement of particles in the SPH set are governed by the SPH simulation. Also, just like proposed by Federico et al. (2012), once a particle crosses the threshold between the SPH- and the inflow area, the particle is assigned to the particle set belonging to the entered area. To avoid possible sudden movement due to changing pressure conditions when a particle enters the SPH area, the transition is gradual rather than sudden. Once the particle touches the SPH area, simulation starts, but simulated forces are only applied proportional to the part of the particle that is already within the SPH area. Therefore, only when the particle is completely within the SPH-area the calculated forces are fully responsible for its movement.

As discussed earlier (see chapter 3.5) the inflow particles are there to provide correct density and pressure conditions at the boundary of the SPH set, therefore the inflow set expands for the length of the SPH-kernel support radius in any horizontal direction from the respective side of the SPH set. Figure 18 displays the setup from above. Yellow particles are SPH particles while green particles are inflow particles. The kernel support radius in figure 18 is 10 times the particle diameter. Therefore the inflow set extends 10 particles from the SPH domain. Encoded in the green channel of the particle color is the calculated density. All SPH particles have the same color, therefore they also have the same density values. The inflow particles can move in any direction perpendicular to the ground, therefore enabling any horizontal wind direction. To allow the SPH particles to react to the terrain below, there are no inflow particles above.

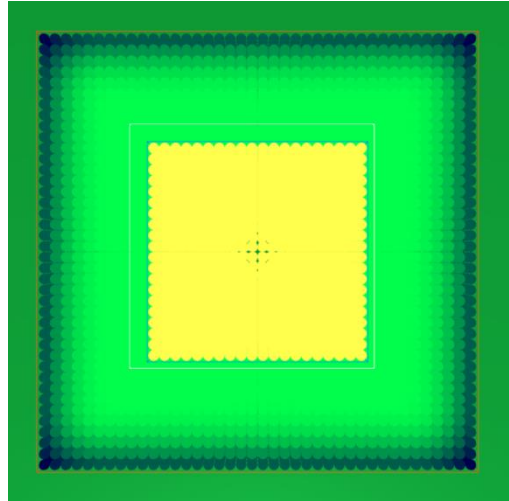


Figure 18: 3D Particle setup from above, yellow particles are SPH particles, green particles are inflow particles. The inflow particles horizontally surround the SPH particles. Color brightness encodes the fluid density at a given particle position. The inflow particles ensure horizontally even density throughout the SPH domain

Particles leaving the inflow area are handled by re-inserting them at the opposite side of the inflow area. In other words, if the particle leaves the inflow area in direction of the x-axis its x coordinate is set to the opposite end of the inflow area. The same is done in z direction. Figure 19 illustrates this in 2D.

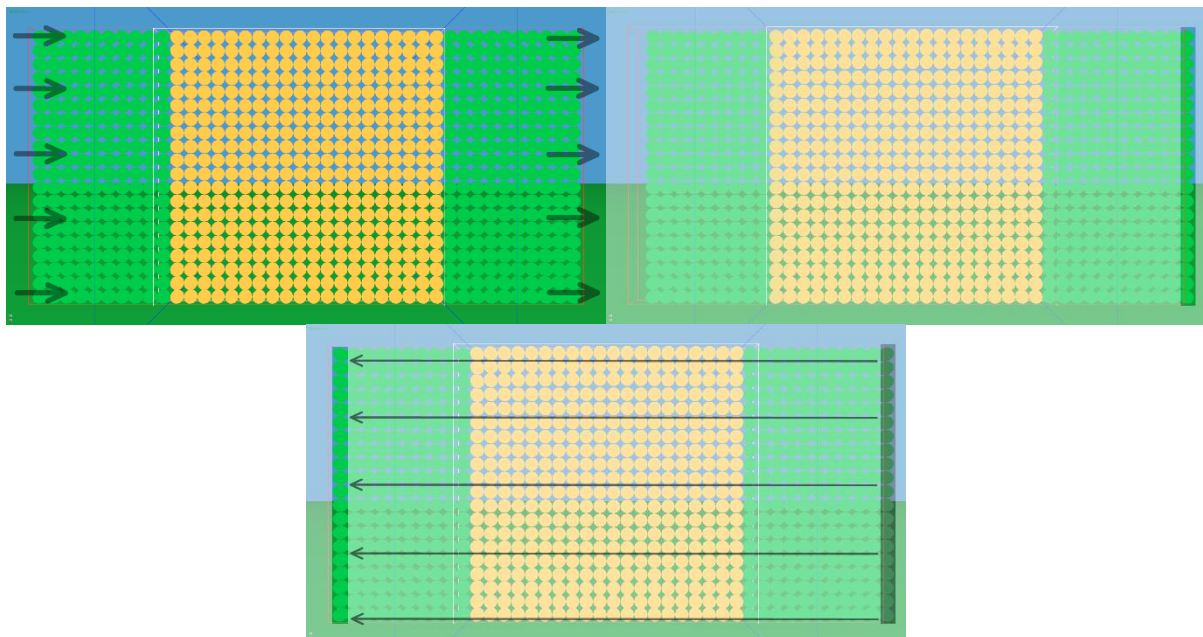


Figure 19: Particles leaving the inflow area downstream are teleported back upstream. Image one shows the initial setup, arrows indicate flow direction. Image two shows a set of particles leaving the inflow area. Image three shows where they end up after teleportation (old positions are greyed out)

All initial particles are conserved, so the overall mass of the simulation is conserved. Note that the number of particles within the SPH set may vary slightly.

Since the inflow particles are incorporated in the density and pressure calculation, the overall mass in the SPH simulation does not change.

The possibility of uneven ground had to be considered in this work, while Federico et al. (2012) assume even ground. Uneven terrain in the SPH area is handled by the rigid body interaction of the SPH simulation, inflow particles on the other hand do not interact with terrain on their own. A simple scheme is used to deal with this problem. Particles store their altitude above ground at initiation time as well as at each time step they spend within the SPH set. When moving as inflow particle, the particle's altitude above ground is constantly adjusted to this stored altitude. The same altitude is used to vertically reposition the particle once it leave, the inflow area and is teleported back upstream.

4.4. Rigid Body Collision

The basics of interaction between particles and solid bodies were discussed in chapter 4.2. As discussed in that chapter, rigid body collision detection and response is used to account for gaps in the formation of ghost particles below the solid body surface. This happens in three stages: recovery of triangle candidates, collision point calculation and collision response.

During the recovery of triangle candidates a list of triangles that might collide with a given particle is created. To find those triangles the particles are tested against the circumsphere of each triangle. The circumsphere closely surrounds the triangle, so only particles close to it will pass the test as illustrated in figure 20. The circumsphere test is considerably less computationally expensive than a precise test against a given triangle. However triangles that do not actually touch the particle may pass it. A precise test will be performed with each passing triangle, the triangle candidates, in the next stage.

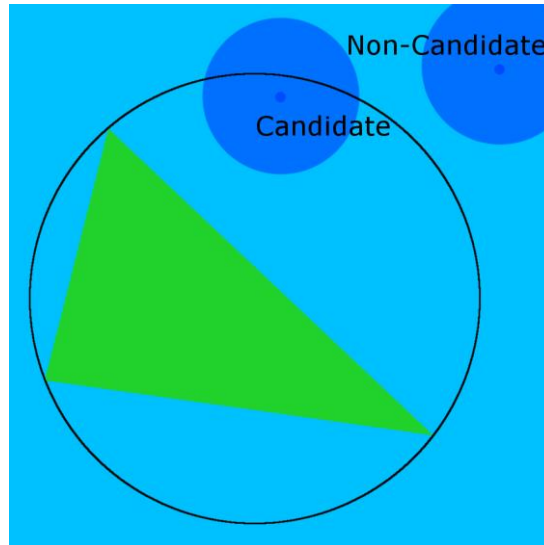


Figure 20: The circumsphere test conducted to find potential particle-triangle collisions. The left particle lies within the circumsphere, therefore the triangle is a valid candidate for a precise collision test. The right particle is outside the sphere and will not be considered during the precise collision test

In the collision point calculation step, the closest distance between a given particle and each triangle candidate from the previous stage is calculated. The triangle which will be the basis for collision response is the one with the least calculated distance. This is also true when more than one triangle is closer than particle radius. The point on the selected triangle closest to the particle will be referred to as contact point.

In the collision response stage, the particle is displaced according to the location and distance of the contact point and the particle's velocity is modified to account for the collision. If the distance of particle center to contact point is greater than the particle radius there was no actual collision. This step is omitted and the particle won't be modified. However if there was a collision, the particle is displaced so that it does not intersect the terrain surface anymore.

To resolve the terrain-particle intersection with the least possible particle displacement, the vector from particle center to contact point, P , will be reflected from the terrain surface. The reflected vector $P_{reflected}$ is scaled to the length of particle's diameter.

The required length is calculated as

$$length = \frac{r}{\cos \alpha} \quad (4.12)$$

where r is the particle radius and α is the angle between surface normal and P . Considering that $\cos \alpha$ is

$$\cos \alpha = \frac{|N|}{|P_{reflected}|} \quad (4.13)$$

it is the relation of the length of the surface normal N and $P_{reflected}$. When reshaping this equation to

$$|P_{reflected}| = \frac{|N|}{\cos \alpha} \quad (4.14)$$

it becomes clear that $P_{reflected}$ can be scaled by scaling N . By setting the length of N to r one ends up at equation 4.12 again.

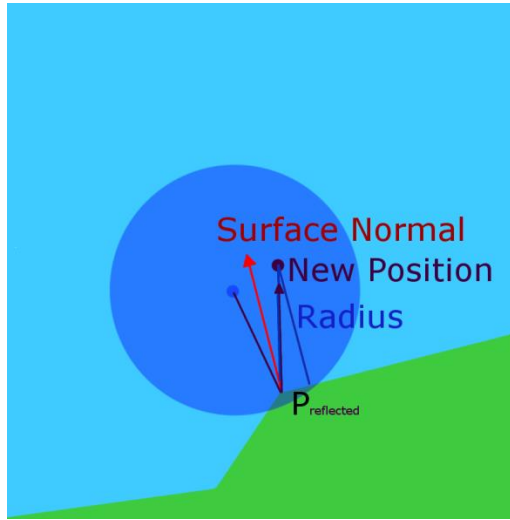


Figure 21: Particle displacement to resolve terrain intersection. The particle is displaced along the reflected line from center to contact point. The minimum distance of the new position to the surface is the particle radius

Furthermore the particle's velocity needs to be modified. Note that evolving the particle according to its velocity is not the responsibility of this part of the simulation. See chapter 4.11 for details. The naïve approach is to reflect a particle's velocity from the collision triangle surface, similar to how a rigid sphere deflects from a solid surface. However during evaluation this approach was found to be unsuitable for the specific application and setup of this work. This would result in a layer of air with strong upwind.

The SPH domain size used during tests in this work is 1 km^3 with a particle resolution of 16 by 16 by 16. Therefore each particle has a diameter of 62.5 meters. The lowest layer of air is therefore 62.5 meters high. As soon as an aircraft would get to

or below this altitude above ground it will experience this upwind, causing unnaturally strong lift forces at this point, as can be seen in figure 22.

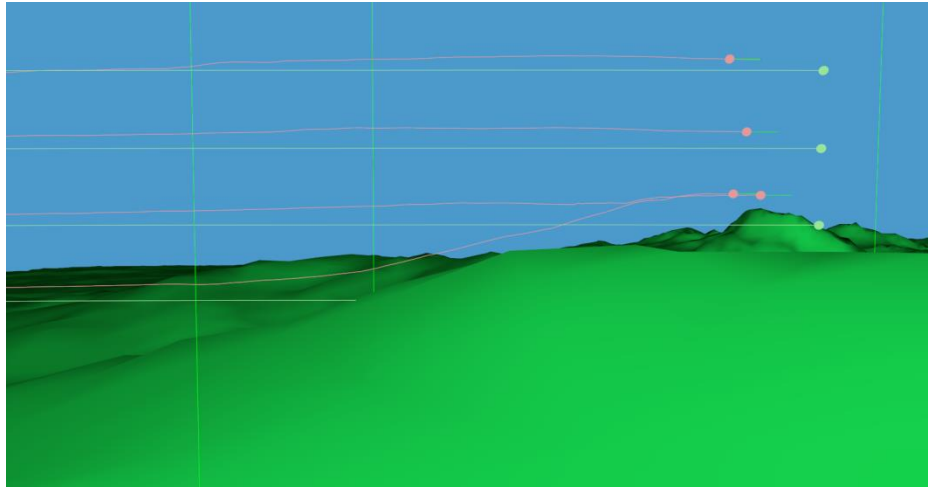


Figure 22: The result of simple particle velocity reflection upon terrain collision. The red spheres are drones influenced by SPH simulated wind. The green spheres mark the start altitude of the red drones. The lowest red drone visualizes the strongly exaggerated lift effect when particle velocity vectors are simply reflected from the terrain surface on collision.

Instead the velocity vector is redirected to run parallel to the collision triangle surface. Note that the horizontal velocity components remain unchanged so that the particle maintains its general movement direction. Fluid internal pressure forces during the next update step will subsequently sort out the exact appropriate velocity vector for collided particles. Note that this may not be an appropriate solution for any application of fluid simulation, in this work it has proven to yield satisfying results.

Additionally to the regular three-stage interaction, an additional test is used to catch particles that have already gotten completely below the terrain. This might happen due to tunneling. Tunneling refers to the effect when an object travels so far within one simulation step that it ends up behind an object on collision course without ever actually touching it. The test simply checks whether a given particle is below the terrain point with the same xz-coordinates. If this is the case the y component of its position is set to the terrain altitude plus the particle radius.

4.5. Lift Force Adjustment

The lift forces created by the SPH simulation were found to be overestimated. A correction model based on altitude above ground and wind speed was created to account for this.

The reason for the overestimation is unknown. It is suspected that the open boundary method, described in chapter 4.3, is the source for this issue. When the simulated wind hits a terrain slope, the lower layers of air will trace the slope. Higher layers will collide with those rising low layers. Due to the open boundaries, the low-layer particles cannot move out of the way of high-layer particles. To do so, they would have to accelerate and move faster than the global wind speed of the simulation. However, at the open boundary they are decelerated back to the global wind speed. Consequently, the low-layer particles cannot move fast enough to make room for high-layer particles and those high-layer particles have to go upwards. Whether this is the actual cause for the lift force overestimation or not was not verified. The developed correction model was introduced instead as a quick and efficient fix.

The correction model minimizes the average difference of the lift forces created by the SPH wind simulation and a given reference model. Said reference is the ridge lift model by Forster-Lewis (2007), in the following referred to as simple-lift model. This model has been implemented in FlightGear to improve the simulation of glider airplanes. Forster-Lewis' method was further investigated by Shah, Menezes and Kolmanovsky (2012). To validate it 10 flights were conducted over the Warner Springs region in California, USA. The paths were recorded to recreate the flights and compare real variometer readings with simulated up- and downwind. Shah, Menezes and Kolmanovsky state that judging by deviation from the recorded flight paths, Forster-Lewis' method achieves a 92.4 % accuracy.

Over a series of tests, lift forces produced by both the SPH method and the simple-lift model were recorded. Recordings were taken at different altitudes along a straight flight path over a terrain varying from even ground over hills to mountain slopes. Recording altitudes were chosen in 50 meter intervals, starting at 50 meters above ground and reaching up to 900 meters above ground, just below the ceiling of the SPH simulation domain. Wind speeds ranging from 3 m/s to 20 m/s were used.

For comparison the average lift forces per altitude above ground per wind speed was calculated as

$$\overline{lift_{alt\ wind}} = \frac{1}{N} \sum_{n=1}^N lift_{alt\ wind\ n} \quad (4.15)$$

where $lift_{alt\ wind}$ are the recorded lift forces for a given altitude above ground (alt) and wind speed ($wind$). Note that positive and negative lift forces were treated separately. The magnitudes of negative lift forces resulting from the SPH wind simulation

were found to be less on average than the positive lift forces of the SPH wind simulation.

Figure 23 shows positive $\overline{lift_{alt\ wind}}$ for the simple-lift model and the SPH wind simulation. The SPH method creates forces up to 30 times greater than the simple method.

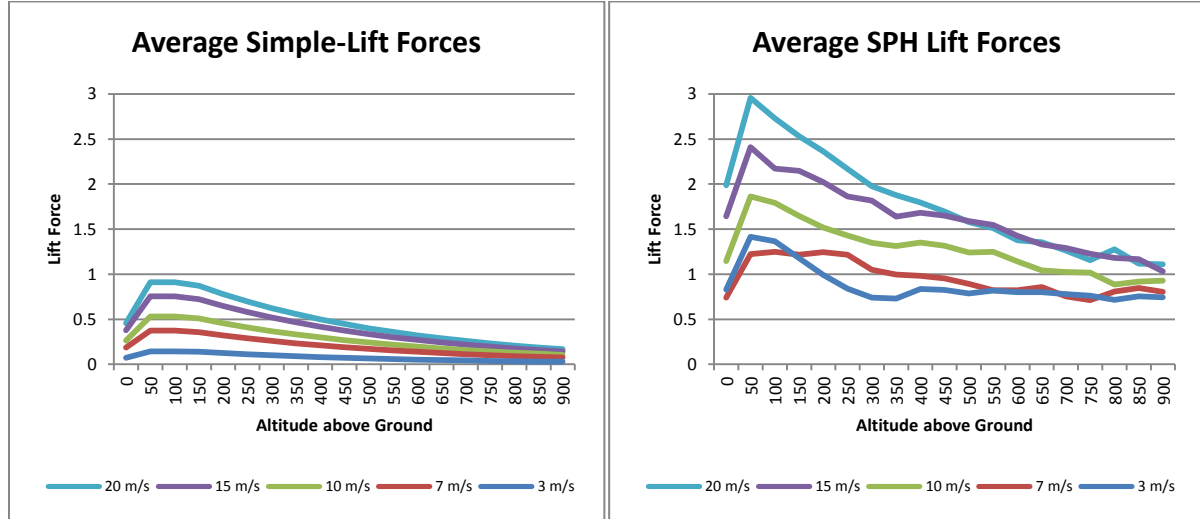


Figure 23: Average lift forces of the simple-lift model by Forster-Lewis (2007) and the SPH wind simulation for wind speeds from 3 m/s to 20 m/s. The SPH method creates significantly higher lift forces than the simple model.

The ratios of the averaged positive lift forces of the SPH method and the simple-lift model are calculated as

$$ratio_{alt\ wind} = \frac{\overline{sphLift_{alt\ wind}}}{\overline{simpleLift_{alt\ wind}}} \quad (4.16)$$

Those ratios are the basis to create a model that maps SPH lift to simple lift. For simplicity altitude above ground and wind speed are treated separately. Figure 24 plots the altitude-wise normalized ratios for all tested wind speeds.

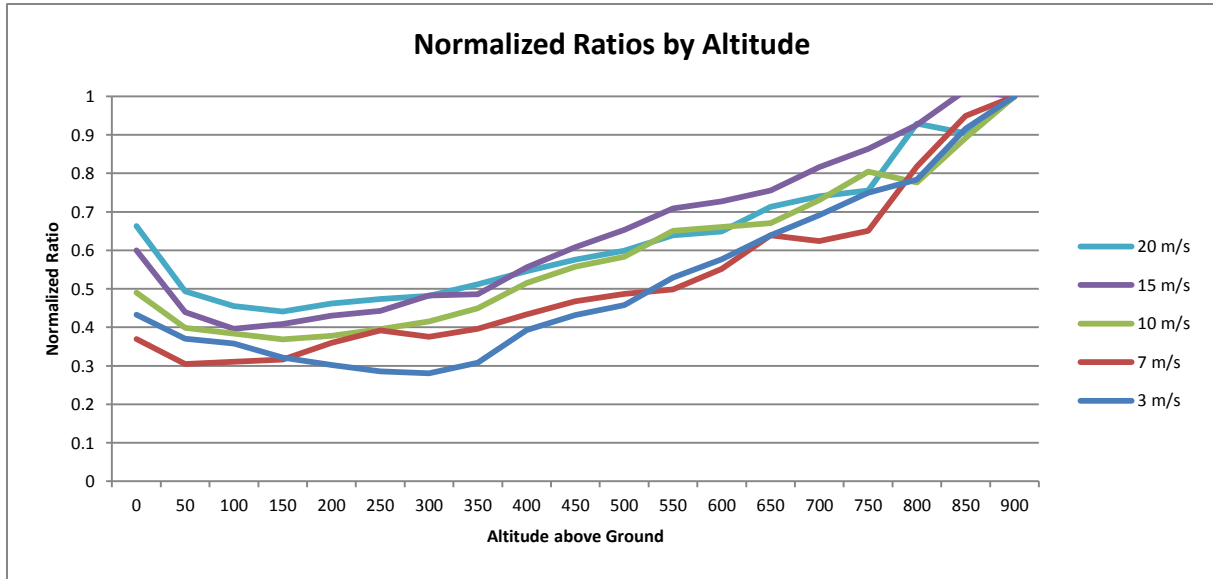


Figure 24: Altitude-wise normalized lift force ratios of the simple-lift model to the SPH method. Each line stands for a different wind speed from 3 m/s to 20 m/s. The closeness and similarity of the lines shows that they all can be approximated with a single function

The average correlation between the lines of figure 24 is 0.97, with a minimum correlation of 0.96. The maximum deviation between all lines amounts to 0.3. The closeness and similarity of the lines shows that they all can be approximated with a single function. The average of the lines in figure 24 is approximated by a quartic function. The function is then scaled by the greatest measured lift force ratio, the ratio at 3 m/s wind speed at 900 meters above ground. Therefore the quartic function

$$f_{alt\ positive}(a) = 12.246 - 4.341 \times 10^{-2}a + 1.896 \times 10^{-4}a^2 - 2.508 \times 10^{-7}a^3 + 1.262 \times 10^{-10}a^4 \quad (4.17)$$

approximates the lift force ratios for 3 m/s wind speed at any altitude for positive lift forces. a is the altitude above ground. The same process was repeated for negative lift forces, resulting in the quartic function

$$f_{alt\ negative}(a) = 12.766 - 5.343 \times 10^{-2}a + 1.451 \times 10^{-4}a^2 - 1.263 \times 10^{-7}a^3 + 5.227 \times 10^{-11}a^4 \quad (4.18)$$

Using equations 4.17 and 4.18 the lift force scale factor for wind speed 3 m/s is obtained by

$$f_{alt}(a, lift) = \begin{cases} f_{alt\ positive}(a), & lift < 0 \\ f_{alt\ negative}(a), & lift \geq 0 \end{cases} \quad (4.19)$$

To get a function that corrects the scale factor for varying wind speed the lift force ratios are normalized wind speed-wise. Figure 25 plots the resulting normalized ratios.

