Hierarchical Vorticity Skeletons

Sebastian Eberhardt
Technical University of Munich

Steffen Weissmann
Google Inc.

Ulrich Pinkall
Technical University of Berlin

Nils Thuerey
Technical University of Munich

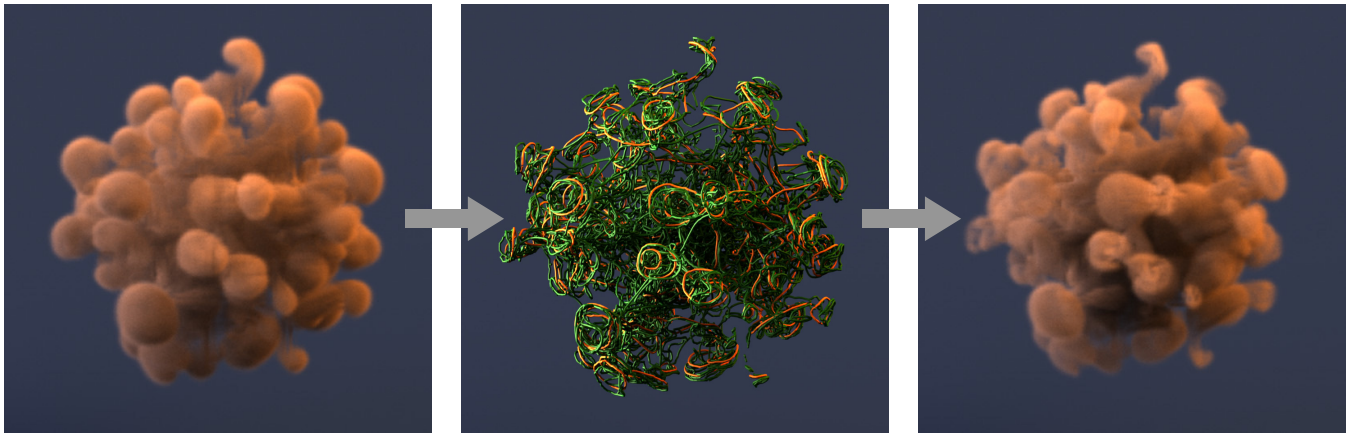

Figure 1: Example of a flow decomposition into a hierarchical vorticity skeleton (HVS) with four strength levels and the reconstruction from the HVS. Our proposed algorithm generates, fully automated, the HVS of any given incompressible velocity field. The HVS, which is sufficient for the complete reconstruction of the flow, saves up to 99.5% of the stored data.

ABSTRACT

We propose a novel method to extract hierarchies of vortex filaments from given three-dimensional flow velocity fields. We call these collections of filaments *Hierarchical Vorticity Skeletons* (HVS). They extract multi-scale information from the input velocity field, which is not possible with any previous filament extraction approach. Once computed, these HVSs provide a powerful mechanism for data compression and a very natural way for modifying flows. The data compression rates for all presented examples are above 99%. Employing our skeletons for flow modification has several advantages over traditional approaches. Most importantly, they reduce the complexity of three-dimensional fields to one-dimensional lines and, make complex fluid data more accessible for changing defining features of a flow. The strongly reduced HVS dataset still carries the main characteristics of the flow. Through the hierarchy we can capture the main features of different scales in the flow and by that provide a level of detail control. In contrast to previous work, we present a fully automated pipeline to robustly decompose dense velocities into filaments.

SCA '17, Los Angeles, CA, USA

978-1-4503-5091-4/17/07…$15.00
DOI: 10.1145/3099564.3099569

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**;

KEYWORDS

Vortex filaments, scale separation, flow Guiding, compression

ACM Reference format:
Sebastian Eberhardt, Steffen Weissmann, Ulrich Pinkall, and Nils Thuerey. 2017. Hierarchical Vorticity Skeletons. In *Proceedings of SCA '17, Los Angeles, CA, USA, July 28-30, 2017,* 11 pages.
DOI: 10.1145/3099564.3099569

1 INTRODUCTION

Despite the widespread use of fluid simulations, they are difficult to deal with in practice: they produce large amounts of dense three-dimensional flow fields which are unintuitive to look at, and even more difficult to edit. We propose a fundamentally different representation of flow data: a hierarchy of vortex filaments that intuitively captures rotating motions from fast large-scale motions down to subtle small-scale fluctuations. As such, these hierarchies represent time-varying *skeletons* of the flow data which dramatically reduce the amount of stored data and provide natural handles to edit and modify the flow. The filament hierarchy can be computed from any incompressible velocity field.

Vortex filaments are lines of constant, concentrated vorticity. They can offer a simplified representation of a flow field from the real world or from simulations. In our approach, several sets of filaments, each with its own vorticity strength, are combined to form a hierarchy. Each of these filament sets represents a certain scale in the flow. In this way, they offer a simplified representation of a fluid motion. In simulated data the number of vortices with different strengths can be very high. This poses a great challenge when trying to find a collection of filaments with a limited number of different strengths which match a given flow field as close as possible. Our hierarchy offers an automated algorithm to achieve this result.

While our algorithm is based on the filament computation method by Weißmann et al. [2014]. This algorithm has inherent shortcomings in its original form. Most prominently, Weißmann [2014] proposes an empirical choice of the filament strength. A priori it is completely unknown which choice of filament strength will produce a good representation of the original flow field, or whether this strength will lead to filaments at all. Fig. 2 shows an example of how strongly results can change when changing this parameter. This figure displays a series of smoke plumes reconstructed from filaments computed with the original algorithm using a varying filament strength. It is apparent that none of the reconstruction results is close to the simulation (far right panel). Thus, it requires a significant amount of manual experimentation to achieve good results with previous work for computing filaments. From this fact we draw our motivation to develop a novel approach for the robust, automatic extraction of a set of filaments which represents the original velocity field very closely. The central contributions of our work are

- an algorithm to decompose a flow field into a hierarchy of vortex filaments,
- an automated strategy to determine circulation strengths,
- an algorithm to improve the filament placement compared to the previous extraction method,
- and a filament-based paradigm for editing and modifying flows.

2 RELATED WORK

Fluid simulations were made popular for computer animation by the early works of Foster and Metaxas [1996], and J. Stam [1999]. Recently a new method based on the evolution of the Schrödinger equation was introduced by Chern et al. [2016] which preserves vorticity very well. We focus on smoke simulations here, but an overview of the larger field is given, e.g., in the book by R. Bridson [2008].

Vortex filaments: A building block for our algorithm is a method for filament decomposition of velocity fields by Weißmann et al. [2014]. However, in contrast to this algorithm, we build a hierarchy of filaments, which turns out to be highly useful for robustness and automation of the decomposition, as well as for editing purposes. It is also possible to directly simulate fluids using filaments as shown in several previous works [Angelidis and Neyret 2005; Weißmann and Pinkall 2009, 2010]. Vortex filaments have also proven themselves useful for obstacle interactions [Vines et al. 2014].

Vorticity methods: Vorticity-based methods are a popular approach within the field, as rotating motions are a crucial component of flows relevant for graphics. An interesting method from the field of computational physics is the vortex-in-cell method [Stock et al. 2008]. For animations, vortex methods have been used in the form of vortex-sheets for liquids [Kim et al. 2009], mesh-based methods [Brochu et al. 2012], or mesh-grid hybrids [Pfaff et al. 2012]. On the other hand, Golas et al. [2012] proposed an interesting combination of vortex-particles and grid-based simulations.

Reverting from vorticity back to velocity poses a challenge that has been addressed in graphics as well. The classic approach is the Biot Savart law, as done by Angelidis and Neyret [2005] and Angelidis et al. [2006]. Zhang and Bridson [2014] redistributed the vorticity of particles to a grid and compute the Biot Savart law with a fast multipole method. The stream functions we employ for converting between vorticity and velocity have likewise seen interesting applications for fluid animations [Ando et al. 2015; Bridson et al. 2007; Elcott et al. 2007].

