Progressive Direct Volume-to-Volume Transformation

Steffen Frey and Thomas Ertl

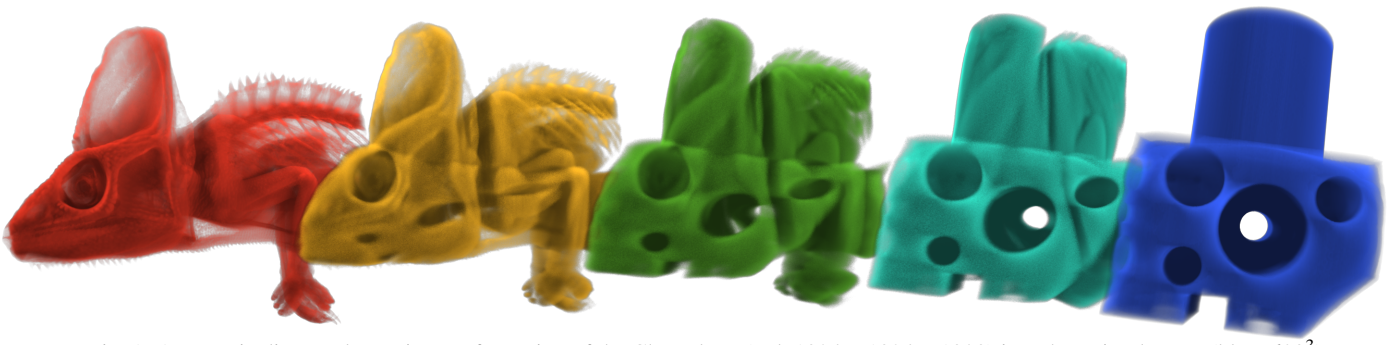


Fig. 1: Automatic direct volumetric transformation of the Chameleon (red, $1024 \times 1024 \times 1080$) into the Zeiss data set (blue, 680^3).

Abstract— We present a novel technique to generate transformations between arbitrary volumes, providing both expressive distances and smooth interpolates. In contrast to conventional morphing or warping approaches, our technique requires no user guidance, intermediate representations (like extracted features), or blending, and imposes no restrictions regarding shape or structure. Our technique operates directly on the volumetric data representation, and while linear programming approaches could solve the underlying problem optimally, their polynomial complexity makes them infeasible for high-resolution volumes. We therefore propose a progressive refinement approach designed for parallel execution that is able to quickly deliver approximate results that are iteratively improved toward the optimum. On this basis, we further present a new approach for the streaming selection of time steps in temporal data that allows for the reconstruction of the full sequence with a user-specified error bound. We finally demonstrate the utility of our technique for different applications, compare our approach against alternatives, and evaluate its characteristics with a variety of different data sets.

Index Terms—Volume transformation, Volume visualization, progressive, automatic, parallel, time-varying data, streaming data

1 Introduction

Advances in parallel computing systems for simulations and high-accuracy measurement techniques drive the generation of volume data sets with increasing resolution in both time and space. Collections of volumes are not only generated as times series data (e.g., from a simulation run), but can also belong to an ensemble (e.g., from a parameter study). Putting different volumes into relation for visual analysis or difference quantification becomes increasingly important and is an active field of research. Here, one promising approach is to create transformations between volumes to both localize and (both visually and numerically) quantify differences. While numerous approaches for volume warping and morphing have been presented, they are not applicable automatically to arbitrary volumes. They are often targeted toward a specific application scenario allowing only certain types of transformations, and require domain knowledge (e.g., via the detection of a skeleton or certain features), and/or manual user selection. In particular, older approaches typically resort to cross-dissolving to be able to generate smooth transitions. Instead, to be able to handle arbitrary volumes, we regard volumes directly as density distributions that are transformed into each other. This may be formulated as a transportation problem (i.e., an individual assignment of each mass unit in the volume distribution), which is typically solved via linear programming. However, their high computational complexity prohibits the direct handling of high-resolution volumes.

Consequently, these previous approaches are not able to generate high-quality transformations between arbitrary high-resolution volumes with no previous knowledge or user adjustment required. To fill this gap, we present a progressive algorithm that quickly computes meaningful transformations between high-resolution volumes (i.e., the mapping of mass/density elements from one volume to the other), iteratively refining by exchanging assigned mass elements. To achieve high performance, we both employ bounding box-based acceleration and an implementation that is designed for parallel execution without the need for explicit synchronization. We demonstrate that our approach provides a useful basis for a diverse set of applications, including temporal reduction, temporal analysis, and similarity sorting of ensembles.

In the remainder of this paper, after a review of related work (Sec. 2), we then discuss our main contributions:

- we formulate the volume-to-volume transformation problem directly based on density distributions without requiring any domain knowledge or user guidance,
- and propose a progressive approach to solve it with a particular focus on efficient parallel execution (Sec. 3).
- On this basis, we introduce a new technique for the streaming selection of time steps in temporal data that guarantees the complete reconstruction within a user-specified error bound (Sec. 4).
- We evaluate the characteristics of our approach, apply it to various data sets and use cases, and compare against alternatives (Sec. 5).
- We then discuss its properties and limitations in Sec. 6.

We finally conclude our work in Sec. 7.

2 Related Work

Volume Warping, Morphing, and Deformation. Gomes et al. [19] define warping as a transformation on a graphical object, while morphing specifically depicts the warping between two objects. Most morphing or warping approaches beyond cross-dissolving are feature-based, i.e., they require prior automatic detection or manual placement of control points [29]. The quality and applicability of such techniques

• *Steffen Frey and Thomas Ertl are with the University of Stuttgart. E-mail: {steffen.frey||thomas.ertl}@visus.uni-stuttgart.de.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

strongly depends on geometric and topological properties [26]. Many early methods for volumes were ports of techniques for images [10]. Lerios et al. [29] first do a feature-based warping between two input volumes (an extension of Beier and Neely's [6] image warping technique to 3D), and then rely on blending the resulting warped volumes to achieve a smooth transition. User tweaking is required to ensure appropriate intermediate objects. He et al. [21] morph between two volumetric data sets via interpolation in the wavelet domain. Bai et al. [3] present an registration-based segmentation method via elastic warping of surfaces to match the anatomy of mice. Fang et al. [13] introduce landmark-based volume deformation, whereas landmarks represent the important geometric and/or biological features (we compare against an implementation of this approach in Sec. 5.4). Lin et al. [30] discuss load balancing for distributed volume morphing and rendering. Rhee et al. [35] create anatomically accurate deformations of body regions by forming correspondences via template volumes. For instance, Correa et al. [11] transform volumetric objects to generate illustrative visualizations, employing physically inspired constraints in the process. Balmelli et al. [5] use volume warping to optimize the extraction of isosurfaces by emphasizing areas of interest.

Surface morphing establishes a correspondence between two surfaces to interpolates in-between. However, issues arise with surfaces if they differ in topology. Liu et al [31] focus on improving geometric consistency. Staten et al. [42] compare mesh morphing methods for 3D shape optimization. For automatic design optimization, Sieger et al. [40] employ triharmonic radial basis functions to morph meshes from simulation into their underlying CAD model. Sanchez et al. [37] employ user-controlled metamorphosis between functionally-based shape models for morphological shape design. For sparse volumetric deformation, Willcocks and Li [46] approximate topology through with the goal to improve the robustness of volume skeletonization.

EMD and the Transportation Problem. To the best of our knowledge, our proposed technique is the first to directly employ the transportation formulation for generating transformations between pairs of volumes. It is related to the earth mover's distance (EMD, also known as the Wasserstein metric), and determines the minimum cost of turning one distribution into the other. It is particularly popular in computer vision to quantify distances between color histograms, e.g., for image retrieval [36]. For rendering, Bonneel et al. [7] decompose distributions (like BRDFs, environment maps, stipple patterns, etc.) into radial basis functions, and then apply partial transport that independently considers different frequency bands. These applications only consider distributions consisting of a comparably small number of elements. The EMD basically involves a transportation problem that is typically solved via linear programming. A popular approach with numerous variations is the cost-scaling push-relabel algorithm [18]. It maintains a preflow and gradually converts it into a maximum flow by moving flow locally between neighboring vertices using push operations under the guidance of an admissible network maintained by relabel operations. In Sec. 5.3, we evaluate and compare our approach against an efficient implementation of a minimum-cost flow algorithm [20]. While it finds an optimal solution (i.e., one with minimal associated cost), the involved polynomial complexity makes it infeasible to directly transform high-resolution volumes. Another approach is to use a heuristic to find a valid solution quickly. A commonly used method to efficiently determine approximate solutions is Vogel's Approximation Method (VAM) [34]: it iteratively creates assignments for the source or target element that exhibits the largest discrepancy between the best and the second best solution. However, it does not perform very well in terms of quality for our volume transformation problem (cf. Sec. 5.3). Loosely related, Optical Flow determines a field of 2D displacement vectors moving points between frames to accommodate for changes from object or camera movement (e.g., [8]). It relies on several assumptions, including static pixel intensities and the similarity of motion of neighboring pixels, none of which apply to our approach.

Visualization and Processing of Volume Data Sequences. Bach et al. [2] review temporal data visualization techniques as operations performed on a space-time cube. In their taxonomy, we fit into "filling" (sub-category of "time interpolation"), i.e., the transformation of a set of disconnected space-time objects into a fully connected space-time object. Woodring and Shen presented Chronovolumes, a static, direct rendering technique that integrates all time-varying data over time into one volume [47]. Jang et al. [24] propose functional representations and an efficient encoding technique to improve the rendering performance with time-varying data. Balabian et al. [4] employ temporal transfer functions and compositors, e.g., to highlight areas of high change. Lee and Shen [27] identify and visualize trend relationships in multivariate time-varying data. Frey et al. [16] employ similarity matrices to detect and explore similarity in the temporal variation of field data.

