Fast Flow-based Distance Quantification and Interpolation for High-Resolution Density Distributions

S. Frey and T. Ertl

University of Stuttgart, Visualization Research Center, Germany

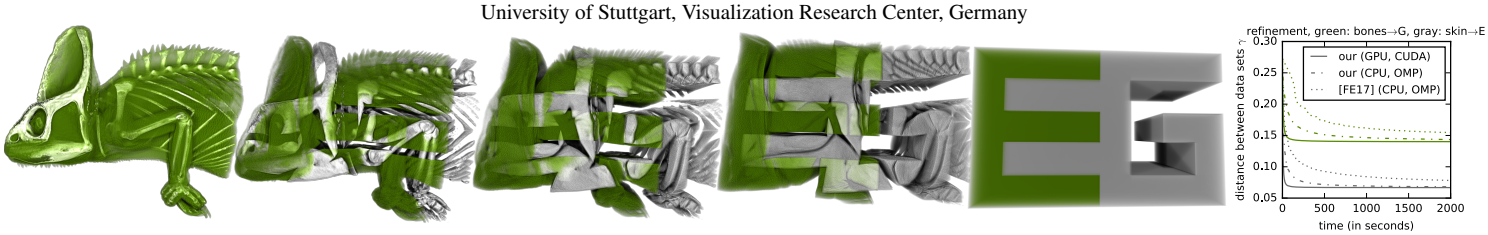


Figure 1: *Transition between the Chameleon and the Eurographics Logo (skin → E, bones → G). Our algorithm redesign substantially improves the performance over the original approach [FE17] on the CPU (17×, 36×), with even larger speedups on the GPU (316×, 511×).*

Abstract

We present a GPU-targeted algorithm for the efficient direct computation of distances and interpolates between high-resolution density distributions without requiring any kind of intermediate representation like features. It is based on a previously published multi-core approach, and substantially improves its performance already on the same CPU hardware due to algorithmic improvements. As we explicitly target a manycore-friendly algorithm design, we further achieve significant speedups by running on a GPU. This paper quickly reviews the previous approach, and explicitly discusses the analysis of algorithmic characteristics as well as hardware architectural considerations on which our redesign was based. We demonstrate the performance and results of our technique by means of several transitions between volume data sets.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction and Motivation

Creating transitions between volumes has a long history in visualization and computer graphics [GDCV98]. Most morphing or warping approaches beyond cross-dissolving are feature-based, i.e., they require the manual or (semi-)automatic detection of control points, skeletons [LGL95], landmarks [FSRR00], or template volumes [RLNN11]. In scientific visualization, transformations of volumetric objects have been used to generate illustrative visualizations, adhering to physically inspired constraints [CSC10]. However, in this work, we focus on generating expressive distances and interpolates between pairs of volumes (i.e., high-resolution density distributions), which is a crucial building block for numerous applications in computer vision, computer graphics, and visualization.

In this paper, we discuss our GPU-targeted algorithm design for the efficient direct computation of distances and interpolates between high-resolution density distributions without requiring any kind of intermediate representation like features. We particularly focus on our changes – and their rationale – to improve the multicore approach by Frey and Ertl [FE17] to a manycore design that yields a massive improvement in refinement performance (cf. Fig. 1 (right)). Conceptually, the approach is based on the earth mover's distance (EMD, also known as the Wasserstein metric), and determines the minimum cost of turning one distribution into the other. The EMD is popular in computer vision to quantify distances between color histograms, e.g., for image retrieval [RTG00]. For rendering, Bonneel et al. [BvdPPH11] decompose distributions (like BRDFs, environment maps, stipple patterns, etc.) into radial basis functions, and then apply partial transport that independently considers different frequency bands. For scientific visualization, Tong et al. [TLS12] compute the distance between data sets via different metrics for the temporal reduction of volume data sets (cf. [FE17] for a detailed overview). These applications only consider distributions consisting of a comparably small number of elements due to the high cost involved. The EMD basically involves a transportation problem that is typically solved via linear programming (e.g., via the the cost-scaling push-relabel algorithm [GT86, Goo15]). These approaches cannot deal with high-resolution volumes due to their high (cubic) complexity, and approximations only yield unsatisfactory results for many visualization and graphics applications (e.g., due to unfitting heuristics [RV58] or required regularization [SdGP*15]). Frey and Ertl [FE17] recently presented an approach addressing this issue.