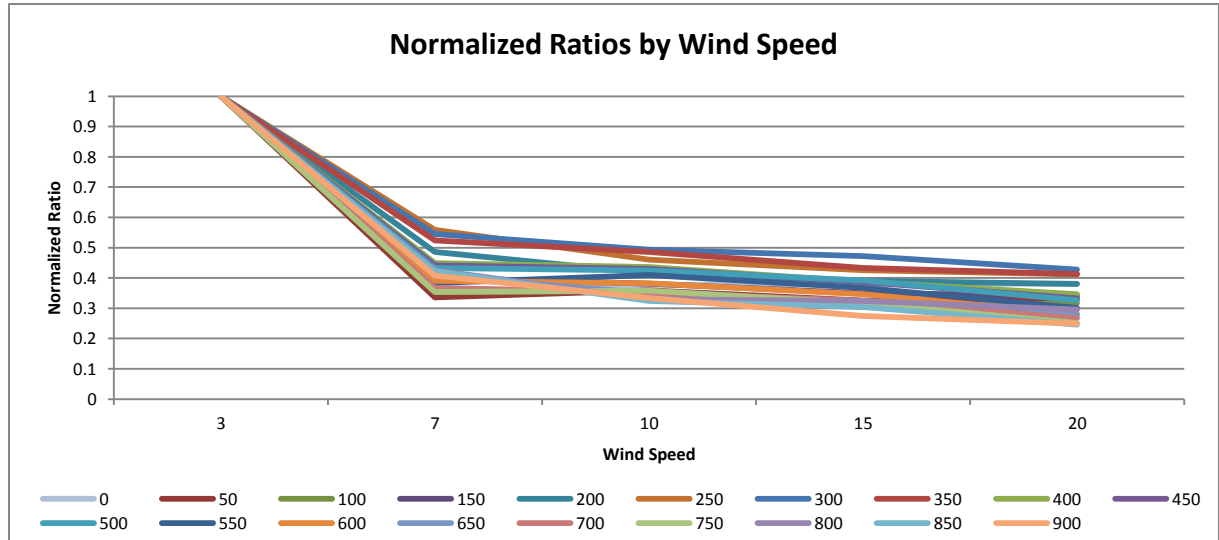


Figure 25: Wind speed-wise normalized lift force ratios of the simple-lift model to the SPH method. Each line represents a different altitude above ground from 0 to 900 meters in 50 meter increments. The similarity of the lines shows that they all can be approximated with a single function

The average correlation of the lines in figure 25 is 0.99 with a minimum correlation of 0.96. The maximum deviation of all lines is 0.18. The average of the lines is approximated by the function

$$f_{speed}(w) = \frac{1}{w} 2.0472 + .03032 \quad (4.20)$$

w is the wind speed. Note that the difference between the wind-speed wise normalized ratios was found to be negligible. No separate formula for negative lift was created. Figure 26 plots equations 4.17, 4.18 and 4.20, showing the similarity to the reference data in figure 24 and 25.

The scaling factor for a given altitude above ground and wind speed is calculated as

$$f_{scale}(a, w, lift) = f_{alt}(a, lift) f_{speed}(w) \quad (4.21)$$

The adjusted SPH lift force is then given by

$$lift_{adjusted} = \frac{1}{f_{normScale}(a, w, lift)} r lift_{base} \quad (4.22)$$

where $lift_{adjusted}$ is the adjusted lift force and $lift_{base}$ is the unadjusted lift force given by the SPH wind simulation.

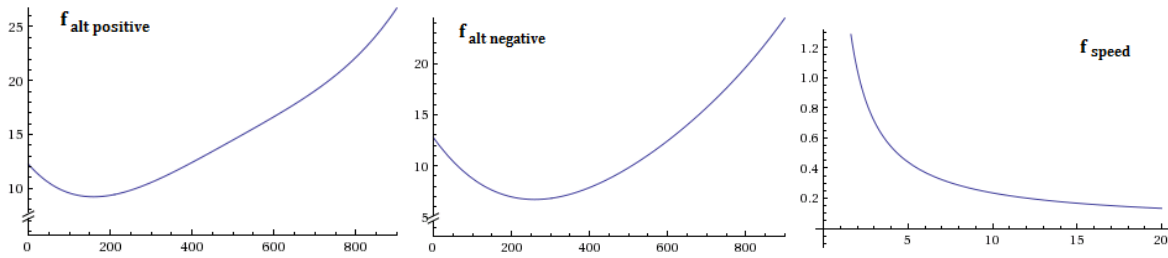


Figure 26: The lift force adjustment functions $f_{alt\ positive}(a)$, $f_{alt\ negative}(a)$ and $f_{speed}(w)$. Depending on whether the unadjusted lift force is positive (upward) or negative (downward) the final adjustment factor will be either $1/f_{alt\ positive}(a)f_{speed}(w)$ or $1/f_{alt\ negative}(a)f_{speed}(w)$. With increasing altitude and decreasing wind speed the SPH lift force needs to be scaled back stronger to match the reference data

4.6. Numerical Stability

Several aspects of the method presented in this paper can introduce instability to the SPH simulation. Inflow particles might end up penetrating each other within the inflow domain since no forces act on them to prevent it. This might happen when the slope angle below two close particles in the inflow area changes. Both of them maintain their altitude above ground, so they might end up penetrating each other.

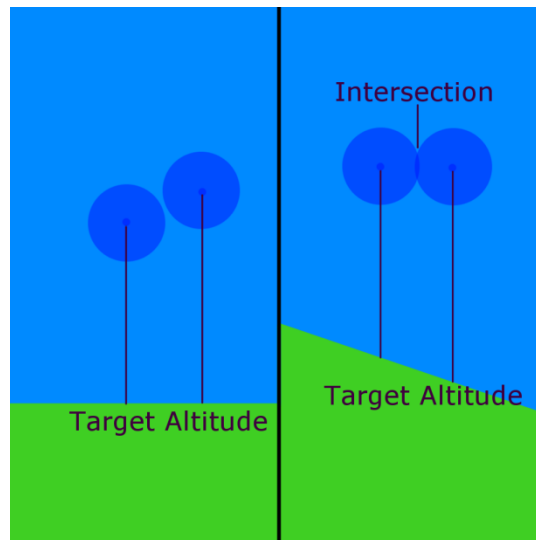


Figure 27: Inflow particle intersection due to change in terrain slope. Two inflow particles start out non-intersecting over even ground. When the slope changes they can end up intersecting because they strictly maintain their target altitude.

Once such a pair of particles reenters the SPH domain, the near density kernels will cause very high forces between those particles. This may in turn start a chain reaction and throw many particles out of the area of interest.

To prevent this from happening an artificial speed limit is introduced. The individual speed of each particle is limited to a multiple of the global wind speed. The speed limitation is applied smoothly. If the speed exceeds wind speed *overSpeed* is calculated as

$$\text{overSpeed} = \min(\text{overSpeedLimit}, \text{speed} - \text{globalWindSpeed}) \quad (4.23)$$

where *overSpeedLimit* is

$$\text{overSpeedLimit} = \text{speedLimit} - \text{globalWindSpeed} \quad (4.24)$$

overSpeed is then normalized using *overSpeedLimit*. A smooth step function is applied to the result x . The used smooth step function has the form

$$s(x) = 1 - (-2(1 - x)^2 + 3(1 - x)^3) \quad (4.25)$$

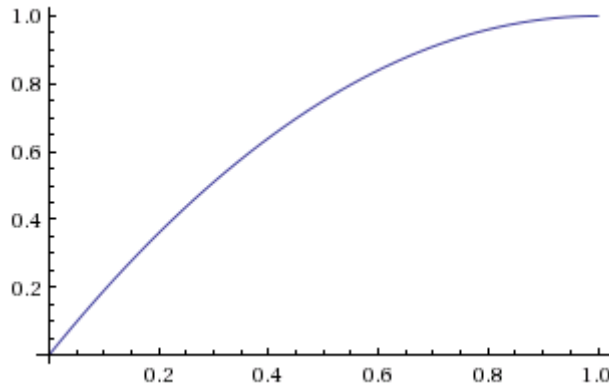


Figure 28: Smooth step function used to cap particle velocity. Particles faster than wind speed will be smoothly slowed down before reaching an absolute speed limit.

The particle's final velocity is given by

$$\text{velocity} = \text{norm}(\text{velocity}) \cdot (\text{globalWindSpeed} + s \cdot \text{overSpeedLimit}) \quad (4.26)$$

with s being the factor calculated using equation 4.25. Therefore the final speed is the wind speed plus the smoothed over-speed of the particle. Smoothing out the area between *globalWindSpeed* and *speedLimit* prevents sudden deceleration of fast particles, which may cause unnatural behavior of decelerated particles.

This measure turned out to not only prevent instability. It also prevents an issue when the body of fluid is flowing downhill. Particles entering the SPH domain uphill will accelerate quickly due to gravity. Particles leaving the SPH domain downhill will slow down to global wind speed again. This will cause particles to dam up downhill and quickly drain the SPH domain further uphill. A consequence of this would be unnatural downwind upstream and upwind downstream. The speed limit prevents this behavior though.

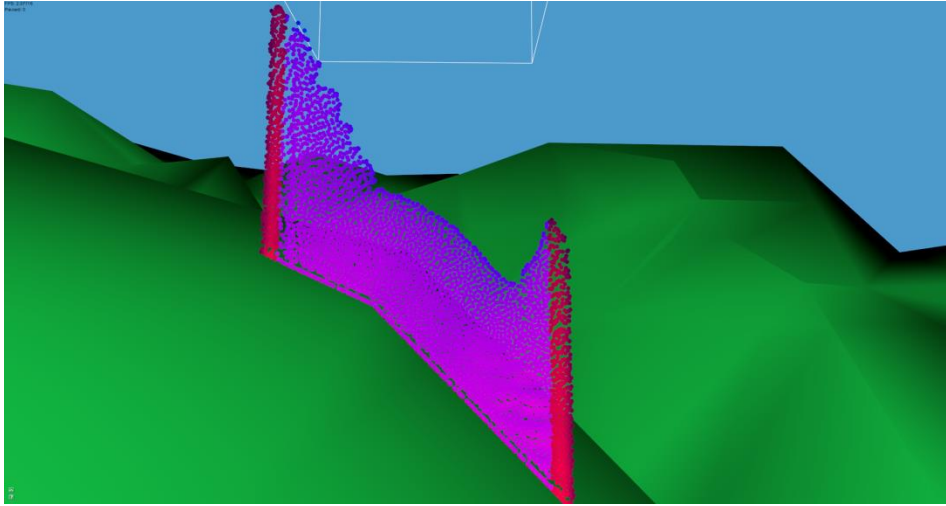


Figure 29: Particles flowing downhill unrestricted. Red particles are inflow, purple particles are SPH particles. The upstream SPH particles gain speed quickly and drain the upper region of the simulation domain. Downhill they are slowed down quickly when hitting the downstream inflow particles.

The choice of speed limit is based on naturally occurring gusts. Gusts are fluctuations in wind speed. Gusts appear suddenly and last for only a few seconds. According to U.S. weather observation practice, they usually last for less than 20 seconds (National Oceanic and Atmospheric Administration, n.d.). They are described in terms of ratio of wind speed during a gust to the 10 minute mean wind speed. This ratio can vary from 1.2 to 1.6 near ground for weak winds to above 2.0 during storms (Ágústsson and Ólafsson 2009).

4.7. Performance Optimization

The complexity of most parts of the simulation depends on the number of particles. For example the calculation of SPH forces needs to determine the closest neighbor particles of each particle at every time step. To reduce the complexity of this operation, a regular-grid container is used. Those containers consist of a fixed number of evenly sized cells. Objects are inserted into cells based on position. Inserting an object has constant complexity $O(1)$. So has retrieving objects contained in a cell.

Consider the nearest neighbor search of the SPH force calculation. Naive implementation yields a complexity of $O(N^2)$, with N being the number of particles in the simulation. Using a 3D regular grid container can reduce this complexity to $O(N)$. The size of the grid cells is set to the influence radius of the SPH particles. Retrieving all possible particles that might be within the influence radius of a particle of interest (POI) requires retrieving the particles from the cell at POI position and the

surrounding 26 cells. If the influence radius is small compared to the SPH domain size, the average number of particles to test in the nearest neighbor search will be greatly reduced.

A 2D regular grid comes to use in terrain sampling. In the naive implementation the complexity would be $O(M)$, with M being the number of triangles in the terrain mesh, for each time a particle samples the terrain. This can be reduced significantly by using a regular grid container. The terrain is generated using the marching cubes algorithm introduced by Lorensen and Cline (1987). The algorithm extracts a triangle surface from input data in form of a trilinear function. To do so it splits space into regular sized cubes. Vertices are created at the intersection points of the surface defined by the input data and the edges of cubes.

Cell size for the regular grid is set to the cube size used during terrain creation. Therefore each triangle of the terrain is within exactly one cell and does not overlap cell boundaries. The complexity of one terrain sampling operation is therefore reduced to a maximum of $O(m)$, where m is no greater than 12, the maximum number of triangles per cell. Note that this number depends on the concrete implementation of the marching cube algorithm, different versions exist. The total maximum of triangles to check is no more than $9m$. This results from the fact that the cell at POI position and its 8 neighbors are used. This prevents missing the closest triangle if the POI overlaps cell boundaries. As a consequence the terrain mesh resolution has no impact on the complexity of terrain sampling anymore.

4.8. Wind Force Recovery

A simulated aircraft needs the wind velocity at a given point to calculate its true air speed. Specifically the velocity needs to be recovered at one or more locations along an airplane's fuselage, wings and control surfaces. Those locations will be referred to as probes.

When choosing the exact locations for probes, the SPH particle resolution and airplane size are to be considered. If the particle size exceeds the wingspan of the airplane, one probe is likely enough. If the particle size is equal or smaller than the wingspan, using more than one probe can yield useful additional information. Velocities at those probes may differ from one another, resulting in differential forces along the airplane's fuselage and wing surfaces. Note that the length of most airplanes is

equal to or smaller than the wingspan. Therefore the same considerations regarding probe count and wingspan hold true for fuselage length.

To make retrieval efficient a regular grid container is used. The grid is filled with the SPH particles. The complexity of inserting a particle is linear. So is the complexity of retrieving the particles of one grid cell. The size of the grid is given by the bounding box around all SPH particles. The grid cell size is determined by particle size or distance between probes, whichever results in the bigger cells.

To get wind velocity at one of the probes, the particles of the grid cell in which the probe is are retrieved. The average of the velocities of those particles is taken as wind velocity at the probe position.

4.9. Visualization Techniques

Air on a small scale is generally not visible. During the day we see that it is there because of light scattering (Rayleigh scattering). Movement of air is not directly visible though. We can however perceive it visually due to other effects caused by it. Some examples are trees moving in the wind, smoke, dust and other particles being blown away or flags waving. In wind tunnels specialized visualization methods are used including wool tufts and smoke streaks.

For the tests used in this work, different visualization techniques are used to provide insight into different aspects of the simulation. The simplest one is rendering SPH particles directly. This is useful for step by step analysis of particle behavior. Since SPH is based on particles with a position and a radius to begin with, it is simple to render them as spheres. Properties like density are encoded in the particle color as needed.

For tests with emphasis on movement of the entire fluid body, a smoke surface visualization technique was implemented. The visualization is based on the paper of von Funck et al. (2008). Smoke is represented as a triangle mesh. The approach is optimized for real-time. As a consequence, mesh topology is fixed to avoid re-triangulation. Multiple planes of smoke parallel to the xz plane are used to emphasize different movement of altitude layers.

Vertices of the smoke mesh are spawned at so-called seed locations, or simply seeds. Those seeds are placed downstream. The vertices are released into the stream in form of particles. Figure 30 shows a smoke surface with particles as white spheres and a wireframe surface mesh. They are released in a row along the intend-

ed plane surface. Every released particle will then be advected with the simulated flow at each time step. Von Funck et al. release a new row of particles into the stream after a fixed number of time steps. This causes triangles created in slow flow to be smaller. Triangle size has influence on the assumed smoke density within the triangle. Details on this will be discussed shortly. In this work new particles are released once the previous row has reached a given distance from the seed. At the initial time step a row is emitted immediately.

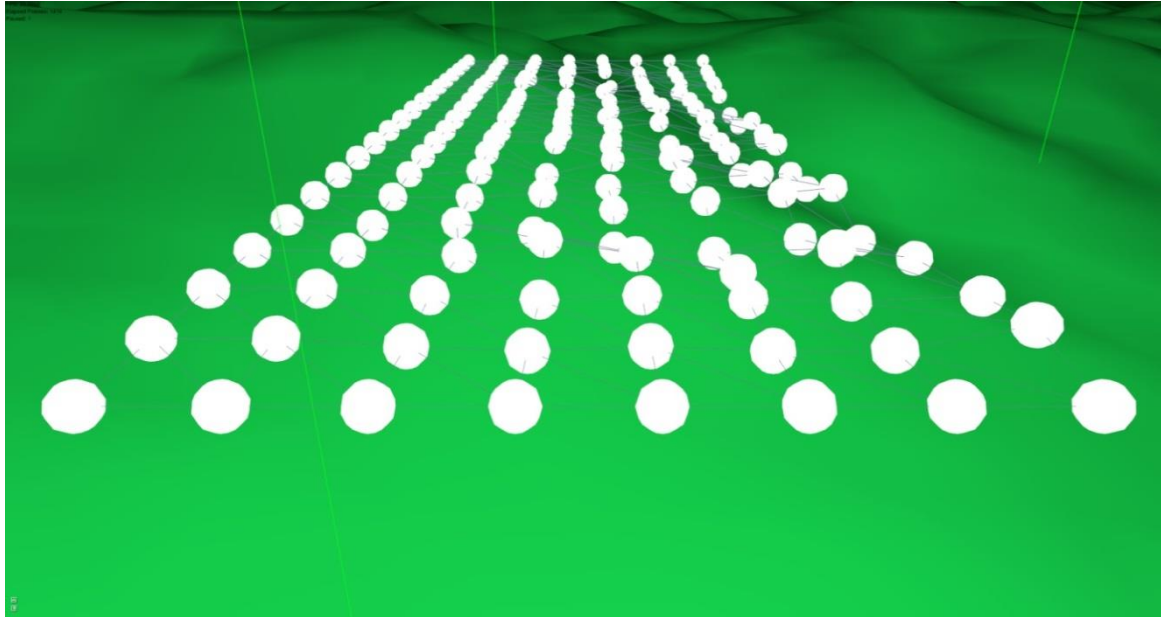


Figure 30: Particles and mesh wireframe of a smoke surface. Wind flows from left to right, carrying the smoke particles with it and therefore deforming the mesh.

Since the seeds are placed at the outer edge of the inflow domain, the flow speed can be assumed to be constant, no variation in triangle size is to be expected. Using a fixed distance instead makes calibration of the visualization algorithm easier, a desired triangle size can be chosen directly.

Rendering the smoke surface requires knowing how dense the smoke is at a given point. Von Funck et al. use an approximation based on triangle shape and size. Each triangle is assumed to be a flat prism, evenly filled with smoke particles. The smaller a triangle gets the higher the density of smoke particles within it becomes. The more dense the smoke is the less transparent it becomes, therefore its alpha value increases. The alpha value from this calculation is referred to as $\alpha_{density}$.

Von Funck et al. also account for triangle deformation. It might happen that one particle of the triangle drifts away, e.g. when the stream splits at an obstacle and the runaway particle ends up at the opposite side than the others. In this particular case

the affected triangle should not be visible at all. This issue is addressed by introducing a measure for shape quality. The ratio of shortest to longest edge is used. When the triangle edges are of equal length, the quality is high and the resulting shape factor is 1. If the triangle has a very short edge compared to the others, the factor will be very small. Therefore triangles with low shape quality can be rendered much more transparent than high quality triangles. The alpha factor from this calculation is referred to as α_{shape} .

The final opaqueness of a smoke triangle depends on how many of the imaginary smoke particles have to be passed by a ray of light when traveling through it. Therefore the angle of the view ray to the triangle surface is considered. The more shallow the angle the more opaque the triangle will be rendered. So the final alpha value is

$$\alpha_{final} = \alpha_{density} k \alpha_{shape}(E \cdot N) \quad (4.27)$$

where E and N are the direction of the camera to the current point of interest and the surface normal at this point respectively. The final result can be seen in figure 31. k is a factor controlling the influence of $\alpha_{density}$ on the final result. Since $\alpha_{density}$ is derived from triangle size k is based on the average triangle size of the mesh. In this work

$$k = 0.1 A_{average} \quad (4.28)$$

is used, where $A_{average}$ is the average triangle area of the smoke mesh. Note that this might not be suitable in all cases, it strongly depends on the deviation that triangles might have from the average triangle area.

Note that von Funck et al. write about calculating values, e.g. $\alpha_{density}$ and α_{shape} , per triangle. To use those in a shader program values are needed per vertex though. To address this a value's average over all adjacent triangles is stored per vertex and passed to the shader program.

This is a simplified implementation of the original technique proposed by von Funck et al. They use additional factors to calculate α_{final} . For one they use α_{fade} which is time dependent and will fade out triangles as their lifetime approaches a given limit. In this work triangles are just deleted as they leave simulation domain. It is not desired for the smoke to disperse before that. They also use a factor called $\alpha_{curvature}$ to address misrepresentation of flow regions with high curvature. If the mesh resolution is too low it might not be able to adequately represent such regions. In such cases $\alpha_{curvature}$ approaches 0 to hide the region. In this work the mesh resolution is set to the resolution of the simulation, therefore this factor is omitted.

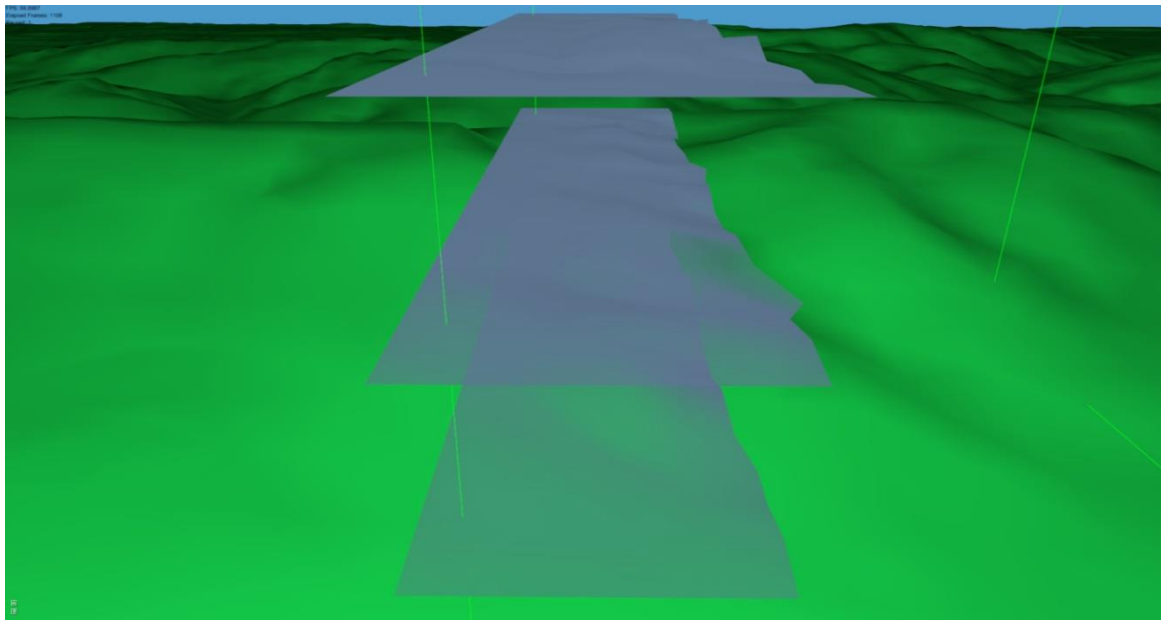


Figure 31: 3 layers of smoke. The influence of view angle on surface transparency is easily visible when comparing the top and the bottom layers.

4.10. Validation Techniques

As a simple validation method, simple flying objects, referred to as drones, were implemented. They serve as stand-ins for aircrafts. The drones have no other properties but a position and a velocity, according to which they move. Depending on the specific purpose of a given drone type, it is also influenced by some additional force.

The first type of drone is influenced by wind velocity retrieved from the SPH wind simulation. How wind velocity is retrieved is described in chapter 4.8. Movement of this drone, referred to as SPH drone, is described by

$$newPosition = oldPosition + (velocity + windVelocity)deltaTime \quad (4.29)$$

To get a baseline to compare the ridge lift effect in the SPH method to, another drone using Forster-Lewis' (2007) lift model was implemented. It will be referred to as simple-lift model. As mentioned in chapter 4.5 this model has been implemented in flight simulators to provide scenarios for glider aircraft. Shah, Menezes and Kolmanovsky (2012) compared it to variometer data recorded in real-life flights. They concluded that the simple-lift model achieves a 92.4 % accuracy.