While we directly compare density distributions, a large body of work in time-dependent volume visualization is based on the extraction of features. Widanagamaachchi et al. [45] employ feature tracking graphs. Lee and Shen [28] use time activity curves (TAC) to visualize time-varying features and their motion. TACs were also used by Fang et al. [14] to analyze volume differences for medical applications. Wang et al. [44] derive an importance curve for each data block based on conditional entropy, characterize its temporal behavior, and cluster curves to classify the underlying data. Silver et al. [41] isolate and track representations of regions of interest. Schneider et al. [38] compare scalar fields based on contours. Lu and Shen [32] propose interactive storyboards composed of volume renderings and descriptive geometric primitives generated via data analysis. Our approach can quantify the difference between volume data sets, which could be utilized for comparative visualization (e.g., via integration into VisTrails [9]).

On the basis of our volume transformation technique, we propose an application for the temporal reduction of volume data sets. Tong et al. [43] compute the distance between data sets via different metrics (among others, using the EMD to assess the similarity of clouds in longitudinal direction), and employ dynamic programming to select interesting time steps. Wang et al. [44] partition the time range into uniformly sized segments and select one time step from each segment based on previously extracted importance values. For in-situ visualization, Fernandes et al. [15] use an intermediate abstraction of raycasting samples (composited clusters of samples along view rays), and use an efficient representation to store time-dependent simulation data.

Algorithm 1 Overview on our progressive volume transformation approach (‖ depicts parallel execution).

```
1: procedure VolTrans (Sec. 3)                          ▷ (Sec. 3)
2:     create initial assignment                        ▷ (Sec. 3.1)
3:     while termination criterion not met do           ▷ (Sec. 3.4)
4:         ‖ generate exchange plan                     ▷ (Sec. 3.2)
5:         ‖ execute exchange plan                      ▷ (Sec. 3.3)
6:     create intermediate volumes, if required         ▷ (Sec. 3.5)
```

3 Parallel Volume Transformation

We directly compute the transformation between two volumes by considering them as (3D) distributions. These distributions can either be taken directly via their (density) values, or may undergo a transfer function mapping ($\mathbb{R} \to \mathbb{R}$) first. In the following, we assume that they are represented by sets of discrete points A and Ω, e.g., from CT scans, simulations, etc. Arbitrary (continuous) data can be used via resampling. We denote samples $\alpha \in A$ as source samples, and samples $\omega \in \Omega$ as target samples. Each sample has an attached position $p(\cdot)$ and mass $m(\cdot)$. For now, we assume a balanced problem ($\sum_{\alpha \in A} m(\alpha) = \sum_{\alpha \in A} m(\omega)$). We then determine a weighted mapping $F : A \to_M \Omega$ from A to Ω. The associated cost of such an assignment F is determined based on the Euclidean distance $d(\alpha, \omega) = |p(\alpha) - p(\omega)|$ between samples:

$$\gamma(F) = \sum_{\alpha \in A} \sum_{\omega \in F(\alpha)} m_{\alpha \to \omega} \cdot d(\alpha, \omega)^2, \tag{1}$$

with $m_{\alpha \to \omega}$ denoting the respective weight (or mass) associated with an individual assignment. On this basis, the goal of our approach is to determine a mapping F to that yields the minimum for $\gamma(\cdot)$.While F can be used to generate transitions between volumes, $\gamma(F)$ quantifies the distance between them. We normalize the masses (such that $\sum m = 1$)

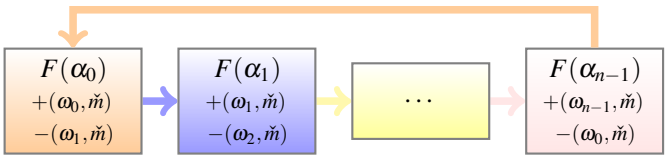


Fig. 2: Planning creates exchange sequence for source elements α (Sec. 3.2). During exchange (Sec. 3.3), for each α one target element ω is determined with associated mass $\check{m}$, and passed around circularly.

and the positions (cf. Sec. 3.1) to improve the expressiveness of $\gamma(\cdot)$. For instance, the distance of a Dirac delta function on the left and the right border of the domain results in $\gamma(\cdot) \approx 1$, independent of the extent of the domain or the height of the pulse. Eq. 1 can be regarded as an instance of the transportation problem [1]. While there are different types of linear programming approaches to solve such problems, they do not fulfill our requirements regarding quality or performance for transforming high-resolution volumes (cf. Sec. 5.3).

Therefore, we propose a new approach (outlined in Alg. 1) that exhibits useful properties for various applications (cf. Sec. 6). We first quickly generate an already valid assignment F (Sec. 3.1). Our algorithm works on the basic idea of source samples α swapping assigned target samples ω. We aim to accomplish this such that it can be run efficiently in parallel without requiring atomics or mutexes.For this purpose, we first generate an exchange plan consisting of mutually exclusive subsets of A (Sec. 3.2). We then execute this plan, determining if and which target nodes to exchange, and carrying out the respective transfer (Sec. 3.3). Different plans are generated in parallel without mutual conflicts (however, less recently generated plans typically lead to fewer exchanges, cf. Sec. 5). Our algorithm carries out multiple iterations of planning and exchange, until some kind of termination criterion is reached (Sec. 3.4). Finally, we create intermediate volumes from determined assignment F (Sec. 3.5).

3.1 Initialization

To generate a valid initial assignment, we first account for different total masses in the source and target volume (i.e., in general $m(A) = \sum_A m(\alpha) \neq m(\Omega) = \sum_\Omega m(\omega)$, we represent masses as integers to avoid issues due to floating point arithmetic). To equalize the total sum of masses $m(A)$ and $m(\Omega)$, we add $|m(\Omega) - m(A)|$ virtual mass elements to the side with the lower value (we assume $m(A) < m(\Omega)$ for the discussion below). Sampling is based on the mass distribution $m(\cdot)$, i.e., the probability of taking a sample from element $\alpha \in A$ is proportional to its mass $m(\alpha)$. We memorize how many elements have been added to each α, and during interpolation, we gradually reduce its contribution to zero (cf. Sec. 3.5). For the evaluation (Eq. 1), virtual mass elements are treated like standard mass elements. We then generate an initial assignment F by creating a shuffled list of all $\alpha \in A$ and randomly assigning elements from Ω to α until $m(\alpha)$ is matched. We assign the respective ω with as much mass as possible, i.e., $\max(m^*(\alpha, m^*(\omega))$, where $m^*(\cdot)$ depicts the mass that has not been assigned yet.

While we concentrate our discussion in this paper on regular grids, our approach works directly on samples (or points) and imposes no restrictions on the underlying data representation (apart from reconstruction, cf. Sec. 3.5). For regular grids with resolution $r = (r_x, r_y, r_z)$ and voxel extents $\delta = (\delta_x, \delta_y, \delta_z)$, we initialize respective element positions from grid cells as follows. We compute the maximum domain extent via $e_{\max} = \max((\delta^A, \delta^\Omega) \times (r^A, r^\Omega)^T)$, and initialize the position of each α with index $\chi = (x, y, z)$ by $p(\alpha) = \chi \times \delta^T / e_{\max}$ (ω accordingly). This transformation of positions is reverted when generating the intermediate volume (cf. Sec. 3.5). Arbitrary volume representation could be handled either via regular resampling or a combination of inverse transform sampling (for continuous representations) and density estimation for reconstruction. While the alignment of volumes only has negligible impact on the resulting assignment F according to our experiments, naturally it heavily influences our distance metric $\gamma(F)$ (cf. evaluation in Sec 5.1). This needs to be considered in cases that rely on this measure for comparison, but have no natural alignment, e.g., due to differences and shifts during data acquisition (cf. Sec. 5.5).

Algorithm 2 Generating sequences of source elements $\alpha \in A$ that are candidates for the exchange of mass assignments (depicted in Fig. 2).

```
1:  procedure PLAN (SEC. 3.2)
2:    A ← randomShuffle(A)                          ▷ shuffle A
3:    P ← {}, C ← {}                                ▷ initialize plan P and candidates C
4:    for all α ∈ A do
5:      if C = {} then
6:        C ← {α}
7:        continue
8:      Δ ← 0, c' ← α, b ← None
9:      for all c ∈ C← do                           ▷ traverse C in reverse order
10:       Δ ← Δ + δ_bbox(c, c')                     ▷ estimated improvement from c to c'
11:       if Δ + δ_bbox(α, c) < 0 then
12:         P ← P + {c, c', ..., back(C), α}        ▷ add sequence to plan P
13:         C ← {}
14:         break
15:       else if Δ < 0 then
16:         b ← ExtendCandidates
17:       c' ← c
18:     if b = ExtendCandidates ∧ C ≠ {} then
19:       C ← C + α
20: procedure δ_BBOX(α_0, α_1)                      ▷ estimate transfer improvement α_0 → α_1
21:   (Δ_0^min, Δ_0^max) ← d_minmax(α_0, bbox(α_0))
22:   (Δ_1^min, Δ_1^max) ← d_minmax(α_1, bbox(α_0))
23:   return mix(Δ_1^min, Δ_1^max, rng_[0,1]) − mix(Δ_0^max, Δ_0^min, rng_[0,1])
```