Algorithm 1 *Overview on the volume transformation approach.*

1: **procedure** VOLTRANS ▷ (Sec. 2)
2: create initial assignment ▷ directly from [FE17]
3: **while** termination criterion not met **do** ▷ directly from [FE17]
4: ***generate exchange plan*** ▷ (Sec. 2c [FE17], Sec. 3b [new])
5: ***execute exchange plan*** ▷ (Sec. 2c [FE17], Sec. 3c [new])
6: create intermediate volumes, if required ▷ directly from [FE17]

This paper identifies potentials for improvement in the original approach, and accounts for them via a redesign for more efficiency as well as the additional support of manycore devices. First, we motivate the problem and review the general concept behind the progressive approach by Frey and Ertl [FE17] (Sec. 2). We then discuss our changes (and their rationale) to improve its performance and device portability (Sec. 3). Finally, we compare our improved version to the original approach at the example of different data sets (Sec. 4), and conclude our work in Sec. 5.

2. Parallel Volume-to-Volume Transformation

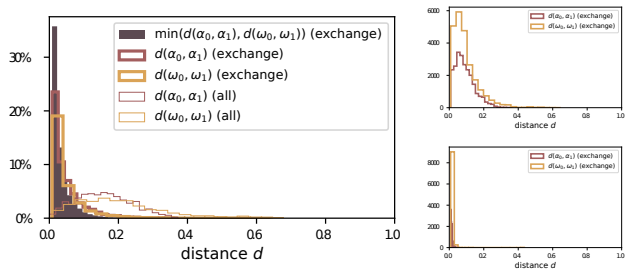
We now describe the underlying problem (a), and outline the basic approach to solve it (b). We then discuss the original implementation of performance-critical steps by Frey and Ertl [FE17] (c).

(a) Problem Description. The transformation between two volumes is directly computed on the basis of their (3D) density 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 (sample) points A and Ω (i.e., as they typically stem from CT scans, simulations, etc.). Arbitrary (continuous) data could 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 positive mass $m(\cdot)$. For now, we assume a balanced problem ($\sum_{\alpha\in A} m(\alpha) = \sum_{\omega\in\Omega} m(\omega)$). 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:

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

with $m_{\alpha\to\omega}$ denoting the respective weight (or mass) associated with an individual assignment. The goal is to determine a mapping F that yields the minimum for $\gamma(\cdot)$.While F is used to generate transitions between volumes, $\gamma(F)$ quantifies the distance between them.

(b) Approach Outline. We now outline the basic structure of the approach by Frey and Ertl [FE17] – which we also use in our refined technique — in Alg. 1. First, an initial assignment F is computed (Line 2, this includes balancing the problem, cf. [FE17] for details). The algorithm then iteratively refines the assignment F of source samples α to target samples ω in two phases (Lines 3–5). First, an exchange plan is generated that consists of mutually exclusive subsets of A (Line 4). Then this plan is executed, determining if and which target nodes to actually exchange, and eventually carrying out the transfer (Line 5). Creating non-overlapping lists of $\alpha \in A$ during planning is beneficial for the parallelization of the exchange part, as



(a) *exchanges over the full course of refinement* **(b)** *early* (↑) *& late* (↓)

Figure 2: *(a) Considering random pairs as exchange sequences, and plotting all of them as well as only the successful ones (i.e., resulting in an exchange) w.r.t. mutual distances. (b) The distribution of successful sequences shifts during the course of refinement.*

no two source elements $\alpha_0, \alpha_1 \in A$ are considered more than once at the same time. Multiple refinement iterations of planning and exchange are carried out, until some kind of termination criterion is reached (like a time or γ limit, Line 3). Finally, intermediate volumes are generated from assignment F (Line 6).

