Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Komprese dynamických polygonálních sítí s konstantní a proměnlivou konektivitou

Ing. Jan Dvořák

Disertační práce
k získání akademického titulu doktor
v oboru Informatika a výpočetní technika

Školitel: Doc. Ing. Libor Váša, Ph.D.
Katedra: Katedra informatiky a výpočetní techniky

Plzeň 2023

University of West Bohemia
Faculty of Applied Sciences

Compression of dynamic polygonal meshes with constant and variable connectivity

Ing. Jan Dvořák

Doctoral thesis
submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in Computer Science and Engineering

Supervisor: Doc. Ing. Libor Váša, Ph.D.
Department: Department of Computer Science and Engineering

Pilsen 2023

Declaration

In Pilsen, on 25 October 2023

. . . .
Jan Dvořák

Abstract

Due to the advances in performance capturing technologies, dynamic surface representations, such as mesh and point cloud sequences, are becoming more and more attractive ways of representing dynamic scenes. It is, however, quite challenging to store such data efficiently or to process it in a temporally global manner, because of the lack of explicit temporal correspondence information, which makes it hard to exploit the temporal coherence. Although two subsequent frames of the sequence may look nearly indistinguishable, it is difficult to establish the surface correspondences, since some parts of the surface may have no corresponding counterpart in some frames, due to self-contact.

Current methods for mesh sequence compression are to some extent able to store such data more efficiently than intra-only approaches, which ignore temporal coherence. Quite prominent are approaches based on temporal models, which, however, are currently limited by the *type of data* they are able to handle (e.g., mesh sequences representing humans). Some of the methods achieve better compression rates by discarding the original structure of the frames (the number of vertices and connections between them), mainly because of the lack of temporal coherence in vertex sampling but also because it is difficult to encode connectivity efficiently if temporal coherence is exploited during geometry encoding. There are also methods that can handle general data and preserve the original structure, but these are *inefficient*, and currently, it is preferable to ignore the temporal coherence and still use an intra-only approach in such a case.

This thesis mainly focuses on the ability to encode more general data and on the efficient encoding of the structure of the frames. We propose a novel temporal model called *tracked centers*, which works with volume instead of surface correspondences. This allows representing more general dynamic surfaces, as long as the volume they enclose does not significantly change throughout the sequence. We also present an improved algorithm for connectivity compression, which can potentially be integrated into existing structure-preserving compression pipelines to achieve better data rates.

This doctoral thesis was supported by the projects 22-04622L, 20-02154S and 17-07690S of the Czech Science Foundation (GACR) and by the University specific projects SGS-2022-015 and SGS-2019-016.

Abstrakt

Díky pokrokům technologií pro snímání pohybu jsou struktury pro popis dynamických povrchů, jako jsou sekvence polygonálních sítí či bodových mračen, čím dále častěji používány pro reprezentaci dynamických scén. Taková data je ovšem náročné efektivně ukládat, či je časově globálně zpracovat, protože nejsou explicitně známy časové korespondence mezi snímky, což komplikuje využití jejich časové koherence. Ačkoli dva po sobě jdoucí snímky mohou vypadat téměř nerozeznatelně, je těžké tyto korespondence mezi povrchy určit, protože některé oblasti na povrchu nemusí mít kvůli kontaktu mezi jeho částmi žádné korespondující protějšky v některých ze snímků.

Metody dostupné v současnosti jsou do určité míry schopny efektivnější komprese nežli metody, které používají pouze prostorovou koherenci a tu časovou ignorují. Často se používají metody založené na časových modelech, které ale omezují *typ dat,* které jsou schopny zpracovat (např. pouze sekvence trojúhelníkových sítí reprezentujících lidskou postavu). Některé metody zase dosahují lepších kompresních poměrů tím, že nezachovávají originální strukturu snímků (počet vrcholů a hrany mezi nimi), zejména kvůli časově nekoherentnímu vzorkování povrchu, ale také protože je náročné kódovat efektivně konektivitu, pokud byla využita časová koherence při kódování geometrie. Existují i metody, které jsou schopny kódovat obecnější data i zachovat strukturu snímků, ty ovšem jsou *neefektivní,* a proto je v takovém případě v současné době výhodnější ignorovat časovou koherenci a ke kódování využít pouze té prostorové.

Tato práce se soustředí na schopnost zpracovat obecnější data a efektivní kódování struktury snímků. Navrhujeme nový časový model nazvaný *trasovaná centra,* který pracuje s objemovými korespondencemi místo povrchových. To umožňuje reprezentaci mnohem obecnějších dynamických povrchů, dokud platí, že objem uzavřený povrchem se po celou sekvenci zásadně nemění. Také představujeme vylepšený algoritmus pro kompresi konektivity, který může být potenciálně zakomponován do existujících kompresních metod, které zachovávají strukturu, pro zlepšení výsledného datového toku.

Tato disertační práce byla podpořena projekty 22-04622L, 20-02154S a 17-07690S Grantové agentury České republiky (GAČR) a univerzitními projekty SGS-2022-015 a SGS-2019-016.

Acknowledgement

I am grateful for the great guidance, valuable feedback, and extreme patience from my supervisor, Doc. Ing. Libor Váša, Ph.D. I would also like to acknowledge my colleagues from the scientific project 20-02154S, Ing. Zuzana Káčereková, Ing. Petr Vaněček, Ph.D., and Ing. Filip Hácha, who contributed significantly to the research presented in this thesis. Last but not least, I would like to thank my family, who were always there to support me.

Contents

Part I
Introduction

Introduction 1

3D surface representations, such as meshes and point clouds, have incredible representative power. When studying a certain object or a collection of objects represented by a mesh or a point cloud, we can observe them from any desired viewpoint and thus we are able to detect details that might not be observable in a static image. However, this is often still insufficient to infer dynamic behaviour. For example, given a scene consisting of a sphere and a box, we cannot tell whether any of the objects are static or if any force is being applied to them, if any collision occurs between the two objects, and if so, whether any of the objects deforms. This can be addressed by multiple representations (called *frames*) capturing the scene at consecutive points in time (see Figure 1.1).

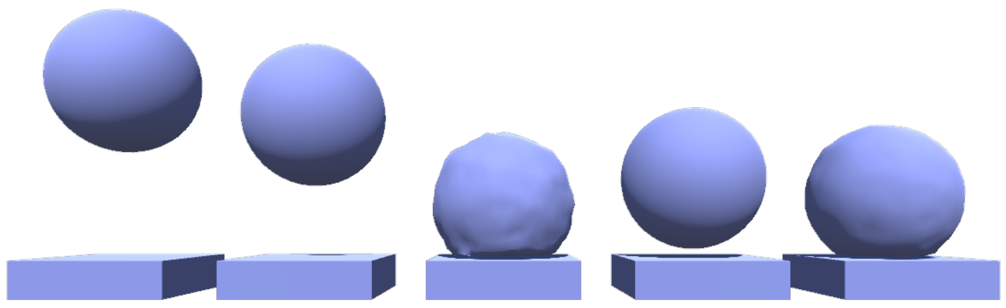


Figure 1.1: Selected frames of a dynamic scene in which a sphere is slightly deformed by a collision with a box. Frames are sorted from left to right in order of appearance.

While modern surface scanning hardware can output such data at sufficient frame rates, there are, unfortunately, only few publicly available datasets of mesh or point cloud sequences. This can be attributed to their large size. For example, the D-FAUST dataset [Bog+17] contains 129 sequences with over 40 000 meshes overall, which requires around 129 GB of storage in an uncompressed form. While the compression of a special class of mesh sequences with constant connectivity, called *dynamic meshes*, is already considered a solved problem and certain recent advancements have been made in the compression of dynamic point clouds, the compression of general triangle mesh sequences (often called *time-varying meshes*) remains an open problem. There are already a few methods that achieve satisfactory compression rates and recently, the problem also recieved the attention of the *Moving Picture Experts Group* (MPEG). However, the current methods either discard the original structure of individual mesh frames, limit the type of input data they are able to process or are outperformed by intra-only approaches which compress each frame separately. We believe that providing an efficient structure-preserving time-varying compression format to the public would motivate others to publish new mesh sequence datasets.

Mesh sequence compression is also needed for *tele-immersion*. Tele-immersion is a way of communicating in augmented or virtual reality, where a person is being captured by a surface scanning device and the resulting 3D model is transmitted in real-time to the receiver (see Figure 1.2). The objectives for compression are different from those in the general scenario. Instead of preserving as much information as possible, it is more important to achieve real-time performance. Although there are time-varying compression methods that claim to achieve near-real-time performance [Dou+14b], these are still too complex to be used in practical scenarios. For this reason, intra-only approaches are currently preferred [Ort+16]. In this thesis, we will focus only on the general scenario, since we believe that an efficient general mesh sequence compression method might be altered to account for the real-time scenario of tele-immersion.

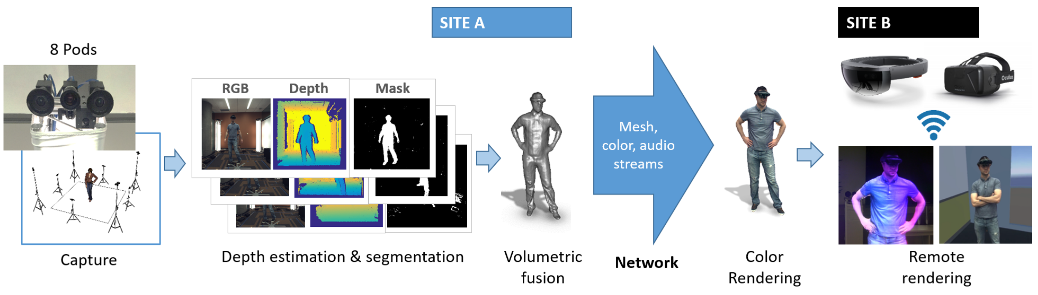


Figure 1.2: Pipeline of the *Holoportation* tele-immersion system [Ort+16].

1.1 Summary of contributions

Error-propagation control in Laplacian mesh compression. Our first contribution is an algorithm for improving the performance of Laplacian mesh compression under mechanistic distortion metrics, which can be used to encode static and dynamic meshes [VD18]. The method is based on a deeper exploration of the decoding process. We have detected that the error accumulation of the original approach is introduced by forward and backward substitution when solving a linear system.

We limit the error accumulation in forward substitution by adjusting the values on the go. This modification brings the performance of Laplacian mesh compression under mechanistic distortion metrics on par with mechanistic-distortion-optimised compression methods.

Model-based molecular trajectory compression. The second contribution is a method for compression of molecular dynamics trajectories [DMV20]. It was inspired by the efficiency of temporal-model-based compression methods. It uses a newly proposed temporal model denoted *canonical molecule*, which captures static local properties of the molecular structure. To the best of our knowledge, this is the first method that efficiently utilizes the atom bond information. It substantially outperforms current state-of-the-art methods.

Tracked centers. The third contribution is a temporal model for representing dynamic surfaces, called *tracked centers* [DVV21; Dvo+22a; DHV23]. We track a fixed set of points (denoted centers) inside a volume enclosed by the surface, each representing a small volume surrounding it, whose positions vary in time. The model was designed to address the limited versatility of current temporal models used in time-varying mesh compression. It can be already used to encode time-varying meshes, albeit without preserving the connectivity. It is also useful for temporally-coherent editing and attribute mapping.

Priority-based connectivity coding for known geometry. In our fourth contribution, we propose a method to encode mesh connectivity if the vertex positions are already known to both the encoder and decoder [Dvo+22b]. The method was designed for time-varying mesh compression, where the geometry is mostly encoded separately from connectivity. The connectivity is encoded during a priority-driven traversal, which results in a symbol order that can be exploited by context-adaptive coding. Combined with newly proposed criteria to predict, which vertices are connected by an edge, and a prediction of boundary edges, the method achieves significant compression performance gains over the state of the art.

1.2 Structure of the thesis

In the rest of this chapter, we will discuss mesh compression from a more general point of view and also point out how the methods usually measure compression performance. In Chapter 2, we will describe the mesh and point cloud representations of dynamic surfaces. In Part II, we will comment on the current state of the art in the compression of dynamic meshes (Chapter 3), time-varying meshes (Chapter 4) and dynamic point clouds (Chapter 5). The main focus will be on the compression of time-varying meshes since this area is the most challenging of the three.

Part III will discuss our contributions. In Chapter 6, we will present our error propagation controlling modification of the Laplacian mesh compression. Then we will discuss our model-based method for compressing molecular dynamics trajectories in Chapter 7. Chapter 8 focuses on the *tracked centers* temporal model including a description of all the published versions of the algorithm for obtaining the model. In the last chapter of this part, we will present the priority-based connectivity coding method for known geometry. Finally, in Part IV, we will conclude the thesis and discuss our future plans in terms of time-varying mesh compression and the future of this area in general.

1.3 Problem Definition

This thesis considers the problem of mesh sequence compression. Compression is the process of transforming input data X, which are represented by a certain sequence of m bits, into a reduced representation of n bits, where $m \gg n$, from which one can obtain a reconstruction $\hat{X}$ by a reverse process called decompression. If the compression is lossless, X and $\hat{X}$ must be identical. In lossy compression, $\hat{X}$ is distorted and is required to resemble X only in a defined sense.

In our case, the input data is a sequence of triangle meshes represented by geometry (positions of the vertices of the mesh), connectivity (information about how the vertices of the mesh are connected) and other properties of each mesh. The scope of our research is the compression of only the geometry and connectivity, with an emphasis on geometric information. The data will be described in more detail in Chapter 2.

In terms of mesh geometry compression, the majority of methods are lossy, meaning the reconstructed positions are different from the original ones. This is because positions are represented by vectors of floating point values, and such data is quite difficult to efficiently encode in a lossless manner. Loss of the information in this case is also less likely to be detected by a viewer than if it occurred in connectivity.

For connectivity, the key criterion to classify a method as lossless is whether a map exists between the vertices of the original and reconstructed mesh, which is an isomorphism. Such a definition permits lossless methods to reorder vertices, as long as the original structure (number of vertices and connections between them) is preserved. In this thesis, mesh sequence compression methods that perform lossless connectivity compression will be denoted structure-preserving. Lossy methods usually perform remeshing, simplification or filtering, which also implies loss of information in geometry. Unless the only purpose of the compressed mesh sequence is to be rendered (e.g., in entertainment or tele-immersion), it is desirable to preserve its original structure.

Although we also propose a method for compression of a special class of mesh sequences where the connectivity between the frames is static, our main focus is the problem of structure-preserving general mesh sequence compression. At first glance, one could assume that it should be simple to achieve better data rates than the intra-only methods by considering temporal coherence. However, due to a lack of temporal coherence in *vertex sampling* (the way the vertices are distributed over the surface), it is surprisingly difficult to exploit such information. In this thesis, we present techniques, that could contribute to a proposal of a novel compression method that addresses the problem of structure-preserving general mesh sequence compression. However, to this date, we have not presented such a method.

1.3.1 Data Reduction Techniques

In this section, we will briefly describe several key concepts of general data compression frequently used in methods discussed in Chapters 3, 4 and 5.

The most prominent tool of lossy geometry compression is *quantization*, a process of transforming a range or a large set of values into a smaller discrete set. The simplest and most commonly used type of quantization is performed by the trivial rounding:

$$\bar{x} = round(x/q),$$

where x is the input value and q is a *quantization constant* that controls the resolution. The values can be reconstructed up to a specified precision by simply multiplying by the quantization constant:

$$\hat{x} = q \cdot \bar{x}.$$

In our scenario, the quantization is usually applied to point coordinates or to data derived from them, which are usually real values represented in a single resp. double-precision floating-point number format requiring 32 (resp. 64) bits for storage in an uncompressed form. Assuming there are n possible values after quantization, we can already reduce the data rate by assigning each of the values a unique integer value represented by $\lceil \log_2 n \rceil$ bits. However, we may achieve better results by combining quantization with other data-reduction techniques.

At the end of the compression pipeline, there is usually a lossless encoding method, which attempts to exploit an underlying model of encoded symbols. In terms of mesh compression, most methods use *entropy coding* (e.g., arithmetic or Huffman coding [Huf52]), which can adjust the number of bits representing each symbol according to its probability of occurrence in the data without considering the actual context. The entropy coding method attempts to obtain an average number of bits per encoded symbol that is close to its optimal lower bound – Shannon's entropy:

$$H = -\sum_{x \in S} p(x) \log_2 p(x), \tag{1.1}$$

where S is a set of all possible symbols and $p(x)$ is a probability of occurence of the symbol x. Less often, a *dictionary-based coding* method (e.g. LZW) is used. Such methods attempt to exploit the context of data by searching for recurring patterns of symbols and encoding a reference to a dictionary of patterns constructed during encoding instead.

Compressed data contains a lot of redundant information that can be simply deduced from the coherence of values. A powerful tool for removing such redundancy is a *prediction*. Instead of encoding the original value, the encoder can predict it using the information available from already processed data (which is also available during the decompression) and encode only the difference. The prediction itself is a lossless process since it does not reduce the number of encoded values, however, when combined with entropic coding, it may result in a decreased bit rate. In terms of mesh sequence compression, we distinguish two types of prediction: *intra-* and *inter*-based. Intra-based prediction exploits the coherence in a single frame, while inter-based prediction exploits the temporal coherence between frames. The majority of compression methods we will discuss use both prediction types to some extent. We will omit the description of intra-only methods, which are equivalent to applying static mesh compression (a well-studied field) to each separate frame.

From Eq. 1.1, it follows that the higher the probability of the encoded symbol, the lower the number of bits required to represent its single instance in entropy coding. Similarly to prediction, one can use the information obtained from already-processed values to reduce the data rate using *context modelling*. Instead of the overall probability of symbol $p(x)$, a conditional probability $p(x|ctx)$, given a context ctx, is utilized by the coder. If ctx is chosen reasonably well, the conditional probability of the correct encoded symbol should be higher. A context can be deduced from various information, e.g., the previously encoded value. Instead of a single context, a method might also consider a set of predefined contexts $C = \{ctx_1, ctx_2, \dots, ctx_n\}$ and during encoding, select one based on certain criteria. This technique is called *context switching*. Another technique, which aims to improve the odds of choosing the best context, is *context selection*. It examines the data beforehand and selects which values should be used to form the context. Other than working with the context directly, it is also possible to use context-adaptive coding (e.g., CABAC [MSW03]) and influence the performance by coding values in a certain order (e.g., ascending order).

Another powerful tool is *dimensionality reduction*. For a certain vector $\mathbf{v} \in \mathbb{R}^n$, the goal is to obtain a transformed vector $\hat{\mathbf{v}} \in \mathcal{X}$ in a certain subspace $\mathcal{X} \subset \mathbb{R}^n$ of dimension $k = dim(\mathcal{X}),\ k \ll n$, which we can represent more compactly. While there is no limitation on how this is achieved, due to its simplicity, most approaches are based on orthogonal projection – for a certain vector $\mathbf{u}$, the closest vector $\mathbf{u}_w$ in the direction of unit vector $\mathbf{w}$ is computed as follows:

$$\mathbf{u}_w = (\mathbf{u} \cdot \mathbf{w})\mathbf{w}.$$

Given a set of orthonormal vectors $B_{\mathcal{X}} = \{\mathbf{x}_1, \dots \mathbf{x}_k\}$ forming a basis of $\mathcal{X}$, the $\hat{\mathbf{v}}$ can be expressed as their linear combination:

$$\hat{\mathbf{v}} = \sum_{i=1}^{k} \alpha_i \mathbf{x}_i, \tag{1.2}$$

where $\alpha_i = \mathbf{v} \cdot \mathbf{x}_i$. It can be shown that $\hat{\mathbf{v}} = \arg\min_{\mathbf{w} \in \mathcal{X}} \|\mathbf{v} - \mathbf{w}\|_2$. While $\hat{\mathbf{v}}$ is still of size n, we can actually encode a vector $\mathbf{a} = (\alpha_1, \dots \alpha_k)^\top$, $\mathbf{a} \in \mathbb{R}^k$. The $\hat{\mathbf{v}}$ can be fully recovered using Eq. 1.2, as long as both the encoder and the decoder have access to $B_{\mathcal{X}}$. Unless $\mathbf{v} \in \mathcal{X}$, the transformation is lossy. The process allows for progressive coding by incrementally extending the basis and encoding additional projection coefficients. The most crucial part is selecting the appropriate $B_{\mathcal{X}}$, with the objective being to achieve the lowest possible data rate while discarding the information that is less likely to be detected as missing.

One way to find $B_{\mathcal{X}}$ is using *Principal Component Analysis* (PCA), which, for a set of vectors in $\mathbb{R}^n$ with centroid at origin, finds an ordered orthonormal basis of the given space, where each basis vector represents a direction with the most variance in data if the information present in previous basis vectors was removed. This is achieved by singular value decomposition (SVD) of a matrix with encoded vectors as rows. The PCA requires all the encoded vectors to be known, and thus the basis cannot be constructed by the decoder. However, for a fairly coherent set of encoded vectors, the majority of the information is present in the first few principal directions, and a significant reduction of the number of encoded values can be achieved even when the basis vectors are encoded alongside the projection coefficients. Note that there are ways to efficiently store PCA data, for example by adaptive quantization, which exploits the fact that the importance of the present information decreases for subsequent principal directions [VS09].