3.2 Exchange Plan

Next, we generate an exchange plan P consisting of mutually exclusive subsets of A that specify which αs are considered for an exchange (and in which order). These then determine the most beneficial target nodes ω to circularly transferred to the next node (Fig. 2). All exchanges take place with $\check{m}$, which is determined the smallest respective mass for any ω. Alg. 2 gives an overview of our approach. Initially, we shuffle A (Line 2) to be able to create arbitrary sequences. Looping over all elements $\alpha \in A$, we then generate potential swap sequences (Line 4). In each iteration of our main loop (Lines 4–19), we determine whether the current α either completes our candidate set to a sequence with a potential transfer of points (Lines 11–14), or at least is a promising extension to our current candidate list $C \subset A$ (Lines 15–16). For this, we iterate over C in back-to-front order (Line 9). Here, we iteratively estimate the potential benefit Δ of transferring one mass element from $c \in C$ to the subsequent source node $c' \in C$ (Line 10). This takes into account the bounding box of the positions of all target nodes $\omega \in F(\alpha)$ currently assigned to α (Lines 20–23). Here, the difference of the maximum possible distance of α_0 to its own bounding box $\text{bbox}(\alpha_0)$ and the minimum possible distance of α_1 to $\text{bbox}(\alpha_0)$ would give the theoretically highest improvement when transferring a mass element from α_0 to α_1. Instead, we randomly choose a number in the range between the minimally and maximally possible distance to a position in the bounding box to avoid cases in which creating plans for a certain longer (necessary) sequence is prevented by always taking (eventually unsuccessful) shorter ones (Lines 21 & 22). Using this in the back-to-front loop over the candidate sequence C, we check the value for the wrap-around assignment of α to c together with the sequence evaluation Δ. This allows us to determine whether we already found a promising candidate set (Line 11). Should this be the case, we add the respective subsequence of C together with α as one sequence to our plan P (Line 12), and exit the candidate loop. Second, if at least the sequence without the wrap-around is promising (Line 15), we indicate that α will be added to our candidate set if the sequence cannot be completed in this iteration (Line 16, Lines 18 & 19). We run multiple (full) instances of PLAN in parallel to utilize multi-core CPUs without having to partition A and Ω. As we can generate arbitrary swap sequences of arbitrary length, our approach will eventually achieve the optimal assignment (i.e., the one with minimum associated cost $\gamma(\cdot)$).

Algorithm 3 Checking sequence lists p of plan P w.r.t. whether distance $\gamma(\cdot)$ is improved, and carrying out the exchange (cf. Fig. 2).

```
1:  procedure ExchangeList (Sec. 3.3)
2:    for p ∈ P do                                   ▷ this can be run in parallel
3:      Δ ← 0, m̆ = ∞                                 ▷ initialize exchange value Δ and mass m̆
4:      for i ∈ {0 ... |p| − 1} do
5:        (ω_i, d_i, m_i) ← σ(F(α_{p(i−1)}), α_{p(i)})   ▷ find best candidate
6:        m̆ ← min(m̆, m_i)
7:        Δ ← Δ + d_i
8:      if Δ < 0 then                                ▷ check if exchange is beneficial
9:        for i ∈ {0 ... n − 1} do                   ▷ exchange points
10:         F(α_i) ← F(α_i) + (ω_i, m̆) − (ω_{(i+1) mod n}, m̆)
```

3.3 Assignment Exchange

Next, we process the exchange sequences $p \in P$ created during planning (Alg. 3). For each p (Line 2), we first determine which ω to exchange for each contained source element $\alpha \in p$ (Lines 3–7). To determine the best candidate from $F(\alpha_{p(i-1)})$ for $\alpha_{p(i)}$, we iterate over all elements in $F(\alpha_{p(i-1)})$ to find the one with the smallest distance d_i ($\sigma(F(\alpha_{p(i-1)}), \alpha_{p(i)})$, Line 5). We also determine pointers where to potentially insert new elements as well as the minimum value of all assigned masses $\breve{m}$ (Line 6). Second, if the overall exchange has been determined to be beneficial (i.e., $\Delta < 0$, Line 8), we exchange the respective ωs (Line 9 & 10). We employ arrays instead of a lists in our implementation, which, while potentially inefficient in terms of space, is better both for performance and portability. In an exchange, if the respective ω_i is already present, we simply add up the masses. Otherwise, if the outgoing ω_{i+1} left open an empty slot (or if another one has been found), we fill it with ω_i, or else we place ω_i right after the last entry. The bounding boxes used by $\delta_{\text{bbox}}(\cdot,\cdot)$ during planning are also updated in this process (omitted for clarity in Alg. 3).

3.4 Termination Criterion

Planning and exchanges iteratively refine the assignment $F(\cdot)$ until we fulfill a certain termination criterion. For this, we implemented different sub-criteria that may be used in combination, whereas at least one needs to be fulfilled to exit the iteration loop.

Time. The specified overall time limit is reached.
Distance. The distance metric falls below a certain target value.
User Interrupt. A user may interrupt refinement at any time.

Different criteria are useful for different application scenarios. **Time Limit** allows to comply to time budgets in batch scenarios where many different morphing operations need to be carried out (e.g., for in-situ scenarios with concurrent simulations). In interactive scenarios, **User Interrupt** allows a user to quit refinement if the achieved results suffices for his purposes. **Distance Limit** is useful to check whether a certain distance between two data sets is exceeded (e.g., to assert that we obey a target distance to the original after reduction, cf. Sec. 4).

3.5 Reconstruction of Intermediate Volumes

For reconstructing intermediate volumes, we do a linear transformation with parameter $\sigma \in [0,1]$ along the direct path of assignment F from A to Ω ($F_{\sigma=0} = A$ & $F_{\sigma=1} = \Omega$). As discussed so far, our approach works directly on samples and does not impose any restrictions on underlying grid types. However, for reconstruction, we concentrate on regular grids as they allow for the linear interpolation between resolution (r_x, r_y, r_z) and voxel spacing $(\delta_x, \delta_y, \delta_z)$. Arbitrary data representations could be handled in the same way via regular resampling. Alternatively, density estimation techniques could be employed to reconstruct a volume from arbitrarily distributed samples (this remains for future work). With our linear interpolation approach, the resolution of an intermediate volume is given by $(1-\sigma)(r_x^0, r_y^0, r_z^0) + \sigma(r_x^1, r_y^1, r_z^1)$ (δ analogously). Based on this, we then reconstruct a regular grid from what is essentially a set of weighted points that is given by the positions in A and Ω and assignment F. To conform with tri-linear volume interpolation, we employ a hat filter with the extent of a cell. Its magnitude is scaled by the respective weight w of the assignment.

Algorithm 4 Selection of time steps S such that the distance of time steps in between to the intermediate volume is smaller than τ_{dist}. Different metrics may be used for distance function Δ. Here, we alternatively employ our flow-based distance metric Δ_{trans} or the RMSE Δ_{RMSE}.

```
1:  procedure TimeSeriesReduction (Sec. 4)
2:    S ← {0}, t ← l, ⊥ ← 0, ⊤ ← ∞
3:    while back(S) ≠ |T| − 1 do
4:      F ← VolTrans(back(S), t, τ_time)                ▷ time limit τ_time
5:      for all t* ∈ shuffle({back(S) + 1, ..., t − 1}) do
6:        t_morph ← F_{σ=(t*−back(S))/(t−back(S))}      ▷ morph in back(S) to t
7:        ▷ Δ_trans(t_0, t_1, χ) : γ(VolTrans(t_0, t_1, χ)) (χ: termination)
8:        ▷ Δ_RMSE(t_0, t_1) : sqrt(Σ_{i∈{0...|t_0|−1}} (m_i(t_0) − m_i(t_1))^2)
9:        if Δ(t_morph, t*, τ_dist ∨ τ_time) > τ_dist then   ▷ Δ_trans or Δ_RMSE
10:         ⊤ ← t
11:         break
12:     if ⊤ ≠ t then
13:       ⊥ ← t
14:     if ⊥ + 1 = ⊤ then      ▷ add last conforming time step to selection
15:       S ← S + (max(⊥, back(S) + 1)
16:       ⊥ ← back(S)
17:       l ← 0.8 · (back(S) − back_2(S))   ▷ adjust l via last two selections
18:     else if ⊤ ≠ ∞ then
19:       t ← (⊥ + ⊤)/2
20:     else
21:       t ← t + l
```

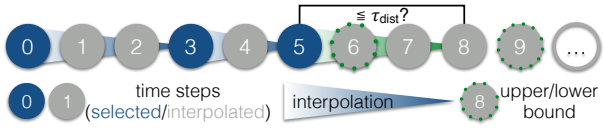


Fig. 3: Temporal selection of time steps such that time steps in between are covered by transformed volumes with a deviation bounded by τ_{target}.