Model-reduction: This area of research targets finding lower dimensional representations from sets of velocity fields. A first algorithm for model-reduced fluid simulations for computer graphics was proposed by Treuille et al. [2006]. Similar approaches have targeted accurately representing individual stable fluids simulations [Kim and Delaney 2013]. The latter algorithm was also specifically used to target compressed flow representations [Jones et al. 2016]. Later on we will demonstrate that our HVSs, when employed as a data compression technique, can yield data sets that are smaller by an order of magnitude.

Simulation upscaling: A different class of algorithms focuses on increasing the resolution of flows without altering its coarse behavior. These procedural turbulence methods have been proposed in different variations [Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008], but it is worth pointing out that they typically employ a simple linear up-scaling of the velocities. In addition, formulations for particle-based methods were developed [Shao et al. 2015; Yuan et al. 2012]. In contrast to the above methods, our filament hierarchy provides a resolution independent description of the flow. Upsampling with filament hierarchies adds detail to the reconstruction since it is not a linear interpolation but adds rotational motion for each filament. These small motions become more visible with increasing resolution of the reconstruction.

Fluid guiding and control: Ever since fluids were used in the graphics area, guiding and controlling them has been an important topic. Angelidis et al. [2006] control fluids during the simulation of vortex filaments. We, however, propose a method to alter existing flow data using filaments. Other popular approaches are the adjoint method [McNamara et al. 2004], or force-based controls [Shi and Yu 2005]. Additionally, works have focused on matching low- and high-resolution behaviour with optimizations [Nielsen et al. 2009], particle sampling processes [Huang et al. 2011], or by using appropriate boundary-conditions for liquids [Nielsen and Bridson 2011]. A recent variant additionally proposed a localized modification of liquids [Pan et al. 2013]. We will demonstrate that our HVS algorithm offers the possibility to modify flows in a natural way.

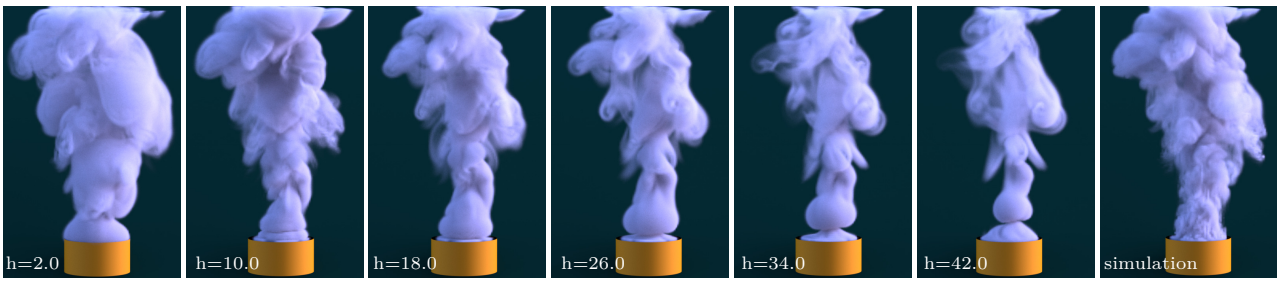

Figure 2: Comparison of several reconstructions employing the original method by Weißmann et al. [2014]. For comparison we show the input simulation at the same time step $\delta_t = 150$. The computational domain consists of $192 \times 192 \times 128$ cells.

3 STATE OF THE ART

We base our method on existing algorithms for extracting filaments from velocity fields and computing velocity fields from vorticity. Both steps are described in the following section and the filament extraction is further detailed in Appendix A.

3.1 Filament Extraction and Reconstruction

The starting point of our algorithm is the computation of filaments from a given velocity $\vec{u}_0$ as described by Weißmann et al. [2014]. For this step we compute a complex eigenvector $\Psi(\vec{u})$ from which we can obtain the filaments. During the extraction we always compute $\Psi(\vec{u})$ as the smallest eigenvalue of a complex energy matrix E^δ. The inputs to this computation are an incompressible velocity field $\vec{u}_0$ and a filament strength h. The filament strength is equivalent to the circulation around a vortex line and is a physical property of any vortex filament. The Helmholtz theorems state that the strength of a vortex filament is constant along its length, and that filaments have to be closed loops.

The second part in our pipeline is to convert a set of filaments F into a velocity field stored on a grid. The concentrated vorticity $\vec{\omega}$ of a filament is distributed into a grid using a Gaussian kernel with size s. As described by Batchelor [1967] we compute a stream function $\vec{\phi}$ with a Poisson equation of the form $\nabla^2\vec{\phi} = -\vec{\omega}$ with Dirichlet boundary-conditions from the redistributed $\vec{\omega}$. In a final step, the curl operator applied to the stream function yields a velocity field: $\vec{u}_{rec} = \nabla \times \vec{\phi}$. The entire reconstruction operation effectively computes $\vec{u}_{rec} = (\nabla\times)^{-1}\vec{\omega}$.

3.2 Discussion

Based on the previous section we can generate filaments of a given strength h from a velocity field. We can also invert this operation. The problem remains to choose h in such a way that the filaments yield a velocity field $\vec{u}_{rec}$ that is as similar to the input velocity field $\vec{u}_0$ as possible. In the following, our goal is to minimize the kinetic energy e_{kin} of the residual, i.e. the velocities not represented with the filaments. The kinetic energy is the defining quantity of many flow fields, and it is directly connected to the motion of the fluid. Without our approach, this finding the right reconstruction strength h results in a significant amount of trial and error.

As pointed out in the introduction, it is highly unlikely that one of these trivial reconstructions fulfills our requirement of being a close representation of the original velocity, minimizing Eq. 3. To demonstrate this fact, Fig. 2 displays a series of reconstructions done with manually chosen filament strength values using the original method [Weißmann et al. 2014]. From the figure it is apparent that h has a significant influence on the result. At the same time the figure shows that none of the reconstructions matches the simulation (right most panel) very well. Besides the visual quality we evaluate how much kinetic energy e_{kin} is left in the residual by computing

$$\epsilon_1 = \frac{e_{kin}(\vec{u}_0 - \vec{u}_1)}{e_{kin}(\vec{u}_0)}, \tag{1}$$

with u_0 being the input velocity field and u_1 the reconstructed velocity.

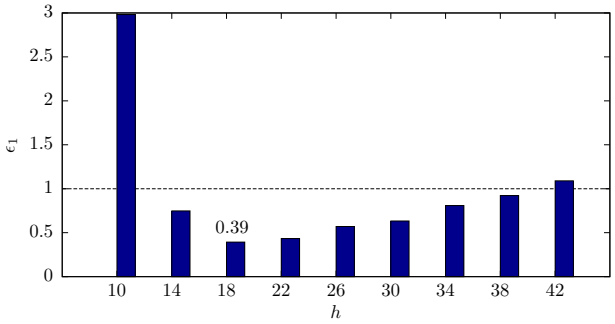

Figure 3: The residual kinetic energy ϵ_1, Eq. 1, from single staged reconstructions with strength h, is displayed by blue bars. The horizontal dashed line marks the kinetic energy of the input. The lowest residual is reached by $h = 18$, still leaving 39% of the energy unrepresented.

Fig. 3 shows the energy residuals ϵ_1 of the reconstructions in Fig. 2. A value of 1.0 (horizontal line) indicates where the kinetic energy of the reconstruction would be zero. These results demonstrate that arbitrary choices of h can result in velocity fields with extremely high energy residual. The reconstruction with $h = 10$ in Fig. 2 and Fig. 3 illustrates this effect. On the other end of the spectrum we can see that the lowest achievable value is 0.39 which means that only 61% of the simulations kinetic energy is captured by the filaments. We will show in Sec. 4.4 that our proposed algorithm recovers more than 90% of the kinetic energy for the same simulation. Our method also does not require a broad search of suitable h values, such as the one from Fig. 3.

