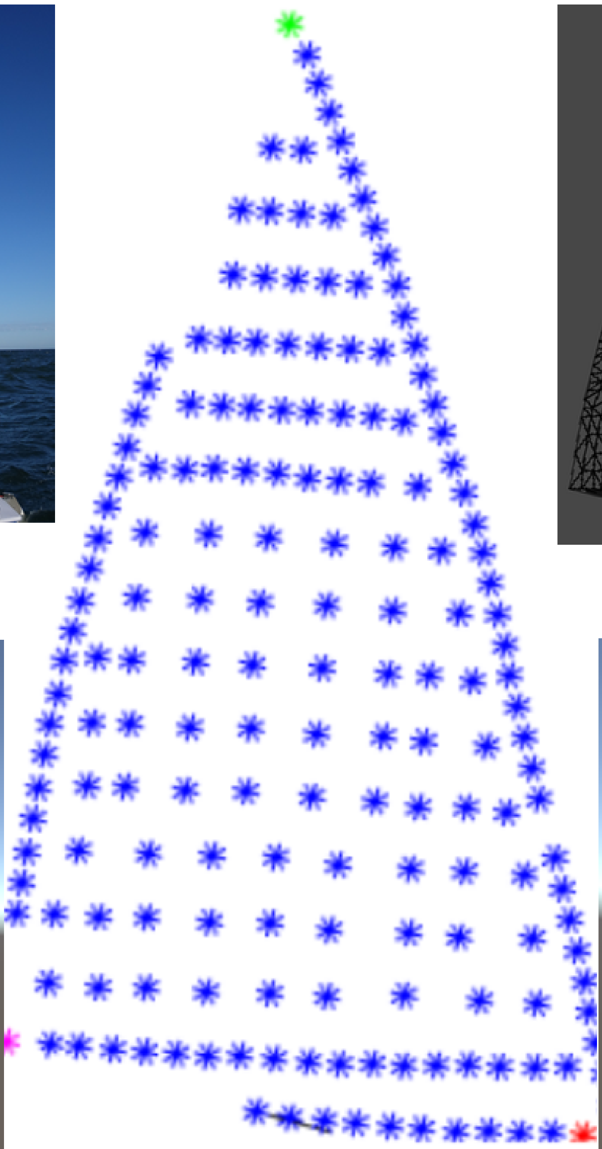
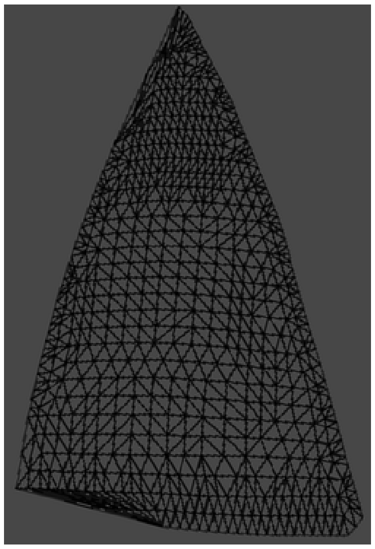
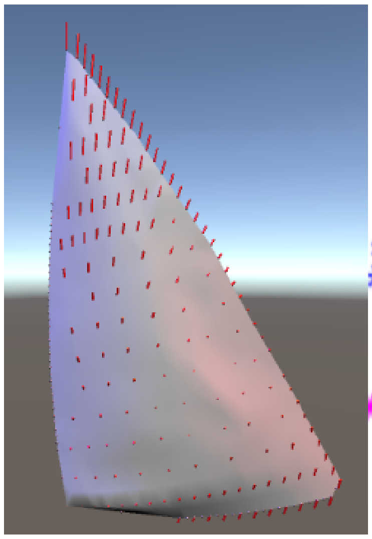
Physics-based sail reconstruction from 3D-markers

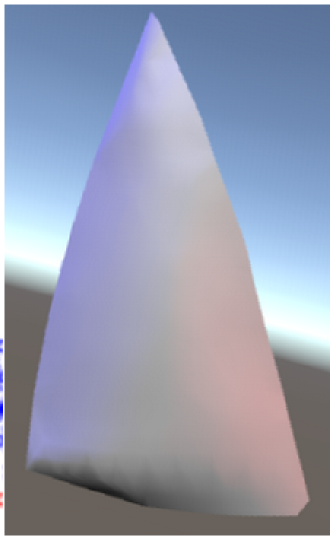
P.J.S. Kools

June 2022









Physics-based sail reconstruction from 3D-markers

by

P.J.S. Kools

Master thesis report to obtain the degree of
Master of Science
at the Delft University of Technology,
to be defended publicly on June 16, 2022 at 14:00.

Thesis committee

University supervisor:	Dr. K. Hildebrandt	EEMCS	TU Delft
University supervisor:	Dr. R. Marroquim	EEMCS	TU Delft
Responsible professor:	Prof. Dr. E. Eisemann	EEMCS	TU Delft
Committee member:	Dr. N. Tömen	EEMCS	TU Delft

Physics-based sail reconstruction from 3D-markers

P.J.S. Kools

June 2022

Abstract. To design more efficient sailing boat sails and to analyze the efficiency of a sail trim on the water, it is very helpful to have the ability to obtain a digital copy of real-life sail configurations. As a step towards obtaining such digital copies, the Sailing Innovation Centre [1] in collaboration with GeoDelta [2] has created point cloud measurements from sails by placing physical markers on them. This paper introduces a physics-based reconstruction method that uses a known sail rest shape combined with this point cloud data to reconstruct the sail configuration at the time that the measurement was taken. By using a physics-based sail model and the knowledge that the sail is subject to gravity and wind forces together with a known sail rest shape, we can create plausible reconstructions even for areas in the sail where fewer to no markers are placed. Using measurements that were taken before this reconstruction method was proposed in combination with an approximated sail rest shape and material properties, our reconstruction method was approximated to have an average reconstruction distance below 1cm for a 2.5m by 4.5m sail. Taking new measurements following our proposed guidelines in combination with having a more correct sail rest shape and better knowing the sail material properties is very likely to yield even better results.

Contents

1 Introduction **1**

2 Related work **2**
- 2.1 Shell models . . . 2
- 2.2 Mesh locking . . . 3
- 2.3 Sail simulation . . . 3
- 2.4 Sail reconstruction . . . 3

3 Physics-based sail simulator **4**
- 3.1 Discrete time integration . . . 4
  - 3.1.1 Integration method . . . 4
  - 3.1.2 Implicit integration using the Optimization Integrator . . . 5
  - 3.1.3 Minimization problem . . . 5
  - 3.1.4 Internal sail energies . . . 6
  - 3.1.5 Internal sail energy gradient and Hessian . . . 6
  - 3.1.6 External forces . . . 7
  - 3.1.7 Minimization . . . 9
  - 3.1.8 Static minimizer . . . 12
  - 3.1.9 Vertex movement constraints . . . 13

4 Reconstruction **13**
- 4.1 Snap vertices to measurement positions . . . 14
- 4.2 Wind adjustment . . . 14

5 Evaluation of the reconstruction method **17**
- 5.1 Artificial sail properties . . . 17
- 5.2 Shared test setup . . . 19
  - 5.2.1 Sail rest configuration . . . 19
  - 5.2.2 Marker configurations . . . 19
  - 5.2.3 Wind configurations . . . 20
  - 5.2.4 Reconstruction distance definition . . . 21
- 5.3 Sail simulator settings . . . 21
- 5.4 Test 1: Basic reconstruction . . . 21
  - 5.4.1 Results . . . 21
  - 5.4.2 Discussion . . . 21
- 5.5 Test 2: Mismatching membrane stiffness . . . 23
  - 5.5.1 Results . . . 23
  - 5.5.2 Discussion . . . 23
- 5.6 Test 3: Mismatching wind model #1 . . . 25
  - 5.6.1 Results . . . 25
  - 5.6.2 Discussion . . . 25
- 5.7 Test 4: Mismatching wind model #2 . . . 25
  - 5.7.1 Results . . . 27
  - 5.7.2 Discussion . . . 28

5.8 Test 5: Missing markers 29
5.8.1 Results 29
5.8.2 Discussion 29
5.9 Test 6: Standard deviation 31
5.9.1 Results 31
5.9.2 Discussion 31
5.10 Test 7: Real world data 33
5.10.1 Sail rest configuration 34
5.10.2 Measurements pre-processing 34
5.10.3 Marker selection 35
5.10.4 Reconstruction distance and marker selection 35
5.10.5 Test setup 35
5.10.6 Results 35
5.10.7 Discussion 35

6 Conclusion **38**

7 Future work **38**

1 Introduction

The Sailing Innovation Centre [1] aims at accelerating innovations in sailing. With an aim toward inventing more efficient sail shapes and evaluating real-life sail performance, it is very useful to have the ability to obtain a digital version of a real-life sail configuration on a sailing boat in real sailing conditions. As part of obtaining this data, the Sailing Innovation Centre in collaboration with GeoDelta [2] has placed special markers onto a sailing boat sail, and took images from this sail on the water. These images were then analyzed to obtain each of the marker positions in 3D space, which together form point cloud data as shown in Figure 1. Converting this point cloud data into a digital sail configuration is not trivial, however. Existing geometric approaches are generally not designed to respect the physics of the sail, thus overstretching areas and pulling areas with wrinkles flat as visualized in Figure 2. These effects become stronger with fewer markers on the sail. In this thesis, we propose a physics-based sail simulator and a point-based sail reconstruction method based on this simulator. Using the known shape of the sail in a rest configuration in combination with its material properties and the knowledge that measurements were taken under conditions with wind and gravity forces, we have a lot more knowledge to base our reconstruction on compared to geometric approaches. We will also provide insights into where markers should be placed on the sail to get relatively good reconstruction results.

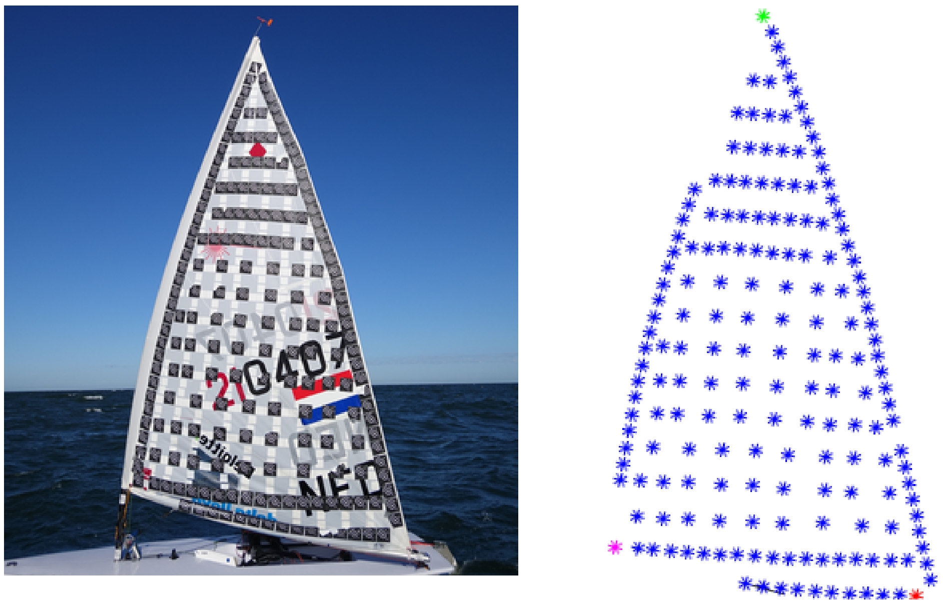


Figure 1: A sailing boat sail measurement as captured by the Sailing Innovation Centre in collaboration with GeoDelta, and the corresponding obtained point cloud data.

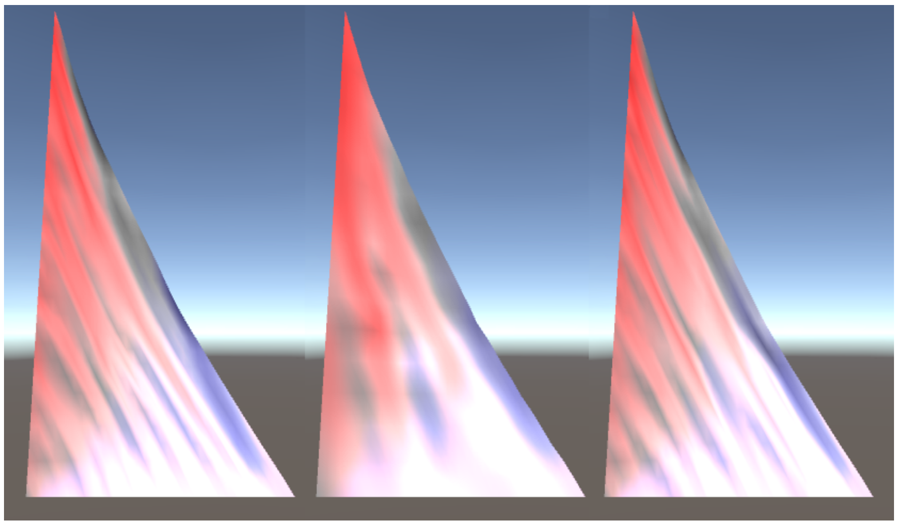


Figure 2: From left to right: An artificial sail configuration that was obtained through simulation, a possible reconstruction result of a geometric approach where many wrinkles have disappeared, and a reconstruction result using our reconstruction method where wrinkles are mostly maintained.