(c) Original Planning & Exchange [FE17]. The performance-critical parts of the approach are (1) to identify promising exchange sequences, and (2) to evaluate and carry out respective exchanges if they are beneficial. Exchange sequences consist of a cyclic list of source elements α in which each α_i passes one assigned target element ω to its successor α_{i+1} and for this receives one target element from its predecessor α_{i-1}. These exchanges are only carried out if they decrease $\gamma(F)$. To identify such sequences, Frey and Ertl [FE17] iterate over a shuffled list of all source elements A and attempt to find promising sequences in a greedy fashion. In the process, they maintain an open sequence C, and for each currently considered $\alpha \in A$ they distinguish between three cases: (1) close C with α to become a new exchange sequence *(if an exchange potentially improves γ for $S = \{C_i, \ldots, C_{|C|-1}, \alpha\}$)*, (2) extend C with α *(if an exchange is beneficial when neglecting the cyclic wrap-around, i.e, $\alpha_{n-1} \to \alpha_0$)*, or (3) discard α *(else)*. To approximate this efficiently, bounding boxes are used that contain all target elements ω assigned to a source element α. With this, the difference of the maximally possible distance of source element α_0 to its own box $\text{bbox}(\alpha_0)$ and the minimally possible distance of α_1 to $\text{bbox}(\alpha_0)$ gives an upper bound for improvement $\Delta\gamma$ when transferring a mass element $m(\omega)$ from α_0 to α_1. Eventually, a random value is used between the minimally and maximally possible distance to avoid always stopping early for eventually unsuccessful short sequences of length $l = n$ when longer ones $l > n$ are actually needed.

A closer investigation shows that, despite the significant effort invested into planning, only few sequences actually lead to an exchange. Additionally, while the exchange procedure exhibits a high degree of parallelism by design, planning is inherently sequential. While Frey and Ertl [FE17] work around this issue by creating multiple plans in parallel, this is only feasible for low degrees of parallelism (i.e., a couple of threads), and there is still the downside of plans degrading in utility if other plans were processed in the meantime. Below, we aim to address these issues by improving the efficiency in general, with a particular focus on many-core devices.

Algorithm 2 *Our new planning approach (Alg. 1, Line 4).*

```
1:  ▷ alternately use m_α and m_ω, log of previous computations l
2:  procedure GENERATEEXCHANGEPLAN(m, l) ▷ m ∈ {m_α, m_ω}
3:    c ← adaptive_distribution_ℝ(l_m)        ▷ randomly choose cell size
4:    for all α ∈ A do
5:      if m = m_α then                       ▷ use position of source
6:        p ← P(α)
7:      else                                  ▷ (m = m_ω) use position of target
8:        p ← P(random_selection(F(α)))
9:      K(α) ← hilbert(p/c) + uniform_distribution(0, 1)
10:   A ← sort(A, K)                          ▷ using THRUST for CPU, CUB for GPU
11:   P ← segment(A, n)                       ▷ partition A into sequences of size n
```

Algorithm 3 *Our new exchange approach (Alg. 1, Line 5).*

```
1:  procedure EXECUTEEXCHANGEPLAN(p ∈ P)      ▷ Fig. 3
2:    i ← 0, j ← 1                            ▷ index initialization
3:    while i + 1 < |p| do                    ▷ identify exchange sequences p[i...j]
4:      if Δγ(p[i...j]) < 0 then              ▷ exchange improves γ(·) (Eq. 1)?
5:        cyclicSwap(p[i...j])                ▷ execute swap (Fig. 3, red)
6:      else if Δ→γ(p[i...j]) < 0 then        ▷ exchange wo/ wraparound?
7:        j ← j + 1                           ▷ extend sequence
8:        if j = |p| then                     ▷ end of partition reached by end index j
9:          i ← i + 1, j ← i + 1              ▷ increment start index i
10:     else
11:       i ← i + 1, j ← i + 1                ▷ increment start index i
```