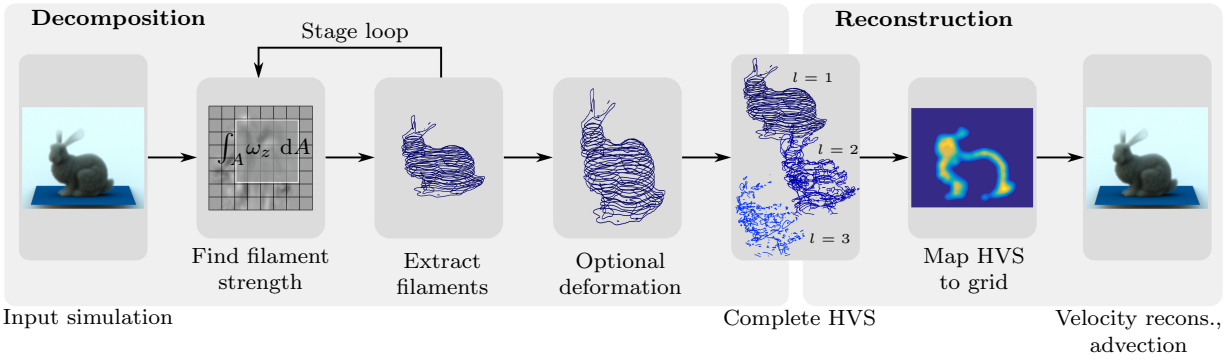

Figure 4: Overview of our decomposition and reconstruction pipeline including the optional deformation step.

4 HIERARCHICAL DECOMPOSITION

Our goal in this work is to obtain a representation of an incompressible input velocity field $\vec{u}_0$ expressed as a set of vortex filaments F which represent the input as accurately as possible. We use the kinetic energy e_{kin} of the residual velocity

$$\vec{u}_{res} = \vec{u}_0 - \vec{u}_{rec} \tag{2}$$

as a minimization quantity

$$\arg\min_F \frac{1}{2} \int_V (\vec{u}_0 - \vec{u}(F))^2 dV = \arg\min_F e_{kin,res}(\vec{u}_0 - \vec{u}(F)) \tag{3}$$

to find the set of filaments F. Here V is the volume for which the kinetic energy is calculated, $\vec{u}_0$ the input velocity field, and $\vec{u}(F)$ the velocity field after our reconstruction from filaments.

While the method described by Weißmann et al. [2014] provides a set of filaments for a single given filament strength h and velocity field $\vec{u}$, the velocity field reconstructed from these filaments will not minimize Eq. 3, as discussed above. Therefore we introduce an automated algorithm which operates hierarchically with several filament extraction cycles, and which automatically determines suitable values for the filament strength h.

In addition to condensing a velocity field into filaments, we also strive to obtain meaningful filaments from the velocity. Thus, we want to extract filaments that correspond to different spatial scales of the input flow. Filaments with large h represent stronger, large scale features, while filaments with low h represent weaker, small scale features. Thus, we aim to successively extract filaments with declining h. The key mechanic of our velocity decomposition to achieve scale separation is a repeated, hierarchical application of the previously described steps. The obtained collection of filaments $F_{0\ldots L}$ from all stages is what we consider to be the *Hierarchical Vorticity Skeleton* (HVS) of the velocity field.

To compute a velocity field from a full HVS with filaments of different strengths, we make use of the fact that the curl is a linear operator. Hence, we accumulate all vorticity of the filaments in an HVS into a single vorticity field by

$$\vec{\omega} = \sum_{l=1}^{L} \vec{\omega}_l \tag{4}$$

with L defined as the total number of stages, and then compute its velocity by a single stream-function solve as described in Sec. 3.1.

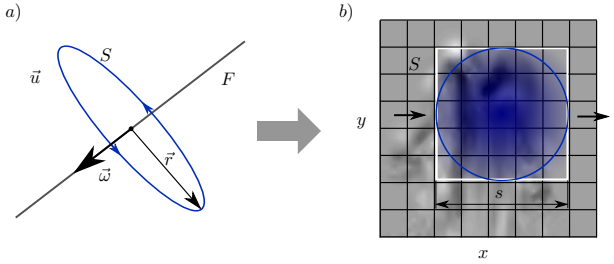

Figure 5: a) Circulation around a vortex line. b) Moving area over which the flow's circulation is computed. The blue shading indicates the Gaussian in the square area to approximate the circulation during reconstruction.

4.1 Filament Strength Calculation

As previously outlined, we require an automatic way of determining a strength value fitting the provided velocity field to realize a hierarchical decomposition into filaments. In accordance with Angelidis and Neyret [2005] we make use of the fact that the filament strength h and the circulation $\Gamma(S)$ around a single vortex line are equal. The circulation of a velocity field $\vec{u}$ around a closed curve S is the line integral $\oint_S \vec{u} \cdot dS$, illustrated in Fig. 5a). With Stokes-theorem, the line integral can be transformed into the vorticity flux through an area A

$$\Gamma(S) = \oint_S \vec{u} \cdot dS = \int_A \vec{\omega} \cdot \vec{n} \, dA. \tag{5}$$

The maximum $\max(\Gamma(S)) = \Gamma^*$ for a typical feature size in the flow domain is a good starting point for the filament extraction, as it represents the maximum local circulation occurring in the flow. Other measures such as velocity or vorticity can not be used directly, as the filaments typically represent larger regions of the flow. We typically use the Gaussian kernel size s as a starting point (details will be given in Sec. 4.4). We aim to extract the dominant features of the given velocity field by using the maximum circulation in each stage. Therefore we search for the maximal circulation Γ^* by evaluating the vorticity formulation of Eq. (5) with a sliding window over the whole domain. The principle is visualized in Fig. 5b). This mechanism robustly computes a suitable filament strength for arbitrary velocity fields, as we will demonstrate in Sec. 4.4.

We compute the entries of the energy matrix E^δ for the face between cell j and k by

$$E^\delta_{jk} = -e^{-i\hat{\eta}_{jk}},\ E^\delta_{kj} = \bar{E}^\delta_{jk},\ E^\delta_{jj} = d,\ \hat{\eta}_{jk} = \frac{2\pi\delta}{h}\vec{u}_{jk} \tag{6}$$

with grid spacing δ and the number of neighboring cells d of cell j. The calculation of E^δ_{jk} imposes a lower limit on the filament strength h. Because $\hat{\eta}_{jk} = 2\pi\delta(u/h)$ in Eq. 6 is the argument of a complex number, it has to be in the interval $[0, 2\pi[$. Due to the rotational nature of $e^{-i\hat{\eta}_{jk}}$, a $\hat{\eta}_{jk} > 2\pi$ will generate the same filaments as $\hat{\eta}_{jk} - 2\pi$. Therefore, we make sure that large velocities extracted first with the final strength value as

$$h = max(\Gamma^*, u_{max}) \tag{7}$$

with h being the final strength, u_{max} the highest velocity in the velocity field and Γ^* the circulation obtained from the sliding box mechanism explained above. While using a lower h than the one obtained from Eq. 7 is not destructive to the algorithm, it lowers the efficiency of the extraction. The reason behind this is that a strength that is to small will result in unnecessarily small contribution to the reconstructed velocity field. This leads to a large residual, which would require more reconstruction stages for reaching the same final residual.

When using the strength search in our hierarchy, we use an initial filter size $s_{l_{max}}$ which equals 5% of the domain size and typically stop the stage iteration after four stages. To account for the weaker and smaller features, in each stage the size s is gradually decreased in accordance with the reconstruction strength, i.e., we apply scaling factor $s_l = s_{l-1}\sqrt{h_l/h_{l-1}}$.

4.2 Optimization of the residual energy

As outlined in Sec. 3.1, filaments are computed from the eigenvector corresponding to the smallest eigenvalue of the energy matrix E^δ. In practice, the values of the smallest eigenvalues are extremely small, e.g., we found that the smallest 100 eigenvalues are often below 0.2% of the largest eigenvalue for our inputs. Therefore, it is possible that other eigenvalues than the smallest one represent important features of the input flow.