If the encoded vector $\mathbf{v} \in \mathbb{R}^n$ represents a signal over a certain discrete domain (e.g., n vertices of a mesh or a graph), $B_{\mathcal{X}}$ can be chosen as a subset of frequency basis. For a relatively smooth signal, most of the information is contained in lower frequencies, and the high-frequency information can be discarded without being noticeable. The dimensionality reduction, in this case, is equivalent to the *Discrete Fourier Transform* (the projection) followed by a low-pass filter (selection of certain elements of frequency basis to form the $B_{\mathcal{X}}$). One can exploit the fact that in the continuous case, the sine and cosine functions forming the frequency basis are eigenfunctions of the Laplace operator and find the $B_{\mathcal{X}}$ as a set of eigenvectors of the Laplacian matrix **L** corresponding to a certain discretization of the Laplace operator over the specified domain, as long as the **L** is real, symmetric, and positive semi-definite. For this purpose, on graphs, the compression methods use a graph Laplacian matrix. On meshes, this matrix corresponds to a discretization mostly referred to as Kirchhoff's (the connection between meshes and graphs will be explained in Section 2.1). Additionally, the Cotan discretization [PP93] can be used. In such a case, the $B_{\mathcal{X}}$ is referred to as a *Manifold Harmonic Basis*. However, since Cotan discretization requires mesh geometry to construct L, some reference geometry must be known, or it can be only used to compress other mesh properties. Full eigendecomposition of **L** is not required since we are interested in only a selected subset of eigenvectors.

1.3.2 Performance Evaluation

The efficiency of lossless compression is quantified by the resulting *data size*. Lossy compression, however, requires relating the size to the amount of introduced *distortion*. In the rest of this section, we will describe how such measures are evaluated in terms of meshes.

While the *overall size* of the compressed data is the simplest measure, it is much more difficult to relate the results between different input datasets. Much more prominent is *data rate*, which measures the number of bits required to represent a certain element of data, e.g., a vertex or a face. Another popular measure is the *compression ratio*, the relationship between the sizes of the original and compressed data.

Measuring the distortion that is present in compressed meshes is a much more difficult problem. Depending on the purpose of the data, one must select the correct metric. Additionally, the performance of various compression methods differs depending on the metrics used in the evaluation.

Technical applications require an upper bound on the error in absolute coordinates, which is usually determined from the precision of the technology processing the data (e.g., in manufacturing). These objectives require *mechanistic* error metrics such as *Mean Squared Error* (MSE) and *Peak Signal to Noise Ratio* (PSNR), which relate corresponding vertex positions in original and distorted meshes. When the isomorphism between meshes is lost, one can use the *Haussdorf distance* or *Chamfer distance*. These are, however, much more expensive to evaluate, because they require frequent closest-point query evaluation. An example of a static mesh compression method that performs well under mechanistic criteria is the *Parallelogram prediction* [TG98].

In other areas (e.g., in entertainment or marketing), visual similarity is much more important. To this end, it is better to use *perceptual* metrics. These metrics are shown to better correlate with human perception of distortion in data by comparing their results to various user studies. Examples of such metrics are *MSDM2* [Lav11], *DAME* [VR12], *FMPD* [WTM12], *TPDM* [TWC14] and *TPDMSP* [Fen+18], to name a few. For more information on this topic, we refer the reader to the survey by Corsini et al. [Cor+13]. An example of a static mesh compression method that performs well under perceptual criteria is *High-pass coding* [SCT03].

An additional concern arises in the dynamic setting, where a sequence of meshes is considered. Even for a small distortion, a large discrepancy between the frames can occur. As an example of such behaviour, Corsini et al. [Cor+13] discuss a case in which one frame of the animation is distorted by applying a sine function, while the next frame is distorted by a cosine function. Although this distortion might not be visible when examining the frames separately, it results in a flickering effect, which is easily detectable when the frames are examined in fast succession. Quite a few methods for compressing mesh sequences use static metrics applied on separate frames even though such an approach cannot detect this behavior. This is because no temporal metric has been proposed for general mesh sequences. Only a few temporal metrics are designed for dynamic meshes, a special class of mesh sequences with common connectivity (e.g. *KG* error [KG04] and *STED* [VS11]).

Not only do various methods perform differently under various metrics, but their performance also differs for different data rates. For a more thorough evaluation of performance, *Rate-Distortion* (RD) curves are used. They show how the change in data rate influences the distortion in a selected metric. Figure 1.3 shows an example comparison of multiple compression methods using RD curves. It is trivial to obtain an RD curve for a method that is controlled by a single parameter directly influencing the data rate (e.g., the quantization constant). A more complex configuration, however, requires an optimization process (e.g., [VP11]) to find a curve that represents the best performance achievable by the method.

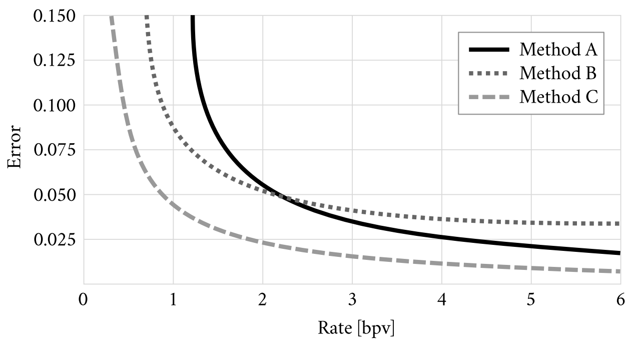


Figure 1.3: An example of an RD curve. While Method A performs better than Method B at higher data rates, it is unclear which method has a better overall performance. On the other hand, Method C outperforms other methods at all data rates.

2 Mesh and Point Cloud Representations of Dynamic Surfaces

Data considered in this thesis is assumed to represent a continuously moving two-dimensional, smooth manifold surface $\mathcal{S}$ embedded in $\mathbb{R}^3$ and sampled at discrete points in time. A representation $\mathcal{F}_i$ (resp. $\mathcal{M}_i$ for mesh, $\mathcal{P}_i$ for point cloud) of each sampled time point t_i will be referred to as a *frame*. The continuous movement implies temporal coherence in frames in the sense that two consecutive frames are often visually nearly indistinguishable.

The temporal coherence between the frames can be described by a *correspondence* function $f_{ij} : \mathcal{F}_i \mapsto \mathcal{F}_j$, which for any point $\mathbf{x} \in \mathcal{F}_i$ assigns a corresponding point $f_{ij}(\mathbf{x}) \in \mathcal{F}_j$ if it exists. Correspondences not only can be used for inter-frame prediction but also allow temporally coherent mapping of values on the surfaces, e.g., texture [BLW12].

There are multiple criteria to consider when choosing an appropriate surface representation. The most important is representation versatility. We are interested not only in what classes of surfaces can be represented but also at what cost. Other things to consider are obtaining, rendering and processing complexity. Since this thesis studies the representations from the compression perspective, we will also list some specific properties for this problem.

2.1 Triangle Mesh Sequences

Triangle mesh is currently considered the most popular representation of 3D surfaces due to its simplicity, approximation quality, and native support on graphics cards. It is a piecewise planar surface usually defined as $\mathcal{M} = (V, T)$, where V is a set of vertices, and T is a set of triangles that connect them. A vertex is usually represented by its geometry (position) and properties (normal, colour, texture coordinate, etc.), while a triangle is represented as an ordered triplet of indices, and the order of vertices induces its orientation (the direction of a normal). Considering a set of edges E that form all the triangles in T, we can also interpret the mesh as an undirected graph $G = (V, E)$. This allows the application of many techniques from graph theory. When talking about mesh geometry, we will refer to the positions of vertices, while connectivity will refer to the combinatorial information (vertex indices, triangles, or edges). For the sake of simplicity, we will focus only on orientable manifold meshes, i.e., meshes in which the orientation of triangles can be unified, vertices coincide only with a single triangle fan, and no edge is connected to more than two triangles. Nevertheless, some of the listed related work in Section 4 did not make such assumptions.

2.1.1 Dynamic Mesh

Dynamic meshes (DMs) are a special class of triangle mesh sequences. The distinguishing property of a dynamic mesh is a *common connectivity* T shared by all frames:

$$T_0 = T_1 = \ldots = T_{n-1} = T,$$

where n is the number of frames. This also indicates that the number of vertices and their order are constant through time. Only the geometry and properties change between frames, and thus the connectivity needs to be encoded only once.

One of the main advantages of a dynamic mesh is that the vertex correspondences are explicitly coded in the connectivity:

$$\forall v_k \in V : f_{ij}(\mathbf{x}_k^i) = \mathbf{x}_k^j$$

for any pair of frames (i, j), where $\mathbf{x}_k^i$ is the position of k-th vertex in the i-th frame. The correspondence function for any point $\mathbf{x}^i \in \mathcal{M}_i$ can be directly generalized from vertex correspondences using barycentric coordinates $(\lambda_a, \lambda_b, \lambda_c) : \mathbf{x}^i = \lambda_a \mathbf{x}_a^i + \lambda_b \mathbf{x}_b^i + \lambda_c \mathbf{x}_c^i$ in triangle $t = (v_a, v_b, v_c)$, which contains x:

$$f_{ij}(\mathbf{x}^i) = \lambda_a f_{ij}(\mathbf{x}_a^i) + \lambda_b f_{ij}(\mathbf{x}_b^i) + \lambda_c f_{ij}(\mathbf{x}_c^i) = \lambda_a \mathbf{x}_a^j + \lambda_b \mathbf{x}_b^j + \lambda_c \mathbf{x}_c^j.$$

For dynamic meshes, the function f_{ij} is always an isomorphism. Instead of treating the geometry of each frame separately by assigning each vertex v_k a position $\mathbf{x}_k^i \in \mathbb{R}^3$, a static mesh $\mathcal{M} = (V, T)$ can be considered, where for each vertex v_k the geometry is represented as a trajectory $\mathbf{t}_k \in \mathbb{R}^{3n}$. This allows for global coding approaches (in terms of time), e.g., PCA coding [VS07].

The simplest dynamic meshes are usually synthetic data obtained by continuously deforming a certain shape using, for example, skinning. It is, however, often more efficient to encode the deformation parameters than to encode the resulting mesh sequence. More complex sequences are obtained by surface tracking or by 4D reconstruction (e.g., [Vla+08; Tev+12]); however, most methods require a prior mathematical model of the data as an input (e.g., template shape), and the whole process is quite complex and prone to errors.

Visualizing a dynamic mesh usually consists of rendering the first frame and then updating only the geometry, which allows for a higher rendering frequency. While simple and easy to work with, the dynamic mesh lacks representation versatility. Not only is the time-evolving topology not allowed, but since the connectivity complexity directly influences the ability to represent fine details, any fine detail must also be accounted for even if it appears only in a single frame.

2.1.2 Time-Varying Mesh

Time-varying mesh (TVM) is any mesh sequence in which the number of vertices and/or connectivity changes over time. While the temporal coherence of the connectivity might be present (e.g., in synthetic data), it cannot be generally assumed. Thus, most of the time, the connectivity must be encoded for each frame separately.

Not only we cannot directly derive the correspondences of time-varying mesh frames from the connectivity, but they are also difficult to estimate. This is caused by the fact that the bijective property of the correspondence function is lost with the merging and separation of parts (see Figure 2.1). This renders exploiting the temporal coherence a much more difficult problem.

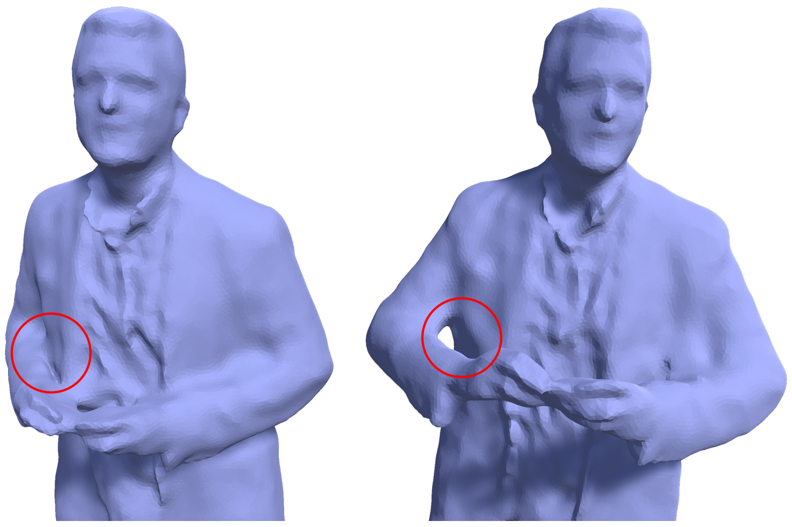


Figure 2.1: Example of bijectivity loss of correspondences. There are no correspondences for vertices at the back of the arm in the highlighted area.

The earliest time-varying meshes were similar to dynamic meshes in the sense that the connectivity changed only by simple updates (e.g., by subdivision, contraction, or vertex removal). More importantly, TVMs have been used to represent dynamic volumetric environments (e.g., fluid simulations). Such data is usually obtained by extracting the iso-surface of each frame from a particular implicit function.

In recent years, improvements in the performance of capturing systems allowed real-time surface capture of dynamic scenes. Such systems can output a large number of mesh frames; however, these contain a considerable amount of noise and self-contact. This leads to frequent spurious topology changes, even for surfaces that initially showed constant topology. While in some scenarios such noisy TVMs could be converted to a dynamic mesh with great difficulty, often it is much more preferable to work directly with the time-varying mesh data.

Visualizing a TVM requires rendering a different mesh in each separate frame, which made it almost impossible to render at satisfying frame rates on past consumer-grade devices. As computer technology advances, this is becoming less of an issue and the advantages of the TVMs prevail. The connectivity can adapt to accommodate any detail in the data at any time, allowing an increase in the complexity of the mesh frame only when it is necessary.

2.2 Dynamic Point Cloud

A *point cloud* is a set of points $\mathcal{P} = \{\mathbf{x}_0, \dots \mathbf{x}_{m-1}\}, \mathbf{x}_k \in \mathbb{R}^3$, sampled from the represented surface, where m is the number of points. Similarly to a triangle mesh vertex, a point can be represented by its geometry and attributes. In the case of point clouds, the literature does not consider any special cases of sequences in which only the positions of points change over time, and any point cloud sequence is usually referred to as a *dynamic point cloud* (DPC). From the compression standpoint, a point cloud frame $\mathcal{P}_i$ is equivalent to a mesh $\mathcal{M}_i = (V_i, \emptyset)$ with an empty set of triangles. For this reason, a dynamic point cloud compression combined with connectivity coding can be considered a method for the compression of time-varying meshes and, conversely, a TVM compression method that ignores the connectivity data can be considered a method for compression of dynamic point clouds.

A correspondence function of point clouds can be defined only on the points since the point cloud is discrete and there is no surface representation between the points. Similarly to TVMs, correspondences on point clouds are generally not bijective and are challenging to estimate.

Dynamic point clouds have also benefited from recent improvements in capturing systems. Compared to time-varying meshes, they can be obtained at a much lower cost. A special type of dynamic point cloud is obtained using LiDAR (Light Detection and Ranging) sensors. Such sensors are usually mounted on a moving vehicle (e.g., a car or drone) and emit light rays in rotating motion while measuring the distance from the point where the ray hits the scanned environment. LiDAR data is commonly used for navigating autonomous vehicles. Unlike the general point clouds, the frames are usually also accompanied by additional measurements (e.g., accelerometer data). While it is possible to treat the LiDAR point cloud as an image, in which the position of each pixel represents cylindrical coordinates and the value represents the distance, several compression methods treat it as a point cloud, while still using its specific properties.

Rendering dynamic point clouds is easier than rendering TVMs since no connectivity has to be passed to the GPU. The points are usually rendered as a certain primitive, e.g., a cube, a sphere, or a disc. Selecting an appropriate scale for the rendered primitives given the current viewpoint should result in the illusion of a smooth surface.

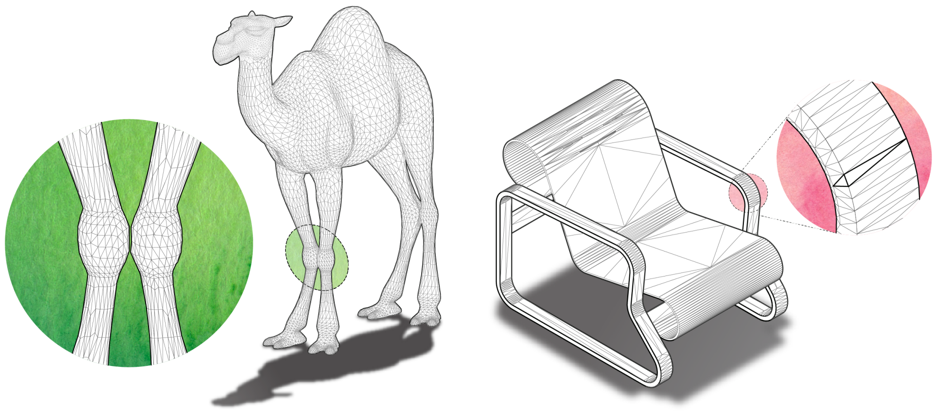


Figure 2.2: Advantages of the mesh representation over point clouds. *Left*: Separation of geodetically distant vertices near in space. *Right*: Highly non-uniform representation. Illustrations are courtesy of Hanocka et al. [Han+19].

The main advantage of the point cloud representation over a triangle mesh is its simplicity, which allows the storage and processing of even large models with points numbered in the millions. On the other hand, since it lacks information about connectivity, obtaining the direct neighbourhood of a point must be done by querying positions in space, which is much more difficult than just examining the edges in the case of a triangle mesh, and the result might contain points that are near in space, but quite far apart in terms of geodetic distance. Additionally, the mesh allows for a highly non-uniform representation in which planar regions with a lack of detail can be represented by a small number of large triangles and, conversely, regions with lots of detail can be represented by a large number of small triangles. In the case of point clouds, highly non-uniform sampling might result in visible holes in a surface during rendering. Both of these advantages of the triangle mesh are shown in Figure 2.2.

2.2.1 Voxelization

A majority of dynamic point cloud compression methods consider so-called voxelized point clouds. Voxelization is a process in which the point cloud is stored in a cubic grid of size $2^d \times 2^d \times 2^d$, where d is a parameter controlling the level of detail. Each cell is marked occupied or unoccupied, based on whether it contains any of the input points. All the points inside an occupied grid cell are discarded and replaced by the centroid of the cell. An octree is usually built from the binary occupancy data (see Figure 2.3) since it allows more efficient storage and progressive coding.

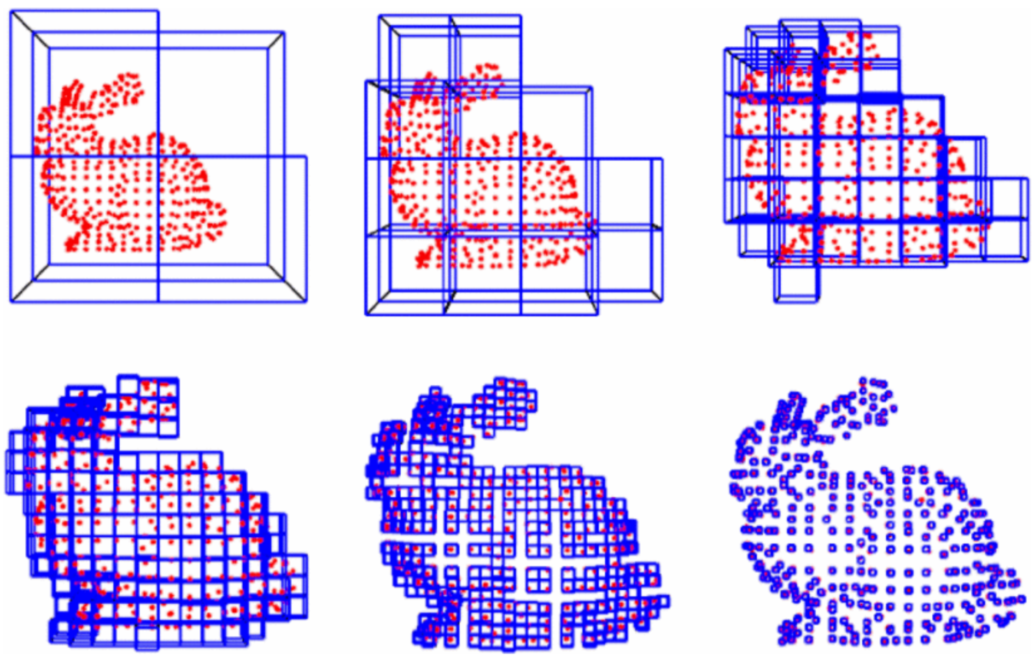


Figure 2.3: Different levels of a single octree representing a voxelized point cloud. Source: [Kam+12]

Voxelization is a process similar to quantization in the sense that the real-valued coordinates of points are transformed into grid indices. However, it also alters the number of points, compromising the one-to-one correspondence between the original and voxelized data. This is why certain dynamic voxelized point cloud compression methods claim to be lossless – the voxelized data already have reduced precision.

Part II
Related work

3 Dynamic Mesh Compression

Due to the advantages of the dynamic mesh representation listed in Section 2.1.1, it is possible to design highly efficient compression methods using both spatial and temporal prediction. This is why the development of dynamic mesh compression was so divergent from the development of time-varying mesh compression. As of 2023, this field is already considered well-studied. Current methods usually use techniques like *segmentation* and *clustering* to group parts (frames or vertices) of similar motion together, dimensionality reduction using *PCA*, *wavelets* to allow progressive coding, or *spatiotemporal prediction* of vertex positions [Mag+15].