4 Error-bound Time Step Selection

Our volume transformation algorithm can (1) interpolate and (2) provide meaningful distances $\gamma(\cdot)$ between data sets. We use both properties to select a set of time steps S from which the whole time series can be reconstructed without exceeding a user-defined error bound τ_{dist}. This allows to save storage space, and can also indicate the progression characteristics of a time series. Missing time steps $t \notin S$ are replaced by interpolating between surrounding time steps in S (Fig. 3). Our algorithm selects time steps S from a time series T on the the fly, and thus could be employed in streaming and in-situ scenarios (Alg. 4). We initialize S with the first time step, and choose an initial step size l (Line 2). We then loop over the remaining time time steps (Line 3). In each iteration, we first generate a transformation F from the last time step of the selection (back(S)) to the current time step t (Line. 4). We terminate the refinement when time budget τ_{time} is exceeded. We now check the distance of all time steps between back(S) and t to their respective intermediate (Line 5–21). For this, we generate a transformation t_{morph} for each time step t^* we compare against (Line 6). Different metrics may be used depending on the requirements: while our transformation-based distance metric Δ_{trans} (Line 7) is more expressive than element-based distances in particular for larger deviations or thin surfaces (cf. Sec. 5.2), Δ_{RMSE} (Line 8) operating on individual elements is a popular standard metric that is faster to compute. The obtained distance Δ is then compared against the maximally allowed target value τ_{dist} (Line 9). In case of using Δ_{trans}, we exit refinement if time budget τ_{time} is exceeded or if $\gamma(\cdot)$ falls below target distance τ_{dist}.

In our implementation, we do not proceed time step by time step (as in streaming data scenarios), but use an adaptive step size l. This saves transformation computations (and hence processing time) particularly when only few time steps are selected. If τ_{dist} is exceeded for any time step between back(S) and t^* (Line 9), we employ bisection to determine the latest time step that does not violate our criterion (it then becomes the next selected time step). For this, we maintain a lower

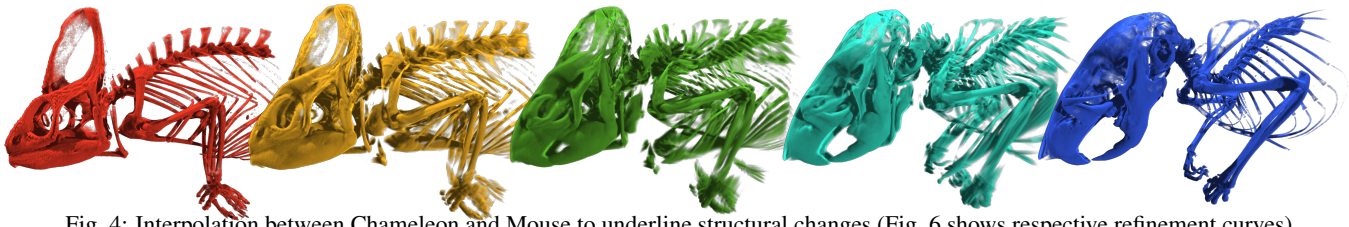


Fig. 4: Interpolation between Chameleon and Mouse to underline structural changes (Fig. 6 shows respective refinement curves).

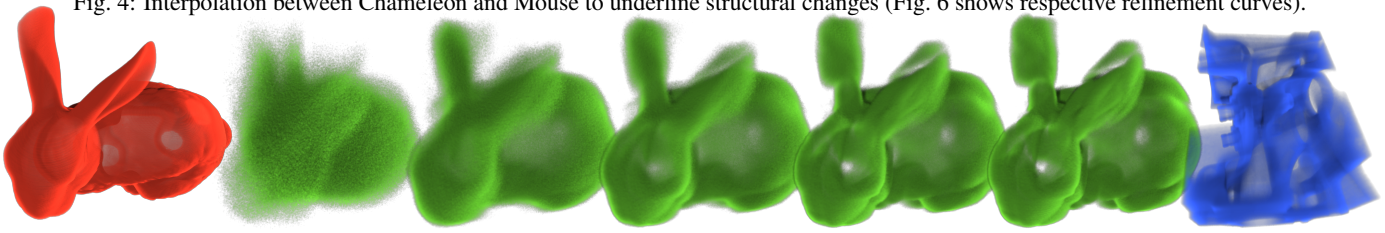


Fig. 5: Transformation from the Bunny (red, $\sigma = 0$) to the Engine (blue, $\sigma = 1$). Green volumes at $\sigma = 0.5$ show different stages of refinement.

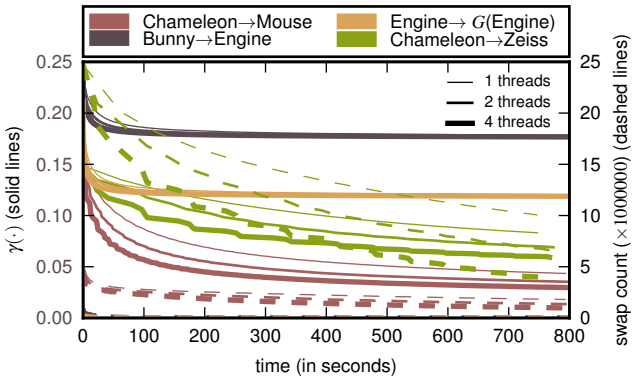


Fig. 6: Refinement of transformations between different data sets.

bound $\perp$ and an upper bound $\top$ that give the latest acceptable and the earliest not acceptable time step determined so far (Lines 10 & 13). We continuously bisect this time range (Line 19) until $\perp + 1 = \top$ (Line 14). We then add $\perp$ to the selection S (Line 15), and initialize the step size to slightly less than the latest gap between selected time steps (Line 17).

5 Results

For our evaluation, we used a machine equipped with a Intel Core i7-4770 CPU (3.4 GHz) featuring four cores and 16 GB of RAM. Our renderings were generated with a CUDA-based raycaster with local lighting and direct shadows. We used a variety of different data sets from CT scans and simulations. Our used three data structures: positions ($p(\cdot)$ of A and Ω), deltas (added mass for equalization to $\alpha \in A$ or $\omega \in \Omega$, cf. Sec. 3.1), and assignments (F). The assignment of each $\alpha \in A$ consists of a list of pairs (ω, m), and its storage is typically most memory-intense. Depending on the data, we use a 4 byte or 8 byte representation for storing indices and masses. In the following, we evaluate the transformation between volumes in detail (Sec. 5.1), compare against element-based, transport-based, and feature-based alternatives (Secs. 5.2, 5.3 & 5.4), and demonstrate the utility of our approach for similarity ordering and time series reduction (Secs. 5.5 & 5.6).

5.1 Transformation between Data Pairs

Our approach creates transitions between arbitrary volumes directly, automatically, and without prior knowledge. Next, we discuss its characteristics with data from CT scans to which suitable transfer functions were applied: Chameleon ($1024^2 \times 1080, \delta_{xy} = 1, \delta_z \approx 1.38$, Figs. 1& 4, left (w/ different transfer functions)), Mouse ($1024^2 \times 487, \delta_{xy} = 1, \delta_z \approx 1.38$, Fig. 4, right, both from the UTCT data archive), Zeiss ($680^3, \delta_{xyz} = 1$, Fig. 1,right, Daimler AG), Bunny ($512^2 \times 360, \delta_{xyz} = 1$, Fig. 5, NLM), and Engine ($256^3, \delta_{xyz} = 1$, Fig. 5, General Electric).

Qualitative Results. The Chameleon→Zeiss transformation shows that our approach is able to create smooth transitions between volumes that differ fundamentally in structure (Fig. 1). Structures slowly and flexibly transform to create new shapes, e.g., how an eye of the Chameleon becomes a hole of the Zeiss workpiece, and a combination of its ear and back morph into a cylinder. Naturally, we can also apply our approach to create transitions between data sets that have some kind of structural similarity, as exemplified by the Chameleon→Mouse transformation (Fig. 4). While our approach is not aware of any underlying semantics, we can particularly see a smooth deformation between skulls, that can already be useful to emphasize the structural deformation. However,the lack of such restrictions also allows parts of the bones that initially belonged to the tongue of the chameleon now become part of a leg of the mouse. Hence, while allowing arbitrary transitions yields high flexibility, some additional constraints could be desirable for a variety of use cases. For the Chameleon→Mouse transition, such a restriction could be based on anatomy-aware registration, only allowing certain parts of the skeleton to be transformed into each other, but not across. These extra restrictions could be additionally considered by our approach, yet this remains for future work. Next, we exemplify that with increasing time for refinement, the transformation becomes less and less blurry and iteratively forms clear structures (Fig. 5). In this example, $t = 0s$ shows the result directly after initial assignment. After 8 seconds, we already see distinct transformation shapes, yet they still look quite blurry. After 64 seconds a lot more refined result is achieved, and with $t = 256s$ a state is reached that is visually very close to the "converged" result after a long run time.

Refinement and Performance. For different data sets, the relative improvement in quality $\gamma(\cdot)$ is very similar in its rapid convergence, yet the actual rate differs due to the different workloads induced (Fig. 6). While the Bunny→Engine transformation sees only smaller changes after 64 seconds (cf. Fig. 5), it takes roughly three to four times as long for Chameleon→Mouse (Fig. 4) to reach that state. The swap count (the number of executed exchanges) declines similarly, i.e., less exchanges occur per iteration when closer to convergence. The timings of the individual stages of our algorithm roughly linearly dependent on the number of source and target elements ($|A|$ and $|\Omega|$, respectively): Chameleon→Zeiss: $|A| \approx 83M, |\Omega| \approx 70M$; Chameleon→Mouse: $|A| \approx 14M, |\Omega| \approx 6M$; Bunny→Engine: $|A| = 3M, |\Omega| \approx 1M$. The initialization time ranges between a couple of seconds for Bunny→Engine and 50 seconds for Chameleon→Mouse. The most time-consuming step of our approach is planning (i.e., the compilation of sequences that are promising candidates for a swap). Overall, as indicated by Fig. 6, the number of of exchanges/swaps decreases during the course of refinement, and directly with this also the total time to execute them. Inversely, the time for planning increases (by approximately 20%) due to the larger candidate set C that needs to be considered to find a promising exchange sequence (cf. Alg. 2). For Bunny→Engine, the planning in the beginning takes on average ≈ 250 ms for each iteration

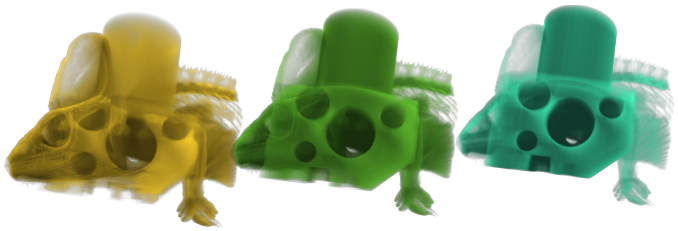


Fig. 7: Blending/Cross-dissolving results for Chameleon to Zeiss transformation for $\sigma = 0.3, 0.5$ and 0.7 (cf. Fig. 1 for our transformation).