To illustrate this more clearly we provide an example in Fig. 6. In a), the figure shows the smoke distribution of a rising plume simulation and the first stage filaments extracted from it. The filaments in the lower section are much smaller in diameter than the plume. This phenomenon can be reproduced with the artificial test in Fig. 6b). The gray area has a pure upwards velocity $u = (0, 1, 0)$ and the remaining domain has no velocity at all, which generates a simplified representation of the smoke plume's lower section. The circles mark the locations at which the filaments cut through the drawing plane. Each color marks filaments computed from a different eigenvalue. Black filaments belong to the smallest eigenvalue Ψ_1. Those filaments exhibit the same behavior of intuitively being too far on the inside, as we expect them to be located at the border of the gray region where the vorticity is highest. Two other sets of filaments from different eigenvalues, here Ψ_3 and Ψ_9, show that it is possible to find filaments which partially match the expected location much better than the ones from Ψ_1. Those could be combined and used for an overall improved placement of filaments. However, as there is no way of determining upfront which eigenvalues will match a certain flow pattern best, there is no upfront analysis to find the optimal eigenvalue(s).

Algorithm 1 Filament optimization

```
1:  procedure OptimizeFilaments(F_l, ω⃗_{l-1})
2:      for f ∈ {F_l} do
3:          L_{2,min} ← +∞
4:          c⃗ ← (Σ x⃗)/n_p
5:          while i < i_max do
6:              x_n ← (x⃗_n − c⃗) α_{f,i} + c⃗
7:              ω⃗_rec ← MapVorticityToGrid(F_l, h)
8:              L_2 ← ||ω⃗_{l-1} − ω⃗_rec||_2
9:              if L_2 < L_{2,min} then
10:                 α_f[i] ← α_{f,i}
11:             i ← i + 1
12:         F_l ← ApplyAllAlphas(F_l, α_f)
```

Therefore we always use Ψ_1 for the basic filaments and employ a posterior approach which scales the filaments to compute the minimization

$$\arg\min_\alpha ||\vec{\omega}_0 - \vec{\omega}\Big(f(\alpha(\vec{x}_n - \vec{c}) + \vec{c})\Big)_{rec}||_2 \tag{8}$$

where $\vec{c}$ is the geometric center of a filament f, $\vec{x}_n$ the points on a filament line and α a scaling factor. In practice, we choose the range of α such that it shrinks or grows filaments no more than 2δ. In Eq. 8 the difference of the input vorticity $\vec{\omega}_0$ and the reconstructed vorticity $\vec{\omega}_{rec,f}$ with the deformed filament f is used for the minimization of the L_2 error. The optimization is done by computing the geometric center of each filament line, see Alg. 1 line 4, and repeatedly expanding the filament around it with a varying factor $\alpha_{f,i}$, in line 6. This results in a line search algorithm which computes the minimization factor α_f for each filament f that minimizes the L_2 error. After an α_f has been found and stored (line 10) for each filament all filaments are scaled with their respective factor before reconstructing the velocity field. For all our examples we used $i_{max} = 10$ steps in this search. For performance reasons we use a different optimization criterion than before in Eq. (3). We will evaluate this choice in more detail below.

After iterating through all filaments, a complete reconstruction is done with all optimized filaments. The rising smoke example discussed in detail in Sec. 4.4 demonstrates a clear improvement when optimizing filament positions as described above.

4.3 The Complete HVS Algorithm

The previous sections describe all components of our HVS extraction. In summary, the procedure consists of the steps shown in Alg. 2 to generate the HVS: The procedure loops through time steps δ_t until it reaches the final time T. The algorithm computes for each stage l the strength h, extracts the filaments F_l, optimizes them and if desired, applies a deformation to the filaments. In practice we use four stages, i.e., $l_{max} = 4$. Following the extraction, the vorticity ω_l from the filaments is mapped to the grid, and a Poisson solve computes the stream function $\vec{\phi}_l$. The residual velocity $\vec{u}_{res}$ then is computed with the curl operator from the stream function.

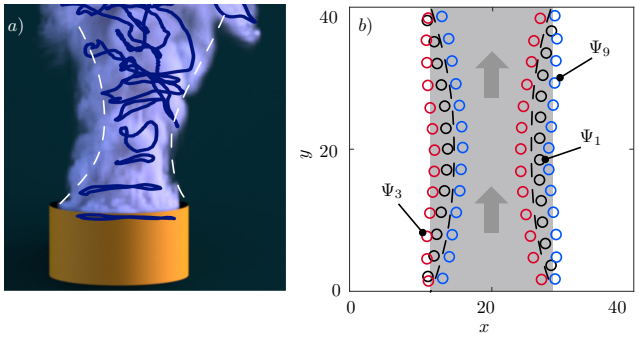

Figure 6: a) Rendering of the smoke plume simulation with overlaid filaments. b) Synthetic reproduction of the lower smoke plume with filaments (circles) of the different eigenvectors Ψ_1, Ψ_3 and Ψ_9. Both show filaments smaller than the plume diameter.

Algorithm 2 Generate the complete HVS

```
1:  procedure ComputeHVS(u_sim, h, bend)
2:      for δ_t in T do
3:          u_1 ← u_sim
4:          u_rec ← 0
5:          for l in [1, l_max] do
6:              u_l ← u_{l-1} − u_rec
7:              h ← ComputeStrength(u_l)
8:              F_l ← ComputeFilaments(u_l, h)
9:              F_l ← OptimizeFilaments(F_l, u_{l-1})
10:             ω_l ← MapVorticityToGrid(F_l, h)
11:             φ_l ← SolvePoisson(ω_l)
12:             u_rec ← ∇ × (φ_l)
13:             Δe_kin ← e_kin(u_l − u_rec)
14:             l = l + 1
15:         u_res ← u_l − u_rec
```

The remaining $\vec{u}_{res}$ after the last stage is the residual velocity of the procedure. The aim of the hierarchical decomposition is to subsequently extract features of the flow that are less and less dominant. This essentially is a scale separation with the most dominant and large coarse features of the flow being extracted in the early stages, and the less dominant and fine structured features in the late stages. Below we will demonstrate that the filaments can be combined and reconstructed into a velocity field with very low residual kinetic energy e_{kin}. Before, we will investigate the effectiveness of our hierarchical extraction without the optimization step.

4.4 Evaluation

To demonstrate the scale separation capability of our algorithm, we execute the decomposition into an HVS on a simple example. It consists of a manually created velocity field induced by three vortex rings with strengths $h_1 = 20$, $h_2 = 10$ and $h_3 = 5$ in a 32^3 grid, as shown in Fig. 7a). We then compute the HVS with three stages using the original strength values starting with h_1 in the first stage, h_2 in the second and h_3 in the last stage. The result, shown in Fig. 7b) in blue, successfully recovers each ring, and shows very

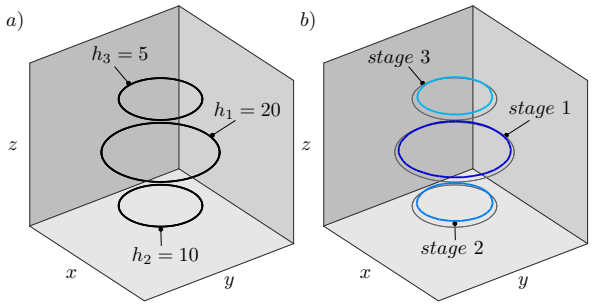

Figure 7: Artificial example on a 32^3 computational grid. *a*) Input vortex rings with their respective strengths $h1, h_2, h_3$. *b*) Reconstructed filaments from three-stages, each stage recovers the isolated ring of the corresponding strength.

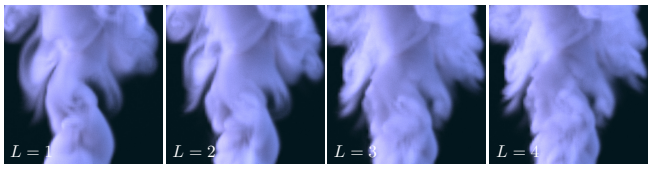

Figure 8: Comparison of reconstructions with increasing number of total stages L. Each stage adds more small scale detail and realism to the reconstruction.

close agreement with the original rings in gray. After the three stage decomposition and reconstruction only 3% of the energy is left in the residual.

Our reconstruction always takes into account the accumulated velocities from all vortex rings. This leads to a slight over-estimation of the filament strengths, and the slightly smaller radii visible in Fig. 7. However, this example demonstrates that our method is successful at extracting and separating vortex rings of different strengths from a single velocity field.