2 Related work

Sails fall under the category of shells, which are thin objects that are not necessarily flat and do not necessarily have a flat rest configuration. A lot of research has been done regarding shell simulation and shell reconstruction. This section adds a broader perspective on some of the techniques used in this paper and subjects that are related to this paper.

2.1 Shell models

Approaching 3D elastic solids from a Mechanical Engineering point of view, the Finite Elements Method is a very convenient 3D discretization where the object is divided into many deformable cubes on which textbook physics formulae can be applied directly to relate displacement, stresses, and forces based on the material properties of the object's material. While this method works well on most object shapes, issues arise when modelling very thin objects. To properly model the bending behavior of a thin object, the discretization should be such that plenty elements are used in the direction in which the object is thin. Using this size elements in the much larger length and width directions of the object would however result in a very large number of elements that, when discretely simulated, each only move a very small distance per time step. For simulations this can become very computationally intensive, or even impossible to simulate due to the small movements being below numerical precision. As a solution for this problem with shells, different alternatives to the Finite Elements Method have been proposed [3][4][5] where shells are approached as a 2D mesh of triangles. By performing this dimension reduction, the mesh triangle sizes are no longer strongly defined by the shell thickness, allowing for larger triangles to be used.

2.2 Mesh locking

Within shell simulators, mesh locking is a common problem. When mesh locking occurs, the configuration of the mesh triangles is such that a part of the mesh behaves much stiffer than expected. When external forces can not overcome this extra stiffness, it is possible that simulators get stuck in undesired configurations. This problem originates from approaching a curved surface with linear elements, and there have been several proposals [6][7] to overcome this problem. Since the sail meshes used in this paper have a sufficiently large mesh resolution with small rotations between triangles and only constraints on two sides, applied external forces can easily overcome increased stiffness and push the sail through potential locking configurations.

2.3 Sail simulation

There appear to not be many sailing simulators around. A sailing simulator game [8] exists with graphics that look plausible, but being closed-source, we don't know what kind of simulation or animation is being used. More promising work has been done in [9], where a finite-element shell model is coupled with a complex fluid model for wind-sail simulation. While this would have been an interesting basis for our reconstruction method, we chose to go with a simulator that has a much simpler wind model. This simpler wind model will allow us to more easily generate a desired wind force on the sail, which is essential for our reconstruction method as will be explained later.

2.4 Sail reconstruction

Interesting work has been done in [10], where a sailing boat sail is reconstructed based on a series of point measurements that were taken in rapid succession. The paper states that while its result was considered satisfactory by domain experts and experienced sailors, it was suggested that additional information about the sail bounds would entail more useful reconstructions for simulation and design evaluation purposes. Basing the reconstructions purely on positions and geometry makes this a very fast method, at the cost of losing local details about wrinkles in the sail and not necessarily respecting the original shape of the sail (unrealistic stretching or compression can be introduced). Depending on the desired information one would want from a sail reconstructor and the state of the recorded sail, this can be a convenient and fast choice.

3 Physics-based sail simulator

The core of our sail reconstruction method is a sail simulator that is purely based on sail material properties, energy, and forces. We will first explain how the simulator works, and then how the simulator is used in our reconstruction method. The sail is modelled as a discrete shell, conveniently approaching the sail as a deformable 2D shape made out of triangles.

3.1 Discrete time integration

To simulate a shell over time, we need a time integration method to solve the equations of motion for each of its vertices:
$\frac{dx}{dt} = v$, $\frac{dv}{dt} = a$ and $F = m * a$,
where x is the vertex position, v is the vertex velocity, a is the vertex acceleration, F is the force applied to the vertex, and m is the mass in a lumped mass model where the mass corresponds with a third of the sum of the mass of triangles that the vertex is part of. Since finding an exact solution through continuous integration for a shell with forces applied to it is a very complex problem, we approach time steps using discrete integration.

3.1.1 Integration method

The naive and easiest approach to taking discrete time steps is by performing explicit time integration on the vertex position:

$$x_{n+1} = x_n + \Delta t_n * v_n + \frac{1}{2} * \Delta t_n^2 * a_n,$$
$$v_{n+1} = v_n + \Delta t_n * a_n,$$

where x_n, v_n and a_n are the vertex position, velocity and acceleration at the beginning of the n^{th} time step, and Δt_n is the duration of the n^{th} time step. Note that only the forces on the shell vertices (and not their derivatives) are required for this method, making it the easiest approach to implement. Since the vertex velocity and acceleration for our application are highly non-linear over time, larger time steps will cause larger extrapolation errors up to the point where the simulation becomes unstable. This problem is shown in Figure 3. Choosing a very small time step will mathematically better approach the continuous integration solution, at the cost of having to perform more time steps. Too small steps should however be avoided, as using limited precision floating point numbers will round steps by precision. In our case, we tried decreasing the time step duration up to a point where the extrapolation errors would no longer prevent the simulation to converge, but as the time step duration became sufficiently small, the simulator started taking unfeasible time to reach a solution. The main cause for this problem is that small changes in vertex positions for very stiff materials result in very large internal shell forces. Since explicit time integration is not suitable for our problem, we used implicit integration instead.

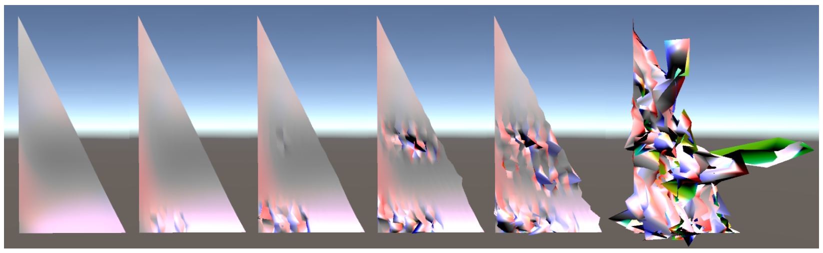


Figure 3: Explicit integration with too large time steps causing the simulation to no longer converge. Images are in chronological order from left to right.

3.1.2 Implicit integration using the Optimization Integrator

Implicit integration uses information from the end of the time step to reach a more robust solution for relatively large time steps compared to explicit integration. Since all forces involved in our shell simulator are very well expressible using gradients of energies, we based our time-stepping method on [11] that conveniently rewrites the root finding problem for force equilibrium to an energy minimization problem that is solved using the backward Euler implicit integration scheme.

3.1.3 Minimization problem

Following [11], a backward Euler discretization can be applied to rewrite the equations of motion to:

$$h(x_{n+1}) = M * \frac{x_{n+1} - x_n - \Delta t * v_n}{\Delta t^2} - f(x_{n+1}, \frac{x_{n+1} - x_n}{\Delta t}), (1)$$

where f are the vertex forces, and the goal for a single time step is to find a solution to $h(x_{n+1}) = 0$. Under the assumptions that we can express lower-bounded internal shell energies Φ_{in}, that we can integrate our external vertex forces to some energy function following $f_{ex} = -\frac{d\Phi_{ex}}{dx}$, and that the sum $\Phi = \Phi_{in} + \Phi_{ex}$ of these energies is bounded from below, we can rewrite (1) to:

$$h(x_{n+1}) = M * \frac{x_{n+1} - x_n - \Delta t * v_n}{\Delta t^2} + \frac{\delta \Phi}{dx}.(2)$$

Knowing that $h(x_{n+1}) = \nabla E(x_{n+1})$, we can integrate $h(x_{n+1})$ to get the energy equation:

$$E(x_{n+1}) = \frac{1}{2 * \Delta t^2} * (x_{n+1} - x_n - \Delta t * v_n)^T * M * (x_{n+1} - x_n - \Delta t * v_n) + \Phi.(3)$$

Under the assumption that a solution exists, solving $h(x_{n+1}) = 0$ is equivalent to minimizing $|h(x_{n+1})|$. This is not very convenient though, as solving this minimization problem using Newton's Method would require us to compute the Hessian of $h(x_{n+1})$, involving the second derivatives of our forces. Instead, we minimize the energy $E(x_{n+1})$, which is equivalent to approaching force equilibrium under the assumption that energies are lower-bounded. This gives us the advantage of only needing the energy, energy gradient, and energy Hessian, requiring only the first derivatives of our forces.

3.1.4 Internal sail energies

Using a triangle-based shell model for our sail where internal forces are modelled through energies, we need to define internal shell energies, their first derivatives (the energy gradient, equivalent to the negated shell forces), and their second derivatives (the energy Hessian) for each vertex in the mesh. A physically-motivated internal shell energy model is conveniently proposed in [3]:

$$W = W_M + W_B,$$
$$W_M = W_L + W_A,$$

where W is the internal sail energy composed of the membrane energy W_M and the bending energy W_B. The membrane energy is composed of the triangle edge length energy W_L that is mainly responsible for stretching resistance, and the triangle area energy W_A that is mainly responsible for shearing resistance. The triangle edge bending energy W_B is responsible for the bending stiffness. These energies are defined as:

$$W_L = k_L * \Sigma_e(1 - \frac{|e|}{|\bar{e}|})^2 * |\bar{e}|,$$

$$W_A = k_A * \Sigma_A(1 - \frac{|A|}{|\bar{A}|})^2 * |\bar{A}|,$$

$$W_B = k_B * \Sigma_e(\theta_e - \bar{\theta_e})^2 * \frac{|\bar{e}|}{\bar{h}_e},$$

where e is an edge, A is a triangle, $|e|$ is the length of an edge, $|A|$ is the area of a triangle, θ_e is the angle between the two triangles that edge e is part of, and h_e is a third of the average triangle height of the triangles $A1$ and $A2$ that edge e is part of (i.e. $h_e = \frac{|A1|+|A2|}{3*|e|}$). A $\overline{bar}$ indicates being in undeformed state. The k_L, k_A and k_B constants together with the shape of the mesh define the E-modulus, Poisson's ratio, and moment of inertia of the sail. As these constants combined with the mesh shape do not nicely map to the material properties, they will in practise be chosen based on a combination of visual behavior for more complex meshes, and computed based on expected strain or bend under load for simple meshes.

3.1.5 Internal sail energy gradient and Hessian

To differentiate the internal sail energy twice, [3] uses an automatic differentiation tool. Since this method is costly, we decided to derive most of the formulae ourselves. For the bending energy Hessian, we followed [12]. Note that their computation of edge normals $m_i = \hat{e}_i \times n$ as given in their algorithm 2 is not applicable for all edges i. The correct edge normals are:

$$m_0 = \hat{e}_0 \times n,$$

$$m_1 = \hat{e}_1 \times n,$$

$$m_2 = n \times \hat{e}_2,$$

$$\tilde{m_0} = \tilde{n} \times \hat{e}_0,$$

$$\tilde{m_1} = \tilde{n} \times \tilde{\hat{e}}_1,$$

$$\tilde{m_2} = \tilde{\hat{e}}_2 \times \tilde{n},$$

The implementation of the internal sail energy, its gradient, and its Hessian can be viewed in the ShellSimulator software [13] that was written as part of this thesis.

3.1.6 External forces

To make our simulator useful for sailing boat sails, we need to include gravity and wind forces into our simulator. To aid in convergence to a minimum energy state and to simulate deformation energy losses in the material, we also model a velocity-based damping force. Since the minimization problem from our simulator is energy-based, we need to integrate our forces and use the knowledge that $F_i = -\frac{\delta \Phi_i}{dx}$ to obtain the corresponding energies. Because we integrate the forces, an integration constant is introduced to indicate where the energy would be 0. For gravitational energy, we could define the energy to be zero on a defined floor level. For wind and damping forces, defining a zero-energy point is not trivial. Luckily, the minimization problem from our simulator only needs to compare the energy from one simulation step to the next. This means that we can work with work (i.e. energy change between simulation steps), rather than with absolute energy, and therefore that our integration constants can simply be 0. The external forces are defined as follows:

Gravity force
Remembering that we use a lumped mass model where triangle masses are distributed over their defining vertices, the gravity force applied to each vertex is

$$F_{gravity,i} = m_i * g,$$

where g is the gravitational acceleration vector. Following $F_{gravity,i} = -\frac{\delta \Phi_{gravity,i}}{dx}$, we can integrate and negate the gravity force to get the gravitational energy:

$$\Phi_{gravity,i} = -m_i * g * x_i,$$

where x_i is the position of vertex i.