Since the main scope of this thesis is TVM compression, we will describe only the most recent dynamic mesh compression methods and we will not group them in sections as we will for TVM and DPC compression. Additionally, most of the current methods usually combine multiple techniques at once, which makes such categorization difficult. For more details about this field, we refer the reader to the 3D mesh compression survey by Maglo et al. [Mag+15].

3.1 Recent methods

Hajizadeh et al. [HE15] proposed to extract a set of keyframes from the sequence based on a clustering of the frames. For each non-keyframe, the method finds the optimum linear blending weights of the keyframes to predict its geometry. They encode the blending weights and residuals.

Hachani et al. [HZP16] segment the vertices into patches based on heat diffusion properties of the surface. Each patch is assigned a sequence of affine transforms which map its positions in the first frame to the rest of the frames. The position of each vertex is predicted as a weighted sum of these affine transforms at a given frame, with fixed weights optimized over all the frames encoded once.

Hou et al. [Hou+17] proposed a data compression method based on *sparse low-rank matrix approximation* (SLRMA), which decomposes a matrix into a product of a sparse column-orthogonal matrix and a coefficient matrix. One of the domains where the authors tested this approach was dynamic mesh compression. In this domain, SLRMA allows for exploiting both spatial and temporal coherence.

Lalos et al. [Lal+17] adapted the PCA approach for interactive scenarios. Their method works with blocks of frames. The SVD is computed only for the first block. For any subsequent block, it reuses the previous PCA basis and adapts it to approximate the current one using *orthogonal iterations* algorithm [Str97] without the need for performing additional SVD. The method is also suitable for out-of-core compression since it does not require all the frames to be available simultaneously. In subsequent work, the orthogonal iterations approach was also combined with laplacian compression to improve perceptual performance [ALM21].

Hajizadeh et al. [HE17] separated the geometry of frames into coarse and fine information. The coarse information is extracted by projecting the geometry into the frequency basis of graph Laplacian. The fine detail information is the difference between the coarse and the original geometry represented in a local rotation-invariant frame. Assuming the fine details do not change, only the details of the first frame must be extracted and transmitted.

A similar approach was also proposed by Chen et al. [Che+18; Che+19]. For extracting the coarse information, they, however, use *manifold harmonics basis* [VL08]. This requires a certain geometry to be known so that a Cotan Laplacian matrix can be constructed. For this reason, they cluster the frames and for each cluster, a keyframe is encoded. To reintroduce the fine detail, they use the *deformation transfer* algorithm [SP04].

Yang et al. [Yan+18a] proposed a progressive coding scheme, which also divides the sequence into subsequences. The bounding frames of these subsequences are selected by finding a certain number of points of highest curvature on a trajectory curve of mesh centroid. These points should correspond to frames where the most significant movement occurs. Each subsequence is then encoded by PCA, with projection coefficients further reduced by *spectral graph wavelet transform* [LJF13]. The wavelet coefficients are encoded using CSPECK [LHS10].

In their subsequent work, Yang et al. [Yan+19] proposed a different approach, albeit also based on studying differential properties of trajectory curves. This time, they studied the similarity of curvature and torsion of vertex trajectory curves to perform both temporal and spatial motion-based segmentation. The trajectories of each segment are transformed into a frequency domain using Graph Fourier Transform and coefficients corresponding to high-frequency information are discarded. The rest of the coefficients are encoded using SPIHT [SP96].

The method proposed by Luo et al. [Luo+19] segments the sequence in both spatial and temporal domains, thus obtaining patches under fairly simple deformation that are encoded more efficiently than they would be if encoded together using PCA. In later work, the authors were able to reduce the data rate further by incorporating different ways of storing PCA data [Luo+20; Luo+21].

To some extent, the method proposed by Hajizadeh et al. [HE19] is similar to the one proposed by Hachani et al. [HZP16]. They also divide the surface into patches, this time using k-means clustering over vectors containing coefficients of affine transforms which map given vertex position between subsequent frames. To predict vertex positions, they estimate a second-degree polynomial transform for each patch. These transforms are also weighted, but the weights are assigned uniformly based on the patches which are incident to the given vertex.

Arvanitis et al. [ALM19] proposed a scalable method, which transmits only a reduced number of vertices for each frame based on network capabilities. The vertices to be transmitted are selected considering both spatial and temporal information. The rest of the geometry is then reconstructed in a coarse and fine process.

3.2 Summary

Current methods for dynamic compression are already quite efficient in terms of data rates, since the reconstructed data already looks indistinguishable from the original at a data rate of 2 to 5 bits per vertex in a single frame, depending on the complexity of the input data. The authors nowadays focus more on scalability [Lal+17; Yan+18a; Yan+19; ALM19; ALM21], interactivity [Lal+17; ALM19; ALM21], or perceptual performance [ALM21] or achieve only incremental improvements in terms of data rates.

Due to the advances in the retrieval of TVMs, and the lack of versatility of dynamic mesh representation, the popularity of DM compression is rapidly decreasing. This is also reflected in the fact that MPEG is already abandoning the original MPEG-FAMC codec for dynamic meshes in favour of a more versatile method which also handles TVMs. We believe that in the following years, this trend will continue and that there might still be a few novel methods proposed, but any major breakthrough is improbable.

Time-Varying Mesh Compression 4

The field of TVM compression is already well-established, with the earliest known publication from the year 2003 [GSK03]. Nevertheless, it is still considered an open problem, which leads to applications preferring to use intra-only methods. In this chapter, we briefly list all the methods we are aware of, which are categorized into sections, based on the type of inter-prediction they use. Then, we will also discuss the current challenges and hint at related methods or paradigms, which could potentially be beneficial if one decides to address these challenges. The content of this chapter is summarized in Table 4.1.

Table 4.1: Overview of existing methods for TVM compression. Methods are in the order in which they appear in the chapter. Highlighted methods are considered state-of-the-art. *Ref.* column contains the most relevant reference for the given method. Columns *Versatility* and *Designed for* show the type of input data the method handles (see Section 4.5.2). Columns *Iso.* and *Conn.* show whether the method preservers structure (Isomorphism) and whether it addresses connectivity coding (see Section 4.5.1). The colour in the Conn. column indicates efficiency (green = efficient, orange = inefficient, red = not addressed). Symbol meaning: ✓ = Yes, ✗ = No, ✱ = Optional.

Method	Ref.	Category	Subcategory	Versatility	Designed for	Iso.	Conn.
Patch ICP	[GSK03]	ME	—	General	Synthetic	✓	✓
EBMA	[HYA07]	ME/Struct.	—	General	General	✓	✗
PCA-aligned patches	[YA10]	ME	—	General	General	✓	✓
Grid occupancy XOR	[HYA08]	Struct.	Grid	General	General	✓	✗
Semi-regular representation	[YKL06]	Model	Tracked surface	Const. GT Top.	Const. Top	✗	-
Reeb graph matching	[TSM07]	Model	Reeb graph	Const. GT Top.	Human	✗	-
Topology dictionary	[TM12]	Model	Reeb graph	General	Human	✗	-
Skinned mesh	[MYA08]	Model	Tracked skeleton	Const. GT Top.	Human	✗	-
Per-bone ICP	[Dou+14b]	Model	Tracked skeleton	Const. GT Top.	Human	✱	✓
Occupancy network	[ZGT23]	Model	Neural model	General	General	✗	-
SkinOff	[HKM04]	Video	Geometry video	Textured	Textured	✗	-
EIP geometry video	[Xia+10]	Video	Geometry video	Face	Face	✗	-
Cut over local extrema	[TM13]	Video	Geometry video	General	Human	✗	-
Polycube geometry video	[Hou+14a]	Video	Geometry video	Const. Top.	Const. Top.	✗	-
VPCC + Edgebreaker	[FJB20]	Video	Projection	General	General	✓	✓
VDMC Nokia	[Alf+22]	Video	Projection	General	General	✗	-
VDMC Tencent	[Hua+22]	Video	Geometry video	General	General	✗	-
VDMC Apple	[Mam+22]	Model/Video	Geometry video	General	General	✗	-

4.1 Motion estimation

Motion estimation (ME) is a technique often used in video compression. It assigns so-called *motion vectors* to blocks of pixels of the previous frame describing how these pixels moved. This information is then used for the prediction (or replacement) of the current frame. It is possible to apply an analogous process for mesh sequence compression by aligning parts of two subsequent frames. While motion estimation is often used in approaches across our specified categories, the methods discussed in this section rely on it solely. In general, these methods proceed as follows (see also Figure 4.1): The first frame is encoded in an intra fashion. For each subsequent frame, a reference shape is obtained by aligning the current frame with the previous frame (or the previous keyframe). Parameters of the alignment are encoded as well so that the reference shape can also be reconstructed by the decoder. This reference shape is then used to predict (or replace) the coded frame. The main advantage of such approaches is their simplicity which comes at the cost of compression performance.

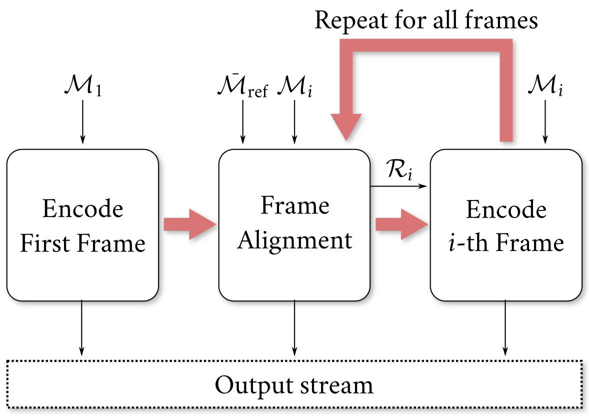


Figure 4.1: Example pipeline of a TVM compression based on motion estimation. Thin arrows represent the data flow, while bold arrows represent the order of operations. $\mathcal{M}_i$ denotes a mesh at the i-th frame and $\mathcal{M}_{\text{ref}}$ denotes a certain mesh (previous frame or keyframe) used to obtain a reference shape $\mathcal{R}_i$. The bar above a symbol denotes data distorted by compression.

The first method to address TVM compression, although only for synthetic sequences, was proposed by Gupta et al. [GSK03]. It uses the iterative closest point (ICP) algorithm [BM92] to rigidly map patches of the coded frame onto the whole mesh of the previous frame. After alignment, the patches are refined to merge segments with a similar motion. Depending on the prediction error of vertex position given the estimated rigid transformations, the vertices are divided into three sets. The first group can be represented solely by the rigid transformation, the second group requires correction vectors to be encoded and the rest is encoded without exploiting temporal information. The method, however, expects temporal coherence in connectivity, since they assume it changes in simple updates (e.g., vertex insertion/removal, subdivision), which, in general, does not occur in real-world data.

Han et al. [HYA07] proposed a method based on the block matching algorithm (BMA) [JJ81], a widely used ME technique in video compression. The method divides the bounding box of the coded frame into cubic blocks of a specified size. The surface patch in each block is then translated for its centroid to lie at the centre of the block. For each block, a corresponding block in a defined search area of the previous frame is found by matching the weighted average normal vectors of the patches. Positions of vertices are predicted using correspondences derived from the nearest neighbour search.

Another method based on the fitting of patches was proposed by Yamasaki et al. [YA10]. Instead of registration, their approach first transforms all the patches into a common local coordinate frame obtained by principal component analysis. Each coded patch is predicted by a patch from the previous frame yielding the smallest correction vectors. To reconstruct such a patch, the decoder needs the index of the corresponding reference patch, the vertex correspondences between patches, the correction vectors, and a rigid transform which moves the patch from the local coordinate frame into the correct global position. Instead of encoding the vertex correspondences explicitly, for each reference vertex, the number of corresponding coded vertices is encoded followed by the matching correction vectors. This way, the decoder can reconstruct the original set of vertices, albeit in a different order.

4.2 Prediction of data structures

Instead of working directly with the geometry, the methods in this category construct spatial data structures over each frame and proceed with the inter-prediction of such structures. Although there is only one method in this category for time-varying meshes, in Section 5.2, we will discuss applying this technique to point clouds, where it is fairly popular due to the input data usually being voxelized point clouds stored in an octree. Compared to motion estimation, the prediction of data structures is often much simpler, since no local matching between subsequent frames is usually required.

Han et al. [HYA08] based their method on cubic binary grids. The method first transforms all the frames so that their centroid lies at the origin to maximize their overlap. Then, a coarse cubic grid is constructed over the sequences bounding box. For each frame, a binary function on the grid is evaluated, which indicates whether a cell contains any frame vertices. The temporal coherence is exploited using XOR operation on two subsequent frames (see Figure 4.2). This information is then encoded using the run-length encoding (RLE). Each cell that contains mesh geometry is then further subdivided at a finer scale, ideally so that each subcell contains at most a single vertex. This finer information is also encoded using RLE without further exploiting the temporal coherence. In subsequent work, Ferreira et al. [Fer+10] extended this approach for progressive coding using multiple levels of finer subdivisions.

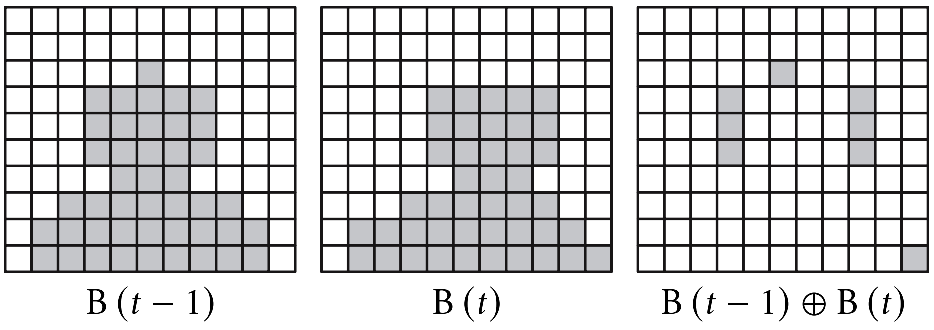


Figure 4.2: Exploiting coherence of binary grids between subsequent frames using XOR operation. Source: [HYA08]

4.3 Temporal-model-based methods

The most effective way to exploit temporal coherence in geometry is by employing temporal models capturing the dynamic behaviour of the sequence. These models are usually constructed with a particular assumption over the properties of the represented surface, such as static topology, articulated motion, or the sequence's representation of human performance. The model-based methods are fairly similar to the ME-based, however, the main difference is that instead of using only the information from the previous frame (or previous keyframe), they exploit the global temporal information to some extent. An example model-based compression pipeline is shown in Figure 4.3. Usually, the method first constructs the model considering all the frames (although some methods construct the model on the go while encoding individual frames). The model and the first frame of the sequence are then encoded. For each subsequent frame, the model is used to construct a specific reference shape $\mathcal{R}$ (not necessarily a mesh or a point cloud) from the previous frame (or previous keyframe). $\mathcal{R}$ is then used for geometry prediction or replacement of the current frame.

The model-based methods can be further divided into categories based on the temporal model they use. The most prominent approach for obtaining a temporal model in TVM compression is *tracking*. Its goal is to obtain a temporally consistent representation of the original surface. This is usually done by gradually deforming some template (e.g., a *surface* [Li+09] or a *skeleton* [MYA08]) to align it with subsequent frames. The models obtained by tracking give us some insight into temporal correspondences. However, different models might capture different underlying information that could be used to reduce the redundancy of the data, for example, the shape or the topological structure, without describing the relations between the frames [ZGT23]. Note that many temporal models (e.g. a tracked surface) are themselves a reduced representation of the mesh sequence. For the sake of simplicity, only the methods that directly concern compression were studied in this section.

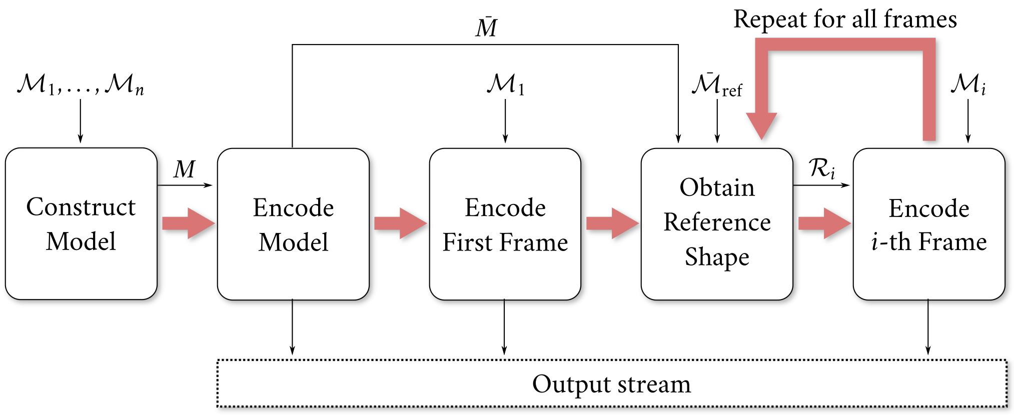


Figure 4.3: Example pipeline of a TVM compression based on a temporal model M. Thin arrows represent the data flow, while bold arrows represent the order of operations. $\mathcal{M}_i$ denotes a mesh at the i-th frame and $\mathcal{M}_{\text{ref}}$ denotes a certain mesh (previous frame or keyframe) used to obtain a reference shape $\mathcal{R}_i$. The bar above a symbol denotes data distorted by compression.

The first model-based method for TVM compression was proposed by Yang et al. [YKL06]. It replaces the original sequence with a dynamic mesh obtained by tracking the remeshed first frame. The remeshing is performed by simplification using the quadric error metric [GH97] followed by butterfly subdivision projected on the original surface. This creates multiple levels of detail. The sequence is then encoded progressively by exploiting the temporal coherence in an uplifting scheme.

Tung et al. [TSM07] utilised augmented multi-resolution Reeb graphs [TS05] of geodesic integral function

$$\mu(v) = \int_{\mathcal{S}} g(v, s)\, dS, \tag{4.1}$$

where g is the geodesic distance between two points on a surface $\mathcal{S}$, to capture the topological properties of the frames. Considering how the graph nodes are connected and whether they are located at the local extrema of the generating function, they show that it is possible to manually devise rules which allow their tracking if the underlying ground truth topology is known. The original mesh frames are replaced by the first frame deformed by this tracked graph structure. To accommodate for the fact that the ground truth topology is usually unknown, the approach was later generalised [TM12] by incorporating a so-called *topology dictionary*. It clusters frames according to the similarity of the underlying Reeb graphs. Frames in each cluster are then replaced by a deformed representative frame (one that is the most similar to all the frames in the cluster).

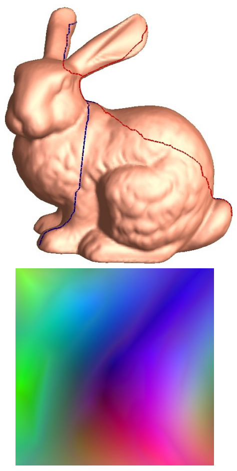
Maeda et al. [MYA08] proposed a compression method for Human TVMs that replaces the sequence with the first frame deformed by a tracked skeleton. In a subsequent work [NYA10], the method was further improved using adaptive simplification and prediction of the first frame by a reference human triangle mesh to allow transmission and displaying of such data on consumer-grade mobile devices at the time (the year 2010).

The method proposed by Doumanoglou et al. [Dou+14a; Dou+14b] also uses a tracked skeleton as a temporal model. The method per-bone aligns the skinned meshes of the current frame and a selected intra-coded keyframe using ICP [BM92]. The aligning transformations are encoded. The aligned intra-coded frame is then used for vertex position prediction identically to the prediction of Yamasaki et al. [YA10]. The method can optionally preserve the original connectivity or the compression rate can be further improved by replacing inter-coded frames with aligned intra-only encoded keyframes. In the original work, the last intra-only frame was always selected. The authors have also experimented with selecting the intra-only frames for prediction based on skeleton matching criteria and periodicity detection [Dou+14a], which in some cases led to an improvement in terms of rate-distortion performance.

Zaghetto et al. [ZGT23] have recently filed a patent on a method that replaces the mesh sequence by a neural representation using occupancy networks [Mes+19]. The method samples a coarse point cloud of the surfaces and learns the occupancy functions from the original geometry. Only the point clouds and the weights of the occupancy network are encoded. The proposition of the method seems promising, however, since it was published through a patent, no performance evaluation was given.

4.4 Video-based methods

Video compression methods are famous for their effectiveness in terms of exploiting temporal coherence. For this reason, several methods for TVM compression have focused on mapping the information contained in each frame into the image domain and encoding such data as a video. These methods perform notably well on textured mesh sequences since the video is a natural representation of time-varying texture information. The main challenge in video-based TVM compression is to find a mapping that produces the most temporally coherent images. Based on the type of mapping approaches, the video-based methods can be further divided into two categories - parametrization-based methods, which usually store geometry as a sequence of *Geometry images* (GI) [GGH02] (positions in 3d as RGB values in the texture), often also referred to as *Geometry videos* (GV) (although GV was initially a term coined to sequence of GIs used to compress dynamic meshes [Bri+03]), and orthogonal-projection-based methods, which project surface patches onto certain projection planes and then represent the geometry using occupancy information (which pixel is part of a certain patch) and the distance from the plane, similarly to depth images (see Figure 4.4). To achieve temporal coherence of images, the parametrization methods focus on the temporally consistent cutting of the mesh frames before mapping them onto the 2D domain, while projection-based approaches focus on the temporally consistent placement of patches. Note that the placement of geometry is not required to be perfectly consistent, since the video codec usually also utilizes ME in the video domain.