After analyzing the synthetic example, we demonstrate our method on simulation data. We consider the same buoyancy driven, rising smoke plume, as in Fig. 2. The simulation is carried out on a $192 \times 192 \times 128$ cells grid and uses slip wall boundaries on the sides and bottom. The top boundary condition is of convective type as described by Dimakopoulos et al. [2012]. To visualize the reconstructions we advect a scalar smoke density in the reconstructed velocity fields, which are rendered with Mitsuba [2010] (this is done for all our examples).

For the example discussed in this section four stages, e_{kin} decreases between stages three and four only by 1.54%. This shows that the algorithm converges to a certain limit. We argue this limit exists because we project the input velocity $\vec{u}_0$ field onto a space of velocities which can be represented by filaments, but $\vec{u}_0$ is not completely contained within the filament space. We use this convergence as a criterion for stopping the algorithm.

We demonstrate the visual improvement of our method through staged decomposition with Fig. 8. The figure displays the effect of conducting reconstructions with an increasing number of stages. From *a*) to *d*) the total number of stages L increases from one to

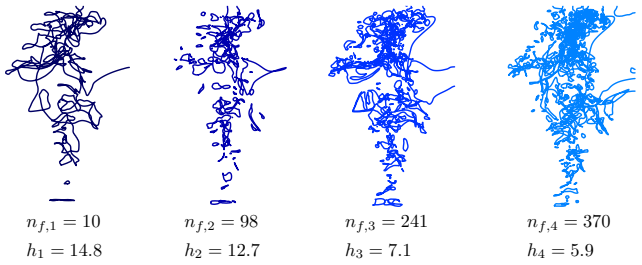

Figure 9: The complete four stage HVS of a simulation from a single time step, displaying the descending strength levels h_l and increasing number of filaments $n_{f,l}$.

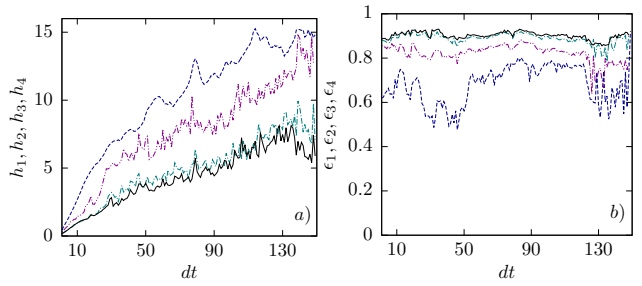

Figure 10: a) Strength h for all four stages of all time steps, b) recovered e_{kin} during each stage of all time steps. Colors: stage 1 in blue, 2 purple, 3 green, and 4 black.

four, and likewise, with growing L, the flow details captured by the reconstruction are increasing. This becomes more evident when looking directly at the filaments in Fig. 9. The figures clearly show an increase in filament number and small scales with each stage. Furthermore, the filament strength h decreases, as expected, for each stage. Fig. 10a) demonstrates the development of the filament strengths through all time steps for the plume simulation. The smoke plume develops from the inlet over time growing in size and speed over the course of the simulation. This explains the increase of all strength levels over time. The consistent decrease of strengths through our hierarchy is also clearly visible when looking at the data points of a single time step. Fluctuations seen in the strength originate from the transient nature of the plume, with flow features, e.g. vortices, developing and decaying while the simulation progresses.

Fig. 10b) very clearly demonstrates the high level of kinetic energy e_{kin} recovered with our method from the simulation's velocity field $\vec{u}_0$ by plotting the ratios of energies

$$\epsilon_l = 1 - \frac{e_{kin}(\vec{u}_0 - \sum_l(\vec{u}_l))}{e_{kin}(\vec{u}_0)}. \quad (9)$$

The subscript 0 denotes the input field and subscript l marks stage l. On average ϵ_4 is 90.2% and peaks at 93% in time step 84 after the fourth stage.

Furthermore the graphs show the robustness of the HVS algorithm: Firstly, the recovered energy converges to a maximum which lies near the level of stage four. The convergence is observable in Fig. 10 as the strength and energy lines of stage three to four are

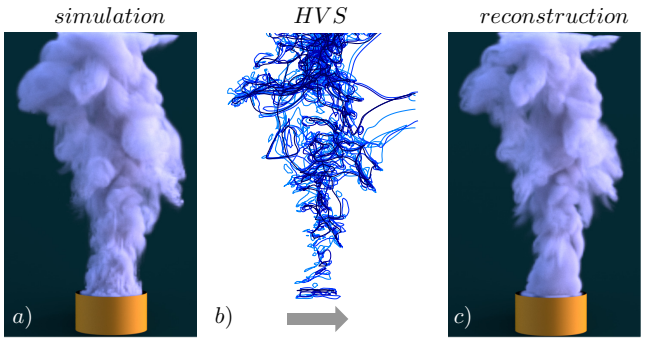

Figure 11: a) Rising smoke plume simulation, b) complete HVS (colors correspond with the stages in Fig. 9), c) reconstruction. Time step $\delta_t = 150$.

located very close to each other. The result in Fig. 8 confirms the convergence as the visual difference between panels $L = 3$ and $L = 4$ is very small.

Secondly, in the region between time steps 20 and 50 of Fig. 10, h_1 does not perfectly fit the current flow field, as the recovered energy is low relative to the adjacent time steps. Since the search size s_1 of the first stage is constant throughout the entire simulation, it is possible that the size is not optimal in the first stage during some of the time steps. This can happen due to local features, such as a vortex which exists for a short range of time steps. Since h is global for each stage, this will have an effect on the overall filament extraction. Our algorithm compensates for this effect with a higher strength h_2 for the next level, extracting the additional information in stage two. This evens out the recovered energy after the fourth stage to an almost constant value, see topmost line in Fig. 10b).

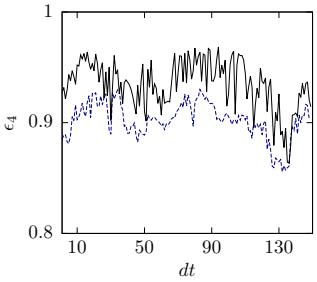

Figure 13: Recovered e_{kin} for the standard algorithm (dashed blue), and with optimization (black).

A single, exemplary time step of the reconstruction is displayed in Fig. 11. The demonstrated high level of recovered kinetic energy is supported by the very good visual agreement of simulation and reconstruction. Only minor, mostly noisy features from the simulation are absent in the reconstruction.

When the optimization, Sec. 4.2, is applied to the reconstruction, the recovered kinetic energy increases further. The ratio ϵ_4 in Eq. 9 after stage four is plotted in Fig. 13 for the standard reconstruction, and the optimized version. A clear improvement can be seen for the optimized version. The average recovered e_{kin} increases to 93.48% and peaks at 96.85% in timestep 95. Note that our optimization could also be evaluated with velocities instead of the vorticity in Eq. (8). While this yields slightly better results, the computational

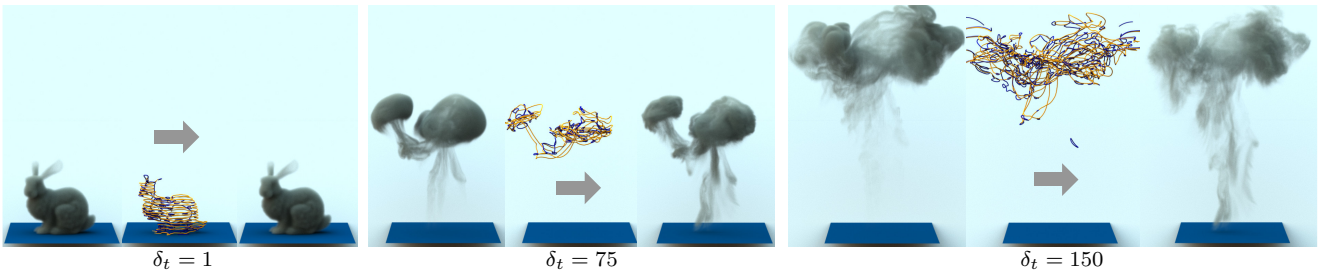

Figure 12: Stanford bunny filled with hot smoke simulated and reconstructed on a $200 \times 300 \times 200$ grid. The panels are at time steps $\delta_t = 1$, $\delta_t = 75$, and $\delta_t = 150$, the filaments are from the first stage. These illustrations display a very good agreement of the simulation's dominating features and their reconstruction.

cost increases due to the very high number of required Poisson solves. Using velocities for the example above recovers 94.9% on average, with a maximum of 98.3%.