Wind force
To model the wind, we chose a simple 3-DOF wind pressure projection model where the wind is described using a single wind pressure vector. The wind force is computed per triangle and then divided equally over the triangle defining vertices. The wind force per triangle is equal to the projection of the wind pressure vector onto the triangle normal. This means that a triangle area perpendicular to the wind pressure vector will yield the highest force, and a triangle area parallel to the wind pressure vector will yield no force at all. A visualization of the wind force on a sail can be seen in Figure 4. The force per triangle i is:

$$F_{wind,i} = windPressureVec \cdot n_i * A_i,$$

where n_i is the triangle normal vector and A_i is the area of triangle i. Distributing the triangle's wind force evenly over the 3 vertices defining that triangle, the force per vertex j caused by its surrounding triangles T is:

$$F_{wind,j} = \sum_{i \in T} \frac{F_{wind,i}}{3}.$$

Following $F_{wind,j} = -\frac{\delta \Phi_{wind,j}}{dx}$ and assuming that simulation steps generally cause only small changes in triangle areas and triangle area normals, we can approximate the wind energy with

$$\Phi_{wind,j} = -F_{wind,j} * x_j,$$

where x_j is the position of vertex j. Note that when this assumption does not hold, the wind might lag one time step behind. We will later discuss that we are only interested in the steady-state result of simulations, making this lag irrelevant as long as the simulation converges to a steady state.

Damping force
Performing a numeric simulation, rounding errors can introduce extra energy into the system, sometimes causing oscillations that prevent it to converge to a steady-state configuration. Adding a velocity or flexure based damping force will remove energy from the system, highly reducing these oscillations in the process. We chose to use velocity-based damping:

$$F_{damp,i}(n) = -k_{damp} * v_i(n+1),$$

where k_{damp} is the damping constant and v_i is the velocity of vertex i. Note that the damping force is dependent on the velocity at the end of the time step, ensuring that the damping force is in the opposite direction of the velocity, even when the velocity switches direction within a time step. Following $F_{damp,i} = -\frac{\delta \Phi_{damp,i}}{dx}$, we can integrate and negate the damping force to get the damping energy:

$$\Phi_{damp,i} = -F_{damp,i} * x_i.$$

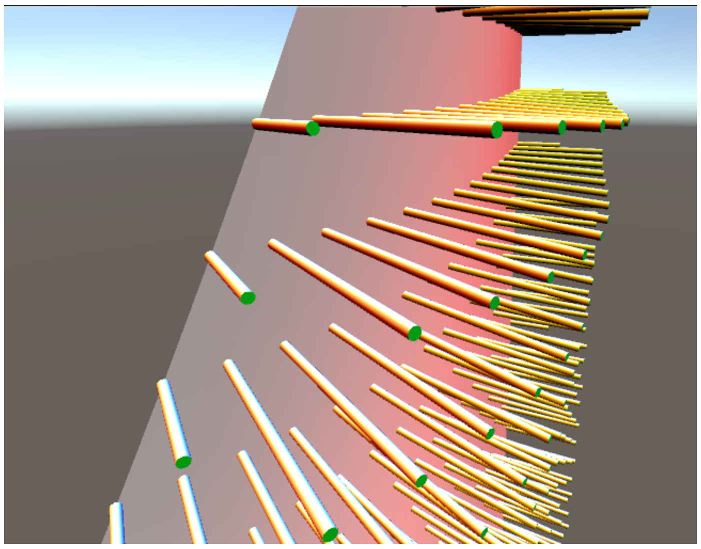


Figure 4: Wind force vectors acting on the vertices of a sail mesh.

3.1.7 Minimization

Having reduced performing a time step in our simulator to minimizing an energy equation, we will describe an algorithm for solving this minimization problem. This algorithm is a modified version of the unconstrained minimization algorithm as used in [11], where the most interesting differences are:

- We use a method from [14] to make the energy Hessian positive definite.
- We take at least χ Newton steps before accepting the solution for a time step.
- We added a lower bound to the line search α parameter to prevent an infinite loop in configurations where the computed step is within numeric precision.

Minimization algorithm
The minimization algorithm for a single time step starts with an initial guess $x^{(0)}$ for the vertex positions at the end of that time step. This initial guess is then modified in an iterative process to grow towards a solution that meets our termination criteria. While we could take the current vertex positions as an initial guess, we found that using the result from an explicit integration step generally gets us closer to the desired solution, decreasing time spent in the iterative process. This makes our initial guess:

$$x_{n+1}^{(0)} = x_n + v_n * \Delta t.$$

For the iterative process, we repeat the following steps:

1. Compute energy gradient ∇E and Hessian H of E at $x^{(i)}$.
2. Make energy Hessian H positive definite.
3. Terminate with solution $x^{(i)}$ if $|\nabla E| < \tau$ and $i >= \chi$.
4. Compute Newton step $\Delta x = -H^{-1} * \nabla E$.
5. Ensure that Δx is in downhill direction.
6. Limit the magnitude of Δx to L.
7. Perform line search to find a suitable positive step multiplier α.
8. Take step $x^{(i+1)} = x^{(i)} + \alpha * \Delta x$.

Positive definite energy Hessian
When computing the Newton step, the energy Hessian is used to perform larger or smaller steps where the change in energy does not change much or changes much respectively. This results in more suitable steps and faster convergence, causing a large runtime improvement over computing the Newton step without the energy Hessian, which is equal to performing gradient descent. Using the energy Hessian as a guideline for our Newton step, the simulator may get stuck on a bad solution at a plateau in the energy function where the energy Hessian simply starts multiplying steps by numbers close to zero. To get around this problem, we alter the result from these plateaus in the energy Hessian by changing the energy Hessian to be positive definite, essentially pretending that the energy function is convex without plateaus. We can do this because while changing the energy Hessian affects how the algorithm steps towards the target, the target itself remains unchanged. Using the definition that a matrix is positive definite if it is symmetric and all its eigenvalues are real and positive, making the (by definition symmetric and real eigenvalued) energy Hessian positive definite requires us to increase its eigenvalues until they become positive. The easiest and most naive approach is to add multiples of the identity matrix I to the energy Hessian until its eigenvalues become positive. This however has a very large effect on the entire energy Hessian and all its eigenvalues, even when it only has a few non-positive eigenvalues. Instead, we adopt a method from [14] where each eigenvalue of the energy Hessian is modified separately by performing the following steps:

1. Perform an eigenvalue decomposition of the energy Hessian H to get its eigenvalues λ_i and corresponding eigenvectors ν_i.
2. For each eigenvalue λ_i below λ_{min}, compute the amount k_i that we want to add to that eigenvalue:

$$k_i = \lambda_{min} - \lambda_i$$

3. Add the desired multiple k_i of the matrix product of the eigenvectors to H to increment its corresponding eigenvalue λ_i with that multiple k_i without affecting other eigenvalues of H:

$$H = H + k_i * (\nu_i^T * \nu_i)$$

Using this method, we only change the eigenvalues that are too low, leaving most of H unaffected and therefore keeping as much information to guide steps with as possible. Note that when we use $\lambda_{min} = 0$ or very close to zero, we could still be in the exact case that we were trying to avoid, where simulation can get stuck or very slow. So we use $\lambda_{min} = 0.1$ to pretend that the plateaus in the energy function are slightly tilted, causing the simulator to no longer get stuck there.

Termination criteria
The termination criteria consist of an energy gradient magnitude threshold τ in $|\nabla E| < \tau$, and a minimum number of iterations χ. The energy gradient magnitude threshold ensures that the algorithm keeps iterating until a sail configuration is found where the energy is sufficiently close to a steady-state. In some sail configurations, however, even very small changes to vertex positions cause large changes in energy, introducing the possibility to require an unfeasible amount of iterations to converge or not converge at all. We can deal with these configurations by increasing τ to a point where the algorithm accepts the sail configuration as being close enough to steady-state to terminate. Having such a high τ does mean that relatively bad steps would be accepted in configurations with a more smooth energy gradient. To prevent this from happening, we always require the algorithm to perform χ iterations, thus reaching a better solution in these easier cases. We found that $\chi = 5$ is generally sufficient to converge to an expected solution, and we used $\chi = 10$ for our experiments to be on the safe side with relatively good accuracy at the cost of a slightly longer simulation time. Note that in our simulator implementation [13], we have introduced a timeout as an additional termination criterion to prevent endless loops in the case that τ is too low for the solver. This merely functions as a safeguard and did not trigger while generating the data presented in this paper.

Newton step linear solver
Rewriting the formula for the Newton step

$$\Delta x = -H^{-1} * \nabla E$$

to

$$H * \Delta x = -\nabla E,$$

we can see that we have a linear system that needs to be solved. Any of the widely available linear solvers can be used to solve this problem. We used the ParDiSo [15][16][17] parallel direct solver for large sparse and symmetric linear systems, which provides an efficient solver that uses optimizations based on the fact that the energy Hessian H is both sparse and symmetric.

Downhill step direction with limited magnitude
Adopting the technique from [11], we consider step Δx to be suitable if $\Delta x * \nabla E < \kappa * |\Delta x| * |\nabla E|$, using $\kappa = 0.01$. If Δx is not suitable, then we try the same check with $-\Delta x$. If neither are suitable, then we fall back to a gradient descent step $-\nabla E$.

Line search
Now that we have a downhill step Δx, we can perform a line search to scale the step up or down to reach lower energy in our energy minimization problem. For the line search, we use the following procedure:

1. Compute energy E_i at the current configuration x_i.
2. Initialize α with its final value from the previous step's line search, or use $\alpha = 1$ if no previous step exists.
3. Compute E_{i+1} at configuration $x_i + \alpha * \Delta x$.
4.
    - If $E_{i+1} <= E_i$, set $\alpha = \alpha * 2$ and enter INCREASE mode.
    - Otherwise, set $\alpha = \frac{\alpha}{2}$ and enter DECREASE mode.
5. Recompute E_{i+1} at configuration $x_i + \alpha * \Delta x$.
6.
    - If $E_{i+1} <= E_i$ and we are in INCREASE mode, set $\alpha = \alpha * 2$ and continue at step 5.
    - If $E_{i+1} > E_i$ and we are in DECREASE mode, set $\alpha = \frac{\alpha}{2}$ and continue at step 5.
    - Otherwise, take step $x_{i+1} = x_i + \alpha_{best} * \Delta x$ with α_{best} from the best encountered (lowest valued) energy configuration E_{best} and terminate.

Being in a perfect steady-state configuration, step Δx will be of magnitude zero, and the algorithm will not terminate. This problem can simply be solved by skipping the line search algorithm for steps of magnitude 0. Another problem arises when α gets so small that the resulting step Δx is too small to affect the vertex positions x due to numerical precision. This can be prevented by defining a threshold value α_{min} which, when passed, selects the best-encountered step and terminates. We found $\alpha_{min} = 1e^{-12}$ to work well, with a side note that this problem rarely occurs.

3.1.8 Static minimizer

Now that we have a fully functional sail simulator, we can perform dynamic sail simulations. Keeping in mind that we will be using our simulator for reconstruction of a steady-state sail, we are merely interested in the steady-state result of a simulation. When we run a simulation where wind is blowing into a sail, we can see that dynamic behavior (i.e. inertia) causes the sail to oscillate for a long time before a steady state is reached. To speed up this process, we fully removed inertia and the earlier described velocity-based damping forces from our simulator, leaving us with a static minimizer. While this changes the path that the simulator takes towards a steady-state solution, the solution itself remains valid to our energy minimization problem. Note that we still use dynamic simulations to generate plausible artificial sail configurations for our tests. This prevents those sail configurations from being biased into a particular steady-state configuration when multiple steady-state configurations exist.

Removing inertia
Removing inertia can be done by removing all velocity-based terms in our minimization problem, essentially assuming that each step starts with zero velocity. Applying this to equation (2) leads to:

$$h(x_{n+1}) = M * \frac{x_{n+1} - x_n}{\Delta t^2} + \frac{\delta\Phi}{dx},$$

and applying this to equation (3) leads to:

$$E(x_{n+1}) = \frac{1}{2 * \Delta t^2} * (x_{n+1} - x_n)^T * M * (x_{n+1} - x_n) + \Phi.$$

Removing the damping force
Without inertia, the vertex position path and velocity towards the steady-state solution are no longer physically correct. This removes the sense of having a velocity-based damping force. Considering that in a steady-state sail configuration the velocity and therefore the damping forces are zero, we can simply remove the damping force from the minimization problem.

3.1.9 Vertex movement constraints

In a sail simulation with wind, it might not come as a surprise that we need to anchor the sail such that it does not just blow away. We do this by adding movement constraints on vertices, where for our application we only have to either not or fully restrict the movement of each individual vertex. We chose to implement this by storing whether a vertex is constrained or not, and simply not applying a simulation step to a constrained vertex (in both the Newton steps and the final time step result).