In Forster-Lewis' method the terrain's altitude is sampled at five points on an axis along the global wind direction hinging on the aircraft's position. The first point is directly below the aircraft. The second, third and fourth points are placed at distances

250, 750 and 2000 meters into the wind, the fifth point lies in the opposite direction 100 meters from the airplane. The slopes are calculated between neighboring points. Those slopes are weighed for distance according to

$$\begin{aligned} slope_0 &= \frac{samplePoint_0 - samplePoint_1}{distance_1} \\ slope_1 &= \frac{samplePoint_1 - samplePoint_2}{distance_2} \\ slope_2 &= \frac{samplePoint_2 - samplePoint_3}{distance_3} \\ slope_3 &= \frac{samplePoint_4 - samplePoint_0}{distance_4} \end{aligned} \quad (4.30)$$

The slopes mapped from a range of negative to positive infinite to plus to minus one. The formula is

$$atanSlope_i = \sin(\text{atan}(5 * (|slope_i|^{1.7}))) * \text{sign}(slope_i) \quad (4.31)$$

which, as figure 32 shows, will exaggerate shallow slopes and attenuate steep slopes.

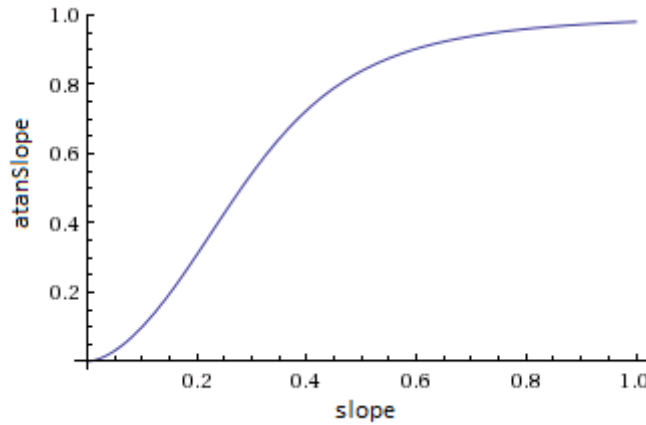


Figure 32: Plot of the normalization function for slopes. Slopes are mapped from a range of minus to plus infinite to a range of minus to plus one. Shallow slopes are exaggerated, steep slopes are attenuated. The normalized slope values are the basis for lift force calculation in Forster Lewis' (2007) ridge lift model

To get from slopes to the lift factor at the airplane position the slopes are modified to account for the individual influence of each one on the final lift factor. Slope 1 has the most influence, its lift factor is simply

$$liftFactor_1 = -atanSlope_1 \quad (4.32)$$

The other slopes lift factor's end up with less weight

$$\begin{aligned} liftFactor_0 &= (atanSlope_1 - atanSlope_0) | liftFactor_1 | \\ liftFactor_2 &= \frac{-atanSlope_2}{1.5} \\ liftFactor_3 &= \frac{atanSlope_3}{4} \end{aligned} \quad (4.33)$$

The sum of the four lift factors multiplied by the wind speed provides the base lift for the airplane position. The last step is to factor in altitude above ground. Generally the higher the aircraft is, the less terrain lift will be experienced. Forster-Lewis defines three altitude ranges to account for the influence of ground friction and lift attenuation. The first range is from ground level to 40 meters above ground. Lift is reduced by the *aglFactor* (*agl* stands for 'above ground level'), which is

$$aglFactor = \exp\left(-2 \frac{altitude}{groundLevel}\right) \quad (4.34)$$

with *altitude* being the aircraft altitude above sea and *groundLevel* being the altitude above sea level of the terrain below the aircraft. This results in a value from 0.5 to 1. Close to the ground friction and turbulence caused by obstacles reduces lift, the higher above ground the less reduction will be experienced. The second altitude range is from 40 meters to 130 meters above ground. In this range the full base lift is applied. The last range starts at 130 meters. The base lift is reduced exponentially with increasing altitude. At 130 meters it is still 1 and goes to 0 at higher altitudes. The decay of lift is calculated as

$$aglFactor = \exp\left(-1 \left(2 + 2 \frac{groundLevel}{4000}\right) \frac{(altitude-130)}{\max(groundLevel, 200)}\right) \quad (4.35)$$

Lift above low ridges will decay faster than lift above high ridges. The idea is that wind deflecting off a high slope, e.g. of a mountain, will rise higher than wind blowing over the side a small hill. As Forster-Lewis points out this will create a good approximation unless in unfavorable conditions. Gentle slopes at high altitude, e.g. on a high plateau will result in the same lift decay as steep slopes peaking at the same altitude.

4.11. Software Architecture

The architecture of the SPH method's implementation, as seen in figure 33, is designed for modularity and quick alterations. The core is the particle set, including the SPH, inflow and ghost particles as they are described in chapters 4.1, 4.2 and 4.3 respectively. The algorithm itself is composed of distinct operations that manipulate the particle set. Details on both the particles and the operations are provided below.

Particles are implemented as objects with a number of given properties from which more properties important to the simulation are derived. The initially given properties are

- Position
- Radius
- Density
- Viscosity

Position and radius simply describe where the particle is and how big it is. The particle's density depends on the fluid that is to be simulated. It is needed to calculate the forces between particles as described in chapter 4.1. Viscosity is also a property based on the type of fluid. By storing those two values per particle the algorithms flexibility is improved, different types of fluids may be simulated at once.

Additional particle properties are either derived from the four initial properties or calculated during the simulation. Those properties are

- Mass
- Force
- Velocity
- RelativeAltitude
- SimulationFactor

Mass is derived from density and radius. Depending on whether the simulation is running in 2D or 3D it is based on circle area or sphere volume. Force is calculated during the simulation and is the basis to calculate the particle's velocity. RelativeAltitude stores the particle's altitude above ground. It is set and updated constantly when the particle is within the SPH domain. When the particle is within the inflow domain it maintains this altitude. The simulationFactor stores whether the particle is in the SPH or inflow domain.

Particles are stored in a container that holds additional information on the fluid body. The left-lower-hind corner and right-upper-front corner of the SPH domain as well as the bounding box of all non-ghost particles are stored as well as the number of SPH particles per dimension.

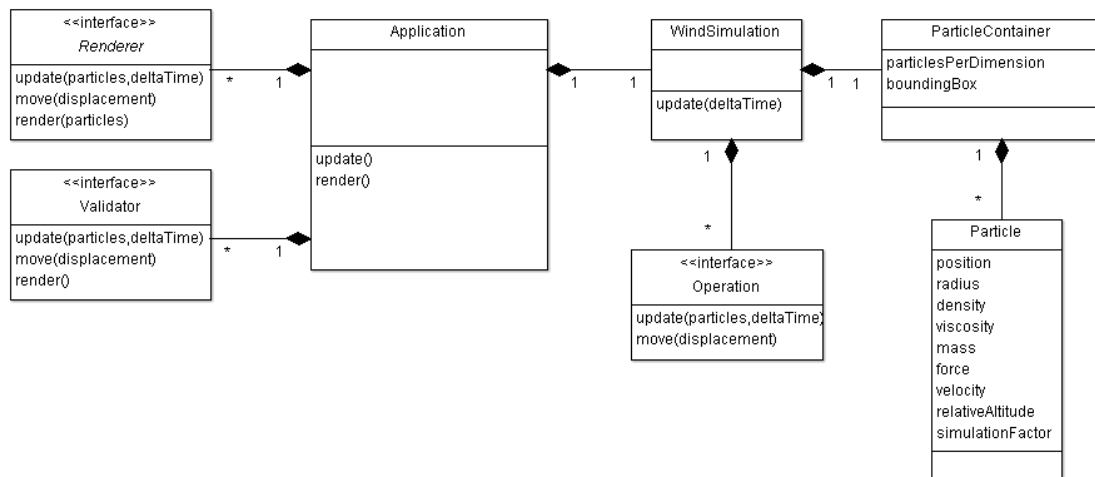


Figure 33: The architecture of the software created in this work. The main loop is found in class Application. The wind simulation is implemented in class WindSimulation and classes that implement the interface Operation. Classes implementing the interfaces Renderer and Validator are used to draw and test the results of WindSimulation.

The simulation itself is split into single operations. The operations share a common interface which makes it easy to plug new ones into the procedure. The interface simply defines an update method and a move method. The move method is called when the simulation domain is moved so that each operation can update spatial information.

The update method is called in every time step and gets a reference to the particle container and the delta time. The operations of the simulation in order of execution are:

OperationGhostParticles which creates and updates the ghost particles described in chapter 4.3. Based on the influence radius used in the SPH simulation layers of ghost particles are created directly below the SPH domain until the influence radius is filled. After that the only responsibility of the operation is to update the ghost particle positions when the simulation domain moves.

OperationCalculateForces is the most complex operation. It implements the SPH algorithm, which was explained in detail in chapters 3.3 and 4.1.

OperationApplyWind enforces that all non-ghost particles move with the global wind velocity by modifying their velocity property. Inflow particles are simply advected according to said wind velocity in a subsequent operation. Their velocity does not need to be changed.

For SPH particles, however, the velocity needs to be modified. The operation enforces that each SPH particle moves along the wind direction with at least the wind speed. Velocity components not parallel to wind velocity remain unchanged.

OperationApplySimpleFriction models friction in between air particles. The operation simply scales each particle's velocity by a factor between zero and one.

OperationLimitVelocity clamps the velocity of SPH particles at a given limit. The limit is based on wind gusts, sudden peaks in wind speed. Wind gusts appear naturally and are described in terms of relation of velocity of the gust to the average wind velocity. See chapter 4.6 for details.

OperationApplyGravity applies a predefined gravity force to all particles.

OperationEvolveSPHParticles moves particles that are neither ghosts nor fully outside the SPH domain. The in previous operations calculated velocity is modified by a given particle's inflow factor. Then the particle's position is updated. Additionally the particle's altitude above terrain is determined and stored.

OperationEvolveInflowParticles moves inflow particles by the inflow factor modified global wind velocity. After that the position's y-component is modified to the terrain altitude below the particle plus the particle's target altitude from the previous operation.

OperationCollideBodies deals with collisions of SPH particles and polygon bodies. Details on collision handling and terrain interaction are provided in chapter 4.4. The operation supports an arbitrary number of polygon bodies.

OperationUpdateSimulationFactor updates the simulation factor property of all non-ghost particles. The factor of particles fully within the SPH domain is 1, the factor of particles fully outside the SPH domain is 0. Particles that are only partly inside have a factor in between, based on how much of the particle is inside.

OperationUpdatedInflowParticles deals with inflow particles leaving the inflow domain downstream, therefore leaving the simulation domain entirely. Leaving particles are teleported back upstream. Details are discussed in chapter 4.3.

The main loop of the application implemented for this work is found in the class `Application`, seen in figure 33. It holds an instance of the wind simulation as well as instances of renderers and validator.

Renderers and validators implement the visualization and validation techniques discussed in chapters 4.9 and 4.10. Interfaces were created for both of them. The interfaces declare update and move methods similar to the operation interface. Additionally render methods are declared. Like the operations of the wind simulation renderers and validators can be added or removed quickly.

Renderers and validators can not manipulate the particle container or its particles. Where needed a constant reference to the particle container is passed to provide necessary information on the state of the wind simulation.

5. Evaluation

In this chapter the tests, conducted on the proposed method are presented and discussed. Those tests have the purpose of validating that the implemented algorithm yields reasonable behavior of the body of fluid and most importantly to see how it performs in regard of the wind effects most interesting for this work. Those effects are ridge lift and turbulence.

First the simulation's behavior in still air is observed to get insight into its behavior in regard of compressibility. Compressing gases takes a lot of effort, e.g. by using a pump. For this work air is assumed to be incompressible. The SPH algorithm does however not guarantee constant volume. Some measures were taken to address this, as described in chapter 4.1. Chapter 5.1 is concerned with validating that the simulated fluid body shows constant or near constant volume.

Chapter 5.2 discusses the effects of interaction of the SPH wind simulation with terrain. Horizontal and vertical deflection of wind on terrain slopes is treated separately. Horizontal wind deflection is discussed in chapter 5.2.1. A simple flat terrain with one cylindrical hill is used. Wind deflection around the hill is measured and compared to reference data.

Chapter 5.2.2 presents the results of tests focusing on vertical wind deflection. When wind hits a slope it causes an updraft because the air is deflected upwards. A slope in the opposite direction on the other hand causes a downdraft since the air flow will drop with the terrain. The drones presented in chapter 4.10 are used to calculate ridge lift over a terrain based on the northern edge of the alps in Upper Austria.

Chapter 5.3 focuses on the proposed method's limitations. The performance of the SPH algorithm is examined in some detail. A real-time framerate was not achieved with the implementation as it is. In chapter 5.3.1 the scaling behavior of the algorithm is tested in regard of different particle count and terrain mesh resolution. Furthermore different parts of the algorithm are timed individually. This gives an insight on which parts of the algorithm are most computationally costly and therefore need most attention in terms of future optimization.

Chapter 5.3.2 focuses on turbulent wind velocity fluctuations produced by the SPH wind simulation method. The turbulence of the simulation is compared to reference data recorded in a real-life flight over mountainous terrain. It will be shown how close to reality the simulation's ability to depict turbulence is.

5.1. Incompressibility Validation

As discussed in chapter 4.1, the SPH algorithm does not inherently guarantee that the volume of a given body of fluid remains unchanged. Measures were taken to avoid compressibility of the simulated body of fluid. The forces between two particles are increased significantly if they come closer to each other than their combined radii. Additional kernels, referred to as near density and near pressure kernels, are used to calculate the additional forces at close proximity. The following tests were conducted to verify the effectiveness of those measures. The setup for the test is a body of fluid with 1 km side length over flat ground. Wind speed is zero. Spatial resolutions from 10 by 10 by 10 to 25 by 25 by 25 SPH particles with increments of 5 per dimension were used. The particle support radius is set to three times of the respective particle diameter.

The fluid's volume was measured by finding the minimum bounding box around all SPH particles and taking its volume. The fluid body is initiated in a cube shape with boundaries to all sides but the top. Its shape is therefore closely approximated by the bounding box. Figure 34 shows the fluid body's volumes frame by frame, measured over 8000 frames.

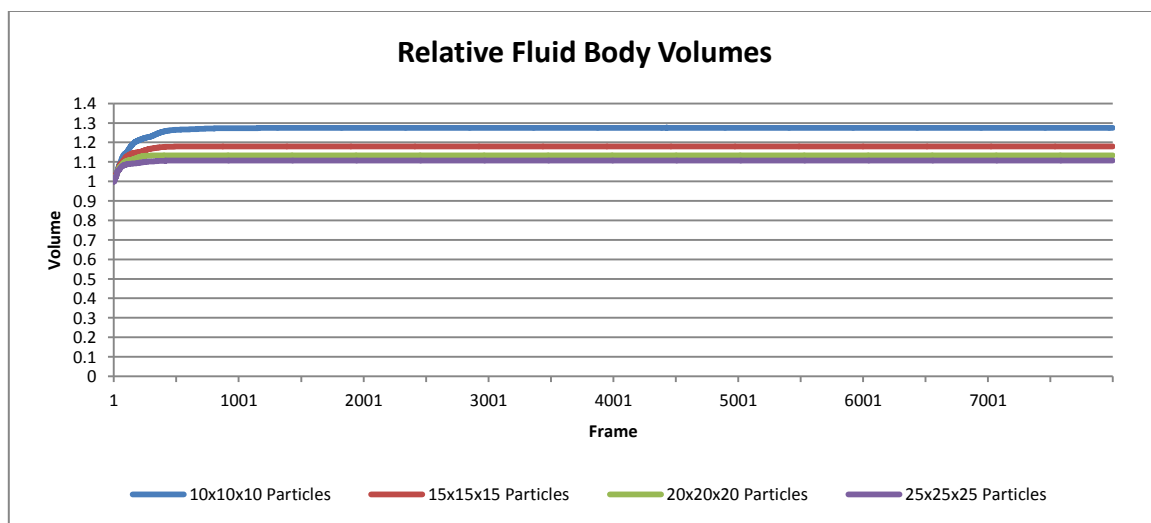


Figure 34: The volume change of a fluid body without wind. After an initial expansion for about 500 frames the volume in all tests remains stable.

All four lines in figure 34 show that the fluid body initially gains volume for about 500 frames, after which it remains the same. From this it is concluded that the initial configuration of particles, on a regular 3D grid, does not result in an equilibrium of pressure forces. The fluid does however settle quickly and remains in a stable state without further volume changes.

As discussed in chapter 3.3, the fluid's density at a given particle of interest (POI) is determined by the number and closeness of other particles within the support radius around the POI. The more particles are within this radius, and the closer they are to the POI's center, the higher the fluid density at the POI's position is. The fluid is in an equilibrium state when the fluid density at all particles is equal to a given rest density.

To verify whether the bodies of fluid in the four conducted tests achieve an equilibrium state, the average as well as the maximum and minimum densities were measured. The plots in figure 35 show those values at the initial state of the simulation, averaged over horizontal particle layers. The number of layers equals the number of particles per dimension.

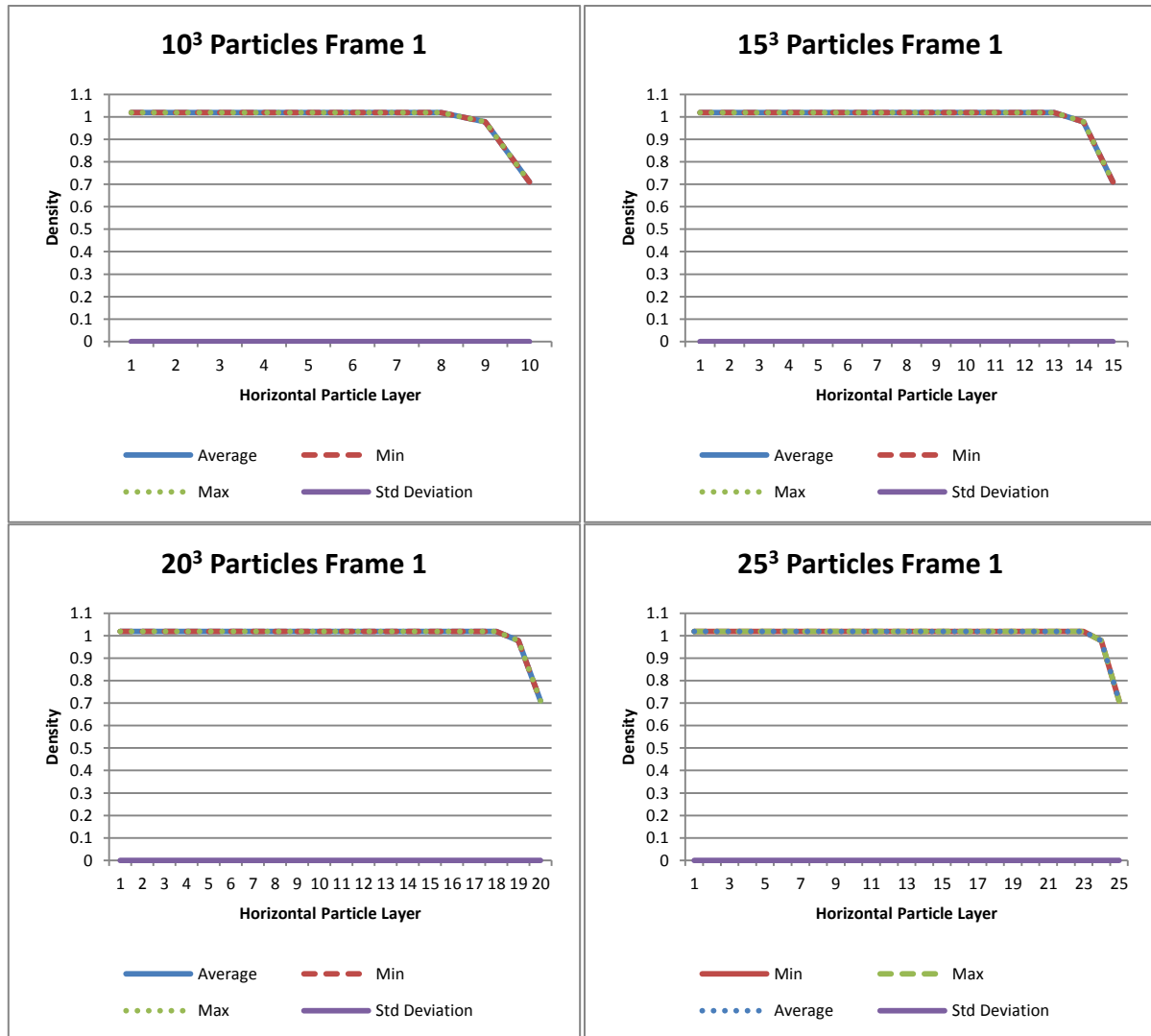


Figure 35: Density distribution per horizontal particle layer for 10^3 , 15^3 , 20^3 and 25^3 SPH particles at frame 1. Layer 1 is the bottom most layer. Density falls off quickly in the top three layers. The density distribution in each layer is even, indicated by similar values for average, minimum and maximum values.

The average, minimum and maximum densities of all particles overlap closely. This indicates that within a single layer the particle densities are close to equal. There are differences between layers though. The average density decreases rapidly at the top layers.

Specifically, the three top-most layers have the lowest average densities. This matches the used particle support radius of three times the particle diameter. At the top layer only the bottom of the sphere, defined by the support radius and particle center, is filled with neighbor particles. Considering this, the observed density drop-off is expected.

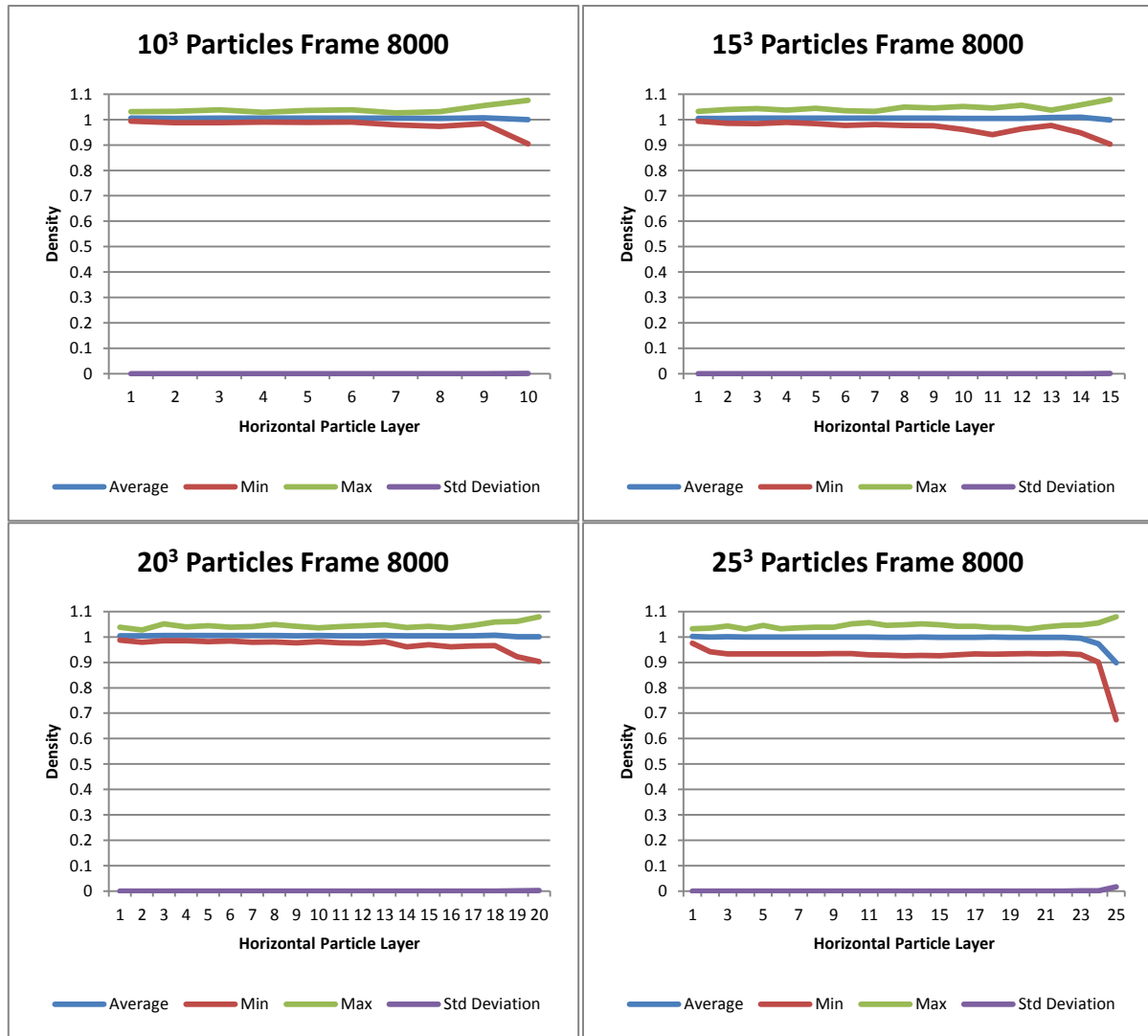


Figure 36: Density distribution per horizontal particle layer for 10^3 , 15^3 , 20^3 and 25^3 SPH particles at frame 8000. The average density is even throughout all layers. Minimum density decreases and maximum density increases at the top most layers.