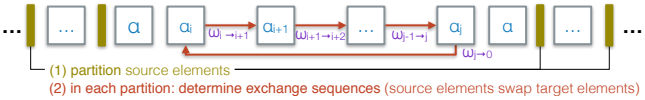


Figure 3: *Partitioning of source elements α into sequences, and then processing them (in parallel) during the exchange procedure.*

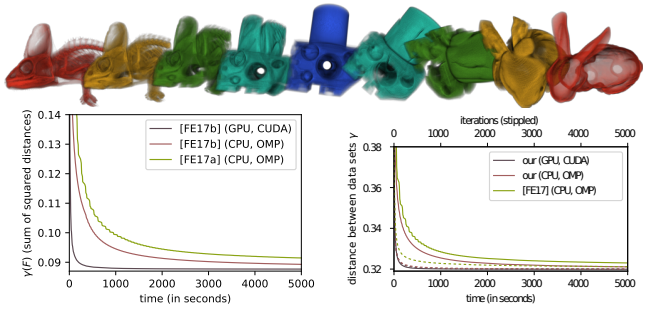
3. Improved Refinement: Planning & Exchange

The planning step is both the most costly as well as the most critical part for the refinement rate of the progressive volume transformation approach. In this section, we first analyze the original planning approach to identify potentials for improvement (a). On this basis, we then discuss our new approach, that both improves the generated exchange sequences and runs efficiently on manycore devices (b). While we focus on these adjustments in the planning step, they also induce changes to the exchange part of the approach (c).

(a) Exchange Analysis. A crucial part of the approach is the identification of source elements that form a promising sequence for a potential exchange. As a basis for our redesign, we conducted an analysis to see which groups of source elements actually exchange target elements (Fig. 2a). For this, we randomly select source elements for an exchange (in the discussion in the following, we limit ourselves to pairs, but the same applies to larger group sizes as well according to our analysis). We then consider the distances $d(\alpha_0, \alpha_1)$ between source elements α_0 and α_1 and distances $d(\omega_0, \omega_1)$ between the respective target elements $\omega_0 \in F(\alpha_0)$ and $\omega_1 \in F(\alpha_1)$. It can be seen that, over a whole refinement run, mutual distance appears to be a good indicator of whether a group actually exchanges elements, and we explicitly consider the following two cases in our algorithm redesign: **(1)** the position of the two sources $P(\alpha_0)$ and $P(\alpha_1)$ is close, or **(2)** the position of the two targets $P(\omega_0 \in F(\alpha_0))$ and $P(\omega_1 \in F(\alpha_1))$ is close. We can also see that the range of beneficial exchange sequences shifts during the course of refinement (Fig. 2b), and we explicitly account for that as well.

(b) Planning: Distance-Based Partitioning. In contrast to the original approach that explicitly creates exchange sequences, our refined techniques only creates partitions of A during planning and lets improving exchanges be determined in the exchange phase (Alg. 2). The partitioning is based on (1) the observation that exchanges typically happen when sources and/or targets are close, and (2) the premise that only operations should be used that efficiently support massive, shared memory parallelism. In each refinement iteration, we alternately consider the distance between sources (mode m_α) or the distance between targets (mode m_ω). We also consider a log of previous exchanges l_m (for each mode m) to determine the size of the cells c of a uniform grid in which we group source elements to account for shifting exchange characteristics. For this, we randomly sample the distribution of previous exchanges given by l_m such that we are more likely to use a cell size that has lead to a high number of executed exchanges in previous iterations. Typically, this distribution is rather broad in early iterations (Fig. 2b *(top)*), and is increasingly skewed toward smaller cell sizes later on (Fig. 2b *(bottom)*). For mode m_α, the considered position p is directly obtained via a lookup $P(\alpha)$ (Line 6), while for m_ω there are numerous options if multiple ω are assigned to one α. In the latter case, we just randomly pick an assigned $\omega \in F(\alpha)$ and use its position $P(\omega)$ (Line 8). We then assign a scalar value $K(\alpha)$ to each α based on the determined position p (Line 9). We do this by determining a (three-dimensional) cell index via p/c, and mapping this to a (one-dimensional) index by using a a pre-computed 3D Hilbert curve (with a resolution of 512^3) to preserve locality between cells. We also randomly offset the αs within one cell. we then simply sort A using the keys K (Line 10), and then segment it into partitions of size n (Line 11). These partitions are then used to create exchange sequences in the subsequent step.