4 Reconstruction

Knowing the material properties, a rest configuration of the sail, and a point cloud of measured positions that correspond with vertices in the sail rest configuration, we define a reconstruction method that reconstructs the sail configuration at the time that that measurement was taken. The reconstruction method consists of an initial vertex snapping phase where vertices for which measurements exist are constrained at these measured positions, and a second phase with an iterative procedure where wind is adjusted to minimize the forces applied by the in the previous phase constrained vertices. Given that we are only interested in the steady-state result of our simulator, we use its static minimizer for all reconstructions. As explained in the simulator section, this highly reduces the time to converge to a steady-state solution.

4.1 Snap vertices to measurement positions

In this first reconstruction phase, we place each vertex for which a measurement exists onto its corresponding measurement position. Doing this instantaneous introduces unrealistically large membrane forces in the sail, which can cause a simulation step to result in a sail configuration where the sail goes through itself. In such a configuration, the bending forces are often not sufficiently large to be able to recover from this faulty configuration. To solve this problem, we alternate between moving the vertices towards their measurement positions with a constant distance and running a simulation step where these moved vertices are movement-constrained, where we found a move distance of $0.05m$ to work well. This works because each movement step causes a relatively small force on the sail, which is then decreased by a simulation step that moves surrounding vertices with the vertices that we are artificially moving. Once all vertices have reached their corresponding measurement positions, we continue with the next reconstruction phase; Wind adjustment.

4.2 Wind adjustment

Maintaining the movement constraints on the vertices that were moved to their corresponding measured positions in the last reconstruction step, we are left with a configuration where these constrained vertices are pulling other vertices in the mesh with artificially generated constraint forces. To create a realistic sail configuration from this, we need to reduce these artificial constraint forces as much as possible. We do this by changing the wind pressure vector from our earlier described wind model while simulating the sail so that the wind forces on average compensate for these artificial constraint forces. Since adjusting the wind will cause the mesh and therefore both the effect of the wind and the constraint forces to change when performing simulation steps, we re-adjust the wind at the start of each simulation step. The sail configuration to which this process converges is our reconstruction result. More details on these steps are described below.

Measurement constraint forces
Since constrained vertices by definition cannot move (i.e. $x_{n+1} = x_n$ and $v_n = 0$), we can rewrite (2) for constrained vertices to

$$h(x_{n+1}) = \frac{\delta \Phi}{dx}.$$

Combining this with $F = -\frac{d\Phi}{dx}$ and filling in the forces results in

$$h(x_{n+1}) = -F = -(F_{internal} + F_{gravity} + F_{wind} + F_{constraints}).$$

Filling in our simulator's target $h(x_{n+1}) = 0$, we get the force equilibrium equation

$$F_{internal} + F_{gravity} + F_{wind} + F_{constraints} = 0,$$

from which we can get our constraint forces

$$F_{constraints} = -F_{internal} - F_{gravity} - F_{wind}.$$

Wind pressure vector computation

Seeing that our goal is to fully replace the artificial constraint forces with wind forces for vertices i with measurement positions, we can fill in $F_{constraints} = 0$ in the force equilibrium equation to find the desired wind force

$$F_{wind,i} = -F_{internal,i} - F_{gravity,i}.$$

Having the desired wind force, we need to adjust the wind pressure vector to create this wind force. Since our wind model only comes with three parameters (the wind pressure vector) and we are trying to create a specific wind force vector at each constrained vertex, this will rarely be a perfect fit. What we can do however, is to apply the on average desired amount of force in the on average desired direction. We do this by first computing the wind force sum direction vector

$$\hat{F}_{wind} = unit(\sum_i F_{wind,i}),$$

followed by computing the effect of applying a unit wind pressure vector $\hat{P}$ in this direction

$$\hat{P} = \hat{F}_{wind},$$

$$F_{wind,unit} = \frac{(\hat{P} \cdot triangleNormal) * triangleArea}{3} * triangleNormal,$$

and finally scaling the wind pressure vector such that on average the desired wind force is generated on the vertices for which measurement positions are available

$$F_{wind,unit,avg} = \frac{F_{wind,unit}}{numVertices},$$

$$F_{wind,desired,avg} = \frac{\sum_i F_{wind,i}}{i},$$

$$P = \hat{P} * \frac{F_{wind,desired,avg}}{F_{wind,unit,avg}}.$$

These forces are visualized in Figure 5, where we can see that the wind force caused by this wind pressure vector reasonably well compensates for the desired wind forces. We now have a wind pressure vector to perform our next simulation step with, which will blow wind in a direction in which the earlier discussed constraint forces will decrease.

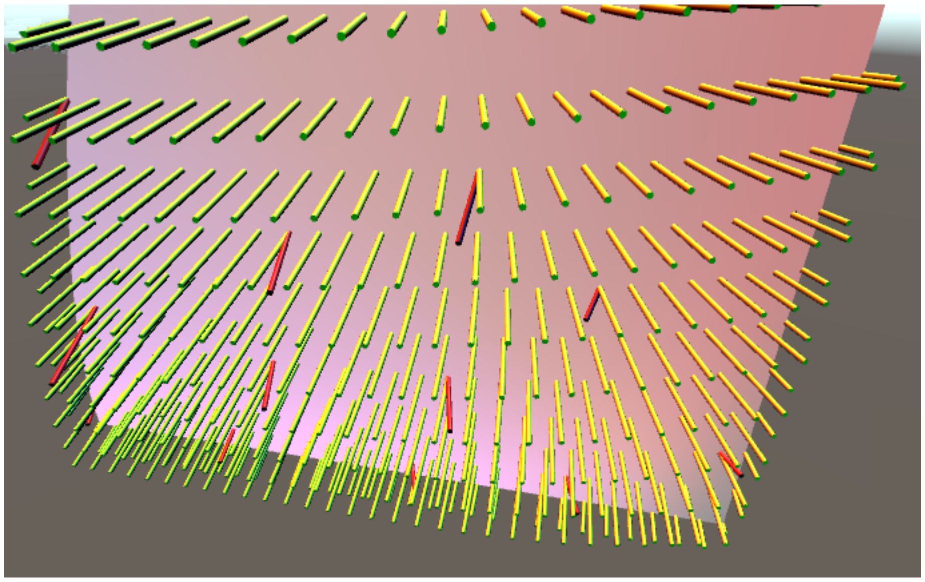


Figure 5: The desired wind forces to compensate for measurement vertex constraint forces in red, and the resulting wind force on each vertex from our wind model where the wind pressure vector was computed based on these desired wind forces in yellow.

Delta wind pressure vector

Applying the explained method, we find the reconstructor nicely converging towards a solution and then oscillating around that solution. These oscillations are mainly caused by the wind forces being updated before each simulation step and then remaining constant for that entire step. The two obvious candidates for this problem are performing smaller steps and adding damping to the system. Since performing smaller steps highly influences the runtime of our reconstructions, we chose to add damping by limiting the magnitude change in the wind pressure vector per iteration instead. Making this delta wind pressure magnitude limit very large causes the sail to move quickly towards the reconstruction result and then oscillate around that solution, and making this limit very small decreases the oscillations to an acceptable minimum at the cost of converging only very slowly towards the reconstruction result. To get the best of both cases, we used a threshold of 50 and linearly decreased it over a duration of 100 simulation steps towards a threshold of 1, after which we continue with a threshold of 1. This at first allows large wind changes, slowly decreasing when approaching the simulation. Note that suitable values are dependent on the mesh size, which in our case was 561 vertices.

Termination criterion
Finding a termination criterion was tough. For example setting a step magnitude threshold either triggered too early when it was too large, or never when it was too small. Using the magnitude sum of the constraint forces that we are trying to minimize sometimes triggered too early when the sail would go through the measurement positions in an oscillation, having vertices without a measurement position still not be close to force equilibrium. Taking a step back after similar attempts, we decided to go with a fixed amount of simulation steps for the wind adjustment reconstruction phase. Using 100 steps to decrease the delta wind pressure magnitude threshold from 50 to 1, we found the sail to be sufficiently steady after at most another 80 simulation steps, so we did 200 steps in total to be on the safe side. Note that suitable values are dependent on the mesh size, which in our case was 561 vertices. The sail configuration at the end of this process is our reconstruction result.

5 Evaluation of the reconstruction method

To evaluate the reconstruction method, we performed reconstructions on both artificial and real-world data. To create reconstruction problems in a controlled environment, we used an artificially generated sail together with our simulator to generate plausible sail configurations from which we then took measurements. Simulating this sail sufficiently long without external forces applied to it results in a rest configuration of the artificial sail, which is passed to the reconstructor together with the taken measurements to obtain reconstructions. These reconstructions can then be compared to the artificial sail configuration that we know the measurements were taken from to evaluate the performance of the reconstruction method. To evaluate whether the reconstruction method is also suitable outside of the controlled artificial environment, we performed several reconstructions based on measurements that were taken from a real-world sailing boat. We will start with the artificial reconstructions, and end with a section about the real world reconstructions.

5.1 Artificial sail properties

The artificial sail that is used to evaluate our reconstruction method is a triangle that is 3.5 meters wide and 7 meters high, being in the range of common medium-sized sailing boats such as the Polyvalk. This triangle is subdivided 5 times using the four-to-one-split method, resulting in a mesh with 561 vertices, 1584 edges, and 1024 triangles. The vertices along the mast and the boom are then movement-constrained as shown in Figure 6, and the inner edge lengths are multiplied by 1.01 to obtain the slight bulge that most sails have. The sail is set to have a flat rest configuration, meaning that the internal bending forces always push towards a flat configuration. For the sail thickness and material density, we picked a rough median of the materials tested in [18], leading to a thickness of $0.25mm$, a weight of $0.23kg/m^2$, and therefore a density of $\frac{1m}{0.00025m} * 0.23kg/m^2 = 920kg/m^3$ (note that the units seem wrong, but that the weight per area actually represents a weight per volume instead, where the height is the sail thickness). To select the length and area energy constants for our material, we first obtained the rough median E-modulus of $E = 130GPa$ and the corresponding maximum strain of $\epsilon = 6\%$ from this same paper [18]. Using Hooke's Law $\sigma = E * \epsilon$, we can obtain the expected stress at the maximum strain. Creating a subdivided square mesh and constraining its top and bottom vertices 106% of the original square height apart gives us a configuration where the maximum strain is applied. When we run simulation steps until the mesh reaches a steady-state, the constraint forces acting on the top and bottom vertices will be close to equal due to force equilibrium, and the sum of forces on the top divided by the top area should correspond with the known stress at this strain. We modified the length and area constants to $k_{length} = 15621$ and $k_{area} = 8125$ to match this desired stress. Lastly, we chose our bending constant by simulating a flat subdivided rectangle with gravity where two rows of vertices on a short side were constrained to artificially clamp it in place, as shown in Figure 7. We found that a bending constant of $k_{bend} = 10$ was visually realistic for sail material.

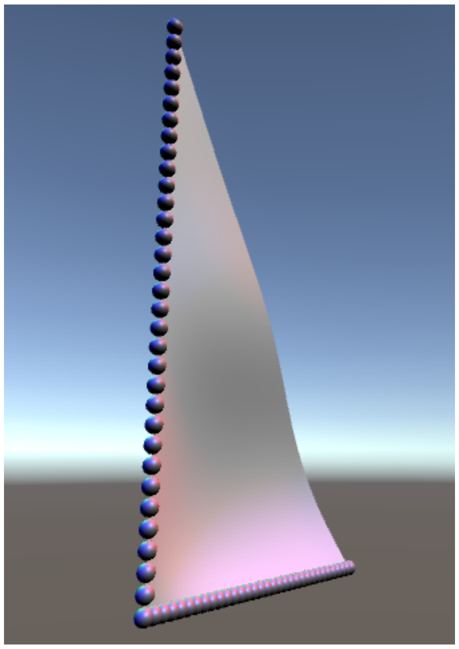


Figure 6: Artificial sail rest configuration with constraints put on the mast and boom vertices.

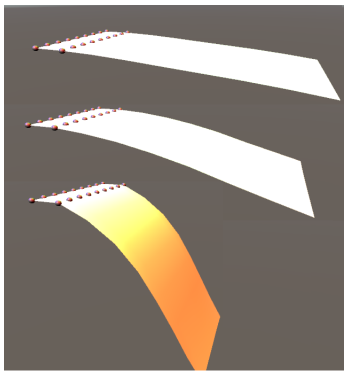


Figure 7: Example of a testing setup where pieces of sailcloth with different bending energy constants are simulated with gravity to be able to visually select a suitable bending constant.

5.2 Shared test setup

Since most tests share a part of their setup, we use this section to explain the basic setup. This applies to all tests unless specified otherwise.

5.2.1 Sail rest configuration

Each reconstruction starts with a rest configuration of a sail. For our tests, this configuration is obtained by simulating the artificial sail without external forces until a steady-state configuration is reached. This is our rest configuration, as shown in Figure 6. Note that with the perfectly flat artificial sail, we do need to introduce a slight disturbance to pop it out of its plane. This can be done by either moving any of its non-constrained vertices out of the plane or by applying a wind force to the sail for as little as one simulation step.