Figure 36 again shows the average, minimum and maximum densities per horizontal particle layer, this time after 8000 frames. As shown in figure 34, the fluid has ceased to expand by this time and, as concluded above, reached an equilibrium state. If there were still layers of high or low density relative to the other layers, the fluid would move to those low density regions. Therefore, even with constant volume, the above conclusion would be incorrect.

The average density per particle layer is now close to equal in all layers. The average variance of the average densities over all four tests is 1.0×10^{-4} . However, minimum densities decrease and maximum densities increase at the top layers. The low standard deviation, 3.86×10^{-4} on average, indicates that the density of only a few particles deviate greatly from the mean density. The above conclusion, that an equilibrium state was achieved, is therefore supported.

To further verify the effectiveness of the incompressibility measures, the volume measurement was repeated without them. The test setup is the same as for the above test. A fluid body with 1 km side length over flat ground and zero wind speed was created. Again, spatial resolutions from 10 by 10 by 10 to 25 by 25 by 25 SPH particles with increments of 5 per dimension were used. Figure 37 shows the relative volume change observed during the test.

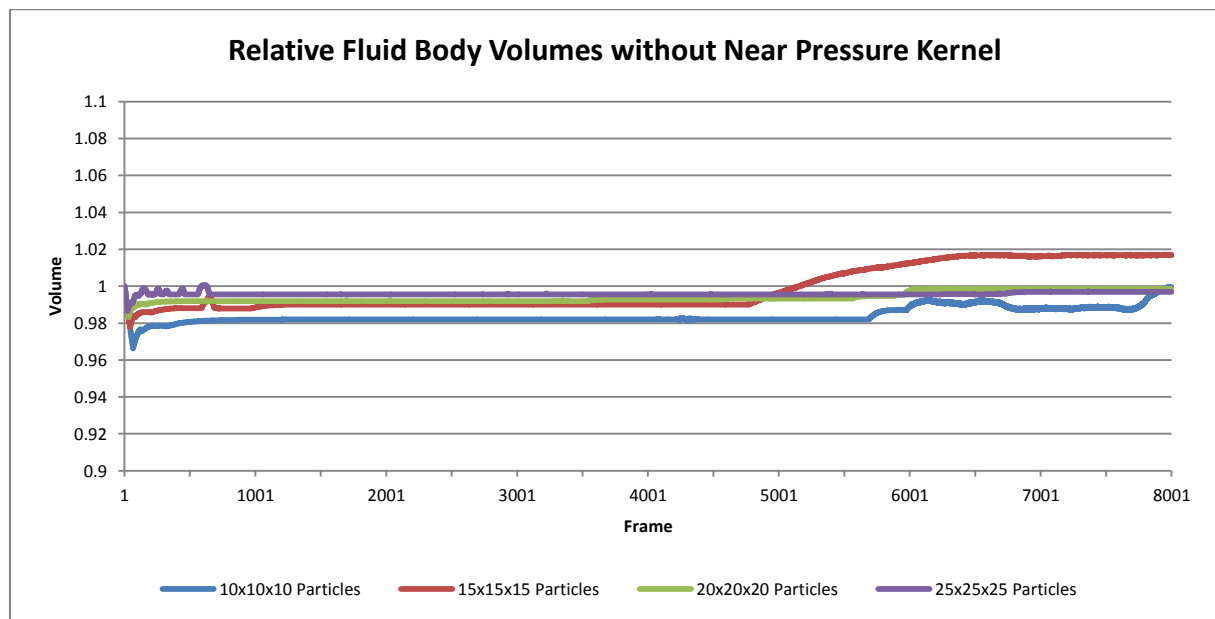


Figure 37: The volume change of a fluid body without wind and no Near Pressure Kernel. Initially the fluid volume slightly decreases. Fluid volume fluctuations then occur after several thousand frames

The volumes shown in figure 37 initially decrease slightly. They stay close to their initial value though. The average volume decrease after 2000 frames is 1.2 %. The average volume increase of the fluid bodies with near kernels after the same amount of elapsed frames is 17.3 %. However, figure 37 reveals that the fluid body's volume does not stabilize like in the fluid bodies with near kernels. A few thousand frames after initialization volume fluctuations are observed. This indicates that without the near pressure and near density kernels the fluid body did not achieve an equilibrium state. Figure 38 shows the average, minimum and maximum densities per horizontal particle layer after 8000 frames corresponding to the fluid bodies of figure 37.

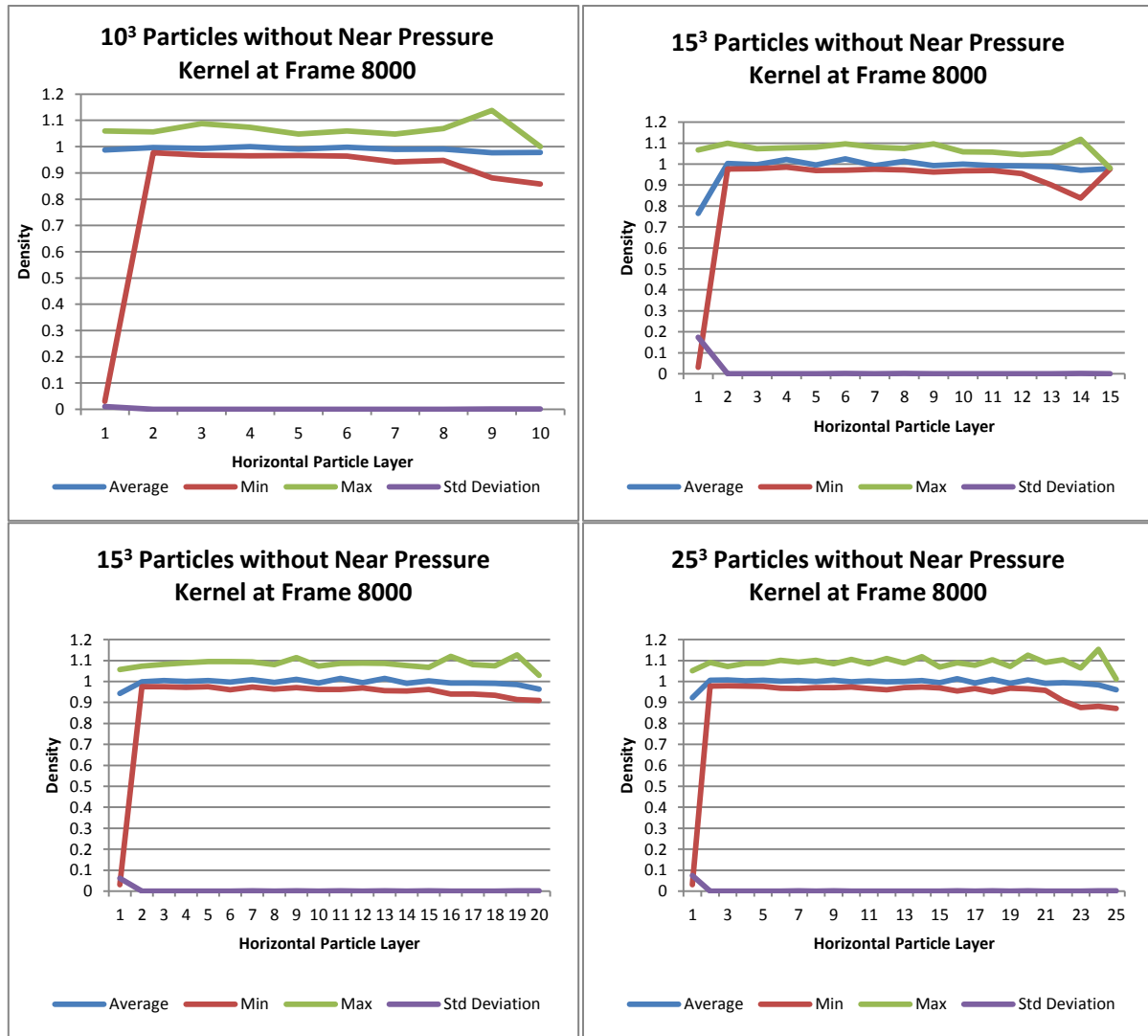


Figure 38: Density distribution per horizontal particle layer for 10^3 , 15^3 , 20^3 and 25^3 SPH particles at frame 8000 without the use of near density and near pressure kernels. Local pressure fluctuations are observed. The fluctuations are significantly higher than in fluids with near kernels

Compared to the fluid bodies with near density and pressure kernels, the density distribution in these four fluid bodies is significantly less even. The average variance of the average densities per particle layer is 1.1×10^{-3} , 11 times higher than with near kernels. The average standard deviation of pressure forces per particle layer is higher as well, 5.6×10^{-3} on average. That is 14 times higher than with near kernels.

5.2. Terrain Interaction Validation

This chapter discusses interaction of the proposed wind simulation and terrain. As wind hits the surface of obstacles, such as hills and mountains, it is deflected. This causes local alterations of wind speed and velocity. The presented tests use the drones introduced in chapter 4.10 to record those alterations.

Horizontal and vertical wind deflection is treated separately. Chapter 5.2.1 deals with horizontal deflection. A simple flat terrain with a single cylindrical hill is used. A drone is set up to fly by close to the hills side. At varying global wind speed, wind velocity near the hill is measured by the drone. The so recorded data is compared to a set of reference data.

The second part of this chapter, sub chapter 5.2.2, deals with vertical wind deflection. It was discussed in chapter 2.1 that when wind hits an obstacle with a wide expanse perpendicular to the wind flow direction, it is deflected upwards. Figure 39 shows simulated air flowing up a hill. This causes a phenomenon called ridge lift. To measure lift forces, drones are set up to fly over a terrain based on a region of Austria. The recorded lift forces are compared to a reference model.

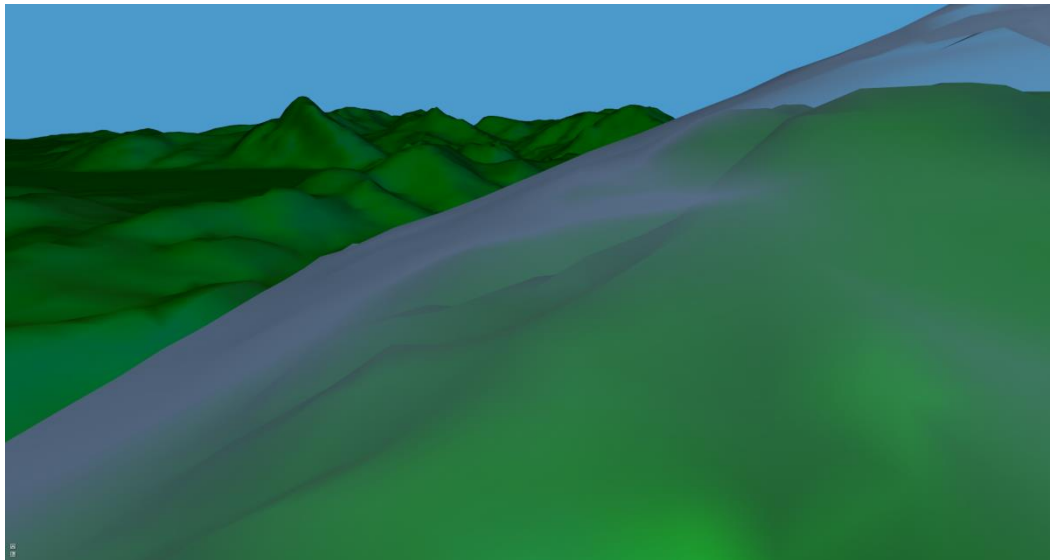


Figure 39: Air flowing up a mountain slope. This flow results in upwind experienced by aircraft flying in this area.

5.2.1. Horizontal Displacement Validation

If wind hits a relatively narrow object, it will flow around it rather than straight over it, as illustrated in figure 40. Consequently, aircraft flying close to such an object experience changes in horizontal wind direction and speed.

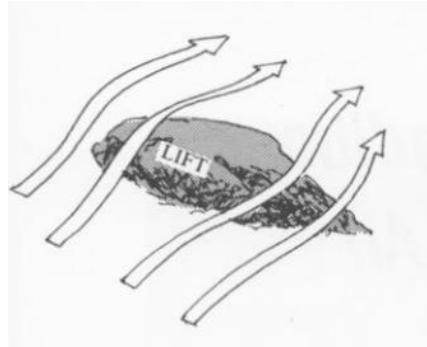


Figure 40: Wind flow around narrow obstacle. Air at the side of the obstacle is deflected sideward rather than upward. Image from Pagen 1992

The following test was conducted to validate the horizontal wind deflection of the SPH wind simulation as proposed in this work. The test setup consists of one cylindrical hill with a diameter of 500 meters. A single SPH drone, as described in chapter 4.10, is set up to pass by the hill at 300 meters above ground, close to its north. The drone's velocity and flight path is recorded. For reference, the airflow around a cylinder with 500 meters diameter was simulated, using the engineering software ANSYS Fluent (ANSYS n.d.). The reference data was computed using Fluent's implementation of the SIMPLE algorithm, an eulerian CFD method.

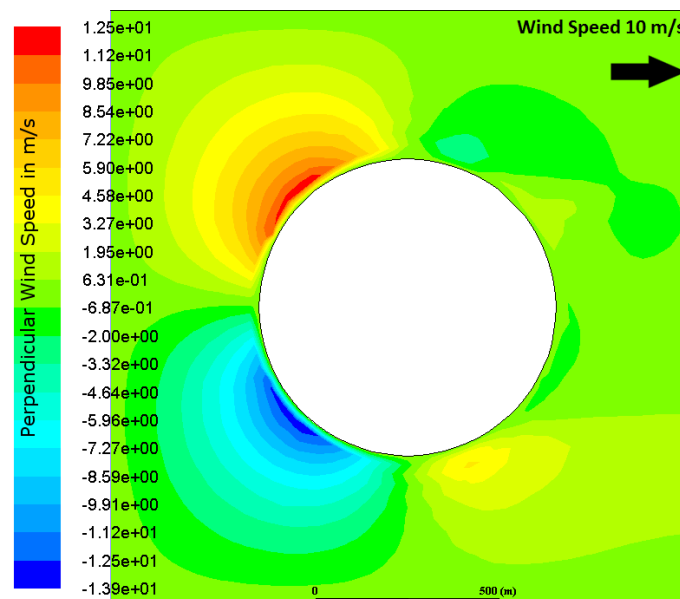


Figure 41: Top down view of wind flow around a cylinder with a diameter of 500 m as simulated by ANSYS Fluent. Colors show the velocity component perpendicular to the wind direction. At the wind facing side of the cylinder wind is strongly deflected away from the cylinder's center. At the cylinder's other side wind flows back towards the center.

Figure 41 shows the reference data. The image is a top down view of the cylinder, which is shown as a white circle. The colors around the cylinder show the wind velocity component perpendicular to the general wind direction. Shades of yellow and red

show a wind deflection towards north, the top of the image. Blue shades show a deflection towards south, the bottom. The west side of the cylinder, which is facing into the wind, deflects the wind away from the cylinder center. On the east side of the cylinder wind is directed back inwards, though with significantly less magnitude.

Drones approaching such a cylindrical hill from the west, either north or south of the hill's center, should therefore be diverted to the north or south respectively. Once they are east of the hill's center point they should be dragged in the opposite direction, though with less force. The test results are shown in figure 42.

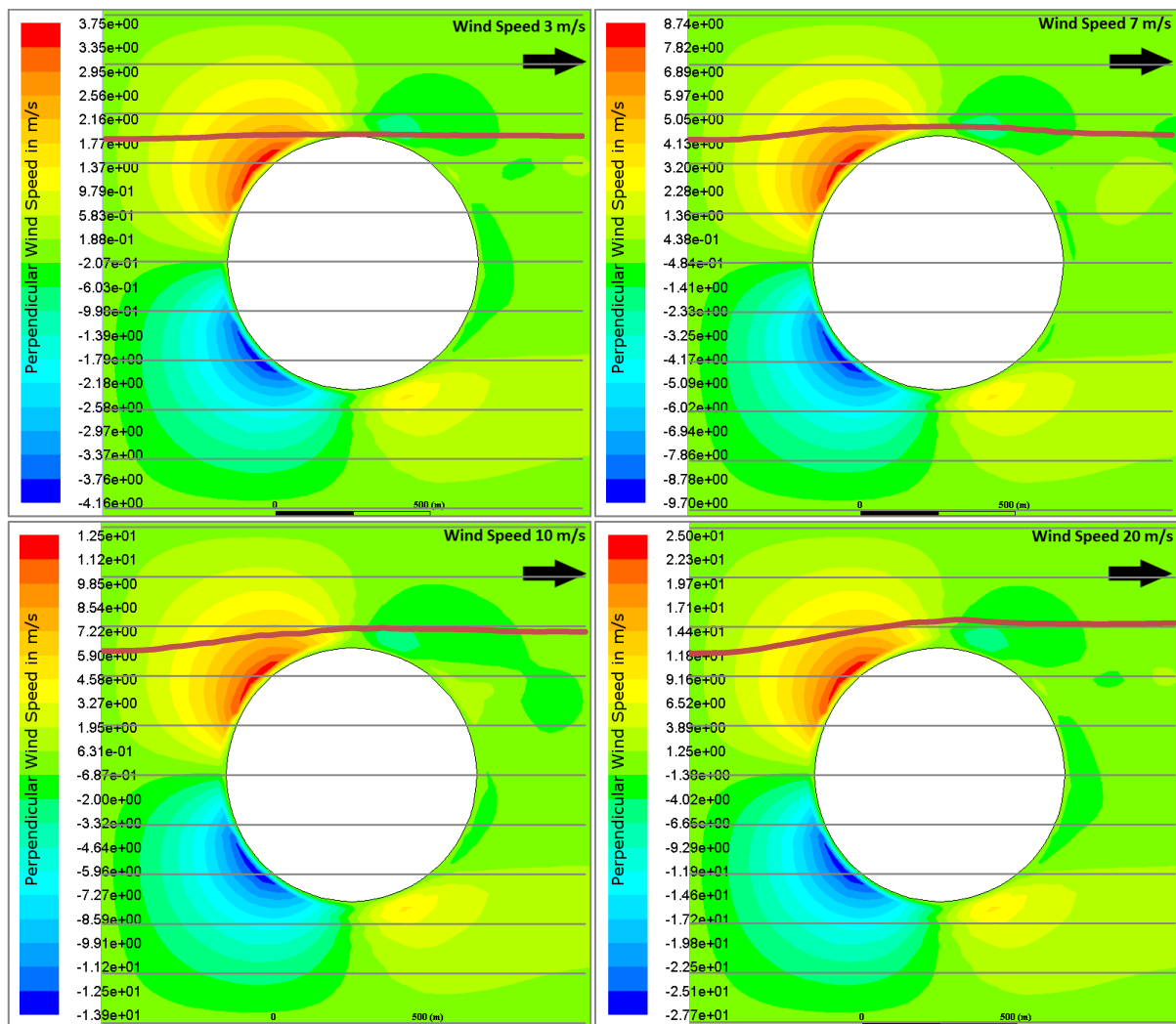


Figure 42: The drone paths around a cylindrical hill plotted against the reference data. Note that with increasing wind speed the magnitude of the winds perpendicular velocity component increases. This is reflected in the drone paths as they divert increasingly far sideways as they approach the cylinder.

Four test runs with increasing wind speed were conducted. A top down view of the drone paths, the red lines, is plotted onto the reference data. Note that the values corresponding to colors in the reference images differ between images.

With increasing wind speed, the magnitude of the perpendicular wind velocity component increases as well. Its maximum is about 1.25 times the global wind speed.

Drone paths in figure 42 follow the indicated wind. On closing in to the cylinder, they start to divert towards north, the top of the image. With increasing wind speed, the northwards diversion increases. This matches with the increasing wind speeds of the ANSYS Fluent simulation. However, it does not show whether the magnitude of the diversions fits the reference wind data. To investigate this, the indicated wind speeds in figure 42 were compared to the recorded wind speeds of the SPH drones at 16 evenly distributed sample points along the flight paths.

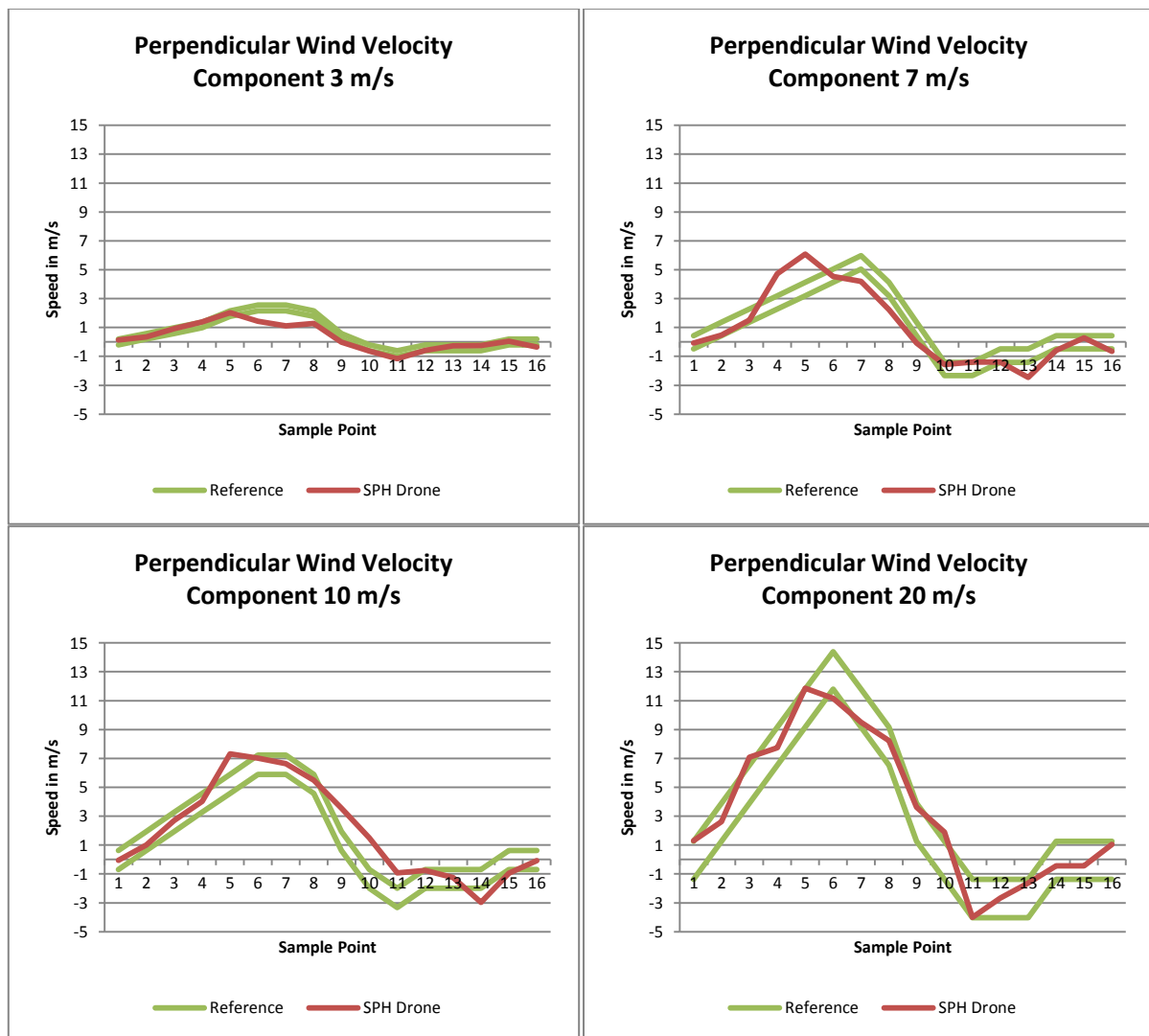


Figure 43: The wind velocity component perpendicular to the wind direction as recorded by the SPH drones is compared to the reference data. The SPH values show closely matching magnitudes and an average correlation of 0.93.