(a) Parameterization. Source: [GGH02]

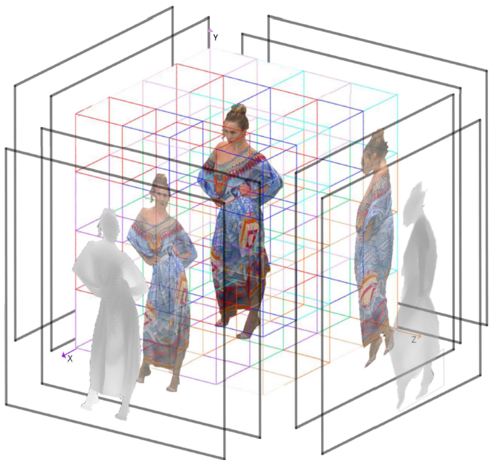

(b) Projection. Source: [Zhu+21]

Figure 4.4: The two main approaches of mapping mesh data onto the 2D domain.

The first authors, who considered video compression for TVMs, were Habe et al. [HKM04]. Their parametrization-based method attempts to place the cut path in places with lower texture complexity. Their main motivation is that the most distorted texture is near the cut path. If the cut path lead through parts of complex texture, the distortion would be much more visible. Although not verified experimentally, the authors also hint that such a cut path could be to some extent temporally consistent.

For compression of TVMs representing facial expressions extracted from motion data (see Figure 4.5(a)), Xia et al. [Xia+10] proposed a so-called expression-invariant parameterization (EIP). The method first cuts the mesh $\mathcal{M}$ along geodesics between the eyes and the mouth to obtain a surface with two boundary curves (an outer boundary $\partial\mathcal{M}_0$ and an inner boundary $\partial\mathcal{M}_1$). These curves are mapped consistently into the parametric domain using arc-length parameterization. Then, harmonic function f on the mesh is constructed by solving the Laplace equations with Dirichlet boundary conditions:

$$\begin{aligned} \Delta f &= 0 \\ f(u) &= 0, u \in \partial \mathcal{M}_0, \\ f(v) &= 1, v \in \partial \mathcal{M}_1. \end{aligned}$$

Analogically, a harmonic function g is also obtained in the parametric domain. Finally, the gradient flow ∇f is identified with the gradient flow ∇g. This process is explained in Figure 4.5. In their subsequent work, the authors were able to improve the compression performance of the approach by proposing different ways of encoding the resulting video data [Xia+12; Hou+12; Hou+13a; Hou+13b; Hou+14b].

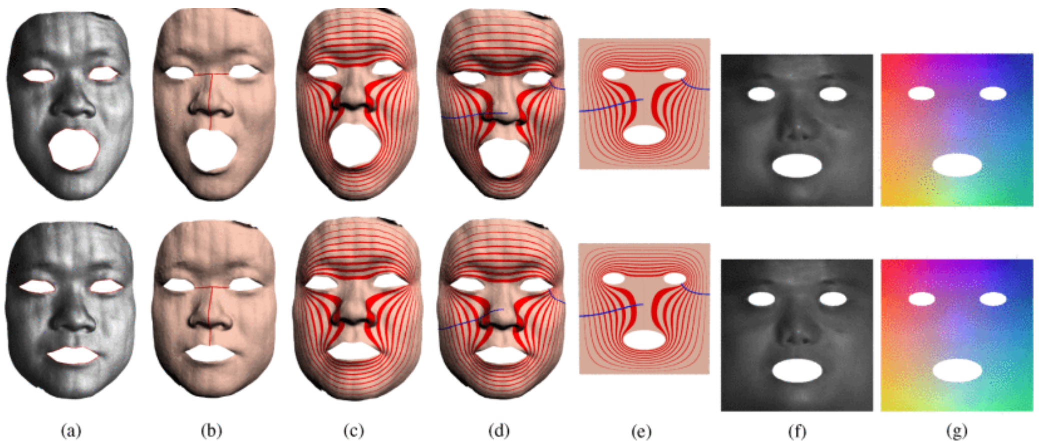


Figure 4.5: Expression-invariant parameterization (EIP). (a) Input data, (b) Performed cut, (c) Harmonic function with Dirichlet boundary conditions, (d) Example gradient flow curves, (e) Corresponding curves in the parametric domain, (f) Parameterization, (g) Geometry image. Source: [Xia+12].

A more sophisticated way of cutting the mesh to obtain temporal coherence was proposed by Tung et al. [TM13]. They track the local extrema of the geodesic integral function (see Eq. 4.1) and construct the cut as the shortest path between them (see Figure 4.6). Authors claim that if selected properly, such points should remain consistent unless a change in topology occurs.

A slightly different approach to compressing TVM as a sequence of geometry images was proposed by Hou et al. [Hou+14a]. Instead of direct planar mapping, the method maps frames to a polycube [Gar+13], which is cut at predefined edges and flattened. Although the structure of the polycube must be known before encoding, temporal coherence can be achieved by mapping consistently tracked salient points (e.g., features of the human body) onto the same positions in the 2D domain (see

Figure 4.7). In the original method, the video was decomposed into a low-rank approximation representation. Some improvements were achieved by representing the sequence as a linear interpolation between a set of keyframes, which were also reordered to increase their temporal coherence [Hou+15].

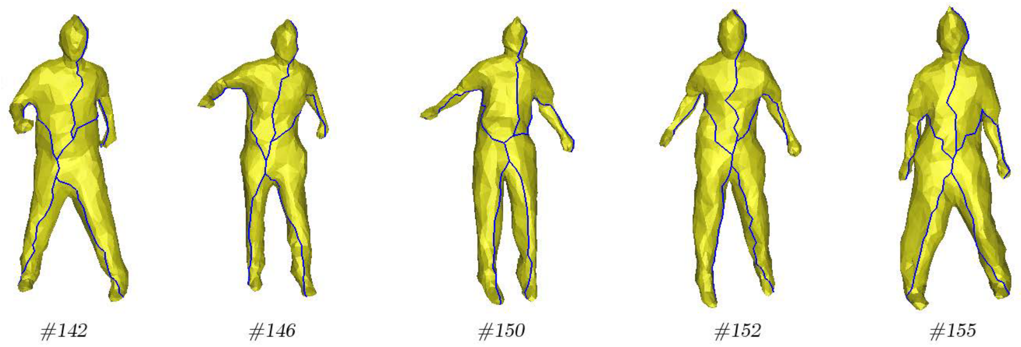


Figure 4.6: Temporally consistent cut between local extrema of the geodesic integral function. Source: [TM13].

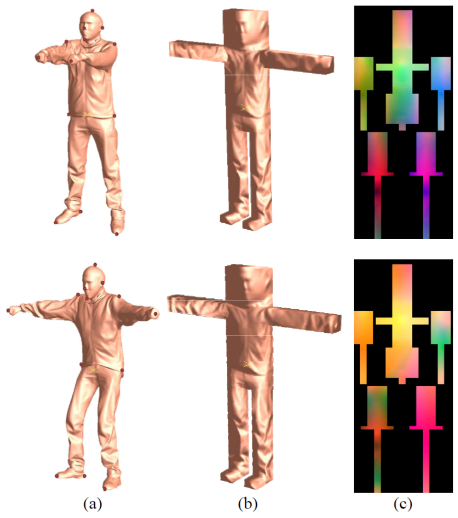


Figure 4.7: The process of obtaining a temporally coherent geometry video using polycube parameterization. (a) Salient points. (b) Polycube parameterization. (c) Geometry video. Source: [Hou+14a]

The MPEG-VPCC standard for compression of dynamic point clouds [ISO19], which will be described in Section 5.4.1, has recently popularized the orthogonal projection approach. Faramarzi et al. [FJB20] treated mesh geometry as a point cloud and encoded it using this method. For connectivity coding, they incorporated Edgebreaker [Ros99] or TFAN [MZP09] followed by encoding a vertex permutation map. Graziosi [Gra21] later improved this approach by adapting it for dense meshes and storing the connectivity as a 2D mesh. A similar pipeline is also described in

Sony's recent patent [GZT22], although it also describes an additional way of storing connectivity using triangle rasterization.

4.4.1 MPEG V-DMC

At the 136th meeting of the MPEG (Moving Picture Experts Group) in October 2021, a call for proposals on TVM compression was issued [ISO21c]. Due to the existence of technologies for encoding geometric information through video already established during the standardization of MPEG-VPCC [ISO19] (video-based point cloud codec, see Section 5.4.1) and MIV [ISO21b] (multi-view video + depth codec), the potential authors were strongly encouraged to incorporate the V3C (Visual Volumetric Video-based Coding) standard [ISO21a], which was created by separating the common generic parts of those two standards. Five companies contributed to this call for proposals: InterDigital, Nokia, Tencent, Sony and Apple [Cho+22b].

In terms of geometry, the approach proposed by InterDigital [Mar+22] is intra-only. It simplifies the frames using quadric error metric [GH97] and encodes them using Google Draco [Gal+18]. The temporal coherence is exploited only in texture data. Each mesh is segmented into patches which are parameterized using *Boundary First Flattening* [SC17] to obtain local UV coordinates. All the patches are then organised into a regular grid in a global UV coordinate system. Intra- and inter-frame reorganization of patches follows to minimize sharp transitions between neighbouring tiles and to place patches of similar RGB values in time at the same tiles.

The response from Nokia [Alf+22] encodes geometry quite similarly to the MPEG-VPCC [ISO19] method. Their patch-based approach performs temporal patch alignment inside a group of pictures by packing patches of similar average positions in 3D to similar areas in the image domain. They also combine the depth and occupancy images into a single image in YCbCr 4:2:0 format, where the depth is contained in the luma channel and occupancy in the chrominance.

The approach proposed by Tencent [Hua+22] uses a *Multi-chart Geometry Image* [San+03] representation. They assume that a temporally coherent parameterization is already known before coding and focus mainly on the means of preserving the watertightness of the input meshes. To this end, they find boundary vertices of all the patches and encode them separately including their UV and XYZ coordinates (in predictive coding) and the information to identify the corresponding vertices at neighbouring patches.

As of the time this thesis was being written (October of 2023), to the best of our knowledge, no document directly describing the pipeline proposed by Sony was released to the public. It was only briefly summarized in an overview paper by Choi et al. [Cho+22a]. We can only assume from this text, that the method is probably very similar to the approach proposed in Sony's recent patent [GZT22].

Although it also incorporates video coding, the method that was proposed by Apple [Mam+22] is much closer to model-based approaches when considering our classification. Their preprocessing of the sequence consists of a simplification followed by a midpoint subdivision projected to the original surface. The simplified mesh (called base mesh) is then encoded using Google Draco [Gal+18]. The displacements between the subdivided mesh and the original surface are encoded using wavelet transform. The wavelet coefficients are encoded as a video. The temporal coherence is exploited by replacing the current base mesh with the base mesh of a reference frame where possible. Then, instead of coding the whole base mesh, only a motion field is required.

Out of the five proposals, the best compression performance was achieved by Apple and InterDigital [Cho+22a]. At the 138th meeting, the proposal by Apple was selected as a basis for the subsequent standardization process [Cho+22b]. The standard is expected to be completed by October 2024 [MPE22].

4.5 Current challenges

Although it is apparent that there exists a redundancy of temporal information in TVMs, for a long period, it was difficult for TVM compression methods to outperform intra-only approaches such as Google Draco [Gal+18] or weighted parallelogram [VB13]. For this reason, the baseline method used to evaluate the performance of MPEG V-DMC proposals was intra-only [ISO21c]. In the last 10 years, the advances in video- and model-based approaches led to a few quite effective methods being proposed, which were able to outperform intra-only methods, however, at the cost of sacrificing the original mesh structure [Mam+22; ZGT23], limitation of the type of input sequences [Dou+14b; Dou+14a] or both [Xia+10; Hou+14a]. For future research, we believe that it is important to focus on minimising such sacrifices and develop a method which can handle general input data while preserving the original structure of the mesh. In the following subsections, we will focus on ways of addressing these challenges individually.

4.5.1 Structure preservation

As already discussed in Section 1.3, preserving the structure means that for each frame, there exists an isomorphic map between the connectivity of the original and the decoded mesh. Whether any of the methods described in this chapter preserves the structure or not and whether it also addresses compression of the connectivity, is summarized in columns Iso. and Conn. of Table 4.1 (the colour indicates connectivity coding efficiency).

The main reason why many approaches tend to reject the original structure (number of vertices and connectivity) is that while the vertex sampling and the connectivity information are linked to the temporally coherent shape, they usually contain no temporal coherence. The model-based approaches usually encode the original meshes only at keyframes (encoded intra-only) which are then gradually deformed to replace the rest of the frames [YKL06; TSM07; MYA08; NYA10; TM12; Mam+22]. The video-based approaches, on the other hand, deduce the connectivity from the decoded data [HKM04; Xia+10; Xia+12; Hou+12; Hou+13a; Hou+13b; Hou+14b; TM13; Hou+14a; Hou+15; Alf+22; Hua+22]. This is because the frames are densely resampled to the image domain, and thus vertices reconstructed from neighbouring pixels can be connected by an edge. A surface extraction from the occupancy network representation [ZGT23] also deduces a connectivity that is different from the original. There is also a second motivation for not preserving the original connectivity: remeshing to obtain a more exploitable structure. This is mainly used in progressive approaches [YKL06; Mam+22], but for example, the proposal by InterDigital for the MPEG V-DMC [Mar+22] also performs simplification as a way of further reducing the data rate.

The approaches based on ME [GSK03; HYA07; YA10] and prediction of data structures [HYA08; Fer+10] can preserve the original set of mesh vertices. However, most of these methods did not directly address the encoding of the connectivity information [HYA07; HYA08; Fer+10].

The earliest method for TVM compression proposed by Gupta et al. [GSK03] is the only one that assumes temporal coherence of connectivity between frames. It thus encodes the connectivity using simple update operations. This, however, works only on synthetic sequences and is highly impractical for real-world data (e.g., 3D-scanned human actors).

Yamasaki et al. [YA10] were the first to acknowledge that the connectivity information occupies a considerable amount of the compressed data stream and should be considered if one proposes an efficient TVM compression method that preserves the mesh structure. In static mesh compression approaches, the mesh connectivity is usually encoded first and then used to drive the geometry coding. Unfortunately, to the best of our knowledge, no one was ever able to propose a connectivity-driven geometry coding method for TVMs that efficiently exploits temporal coherence. Faramarzi et al. [FJB20] attempted to do this but failed in comparison to intra-only approaches. Alternatively, the vertex positions can be encoded more efficiently separately, which leads to their reordering. Conventional connectivity coding methods (e.g., Edgebreaker [Ros99] or TFAN [MZP09]), however, also reorder vertices. Some approaches used permutation maps to relate these two reordered sets [YA10; FJB20], but this requires additional $\sum_{i=0}^{n-1} \log_2 (m_i!)$ bits of data, where n is the number of frames and m_i is the number of vertices of the frame $\mathcal{M}_i$, to be transmitted.

Graziosi [Gra21] proposed to store the connectivity as a part of a 2D mesh. The mesh shares the connectivity with the original frame, but the position of each vertex stores the pixel coordinate in the image domain to which the original vertex was mapped. This 2D mesh is then encoded using an intra-only method (e.g., Google Draco). In Sony's patent [GZT22], the connectivity is rasterized into an image and then reconstructed by the decoder using segmentation. These two approaches are, however, still quite impractical.

Connectivity coding for known geometry

From all the existing structure-preserving methods, the most effective way of storing the connectivity of a TVM was proposed by Doumanoglou et al. [Dou+14b]. They encode the geometry first and then use this information to predict the connectivity based on the fact that if two vertices are close enough, they are very likely connected by an edge. Their method uses a modified TFAN [MZP09] algorithm, which instead of the conventional TFAN symbols encodes indices to the list of k nearest neighbours.

Although not used in TVM compression, there are quite a few other connectivity coding methods given a fixed geometry. The field, in which the motivation for this type of connectivity coding first arose, is progressive compression. Gandoin et al. [GD02] have pointed out that for non-manifold meshes, it is better to encode the geometry independently before the connectivity. Their method is based on kd-tree decomposition. Subdividing a tree cell is equivalent to splitting a vertex into two, with a certain connectivity update, which is encoded using the geometry as a prediction. A similar connectivity update was also used by Peng et al. [PK05].

GEncode, a single-rate general mesh (e.g., surface or volume mesh) compression scheme proposed by Lewiner et al. [Lew+06] encodes connectivity during a traversal through the mesh. To signal which vertex should be connected to the currently coded cell (e.g., a face in a surface mesh) the method encodes an index in the list of candidate vertices given a certain geometric function and a selected range of its values. The method works for meshes of arbitrary topology and dimension embedded in spaces of an arbitrary dimension.

Marais et al. [MGS07] claimed that due to advancements in point cloud compression, it is possible to encode the geometry separately as a point cloud and then exploit the global vertex position information to improve the performance of the connectivity encoding. The triangles are encoded in a slightly modified fixed traversal similar to the one used in the Edgebreaker algorithm [Ros99]. A position of a tip vertex of the currently coded triangle is predicted using the parallelogram or the midpoint scheme, which is then rotated and scaled to better match the vertices ahead of the current gate, and the rank of the correct tip vertex in the list of nearest neighbours around the prediction is encoded.

For highly regular data, Chaine et al. [CGR09] proposed a connectivity coding approach based on surface reconstruction. Both the encoder and the decoder perform an iterative surface reconstruction algorithm, and the encoder signals the differences between the actual and the reconstructed surface. The method applies only to triangles that are part of a Delaunay tetrahedralization of points. The rest of the triangles must be encoded less efficiently.

4.5.2 **Versatility**

By dropping versatility (ability to process general input), one can make more assumptions about the character of the encoded data. It is also important to distinguish between whether the method fails to process more general data or is merely inefficient when compressing such data. Both of these properties for each method are summarized in columns *Versatility* and *Designed for* of Table 4.1.

The most limiting assumption on input data was made by Gupta et al. [GSK03]. They expected synthetic data on input, which allowed them to efficiently encode the connectivity, as was described in the previous section. The method does not fail for general input sequences, but its performance is very poor on such data.

All the methods based on expression invariant parameterization [Xia+10; Xia+12; Hou+12; Hou+13a; Hou+13b; Hou+14b] can handle only facial data and only in a very specific form, which contains one outer boundary and three inner boundaries located at both the eyes and the mouth. This is due to the method being incorporated directly into the data acquisition pipeline.

The polycube-parameterization-based approach [Hou+14a; Hou+15] is limited to sequences of constant topology, given by the fact that it uses a static polycube provided by the user, which reflects this topology. Although there are algorithms for obtaining the polycube parameterization automatically even for surfaces of general topology [Yu+14], to the best of our knowledge, there is no approach that also achieves temporal coherence. Additionally, the authors only theorize that the method works on TVMs and for experiments, they used dynamic meshes.

Since currently, the TVMs are primarily used for representing human actors, there have been quite a few methods (mainly model-based) that are optimized to work on human data. Assuming the TVM captures a performance of a human actor, the method can expect a specific ground truth underlying structure (head, arms and legs connected to the torso) and type of movement (articulated around joints), even though noise introduced into data, for example, during scanning, might distort the actual topology and motion. These approaches can be divided into two classes: those using tracked skeletons [MYA08; NYA10; Dou+14a; Dou+14b] and those using Reeb graphs [TSM07; TM12; TM13]. While not stated in the original papers, it is theoretically possible to adapt the skeleton-based approaches to sequences representing articulated surfaces of different constant ground-truth topologies (*Const. GT Top.* in Table 4.1) (e.g., animals, articulated robots), but this ground-truth structure must be known before encoding. The Reeb-graph-based approaches can handle general sequences of varying topology, although not efficiently, except the original approach of Tung et al. [TSM07], which uses heuristic rules to detect spurious self-contact in the topology of the graph, as these rules are designed only for human sequences. Theoretically, a different set of rules could be designed for a different surface of constant ground-truth topology, but this adaptation is much more difficult than the adaptation of skeleton-based approaches for such data.

Constant ground-truth topology (*Const. GT Top.*) is also assumed by Yang et al. [YKL06]. Although their method is agnostic of the character of the underlying represented data, it is, to some extent, more limited than the approaches discussed in the previous paragraph. Since it replaces the sequence by the first frame propagated in time, it is crucial for the performance of the method that the first frame reflects the underlying topology. For this reason, it works best on data of constant actual topology of the frames.

The rest of ME-based [HYA07; YA10], prediction of data structure-based [HYA08] and video-based [HKM04; FJB20; Gra21; GZT22; Mar+22; Alf+22; Hua+22] methods put no assumptions on the shape and structure of the represented data other than the assumption of the presence of temporal coherence.

Versatile temporal models

While most efficient in terms of geometry compression, the model-based approaches are notorious for limiting the type of data they can handle. This is, however, not necessarily an issue of the used models, but usually of the approaches themselves. There are model-based approaches that can handle general input [TM12; Mam+22; ZGT23], unfortunately, none of the models used in these methods is directly suitable for structure-preserving compression.

It is certainly possible to extend the versatility of a constrained model. One good example is the tracked template used by Yang et al. [YKL06]. Bojsen-Hansen et al. [BLW12] presented a tracking pipeline for surfaces of evolving topology, which can update the tracked template if a change in topology is detected and record mapping for the updated parts to preserve inter-frame one-to-one correspondences. The result is a Time-varying mesh with temporally coherent connectivity. Unfortunately, for structure-preserving compression, this model still has a large footprint, which makes it impractical for structure-preserving TVM compression, since it must be encoded alongside the data.