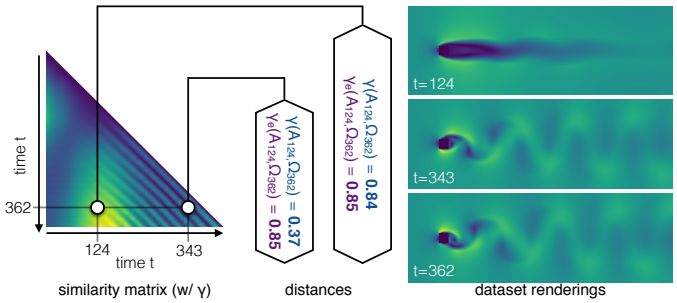


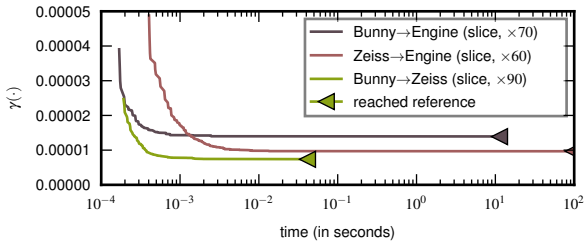
Fig. 8: Difference between our transformation-based $\gamma(\cdot)$ and the element-base distance measure $\gamma_e(\cdot)$ for three time steps of the von Kármán vortex street (distances are normalized w.r.t. the maximum).

(i.e., the generation of four plans in parallel takes about one second), while the actual exchange takes ≈ 15 ms in the beginning (changing to ≈ 270 ms and ≈ 5 ms toward the end). For Chameleon→Mouse, planning takes on average around ≈ 1.25 s in the beginning, while the exchange takes ≈ 75 ms (this changes to ≈ 1.5 s and ≈ 18 ms during refinement). Chameleon→Zeiss is our most costly transformation: the planning initially takes ≈ 37 s seconds (increasing to ≈ 40 s), and the exchange takes ≈ 7 s (decreasing to ≈ 160 ms). The chart also exhibits some small-scale alternating ("wiggling") behavior in the exchange counts for multi-threaded runs as plans are generated in parallel, and the more exchange iterations have taken place since planning, the less swaps occur (exchange processes happening in between can make some exchange sequences p obsolete). This also has a significant impact on thread scaling characteristics. Due to many of these interferences happening early, scaling behavior is hampered in the beginning (e.g., reaching $\gamma(\cdot) = 0.08$ for Chameleon→Mouse takes ≈ 237 s for one thread, ≈ 170 s for two threads, and ≈ 134 s with four threads). Yet, during the course of refinement a decreasing number of these conflicts occurs, and we approach linear scaling (for $\gamma(\cdot) = 0.027$, ≈ 4000 s for one thread, ≈ 2136 s for two threads, and ≈ 1233 s with four threads).

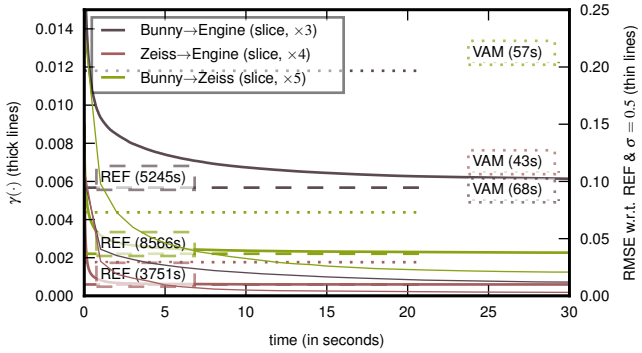
5.2 Element-based Distances and Interpolation

In contrast to our approach, element- (or cell-)based transitions consider each voxel individually without taking into account any type of local or global neighborhood. First, we compare our transformations against cross-dissolving, before we look at distance computations. Basically, cross-dissolving produces a linear interpolation between corresponding pixels or voxels of two data sets. Respective morphing sequences can create smooth transitions if the two involved data sets are reasonably similar and well-aligned. It is used for artistic as well as scientific purposes. For instance, Chimera, a popular visualization system for exploratory research and analysis, uses it to generate so-called Morph Maps between related data sets with the same underlying grids [33]. However, even if extended to arbitrary grids, this cannot capture translational motion adequately, which is required for larger changes. Fig. 7 exemplifies this by attempting the recreate the transition from Fig. 1 via fading out source elements A and blending in target elements Ω with increasing σ. It shows that for distinct data sets, cross-dissolving does not create the impression of a smooth transformation (e.g., for $\sigma = 0.5$, the two volumes can clearly be distinguished).

To compare our transformation-based against an element-based distance measure, we consider the velocity magnitude of a von Kármán vortex street from a CFD simulation (Fig. 8). A von Kármán vortex



(a) Refinement until reaching the optimum (triangle).



(b) Refinement against optimum (ours: solid, REF: dashed, VAM: dotted).

(c) Results with our and simplex method for Bunny→Engine (visually identical).

(d) Results with Vogel's approximation method for Bunny→Engine.

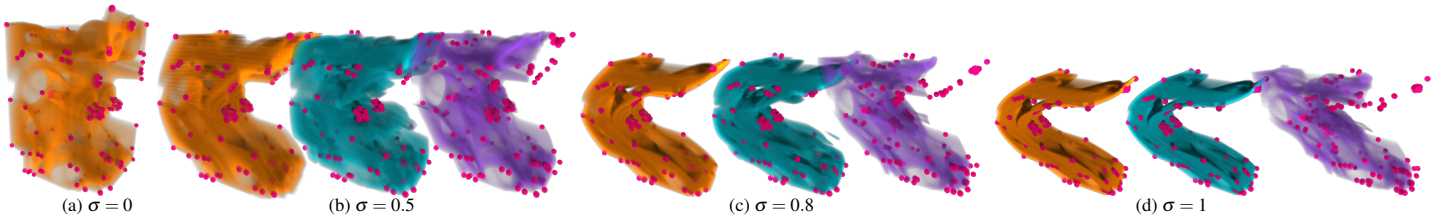
Fig. 9: Transformation between downsampled slices of Bunny, Engine, and Zeiss, for (a) demonstrating convergence until optimum, and (b) comparing against the Simplex Method (REF) and VAM . $\gamma(\cdot)$ gives the respective distance between data sets, while RMSE is computed w.r.t. the optimal solution of REF for reconstructed intermediate $\sigma = 0.5$. Intermediates are shown for Bunny→Engine (c,d).

street is a repeating pattern of swirling vortices caused by the unsteady separation of fluid flow around blunt objects. Here, time steps $t = 0$ to approximately $t = 200$ cover the buildup phase, with recurrent behavior occurring afterwards. This also shows in the similarity matrix created by our approach (Fig. 8 left). We now analyze the qualitative differences of element-based and transformation-based distances via the similarity relationship between three points in time ($t = 124, 343, \&362$). Although visually and simulation state-wise, time steps $(343, 362)$ quite similar in comparison to $(124, 362)$, the element-based distance measure $\gamma_e(\cdot)$ yields roughly the same distance (both ≈ 0.85) because changes are only considered per voxel, and small translations cannot be distinguished from larger shifts. In contrast, our distance measure $\gamma(\cdot)$ reflects this difference in distance much more adequately (≈ 0.37 vs. ≈ 0.84). However, it also induces significant cost when looking at the computation of a complete similarity matrix, requiring a total of $400^2/2 = 80000$ transformations here ($\gamma(t_0,t_1) = \gamma(t_1,t_0)$). It took over a week on a single machine (terminating after 4096 refinement iterations), while the computation of per-voxel differences for the element-based metric is in the order of seconds to minutes.

5.3 Comparison against Transport-based Alternatives

Next, we evaluate our exchange-based technique against two fundamental approaches for solving the transportation problem.

Simplex Method (Reference) We employ a cost-scaling push-relabel algorithm: it maintains a preflow and gradually converts it into a maximum flow by moving flow locally between neighboring vertices



(a) $\sigma = 0$ (b) $\sigma = 0.5$ (c) $\sigma = 0.8$ (d) $\sigma = 1$

Fig. 10: Comparison between feature-based morphing (using Locally Affine Shepard (LASM), purple) and our transformation (blue) at the example of the Engine data set that is bent and contracted via transformation G (orange). Feature points employed for LASM are shown in pink.