(c) Exchange: Sequences from Partition. The original approach directly identifies potential exchange sequences in the planning process. In contrast, our revised version just identifies a partition during planning, and then in the exchange step flexibly determines exchange sequences $p[i \dots j]$ as subsequences of each partition $p \in P$ (Alg. 3, Fig. 3). Also, multiple exchange sequences may be processed for each partition. Our exchange procedure loops over different start indices i (Line 3) and end indices j ($j \geq i+1$) and distinguishes between three different cases. First, if a certain subsequence $p[i \dots j]$ decreases the evaluation value γ (Line 4), the respective cyclic transfer of target elements ω is carried out like in the original approach of Frey and Ertl [FE17] (Fig. 3). If, however, the subsequence $p[i \dots j]$ would only be beneficial without the passing of a target element from $p[j]$ to $p[i]$ (i.e., the wraparound, $w_{j \to i}$ in Fig. 3, Line 6), we attempt to make it beneficial by extending it to a longer sequence (Line 7). If none of the above applies (Line 10), or if we reach the end of the partition (Line 8), we start over with an incremented starting index i (Line 9 or 11).



(a) *Chameleon (red)→Zeiss (blue)* **(b)** *Zeiss (blue)→Bunny (red)*

Figure 4: *Refinement (i.e., decreasing γ) for different transitions (a, b) and variants w.r.t. time (solid line) and iteration count (dashed).*

4. Results

For the evaluation, we use a machine equipped with an Intel Core i7-4770 CPU, an NVIDIA GeForce GTX TITAN X, and 32 GB of RAM. In our CPU and GPU implementation, we used OpenMP (Thrust for sorting) and CUDA (CUB for sorting), respectively. We evaluate the refinement behavior using the following volumes: Chameleon (Figs. 1 & 4a, $512 \times 1024 \times 1080$), Zeiss (Figs. 4a & b, 680^3), Bunny (Fig. 4b, $512^2 \times 361$), and EG (Fig. 1, $256 \times 750 \times 1200$) with different transfer functions (cf. renderings). We compare three variants: (1) the original approach by Frey and Ertl [FE17] using OpenMP (CPU), as well as our improved approach implemented via (2) OpenMP (CPU) and (3) CUDA (GPU). In our evaluation, we let each program refine a transition for 50 000 s, and depict refinement in terms of decreasing evaluation values γ with respect to the time spent (depicted in Figs. 1 and 4 for the initial 5000 s). In each case, we can see that the refinement with our improved approach on the CPU is much faster in comparison to the original approach. Our GPU version further significantly outperforms our CPU implementation. We quantify the differences in performance via a scaling factor. It is computed by taking the final value of the refinement with the original approach γ', and then determining when this γ' was first surpassed in the refinement with our improved approaches ($t_{\text{[FE17]}}(\gamma') = 50\,000\,\text{s}, t_{\text{OpenMP}}(\gamma'), t_{\text{CUDA}}(\gamma')$). We then compute the scaling factor via $t_{\text{[FE17]}}(\gamma')/t_{\text{OpenMP}}(\gamma')$ and $t_{\text{[FE17]}}(\gamma')/t_{\text{CUDA}}(\gamma')$, respectively. This yields the following results: Chameleon→EG (gray, OpenMP: 35.8×, CUDA: 511.1×), Chameleon→EG (green, OpenMP: 17.2×, CUDA: 315.0×), Chameleon→Zeiss (OpenMP: 7.28×, CUDA: 152.9×), and Zeiss→Bunny (OpenMP: 8.04×, CUDA: 177.26×). Fig. 4 additionally shows the refinement not only with respect to passed time but also in terms of iteration counts. Naturally, the curves for both implementations of our improved approach are congruent, as they only differ in the time required for an iteration. In comparison to the original approach, our new refinement iterations are much more efficient, leading to larger improvements in γ even when neglecting the computation time. On average, an iteration with the original takes 4.71 s [FE17] / 3.03 s (OpenMP) / 0.10 s (CUDA) for Chameleon→EG (gray), 10.99 s [FE17] / 11.26 s (OpenMP) / 0.45 s (CUDA) for Chameleon→EG (green), 7.30 s [FE17] / 12.88 s (OpenMP) / 0.38 s (CUDA) for Chameleon→Zeiss, and 5.38 s [FE17] / 8.27 s (OpenMP) / 0.37 s (CUDA) for Zeiss→Bunny. These variations in planning timings give an indication on the variations in speedups observed above, but, while our preliminary results look very promising, a closer investigation of performance characteristics remains for future work.