Figure 43 shows the reference wind speeds and the wind speeds recorded by the SPH drone at the sample points. Note that like in figure 42, the reference speeds are given as a range. The two green lines mark this range. The average correlation of the

recorded data points over all four tests is 0.93. Velocity peaks and valleys do not exactly overlap with the reference data, their magnitude however match.

5.2.2. Ridge Lift Validation

Wind flowing over terrain slopes and large obstacles causes up- and downwind, as discussed in chapter 2.1. The following tests were conducted to analyze the proposed method's ability to depict this phenomenon. The setup for this series of tests, as seen in figure 44, consists of multiple pairs of SPH and simple-lift drones, as they are described in chapter 4.10. The simple-lift drones provide reference data. They use the lift calculation method by Forster-Lewis (2007), which was validated by Shah, Menezes and Kolmanovsky (2012), using variometer data recorded during real aircraft flights.

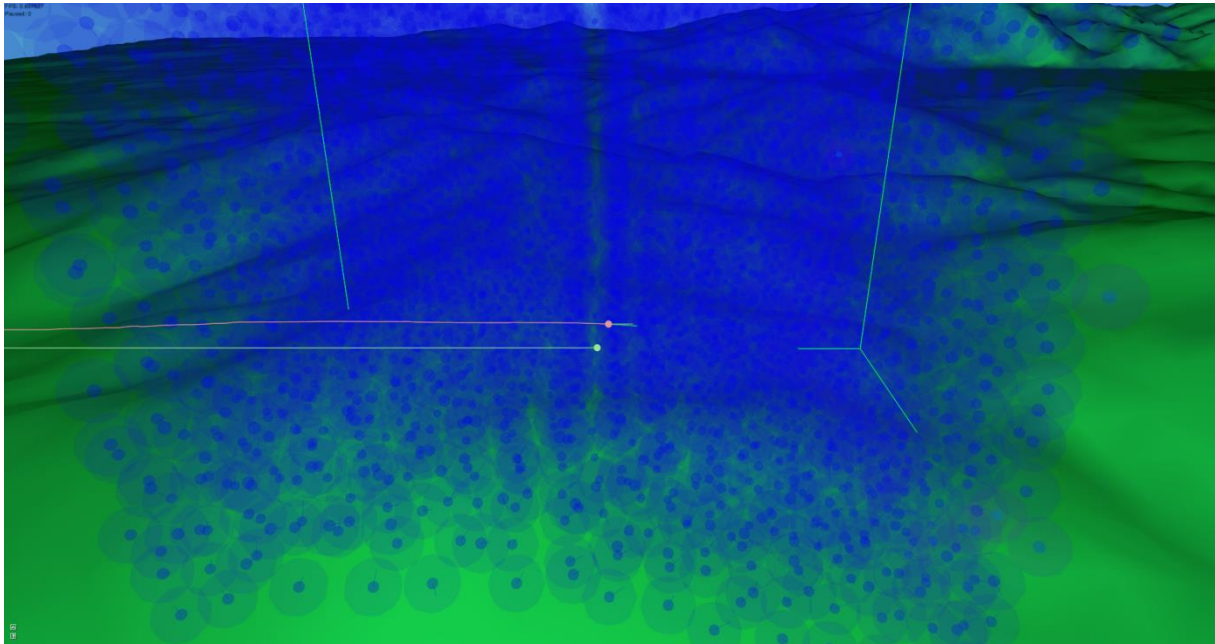


Figure 44: The basic setup of a ridge lift test. The red and green spheres are drones. They are trailed by lines marking their flight path. The red drone also has a velocity vector, showing where it is about to go. SPH particles are rendered as blue spheres. The green lines in the background are parts of the SPH domain bounding box.

A terrain is generated based on parts of south Upper Austria. Figure 45 shows the terrain from above. The area includes the lakes Traunsee and Attersee. It spans roughly from the mountain Katzenstein in the east to the mountain Hohegg in the west, the town Vöcklabruck in the north and Bad Ischl in the south. The terrain spans about 900 km² with a side length of about 30 km. This area was chosen because it offers a variety of flat ground, hills and alpine terrain, with high ridges and deep valleys. It was also selected because of the authors familiarity with this area.

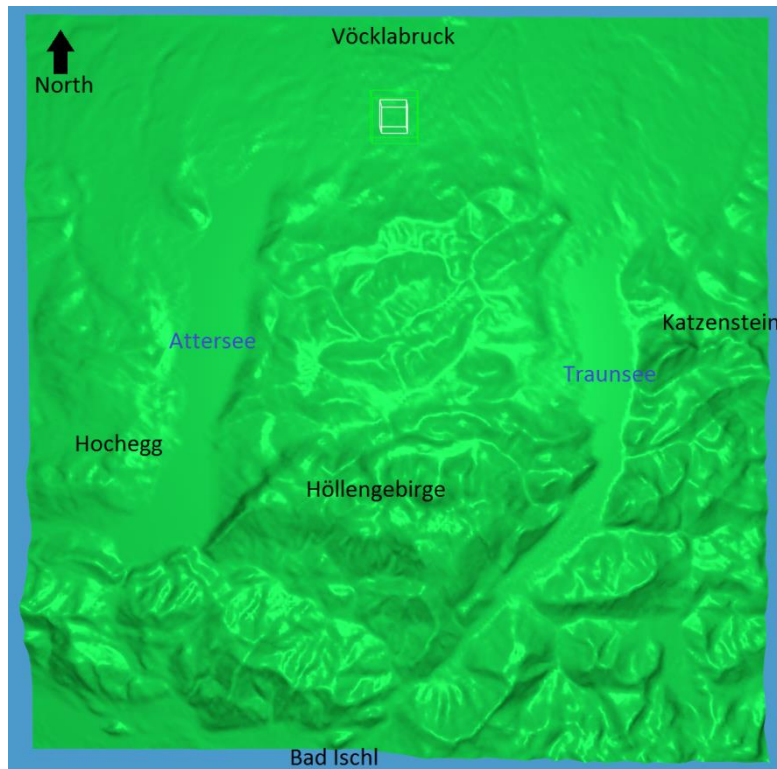


Figure 45: The terrain used in the ridge lift tests. This is an area in the south of Upper Austria. It features rather flat terrain to the north, high mountains in the south and hills in between. The area has a side length of roughly 30 km. The variety of terrain features makes it suitable for wind-terrain interaction tests

Most of the following tests take place over the same region. A cross section of this region is given in figure 46. For the following discussion, names for the major terrain features, seen in this cross section, are defined. Those features are mainly peaks and valleys. The plot runs directly north to south with north being to the left. For better geographical understanding, just north (left) of the plot lies the town Vöcklabruck (Upper Austria) and south lie the mountains Höllengebirge. The distance from plot start to end is about 14 km. From north to south, there are three major peaks. Figure 46 labels them Peak 1 through 3, which is how they will be referred to in the following discussion. Similarly, three valleys were labeled Valley 1 through 3. South of Peak 2 and north of Valley 2 is an area with a number of smaller ridges of roughly the same altitude. This area will be called Plateau 1. Lastly, there is a tall slope to the far south of the plot. This slope is part of the mountains Höllengebirge. It will be referred to simply as Wall.

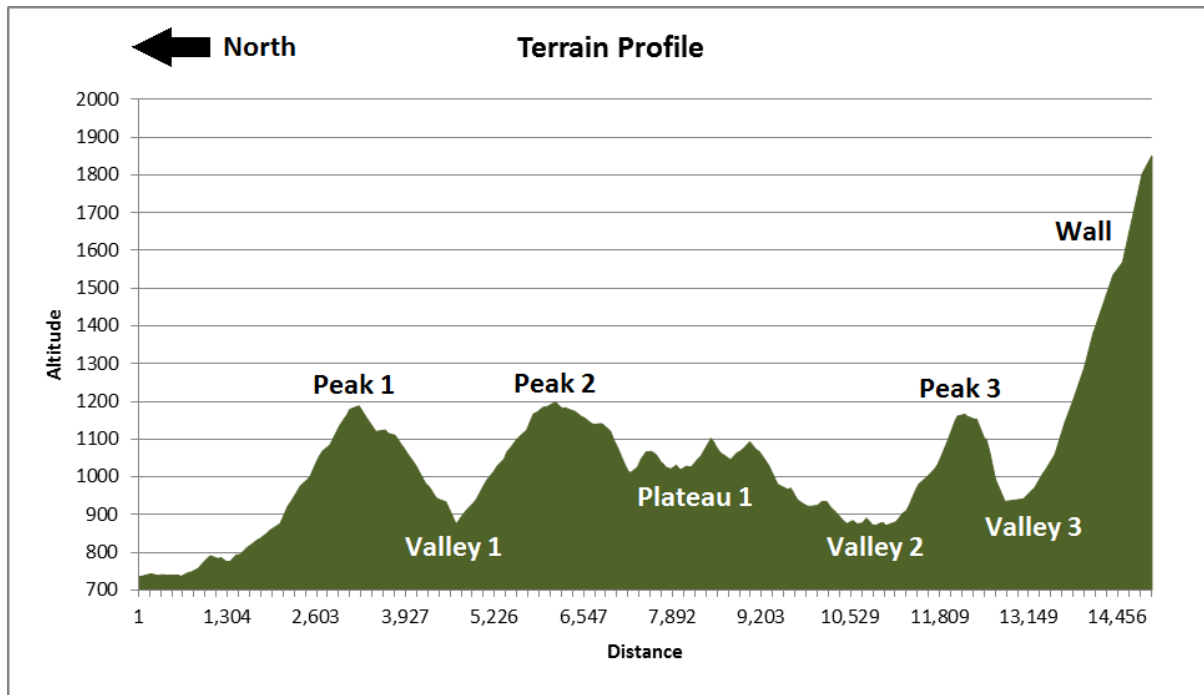


Figure 46: The terrain profile below the flight path of the drones during most of the ridge lift tests. The most significant terrain features are assigned names for future reference

The used SPH domain has a side length of 1000 meters. A particle resolution of 16 by 16 by 16 is used. This resolution was chosen as a compromise between performance and accuracy. The more particles are used the more accurate the simulation is. With the chosen resolution 0.3 frames per seconds on average are achieved on the used test computer.

The drones are given a velocity of 40 m/s. This is roughly the cruising speed of a Cessna 150, a small airplane that is common in general aviation. Wind speeds in a range from 3 m/s to 20 m/s are used. According to the Beaufort scale, 3 m/s is considered a light breeze while 20 m/s is strong wind (Met Office n.d.).

In each test run, 5 pairs of SPH drones and simple-lift drones were used. Their altitudes start at 500 meters above ground and range up to 900 meters above ground, in 100 meter increments. The paths of the SPH drones are drawn in red, the simple-lift drone paths in green. Note that collisions of drones and terrain are ignored. Some paths, mainly the ones of the lowest drones, pass through hills. While the drones are below the terrain surface, they maintain altitude and move on according to their own velocity and global wind velocity. The SPH wind and simple-lift model respectively are applied again after the drone reemerges from beyond the surface. Each test ends, however, as the drone hits the wall.

The resulting flight paths of four test runs, conducted with 3 m/s, 7 m/s, 10 m/s and 20 m/s tailwind, are depicted in figure 47.

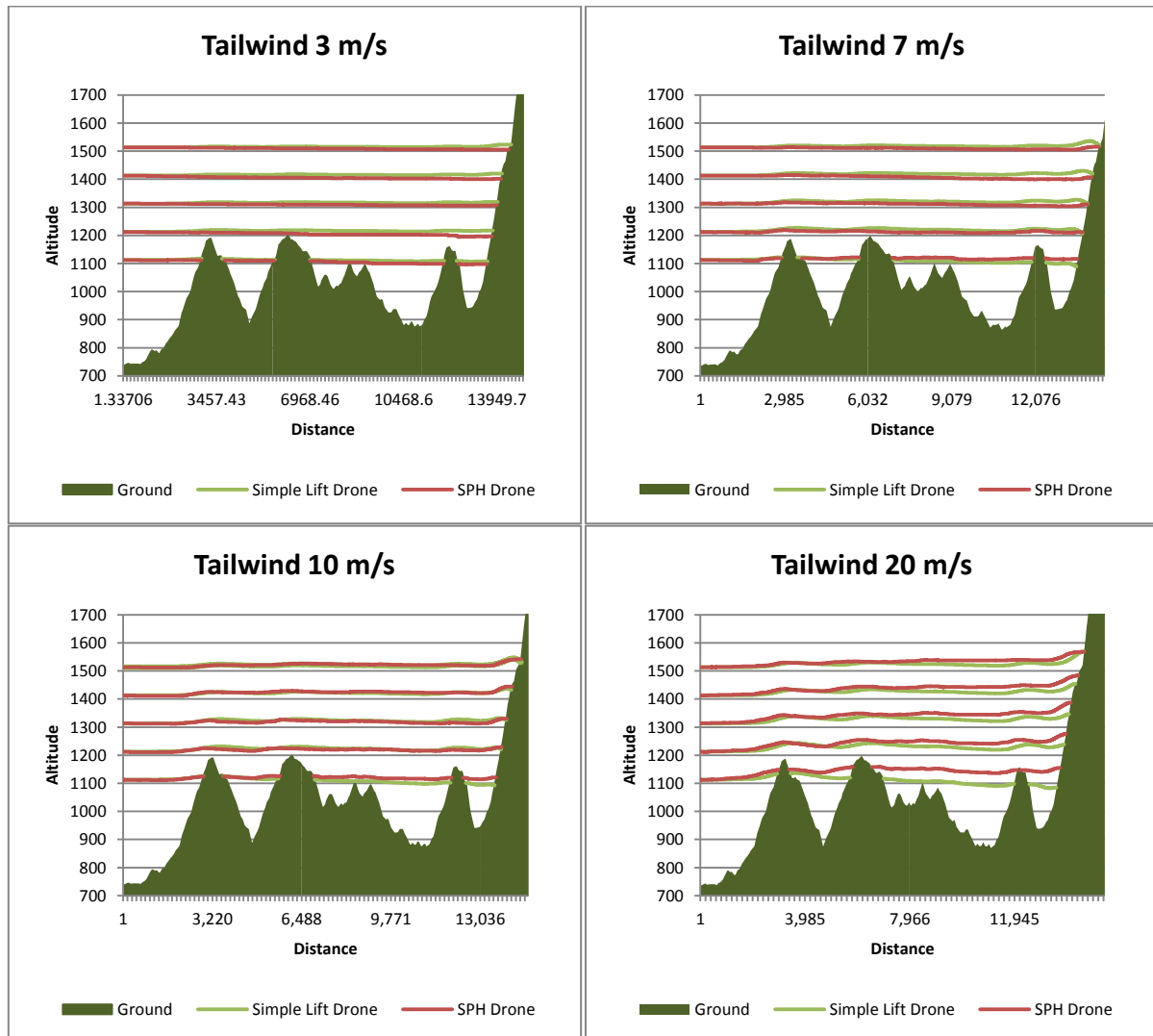


Figure 47: Ridge lift tests with tailwind at varying speeds. Wind direction is left to right. Slopes facing left cause upwind, slopes facing right cause downwind

Figure 47 shows, that the paths of both drone types are similar. All paths end at a higher altitude than their respective start altitude. Most drone pairs hit the wall within a 30 meters altitude difference. Note that some paths in figure 47 intersect the terrain profile. As mentioned, drone-terrain collisions are ignored until reaching the wall. Intersections signal points where drones collided with the terrain, and later reemerged.

The average and maximum altitude differences, as well as the standard deviation thereof, are given in table 1 for all drone pairs.

Wind Speed		500m AGL	600m AGL	700m AGL	800m AGL	900m AGL
3 m/s	Avg. Diff.	6.54	11.53	7.51	11.07	6.51
	Max. Diff.	11.46	23.00	13.74	19.61	17.33
	Std. Dev.	3.84	5.32	3.28	4.08	3.64
7 m/s	Avg. Diff.	8.03	7.53	8.35	11.19	7.91
	Max. Diff.	27.49	13.39	22.75	28.57	22.16
	Std. Dev.	5.82	3.60	5.44	7.04	4.80
10 m/s	Avg. Diff.	9.76	4.03	4.32	2.93	2.33
	Max. Diff.	28.56	10.83	12.75	11.42	10.12
	Std. Dev.	7.47	3.16	3.42	2.05	1.42
20 m/s	Avg. Diff.	30.38	12.88	13.12	12.26	8.70
	Max. Diff.	68.46	37.23	40.25	31.75	23.41
	Std. Dev.	20.57	8.56	9.13	8.57	6.46

Table 1: Average, maximum and standard deviation of altitude differences for the tailwind test runs. The altitude differences increase with increasing wind speed and decreasing altitude.

At 20 m/s tailwind the maximum altitude differences increase to up to 68.46 meters. The drone pair with the highest altitude difference is the lowest pair. The according plot in figure 47 shows that this pair separates while approaching peak 2. The upslope of the peak causes the simple-lift drone no significant altitude gain. The SPH drone however clearly reacts to the peak's slope.

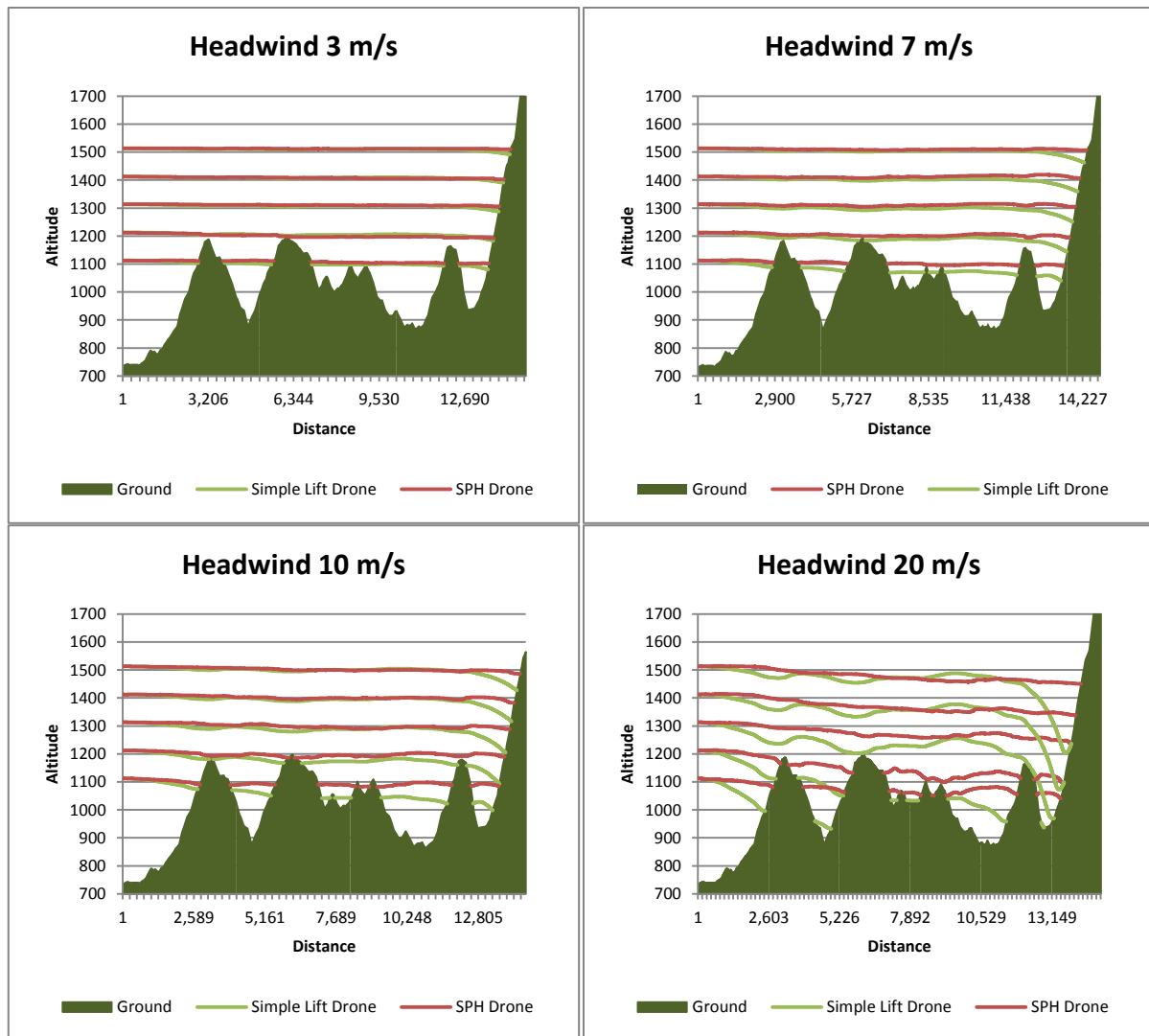


Figure 48: Ridge lift tests with headwind at varying speeds. Wind direction is left to right. Slopes facing right cause upwind, slopes facing left cause downwind.

Figure 48 shows the flight paths of headwind tests with 3 m/s, 7 m/s, 10 m/s and 20 m/s. The altitude difference in this test series tends to be higher than in the tail-wind test, especially at high wind speeds. Table 2 gives the average and maximum altitude differences between pairs of drones, together with the respective standard deviation. The maximum altitude difference of the headwind test is 279.68 meters, occurring at 20 m/s wind speed.

Wind Speed		500m AGL	600m AGL	700m AGL	800m AGL	900m AGL
3 m/s	Avg. Diff.	6.94	4.76	3.67	1.92	2.91
	Max. Diff.	21.66	9.79	19.03	11.93	18.50
	Std. Dev.	4.48	3.10	2.95	1.98	2.97
7 m/s	Avg. Diff.	23.00	12.65	13.09	10.99	6.21
	Max. Diff.	54.43	51.28	53.72	48.68	44.60
	Std. Dev.	14.28	8.57	8.92	9.04	7.42
10 m/s	Avg. Diff.	34.59	20.44	12.28	8.55	6.19
	Max. Diff.	91.90	100.36	84.90	66.88	60.38
	Std. Dev.	24.42	17.00	14.07	11.96	10.05
20 m/s	Avg. Diff.	54.69	72.78	48.63	31.81	28.20
	Max. Diff.	145.28	189.14	279.68	274.18	249.86
	Std. Dev.	32.22	49.10	47.67	49.95	46.89

Table 2: Average, maximum and standard deviation of altitude differences for the headwind test runs. The altitude differences increase with increasing wind speed and altitude. They differences are on average significantly higher than in the tailwind test

The altitude deviation increases faster the closer the drones get to the wall. At 10 m/s wind speed and faster, all simple-lift drones start to descent overhead peak 3 for multiple hundred meters. To give insight into the cause of the strong downwind figure 49 illustrates the distribution of terrain sampling points of simple-lift drones overhead peak 3. As described in chapter 4.10, sample points 1, 2 and 3 are placed from the drone position 250 meters, 750 meters and 2000 meters respectively in upwind direction.