γ	$x=-1$		$x=0$		$x=1$	
	keep shift	revert shift	keep shift	revert shift	keep shift	revert shift
$y=-1$	2.00539	1.37216e-5	1.00589	1.37216e-5	2.0064	1.37216e-5
$y=0$	0.999508	1.37216e-5	1.37216e-5		1.00052	1.37216e-5
$y=1$	1.99363	1.37216e-5	0.994136	1.37216e-5	1.99464	1.37216e-5

Table 1: Transformations F computed with shifted $\Omega_{x,y}$, distance $\gamma(F)$ given w.r.t. kept and reverted shift (w/ Bunny→Engine, cf. Fig. 9).

using push operations under the guidance of an admissible network maintained by relabel operations [18]. It finds an optimal solution (with minimal associated cost $\gamma(\cdot)$), and is regarded to be one of the most efficient solvers (we use Google's implementation [20]). Its complexity is $O(n^2 \cdot m \cdot log(n \cdot c))$, where $n = |A| + |\Omega|$ denotes the number of nodes in the graph, $m = |A| \cdot |\Omega|$ gives the number of edges, and c is the largest induced edge cost (i.e., the largest distance between positions $p(\alpha)$ and $p(\omega)$).

Vogel's approximation method (VAM) VAM is considered to yield good (initial) solutions that are close to the optimum for many transportation problems [23]. It works by taking into account the costs associated with each route alternative: for each $\alpha \in A$ and $\omega \in \Omega$, it iteratively computes the difference in cost between the best and the second best alternative and takes the largest one [34].

For evaluation, we consider transformations between Bunny, Zeiss, and Engine (Fig. 9). To allow for the computation of reference results, we only consider the downsampled middle slice of each data set. To evaluate the convergence properties of our approach, we first compute the reference solution via the Simplex method and let our approach run until it yields an optimal result (Fig. 9a). As indicated previously (e.g., Fig. 6), $\gamma(\cdot)$ rapidly improves in the beginning, with the rate of improvement decreasing significantly after a while, yet an optimal result is reached eventually. Using a smaller downsampling ratio, we investigate more closely our properties in comparison to VAM and the reference solution, both considering $\gamma(\cdot)$ (i.e., the distance between the two chosen data sets based on the currently best assignment F), as well as the root-mean-square error (RMSE) between the intermediate images for $\sigma = 0.5$ (Fig. 9b). Both metrics indicate that our approach quickly approaches the optimal solution in a matter of seconds, while it takes the Simplex Method a long time to reach a result (and becomes completely infeasible for larger resolutions). We also quickly surpass the quality achieved by VAM. While VAM generates close-to-optimal results for many transportation problems [23], the "dense" structure of our underlying transportation problem is probably unsuitable for this heuristic. This is also shown in the respective intermediate renderings, where the Simplex method and our approach (after a minute with four threads) produce visually identical results (Fig. 9c), while VAM produces rather "crumbly" intermediates (Fig. 9d). While, according to our experiments, the alignment has negligible impact on F, it has significant impact on the distance between volumes $\gamma(\cdot)$ (i.e., the assignments F remain almost unchanged for shifted volumes, cf. Tab. 1).

5.4 Comparison against Feature-based Morphing

Next, we compare against a feature-based morphing approach discussed by Fang et al. [13]. While feature points (or landmarks) are often selected manually, we employed the Shi and Tomasi method for automatic extraction [39]. We first compute the structure tensor for each voxel via derivatives obtained from the Sobel operator. From this, we then compute the eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ of this tensor. A feature is detected if $\min(\lambda_1, \lambda_2, \lambda_3) > 0.1$ (selecting corner points with significant changes in each direction). Applied to the Engine, this yields 166 feature points Θ (pink spheres in Fig. 10a). One critical part of feature-based approaches is the matching between feature points across data sets. To avoid introducing errors, we apply an analytic transformation G both to the Engine and its features:

$$G : \mathbb{R}^3 \to \mathbb{R}^3, (\alpha_x, \alpha_y, \alpha_z) \mapsto (\alpha_x^{14(\alpha_y - 0.5)^2 + 0.4}, \alpha_y, \alpha_z^{2(p.y - 0.6)^2 + 0.6}), \tag{2}$$

with $(\alpha_x, \alpha_y, \alpha_z)$ denoting 3D coordinates. Effectively, G both bends and contracts the volume (Fig. 10d *orange*). We consider three different morphing alternatives:

1. G is directly taken as assignment map F (Fig. 10 *orange*). Intermediate volumes are generated as outlined in Sec. 3.5.
2. Our transportation-based approach computes F (Fig. 10 *blue*).
3. Morphs (Fig. 10 *purple*) are computed via coordinate mapping on the basis of transformed markers $\Theta_\sigma = (1-\sigma) \cdot \Theta + \sigma \cdot G(\Theta)$. For this, we employ the Locally Affine Shepard Method (LASM) [13]. In brief, it assumes that a local affine interpolant exists for each feature point $\theta \in \Theta_\sigma$, and employs an energy minimization process to construct local interpolants. These interpolants are then used to map coordinates in the intermediate volume to coordinates in the source volume. The source volume is accessed with these coordinates to finally obtain a (morphed) value for each voxel.

First of all, looking at the final morph (i.e., $\sigma = 1$, Fig. 10d), it can be seen that LASM (purple) cannot match the rather complex (non-linear) transformation, despite the "reference"-quality feature points (it can only match the bending on the lower but not on the upper part). Furthermore, changes occur in the "density" of masses both due to the bending and the contraction: the transformed target volume (orange) is much denser than the source volume, as visually indicated via its transparency. However, this is not supported by LASM (or in fact any morphing method based on the transformation of coordinates), as can be seen by the unmodified level of transparency in Fig. 10d (purple). In contrast, by design, our transportation-based approach (blue) exactly matches the target volume (orange), and smoothly also interpolates the densities in between. The intermediate volumes for $\sigma = 0.5$ and $\sigma = 0.8$ (Fig. 10b and c) show that the LASM smoothly drifts away from the morph specified by G (indicated by the respective features points Θ_σ). Our transportation-based approach yields a results that is closer to the specified transformation G, yet it needs to differ as G does not provide the cost-optimal transformation from source volume A to target volume $\Omega = G(A)$ according to our criterion (i.e., there is a cheaper assignment F with $\gamma(F) < \gamma(G)$ to yield Ω from A).

5.5 Application Example: Similarity Ordering

To exemplify the utility of our approach for the generation of meaningful distances between volumes, we discuss an application to sort a data ensemble such that transitions between successive data sets are as smooth as possible. For this, we look at a set of flowers obtained via CT scans [22] (Fig. 11). We first moved the volumes such that their barycenter matches (from the positions weighted by their mass). On this basis, we then use our approach to compute a similarity matrix containing all mutual distances of the set of flowers (Fig. 11, left). From this, we can see that, among other factors, object rotation has quite some impact here. For instance, (d) and (g) have a comparably small distance, despite their differences in shape. On the contrary, (a) and

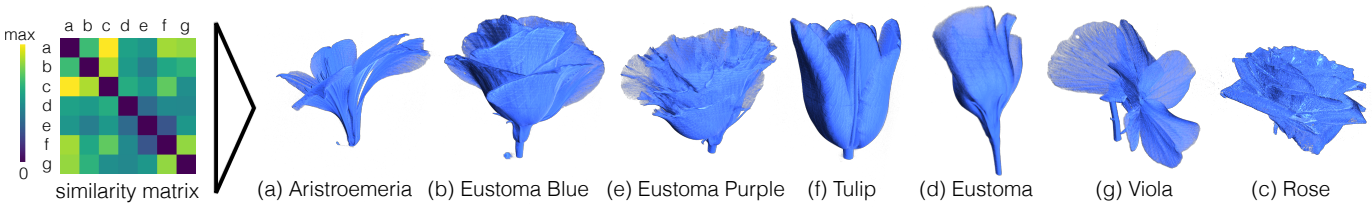


Fig. 11: Ordering of flowers to achieve minimal possible distances between successive data sets (data provided by Ijiri et al. [22]).

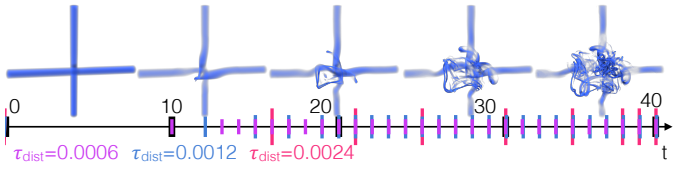


Fig. 12: Selections of time steps from the λ_2 data set for the purpose of demonstrating the adaptivity both w.r.t. the underlying data and τ_{dist}.

(g) have a relatively large distance, despite some similarities in shape. However, we can see that the visually most similar flowers also have a low distance, like (b) and (e). This similarity matrix is then used to determine an ordering by basically solving a traveling salesman-type problem (without return to the origin, Fig. 11, right). This provides a sequence in which each transition is comparably smooth, and many of the visually most similar pairs are next to each other (e.g., (f) and (d), (b) and (e)). While more elaborate registration could mitigate this, we do not necessarily end up with the same or similar types of flowers next to each other but rather the ones that are similar in their volumetric representation. For a more type-aware approach, certain features would have to be extracted, classified, and matched to, e.g., more expressively compare an open and closed type of Eustoma (e.g., (b) and (d), cf. [22]).

5.6 Application: Time Series Reduction

Next, we evaluate our time series reduction approach (cf. Sec. 4). First, we demonstrate the adaptivity of the selection at the example of the λ_2 data set (529^3, 40 time steps). It contains the λ_2 vortex extraction criterion on the basis of a fluid simulation (Fig. 12). Initially, the data degrades comparably slowly, and then increasingly quickly decomposes toward the end. Our selection adapts to this: the small gradual changes in the beginning can be covered adequately by fewer selected time steps, while for the rather chaotic behavior toward the end basically no time steps may be omitted with an acceptable error. Overall, a lower acceptable error τ_{dist} leads to a denser selection of the time series.