Although, to the best of our knowledge, there is only a single method for TVM compression based on deep learning techniques [ZGT23], in the future, this approach will likely become increasingly popular. There are many possible places in the compression pipeline, where deep learning could be utilized, and it already is used in different compression domains, for example, for context modelling of entropy coder in point cloud compression [Bis+20], but the most probable part is once again the temporal model. We will not go into depth on this topic, but examples of deep learning methods that could be potentially incorporated as a versatile temporal model are the deformation field of *OccupancyFlow* [Nie+19] and *Neural Deformation Graphs* proposed by Božič et al. [Bož+21], to name a few.

4.6 Summary

The simplest methods for TVM compression are based on ME or the prediction of spatial structures. These approaches are simple, can be simply adjusted to preserve the original mesh connectivity and can handle general input. Their main drawback is their inefficiency.

Current model-based methods are the most efficient in terms of compression of TVM geometry. Both state-of-the-art methods for TVM compression can be classified as model-based: the structure-preserving method of Doumanoglou et al. [Dou+14b] and the future MPEG V-DMC method based on the proposal by Apple [Mam+22].

For textured TVMs, it is best to use a method which uses video compression. These also work well for densely sampled TVMs with colours as vertex attributes. They can mostly handle general input, but they are difficult to modify to preserve the original connectivity.

We believe the research area of TVM compression still has a lot of gaps, mainly the structural preservation and versatility, which were presented in the previous section. Other than that, there is also a motivation for real-time compression [Ort+16], where these two challenges can be potentially ignored, but current methods are either too complex or inefficient to be used instead of intra-only approaches. For our research, we focus on offline compression, with the potential of future adjustments for real-time scenario once satisfactory compression rates are achieved.

The field of perceptual metrics of distortion for TVMs was also mostly neglected so far. It was only considered by MPEG when evaluating the proposals for MPEG V-DMC, who, however, based their evaluation on perceptual metrics for videos. To the best of our knowledge, there is no perceptual metric for TVMs that works directly with its geometry.

Dynamic Point Cloud Compression 5

The TVM compression is much more closely related to the compression of DPCs than the compression of DMs. This is because existing methods use similar techniques for data reduction since there are also no explicit correspondences between the frames. That is why we chose the same categorization into sections, albeit in a different order, which reflects better the timeline of development in this field. The content of this chapter is summarized in Table 5.1.

Table 5.1: Overview of existing methods for DPC compression. Methods are in the order in which they appear in the chapter. Highlighted methods are considered state-of-the-art. *Ref.* column contains the most relevant reference for the given method. Columns *Versatility* and *Designed for* show the type of input data the method handles. Columns *Iso.* show whether the method preservers structure (Isomorphism). Symbol meaning: ✓ = Yes, ✗ = No, ❋ = Optional.

Method	Ref.	Category	Subcategory	Versatility	Designed for	Iso.
Projection around skeleton	[LKB10]	Model/Video	Skeleton/Projection	Const. GT Top.	Human	✗
Fitted triangle mesh	[ACO16]	Model	Tracked surface	Const. GT Top.	Const. Top.	✗
Face detection	[Yan+18b]	Model	Face detection	Human	Upper body	✗
Octree XOR	[Kam+12]	Struct.	Octree	General	General	✓
Sorted occupancy values	[GQ17]	Struct.	Octree	General	General	✓
Distance-based context	[Que+18]	Struct.	Octree	General	General	✓
Super-resolution for octree	[Gar+19]	Struct.	Octree	General	General	✓
MuSCLE	[Bis+20]	Struct.	Octree	General	General	✓
Cellular automata transform	[MPL20]	Struct.	Octree	General	General	✓
Silhouette4D	[RPM21]	Struct.	Silhouette tree	General	General	✓
Super-resolution + CNN context	[KT22]	Struct.	Octree	General	General	✓
Curve fitting	[Dar+12]	ME	—	Struct. light sc.	Struct. light sc.	✓
Feature change detection	[CK12]	ME	—	General	General	✗
Graph-based ME	[TCF16]	ME/Struct.	—	General	General	✓
ICP macroblocks	[MBC16]	ME	—	General	General	✗
k-d tree partitioned ME	[Kat+17]	ME	—	General	General	✓
Correspondence-driven MC	[QC17]	ME	—	General	General	✗
TSS extension to MPEG G-PCC	[San+21]	ME	—	General	General	✗
Plane fitted LiDAR data	[FLZ20]	Video	Range image	LiDAR	LiDAR	✗
MPEG V-PCC	[ISO19]	Video	Projection	General	General	❋
Projection on rotating planes	[Sch+19]	Video	Projection	General	General	❋
Global patch packing	[Liu+19]	Video	Projection	General	General	❋
Patch resegmentation	[SL22]	Video	Projection	General	General	❋
3D ME hint for HEVC	[Li+20]	Video	Projection	General	General	❋
MPEG V-PCC + ME	[Kim+20]	Video/ME	Projection	General	General	✗
Body part segmentation	[Cao+20]	Video/Model	Projection	Human	Human	✗
View-PCC	[Zhu+21]	Video	Projection	General	General	❋

5.1 Temporal-model-based methods

As far as we know, the earliest DPC compression method that incorporates a temporal model as a primary paradigm was proposed by Lien et al. [LKB10]. They combine compression using a tracked skeleton with the video-based approach to achieve interactive performance for a tele-immersion system. They fit the first frame of the sequence to each subsequent frame using an articulated version of the ICP algorithm [BM92] driven by the skeleton. Points around the links of the skeleton of both the encoded and the first frame are projected into a cylindrical grid and the difference between the two images is encoded.

Anis et al. [ACO16] replaced the original DPC with a semi-regular TVM. For each selected keyframe, they construct a coarse Poisson surface reconstruction [KH13] which is then iteratively subdivided and fitted to the point cloud frame. This iterative process is also applied to non-keyframes, albeit with the base mesh of the last keyframe on the input. This way, each subsequence is represented by a dynamic mesh and only the motion vectors must be encoded for non-keyframes to represent the vertex trajectories. The hierarchical structure of the subdivided mesh allows encoding the values using graph wavelet transform [NO13].

For the purposes of tele-immersion, Yang et al. [Yan+18b] designed a compression method for human upper bodies. Assuming a minimal movement of the head of the captured person, a complete point cloud is encoded only for once every four frames (I-frame). For the rest of the frames (P-frame), the method detects a human face; only this part is transmitted and placed over the geometry of the I-frame.

5.2 Prediction of data structures

Since DPCs are usually stored in the voxelized form in octrees, many methods incorporate the prediction of data structures.

The earliest method using prediction of data structures was proposed by Kammerl et al. [Kam+12]. It is based on a binary serialization of an octree (see Figure 5.1). When processing a certain non-empty node of the current frame, only the XOR difference between its value and the value of the corresponding node in the previous frame is encoded. This approach is also one of the few that works even for data that was not voxelized. While it stores the data in an octree, it allows optional encoding of relative positions of points inside a cell.

Garcia et al. [GQ17] proposed a method which combines multiple previous octrees into one for prediction. The nodes of the reference octree are sorted in ascending order of the bytes representing their occupancy, which results in a permutation map of the nodes. This permutation is then applied to the nodes of the coded frame. Since the values of both the reference and the coded frame are expected to be similar, the values of the coded nodes should be almost in an ascending order, which can be exploited by context-adaptive coding.

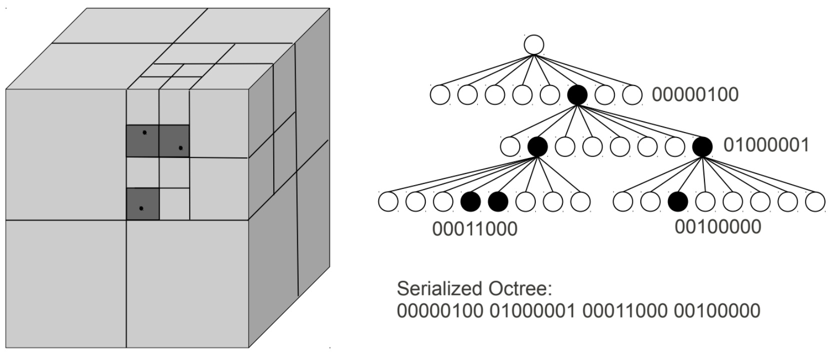
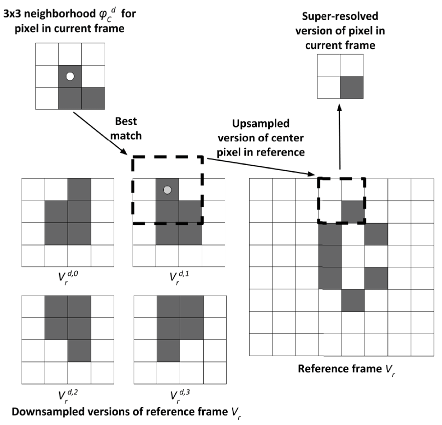


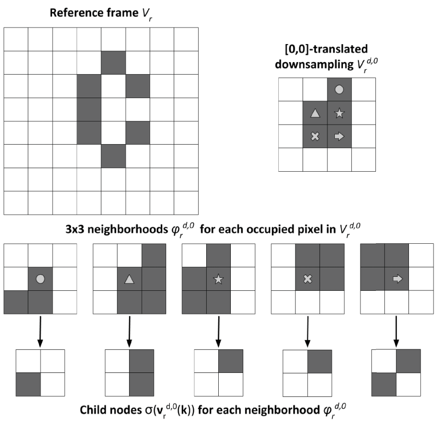
Figure 5.1: Serialization of an octree. A byte value is assigned to each non-leaf node based on the occupancy of its child nodes. Source: [Kam+12]

Queiroz et al. [Que+18] modelled the occupancy probabilities given a distance to the closest occupied voxel in a reference frame, which is obtained either as the previous frame or a prediction of the current frame level-of-detail given the previous frame and all the already encoded levels of detail of the current frame.

Another method proposed by Garcia et al. [Gar+19] uses context switching based on the predicted occupancy. The prediction is constructed hierarchically during encoding based on the previous frame and the previously encoded level of detail of the current frame by *super resolution* (see Figure 5.2).


3x3 neighborhood φ_C^d for pixel in current frame
Super-resolved version of pixel in current frame
Best match
Upsampled version of center pixel in reference
$V_r^{d,0}$
$V_r^{d,1}$
$V_r^{d,2}$
$V_r^{d,3}$
Reference frame V_r
Downsampled versions of reference frame V_r

(a) By Example


Reference frame V_r
[0,0]-translated downsampling $V_r^{d,0}$
3x3 neighborhoods $\varphi_r^{d,0}$ for each occupied pixel in $V_r^{d,0}$
Child nodes $\sigma(\mathbf{v}_r^{d,0}(\mathbf{k}))$ for each neighborhood $\varphi_r^{d,0}$

(b) By Neighborhood Inheritance

Figure 5.2: Super-resolution modes for constructing the reference octree. Source: [Gar+19]

Biswas et al. [Bis+20] trained a neural network to model the context-based probabilities of octree occupancy symbols. The considered context consists of the occupancy of the ancestor nodes and the occupancy of the previous frame.

Milani et al. [MPL20] used cellular automata transform on octree occupancy symbols to assign the most frequent symbols larger values. The resulting serialization contains large runs of 1-value bits, which is preferable when encoding such data using arithmetic coding. Since the decoder does not know the symbol frequencies, they can be deduced from the previous frame.

Peixoto et al. [PMR20] constructed a binary tree over a regular occupancy grid, where each tree node contains a slice of the geometry along a selected principal axis. The geometry in the slice is represented as a silhouette image (projection along the selected axis). The structure of the tree and the encoding order allow some of the values to be omitted from the coding stream. When estimating current symbol probability, the rest of the values are encoded with context modelling, which considers values at specified positions (e.g., neighbouring cells, the corresponding cell in the previous frame). In their subsequent work [RPM21], the authors improved the method's compression rate using context selection, in which a specified number of positions is selected from 24 predefined during preprocessing so that the entropy is minimized.

Kaya et al. [KT22] used a convolutional neural network (CNN) to encode occupancy values of an octree. A certain level of detail of the octree is encoded in layers along a specified axis, each layer in blocks of size 2×2. Coding context is obtained by upsampling the previously encoded level of detail from the current frame and processed by the CNN to model the symbol probabilities. The method exploits temporal coherence only to some extent: the CNN is pre-trained on a subset of frames and encoded alongside the data. The neural network is then adapted to each frame during encoding.

5.3 Motion estimation

There are only a few methods that are based solely on ME. For this reason, we will list here all that use ME as the main technique for data reduction.

Daribo et al. [Dar+12] exploited that in the grid-pattern-based 3D scanned frame, subsequent points usually lie on a spatial curve. They proposed multiple spatio-temporal predictors to encode these curves, which also fit curves between the frames.

Champawat et al. [CK12] perform corresponding octree node comparison using specified features (number of points, mean position, eigenvectors and eigenvalues of covariance matrix of positions and colours). Under specific conditions, the node from the previous frame can be reused as is or translated or rotated to replace the current node. Otherwise, the current node is encoded the standard way (e.g. XOR approach [Kam+12]).

The method proposed by Thanou et al. [TCF16] is an ME extension of the XOR approach of Kammerl et al. [Kam+12]. They construct k-NN graphs of leaf nodes of the octree constructed over the geometry of the previous and the current frame. All the points are assigned feature vectors, which are then used to find correspondences between the two frames. A few of the points are selected for motion estimation and their motion vectors are propagated through the graph. Finally, the previous frame is motion-compensated and the geometry is encoded as in the XOR approach [Kam+12].

Mekuria et al. [MBC16] designed a lossy progressive compression algorithm with near real-time performance based on ICP registration between macroblocks (nodes of an octree K levels above the final level of detail, where K is a parameter). When all specific criteria are met (e.g., a similar number of points), the coded macroblock is replaced by the rigidly transformed geometry of the corresponding macroblock in a reference frame.

Kathariya et al. [Kat+17] proposed a compression method based on motion compensation between blocks represented by cells of a k-d tree. It divides DPC into subsequences. For each frame, the geometry is partitioned into blocks using splitting planes of a k-d tree constructed over the first frame of its subsequence. The frame is then encoded block-by-block using the previous frame. Points are inserted to (resp. removed from) each block of the reference frame so that it contains the same number of points as the corresponding coded block. The method then finds one-to-one correspondences between the points of the two blocks. These are encoded alongside residues.

Queiroz et al. [QC17] re-used the correspondences found during 3D surface reconstruction [Dou+15] to compute the motion vectors. The frames are split into cubic blocks of specified size. Currently coded block can be replaced by the motion-compensated corresponding previous block if it provides a good approximation. They also employ smoothing and morphological operations to close gaps caused by the lossy compression.

Santos et al. [San+21] extended the intra-only algorithm of the MPEG G-PCC standard [ISO20] to consider temporal coherence by incorporating the Three-Step Search (TSS) algorithm [Kog81] commonly used in video compression. Matching of the macroblocks and their final alignment for motion compensation was done using ICP [BM92].

5.4 Video-based methods

Video-based dynamic point cloud compression methods are mostly derived from MPEG V-PCC [ISO19] which uses projection to transform geometry into the image domain. To the best of our knowledge, only two methods were proposed independently of MPEG V-PCC, one of which was already described in Section 5.1. The second independent method was proposed by Feng et al. [FLZ20]. It converts LiDAR point cloud sequences into range images. Plane fitting is used to represent pixel regions. The method exploits the accompanying sensory data to fit a coded frame onto a keyframe. The method attempts to represent geometry by planes parallel to those of the keyframe and if successful, only the offsets of the planes are encoded.

5.4.1 MPEG V-PCC and related methods

The growing need for an efficient compression method for point cloud data was also recognized by the MPEG. This led to the call for proposals in early 2017 [ISO17]. The work of Mekuria et al. [MBC16] was selected as a baseline for comparison with all the proposed methods. As a result, 13 proposed solutions were collected from various industry and research contributors and three different test model cases were identified: TMC1 for static data (e.g., cultural heritage), TMC2 for dynamic data, and TMC3 for dynamically acquired data (e.g., LiDAR). Eventually, due to the similarities in the approaches, TMC1 and TMC3 were merged to form TMC13, which led to the development of geometry-based point cloud compression (MPEG G-PCC), the ISO/IEC 23090-9 standard [ISO20]. The method evolved from TMC2 is called video-based point cloud compression (MPEG V-PCC), the ISO/IEC 23090-5 standard [ISO19]. Although there are plans to incorporate temporal prediction into the G-PCC standard or its successor [LF19], as of now (October 2023) it is still intra-only. For this reason, only the V-PCC standard and the work directly related to it, which considers inter-prediction, will be described in this section.

As mentioned in Section 4.4, MPEG V-PCC [ISO19] is based on the orthogonal projection of geometry into planar patches. First, the point normals are estimated. Each point is then assigned to one of the predefined planes around the frame (e.g., planes that form the axis-aligned bounding box) with the closest normal. The points are then clustered by grouping neighbouring points with similar orientations of normals to create patches. Patches are then projected onto the corresponding plane and assigned positions in the parametric domain by *patch packing*. In the first frame in a group of frames, this process places the patches in order induced by the patch index so that it is guaranteed that there is no overlap. For the rest of the frames, patches are matched using intersection over union (IOU) and placed in similar positions to achieve temporal coherence. Three different images are generated: depth (distance of a point to the projection plane), occupancy (information on whether the pixel contains geometry information, which is required due to the complex shape of the patch) and attribute (e.g., colour). Note that two points in the same patch can be projected onto the same position. In this case, V-PCC allows encoding multiple images to allow lossless coding. Occupancy information is usually encoded with reduced resolution, while depth and attribute images are padded with smooth transitions between values. The method allows additional smoothing in post-processing to reduce the gaps between patches caused by quantization.

The method proposed by Schwarz et al. [Sch+19] was one of the contributions to the initial call for proposals for point cloud compression. It was similar to the proposal which eventually evolved into the V-PCC standard. Instead of segmenting the point cloud into patches, the geometry is orthogonally projected as is onto a series of planes rotating around the bounding box. Instead of using the occupancy map, this information is signalled through the depth image. The temporal coherence is exploited implicitly by the fact that points of two subsequent frames projected on the same pixels are usually close to each other spatially.

To improve the temporal placement of patches, Liu et al. [Liu+19] used global patch packing. This method performs the patch packing on all frames inside a group of frames by tracking corresponding patches identified by IOU. While this achieves a larger overlap of corresponding patches, the resulting video can be less compact due to slight differences in the patch shapes between the frames. This was already addressed by Shi et al. [SL22], who proposed to resegment the patches to remove uncorrelated parts.

Li et al. [Li+20] focused on improving the V-PCC coding performance when assuming the HEVC [Sul+12] algorithm is used for video compression. They used 3D motion estimation to hint to the video coder the 2D motion vectors between the pixel positions of corresponding points.

This approach was extended by Kim et al. [Kim+20] to work regardless of the video codec utilized. They encode keyframes using the unmodified V-PCC. For the rest of the frames, only the 3D motion vectors and attribute differences are encoded. Keyframes are selected as the frames, where the motion-compensated approximation achieves an error above a specified threshold.

Cao et al. [Cao+20] proposed a model-based human DPC compression method which utilizes MPEG V-PCC. The method uses deep learning to segment the point cloud into body parts. For each body part, a sequence of affine transforms is found representing its movement. The method selects a few keyframes, which are encoded using V-PCC, and then used to replace the rest of the frames when deformed using the found affine transforms.

View-PCC algorithm proposed by Zhu et al. [Zhu+21] uses global orthogonal projection similarly to Schwarz et al. [Sch+19] but only to four planes (front, back, left and right). Patches are used only to represent points that were occluded in the global projection. The method attempts to find corresponding local patches using the IOU of patch 3D bounding boxes and then places corresponding patches at similar positions in the image domain. If a camera viewpoint information is provided, the method can further reduce the data rate by encoding only the visible parts.

5.5 Current challenges

As of 2023, DPC compression is already considered a well-studied research area. Nevertheless, it seems that new ideas are still being introduced. Currently, in terms of inter-frame geometry compression, the researchers are still focused on improving performance in the approaches based on the prediction of data structures, or in video-based compression.

In the case of the prediction of data structures, the focus is on efficient ways of modelling the context for occupancy symbols [GQ17; Que+18; Gar+19; Bis+20; PMR20; RPM21; KT22]. The hierarchical property of these approaches also recently gave motivation for efficient super-resolution, in which the goal is to predict the structure of the next level of detail given the current level and previously encountered information [Gar+19; KT22].

Current video-based approaches are usually based on or inspired by the MPEG V-PCC standard [ISO19]. The temporally coherent placement of projected patches remains the main challenge [Liu+19; SL22], although this can be addressed by improving motion estimation in video coding [Li+20].

5.6 DPC compression in the context of TVMs

Despite the similarities between the problems of DPC and TVM compression, these two research areas developed in quite different ways. One might argue that DPC compression is already quite ahead in terms of compression efficiency. In this section, we will point out some of the most interesting differences between the DPC and TVM compression approaches. We will also discuss, which ideas from DPC compression might be relevant for TVMs.

5.6.1 Differences

The most notable difference between the two areas is that it is much more common to preserve the original structure of data in DPC compression. The reason is that it is much simpler to do so since there is no connectivity to be encoded and the only thing a method must preserve is the original number of points. Surprisingly, the structure is mostly discarded by ME-based approaches [CK12; MBC16; QC17; San+21], which in TVM compression is one of the categories where methods preserve this information [GSK03; HYA07; YA10].

The most popular approaches in TVM compression are based on temporal models or video coding. While video-based methods are also popular in DPC compression, the most prominent approaches are based on the prediction of spatial data structures. As already mentioned in Section 5.2, this is mainly due to the voxelized data on input. While in TVM compression, this approach was used in only two algorithms, which were proposed quite early on and only used spatial grids, in DPC compression, it is used to this date and the current approaches are quite sophisticated.