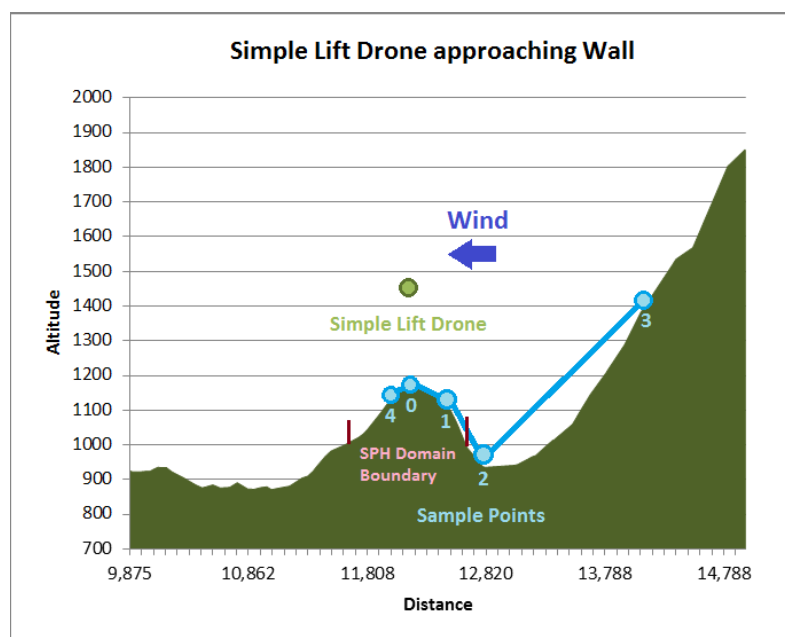


Figure 49: A simple-lift drone approaching the wall in headwind. The great downslope between sample points 2 and 3 causes strong downwind.

Sample points 2 and 3 form a downslope with an altitude difference of roughly 450 meters. This downslope is multiple times higher than the upslopes between sample point 0 and 1 and 1 and 2. The resulting lift factor is therefore negative. As the drone moves further towards the wall, the two upslopes will eventually turn into downslopes. This causes the magnitude of downwind to increase significantly. This can be seen in figure 48, as the simple drones pass over peak 3 and get closer to the wall.

For comparison the northern and southern boundaries of the SPH domain are marked in figure 49. They are 500 meters up- and 500 downwind from the drone position. The wall is not yet within the domain, therefore, it does not yet influence the wind of the SPH-based method. Figure 50 shows a test run with 16 by 16 by 16 SPH particles at 20 m/s headwind and an increased domain size of 3 by 3 by 3 km.

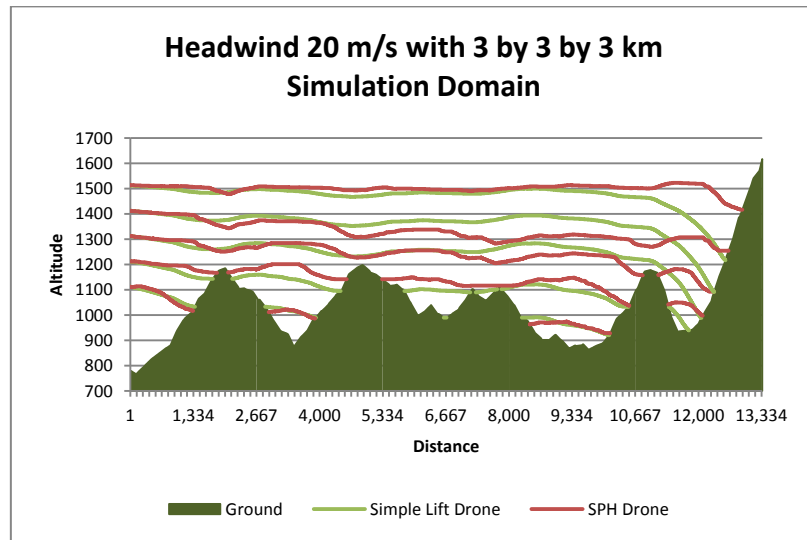


Figure 50: Test run with a 3 by 3 by 3 km SPH domain and 20 m/s headwind. The SPH drones react to more distant terrain features as compared to the 1 by 1 by 1 km domain used in other tests.

In this configuration, the wall enters the SPH domain 1 km sooner. The SPH drones close to the wall start to descent sooner and descent further. Table 3 shows the average, maximum, and standard deviation of altitude differences corresponding to figure 50. The maximum altitude differences decrease compared to the 20 m/s headwind values in table 2.

Wind Speed		500m AGL	600m AGL	700m AGL	800m AGL	900m AGL
20 m/s	Avg. Diff.	8.53	31.32	22.45	44.08	31.59
	Max. Diff.	26.04	102.79	128.16	174.81	223.53
	Std. Dev.	5.41	20.41	23.92	33.85	45.37

Table 3: Average, maximum and standard deviation of altitude differences for the headwind with increased SPH domain size of 3 by 3 by 3 km. The difference on average is less than with a 1 by 1 by 1 km domain.

The downside of increasing the SPH domain size is either a lower spatial resolution or an increase in computation time. The particles of the small domain size have diameters of 62.5 meters, while the particles of the large domain have diameters of 187.5 meters. To match the resolution of the small domain, the large domain would need 27 times more SPH particles. This, in term, increases the computation time 27 times. Performance is discussed in detail in chapter 5.3.1.

Note that compared to the drone paths of figure 48, the paths of figure 47 show significantly less altitude change after peak 3. Note that the wind in figure 47 flows in the opposite direction than in figure 48. The sample points 1, 2 and 3 of the simple-lift drones therefore lie north of the drone and sample point 4 lies to the south. Sample point 4 lies 100 meters from the drone position. The wall influences the simple-lift drone's lift factor significantly later under those circumstances.

Especially at low altitudes, the SPH wind simulation method simulates airflow over terrain with more detail than the simple-lift reference method. Small terrain details have little influence on simple-lift drones. As shown in figure 49, they sample the terrain five times over a range of 2100 meters. The SPH wind simulation, on the other hand, samples terrain at the position of every particle at the bottom of the simulation domain. The number of particles can be set as high as desired. As the particle count increases, the size of individual particle size decreases. Consequently, smaller terrain details can be considered by the simulation.

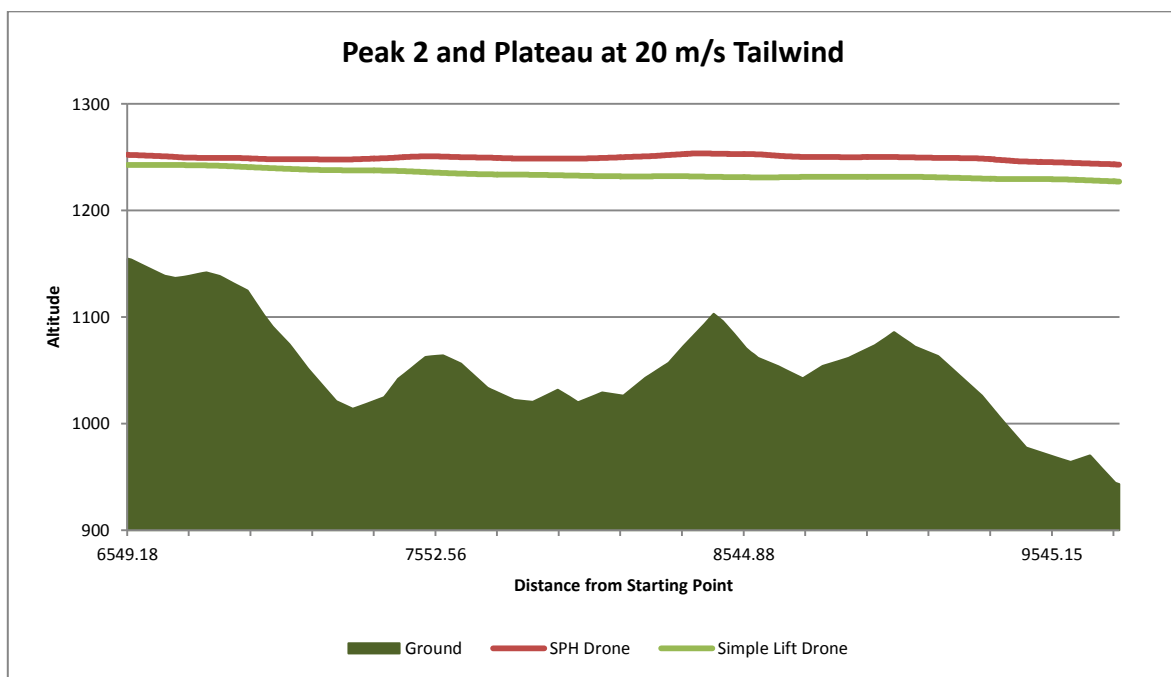


Figure 51: Drone flight paths over the plateau tailwind of 20 m/s. Other than the SPH drone, the simple-lift drone does not react to the low ridges along the plateau.

Figure 51 shows the flight paths of two drones flying over the plateau, with tailwind at 20 m/s. The simple-lift drone shows a steady descent, even overhead wind facing slopes. The SPH drone on the other hand shows slight ascends at those slopes. The SPH drone path and the terrain profile in figure 51 have a correlation of 0.84. The simple-lift drone's path and the terrain profile have a correlation of 0.64. Expanding this analysis to the five SPH drone paths of the test with 20 m/s wind speed in figure 47 results in an average correlation of 0.80. The five corresponding simple-lift drone paths have a correlation with the terrain profile of 0.71. Sections of the drone paths that are underneath the terrain surface were not considered. The reason is that drones are not subject to simulated forces as long as they are underneath the surface. As pointed out earlier, if a drone moves below the terrain level, it maintains its current altitude. Once it moves above the surface again it is subject to lift forces once more.

Figure 52 shows the paths of an SPH and a simple-lift drone over the plateau, with headwind of 20 m/s. Note that they are not as close to each other as the paths in figure 51. The paths in both figure 51 and 52 are taken from the same data sets as the paths in figures 47 and 48. The deviation of simple-lift and SPH drone paths in figure 48 was discussed above in some detail. The paths depicted in figure 52 are the lowest paths of each drone type that are above the terrain surface at the plateau.

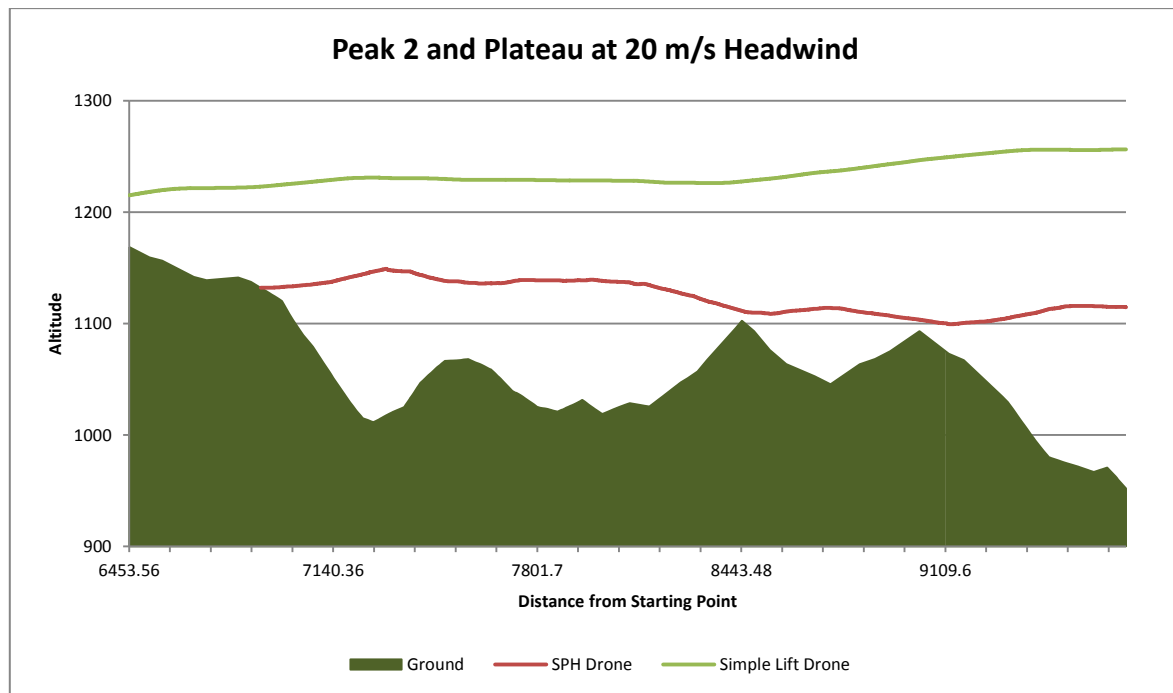


Figure 52: Drone flight paths over the plateau with headwind of 20 m/s. Other than the SPH drone, the simple-lift drone does not react to the low ridges along the plateau.

The simple-lift drone in figure 52 shows ascends over terrain slopes facing into the wind and a slight descent over most of the plateau. The three ridges of the plateau show little effect on it. The SPH drone path shows descends over those three ridges. The correlation of the terrain profile and the SPH drone path is -0.74. The correlation of the terrain profile and the simple-lift drone path is -0.67. For the entire SPH drone paths as shown in figure 48 the average correlation with the terrain profile is -0.60. For the entire simple-lift drone paths and the terrain profile the correlation is -0.47.

5.3. Limitations

In this chapter, some limitations of the proposed SPH wind simulation method are discussed. Even though some basic performance optimizations were implemented, as discussed in chapter 4.7, it does not achieve real-time. Applying it in desktop flight simulation is therefore not possible without further performance optimizations. A detailed discussion on computation time of the method and its parts is presented in chapter 5.3.1.

Mechanical turbulence, as discussed in chapter 2.2, is a phenomenon that can affect the safety of aircraft in mountainous terrain, or at least the comfort of its passengers. Realistic depiction of mechanical turbulence is not achieved by the SPH wind simulation method. Chapter 5.2.2 discusses this limitation in detail.

5.3.1. Performance Evaluation

The following test give insight into the performance of this work's implementation of the proposed wind simulation. Basis of all presented time measurements in this chapter is the Windows API function `QueryPerformanceCounter`, which is used to acquire elapsed processor ticks. The algorithm runs on a single thread on the CPU. The CPU used for testing is an Intel Core i7-6700.

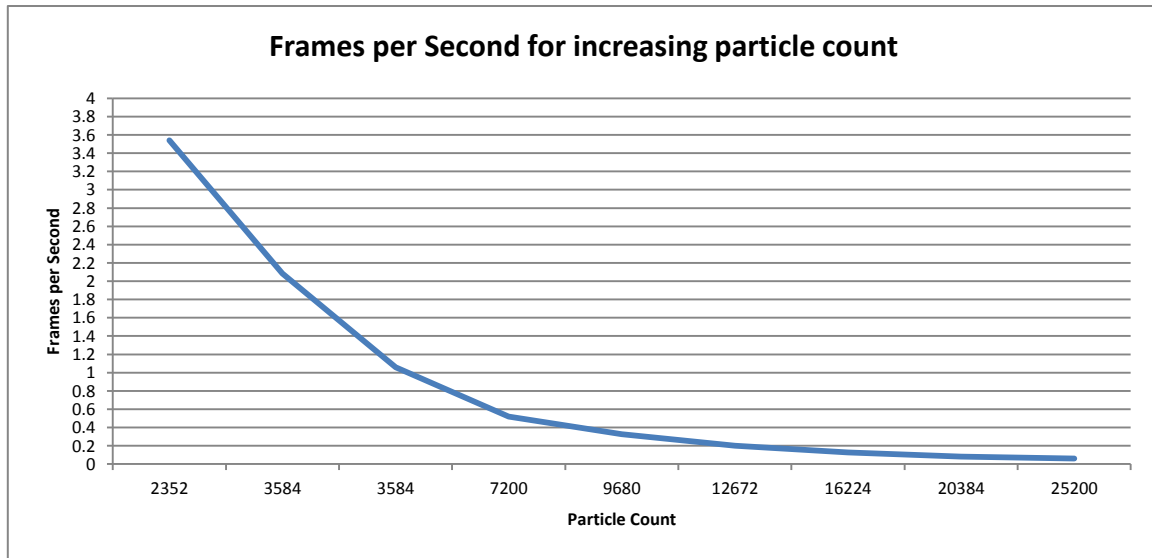


Figure 53: Frames per second (FPS) for increasing particle count. Even with a low particle count FPS is significantly less than real-time

Figure 53 shows the average frames per second (FPS) over 500 frames for increasing number of particles. At the lowest tested particle count, about 3.6 FPS were achieved. An almost 7 times speedup would be required to achieve real-time with 25 FPS. In the following, the most time consuming parts of the algorithm are identified.

First it is determined which parts of the algorithm are the most costly. As it was discussed in chapter 4.11, the simulation is segmented into distinct operations. The computation time of each operation was measured over 1000 frames. For this test 7200 particles were used. Figure 54 plots the relative computation time of each operation frame by frame.

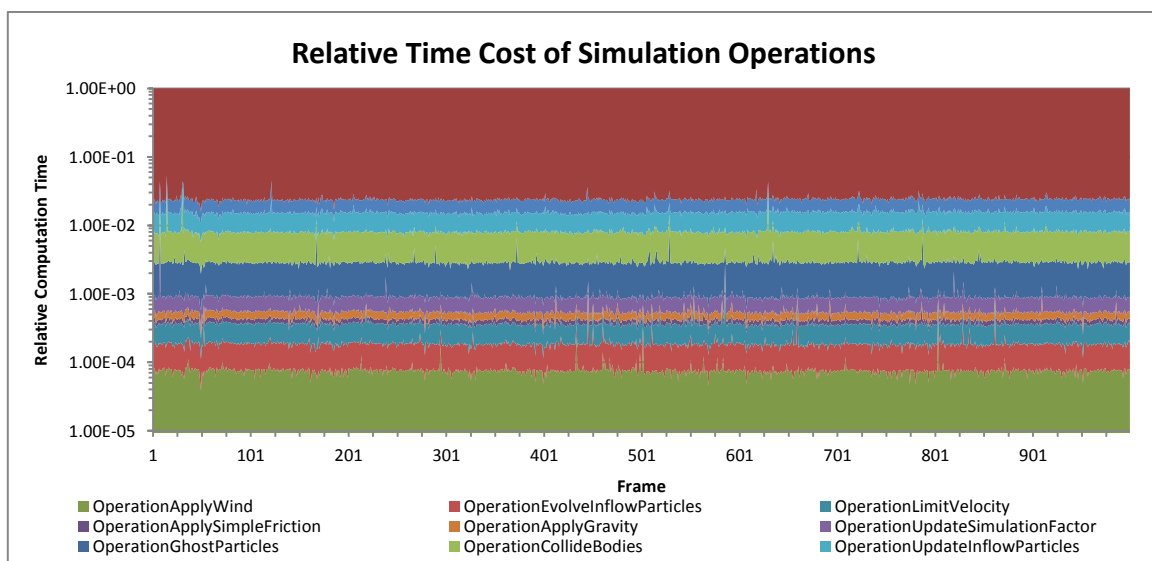


Figure 54: Relative computation time of all simulation operations. About 97 % of the computation time is used by OperationCalculateForces, which is responsible for calculating the fluid's pressure and viscosity forces

As figure 54 shows, about 97 % of computation time is spent on *OperationCalculateForces*. As discussed in chapter 4.11, this operation implements the SPH algorithm as presented in chapter 4.1. Given that it takes up almost all of the computation time, its scaling behavior in regard to the number of SPH particles is now further investigated. Figure 55 shows the average computation time for *OperationCalculateForces* with increasing particle count.

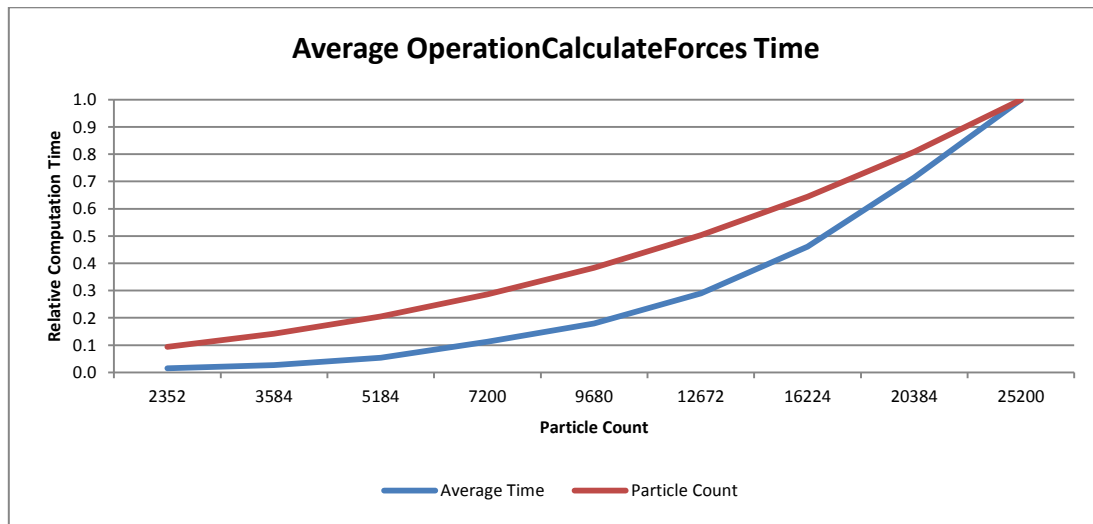


Figure 55: The blue line shows the average computation time of *OperationCalculateForces* relative to the maximum average time. The red line shows the particle count relative to the maximum particle count. The similarity of the lines indicates a strong correlation of particle count and computation time.

Note that the particle count given in figure 55 includes SPH, inflow and ghost particles. As discussed in chapters 4.2 and 4.3, the number of inflow and ghost particles depends on the number of SPH particles and the particle support radius. For a 3D fluid body it can be calculated as

$$p = (s + 2r_d)^2(s + r_d) \quad (5.1)$$

where p stands for the overall particle count, s is the number of SPH particles per dimension and r_d is the support radius in multiples of the particle diameter. The particle counts in figure 55 result from s of 10 to 26 in increments of 2 and r_d of 2.

The scaling in computation time of *OperationCalculateForces* shows a strong correlation with the particle count. Correlation of the results depicted in figure 55 is 0.98.

The scaling behavior of the operations other than *OperationCalculateForces* is now investigated. If they scale worse than *OperationCalculateForces* they can have a significant impact on the overall computation time as the particle count increases. First it is determined which operations take up the most computation time, besides *OperationCalculateForces*. Figure 56 shows the results depicted in figure 54 without *OperationCalculateForces*.