5.2.2 Marker configurations

In the real world, markers placed onto the sail define which points on the sail will be measured. On the artificial sail, markers are added using farthest-first vertex traversal, starting at an arbitrary vertex (which in our case is the vertex in the middle of the diagonal edge of the sail) as shown in Figure 8. Since constrained vertices at the mast and boom contribute no extra measurement information, these vertices are not considered in the vertex traversal. Using this method, we ensure that configurations with less markers are always a subset of configurations with more markers. This is a valuable property when comparing reconstructions where only the amount of markers is varied, as configurations with less markers can never have more information available than configurations with more markers (e.g. placing 100 markers around the constrained edges would likely perform worse than placing 10 markers evenly distributed over the sail).

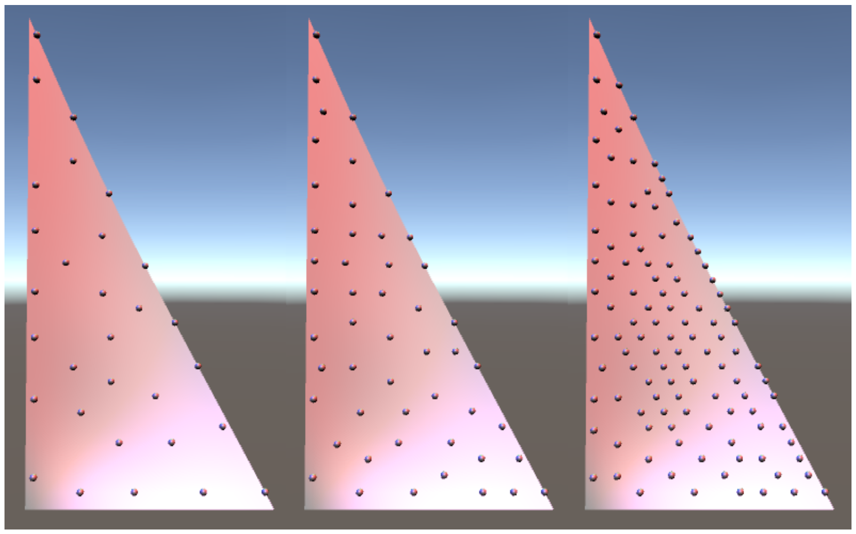


Figure 8: Farthest-first marker placement on the artificial sail rest shape for 30, 50 and 110 markers.

5.2.3 Wind configurations

Using a wind pressure vector P as input for our wind model, we can vary the wind magnitude and direction. To compute the wind pressure magnitude $|P|$, we use the drag coefficient formula

$$c_{drag} = \frac{2 * F_{drag}}{\rho * u^2 * A},$$

where c_{drag} is the drag coefficient, F_{drag} is the drag force, u is the flow speed, ρ is the medium's density, and A is the surface area on which the the flow is applied. Since we want the wind pressure, we can fill in $|P| = \frac{F}{A}$ and rewrite to

$$|P| = \frac{F_{drag}}{A} = \frac{c_{drag} * \rho * u^2}{2}.$$

We can now fill in $c_{drag} = 1.17$ for blowing against a flat surface, and $\rho = 1.2kg/m^3$ for air at sea level to obtain

$$|P| = \frac{1.17 * 1.2 * u^2}{2} = 0.7020 * u^2.$$

For our tests, we use a wind velocity within $3Bft$ and $5Bft$, being $u = 9.6m/s$ and $u = 38.4m/s$ respectively. This corresponds with a calm average day, and a more exciting day for sailing. Filling in these velocities in our equation leads to $|P_{3Bft}| = 64.6963N/m^2$ and $|P_{5Bft}| = 1035.1N/m^2$.
For the wind direction, we chose to use an angle of 30, 45 and 60 degrees from the boom. These angles cover the desired sail positions in most courses for average sailing boats.

5.2.4 Reconstruction distance definition

Many tests results will contain an average and maximum reconstruction distance. The total reconstruction distance is defined as the sum of distances between all vertices and their corresponding positions in the mesh from which we sampled the measurements. The average reconstruction distance is this total distance divided by the amount of vertices, and the maximum reconstruction distance is the vertex-wise largest distance.

5.3 Sail simulator settings

Throughout the explanation of our simulator, well-performing settings were mentioned. These settings are used for all tests unless specified otherwise.

5.4 Test 1: Basic reconstruction

To get an indication of how accurate our reconstructor can be for different amounts of markers under ideal synthetic conditions, we use our simulator with the sail rest configuration to generate steady-state sail configurations for different wind configurations. We then sample measurements from these sail configurations. Using the sail rest configuration in combination with these measurements, we reconstruct each sail and compare this to the sail configuration from which we took the measurement. Since we use the same simulator parameters, sail material properties, wind model, and sail model for both the generation of the measurement data and the reconstruction, we can expect relatively good results compared to the real world. The results from this test will give us a baseline to compare other test results to where we introduce parameter and model errors that we could expect in the real world.

5.4.1 Results

The results are shown in Figure 9 and Figure 10.

5.4.2 Discussion

Looking at both figures, we can see that the reconstruction result heavily improves up to around 50 markers. Between 50 and 130 markers, the reconstructor performs very well with average reconstruction distance of approximately 0.1mm to 0.5mm and a max reconstruction distance of approximately 0.9mm to 4mm. On a 3.5m by 7m sail, these ranges are very good. When choosing over 130 markers however, we see that some results are getting worse. This seems to be caused by vertices that are close to completely surrounded by markers getting locked into a bulge in the wrong direction. The marker vertices cannot move, and the wind in these situations does not blow hard enough to overcome the membrane forces and snap the locked vertex into place. The solution to this problem would be to increase the mesh size, introducing extra flexibility to change into the other bulge in these areas. We did however stick to testing with

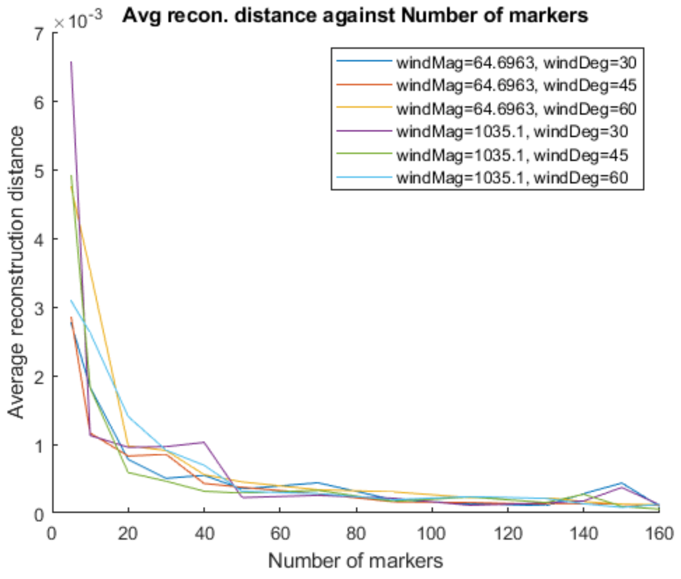


Figure 9: The average reconstruction distance against the number of measurements for 6 different wind configurations.

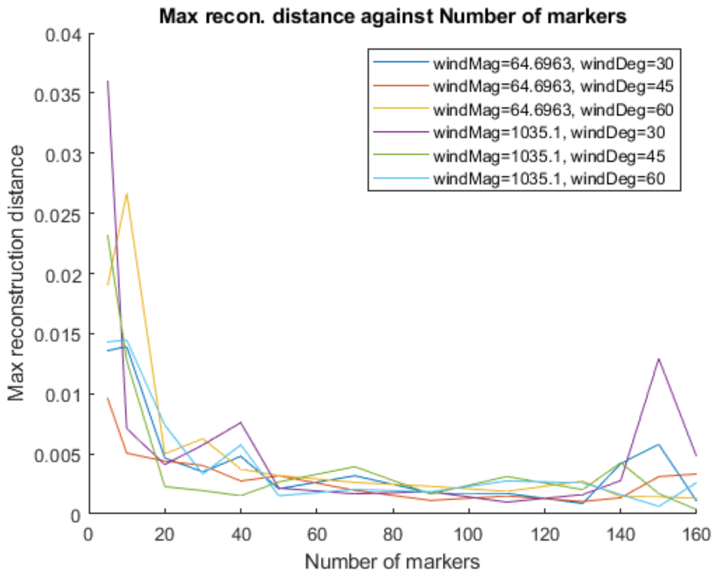


Figure 10: The maximum reconstruction distance against the number of measurements for 6 different wind configurations.

561 vertices, as runtime of reconstructions can get very large for larger meshes. Looking at the different wind configuration results, there does not seem to be a significant difference in the reconstruction error. We do have to keep in mind that these results were obtained under ideal conditions, thus being relatively good compared to real-world data reconstructions, but this does show that the reconstructor is functioning as intended, and it sets a baseline for following tests to compare results with.

5.5 Test 2: Mismatching membrane stiffness

In a real-world application, the material properties of the material used might not always be known. Sail thickness and density are easy to measure precisely, but selecting the membrane energy constants based on an approximated E-modulus is much less precise. In this test, we introduce an error in these membrane energy constants by multiplying them with the same varied factor before starting the reconstruction. This allows us to get an indication of how mismatching membrane energy constants will affect the reconstruction result, which can be used to select a range in which these material properties have to be approximated in order to still yield acceptable reconstruction results. We run these reconstructions for the wind configuration where $|P| = 1035.1N/m^2$ at an angle of 45 degrees from the boom.

5.5.1 Results

The results are shown in Figure 11 and Figure 12.

5.5.2 Discussion

Looking at the results, we can see that results around a stiffness factor of 1 (i.e. 100%) clearly perform best. This means that getting the membrane energy constants right does lead to better reconstruction results. We can also see that reconstructions with more markers performed better, and are influenced less by the stiffness factor compared to reconstructions with fewer markers. Looking specifically at Figure 11, we can see that the average reconstruction error for $n = 110$ is even slightly decreasing below a stiffness factor of 1. This is likely being caused by a combination of the standard deviation, the stiffness to have a relatively low effect on the reconstruction result when having a lot of markers, and this marker configuration covering relatively many vertices, thus defining the bulge in the sail pretty well already. Overall we can see that with the 50 marker configuration, choosing a membrane stiffness that is within 30% of the actual stiffness yields results that are very similar to choosing it exactly right. For the 110 marker configuration, we can even go from half to twice the stiffness for similar results. In cases where the expected error is larger, it is advised to add more markers in order to decrease the effect of this error.

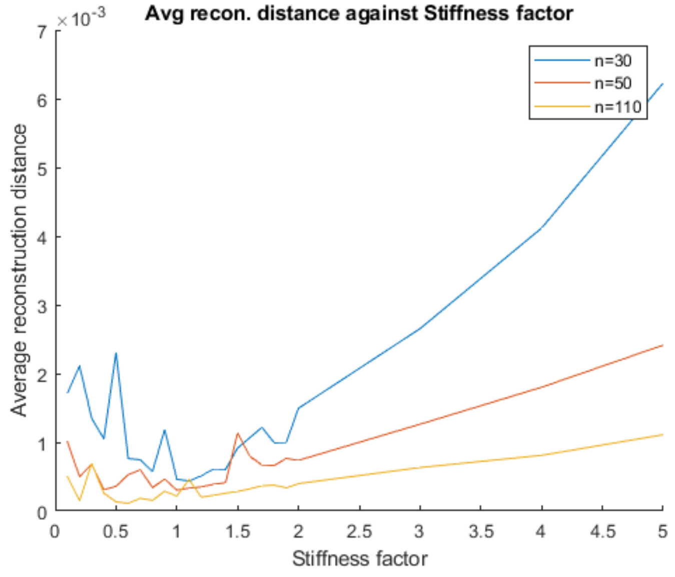


Figure 11: The average reconstruction distance against the stiffness factor for reconstructions with 30, 50, and 110 markers.

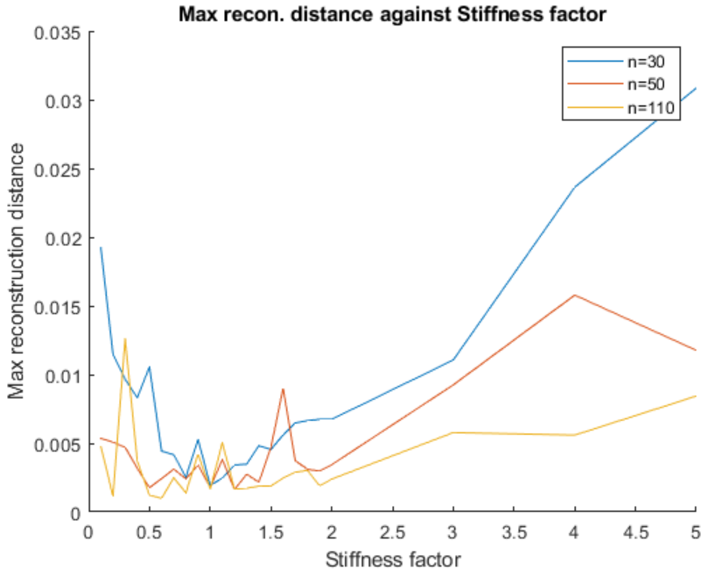


Figure 12: The maximum reconstruction distance against the stiffness factor for reconstructions with 30, 50, and 110 markers.