On the other hand, there are not as many model-based methods, although the models used are fairly similar to the ones used in TVM compression. We do not know why these approaches are not as popular as in TVM compression other than the fact that the prediction of data structures is a much more intuitive approach in this scenario. This, however, means that DPC compression methods are generally more versatile, even though dynamic point clouds are used to represent the same scenes as the ones represented by TVMs (mainly the human performances). The only exception in versatility is LiDAR point cloud compression, in which if data obtained during one rotation of the scanner is represented as a single frame, a specialized method can be used [FLZ20]. Such a method can exploit the sensory data of the scanner to align the frames and represent the data as range images since due to the way the geometry was obtained, no line through the scanning position and any point contains any other point.

Although video-based approaches are popular in both areas, before MPEG standardization efforts, these were not as common in DPC compression as they were in TVM compression. This increase in popularity can be attributed to the increase in the sampling density of the data. On coarsely sampled surfaces, which were more common in the past, video-based approaches are inefficient in terms of geometry compression. The reason why there were more video-based methods for TVM compression before MPEG standardization is that their efficiency lay in texture compression since this information is dense. However, texture also assigns attributes to points on faces between the vertices. In the case of point clouds, even though there are also methods for point cloud parameterization [SSC19], attributes are usually only assigned to sampled points, which is equivalent to vertex attributes on meshes. As a consequence, all the video-based DPC compression methods use orthogonal projection to map points into the image domain.

In DPC compression, it is also more common to incorporate deep learning, mostly used for context modelling [Bis+20; KT22]. In the future, we believe that this slight difference will disappear and that more and more approaches in both areas will incorporate neural networks as a part of their pipeline.

5.6.2 Implications

The fact that the development in DPC compression is useful for TVMs has been already shown multiple times with MPEG V-PCC. Faramarzi et al. [FJB20] treated mesh geometry and attributes as point cloud data and compressed it using the V-PCC. Instead of incorporating the approach into their pipeline, the proposals for MPEG V-DMC adjusted it to work with mesh data [Mam+22; Mar+22; Alf+22; Hua+22; GZT22]. We argue that MPEG V-PCC is not the only DPC approach that can serve as an inspiration for TVM compression.

In terms of mesh geometry compression, more interesting approaches to examine are the ones using the prediction of spatial structures due to their performance on coarsely sampled surfaces. One could once again combine a certain point cloud codec (e.g. the Silhouette4D approach or its variants [PMR20; RPM21]) with connectivity compression, although the connectivity coding must account for vertex reordering. Note that as was shown by Thanou et al. [TCF16], it is also possible to combine these approaches with other paradigms, for example, motion estimation or temporal models.

Other than incorporating such methods as a whole, we can also find inspiration in the techniques they use. One such example is working with the context of encoded symbols, such as reordering values to an order exploitable by context adaptive coding [GQ17; MPL20], context modelling [Que+18; Gar+19; Bis+20; PMR20; RPM21; KT22], context switching [Gar+19; PMR20] or context selection [RPM21], which can be used when encoding any value, not necessarily the occupancy of a voxel. This could be a way of exploiting any remaining coherence in the already temporally predicted data.

Although it is also a consequence of the way the input data is stored, DPC compression methods are more frequently hierarchical, which means that the geometry is processed from coarse to fine levels of detail. This way, one can also use the coarse information combined with the temporal information to predict the current level of detail. In Section 5.5, we have already mentioned that currently, the most prominent way to achieve this in DPC compression is super-resolution [Gar+19; KT22]. Some TVM compression methods already use coarse-to-fine processing of the geometry [YKL06; HYA08; Mam+22], but only the method proposed by Han et al. [HYA08] does not require remeshing.

5.7 Summary

Although the fields of compression of TVMs and DPCs have similar objectives and settings, and the approaches use similar paradigms, their development was considerably different. From the two, the development of DPC compression can be considered a bit ahead, mainly due to some challenges (e.g., connectivity coding) being not of concern when compared to TVM compression. Regardless of the field possibly being less challenging, there is lots of inspiration to be drawn for possible further development of the field of TVM compression.

Current state-of-the-art methods preserving the structure use either video compression (for dense point clouds) [ISO19] or prediction of spatial data structures (for coarser data) [RPM21; KT22] and can achieve lossless compression of voxelized data with rates of around 0.8 bits per occupied voxel [KT22]. Further reduction is achieved by approaches that do not preserve the structure, which are mainly ME-based [San+21], but there are also approaches based on MPEG V-PCC [Kim+20; Cao+20; Zhu+21] that discard part of the data.

Even though the area of DPC compression is well studied, further improvements for both lines of the research (video-based, prediction of spatial data structures) are expected in the future. Also, deep learning is being incorporated more and more and we believe this trend will continue.

Part III
Our contribution

6 Error Propagation Control in Laplacian Mesh Compression

This chapter describes our first contribution relevant to the topic of this thesis. In this work, we have focused on improving the performance of *high-pass coding* [SCT03] under mechanistic distortion metrics. It was our only contribution to the field of DM compression. The paper was presented at *SGP 2018* and published in the *Computer graphics forum* journal [VD18]. An executable binary of our reference implementation is available at `http://meshcompression.org/sgp2018`.

6.1 Background

High-pass coding, often called Laplacian mesh compression, is a compression strategy based on encoding differential coordinates obtained by applying a discrete Laplace operator to the input geometry [SCT03]. Assuming the geometry is represented by a matrix $\mathbf{X} \in \mathbb{R}^{|V|\times 3}$, the differential coordinate matrix $\mathbf{D} \in \mathbb{R}^{|V|\times 3}$ is obtained as follows:

$$\mathbf{D} = \mathbf{M}\mathbf{X},$$

where $\mathbf{M} \in \mathbb{R}^{|V|\times|V|}$ is a Laplacian matrix corresponding to the chosen Laplace operator discretization. Naïvely, one could try to quantize and encode the values in $\mathbf{D}$ and reconstruct the geometry by solving the Poisson equation:

$$\mathbf{M}\hat{\mathbf{X}} = \hat{\mathbf{D}},$$

where $\hat{\mathbf{D}}$ is the matrix of differential coordinates distorted by quantization. However, $\mathbf{M}$ is a singular matrix (differential coordinates are translation invariant) and the problem cannot be solved trivially. To this end, the original authors proposed to select one or more vertices per connected component of the mesh as *anchor vertices* [SCT03]. For each anchor vertex, a row with a single nonzero unit element at a position corresponding to the index of such vertex is appended to $\mathbf{M}$, and a rectangular Laplacian matrix $\mathbf{M}_+$ is obtained. Using the rectangular Laplacian matrix, an extended differential coordinate matrix $\mathbf{D}_+$ is obtained. The values that are added w.r.t. $\mathbf{D}$ are the positions of anchor vertices. With the linear system extended by anchor data, $\bar{\mathbf{X}}$ can be reconstructed using the least squares method:

$$\mathbf{M}_+^\top \mathbf{M}_+ \bar{\mathbf{X}} = \mathbf{M}_+^\top \bar{\mathbf{D}}_+.$$

Due to the properties of the normal matrix $\mathbf{M}_+^\top \mathbf{M}_+$ (symmetric, sparse, PSD), the problem can be solved using Cholesky factorization. Since the geometry is the information we want to encode and is thus not available to the decoder, only the so-called *combinatoric* discretizations of the Laplace operator, which can be constructed only with the knowledge of mesh connectivity (Kirchhoff or Tutte discretizations), can be used in High-pass coding.

The Laplacian mesh compression was also adapted to dynamic meshes [VP11]. The process is more or less the same, the only difference is that the input matrix $\mathbf{X}$ now has different dimensions and stores vertex trajectories in an uncompressed form or already reduced to some extent, for example using PCA [Váš+14] or sparse and low-rank matrix approximation [Hou+17]. Additionally, the geometric discretizations of the Laplace operator can be used (Cotan [PP93], MV [Flo03], Intrinsic Delaunay [BS07]), as long as some geometry (e.g., an average mesh) is encoded beforehand [Váš+14].

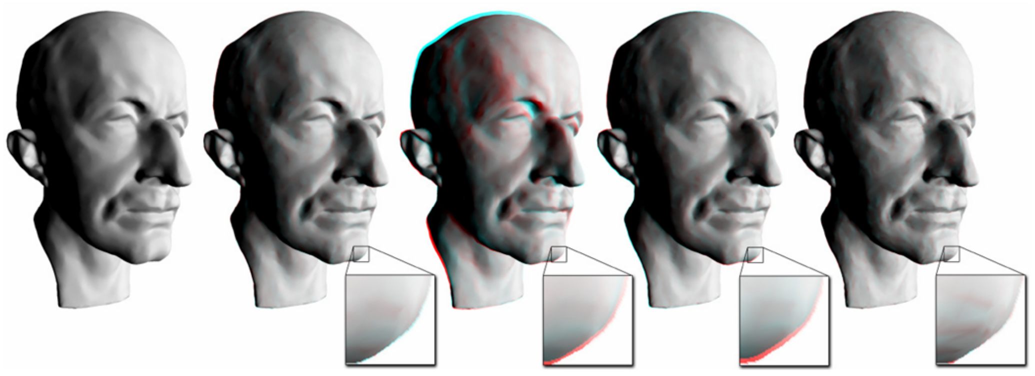


Figure 6.1: A typical result obtained by mesh compression. The original on the far left has been compressed down to a 35kB file (i.e. at 11 bpv) using four methods, from left to right: the proposed approach, high-pass coding, error diffusion and weighted parallelogram encoding. The red channel was preserved from the original, i.e. any red/cyan tint indicates an error. While the weighted parallelogram result appears noticeably rougher than the original, the Laplacian-based methods preserve the original smoothness. The high-pass coding result is, however, visibly deformed, and the chin close-up reveals that not even the error diffusion modification eliminates deformations completely.

Mechanistic approaches (e.g., parallelogram prediction [TG98] or weighted parallelogram [VB13]), which work directly with Euclidean coordinates of vertices, tend to produce high-frequency artifacts at low data rates that are easily visible, especially in flat or generally smooth parts of input surfaces. The high-pass coding approach, on the other hand, tends to produce low-frequency distortion, which is much less likely to be detected visually. However, it does not perform well under mechanistic measures since it is not able to control the upper bound of the absolute coordinate error – while the positions of anchor points are distorted only by the quantization, the distortion of the positions of other vertices is influenced by the distortion of all the vertices lying between the given vertex and the nearby anchor points. This issue was already addressed in [AK15; LV14], however, while both approaches improve the mechanistic properties, neither of them brings the performance on par with parallelogram prediction. A visual comparison of the distortion introduced by these approaches is shown in Figure 6.1.

6.2 Algorithm description

Our approach is based on two proposals. The first is to encode the anchor data separately and use a reduced Laplacian matrix $\mathbf{M}'$ instead of $\mathbf{M}_{+}$. It is obtained by removing all the rows and columns corresponding to anchor vertices from the original matrix and updating $\mathbf{D}$ so that it contains known values. If at least one anchor point is selected per connected component, $\mathbf{M}'$ has full rank and the reconstruction can be done by solving a simple linear system. If the original Laplacian matrix $\mathbf{M}$ was symmetric and positive semi-definite (PSD), which is the case for Kirchhoff and Cotan discretizations, the Cholesky decomposition can be applied to the reduced matrix as well. This approach was already studied in the original work [SCT03]; however, the authors pointed out that it led to higher error accumulation and caused spikes around anchor points. Nevertheless, we show that when it is combined with the following process, these issues are mitigated.

The key novelty of our approach is based on a deeper exploration of the reconstruction process. The decoder first constructs the factor of the Cholesky decomposition $\mathbf{L}$ such that $\mathbf{L}\mathbf{L}^\top = \mathbf{M}'$. Then, it solves $\mathbf{L}\bar{\mathbf{Y}} = \bar{\mathbf{D}}$ by forward substitution. Finally, $\bar{\mathbf{X}}$ is obtained by solving $\mathbf{L}^\top\bar{\mathbf{X}} = \bar{\mathbf{Y}}$. Our focus is on the forward substitution. An i-th row of $\bar{\mathbf{Y}}$ is computed as follows:

$$\hat{\mathbf{y}}_i = \frac{\hat{\mathbf{d}}_i - \sum_{j=1}^{i-1} L_{i,j}\hat{\mathbf{y}}_j}{L_{i,i}}.$$

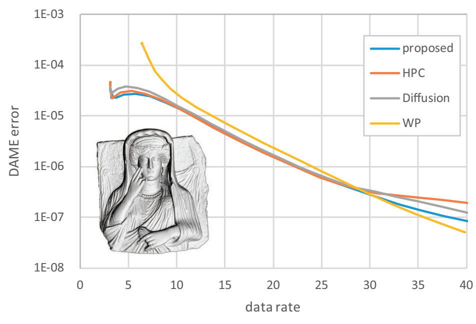
The error accumulation is caused by the influence of the distortion introduced in the previous $i - 1$ rows. However, the encoder knows the ground-truth value of $\mathbf{Y} = \mathbf{L}^\top\mathbf{X}$. We thus propose adjusting the value of $\hat{\mathbf{d}}_i$ on the fly so that $\hat{\mathbf{y}}_i$ is as close as possible to its original value. While some error accumulation is also present in backward substitution, the total effect is substantially reduced by having an error-limited $\bar{\mathbf{Y}}$.

6.3 Experimental results

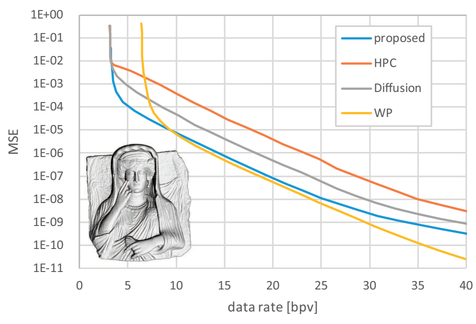
We have tested our approach on both the static and dynamic meshes. To measure the performance, we used both mechanistic and perceptual distortion metrics. The data rates reported in this section also contain connectivity information, which was in all the cases encoded using the Edgebreaker algorithm [Ros99]. In all experiments, we use 0.1% of vertices as anchor points, chosen at random. Other than that, we have also measured the computational performance of our method.

6.3.1 Static meshes

In the case of static meshes, we have compared the results of our approach with the results achieved using weighted parallelogram (WP) [VB13], high-pass coding (HPC) [SCT03] and error diffusion (Diffusion) [AK15]. HPC and Diffusion used Tutte Laplacian, while our approach worked with the Kirchhoff discretization since that is the one that yields the PSD Laplacian matrix. The mechanistic error was measured using MSE, while perceptual performance was evaluated using dihedral-angle mesh error (DAME) [VR12].



(a) DAME



(b) MSE

Figure 6.2: RD curve comparison for *Palmyra* model.

An example comparison of RD curves for both error metrics is shown in Figure 6.2. In terms of MSE, our method achieves results comparable to WP. Moreover, at very low data rates (between 2.5 and 7.5 bpv for the *Palmyra* model), it even achieves the best results from all four approaches. In terms of perceptual performance, it achieves comparable results to all HPC-based approaches.

We have also compared the results of the four approaches with the quantisation set to achieve the data rates of 10 and 15 bpv. This experiment was performed on 7 selected models commonly used for benchmarking in computer graphics (see Table 6.1) and 1723 models from *thingi10k* dataset [ZJ16], which were all the models that had more than 2000 vertices and did not cause a failure of any of the four compression methods. In the case of *thingi10k* dataset, we have measured relative performance to the state-of-the-art method given the error metric (see Table 6.2). Both results show that our method performs comparably to state-of-the-art methods in both mechanistic and perceptual criteria.

Table 6.1: Static mesh compression results in comparison with high-pass coding (HPC) [SCT03], error diffusion [AK15], and weighted parallelogram (WP) [VB13].

		DAME				MSE			
	rate [bpv]	proposed	HPC	Diffusion	WP	proposed	HPC	Diffusion	WP
bunny	10	4.046E-08	3.507E-08	4.368E-08	8.894E-08	9.877E-09	6.819E-07	3.758E-08	8.874E-09
35946	15	1.166E-08	1.063E-08	1.291E-08	2.596E-08	7.318E-10	4.203E-08	3.541E-09	7.490E-10
bimba	10	9.975E-06	8.628E-06	1.060E-05	3.118E-05	1.094E-05	5.962E-04	5.643E-05	7.449E-06
8857	15	6.056E-07	5.965E-07	6.807E-07	8.626E-06	9.278E-07	4.329E-05	5.063E-06	5.919E-07
fandisk	10	1.520E-04	1.141E-04	1.573E-04	5.909E-04	3.748E-05	1.278E-02	1.639E-04	4.143E-05
6475	15	3.756E-05	3.018E-05	4.066E-05	1.221E-04	1.893E-06	8.643E-04	1.058E-05	2.975E-06
maxplanck	10	1.645E-05	1.605E-05	1.877E-05	3.347E-05	4.820E-06	3.367E-04	3.137E-05	4.322E-06
25445	15	5.411E-06	5.348E-06	6.073E-06	9.277E-06	4.795E-07	3.062E-05	2.408E-06	3.396E-07
chindragon	10	9.577E-04	8.346E-04	1.038E-03	2.315E-03	4.843E-04	2.319E-02	2.212E-03	5.310E-04
585018	15	3.622E-04	3.251E-04	3.866E-04	7.176E-04	5.463E-05	2.384E-03	2.522E-04	5.072E-05
palmyra	10	1.476E-05	1.450E-05	1.635E-05	2.757E-05	8.002E-06	3.893E-04	4.888E-05	7.350E-06
492465	15	4.749E-06	4.541E-06	5.031E-06	7.240E-06	7.950E-07	3.563E-05	4.421E-06	5.322E-07
welshdragon	10	1.086E-02	1.180E-02	1.187E-02	2.083E-02	6.320E-03	3.391E-01	3.332E-02	5.401E-03
291892	15	3.446E-03	3.529E-03	3.599E-03	5.188E-03	5.810E-04	3.039E-02	3.065E-03	3.568E-04

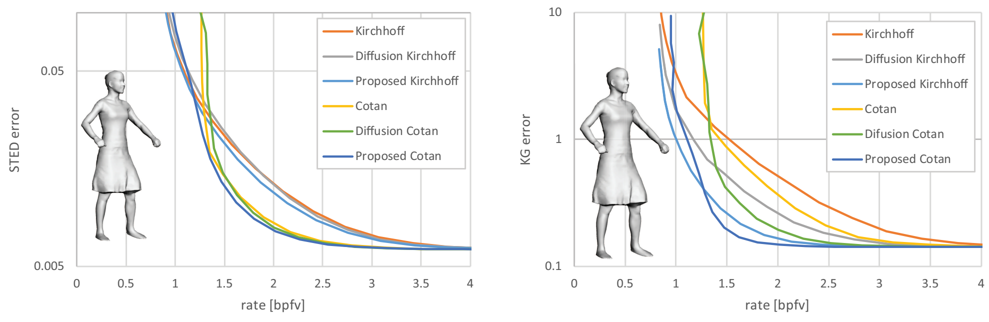
Table 6.2: Relative compression performance comparison for results obtained from 1723 models selected from *thingi10k* dataset [ZJ16].

		10 bpv				15 bpv			
		proposed	HPC	Diffusion	WP	proposed	HPC	Diffusion	WP
DAME	avg	1.380	1.192	1.307	38.87	1.315	1.176	1.232	24.15
	stdev	0.781	0.628	0.503	1076	0.802	0.498	0.469	378.8
	median	**1.127**	**1.000**	**1.169**	**2.490**	**1.121**	**1.000**	**1.140**	**2.090**
MSE	avg	13732	16038	13756	185.9	289659	300835	289963	237.7
	stdev	568038	569510	568585	3621	1.2E+07	1.2E+07	1.2E+07	6767
	median	**1.054**	**124.7**	**5.438**	**1.000**	**1.322**	**117.1**	**6.511**	**1.000**

6.3.2 Dynamic meshes

For dynamic mesh compression, we have compared 3 different coding approaches: unmodified Laplacian mesh compression [VP11; Váš+14], error diffusion [AK15] and our approach, combined with 2 discretizations of Laplace operator yielding positive semi-definite Laplacian matrix (Kirchhoff and Cotan), which overall generated 6 different configurations. Vertex trajectories on input were already reduced using PCA projection on the first 100 basis vectors. The comparison was done on selected models from commonly used dynamic mesh datasets [AG04; Vla+08]. We have used KG error [KG04] for mechanistic and STED [VS11] for perceptual performance.

RD curve comparison for the *samba* sequence is shown in Figure 6.3. Due to poor conditioning of the normal matrix, Cotan discretization yields worse results at lower data rates. However, since our approach works with the reduced Laplacian matrix, this issue is prevented and the results are always on par with the best approach at a given data rate. Table 6.3 shows that on all the tested datasets at data rates of 1.5 bits per frame per vertex (bpfv), our method achieves the best results, or is on par with the best approach for a given sequence.



(a) STED (b) KG error

Figure 6.3: RD curve comparison for *samba* sequence.