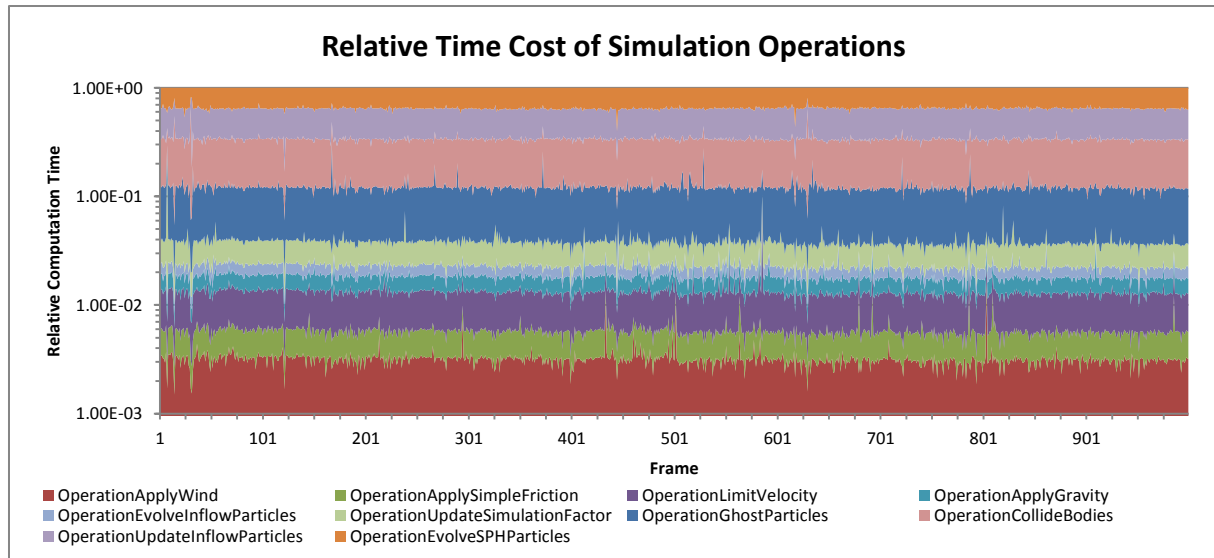


Figure 56: Relative computation time of all simulation operations but *OperationCalculateForces*. *OperationEvolveSPHParticles*, *OperationUpdateInflowParticles* and *OperationCollideBodies* collectively take up about 90 % of the time that is not used for *OperationCalculateForces*

As figure 56 shows, the operations *OperationEvolveSPHParticles*, *OperationUpdateInflowParticles* and *OperationCollideBodies* take up about 90 % of the computation time that is not used for *OperationCalculateForces*. *OperationGhostParticles* uses approximately another 7 %.

All operations iterate all particles of the simulation. Only the four operations mentioned by name do sample terrain altitude for all or some particles. *OperationEvolveSPHParticles* updates the altitude above ground for each SPH particle. *OperationUpdateInflowParticles* updates the position of inflow particles to match a given target altitude above ground. *OperationGhostParticles* does the same thing for ghost particles. *OperationCollideBodies* conducts collision tests of terrain and SPH particles.

The computation time of those operations depends not only on particle count, but also on the complexity of terrain sampling. Terrain sampling was, as discussed in chapter 4.7, optimized so that it has constant complexity. It is therefore assumed that computation time for terrain sampling does not increase with increasing terrain polygon count. To validate this assumption, the computation time of the four operations in question was measured with constant particle count and increasing terrain resolution. Figure 57 shows the result of the test.

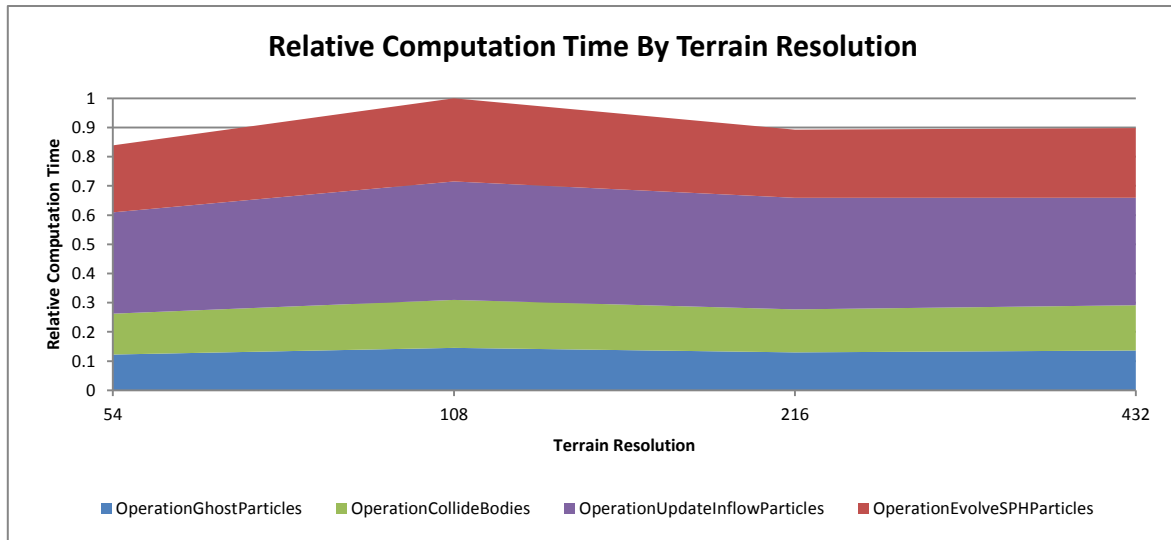


Figure 57: Relative computation time of OperationGhostParticles, OperationEvolveSPHParticles, OperationCollideBodies and OperationUpdateInflowParticles with increasing terrain resolution. Terrain resolution is given in cubes per dimension used during marching cube terrain setup. The computation time shows no significant increase with increase terrain resolution

The terrain resolution in figure 57 is given in number of cubes per dimension used by the marching cube algorithm during terrain creation. Assuming a flat terrain, twice the number of cubes per dimension results in four times the number of polygons. Therefore, the number of terrain polygons in the last test run was 64 times higher than in the first test run. As figure 57 shows, the computation time over all four operations does not vary significantly.

Figure 58 shows the relative computation time for the same four operations with increasing particle count.

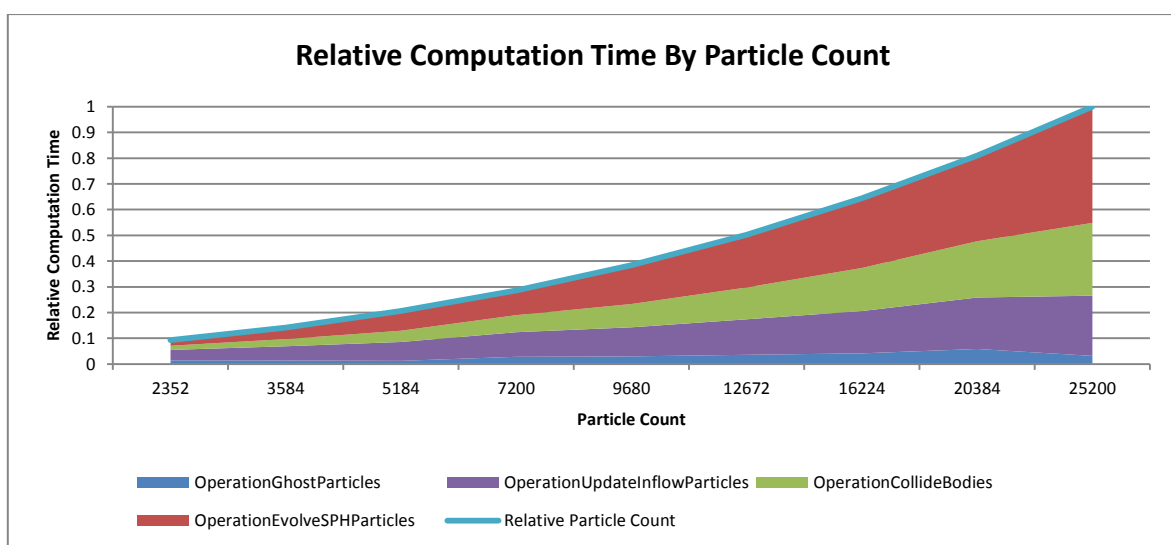


Figure 58: Relative computation time of OperationGhostParticles, OperationEvolveSPHParticles, OperationCollideBodies and OperationUpdateInflowParticles with increasing particle count. The blue line shows the particle count relative to the highest particle count for reference. The accumulated computation time of the four operations shows strong correlation with the particle count

Individually, the computation time of each of the four operations depends on a different particle subset. The accumulated computation time of the four operations in figure 58 shows a strong correlation of 0.98 with the overall particle count.

As mentioned above, the same is true for *OperationCalculateForces*. It is therefore concluded that the relation of computation time between this one and the four operations in figure 58 remains the same for any given particle count.

5.3.2. Turbulence Spectrum Evaluation

This chapter analyses the proposed SPH wind simulation in regards to mechanical turbulence, which was described in chapter 2.2. Turbulence consists of eddies in the air. When those eddies pass a point in space, the wind direction and wind speed will deviate from the average wind speed and direction. The interaction of wind and terrain, which is the focus of this work, causes so called mechanical turbulence.

Note that the formation of eddies in the presented wind simulation method was not observed. Karacostas and Marwitz (1980) show that based on measurements taken around Elk Mountain, Wyoming, the largest eddies in mountainous terrain have a wavelength of 2.5 km. The simulation domain used in the validation process of this work has a side length of 1 km. Therefore large scale eddies can not form. As described in chapter 4.6, the maximum deviation of the velocity of a single particle from the global wind velocity is limited. This inhibits the formation of smaller eddies, which would fit into the simulation domain. Small eddies with wavelength of only several meters are prevented by the size of particles used throughout the validation process. For most tests, particles with a diameter of 62.5 meters were used.

The SPH algorithm is generally not well suited to simulate turbulent flow and eddies. Monaghan (2011) proposes a modified version of the algorithm, called SPH- ϵ , that improves this shortcoming. He achieves good results simulating 2D turbulent flow in a 1 by 1 meter box. Compared to a simple SPH implementation, the computation time of SPH- ϵ increases by about 20 %. For good results 75 to 150 particles per dimension are needed, however. This would significantly increase the computational cost compared to the test setups in this work. Therefore SPH- ϵ was not used.

Even though eddies did not form in the validation process, wind speed and direction fluctuations were observed. Those fluctuations are compared to the results of the measurements taken by Karacostas and Marwitz (1980). Wind speed and direction was recorded on a flight around Elk Mountain at altitudes ranging from 60 meters

above ground up to 1800 meters above ground. 16 measurements per second were taken and then averaged to 8 per second. The data was then partitioned into blocks with 512 data points. The mean value and trends of each block were subtracted from the data points.

To obtain a comparable data set, Elk Mountain and the surrounding terrain was recreated. The three crosswind parts of the real-life flight around Elk Mountain were simulated and the wind velocity components were recorded. The path of the real-life flight is shown in figure 59.

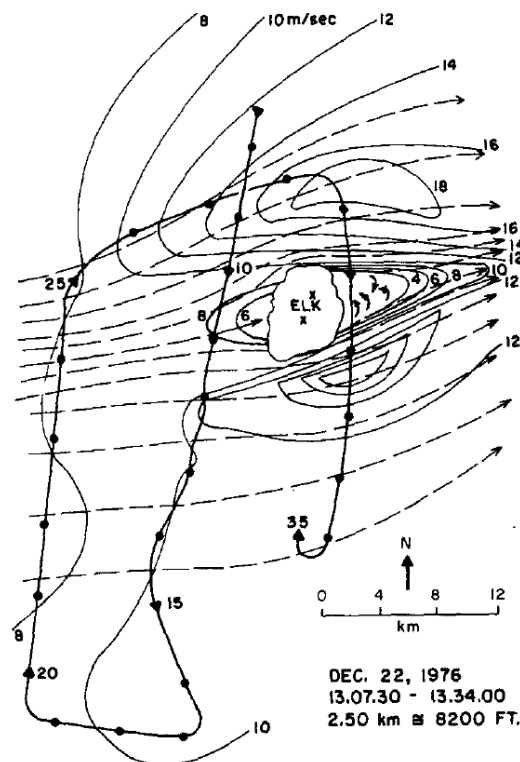


Figure 59: Flight path around Elk Mountain conducted for the measurements of Karacostas and Marwitz. The solid line with dots is the flight path. The intervals between the dots indicate 1 minute of flight time. The thin solid lines are drawn along regions of equal wind speed and the dashed lines are streamlines around the mountain. Image from Karacostas and Marwitz 1980

Wind velocity and direction was set to match the wind conditions depicted in figure 59 for each simulated flight. Similar to the tests in chapter 5.2, the simulation domain size was set to 1 by 1 by 1 km, with 16 by 16 by 16 SPH particles. To match the data recording frequency of Karacostas and Marwitz (1980), the simulation's delta time was set to a fixed value of 0.03125 seconds (32 Hz). The wind velocity components were recorded at every second frame (16 Hz) and the recorded values were averaged to 8 Hz.

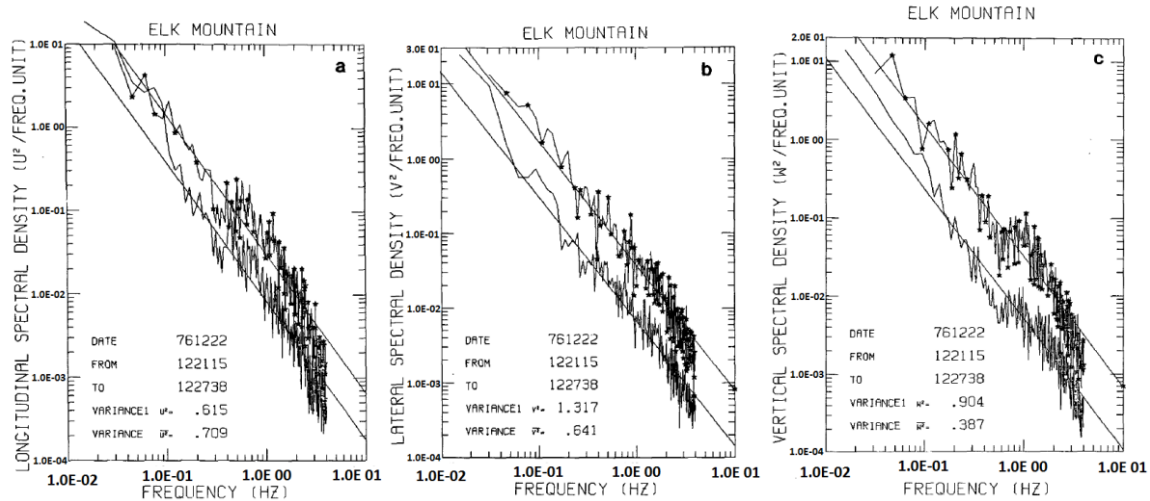


Figure 60: Power spectra of longitudinal (a), lateral (b) and vertical (c) wind components after subtracting means and trends recorded upwind from Elk Mountain. The spectra lines with asterisks show the power spectra at 500 meters above ground, the other lines show the power spectra 1300 meters above ground. The respective straight lines represent the $-5/3$ power law. Image from Karacostas and Marwitz 1980

Figure 60 shows the wind component wise power spectra of Karacostas' and Marwitz' (1980) measurements west of Elk Mountain after means and trends were subtracted. The line marked with asterisks shows data recorded 500 meters above ground, the other line shows data 1300 meters above ground. Figure 61 shows the corresponding results of the SPH wind simulation.

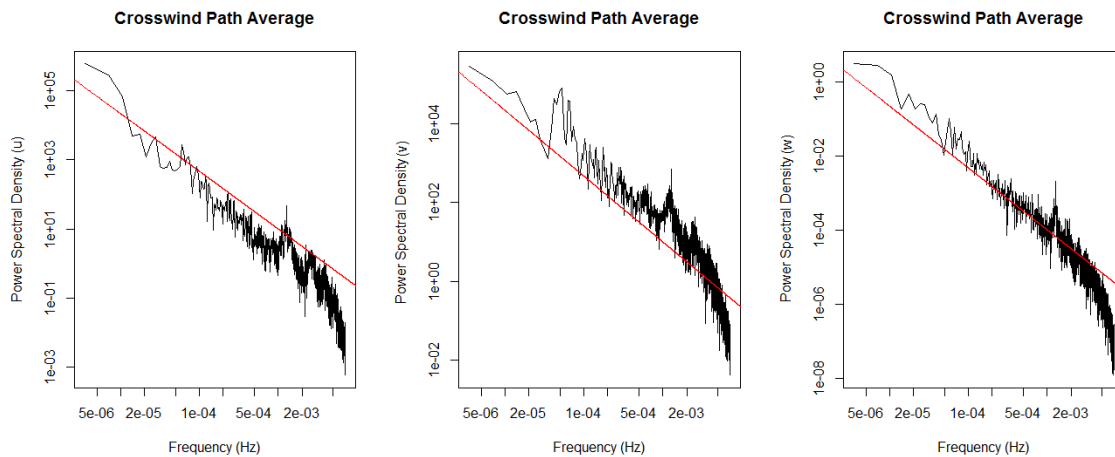


Figure 61: Power spectra of the proposed SPH wind simulation for the longitudinal (u), lateral (v) and vertical (w) wind components after subtracting means and trends. The data was recorded 500 meters above ground. The red lines represent the $-5/3$ power law. Frequency and intensity of the simulated turbulence differs significantly from real-life turbulence

The power spectra in figure 61 were recorded 500 meters above ground, therefore they correspond to the asterisk lines in figure 60. The frequencies of velocity fluctuations observed in the simulation range from 5×10^{-6} Hz to 1×10^{-2} Hz. This is

significantly lower than the frequencies measured by Karacostas and Marwitz, which range from 1×10^{-2} Hz to about 4×10^0 Hz. Furthermore, the intensity of the wind velocity fluctuations of the simulation is higher than the intensities measured in the real-life flight. The vertical wind component is an exception. Consider that, as described in chapter 4.5, the vertical wind velocity component of the SPH simulation is scaled down. This is necessary to achieve a realistic depiction of ridge lift. Consequently, the intensity of vertical wind velocity fluctuations is lower than in the horizontal fluctuations in the simulation.

Karacostas and Marwitz (1980) state that the intensity of turbulence around Elk Mountain increases with decreasing distance from the mountain. It peaks downwind, at close distance from the mountain. This could not be observed in the simulation. All three simulated flights had similar results, regardless of closeness to the mountain. Consider that the SPH simulation domain in this test is a cube with 1 km side length. Therefore, only obstacles closer than 500 meters from the cube center can influence the simulation. According to figure 59, the closest pass of the mountain was about 1 km downwind.

In summary, the simulation results in a lower frequency of wind velocity fluctuations. While the component-wise fluctuations measured by Karacostas and Marwitz (1980) were similar over all dimension, the vertical fluctuations in the simulation have a significantly lower intensity. Contrary to the results of the real-life flight, due to the limited range of the SPH wind simulation, no increasing turbulence intensity could be measured while getting closer to Elk mountain.

6. Summary

A method for simulating wind at altitudes below 1000 meters above ground over vast terrains using smoothed particle hydrodynamics was created. To limit the computational cost of the fluid simulation, it was confined to a region of interest (ROI) of 1 km^3 , centered around a point of interest, e.g. an aircraft.

The SPH algorithm simulates fluids by dividing a given body of fluid into particles and calculating fluid internal forces between them. Particles act as blobs of fluid with fixed volume and mass. The fluid's density at a given particle's position is calculated based on the closeness of other particles. Particles will move from areas with many close particles, therefore high pressure, to areas with less particles, low pressure. The tendency of a fluid to withstand deformation, called viscosity, is simulated as well. The difference of a fluid's velocity to the average velocity of its neighbors is reduced. Fluids with high viscosity allow for less velocity difference than fluids with low viscosity. The algorithm was discussed in detail in chapter 3.3, while implementation details were provided in chapter 4.1.

Without artificial boundaries, the particles of the SPH simulation would simply disperse. Introducing solid boundaries to confine the particles to the ROI is not suitable for the given scenario. To simulate wind, the particles have to be free to move independently of the ROI's position and movement. Open boundaries were added to the simulation to enable this.

Based on the work of Federico et al. (2012), an open boundary method was developed as presented in chapter 4.3. Additional fluid particles surround the ROI in all horizontal directions. Those particles, called inflow particles, do not follow forces calculated with the SPH algorithm. Instead they follow a predefined velocity. This velocity is set to the mean global wind velocity. In any case the inflow particles maintain their altitude above ground.

The inflow particles assure that the SPH particles do not disperse. They also ensure even pressure conditions throughout the set of SPH particles. The area in which inflow particles exist, called inflow area, is set to be large enough to fill the support radii of the outer-most SPH particles.

Since inflow particles move with the global wind velocity they may leave the inflow area. If an inflow particle leaves at a given border, it gets teleported back to the opposite border of the inflow area. So if a particle leaves the inflow area in downstream

direction, it teleports back upstream to the opposite inflow area border. This guarantees a constant number of inflow particles and therefore preservation of mass within the inflow area.

When particles cross the border between the inflow and SPH set however, they switch to the according particle set. Consequently, inflow particles switching to the SPH set are henceforth treated as SPH particles and vice versa. This guarantees that SPH particles can flow with the global wind while still preventing them from dispersing.

To ensure correct pressure conditions at solid boundaries, e.g., the terrain, a method based on Marrone et al.'s (2011) ghost particle method was developed and presented in chapter 4.2. Layers of static fluid particles are placed below the terrain surface. Enough layers are created to fill the support radii of the bottom-most SPH particles.

To keep the ROI centered around a movable object like an aircraft, it has to be freely movable. Particles however need to be able to move independently of the ROI's own movement along the wind flow. Both of those conditions were implicitly fulfilled through the open boundary method. To move the simulation's ROI, the SPH and inflow areas are displaced, the positions of particles however are not modified directly. The open boundary method ensures that upon leaving the inflow area, particles will be teleported back into it. Simultaneously particles that end up within the SPH area after displacement will switch to the SPH set. The ghost particles used for solid boundaries are horizontally displaced with the ROI. Their vertical positions are updated so that they maintain their altitudes relative to the ground.

7. Conclusion

In chapter 5.2 the interaction of the proposed SPH-based wind simulation with terrain was analyzed in regards to horizontal wind deflection and ridge lift. To validate horizontal wind deflection, air flow around a cylinder with a diameter of 500 meters was simulated at varying wind speeds from 3 m/s to 20 m/s. Wind velocities along a flight path close to the cylinder were recorded and compared to a reference data set. The directions of the simulated velocities show an average correlation to the reference data of 0.93. Similarly, the simulated wind speeds along the flight path was found to match the reference data.

Vertical wind deflection was validated by comparing results of the SPH-based wind simulation to a reference method for calculating terrain-induced up- and downwind used in flight simulations current to the date of writing. It was found that the SPH-based method is able to depict the influence of small terrain features better than the reference method. The correlation of the terrain profile and the SPH-based up- and downwind was on average about 10 % higher than the correlation of the terrain profile with the up- and downwind calculated by the reference model.

It was found that in order to depict the influence of distant terrain features, the size of the SPH simulation domain needs to be increased significantly. The reference method probes the terrain up to a distance of 2 km into the wind from the point of interest. To take terrain features at that distance into account, the size of the SPH simulation domain needs to be set to a side length greater than the tested 1 km. This claim was validated by conducting a test with an SPH domain of 3 km side length. The downside of increasing the SPH domain size is an increase in computation time or a decrease of the spatial resolution of the simulation.

The proposed SPH-based method offers some advantages over the methods discussed in chapter 3.1. Those can be summed up as

- Improved detail
- No manual setup
- No pre-processing
- Simulation in 3D

The SPH method offers an improved level of detail over Forster-Lewis' (2007) simple-lift model. The simple-lift model uses five sample points along a single axis

between 100 meters downstream and 2000 meters upstream from the point of interest to calculate terrain slopes. Smaller terrain features are therefore easily missed. The resolution of the SPH method on the other hand can be set as needed. However, increased computational costs are to be considered when increasing the particle resolution.