5.6 Test 3: Mismatching wind model #1

Our wind model is very simplistic, projecting a wind pressure vector on triangle normals to obtain the pressure magnitude that it applies to the triangle surface in the triangle normal direction. In the real world, the air that is blown into the sail builds up and partially forms a static pressure, rather than this directed pressure that our model is using. In this test, we run our usual reconstructions on measurements that were generated using an alternative wind model. In this alternative wind model, we set a fraction of the wind pressure vector that will be applied as direct pressure in the triangle normal direction, rather than being projected onto triangle normals. This results in

$$F_{wind,proj} = \frac{(P \cdot triangleNormal) * triangleArea}{3} * triangleNormal,$$

$$F_{wind,pres} = \frac{|P| * triangleArea}{3} * triangleNormal,$$

$$F_{wind} = a * F_{wind,pres} + (1 - a) * F_{wind,proj},$$

where $F_{wind,proj}$ is our usual projected wind force, $F_{wind,pres}$ is the directly applied wind pressure, and $a \in (0, 1)$ is the wind pressure factor. Reconstructing a sail configuration that was generated with a different wind model allows us to get an indication of how a mismatch between these wind models affects reconstruction results. We run these reconstructions for the wind configuration where $|P| = 1035.1N/m^2$ at an angle of 45 degrees from the boom.

5.6.1 Results

The results are shown in Figure 13 and Figure 14.

5.6.2 Discussion

Looking at the results, we interestingly do not see a clear trend in how the wind model mismatch affects the reconstruction result. We can see that the $n = 30$ case has some peaks, but these fall within the standard deviation for the case with only 30 markers. It seems that our wind pressure vector can quite well mimic the direct wind pressure model, which makes sense if we consider that small wind force errors are being compensated with marker vertex constraint forces everywhere where markers are added, thus not allowing these forces to build up and cause the bulge in the sail to change significantly. Overall this means that our simple wind model is flexible enough to handle this case of mismatching wind without noticeable effect, which is the most ideal result that we could get.

5.7 Test 4: Mismatching wind model #2

Just like the previous test, we generate measurements from sail configurations that were obtained using an alternative wind model that mimics an extra aspect

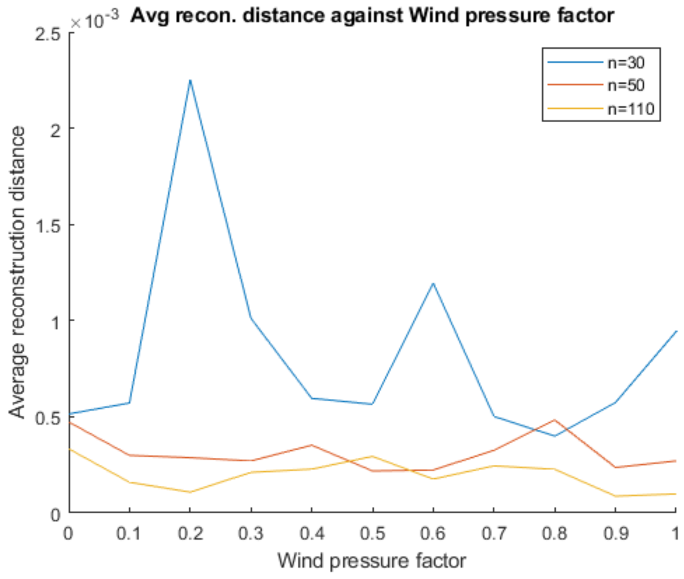


Figure 13: The average reconstruction distance against the wind pressure factor for reconstructions with 30, 50, and 110 markers.

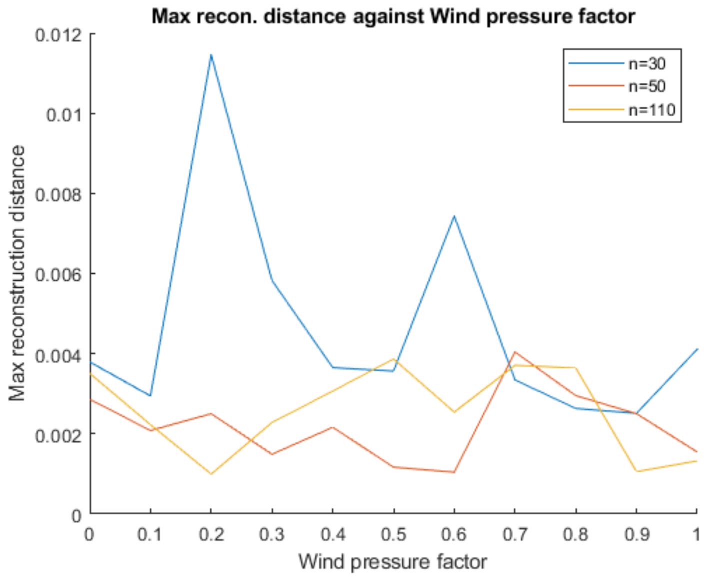


Figure 14: The maximum reconstruction distance against the wind pressure factor for reconstructions with 30, 50, and 110 markers.

that we know exists in real-world wind behavior. In this case, we mimic the wind pressure in the back of the sail being higher near the mast and lower at the end of the boom. This is caused by air being blown into the back of the sail due to the speed of the boat. We approximate this effect by applying the following linear wind force factor to the wind force for each vertex based on its coordinate along the boom

$$forceFactor = 1 + a * (\frac{x}{c} - 1),$$

where x is the position along the boom with its origin at the mast-boom intersection, c is half the sail width (i.e. half the boom length), and $a \in (0, 1)$ defines the force slope. We vary this force slope from 0 to 1 to go linearly from our usual wind model to the case where the front of the sail receives no wind force anymore, and the back of the sail (at the end of the boom) receives double the wind force. This allows us to get an indication of how a mismatch between these wind models affects the reconstruction results. We run these reconstructions for the wind configuration where $|P| = 1035.1N/m^2$ at an angle of 45 degrees from the boom.

5.7.1 Results

The results are shown in Figure 15 and Figure 16.

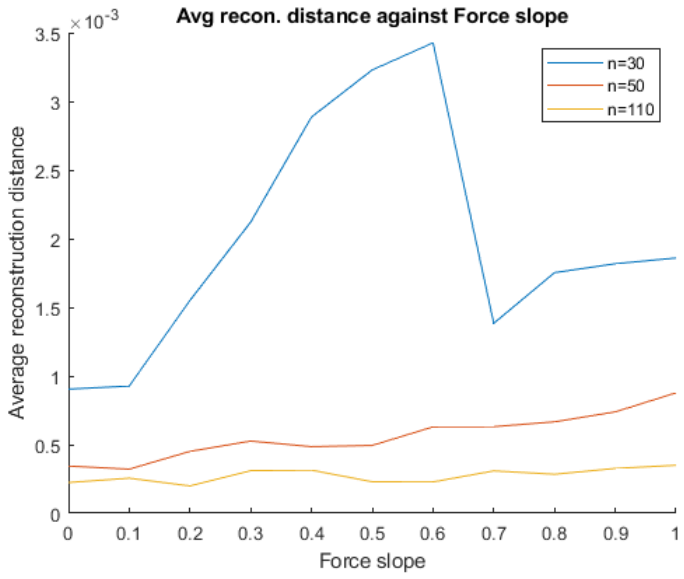


Figure 15: The average reconstruction distance against the force slope for reconstructions with 30, 50, and 110 markers.

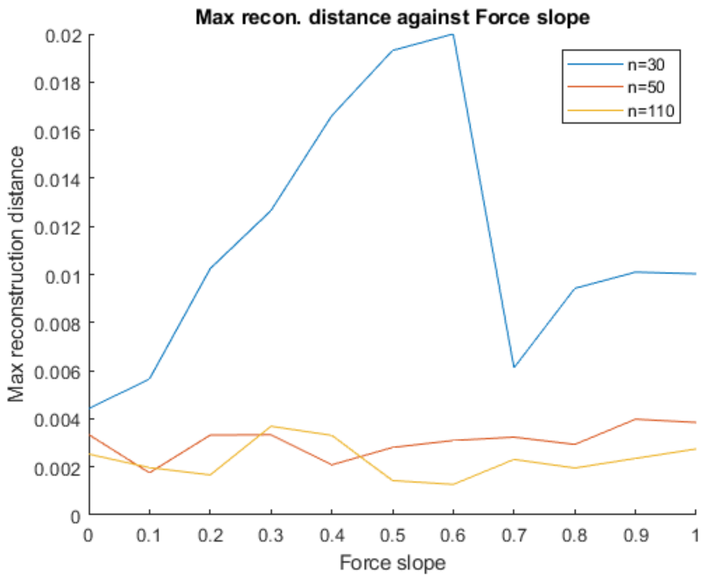


Figure 16: The maximum reconstruction distance against the force slope for reconstructions with 30, 50, and 110 markers.

5.7.2 Discussion

Looking at both the average and the maximum reconstruction distance in the results, we can see that in the $n = 30$ case the reconstruction error is greatly influenced by the mismatch with the alternative wind model. We see a factor of 3.5 in average reconstruction distance and a factor of 4 in maximum reconstruction distance for this marker configuration. We can also see this rate decreasing at a high force slope rate. The standard deviation can account for approximately a 1mm difference in the average reconstruction error, but this does not account for the range from 3.5mm to 2mm that we see in the results. A possible explanation could be that with a higher force factor, wind forces in the front of the sail are close to zero, allowing the vertices close to the mast to be relatively free and get locked into the wrong direction bulge in the sail. This bulge will not be reconstructed in this direction unless a marker exists within this bulge, pulling it in a direction that is not logical from our wind model's perspective. This effect will be larger for reconstructions with fewer markers, hence why it could explain a jump in our 30 marker reconstruction results. Looking at the cases with 50 and 110 markers, we can see a slight but clear upward trend in the average reconstruction error which's slope seems to depend on the number of markers. For 50 markers, the average reconstruction error increases from approximately 0.4mm to 0.9mm. For 110 markers, this is approximately 0.2mm to 0.3mm. This does mean that we can expect this effect to have quite an impact on the reconstruction result if too few markers are used. While this is unfortunate, we do see that the 110 marker case still performs very well in the worst case at approximately 0.3mm average reconstruction distance and 3mm maximum reconstruction distance. When looking at the maximum reconstruction distance, we don't see much change for 50 and 110 markers. There does not seem to be a trend here.

5.8 Test 5: Missing markers

Taking measurements of a sail in the water can be challenging. Especially on small boats, the measuring equipment is likely to be on another boat. This means that the sailor or a part of the boat can get between the camera and the sail, thus causing certain markers not to be measured. To get an indication of how this affects the reconstruction result, we run reconstructions using measurements from which we remove sections of markers. Instead of just doing this for sections where the sailor is likely to be in front of, we also remove markers in certain other sections. This allows us to get an indication of which sections are more important for the reconstructor, and which sections might not matter much at all. When interpreting the results from this, we should keep in mind that these reconstructions are still happening in the ideal synthetic world. This means that even when certain sections do not contribute much in this test, they might contribute more in the real world. If removing a section significantly worsens the reconstruction result though, we can expect this to also happen in the real world. To remove markers from measurements, we define spheres in which all measurements are removed. These spheres are defined by their position and radius. Ignore sphere configurations were chosen based on where we think might be interesting zones in the sail. The used ignore sphere configurations are shown in Figure 17. We run these reconstructions for the wind configuration where $|P| = 1035.1 N/m^2$ at an angle of 45 degrees from the boom.

5.8.1 Results

The results are shown in Figure 18 and Figure 19.

5.8.2 Discussion

Looking at the results, ignoring markers in the bottom corners (cases 1 and 2) does not seem to have a significant impact on the reconstruction result, given that enough markers are present. Case 3 where all markers from the top of the sail are removed does however have a very large impact on the reconstruction result. In this case, the reconstructor gets nearly no feedback about the top of the bulge in the sail, adjusting the wind to a direction that comes much more from above than the direction used to generate configuration that is being reconstructed. From this we can conclude that having markers in multiple places around the bulge highly benefits the reconstruction result. Looking further, we can see that removing many markers from the middle of the sail (cases 4 and 5) also has a relatively large negative impact on the reconstruction result.