Table 6.3: Dynamic mesh compression results at 1.5 bpfv in comparison concerning different Laplace operator discretizations.

	dataset	Kirchhoff	Diffusion Kirchhoff	Proposed Kirchhoff	Cotan	Diffusion Cotan	Proposed Cotan
STED	samba	0.0236	0.0247	0.0215	0.0145	0.0141	0.0125
	march	0.0221	0.0223	0.0200	0.0148	0.0147	0.0138
	handstand	0.0376	0.0405	0.0345	0.0358	0.0502	0.0246
	jump	0.0220	0.0213	0.0209	0.0199	0.0193	0.0197
KG	samba	1.0474	0.5165	0.2643	0.8621	0.4093	0.1968
	march	0.8566	0.4777	0.2943	0.5882	0.3659	0.2665
	handstand	1.5786	0.8550	0.4320	1.7947	1.3516	0.3721
	jump	0.6632	0.3213	0.2901	0.4956	0.2991	0.2879

6.3.3 Computational performance

Working with the reduced Laplacian matrix instead of the normal matrix results in a performance improvement in decompression. This improvement is due to the algorithm not having to evaluate the normal matrix and the reduced Laplacian matrix having a smaller fill-in, which also means a smaller fill-in of the factor matrix. The normal matrix also has a roughly squared condition number in comparison with the reduced Laplacian. This can be seen in Table 6.4, where we measured the condition number of the matrix used for decompression, and times spent in factorization and solving phases. One drawback of our method in terms of computational performance is that the encoder now also has to perform the factorization to propagate the error. Nevertheless, the decoding time is usually considered more important since the data is expected to be encoded only once but decoded possibly multiple times.

Table 6.4: Condition number and times spent during factorization and solving phases (ms) for selected static meshes.

	reduced Laplacian			rectangular Laplacian		
dataset	cond. number	factor [ms]	solve [ms]	cond. number	factor [ms]	solve [ms]
bunny	6.24E+03	145	29	3.54E+06	952	47
bimba	6.21E+03	15	2	2.39E+06	84	11
fandisk	5.64E+03	15	2	3.13E+06	77	10
maxplanck	6.22E+03	70	7	3.35E+06	518	35
chindragon	1.54E+04	12158	490	4.08E+07	99635	1423
palmyra	8.75E+03	6175	274	1.02E+07	67793	890
welshdragon	1.73E+04	945	123	4.66E+07	6782	344

6.4 Summary

We have proposed a modification of the high-pass coding scheme, which achieves better control over the propagation of error in the reconstruction process by adjusting the quantized values during forward substitution. Some error is still accumulated during the subsequent back-substitution. Nevertheless, we have shown that this modification considerably improves the mechanistic performance of the method on both static and dynamic meshes when compared to the current state of the art. The modification also improves the computational performance of the decoder. The most significant limitation of the approach is, however, its performance on highly regular meshes, where the differential coordinates are usually of small magnitude. On such surfaces, the method tends to overcompensate in a zig-zag pattern, raising the entropy to higher values than those of the original method.

Although we planned to further improve upon this approach, due to the decline in the popularity of the fields of static and dynamic mesh compression since publishing this work, we have eventually decided to pursue different goals, mainly the design of temporal models for TVM compression.

7 Model-based Molecular Trajectory Compression

In this chapter, we will discuss our method for compressing molecular dynamics (MD) trajectories. To exploit the temporal coherence in such data, we have developed a temporal model denoted *canonical molecule*, which captures the general features common in all the input frames. This work was published in the *Journal of Molecular Graphics and Modelling* [DMV20]. The source code of the method was released under Apache License v2.0 and its current version is available at `https://jandvorak-uwb.github.io/pmc/`.

7.1 Molecular trajectories

MD trajectories represent the movement of molecular systems computed during MD simulations. Analogously to dynamic surface representations (TVMs, DMs and DPCs), we can think of the MD trajectories as a sequence of captured states of the represented smoothly moving molecular system, sampled at different discrete points in time. Assuming the studied molecules do not undergo any chemical reaction throughout the simulation (i.e., bonds between atoms do not change), we can represent the MD data as a single unoriented graph $G = (V, E)$ (each molecule corresponding to a single connected component), where V is a set of atoms and E is a set of bonds between them. Each atom is assigned a vector $\mathbf{v}_i = \left[\mathbf{v}_i^1, \mathbf{v}_i^2, \dots, \mathbf{v}_i^n\right] \in \mathbb{R}^{3n}$, where $\mathbf{v}_i^f$ is the position of the atom in the frame f, and n is the number of frames. This is similar to the graph representation of a dynamic mesh.

With increasing computational power, more complex macromolecules and longer trajectories of their atoms can be simulated. However, this data also becomes increasingly large, with datasets often occupying several gigabytes of disk space. For molecular data, compression approaches aiming to minimise mechanistic distortion are more desirable (yet there are few methods based, for example, on PCA, e.g., the PCZ format) since the compressed data is expected to be further analyzed after decompression. Most of the compression formats in use are being developed as a part of various MD simulation tools (e.g., Gromacs XTC [GRO18] and TNG [SLS11]). HRTC [Huw+16], the most recent compression format, promises data rates below 1 bit per coordinate (bpc), however, we were unable to reproduce such results in our experiments. The current limitation of the previous work, in our opinion, is that the atom bond information is not exploited to its full potential.

7.2 Algorithm description

Our key observations are that the molecular movement is quite constrained (e.g., the distance between two atoms forming a bond does not change much throughout the sequence) and that the greatest variance in atom positions over time occurs in so-called *dihedral angle movement* (see Figure 7.1). To exploit this information, we propose to construct the canonical molecule, which is a hypothetical molecule frame with connectivity identical to the compressed data and atom locations $\hat{\mathbf{v}}_i$ chosen to locally fit all the frames.

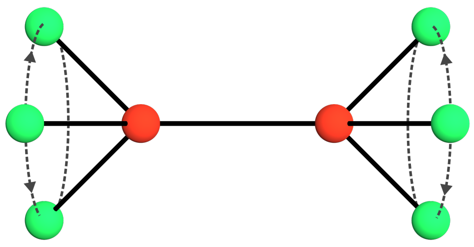


Figure 7.1: Molecular movement is mainly constrained to so-called *dihedral angles*.

The algorithm proceeds as follows: first, the canonical molecule is constructed and encoded (see Section 7.2.1). Then the data is encoded during a bond-induced connectivity traversal. In each traversal step, unprocessed neighbours of a particular atom are frame-by-frame encoded by locally aligning the canonical molecule to the currently processed frame and coding the alignment information and local corrections (see Section 7.2.2).

7.2.1 Canonical molecule

The canonical molecule is constructed incrementally, starting with a single atom and its neighbourhood, and determining additional atom locations in a Depth First Search (DFS) traversal of the connectivity.

The first atom i is chosen arbitrarily while preferring vertices of valence at least 2. Its position $\hat{\mathbf{v}}_i$ in the canonical molecule is the average position computed over all frames. The position of each neighbour j is determined concerning the relative position in the first frame as follows:

$$\hat{\mathbf{v}}_j = \hat{\mathbf{v}}_i + \left(\mathbf{v}_j^1 - \mathbf{v}_i^1\right).$$

Next, a DFS is started from the initial atom and in each step, the locations of all neighbours of a particular atom are determined. In each step, there is an atom i and its predecessor in DFS with an index p, whose positions $\hat{\mathbf{v}}_i$ and $\hat{\mathbf{v}}_p$ were already determined, and a set of indices of neighbours $\mathcal{C}$, whose positions are yet to be determined:

$$\mathcal{C} = \{j \in \mathcal{N}(i) \mid j \notin \mathcal{D}\},$$

where $\mathcal{N}(i)$ is the neighbourhood of i-th atom, and $\mathcal{D}$ is a set of indices of all vertices, whose canonical positions are already determined.

First, we center all the local neighbourhoods in all frames, so that $\hat{\mathbf{v}}_i$ coincides with the origin. Next, we align all the local neighbourhoods in all frames so that the vector $\hat{\mathbf{v}}_p$ - $\hat{\mathbf{v}}_i$ coincides with the negative direction of the Z axis of the coordinate system. Having a unit vector $\mathbf{v} = (x, y, z)$ to be aligned, an aligning rotation matrix $\mathbf{M}_\mathbf{v}$ can be constructed as

$$\mathbf{M}_\mathbf{v} = \begin{bmatrix} \frac{-y}{h} & \frac{x}{h} & 0 \\ \frac{zx}{h} & \frac{zy}{h} & -\left(\frac{x^2}{h} + \frac{y^2}{h}\right) \\ -x & -y & -z \end{bmatrix},$$

where $h = \sqrt{x^2 + y^2}$.

In each frame, we construct the previous edge direction

$$\mathbf{e}^f = \left(\mathbf{v}_p^f - \mathbf{v}_i^f\right) / \|\mathbf{v}_p^f - \mathbf{v}_i^f\|,$$

and transform all neighbouring vertices into a common coordinate system as

$$\bar{\mathbf{v}}_j^f = \mathbf{M}_{\mathbf{e}^f}\left(\mathbf{v}_j^f - \mathbf{v}_i^f\right).$$

Now, we must find a set of positions that captures the distribution of the local neighbourhoods. Note that the alignment by matrix $\mathbf{M}_\mathbf{v}$ results in an arbitrary rotation around the Z axis. We must account for this by solving for an angle of rotation for each frame as well. We formulate the problem as a variational problem of minimizing the following energy:

$$E = \sum_f \sum_j \|\hat{\mathbf{v}}_j - \mathbf{R}_{\alpha_f}\bar{\mathbf{v}}_j^f\|^2,\ \mathbf{R}_{\alpha_f} = \begin{bmatrix} \cos\alpha_f & -\sin\alpha_f & 0 \\ \sin\alpha_f & \cos\alpha_f & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.1}$$

over the unknown positions $\hat{\mathbf{v}}_j$ of the vertices $j \in \mathcal{C}$ in the canonical molecule and unknown rotation angles α_f. The energy is optimised iteratively. Starting with some initial guess on the positions $\hat{\mathbf{v}}_j$, we alternate two steps: determining optimal rotation angles with positions fixed, and determining optimal positions with rotation angles fixed (See Figure 7.2). Since the total energy is positive and decreases in each step, the algorithm converges to a minimum. In our verification experiments, this minimum always turned out to be the global minimum of the energy, and only five iterations sufficed to obtain a result that was very close to it.

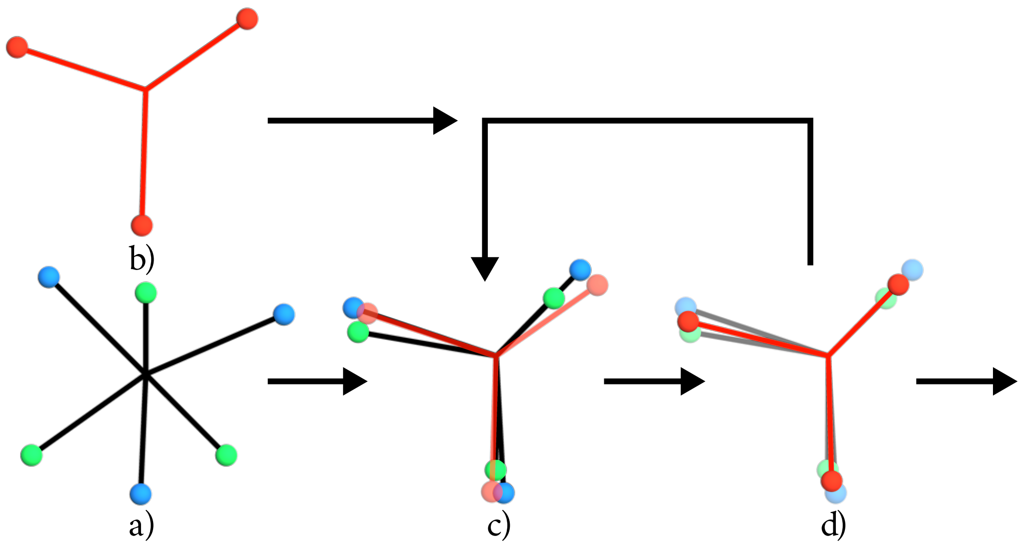


Figure 7.2: Construction of a canonical molecule for the neighbourhood of a single atom. a) Input positions (each colour represents the situation in a different frame). b) The initial guess. c) Determining optimal rotation angles. d) Determining optimal positions. The result of d) will be used as the initial guess for the next iteration until convergence.

Having the canonical positions fixed and considering a single unknown angle α_f, minimizing Eq. (7.1) can be rewritten as a 2D rotation alignment problem

$$\alpha_f = \arg\min_{\alpha} \sum_j \|\hat{\mathbf{v}}_j - \mathbf{R}_\alpha \bar{\mathbf{v}}_j^f\|^2. \tag{7.2}$$

A closed-form solution is derived as follows:

$$\begin{aligned} \alpha_f &= \arg\min_{\alpha} \sum_j \left(\hat{\mathbf{v}}_j - \mathbf{R}_\alpha \bar{\mathbf{v}}_j^f\right)^\top \left(\hat{\mathbf{v}}_j - \mathbf{R}_\alpha \bar{\mathbf{v}}_j^f\right) \\ &= \arg\min_{\alpha} \sum_j \left(\hat{\mathbf{v}}_j^\top \hat{\mathbf{v}}_j - 2\hat{\mathbf{v}}_j^\top \mathbf{R}_\alpha \bar{\mathbf{v}}_j^f + \left(\mathbf{R}_\alpha \hat{\mathbf{v}}_j\right)^\top \mathbf{R}_\alpha \bar{\mathbf{v}}_j^f\right) \end{aligned}$$

Since $\mathbf{R}_\alpha^\top = \mathbf{R}_\alpha^{-1}$, the last term can be rewritten as $\left(\mathbf{R}_\alpha \hat{\mathbf{v}}_j\right)^\top \mathbf{R}_\alpha \bar{\mathbf{v}}_j^f = \hat{\mathbf{v}}_j^\top \mathbf{R}_\alpha^{-1} \mathbf{R}_\alpha \bar{\mathbf{v}}_j^f = \hat{\mathbf{v}}_j^\top \bar{\mathbf{v}}_j^f$. This term is constant and thus can be omitted in the minimization, as well as the first term $\hat{\mathbf{v}}_j^\top \hat{\mathbf{v}}_j$. Minimization of the remaining term may be further simplified by omitting the multiplicative constant and rewriting it as maximization of its negative as follows:

$$\begin{aligned}
\alpha_f &= \arg\max_{\alpha} \sum_j \hat{\mathbf{v}}_j^\top \mathbf{R}_\alpha \bar{\mathbf{v}}_j^f \\
&= \arg\max_{\alpha} \sum_j \left(\hat{x}_j \left(\cos\alpha\bar{x}_j^f - \sin\alpha\bar{y}_j^f\right) + \right. \\
&\qquad \left. \hat{y}_j \left(\sin\alpha\bar{x}_j^f + \cos\alpha\bar{y}_j^f\right) + \hat{z}_j\bar{z}_j^f\right) \\
&= \arg\max_{\alpha} \left(\cos\alpha \sum_j \left(\hat{x}_j\bar{x}_j^f + \hat{y}_j\bar{y}_j^f\right) + \right. \\
&\qquad \left. \sin\alpha \sum_j \left(\hat{y}_j\bar{x}_j^f - \hat{x}_j\bar{y}_j^f\right)\right) \\
&= \operatorname{atan2}\left(\sum_j (\hat{y}_j\bar{x}_j^f - \hat{x}_j\bar{y}_j^f), \sum_j (\hat{x}_j\bar{x}_j^f + \hat{y}_j\bar{y}_j^f)\right).
\end{aligned}$$

Since the angles are independent of each other, they can be solved for independently. Next, having the angles fixed, the canonical molecule positions are updated. This is trivial, since the solution of minimizing the energy given by Eq. (7.1) with fixed angles can be expressed as

$$\hat{\mathbf{v}}_j = \frac{1}{F} \sum_f \mathbf{R}_{\alpha_f} \bar{\mathbf{v}}_j^f.$$

This iterative process yields the shape of the local neighbours with the entry edge aligned with the Z axis. To obtain global positions, the reverse mapping is applied. With the global position of the preceding atom $\bar{\mathbf{v}}_p$ and the global position of the current atom $\bar{\mathbf{v}}_i$, the vector $\mathbf{p} = \left(\bar{\mathbf{v}}_p - \bar{\mathbf{v}}_i\right) / \|\bar{\mathbf{v}}_p - \bar{\mathbf{v}}_i\|$ is used to build the required rotation matrix $\mathbf{M}_\mathbf{p}^\top$.

After the canonical molecule is constructed, it is encoded using a simple predictive compression method. The molecule is again traversed during a DFS and the traversal predecessor positions are used as predictions. The corrections are then quantized and encoded using an entropy coder.

7.2.2 Prediction scheme

The MD trajectories are also encoded during a DFS traversal; however, the canonical frame is used for prediction.

The position of the initial vertex $\mathbf{v}_j^f$ in the current frame f is predicted by its position in the canonical molecule. The corresponding correction vector is computed as:

$$\mathbf{c}_j^f = \mathbf{v}_j^f - \hat{\mathbf{v}}_j.$$

The encoded position of the initial vertex is then used to predict the positions of its neighbours.

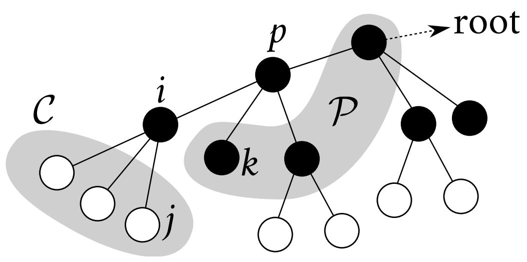


Figure 7.3: The situation during one step of a DFS traversal. Already encoded vertices are black, vertices to be encoded are white, and the current vertex is i.

In each step of the DFS traversal, the newly visited vertex will be denoted i and its predecessor on the path to the root will be denoted p, see Figure 7.3. A set $\mathcal{C}$ contains all neighbours of i that are still to be encoded and the set $\mathcal{P}$ of preceding neighbours of p is defined as:

$$\mathcal{P} = \{k \in \mathcal{N}(p) \mid k \neq i\}.$$

The local neighbourhood in the canonical molecule and the local neighbourhood in the current frame f are transformed into a common coordinate system, where both $\bar{\hat{\mathbf{v}}}_i$ (transformed $\hat{\mathbf{v}}_i$) and $\bar{\mathbf{v}}_i^f$ (transformed $\mathbf{v}_i^f$) coincide with the origin and both edges to the predecessor p are aligned to the negative direction of the Z axis. This is almost identical to the alignment discussed in Section 7.2.1, only this time it is necessary to also transform $\hat{\mathbf{v}}_k$ and $\mathbf{v}_k^f$ for all $k \in \mathcal{P}$. After that, a registration is applied using Eq. (7.2) to align all $\bar{\hat{\mathbf{v}}}_k$ to their corresponding counterparts $\bar{\mathbf{v}}_k^f$. The result is a rotation matrix $\mathbf{R}_\beta$, which transforms both local neighbourhoods into a unique initial state. This rotation is then applied to positions of all $j \in C$ as follows:

$$\mathbf{p}_j = \mathbf{R}_\beta \bar{\hat{\mathbf{v}}}_j,$$
$$\mathbf{u}_j^f = \mathbf{R}_\beta \bar{\mathbf{v}}_j^f.$$

To compensate for the dihedral angle movement, the encoder estimates an angle α which represents rotation $\mathbf{R}_\alpha$ aligning $\mathbf{p}_j$ and $\mathbf{u}_j^f$, similarly to Eq. (7.2). The corrections are then calculated as follows:

$$\mathbf{c}_j^f = \mathbf{u}_j^f - \mathbf{R}_\alpha \mathbf{p}_j.$$

The alignment angle α and all the corrections are then quantized and encoded using a Huffman coder [Huf52]. To exploit the temporal coherence in the alignment angles, only the differences between two subsequent quantized values are encoded.

To reconstruct the atom position during decoding, the decoder follows the encoding process to obtain the predictions $\mathbf{p}_j$. The reconstructed positions of decoded vertices $\mathbf{r}_j^f$ are then calculated as:

$$\mathbf{r}_j^f = \mathbf{r}_i^f + \mathbf{M}_{\mathbf{e}_f}^\top \left(\mathbf{R}_\alpha \mathbf{p}_j + \mathbf{c}_j^f \right),$$

where $\mathbf{M}_{\mathbf{e}_f}$ is a rotation matrix mapping the edge between vertices i and p to the Z-axis. Note that α, $\mathbf{p}_j$ and $\mathbf{c}_j^f$ are all distorted by the compression.

7.3 Experimental results

We demonstrate the applicability of our compression method on several datasets. Protein datasets (p53, ARID and DhaA31) come from the research of certain enzymes and their mutations and from cancer research. Models of liquids (ethanol, water) are also included. Quantitative information about all tested datasets is listed in Table 7.1.

Table 7.1: Datasets information - the number of atoms, snapshots, the timestep Δt and the absolute size of uncompressed data for 32 bpc (1 GB = 10^9 bytes).