5. Conclusions

We presented a GPU-targeted algorithm for the efficient direct computation of distances and interpolates. It significantly improves the performance of the multi-core approach by Frey and Ertl [FE17]. Performance is substantially increased already on CPUs due to algorithmic improvements (our evaluation demonstrates speedups between 7× and 35×). Via a CUDA-based GPU implementation, we even achieve speedups between 153 and 511×. Our redesign was based on the analysis of algorithmic characteristics and the usage of manycore-friendly algorithms. For future work, we aim to further improve the planning efficiency, consider distributed and hybrid setups, and apply our approach to different applications in visualization and computer graphics.

Acknowledgements

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

[BvdPPH11] BONNEEL N., VAN DE PANNE M., PARIS S., HEIDRICH W.: Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph. 30*, 6 (2011), 158:1–158:12. 1

[CSC10] CORREA C. D., SILVER D., CHEN M.: Constrained illustrative volume deformation. *Comput. Graph. 34*, 4 (2010), 370–377. 1

[FE17] FREY S., ERTL T.: Progressive direct volume-to-volume transformation. *IEEE TVCG 23*, 1 (Jan 2017), 921–930. 1, 2, 3, 4

[FSRR00] FANG S., SRINIVASAN R., RAGHAVAN R., RICHTSMEIER J. T.: Volume morphing and rendering - an integrated approach. *Comput. Aided Geom. Des. 17*, 1 (2000), 59–81. 1

[GDCV98] GOMES J., DARSA L., COSTA B., VELHO L.: *Warping and Morphing of Graphical Objects*. Morgan Kaufmann Publishers, 1998. 1

[Goo15] GOOGLE: Google Optimization Tools, 2015. 1

[GT86] GOLDBERG A. V., TARJAN R. E.: A new approach to the maximum flow problem. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing* (1986), pp. 136–146. 1

[LGL95] LERIOS A., GARFINKLE C. D., LEVOY M.: Feature-based volume metamorphosis. SIGGRAPH '95, ACM, pp. 449–456. 1

[RLNN11] RHEE T., LEWIS J. P., NEUMANN U., NAYAK K.: Scan-based volume animation driven by locally adaptive articulated registrations. *IEEE TVCG 17*, 3 (2011), 368–379. 1

[RTG00] RUBNER Y., TOMASI C., GUIBAS L.: The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision 40*, 2 (2000), 99–121. 1

[RV58] REINFELD N., VOGEL W.: *Mathematical programming*. Prentice-Hall, 1958. 1

[SdGP*15] SOLOMON J., DE GOES F., PEYRÉ G., CUTURI M., BUTSCHER A., NGUYEN A., DU T., GUIBAS L.: Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Trans. Graph. 34*, 4 (July 2015), 66:1–66:11. 1

[TLS12] TONG X., LEE T.-Y., SHEN H.-W.: Salient time steps selection from large scale time-varying data sets with dynamic time warping. In *IEEE LDAV* (2012), pp. 49–56. 1