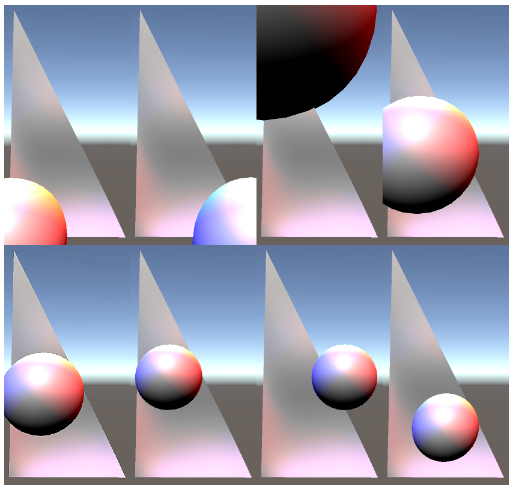


Figure 17: Ignore sphere configurations 1 to 8 for all missing markers test cases visualized on the sail, counted from left to right and top to bottom. The ignore sphere origin coordinates have their origin at the mast-boom intersection, their x-axis in the direction of the boom, and their y-axis in the direction of the mast.

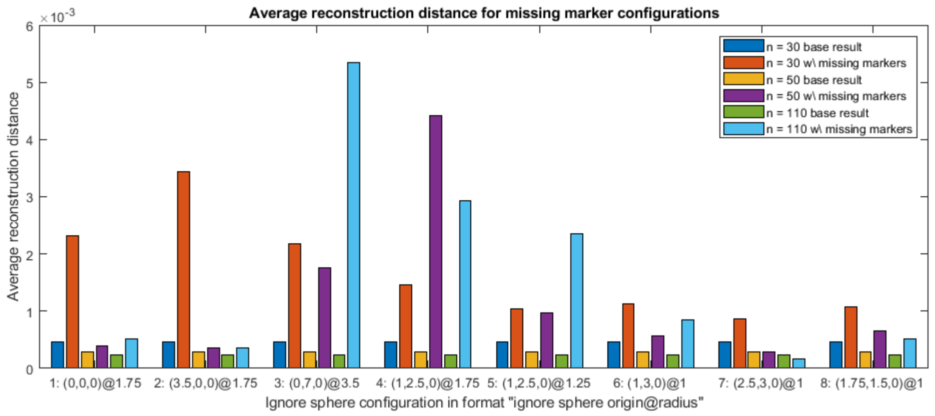


Figure 18: The average reconstruction distance for configurations where markers within spheres have been removed for reconstructions with 30, 50, and 110 markers.

This is an expected result from which we can conclude that having markers in the middle of the sail is more important than having markers at the edges (keeping in mind that markers should be spread around the bulge). Going from case 5 to case 6, more markers are added in the bulge and the reconstruction result improves slightly. Looking at cases 6 and 8, we can see that with this ignore sphere radius the reconstruction results are on average about half as

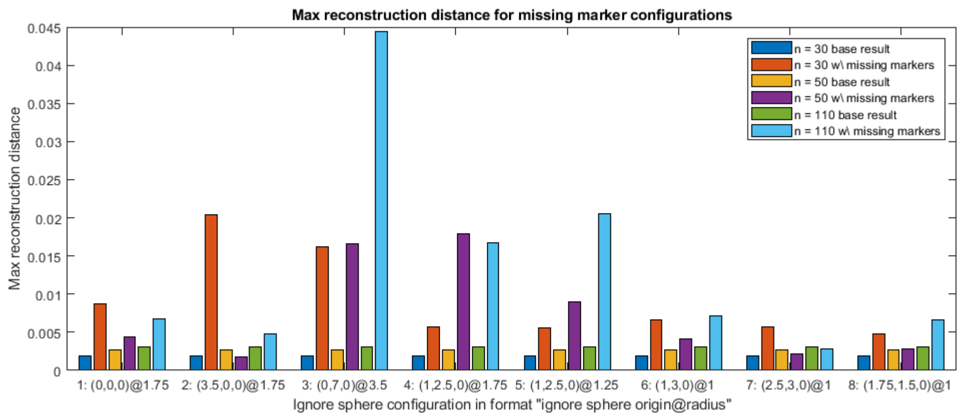


Figure 19: The maximum reconstruction distance for configurations where markers within spheres have been removed for reconstructions with 30, 50, and 110 markers.

good. Lastly, we can see that removing the few markers on the sail edge middle (case 7) barely affects the reconstruction result. It should be noted that only a few markers are removed in this case. Overall, we can see that removing markers from different areas does barely to highly affect the reconstruction result depending on the area. We see the average reconstruction distance increase with a factor of approximately 3 to 10 if we do not have markers spread out around the bulge, and we see an increase with a factor of approximately 1 to 2 for removing markers in less important zones in the sail. It should be noted that regardless of these high factors, the maximum reconstruction distance still generally stays under $2cm$ on a $12.25m^2$ sail.

5.9 Test 6: Standard deviation

To get an indication of the spread in reconstruction results that we can expect for the commonly used marker configurations, we run 20 reconstructions per marker configuration and compute the standard deviation. We run these reconstructions for the wind configuration where $|P| = 1035.1N/m^2$ at an angle of 45 degrees from the boom.

5.9.1 Results

The results are shown in Figure 20 and Figure 21.

5.9.2 Discussion

Looking at both average and maximum reconstruction distances in the results, we can clearly see that having more markers improves the reconstruction results. In the cases with 50 and 110 markers, we can see that the average reconstruction

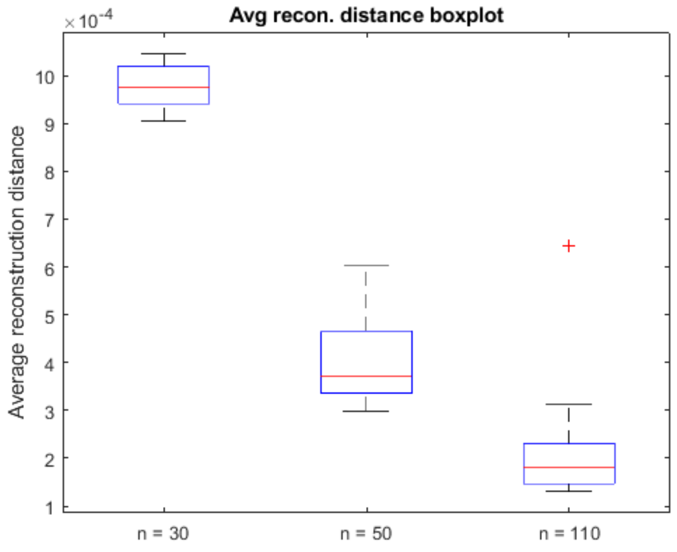


Figure 20: The average reconstruction distance boxplot for reconstructions with 30, 50, and 110 markers.

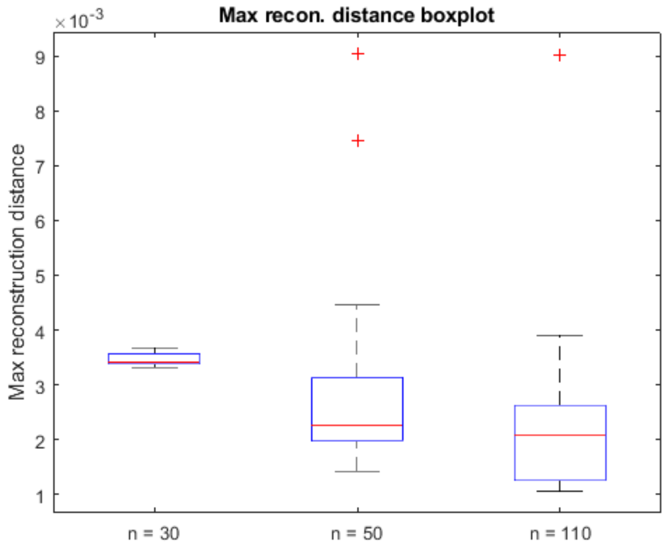


Figure 21: The maximum reconstruction distance boxplot for reconstructions with 30, 50, and 110 markers.

distance has an interquartile range where the minimum and the maximum are about a factor 2 apart. While this could be considered a quite large range, it must be noted that we are looking at ranges within 0.5mm average distance. In the reconstruction case with 30 markers, it seems that the result did not depend much on the starting position, resulting in very steady results. We believe this to be very problem-dependent, and likely not representative for other reconstruction setups. In general, this gives us an insight into what order of magnitude deviation we can expect from results, but we would have to repeat this experiment for all other test setups to get their deviation.

5.10 Test 7: Real world data

The Sailing Innovation Centre [1] has in collaboration with GeoDelta [2] taken point measurements from a sailing boat sail under wind conditions by placing marker stickers onto the sail, and then taking and processing images from that sail. One of these images can be seen in Figure 22. Using these real-world

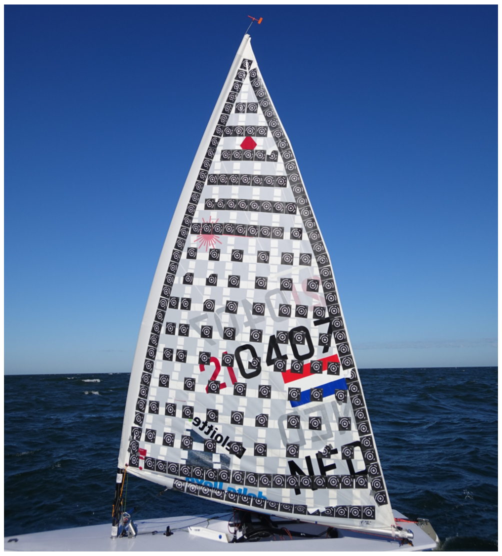


Figure 22: An image from which a sail measurement was generated by the Sailing Innovation Centre in collaboration with GeoDelta.

measurements, we perform several reconstructions to validate that our reconstruction method also works outside of the controlled artificial environment, as well as to see how well it performs on real-world data. We will first explain some extra pre-processing that we have to do to be able to run reconstructions using this data, and define how we measure the reconstruction distance without knowing the correct reconstruction result.

5.10.1 Sail rest configuration

Since all real-world data was created before this thesis, we unfortunately do not have a sail rest configuration available to reconstruct the taken measurements. To solve this problem, we took the measurement where most markers were recorded and performed a Delaunay triangulation on that measurement to obtain a mesh, trimming very long and small triangles at edges to prevent introducing an undesired infinite bending stiffness along those edges. Since this measurement consists of only 183 markers, we perform an extra triangle subdivision over this to obtain a finer mesh. This process is shown in Figure 23. For our convenience and to ensure that gravity will act in approximately the right direction, we define the mast-boom intersection to be the origin in our 3D space, we rotate the mast top vertex to be on the positive y-axis, and we rotate the boom end vertex to be on the positive x-axis. Constraining the mast and boom edges, this leaves us with a mesh that can be used as sail rest configuration.

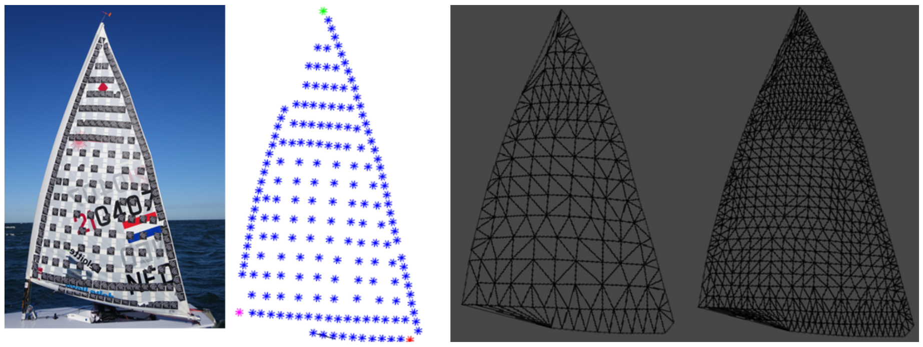


Figure 23: From left to right: The measurement image, the point-cloud measurement data, the Delaunay triangulation of the point-cloud measurement data, and its four-to-one-split subdivided mesh that is used as the real-world test rest configuration.

5.10.2 Measurements pre-processing

Since measurements were taken from outside of the sailing boat, their 3D space origin and rotation do not yet match the sail rest configuration. To solve this, we first translate the measurements to share the mast-boom intersection marker as the origin. We then rotate the measurements such that the measurement marker corresponding with the rest configuration mast top marker is on the positive y-axis, and the measurement marker corresponding with the rest configuration boom end marker is in the positive x-axis direction. We do have to note that the boat from which the measurements were taken has a flexible mast, which this alignment does not account for. This means that this sail rest shape will cause large reconstruction distances when the mast in the measurement is bent in a different way. A possible solution for this problem is to place markers onto the mast and boom themselves to use for alignment and for sail rest shape constraint adjustment, but we cannot do this with the measurements that we have because no markers are placed directly onto the mast and boom there. Leaving this as a recommendation for future real-world measurement generation, we can still perform reconstructions for measurements with a similar mast bend compared to the sail rest configuration.

5.10.3 Marker selection

As with the artificial test configurations, we select a desired amount of markers using farthest-first traversal. The only difference here is that we now traverse all available markers in a measurement, rather than all vertices in an artificially generated mesh.