We now consider the Droplet data set (256^3 with 500 time steps). It was created with multiphase flow solver FS3D [12], and depicts two droplets colliding asymmetrically (Fig. 13; courtesy of C. Meister, Institute of Aerospace Thermodynamics, University of Stuttgart, cf. [25] for details). Here, we chose a comparably high distance threshold τ_{dist} to demonstrate both the high potential for data reduction, as well as the potential quality impact. The chart in Fig. 13 shows which time steps are selected, with the respective renderings on top. The distances to the reference have been computed in a separate run, as during the actual selection procedure, we are only interested in not exceeding the distance threshold τ_{dist}, and accordingly interrupt refinement as soon as this is assured. They are consistently below τ_{dist}, yet close to this limit in between (the last segment is an exception as the last time step is always selected, cf. Sec. 4). The distance of interpolates to the original data generally gets smaller the closer they are to a selected time step. In the bottom, we compare renderings of the reference time series in the middle between two selected time steps to our interpolated volume. This exemplifies the different types of deviation that can occur. For $t = 41$, the shapes are almost identical, yet the droplets do not quite move linearly towards each other, which means that there is a deviation in terms of position. Conversely, for $t = 241$ primarily the shapes differ: as we interpolate between a state with one connected component of mass ($t = 170$) and a state that has largely decomposed into smaller droplets ($t = 312$), also the interpolate will slowly decompose, although

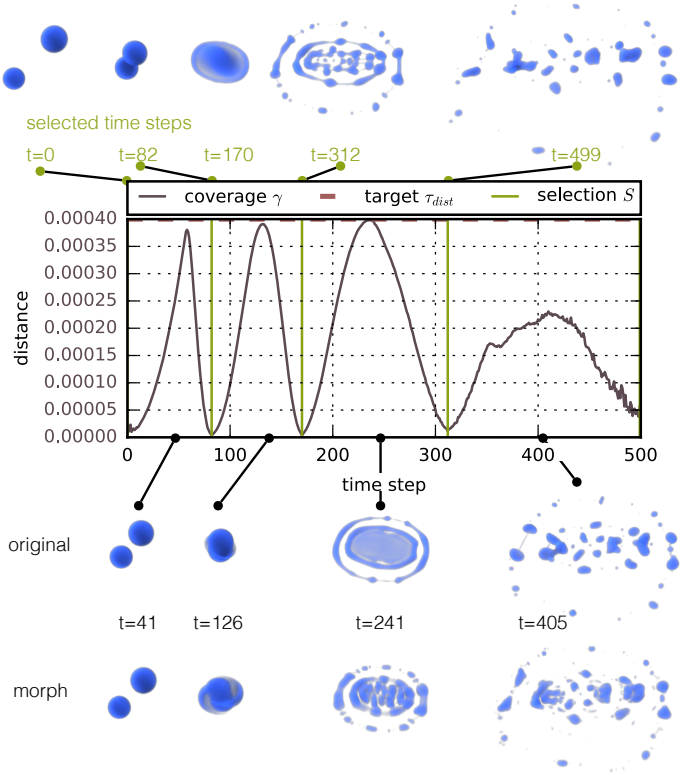


Fig. 13: Droplet data set selected with comparably high target distance $\tau_{\text{target}} = 0.0004$ (we used $\tau_{time} = 120s$ as termination time limit). This way, the whole data set of 500 is represented by 5 time steps only, yet at the cost of decreased quality between selected time steps.

ahead of the real decomposition in the data set. Here, we chose this coarse target distance τ_{dist} for the sake of demonstration of the fundamental properties. According to our experiments, significantly reducing τ_{dist} (i.e., ten times smaller and beyond) yields a much denser selection, but exhibits only minor visual differences to the original. In general, as already discussed above at the example of λ_2, the characteristics of the underlying data naturally have a big impact on the selection. In the case of the Droplet data set, the selection nicely segments the different stages of the simulation (cf. Fig. 13). First, the two droplets move toward each other, then they go through initial contact and deformation, before radially splashing in the form of smaller droplets. Particularly while droplets move largely independently from each other, time steps need to be selected less densely as they can be approximated nicely via our linear interpolation (e.g., from $t = 305$). In contrast, both the beginning and the end of the initial contact of the two droplets needs to be represented, as this cannot be adequately depicted by interpolation.

Also for the Droplet data set, we look at the compression results that can be achieved when specifying limits for the root-mean-square error (RMSE) that may not be exceeded (Fig. 14). Even for small RMSE values (i.e., only minor deviations), we already reduce the number of time steps by a couple of times. Even high compression rates can be achieved with relatively low RMSE values already, as our transformations are able to approximate the underlying changes in the data well for the Droplet data set.

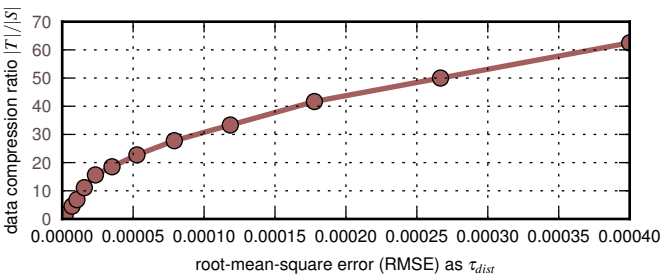


Fig. 14: Data compression ratio $|T|/|S|$ for the Droplet data set with varying limits $\tau_{\text{dist}}$for the root-mean-square error (RMSE).

6 Discussion of Properties and Limitations

Next, we briefly summarize the properties of our algorithm, before discussing its limitations on the basis of fundamental design decisions and the evaluation results from Sec. 5.

Properties. In this paper, we discussed and demonstrated that our approach exhibits the following properties:

Transportation-based Distances and Interpolation. Meaningful distances and smooth intermediates are achieved via assignment-based distances and paths.

Direct. Our approach works directly on the representation of the data, and does not require intermediates like features, shapes, or skeletons.

Automatic. We do not require any parameters, selections, or other kinds of specifications provided by the user.

Generic. No domain knowledge is required.

Progressive. Our iterative approach continuously refines F and can be interrupted at any time.

Distance Upper Bound. Our approach provides iteratively lowered upper bounds for the distance between two data sets.

Convergence to Optimum. We will eventually reach the optimal solution with the minimal associated distance.

Parallel. Our approach was designed for efficient parallel execution.

Applicability and Limitations. While we are not aware of cases in which our approach breaks or fails completely, there are different application scenarios for which it inherently exhibits certain shortcomings (at least in its current form without respective extensions). In the following, we categorize these cases in the form of two (successive) questions: *(a)* does the Wasserstein metric, and *(b)* does our respective progressive computation fit the application purpose?

(a) While we consider generality and automaticity as strong features of our approach as they enable wide and convenient applicability, some application scenarios need to incorporate domain-specific knowledge to yield certain properties, and therefore have to incorporate certain constraints and/or user input (cf. Sec. 2). For instance, this applies to (medical) data classified via segmentation (e.g. [35]), or characteristic points with a pre-defined matching (as discussed earlier for Chameleon→Mouse in Sec. 5.1, and the Flower ensemble in Sec. 5.5). Another example are physically-based transformations (e.g. [11]). In contrast, the basic Wasserstein metric (and our $\gamma(\cdot)$ accordingly) only considers Euclidean distance without further influence factors (discussed at the example of morphing with an arbitrary function G in Sec. 5.4). However, our approach could easily be extended by restricting the exchanges allowed (e.g., source elements α belonging to a certain region of a skull may only be assigned to target elements ω belonging to the respective region), or by modifying $\gamma(\cdot)$ (e.g., to consider physical forces). Another potential issue concerns the reconstruction of intermediate volumes. Here, the local density (per cell) may exceed certain limits (e.g., the maximum value), leading to clamping, while neighboring cells may end up with a significantly lower mass (in the worst case, this can locally lead to a checkerboard-type look). This could be overcome by a more advanced reconstruction scheme.

(b) Our approach typically produces "blurry" intermediate results in early stages of refinement (cf. Sec. 5.1, Fig. 5), but also unpleasant effects may appear due to respective reconstruction issues. As already discussed similarly in *(a)*, this could be mitigated by more advanced reconstruction schemes that explicitly take this into account. Likewise, we limit ourselves to the (reconstruction of) regular grids in this paper, but arbitrary data could be used via resampling (either regularly, or with a combination of inverse transform sampling and density estimation, cf. Sec. 3). Furthermore, our approach was designed with the goal to achieve close-to-optimal results as quickly as possible. It will eventually reach the optimum, but it might take a long time until it gets there (cf. Sec. 5.1, Fig. 9a). Accordingly, traditional approaches to solve the transportation problem are better suited for very small problem sizes (orders of magnitude below typical volume resolutions), in particular if cost-optimal solutions are strictly required. Furthermore, we use a two-phase approach for parallelization to avoid explicit synchronization (e.g., via mutexes). First, each thread individually generates a full plan, and second, each plan is carried in parallel by all threads. While this works well for our multi-core implementation and yields close-to-linear scaling in the long run, we also discussed that the "efficiency" of plans is comparably low in early stages of refinement when other plans that were generated in parallel were executed previously (cf. Sec. 5.1, Fig. 6). Potentially, this could lead to issues when directly porting the approach to many-core architectures with hundreds of cores like a GPU. This could be overcome by a hybrid parallelization approach that exploits the implicit (lockstep) synchronization of threads running on the same multiprocessor of a GPU (e.g., similar to [17]).