5 APPLICATIONS

In the following section we present flow decomposition examples of varying complexity and use them to demonstrate the key aspects of our decomposition and modification method.

5.1 Compression

Using an HVS to represent a three-dimensional velocity field yields a specialized, yet powerful compression mechanism for fluid motions. The information which needs to be saved in order to reconstruct the flow from an HVS are the (1) filament lines F_l, (2) filament strength h_l and (3) the reconstruction radius s_l. Smoke density does not need to be saved as it can be reconstructed by advecting a passive scalar through the reconstructed velocity field at almost no cost (see advection cost in Tab. 2). Since there is a small residual velocity which is lost, the compression by HVS is a lossy one.

To demonstrate the generality of the HVS compression we provide additional examples of reconstructions. All examples are listed in Tab. 1. We achieved the best compression with the simulation of a bunny shaped, heated smoke volume, displayed in Fig. 12. It amounts to less than 0.5% of the size of the input simulation. Besides the compression ratio, the table lists the number of filaments in all stages for all decompositions. The increasing number of filaments with increasing stage l underlines the scale separation achievable with the HVS decomposition.

We want to emphasize that all listed decompositions save more than 99% of storage space when comparing the filaments to three-dimensional velocity fields. In order to not make the comparisons look unnecessarily bad, all datasets have been saved in a gzip compressed binary format. With uncompressed raw data the compression ratio of HVSs would be even larger.

5.2 Flow editing

Vortex filaments also provide a more intuitive way of editing flows than directly working with volumetric velocity or density fields. We can understand the HVS as a *skeleton representation* of a flow. This skeleton is made up from simple closed lines which could easily be edited by a user. To demonstrate a global deformation of the flow field, we use the rising smoke setup introduced in Sec. 3.2.

Table 1: Size and compression information of all examples in this work. The table demonstrates the scale separation by increasing filament numbers through stages l and a very high compression by saving more than 99% of data in all examples. All data sets include the complete simulations.

	Smoke plume	Stanford bunny	Buoyancy sphere
$l = 1$	2981	3943	16884
$l = 2$	3499	6265	18789
$l = 3$	28267	68022	99742
$l = 4$	47534	117087	186336
total	82281	195317	321751
filament size [MB]	41.7	93.4	125.6
% [MB] of $\vec{u}_0$	0.59	0.49	0.76
number of timesteps	150	150	200
$\vec{u}_0$ resolution	$192^2 \times 128$	$200^2 \times 300$	200^3

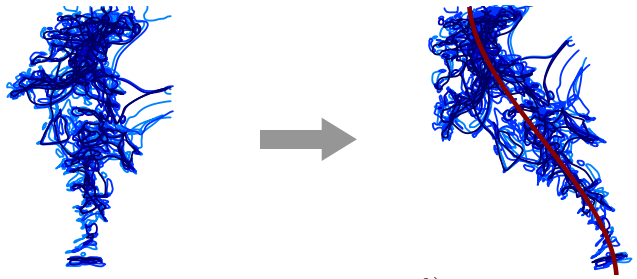

Figure 14: a) Original HVS filaments, b) deformed HVS filaments translated and rotated along the red spline.

Fig. 14 displays the original filaments of the HVS together with the deformed filaments and the deformation curve in red. To achieve the deformation, we define a cubic Hermite spline with its starting point at the base of the plume. The points $\vec{x}_i$ of all filaments in the HVS are easily translated and rotated with this spline deformation. A direct shearing deformation of the density results in a very stretched result, especially in the mid-section, as demonstrated in Fig. 15b).

5.3 Resolution Changes

The HVS and vortex filaments in general provide a representation of a flow which is independent of the grid on which the simulation

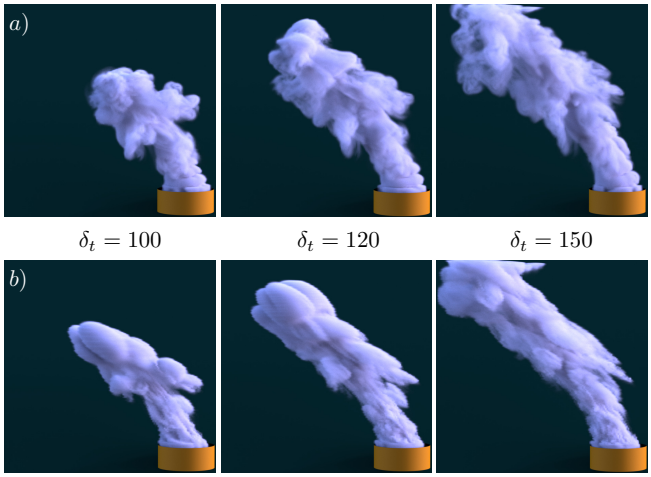

Figure 15: Time development of smoke being advected through a) $\vec{u}_{rec}$ generated by bent filaments showing the original dynamic behavior of the flow and b) the simulation's smoke density shifted along the spline showing strong stretching and grid artifacts.

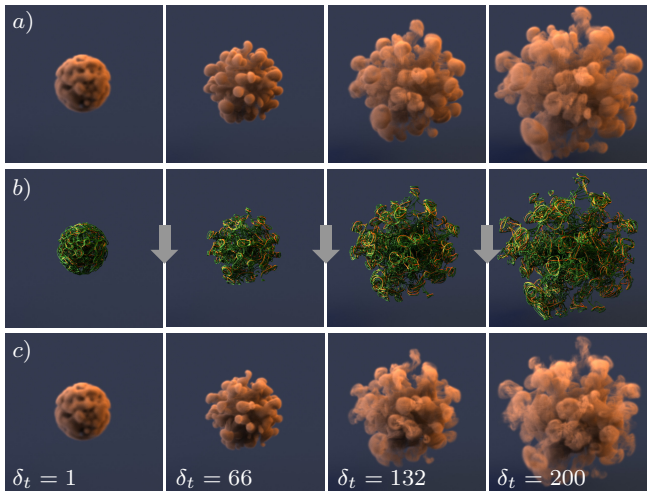

Figure 16: a) Simulation on a $n = 200^3$ grid, b) Filaments of all stages (HVS), c) reconstruction on a $n = 200^3$ grid. The setup is a spherical flow driven by radial buoyancy. The reconstruction stays very close to the input.

has been carried out. As a consequence, the reconstruction can be done with an arbitrary resolution of choice, potentially also much higher ones than the original flow field.

To demonstrate this effect we simulate a smoke sphere which is driven by buoyancy pointing radially outward from the spheres center. A time series of images displaying the simulation and reconstruction, both conducted on a 200^3 grid, is shown in Fig. 16. In Fig. 17a) we linearly upscaled a reconstruction from a $n = 200^3$ grid to $n = 400^3$. This version lacks sharpness and detail despite the increase in resolution. In contrast we provide a result of filaments which have been directly reconstructed on the $n = 400^3$ grid in Fig. 17b). The figure shows that the direct high resolution reconstruction clearly features more small scale details and sharpness than the simple linear upscaled version. This is due to the fact that small scale motions by the filaments of stages three and four become more visible in a high resolution grid, an effect which sets the HVS approach further apart from the original single stage algorithm.