5.10.4 Reconstruction distance and marker selection

Since the correct reconstruction result is unknown for real-world measurements, we approximate the reconstruction distance by using only a subset of measured markers as measurements and using the remaining markers to approximate the average and maximum reconstruction distance of the entire reconstruction. We know that these markers are fairly well spread across the mesh because these will be the markers remaining after a farthest-first traversal selection.

5.10.5 Test setup

Using the sail rest configuration as defined above, we selected 4 measurements to run reconstructions on. One measurement (measurement 2 in the results) is the measurement that was used to generate the sail rest shape, meaning that marker alignment is ideal, but also that the mesh edge rest lengths form an ideal solution. This might cause the reconstruction result to be able to rely on additional information that it won't have when we would have a perfect sail rest configuration. The other three measurements were taken in the same rack under unknown but likely similar wind conditions. The reconstructions were run with 30, 50, 80, and 110 markers to get an indication of how the number of markers affects the reconstruction distance on real-world data. Note that we did not choose more markers to ensure that enough markers are left to estimate the reconstruction distance from.

5.10.6 Results

The results are shown in Figure 24 and Figure 25.

5.10.7 Discussion

Looking at the results for marker configuration 2, we can clearly see that the reconstruction was successful and that the result is very good with 0.3-1.8mm average reconstruction distance and 6.6-38mm maximum reconstruction distance.

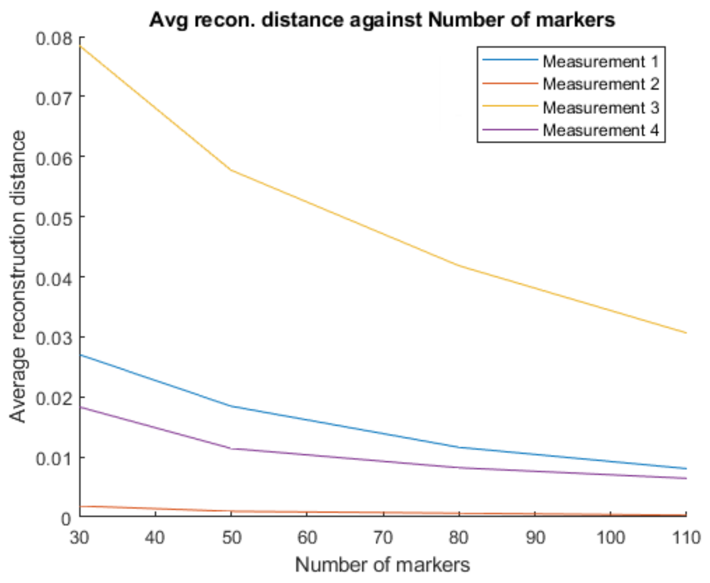


Figure 24: The average reconstruction distance against the number of measurements for different real-world measurements.

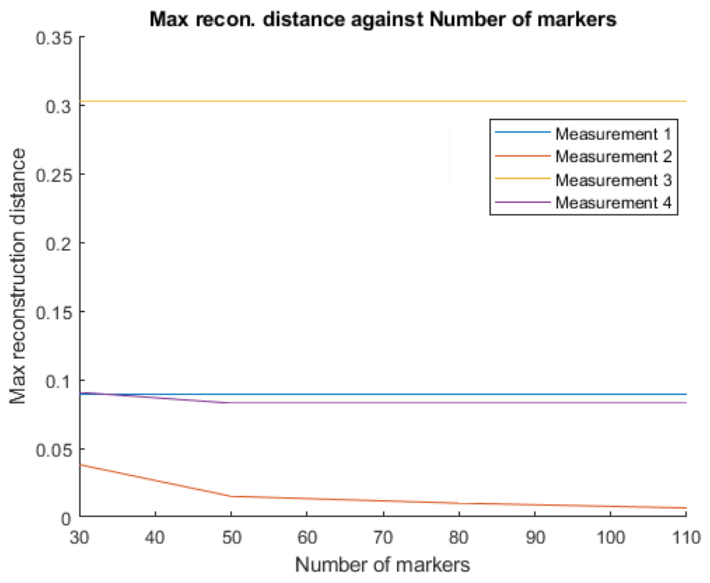


Figure 25: The maximum reconstruction distance against the number of measurements for different real-world measurements.

We did however already note that since this marker configuration was used to generate the sail rest shape mesh with, the result might be better than we can expect from a real-world data reconstruction. Looking at marker configurations 1 and 4, we see a steady maximum reconstruction distance of 9cm for 50 or more vertices. In both cases, this is caused by a marker that is constrained in the sail rest shape mesh as being part of the mast, as can be seen in Figure 26. Looking at the average reconstruction distances for marker configurations 1 and 4, we find distances between 6.5-27mm, where using more markers certainly improves the reconstruction result. Given that measurement marker alignment is not perfect and that a non-ideal sail rest shape was used for these reconstructions, these results are better than expected. The expected result from misalignment can be seen very clearly in marker configuration 3. Adding more markers does pull more of the sail towards the correct position, but the maximum reconstruction distance shows that a constrained mast or boom marker has been misaligned by approximately 30cm. This emphasises the recommendation to place markers directly on the mast and the boom to more easily align measurements with the sail rest shape, as well as modify sail rest shape movement constraints to have the correct mast bend in case that the sailing boat has a flexible mast.

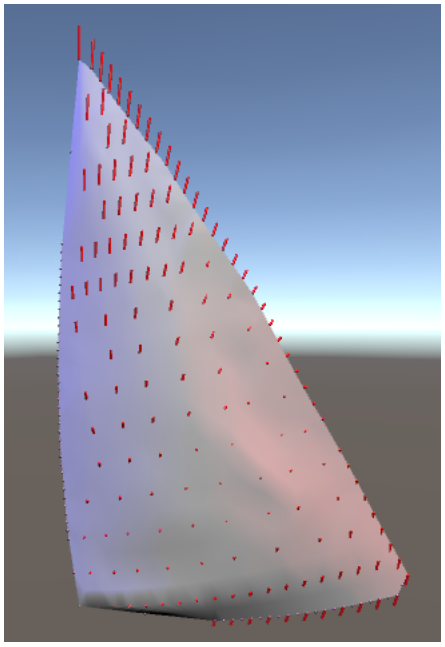


Figure 26: The real-world sail rest shape with red lines between the vertices and their corresponding location in the first real-world measurement.

6 Conclusion

We introduced a physics-based sail simulator, and a point-based sail reconstruction method based on this simulator. To our knowledge, this is the first sail reconstruction method that respects the physical shape and properties of the sail, whilst also using knowledge about the physical environment in which sail measurements were captured.

In a series of tests on artificial data, we have shown that our reconstruction method yields promising reconstruction results where the average reconstruction distance over all vertices in the sail mesh is in the magnitude of millimeters for a $12.25m^2$ sail. We have also shown how this reconstruction distance gets worse when we perform reconstructions using measurements that were generated using different material constants and alternative wind models, and that these reconstruction results remain very good with reasonable amounts of markers on the sail.

Regarding the number of markers, we found that 50 markers distributed using farthest-first vertex traversal performed considerably well on the artificial tests and that more markers yield a better reconstruction result, combined with increased stability against mismatching material properties and a mismatching wind model. Since we cannot be certain about the magnitude of such mismatches in all real-world cases, we can merely suggest the placement of at least 50 markers. We did find the reconstruction result to be best when markers are present on the bulge in the sail and evenly spread around it. Having a majority of markers on one side of this bulge should be avoided as it has a significantly negative impact on the reconstruction result. It should be noted that we chose farthest-first vertex traversal for marker placement in our tests to evenly distribute markers over the sail, but that this might not be a very practical recommendation for placing real-world markers onto a sail. We therefore recommend any marker placement method that approaches an even distribution, as this is very likely to yield similar results.

Through reconstructions performed on real-world data, we have shown that the reconstruction method does yield reasonable results outside of a controlled artificial environment, even though the sail rest configuration was quite roughly approximated. We found that misalignment between measurements and the sail rest configuration (including its movement constraints) has a large negative impact on the reconstruction result, and thus we recommend placing markers directly on the mast and the boom to aid in this alignment. Having such markers also allows for constraining the mast in the sail rest configuration in the right bend, allowing for the sail rest configuration constraints to be correctly set for sailing boats with a flexible mast.

7 Future work

Shell membrane energy
Selecting the membrane energy material constants k_L and k_A based on a known

E-modulus and Poisson's ratio proves to be difficult. Replacing this membrane energy equation with an equation that takes these known material properties as direct input will make it easier to configure the reconstructor for a new sail. A possible candidate is the energy proposed in section 5 equation 8 of [4].

Runtime performance
In our test cases, each reconstruction on a 561 vertex mesh took approximately 5-6 minutes. Increasing the mesh resolution allows for better matching a real sail, but also increases the reconstruction runtime. Performing the parallelizable computations on the GPU instead of the CPU could speed up reconstructions, allowing them to be feasible for higher mesh resolutions.

Wind model
Our simulator and reconstructor work with a very simplistic 3-DOF wind model. It would be interesting to see whether reconstruction results on real-world data would improve when a more complex wind model is used instead. When making no additional changes to the reconstructor, this means that the wind model must be able to provide us with a wind setting that causes wind forces to on average compensate for marker vertex constraint forces. This restriction does not count for the usage of solely the simulator, where we could use any wind model that provides us with wind forces.

Marker placement
We have provided some guidelines for the marker placement on the sail and a minimum amount of markers, which is limited to global zones in the sail. It would be interesting to repeat a part of the performed tests using a genetic algorithm with genes that tell whether a marker is placed for each vertex position in combination with a penalty for using more markers, such that we can find more complex marker patterns that cause reconstructions on them to perform relatively well.

References

[1] Sailing innovation centre. https://www.sailinginnovationcentre.nl, 2022.

[2] Geodelta. https://www.geodelta.com, 2022.

[3] Eitan Grinspun, Anil Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 07 2003.

[4] Behrend Heeren, M. Rumpf, Max Wardetzky, and B. Wirth. Time-discrete geodesics in the space of shells. *Computer Graphics Forum*, 31:1755–1764, 08 2012.

[5] C. Weischedel, A. Tuganov, Tomas Hermansson, Joachim Linn, and Max Wardetzky. Construction of discrete shell models by geometric finite differences. 05 2012.

[6] Hsiao-yu Chen, Paul Kry, and Etienne Vouga. Locking-free simulation of isometric thin plates, 2019.

[7] Z. Zou, M.A. Scott, D. Miao, M. Bischoff, B. Oesterle, and W. Dornisch. An isogeometric reissner–mindlin shell element based on bézier dual basis functions: Overcoming locking and improved coarse mesh accuracy. *Computer Methods in Applied Mechanics and Engineering*, 370:113283, 2020.

[8] Sail simulator. https://www.sailsimulator.com, 2022.

[9] Matteo Lombardi, Massimiliano Cremonesi, Andrea Giampieri, Nicola Parolini, and Alfio Quarteroni. A strongly coupled fluid-structure interaction model for wind-sail simulation. 03 2012.

[10] Luiz Maciel, Ricardo Marroquim, Marcelo Vieira, Kevyn Ribeiro, and Alexandre Alho. Monocular 3d reconstruction of sail flying shape using passive markers. *Machine Vision and Applications*, 32(1):26, Jan 2021.

[11] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran. Optimization integrator for large time steps. *IEEE Transactions on Visualization & Computer Graphics*, 21(10):1103–1115, oct 2015.

[12] Rasmus Tamstorf and Eitan Grinspun. Discrete Bending Forces and Their Jacobians. *Graph. Models*, 75(6):362–370, November 2013.

[13] P.J.S. Kools. Shellsimulator, commit aa259de51c2fd55bc2f4392075d2d340428dd223. https://github.com/Pieter12345/ShellSimulator, 2022.

[14] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, page 131–140, Goslar, DEU, 2004. Eurographics Association.

[15] Christie Alappat, Achim Basermann, Alan R. Bishop, Holger Fehske, Georg Hager, Olaf Schenk, Jonas Thies, and Gerhard Wellein. A recursive algebraic coloring technique for hardware-efficient symmetric sparse matrix-vector multiplication. *ACM Trans. Parallel Comput.*, 7(3), June 2020.

[16] Matthias Bollhöfer, Olaf Schenk, Radim Janalik, Steve Hamm, and Kiran Gullapalli. State-of-the-art sparse direct solvers. pages 3–33, 2020.

[17] Matthias Bollhöfer, Aryan Eftekhari, Simon Scheidegger, and Olaf Schenk. Large-scale sparse inverse covariance matrix estimation. *SIAM Journal on Scientific Computing*, 41(1):A380–A401, 2019.

[18] Michele Calì, Salvatore Oliveri, Antonio Gloria, Massimo Martorelli, and Domenico Speranza. Comparison of commonly used sail cloths through photogrammetric acquisitions, experimental tests and numerical aerodynamic simulations. *Procedia Manufacturing*, 11:1651–1658, 12 2017.