The lift boxes used in Microsoft's Flight Simulator X and Lockheed Martin's Prepar3D require manual setup. Each box needs to be placed by hand. The time that scenario designers can spend on this task limits the area and the level of detail of where up- and downwind will be encountered. Furthermore, a change in wind direction would cause the need to update the positions of lift boxes. This means that the scenario needs to be changed by hand. The proposed SPH method does not require any manual setup. Changes in wind direction and speed are handled automatically.

Some versions of CumulusX require a data base that stores slope directions and angles for points along the terrain surface. This data base has to be created from the elevation data of a given terrain before CumulusX can be used. The SPH method can interact with a given terrain mesh directly, no additional data is needed.

Lastly, both Forster-Lewis' (2007) lift model and CumulusX only calculate lift forces, or in other words vertical wind deflection. The SPH method calculates wind deflection in any direction. Slopes that are horizontally not exactly parallel or perpendicular to the global wind direction will cause local alterations of the wind direction. The SPH-based method was evaluated not only in regard of ridge lift but also in regard of its ability to correctly depict those horizontal alterations of wind direction and found to be reasonably accurate. Figure 62 shows how the SPH wind simulation causes side-ward deviation to the path of an aircraft.

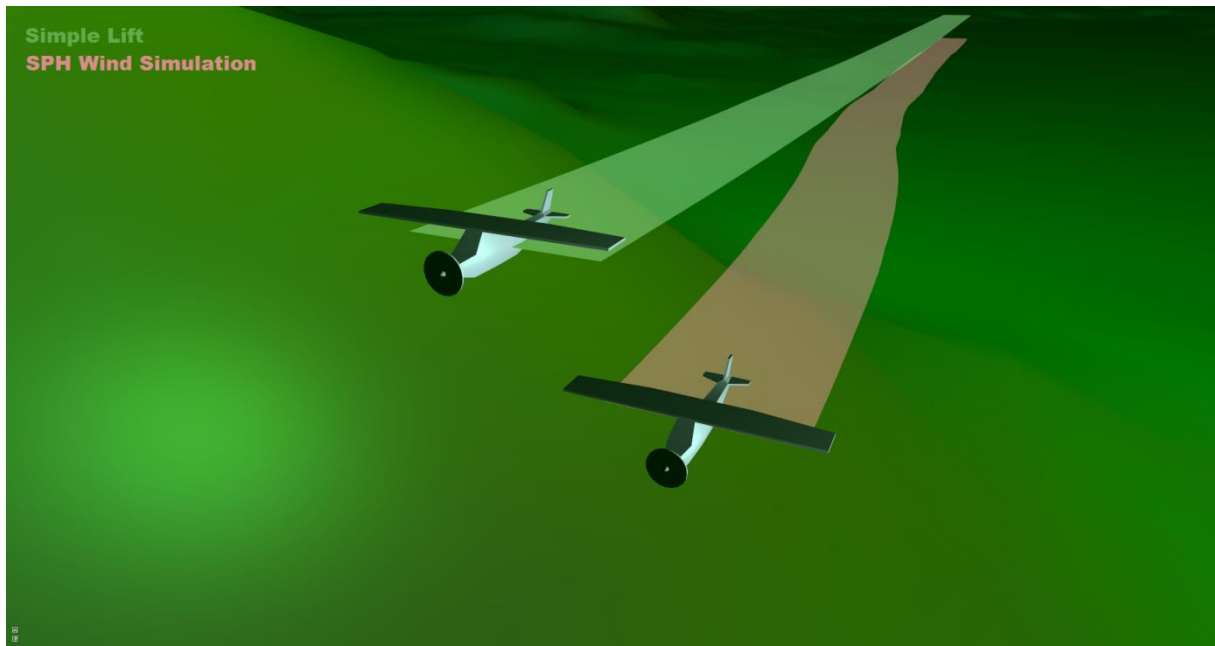


Figure 62: A drone using the SPH wind simulation (red trail) and a drone using the simple-lift model of Forster-Lewis (2007) (green trail). While the green drone flies straight along the nearby hill slope the red drone diverts to its left, due to wind being deflected sideways. The simple-lift model does not take sideways wind deflection into account.

The proposed SPH wind simulation method has some significant limitations. The performance of the method as presented in this work is not sufficient for use in real-time flight simulation. With the test setup used for the terrain interaction tests in chapter 5.2, an average of 0.3 frames per seconds was achieved. This setup used 16 by 16 by 16 SPH particles and an SPH domain side length of 1 km. As discussed in chapter 5.3.1, about 97 % of the computation time is spent on calculating the fluid's internal forces according to the SPH algorithm. Computation time increases linearly with the particle count.

While the SPH wind simulation does produce wind velocity fluctuations reminiscent of air turbulence, it was found to be inadequate to realistically depict this phenomenon. The simulation was compared to real-life recordings of air turbulence in mountainous terrain. Neither the frequency nor the intensity of turbulence in the simulation matches the reference data. The SPH algorithm is generally not well suited to simulate turbulent flow and eddies. Some work on this issue has been done in recent years. Monaghan (2011) for example proposes a modified version of the algorithm called SPH- ϵ that improves this shortcoming. As Monaghan (2011) states, SPH- ϵ requires about 75 to 150 particles per dimension to achieve good results. Due to high computation cost it was not implemented in this work.

8. Outlook

The need for the lift force adjustment, which was discussed in chapter 4.5, indicates an issue that still needs to be addressed. As a quick and efficient fix for an overestimation of lift forces, a correction model was introduced. Future work should find and resolve the cause for this overestimation. It is suspected that the open boundary method, as described in chapter 4.3, is the reason for the issue. The details on why the open boundaries are suspected to be the cause are given in chapter 4.5. Future work needs to investigate this. If the open boundaries can be confirmed to cause the lift force overestimation, this part of the wind simulation method needs to be modified to resolve the issue.

Another possible goal for future work is to develop an implementation of the SPH wind simulation method that can be applied to a desktop flight simulation. In order to do so, the performance limitations have to be addressed. The average frame rate achieved during the tests in chapter 5.2 was 0.3 frames per second. It was shown that about 97 % of the computation time is spent on calculating the fluid's internal forces. Methods to improve the performance of the SPH algorithm have been presented in the past.

Goswami et al. (2010) and Krog and Elster (2010) for example propose a parallel SPH implementation on the GPU using CUDA. Both works acknowledge the neighborhood search as computationally costly. A naive implementation has complexity of $O(N^2)$, where N is the number of particles. Each particle's distance to each other particle has to be computed. Storing the neighborhood removes the need to repeat this for density, pressure and viscosity calculation, as well as for possible additional physical properties. Nevertheless, it remains very costly. Using regular grids is a common optimization scheme for this problem. Grid cells can be created with a side length equal to the kernel support radius. Particles only need to check other particles in the same cell and the direct neighbor cells. Krog and Elster use it to determine neighborhoods. To optimize subsequent access of particles and neighbors, they sort the particle array so that particles and neighbors are close to each other. Goswami et al. propose an approach also based on a regular grid. They present a way to calculate neighborhood efficiently on the GPU.

As Goswami et al. (2010) point out, calculating density, pressure and viscosity cannot be done simultaneously since pressure depends on the result of density cal-

culatation and viscosity depends on the velocities resulting from pressure calculation. Calculating a given property for one particle does not influence the other particles however. This means that density can be calculated simultaneously for all particles. The same is true for pressure, viscosity force and any other fluid property. Therefore moving these calculations onto the GPU can yield a significant performance increase.

As discussed in chapter 5.3.2, turbulent air is not realistically depicted by the proposed SPH wind simulation method. Neither frequency nor intensity of turbulence in the simulation was found to be realistic. In recent years there has been some work on improving the simulation of turbulent flows with the SPH algorithm. Monaghan (2011) introduced a slightly compressible SPH algorithm with an added turbulence model called SPH- ϵ . Monaghan applies the SPH- ϵ algorithm to a turbulent flow within a two-dimensional box with no-slip boundaries for validation, a setup that has been well studied experimentally. He concludes that this algorithm can match the used reference data given a sufficient spatial resolution. The additional computation time compared to a standard SPH implementation is about 20 %. However, the spatial resolution required to produce good results was found to be 75 to 150 particles per dimension. The computation time for the SPH wind simulation method of this work with this kind of particle count would be very high and far beyond interactivity. Provided that a sufficient performance improvement is achieved in future implementations of the SPH wind simulation, the addition of a turbulence model would be worthwhile since mechanically induced turbulence is a result of real wind-terrain interaction.

In order to further improve the realism of the terrain interaction of the proposed wind simulation, real-life flight data as reference will be needed. The reference data used in this work is based on the ridge lift model by Forster-Lewis (2007). This model is however limited to calculating up- and downwind, while the proposed SPH wind simulation method simulates wind flow in all three 3 spatial dimensions. Therefore, a more detailed set of reference data would benefit the further development of the SPH wind simulation method. This leads to the related topic of obtaining suitable data.

Shah, Menezes and Kolmanovsky (2012) use flight data recorded by gliders. Flight path recording using GPS sensors is common in gliders. Recorded paths are the basis for glider competitions. Those recorded paths consist of latitude and longitude coordinates as well as altitude of aircraft at regular time intervals. Based on this data

and additional information on the location of thermals, Shah, Menezes and Kolmanovsky (2012) show how to extract information on ridge lift.

Karacostas and Marwitz (1980) recorded in-flight wind data to characterize air turbulence over mountainous terrain. They used sensors, consisting of an inertial navigation system (INS) and a special sensor boom to measure gusts around the aircraft. The so recorded data is more detailed than the data obtained from glider flight paths. The used sensors are not widely used in small aircraft though.

As an alternative approach, the use of smart-phones can be explored. Many smart-phones offer a number of sensors that can be useful for this application. GPS can be used to record flight paths like in the discussed glider data. Acceleration sensors and gyroscopes can be used to record the movement and rotation of the aircraft. If the quality of the phone sensor data is sufficient, information on wind gusts around the aircraft can be derived. Based on this data, up- and downwind produced by the SPH wind simulation as well as turbulence can be validated.

9. List of Figures

Figure 1: Obstacles deflecting flowing air. Pagen 1992	6
Figure 2: Upwind based on slope angle and wind velocity. The steeper the slope and the stronger the wind, the stronger the resulting upwind will be. Pagen 1992	7
Figure 3: An eddy flowing by point A. Pagen 1992	7
Figure 4: Mechanically induced eddies around an obstacle. Pagen 1992	8
Figure 5: Flow in and out of a rectangular region in space. Pedley 1997	18
Figure 6: Fixed ghost particles and interpolation points along a surface. Marrone et al. 2011	24
Figure 7: Misplaced ghost particle due to a sharp boundary edge. Dobusch 2016	25
Figure 8: Inflow, SPH and outflow particles with fixed ghost particles underneath. Marrone et al. 2012	26
Figure 9: The images show two fluids with different densities. Solenthaler and Pajarola 2008	31
Figure 10: 2D poly6 kernel used for density estimation. Dobusch 2016	32
Figure 11: 2D poly 6 kernel gradient. Dobusch 2016	33
Figure 12: Gradient of 2D spiky kernel. Dobusch 2016	34
Figure 13: Laplacian of 2D poly6 kernel. Dobusch 2016	35
Figure 14: Laplacian of 2D spiky kernel. Dobusch 2016	35
Figure 15: Laplacian of 2D viscosity kernel. Dobusch 2016	36
Figure 16: Particle distribution after 270 frames. Dobusch 2016	37
Figure 17: Distribution of ghost particles below the terrain surface. Dobusch 2016	39
Figure 18: 3D Particle setup from above, yellow particles are SPH particles. Dobusch 2016	41
Figure 19: Particles leaving the inflow area downstream are teleported back upstream. Dobusch 2016	41
Figure 20: The circumsphere test conducted to find potential particle-triangle collisions. Dobusch 2016	43
Figure 21: Particle displacement to resolve terrain intersection. Dobusch 2016	44
Figure 22: The result of simple particle velocity reflection upon terrain collision. Dobusch 2016	45
Figure 23: Average lift forces of the simple-lift model by Forster-Lewis (2007) and the SPH wind simulation for wind speeds from 3 m/s to 20 m/s. Dobusch 2016	47
Figure 24: Altitude-wise normalized lift force ratios of the simple-lift model to the SPH method. Dobusch 2016	48

Figure 25: Wind speed-wise normalized lift force ratios of the simple-lift model to the SPH method. Dobusch 2016.....	49
Figure 26: The lift force adjustment functions <i>f_{alt positive}</i> , <i>f_{alt negative}</i> and <i>f_{speedw}</i> . Dobusch 2016.....	50
Figure 27: Inflow particle intersection due to change in terrain slope. Dobusch 2016.....	50
Figure 28: Smooth step function used to cap particle velocity. Dobusch 2016.....	51
Figure 29: Particles flowing downhill unrestricted. Dobusch 2016.....	52
Figure 30: Particles and mesh wireframe of a smoke surface. Dobusch 2016.....	55
Figure 31: 3 layers of smoke. Dobusch 2016.....	57
Figure 32: Plot of the normalization function for slopes. Dobusch 2016.....	58
Figure 33: The architecture of the software created in this work. Dobusch 2016.....	61
Figure 34: The volume change of a fluid body without wind. Dobusch 2016.....	65
Figure 35: Density distribution per horizontal particle layer for 10^3 , 15^3 , 20^3 and 25^3 SPH particles at frame 1. Dobusch 2016.....	67
Figure 36: Density distribution per horizontal particle layer for 10^3 , 15^3 , 20^3 and 25^3 SPH particles at frame 8000. Dobusch 2016.....	68
Figure 37: The volume change of a fluid body without wind and no Near Pressure Kernel. Dobusch 2016.....	69
Figure 38: Density distribution per horizontal particle layer for 10^3 , 15^3 , 20^3 and 25^3 SPH particles at frame 8000 without the use of near density and near pressure kernels. Dobusch 2016.....	70
Figure 39: Air flowing up a mountain slope. Dobusch 2016.....	71
Figure 40: Wind flow around narrow obstacle. Pagen 1992.....	72
Figure 41: Top down view of wind flow around a cylinder with a diameter of 500 m as simulated by ANSYS Fluent. Dobusch 2016.....	72
Figure 42: The drone paths around a cylindrical hill plotted against the reference data. Dobusch 2016.....	73
Figure 43: The wind velocity component perpendicular to the wind direction as recorded by the SPH drones is compared to the reference data. Dobusch 2016.....	74
Figure 44: The basic setup of a ridge lift test. Dobusch 2016.....	75
Figure 45: The terrain used in the ridge lift tests. Dobusch 2016.....	76
Figure 46: The terrain profile below the flight path of the drones during most of the ridge lift tests. Dobusch 2016.....	77
Figure 47: Ridge lift tests with tailwind at varying speeds. Dobusch 2016.....	78

Figure 48: Ridge lift tests with headwind at varying speeds. Dobusch 2016	80
Figure 49: A simple-lift drone approaching the wall in headwind. Dobusch 2016.....	81
Figure 50: Test run with a 3 by 3 by 3 km SPH domain and 20 m/s headwind. Dobusch 2016	82
Figure 51: Drone flight paths over the plateau tailwind of 20 m/s. Dobusch 2016.....	83
Figure 52: Drone flight paths over the plateau with headwind of 20 m/s. Dobusch 2016	84
Figure 53: Frames per second (FPS) for increasing particle count. Dobusch 2016	86
Figure 54: Relative computation time of all simulation operations. Dobusch 2016.....	86
Figure 55: The blue line shows the average computation time of OperationCalculateForces relative to the maximum average time. Dobusch 2016.....	87
Figure 56: Relative computation time of all simulation operations but OperationCalculateForces. Dobusch 2016.....	88
Figure 57: Relative computation time of OperationGhostParticles, OperationEvolveSPHParticles, OperationCollideBodies and OperationUpdateInflowParticles with increasing terrain resolution. Dobusch 2016.....	89
Figure 58: Relative computation time of OperationGhostParticles, OperationEvolveSPHParticles, OperationCollideBodies and OperationUpdateInflowParticles with increasing particle count. Dobusch 2016.....	89
Figure 59: Flight path around Elk Mountain conducted for the measurements of Karacostas and Marwitz. Karacostas and Marwitz 1980.....	91
Figure 60: Power spectra of longitudinal (a), lateral (b) and vertical (c) wind components after subtracting means and trends recorded upwind from Elk Mountain. Karacostas and Marwitz 1980 .	92
Figure 61: Power spectra of the proposed SPH wind simulation for the longitudinal (u), lateral (v) and vertical (w) wind components after subtracting means and trends. Dobusch 2016.....	92
Figure 62: A drone using the SPH wind simulation (red trail) and a drone using the simple-lift model of Forster-Lewis (2007) (green trail). Dobusch 2016.....	98

10. List of Tables

Table 1: Average, maximum and standard deviation of altitude differences for the tailwind test runs.	79
Table 2: Average, maximum and standard deviation of altitude differences for the headwind test runs.	81
Table 3: Average, maximum and standard deviation of altitude differences for the headwind with increased SPH domain size of 3 by 3 by 3 km.....	82

11. List of Abbreviations

2D.....	two-dimensional
3D.....	three-dimensional
CFD.....	computational fluid simulation
FPS.....	frames per second
GPS.....	Global Positioning System
Hz.....	Hertz
INS.....	Inertial Navigation System
km.....	kilometer
m/s.....	meters per second
POI.....	particle of interest
ROI.....	region of interest
SPH.....	smoothed particle hydrodynamics

12. References

- Ágústsson, Hálfván. Ólafsson, Haraldur. 2009. "Forecasting wind gusts in complex terrain." *Meteorology and Atmospheric Physics* 103 (1): 173-185
- ANSYS. n.d. ANSYS Fluent – CFD Software. <http://www.ansys.com/Products/Fluids/ANSYS-Fluent> (accessed April 24, 2016)
- Berndt, Jon. Peden, Tony. Culp, David. De Marco, Agostino. Duke, Lee. Froehlich, Mathias. Megginson, David. Hofman, Erik. Gidenstam, Ander. 2009. JSBSim source code (Version 1.0). https://sourceforge.net/projects/jsbsim/files/JSBSim_Source/JSBSim%20v1.0%20Release%20Candidate%202/ (accessed January 18, 2016)
- Bridson, Robert. 2008. "Fluid Simulation for Computer Graphics." Wellesley: A K Peters Ltd
- Colagrossi, Andrea. Landrini, Mauricio. 2003. "Numerical Simulation of Interfacial Flows by Smoothed Particle Hydrodynamics." *Journal of Computational Physics* 191 (2): 448-475
- Desbrun, Mathieu. Gascuel, Marie-Paule. 1996. "Smoothed particles: a new paradigm for animating highly deformable bodies." *Proceeding of the Eurographics workshop on Computer animation and simulation '96*: 61-76
- Desbrun, Mathieu. Kanso, Eva. Tong, Yiyi. 2006. "Discrete differential forms for computational modeling." *ACM SIGGRAPH 2006 Courses*: 39-54
- Federico, Ivan. Marrone, Salvatore. Colagrossi, Andrea. Aristodemo, Francesco. Antuono, Matteo. 2012. "Simulating 2D open-channel flows through an SPH model." *European Journal of Mechanics – B/Fluids* 34: 35-46
- FlightGear Wiki. n.d. FlightGear. FlightGear. <http://wiki.flightgear.org/FlightGear> (accessed March 20, 2014)

- Forster-Lewis, Ian. 2007. "Efficient simulation of topographic lift in Microsoft Flight Simulator." http://carrier.csi.cam.ac.uk/forsterlewis/soaring/sim/fsx/dev/sim_probe/sim_probe_paper.html (accessed October 25, 2015)
- Galway, David. 2009. "Urban Wind Modeling with Application to Autonomous Flight." Master thesis, Carleton University
- Gingold, Robert. A. Monaghan, Joseph J. 1977. "Smoothed particle hydrodynamics: theory and application to non-spherical stars." *Monthly Notices of the Royal Astronomical Society* 181 (3): 375-389
- Goswami, Prashant. Schlegel, Philipp. Solenthaler, Barbara. Pajarola, Renato. 2010. "Interactive SPH simulation and rendering on the GPU." *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*: 55-64
- Karacostas, Theodore S. Marwitz, John D. 1980. "Turbulent Kinetic Energy Budgets over Mountainous Terrain." *Journal of Applied Meteorology* 19 (2): 163-174
- Krog, Øystein E. Elster, Anne C. 2010. "Fast GPU-Based fluid simulation using SPH." *Proceedings of the 10th international conference on Applied Parallel and Scientific Computing* (2): 98-109
- Lee, Dooyong. Sezer-Uzol, Nilay. Horn, Joseph F. Long, Lyle N. 2005. "Simulation of Helicopter Shipboard Launch and Recovery with Time-Accurate Airwakes." *Journal of Aircraft* 42 (2): 448-461
- Lorensen, William E. Cline, Harvey E. 1987. "Marching cubes: A high resolution 3D surface construction algorithm." *SIGGRAPH '87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques*: 163-169
- Lucy, Leon B. 1977. "A numerical approach to the testing of the fission hypothesis." *Astronomical Journal* 82 (12): 1013-1024
- Lürkens, Peter. 2015. "CumulusX! Soaring Environment Generator for Flight Simulator X." <ftp://4mboc5ivzaqqwcnb.myfritz.net/CumulusX/EN/Help.pdf> (accessed April 5, 2016)
- Marrone, Salvatore. Antuono, Matteo. Colagrossi, Andrea. Golicchio, Giuseppina. Le Touzé, David. Graziani, G. 2011. "δ-SPH model for simulating violent impact flows." *Computer Methods in Applied Mechanics and Engineering* 200 (13-16): 1526-1542
- Mathew, Joseph. 2010. "Large Eddy Simulation." *Defence Science Journal* 60 (6): 598-605
- Met Office. n.d. Beaufort wind force scale. <http://www.metoffice.gov.uk/guide/weather/marine/beaufort-scale> (accessed Mai 1, 2016)
- Microsoft. n.d. Weather Systems. <https://msdn.microsoft.com/en-us/library/cc526964.aspx> (accessed April 5, 2016)
- Monaghan, Joseph J., 2011. "A turbulence model for Smoothed Particle Hydrodynamics" *European Journal of Mechanics - B/Fluids* 30 (4): 360 - 370
- Müller, Matthias. Charypar, David. Gross, Markus. 2003. "Particle-based fluid simulation for interactive applications." *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*: 154-159
- National Oceanic and Atmospheric Administration. n.d. Wind Gust Definition. <http://graphical.weather.gov/definitions/defineWindGust.html> (accessed April 6, 2015)
- Olson, Jerry S. Watts, Julia A. Allison, Linda J. 1983. "Carbon in Live Vegetation of Major World Ecosystems." Environmental Science Division, Publication No. 1997.
- Pagen, Dennis. 1992. "Understanding the Sky." Spring Mills: Sport Aviation Publications.
- Pedley, Timothy J. 1997. "Introduction to Fluid Dynamics." *Scientia Marina* 61(1): 7-24

- Porté-Agel, Fernando. Wu, Yu-Ting. Lu, Hao. Conzemius, Robert J. 2011. "Large-eddy simulation of atmospheric boundary layer flow through wind turbines and wind farms." *Journal of Wind Engineering and Industrial Aerodynamics* 99 (4): 154-168
- Robinson, Andrew., Mania, Katerina., Perey Philippe. 2004. "Flight simulation: research challenges and user assessments of fidelity." *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*: 261-268
- Shah, Dhaval D. Menezes, Amor A. Kolmanovsky, Ilya V. 2012. "Glider flight environment modeling for optimal control." Paper presented at the American Control Conference, Montréal, Canada, June 27-29
- Solenthaler, Barbara. Pajarola, Renato B. 2008. "Density contrast SPH interfaces." *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*: 211-218
- von Funck, Wolfram. Weinkauff, Tino. Theisel, Holger. Seidel, Hans-Peter. 2008. "Smoke Surfaces: An Interactive Flow Visualization Technique Inspired by Real-World Flow Experiments." *IEEE Transactions on Visualization and Computer Graphics* 14 (6): 1396-1403