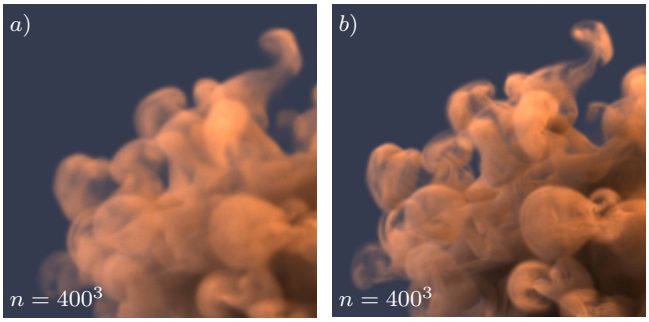

Figure 17: a) Reconstruction on a $n = 200^3$ resolution grid, linearly upscaled to $n = 400^3$, b) Reconstruction from the HVS directly on a $n = 400^3$ grid. The direct reconstruction clearly adds detail and quality. Time step $\delta_t = 100$.

5.4 Implementation and Performance

To compute the smallest eigenvalue and it's eigenvector of E^δ we use the PRIMME [Statopoulos and McCombs 2010] eigen-solver with AGMG [Notay 2014] preconditioning, expanding the complex entries of E^δ into real-valued matrices [Weißmann et al. 2014]. All remaining functionality for the extraction and reconstruction of HVSs is implemented in mantaflow [Thuerey and Pfaff 2015].

As long as no optimization is used, during the generation of filaments the eigenvalue computation with PRIMME is the most time consuming part. The optimization then adds further substantial duration to the runtime of the decompositions. The exact additional time depends linearly on the number of filaments which are generated. In our examples the optimization using $||\vec{\omega}||_2$ adds a factor between two and five, although we want to point out that our implementation of this step is not fully optimized. A detailed breakdown of the average computational times per time step is given in Tab. 2 for all presented decompositions. The row denoting the total time sums up the time relevant to the HVS algorithm with respect to the total runtime. All measurements were collected on a workstation PC with an Intel Xeon E5-1650 and 32Gb of random access memory.

During the reconstruction of filaments loaded from disk, the Poisson solve for the stream function consumes most of the computation time. Since the Poisson solve is the computationally most expensive part in regular simulations as well, the reconstruction operates in a similar time frame. The time for advecting smoke density through a velocity field is extremely short, see Tab. 2.

Table 2: Average run times for optimized and non-optimized versions for a single time step of all presented examples.

	Smoke plume		Stanford bunny		Buoyancy sphere	
no optimization						
advect	0.1s	0%	0.3s	0%	0.2s	0%
h_l	0.5s	0.1%	1.3s	0.1%	0.8s	0.2%
$F_l \rightarrow \vec{\omega}_{gr}$	1.1s	0.3%	3s	0.3%	1.3s	0.2%
Ψ_l	278s	70.2%	624s	62.2%	373s	69.7%
$F_l(\Psi_l)$	6s	1.6%	26.1s	2.6%	7s	1.4%
Φ_l	105s	26.5%	340s	33.8%	145s	27.1%
total	396s	98.7%	1005s	99.0%	535s	98.6%
$\|\|\vec{\omega}\|\|_2$ optimization (used for all presented examples)						
advect	0.1s	0%	0.3s	0%	0.2s	0%
h_l	0.5s	0.1%	1.3s	0%	0.8s	0%
$F_l \rightarrow \vec{\omega}_{gr}$	2.1s	0.2%	5.4s	0.2%	2.7s	0.1%
Ψ_l	264s	38.1%	651s	24.9%	373s	14.2%
$F_l(\Psi_l)$	7s	1.1%	34s	1.3%	10.1s	0.3%
Φ_l	107s	15.4%	349s	13.3%	148s	5.6%
optimize	313s	45.2%	1572	60.2%	2091s	79.6%
total	694s	99.8%	2614s	99.9%	2626s	99.8%

5.5 Limitations and Discussion

The decomposition of a flow into an HVS can be viewed as a compression accompanied by a certain loss of information. Small and weak features are can be removed. We argue that this is to be weighed against the extremely large compression factors demonstrated in Sec. 5.1. In some cases, omitting these small scale features can actually lead to improvements in the form of a less noisy appearance. Additionally, our method works well with smaller resolutions for the large-scale filaments. We have verified this, but all examples in this paper use a fixed resolution. Using coarser resolutions could further reduce extraction time.

We also note the computational cost required for reconstructing the final velocities. However, the reconstruction step could potentially be sped up by using solvers based on Fast Multipole Methods. Especially for large resolutions, this could lead to significantly reduced runtimes.

The filament extraction operates on a per time step basis, meaning there is no temporal coherence between filaments of different time steps. This can cause jitter if filaments are played back as a video. However, we do not consider this an issue as our algorithm optimizes for a low residual energy, and not for filaments to be visualized directly.

While it may seem an alternative to use the fluid simulation itself as a means of compression, we believe that this is not a practical alternative. Using the fluid simulation itself always requires complete boundary conditions and force fields, as well as a potentially complex implementation in order to retrieve the flow field. For practical applications, e.g., in visual effects, complex boundary interactions can require huge triangle meshes, and thus large amounts of memory. Our method, on the other hand, only requires three simple Poisson solves to reconstruct a velocity via the stream function, thus it is independent of a particular choice of solver, and does not require any information about the boundary conditions of the simulation that generated the original flow fields.

Note that obstacles in the flow field would not be problematic for our method, as long as the obstacle is represented in the velocities. I.e., our method does not make any assumptions about the inputs, and is general in the sense that it will represent arbitrary divergence-free flow fields, no matter what their content is.

6 CONCLUSION

We have presented a novel way to decompose fluid simulation data into sets of individual vorticity lines of varying strength. Our approach simultaneously serves as a scale separation, as well as a compression mechanism for arbitrary divergence-free velocity fields. Additionally, we demonstrated that our *Hierarchical Vorticity Skeletons* enable animators to modify flows in an intuitive way. Our approach makes computing filaments a fully automated process that eliminates the need for parameter tuning in order to achieve a good reconstruction.

In the future, our HVS-based compression could potentially be used for data driven fluid simulations. Since the HVS compression automatically provides a way of saving huge amounts of storage space, it could be a starting point to let artists work work with large collections of flow data.

REFERENCES

Ryoichi Ando, Nils Thuerey, , and Chris Wojtan. 2015. A Stream Function Solver for Liquid Simulations. *Transactions on Graphics (SIGGRAPH)* 34 (2) (August 2015), 8.

Alexis Angelidis and Fabrice Neyret. 2005. Simulation of Smoke Based on Vortex Filament Primitives. In *Symposium on Computer Animation (SCA '05)*, Ken Anjyo and Petros Faloutsos (Eds.). ACM-SIGGRAPH/EG, ACM Press, Los Angeles, United States, 35–48. DOI: https://doi.org/10.1145/1073368.1073380

Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 25–32.

G. K. Batchelor. 1967. *An Introduction to Fluid Dynamics*. Cambridge University Press.

Robert Bridson. 2008. *Fluid Simulation for Computer Graphics*. AK Peters/CRC Press.

Robert Bridson, Jim Houriham, and Marcus Nordenstam. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 46.

Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 87–95.

Albert Chern, Felix Knöppel, Ulrich Pinkall, Peter Schröder, and Steffen Weißmann. 2016. SchröDinger's Smoke. *ACM Trans. Graph.* 35, 4, Article 77 (July 2016), 13 pages. DOI: https://doi.org/10.1145/2897824.2925868

Yannis Dimakopoulos, George Karapetsas, Nikolaos A. Malamataris, and Evan Mitsoulis. 2012. The Free (Open) Boundary Condition at inflow boundaries. *Journal of Non-Newtonian Fluid Mechanics* 187âĂŞ188 (2012), 16 – 31. DOI: https://doi.org/10.1016/j.jnnfm.2012.09.001

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics (TOG)* 26, 1 (2007), 4.

Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. *Graph. Models Image Process.* 58, 5 (Sept. 1996), 471–483. DOI: https://doi.org/10.1006/gmip.1996.0039

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 148.

Ruoguan Huang, Zeki Melek, and John Keyser. 2011. Preview-based sampling for controlling gaseous simulations. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. 177–186.

Wenzel Jakob. 2010. Mitsuba renderer. (2010). http://www.mitsuba-renderer.org.