7 Conclusion

We presented a technique to directly and automatically transform between different volumes, and demonstrated its application and utility for the (visual) analysis of various volume data sets. In contrast to conventional morphing techniques, it requires no parameters, user guidance, intermediate representations (like extracted features), or a combination of techniques (e.g., with blending), and imposes no restrictions regarding shape or structure. Volumes are handled as generic density distributions, and while linear programming approaches exist to solve the involved transformation, their complexity makes them infeasible for high-resolution data sets. Our technique is designed for efficient parallel and progressive execution that iteratively improves, can be interrupted anytime, and quickly achieves close-to-optimal solutions. Most importantly, our approach both generates smooth interpolates between two arbitrary data sets as well as meaningful distances (along with an upper bound for the optimal solution). We demonstrated that this enables a variety of different techniques beyond the analysis of just a pair of volumes. In particular, we presented a technique for selecting time steps in a temporal data set that allows to reconstruct the full sequence, while not exceeding a pre-defined distance between reconstructed and original volume. We evaluated the characteristics and utility of our transformation approach in the context of different use cases and data sets, and compared it against alternative approaches.

For future work, we plan to create a GPU implementation of our approach. We believe that its overall structure is well-suited not only for multi-core but also for many-core architectures, and we aim to study and evaluate this in detail (cf. Sec. 6). While we already presented various applications in this paper, we further aim to evaluate the utility of our approach in more breadth and depth by considering additional scenarios. Among others, we plan to look into employing our approach for in-situ analysis and reduction in the context of large-scale simulations on supercomputers. We further aim to extend our approach to additionally consider certain restrictions, incorporating user guidance and/or domain- or application-specific knowledge (e.g., anatomy in the field of biology), and then evaluate against previous approaches in the field. Finally, we plan to examine intermediate volume reconstruction for arbitrary volume representations, particularly considering the two options of (1) regular resampling and (2) a combination of inverse transform sampling and density estimation (cf. Sec. 3).

Acknowledgments

The authors would like to thank the German Research Foundation (DFG) for supporting the project within project A02 of SFB/Transregio 161 and the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. A Review of Temporal Data Visualizations Based on Space-Time Cube Operations. In R. Borgo, R. Maciejewski, and I. Viola, editors, *EuroVis - STARs*. The Eurographics Association, 2014.

[3] B. Bai, A. Joshi, S. Brandhorst, V. D. Longo, P. S. Conti, and R. M. Leahy. A registration-based segmentation method with application to adiposity analysis of mice microct images, 2014.

[4] J.-P. Balabanian, I. Viola, T. Möller, and M. E. Gröller. Temporal styles for time-varying volume data, 2008. POSTER PRESENTATION.

[5] L. Balmelli, C. J. Morris, G. Taubin, and F. Bernardini. Volume warping for adaptive isosurface extraction. In *Proceedings of the Conference on Visualization '02*, VIS '02, pages 467–474, Washington, DC, USA, 2002. IEEE Computer Society.

[6] T. Beier and S. Neely. Feature-based image metamorphosis. *SIGGRAPH Comput. Graph.*, 26(2):35–42, 1992.

[7] N. Bonneel, M. van de Panne, S. Paris, and W. Heidrich. Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph.*, 30(6):158:1–158:12, 2011.

[8] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[9] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: Visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 745–747, New York, NY, USA, 2006. ACM.

[10] M. Chen, M. W. Jones, and P. Townsend. *Image Processing for Broadcast and Video Production: Proceedings of the European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production, Hamburg, 23–24 November 1994*, chapter Methods for Volume Metamorphosis, pages 280–292. Springer London, London, 1995.

[11] C. D. Correa, D. Silver, and M. Chen. Constrained illustrative volume deformation. *Comput. Graph.*, 34(4):370–377, 2010.

[12] K. Eisenschmidt, M. Ertl, H. Gomaa, C. Kieffer-Roth, C. Meister, P. Rauschenberger, M. Reitzle, K. Schlottke, and B. Weigand. Direct numerical simulations for multiphase flows: An overview of the multiphase code FS3D. *Applied Mathematics and Computation*, 272, Part 2:508 – 517, 2016.

[13] S. Fang, R. Srinivasan, R. Raghavan, and J. T. Richtsmeier. Volume morphing and rendering - an integrated approach. *Comput. Aided Geom. Des.*, 17(1):59–81, 2000.

[14] Z. Fang, T. Möller, G. Hamarneh, and A. Celler. Visualization and exploration of time-varying medical image data sets. In *Proceedings of Graphics Interface 2007*, GI '07, pages 281–288, 2007.

[15] O. Fernandes, S. Frey, F. Sadlo, and T. Ertl. Space-time volumetric depth images for in-situ visualization. In *IEEE Large Data and Visualization 2014 (LDAV14)*, 2014.

[16] S. Frey, F. Sadlo, and T. Ertl. Visualization of temporal similarity in field data. *IEEE Transactions on Visualization and Computer Graphics*, 18:2023–2032, 2012.

[17] S. Frey, T. Schlömer, S. Grottel, C. Dachsbacher, O. Deussen, and T. Ertl. Loose capacity-constrained representatives for the qualitative visual analysis in molecular dynamics. In *Visualization Symposium (PacificVis), 2011 IEEE Pacific*, pages 51–58, 2011.

[18] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 136–146, 1986.

[19] J. Gomes, L. Darsa, B. Costa, and L. Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann Publishers Inc., 1998.

[20] Google. Google Optimization Tools, 2015.

[21] T. He, S. Wang, and A. Kaufman. Wavelet-based volume morphing. In *Visualization, 1994., Visualization '94, Proceedings., IEEE Conference on*, pages 85–92, CP8, 1994.

[22] T. Ijiri, S. Yoshizawa, H. Yokota, and T. Igarashi. Flower modeling via x-ray computed tomography. *ACM Trans. Graph.*, 33(4):48:1–48:10, 2014.

[23] P. Iyer. *Operations research*. Tata McGraw-Hill, New Delhi, 2008.

[24] Y. Jang, D. Ebert, and K. Gaither. Time-varying data visualization using functional representations. *Visualization and Computer Graphics, IEEE Transactions on*, 18(3):421–433, 2012.

[25] G. K. Karch, F. Sadlo, C. Meister, P. Rauschenberger, K. Eisenschmidt, B. Weigand, and T. Ertl. Visualization of piecewise linear interface calculation. In *IEEE Pacific Visualization Symposium*, pages 121–128, 2013.

[26] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 47–54, New York, NY, USA, 1992. ACM.

[27] T.-Y. Lee and H.-W. Shen. Visualization and exploration of temporal trend relationships in multivariate time-varying data. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1359–1366, 2009.

[28] T.-Y. Lee and H.-W. Shen. Visualizing time-varying features with tac-based distance fields. In *Visualization Symposium, 2009. PacificVis '09. IEEE Pacific*, pages 1–8, 2009.

[29] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 449–456, New York, NY, USA, 1995. ACM.

[30] L. Lin, C. Lee, and T.-Y. Lee. Distributed volume morphing. *Cluster Computing*, 2(3):219–227, 1999.

[31] Y.-S. Liu, H.-B. Yan, and R. R. Martin. As-rigid-as-possible surface morphing. *Journal of Computer Science and Technology*, 26(3):548–557, 2011.

[32] A. Lu and H.-W. Shen. Interactive storyboard for overall time-varying data visualization. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 143–150, 2008.

[33] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin. UCSF ChimeraA visualization system for exploratory research and analysis. *Journal of Computational Chemistry*, 25(13):1605–1612, 2004.

[34] N. Reinfeld and W. Vogel. *Mathematical programming*. Prentice-Hall, 1958.

[35] T. Rhee, J. P. Lewis, U. Neumann, and K. Nayak. Scan-based volume animation driven by locally adaptive articulated registrations. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):368–379, 2011.

[36] Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[37] M. Sanchez, O. Fryazinov, T. Vilbrandt, and A. Pasko. Morphological shape generation through user-controlled group metamorphosis. *Computers and Graphics*, 37(6):620 – 627, 2013. Shape Modeling International (SMI) Conference 2013.

[38] D. Schneider, A. Wiebel, H. Carr, M. Hlawitschka, and G. Scheuermann. Interactive comparison of scalar fields based on largest contours with applications to flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1475–1482, 2008.

[39] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[40] D. Sieger, S. Menzel, and M. Botsch. RBF morphing techniques for simulation-based design optimization. *Engineering with Computers*, 30(2):161–174, 2013.

[41] D. Silver and X. Wang. Tracking and visualizing turbulent 3d features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, 1997.

[42] M. L. Staten, S. J. Owen, S. M. Shontz, A. G. Salinger, and T. S. Coffey. *Proceedings of the 20th International Meshing Roundtable*, chapter A Comparison of Mesh Morphing Methods for 3D Shape Optimization, pages 293–311. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[43] X. Tong, T.-Y. Lee, and H.-W. Shen. Salient time steps selection from large scale time-varying data sets with dynamic time warping. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, pages 49–56, 2012.

[44] C. Wang, H. Yu, and K.-L. Ma. Importance-driven time-varying data visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1547–1554, 2008.

[45] W. Widanagamaachchi, C. Christensen, P.-T. Bremer, and V. Pascucci. Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, pages 9–17, 2012.

[46] C. G. Willcocks and F. W. B. Li. Feature-varying skeletonization. *The Visual Computer*, 28(6):775–785, 2012.

[47] J. Woodring and H.-W. Shen. Chronovolumes: A direct rendering technique for visualizing time-varying data. In *Eurographics/IEEE TVCG Workshop on Volume Graphics*, pages 27–34, 2003.