Aaron Demby Jones, Pradeep Sen, and Theodore Kim. 2016. Compressing fluid subspaces. In *Proc. Symposium on Computer Animation*. ACM/Eurographics, 77–84.

Doyub Kim, Oh-young Song, and Hyeong-Seok Ko. 2009. Stretching and wiggling liquids. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 120.

Theodore Kim and John Delaney. 2013. Subspace Fluid Re-simulation. *ACM Transactions on Graphics* 32, 4, Article 62 (July 2013), 9 pages.

Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008. Wavelet Turbulence for Fluid Simulation. *ACM Trans. Graph.* 27, 3, Article 50 (Aug. 2008), 6 pages. DOI: https://doi.org/10.1145/1360612.1360649

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid Control Using the Adjoint Method. *ACM Transactions on Graphics* 23, 3 (2004), 449–456.
Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph.* 27, Article 166 (December 2008), 8 pages. Issue 5.
Michael B. Nielsen and Robert Bridson. 2011. Guide Shapes for High Resolution Naturalistic Liquid Simulation. *ACM Trans. Graph.* 30, 4, Article 83 (July 2011), 8 pages. DOI: https://doi.org/10.1145/2010324.1964978
Michael B. Nielsen, Brian B. Christensen, Nafees Bin Zafar, Doug Roble, and Ken Museth. 2009. Guiding of Smoke Animations Through Variational Coupling of Simulations at Different Resolutions. In *Proc. Symposium on Computer Animation*. ACM, New York, NY, USA, 217–226. DOI: https://doi.org/10.1145/1599470.1599499
Yvan Notay. 2014. *AGMG software and documentation; see http://homepages.ulb.ac.be/~ynotay/AGMG.*
Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive Localized Liquid Motion Editing. *ACM Transactions on Graphics (SIGGRAPH Asia 2013)* 32, 6 (Nov. 2013).
Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 112.
H. Schechter and R. Bridson. 2008. Evolving sub-grid turbulence for smoke animation. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. 1–7.
Xuqiang Shao, Zhong Zhou, Jinsong Zhang, and Wei Wu. 2015. Realistic and stable simulation of turbulent details behind objects in smoothed-particle hydrodynamics fluids. *Computer Animation and Virtual Worlds* 26, 1 (2015), 79–94. DOI: https://doi.org/10.1002/cav.1607
Lin Shi and Yizhou Yu. 2005. Taming Liquids for Rapidly Changing Targets. In *Proc. Symposium on Computer Animation*. ACM, New York, NY, USA, 229–236. DOI: https://doi.org/10.1145/1073368.1073401
Jos Stam. 1999. Stable fluids. In *SIGGRAPH 1999*. 121–128.
A. Statopoulos and J.R. McCombs. 2010. PRIMME: PReconditioned Iterative Multi-Method Eigensolver: Methods and Software Description. *ACM Trans. Math. Softw.* 37, 2 (2010), 21:1–21:30.
Mark J Stock, Werner JA Dahm, and Grétar Tryggvason. 2008. Impact of a vortex ring on a density interface using a regularized inviscid vortex sheet method. *J. Comput. Phys.* 227, 21 (2008), 9021–9043.
Nils Thuerey and Tobias Pfaff. 2015. MantaFlow. (2015). *http://mantaflow.com.*
Adrien Treuille, Andrew Lewis, and Zoran Popović. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics* 25, 3 (July 2006), 826–834.
Mauricio Vines, Ben Houston, Jochen Lang, and Won-Sook Lee. 2014. Vortical inviscid flows with two-way solid-fluid coupling. *Visualization and Computer Graphics, IEEE Transactions on* 20, 2 (2014), 303–315.
Steffen Weißmann and Ulrich Pinkall. 2009. Real-time Interactive Simulation of Smoke Using Discrete Integrable Vortex Filaments. In *Workshop in Virtual Reality Interactions and Physical Simulation V̈RIPHYS(̈2009)*, Hartmut Prautzsch, Alfred Schmitt, Jan Bender, and Matthias Teschner (Eds.). The Eurographics Association. DOI: https://doi.org/10.2312/PE/vriphys/vriphys09/001-010
Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based Smoke with Vortex Shedding and Variational Reconnection. *ACM Trans. Graph.* 29, 4, Article 115 (July 2010), 12 pages. DOI: https://doi.org/10.1145/1778765.1778852
Steffen Weißmann, Ulrich Pinkall, and Peter Schröder. 2014. Smoke Rings from Smoke. *ACM Trans. Graph.* 33, 4, Article 140 (July 2014), 8 pages. DOI: https://doi.org/10.1145/2601097.2601171
Zhi Yuan, Ye Zhao, and Fan Chen. 2012. Incorporating stochastic turbulence in particle-based fluid simulation. *The Visual Computer* 28, 5 (2012), 435–444.
Xinxin Zhang and Robert Bridson. 2014. A PPPM Fast Summation Method for Fluids and Beyond. *ACM Trans. Graph.* 33, 6, Article 206 (Nov. 2014), 11 pages. DOI: https://doi.org/10.1145/2661229.2661261

A FILAMENT EXTRACTION

Without loss of generality, we assume a Cartesian grid with uniform cell spacing δ and staggered velocities for the following description. We begin the calculation with the complex minimizer Ψ of the energy

$$E(\Psi) = \frac{\hbar^2}{2}||d^{\nabla}\Psi||^2 \tag{10}$$

with $\hbar = h/2\pi$ and $d^{\nabla}\Psi := d\Psi - i\hat{\eta}\Psi$ ($\hat{\eta}_{jk} = (\delta/\hbar)\vec{u}_{jk}$, see Eq. 11). The minimizer Ψ can be computed as the eigenvector corresponding to the smallest eigenvalue of a Laplace-like energy matrix. The matrix entries E^{δ} for the cell interface between cells j and k are calculated from the velocity field $\vec{u}$ as

$$E^{\delta}_{jk} = -e^{-i\hat{\eta}_{jk}},\ E^{\delta}_{kj} = \bar{E}^{\delta}_{jk},\ E^{\delta}_{jj} = d,\ \hat{\eta}_{jk} = \frac{\delta}{\hbar}\vec{u}_{jk} \tag{11}$$

with u_c being the velocity component at the face between two

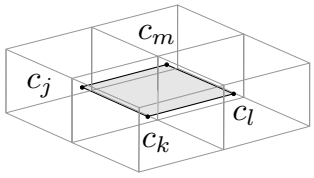

Figure 18: Naming convention for the cell indices at the centers of four adjacent cells.

grid cells, d the number of neighboring cells and grid spacing δ. For a zero velocity in 3D, these matrix entries yield the typical finite-difference Laplace stencil, while larger values yield different off-diagonal entries.

The vortex filaments are given by the intersection lines of the smallest eigenvector Ψ, the iso-surfaces of $\mathrm{Re}(\Psi) = \mathrm{Im}(\Psi) = 0$. These zero curves can be found via a combination of the winding number n_{jklm} and a tri-linear interpolation of Ψ, with a line version of Marching-cubes. The winding number n_{jklm} is calculated as

$$n_{jklm} = \frac{1}{2\pi}\left(\arg\left(\frac{\Psi_k}{\Psi_j}\right) + \arg\left(\frac{\Psi_l}{\Psi_k}\right) + \arg\left(\frac{\Psi_m}{\Psi_l}\right) + \arg\left(\frac{\Psi_j}{\Psi_m}\right)\right) \tag{12}$$

were $\Psi_{j,k,l,m}$ are the Ψ values on the centers of four consecutive cells in the grid, see Fig. 18 and the *arg* function computes the arc tangent. The resulting winding number can only be minus one, zero or one. The zero intersection curves of Ψ only intersect cells with winding number ± 1. The location of these line intersections is calculated using

$$0 = (1-v)((1-u)\Psi_j + u\Psi_k) + v((1-u)\Psi_m + u\Psi_l). \tag{13}$$

Because the winding number in intersecting cells is ± 1, we know that one of the (u, v) coordinate pairs from Eq. (13) lies in $(0, 1) \times (0, 1)$.

While it would be possible to post-process the extracted line segments to further reduce the size of the data to be stored to represent the filaments, we have not done so in this work.