	Atoms	Snapshots	Δt [ps]	Size [GB]
p53	3 008	11 501	4	0.42
p53-0.05	3 008	12 001	0.05	0.43
ARID	2 127	62 501	16	1.60
DhaA31	4 645	100 000	2	5.57
ethanol	4 500	10 001	1	0.54
water	4 000	10 001	1	0.48

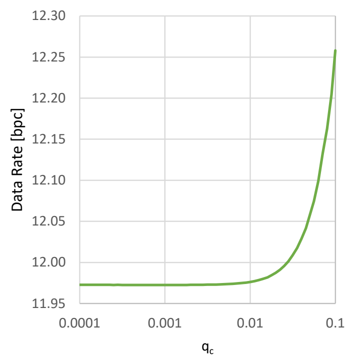
7.3.1 Configuration

The compression performance of our method is controlled by three quantization parameters:

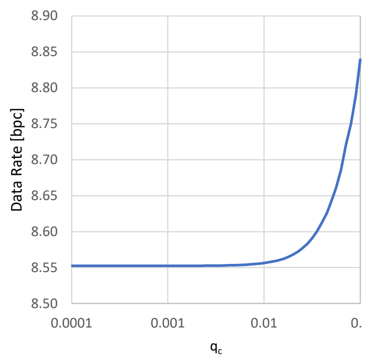
- Canonical molecule quantization q_c
- Angle quantization q_a
- Residual quantization q_r

To achieve low data rates, it is crucial to assign these parameters an appropriate value. Although only q_r influences the data distortion, it is necessary to adjust the remaining parameters as well. Optimal values of q_a and q_c depend on q_r. When encoding with very low precision, it is redundant to use too precise prediction. On the other hand, highly inaccurate prediction causes a significant increase in data rate, which is more apparent at higher precision.

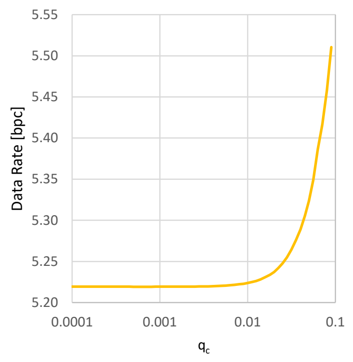
For $q_r \in \{0.1, 0.01, 0.001, 0.0001\}$, we first fixed q_a and measured the bitrate for various values of q_c, then the same experiment was executed with fixed q_c and various values of q_a. To capture the relation between q_r and the remaining parameters, we identified for each tested value of q_r the best configuration of the remaining parameters.



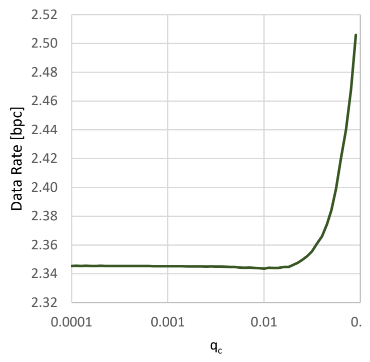
(a) $q_r = 0.0001$



(b) $q_r = 0.001$



(c) $q_r = 0.01$



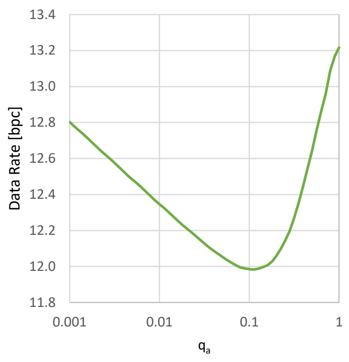
(d) $q_r = 0.1$

Figure 7.4: Relation between data rate and precision of quantization of canonical molecule on DhaA31 dataset. Data rate axes are not consistent between charts.

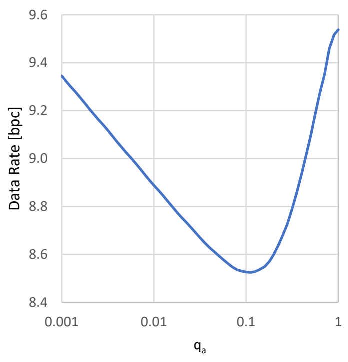
Figure 7.4 shows the relation between data rate in bits per coordinate (bpc) and q_c for dataset DhaA31. For other datasets, the results exhibit similar behaviour. When focusing on the results of a single value of q_r (for example Figure 7.4c), one can observe the increase of bitrate with very low precision. However, very high precision still does not cause any significant data rate increase, since the size of the canonical molecule data is only a small fraction of all the encoded data. We expect that the influence of a higher precision of canonical molecule is significant for datasets with fewer frames, but such datasets usually do not require any compression at all. When comparing the results of different values of residual quantization q_r, one can see that for a lower residual precision, the point where the entropy starts to significantly increase is lower than in the case of the higher precision. However, the difference is quite small. Additionally, for any of the tested values of q_r, we can select any value of q_c between 0.00015 and 0.0015 and obtain satisfactory results. This is supported by Table 7.2, which shows the best-found values of q_c for each q_r and each dataset.

Table 7.2: Best found values of q_c for all tested datasets

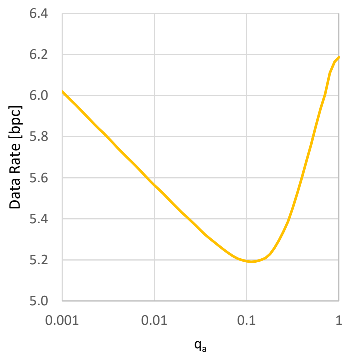
	$q_r = 0.0001$	$q_r = 0.001$	$q_r = 0.01$	$q_r = 0.1$
ARID	0.00063	0.00125	0.00125	0.00315
DhaA31	0.0008	0.0008	0.0008	0.0112
p53	0.0018	0.0018	0.002	0.01
p53-0.05	0.002	0.00224	0.00224	0.0112
ethanol	0.01	0.018	0.01	0.01
water	0.08	0.1	0.1	0.1



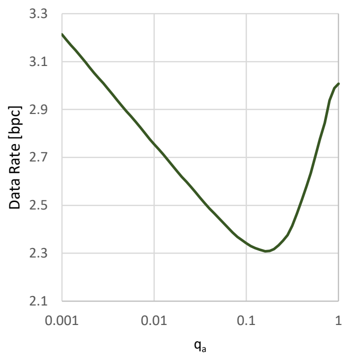
(a) $q_r = 0.0001$



(b) $q_r = 0.001$



(c) $q_r = 0.01$



(d) $q_r = 0.1$

Figure 7.5: Relation between data rate and precision of quantization of angles on ARID dataset. Data rate axes are not consistent between charts.

Figure 7.5 visualizes how changing values of q_a affects the data rate for the ARID dataset. As with the experiment with q_c, the results for various datasets were very similar with a single exception being the *water* dataset, for which no angle data is encoded. In all cases, the best values were found around $q_a = 0.1$ (see Table 7.3). However, the q_a still depends on q_r since with increasing precision of residual quantization the best precision of angles also increases.

Table 7.3: Best found values of q_a for all tested datasets

	$q_r = 0.0001$	$q_r = 0.001$	$q_r = 0.01$	$q_r = 0.1$
ARID	0.112	0.112	0.112	0.16
DhaA31	0.112	0.112	0.112	0.16
p53	0.1	0.112	0.112	0.16
p53-0.05	0.1	0.112	0.112	0.14
ethanol	0.16	0.28	0.28	0.355

The results show that a connection between q_r and the rest of the configuration parameters exists, however, it is so weak, that it is possible to pick static parameters and the results are still satisfactory. For this reason, we have decided to use static configuration $q_c = 0.001$ and $q_a = 0.1$ in the rest of the experiments. This makes the q_r the only parameter that controls the rate and the distortion. Tables 7.4 and 7.5 show the data rate cost of using static configuration instead of the experimentally found best values.

Table 7.4: Data rate difference in bpc when using $q_c = 0.001$ instead of best found values

	$q_r = 0.0001$	$q_r = 0.001$	$q_r = 0.01$	$q_r = 0.1$
ARID	8.17E-04	2.82E-05	3.43E-05	1.72E-04
DhaA31	1.26E-05	6.99E-06	2.99E-05	1.62E-03
p53	2.73E-04	1.87E-04	1.86E-04	1.57E-03
p53-0.05	3.33E-04	1.80E-04	2.33E-04	2.49E-03
ethanol	1.67E-03	1.01E-03	8.55E-04	7.89E-04
water	1.17E-03	1.05E-03	1.04E-03	9.34E-04

Table 7.5: Data rate difference in bpc when using $q_a = 0.1$ instead of best found value

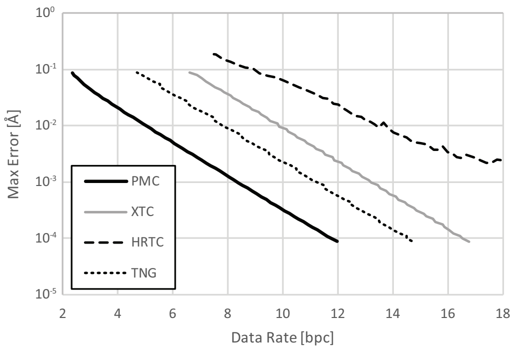
	$q_r = 0.0001$	$q_r = 0.001$	$q_r = 0.01$	$q_r = 0.1$
ARID	1.64E-03	1.65E-03	3.17E-03	3.36E-02
DhaA31	3.23E-03	2.95E-03	3.96E-03	3.62E-02
p53	0	3.48E-04	2.01E-03	3.35E-02
p53-0.05	0	1.62E-03	3.40E-03	2.62E-02
ethanol	1.40E-02	4.15E-02	4.31E-02	6.49E-02

7.3.2 Rate-distortion comparison

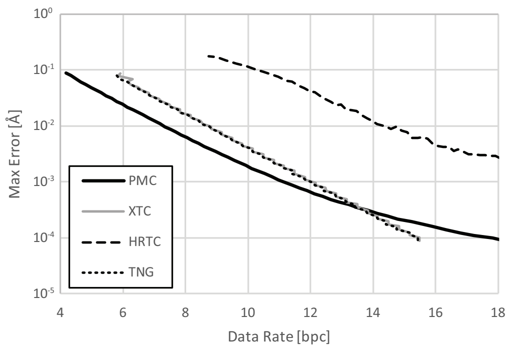
RD comparison was done against XTC [GRO18], TNG [SLS11] and HRTC [Huw+16]. To measure the distortion, we used the maximal Euclidean distance between the original and the reconstructed position of any atom in any frame:

$$err = \max_{i,f} \left\| \mathbf{r}_i^f - \mathbf{v}_i^f \right\| . \tag{7.3}$$

Note that this error measure is more sensitive than MSE. The default configuration was used for competing methods with one exception of *qp-ratio* (the ratio of quantization to approximation error) of HRTC, which was set to $qp = 0.1$ since it provided consistently better results than the default value. The main interest was in the precision of 2 decimal places (in Å) since this is the same as the default precision of the XTC [GRO18] and TNG [SLS11] formats (3 decimal places, in nanometers). The rate-distortion comparison was evaluated on ARID, DhaA31, p53, water and ethanol datasets. Example RD curve for protein datasets is shown in Figure 7.6a and for liquids in Figure 7.6b.



(a) DhaA31 (protein)



(b) water (liquid)

Figure 7.6: Comparison of rate-distortion curves of tested compression methods on selected datasets.

For proteins, our proposed compression scheme outperformed all compared methods. The difference w.r.t. the XTC algorithm was usually around 5 bpc and 3-4 bpc w.r.t the TNG algorithm when comparing results of equal precision. The HRTC performed worst in all cases. In the case of liquids, the difference w.r.t TNG and XTC was less significant. In fact, at very high precision, both methods obtained better results. However, our method still performed best at the precision usually used in lossy compression.

7.3.3 File size comparison

We have compared the results of PMC against other state-of-the-art methods and file formats in terms of bpc and the absolute compressed file size in gigabytes. Other than the methods compared in the previous section, we have also included the MD-Traj H5 file format and a PCA-based compression scheme implemented in the PCA Suite package (PCZ). The parameters of all tested methods were tuned to achieve a good compression with a limited maximal error given by Eq. (7.3). We have considered the default precision limit of XTC coordinates in Gromacs, which allows the deviation of each coordinate from its original value by $+/-$ 0.005 Å. This gives us the maximal error limit $err = \sqrt{0.005^2 + 0.005^2 + 0.005^2} \approx 0.00866$ Å. The results are summarized in Table 7.6. In the case of HRTC and PCZ, we were not able to always achieve these error limits. For this reason, we also list the corresponding errors.

Our method (PMC) achieved on average 5.2 bpc on protein datasets and 7.25 bpc on liquids for the error threshold 0.00866 Å, which is better than the results of other tested methods. Nevertheless, we believe that other methods can be more powerful in certain situations, for example, when atoms do not move much between snapshots (higher sampling rate) or when the model does not have a stable three-dimensional structure.

Table 7.6: Comparison of the proposed PMC against other formats in terms of the absolute and relative size of compressed data (lower values are better, 1 GB = 10^9 B). The maximal allowed error limit (Eq. (7.3)) was $\approx$ 0.00866 Å. HRTC and PCZ may exceed this limit.

	Relative size of compressed data [bpc]						Absolute size of compressed data [GB]						Max. error [Å]	
	H5	XTC	TNG	HRTC	PCZ	PMC	H5	XTC	TNG	HRTC	PCZ	PMC	HRTC	PCZ
p53	13.4	10.1	8.0	10.2	9.4	5.2	0.17	0.13	0.10	0.13	0.12	0.07	0.00875	2.54
p53-0.05	11.9	10.2	6.4	8.2	9.7	5.1	0.16	0.14	0.09	0.11	0.13	0.07	0.00880	1.55
ARID	14.1	10.2	9.0	16.5	5.6	5.2	0.70	0.51	0.45	0.82	0.28	0.26	0.00050	3.72
DhaA31	14.7	10.1	8.1	12.9	6.6	5.2	2.57	1.77	1.41	2.24	1.15	0.91	0.00906	3.56
ethanol	15.5	9.4	9.4	14.6	7.5	7.0	0.26	0.16	0.16	0.25	0.13	0.12	0.00883	25.79
water	15.4	8.8	8.8	13.8	14.7	7.5	0.23	0.13	0.13	0.21	0.22	0.11	0.00876	11.91

7.3.4 Running time evaluation

We have also compared the running time of the proposed PMC method against implementations of the XTC, TNG and HRTC compression algorithms on all tested datasets. We were able to parallelize the calculation and encoding/decoding of the predictions, while the encoding of time-coherent angles remained single-threaded. All tests were performed on an Intel© Core™ i7-4930K CPU running at 3.40GHz. This CPU offers 12 logical cores (6 physical cores with hyper-threading support) and the system has 64 GiB RAM (1600 MHz). The results in Tables 7.7 show the comparison of encoding and decoding times. For large molecules, the proposed method is approximately twice as slow compared to XTC, which turned out to be the fastest single-threaded method, but the decoding performance is similar for both methods. In the case of liquids, the decoding times are three times slower.

Table 7.7: Performance of the proposed PMC method compared against XTC, TNG and HRTC.

	Encoding time [s]					Decoding time [s]				
Method	PMC		XTC	TNG	HRTC	PMC		XTC	TNG	HRTC
Threads	12	1	1	1	1	12	1	1	1	1
p53	3.1	9.0	3.9	5.6	161.9	3.3	6.0	4.9	5.6	18.5
p53-0.05	4.4	10.8	5.3	7.1	161.6	3.4	6.1	5.0	5.6	18.4
ARID	11.5	34.9	16.8	22.5	616.8	8.7	20.1	19.9	22.1	73.1
DhaA31	40.8	124.2	56.7	80.6	2191.9	29.2	69.8	69.5	80.7	270.0
ethanol	7.2	10.8	6.5	9.0	222.3	12.9	14.2	6.1	7.4	26.5
water	7.1	7.1	5.6	14.7	190.3	16.2	16.0	5.3	28.9	22.2

7.4 Summary

We have presented a new method for the compression of MD trajectories. The method utilizes the information about atomic bonds in a molecule, captures the local, mostly rotational, movement of atoms with respect to their bonded neighbours, and uses this information for the prediction of atom positions in each frame. This approach allows us to exploit the local stiffness that results from chemical bonds, rather than relying on global stability. For proteins, the method achieves the average data rate of 5.2 bpc (bits per coordinate) with the maximal error 0.00866 Å. The results are substantially better than the results obtained with other state-of-the-art methods and allow either saving 1.3-3.8 bits per coordinate at the same precision or providing up to 10× better precision at the same data rate.

8 Volume Element Tracking for Time-Varying Meshes

Inspired by the success of the temporal-model-based approaches, such as the one proposed by Doumanoglou et al. [Dou+14b], we have decided to focus on ways of extending these approaches to more general mesh sequences. As a result, we have proposed a novel temporal model for TVMs denoted *tracked centers*. The initial work presented at *ICCS 2021* [DVV21] is discussed in Section 8.3. Since then, we have been able to publish two additional improvement papers. The first improvement was presented at *SMI 2021* and published in *Computers & Graphics* [Dvo+22a] (see Section 8.4). The current version of the model was presented at *ICCS 2023* [DHV23]. Since this paper presents two distinct contributions, these will be discussed in separate sections (Sections 8.5 and 8.6). The source code containing the versions of our method presented in our two most recent papers was released under MIT License and is available at `https://gitlab.kiv.zcu.cz/jdvorak/arap-volume-tracking`.

8.1 Motivation

The most popular way of obtaining a temporal model to represent TVMs is surface tracking. Usually, a certain template surface is sequentially aligned to all the frames using non-rigid registration [MS10; Li+09]. The simplest methods rely on the template surface being given a priori and on an assumption that it reflects the ground-truth topological information. Such methods usually fail in the presence of frequent self-contact in the input. This issue might be mitigated to some extent by subdividing the surfaces into patches [CBI10; HBI13], or by identifying frames with significant changes in appearance and working with subsequences between such frames [Guo+15; Col+15; Pra+17; Moy+21]. Bojsen-Hansen et al. [BLW12] detect topology changes and adjust the template shape accordingly, however, they incorrectly assume surface correspondences to be bijective outside of the adjusted parts.

Budd et al. [Bud+12] build a shape similarity tree, which allows alignment of the more similar rather than subsequent frames.

In many practical scenarios where a surface is captured from multiple viewpoints, the overall enclosed volume changes negligibly – it does not suddenly appear or disappear. This, however, does not hold for the surface itself: a part might disappear due to self-contact (see Figure 8.1). For this reason, some methods incorporate volume information into tracking. For example, Wuhrer et al. [Wuh+15] used a finite-element method to model the deformation of the provided template in places where no correspondence was found. Slavcheva et al. [SBI17] used eigenfunctions of the Laplacian operator on signed distance field evolution to model correspondences. The most relevant approach to our work was proposed by Huang et al. [Hua+16; Hua+18], who performed non-rigid registration of centroidal Voronoi tessellations (CVTs). However, their approach does not enforce the smooth movement of the represented surface, which might result in high-frequency tracking errors.

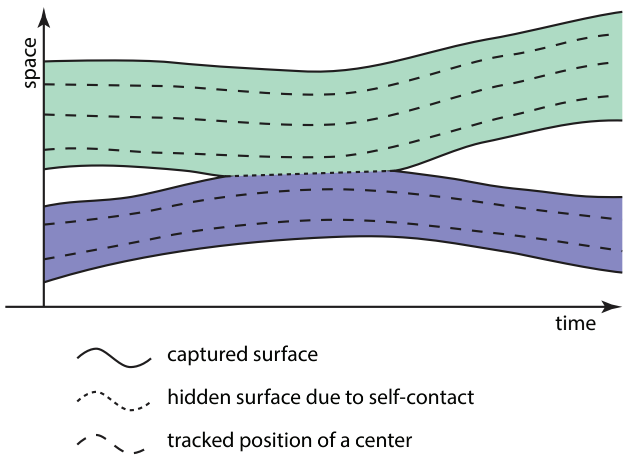

space
time
captured surface
hidden surface due to self-contact
tracked position of a center

Figure 8.1: Schematic of volume tracking in 1D space: During the sequence, two objects (green and blue) touch and then separate.

Recently, machine learning models became popular for representing temporal sequences. These are especially successful on sparse data (e.g., single-view RGBD video). Relevant to our work is, for example, OccupancyFlow [Nie+19], a learned occupancy function deformed by a neural vector field, as well as the work of Božič et al. [Bož+21] who train a neural deformation graph. The main limitation of neural models is, however, that working with them is less intuitive.

One additional thing to consider, if one plans to use a certain model in compression, is its data footprint since the model will be encoded alongside the data. While the tracked template surface can be encoded efficiently as a dynamic mesh, attempts to adapt it to the encountered topology changes (e.g., by Bojsen-Hansen et al. [BLW12]) quite often require large amounts of additional data to be stored.

8.2 Tracked centers

Instead of points on the surface, we propose to track a fixed set C of k points (denoted centers), each representing a small volume surrounding it, whose positions vary in time. Each center follows a certain trajectory $\mathbf{c}_i = \left[\mathbf{c}_i^{(0)}, \mathbf{c}_i^{(1)}, \ldots, \mathbf{c}_i^{(n-1)}\right] \in \mathbb{R}^{3n}$, where n is the number of frames and $\mathbf{c}_i^{(f)}$ is the position of the i-th center in the f-th frame. Regardless of the method used to obtain them, the centers should

- cover all parts of the input object in each frame,
- be distributed evenly over the volume of the objects in each frame and
- move consistently in time, i.e. neighbouring centers should move coherently.

As will be discussed in the following sections, the difference between individual proposed approaches is mainly in the interpretation of the last objective: whether the movement consistency is global or only frame-by-frame, what information is used to model center neighbourhood and what moving coherently means. These differences are summarized in Table 8.1.

Table 8.1: Major differences between our proposed pipelines for obtaining tracked centers.

Method	Ref.	Consistency	Neighborhood	Movement
PBT	[DVV21]	Frame-by-frame	Proximity	Smooth vector field
ARAP tracking	[Dvo+22a]	Frame-by-frame	Proximity and IIR filtered motion dissimilarity	As-rigid-as-possible
Max-based affinity	[DHV23]	Frame-by-frame	Maximum of both proximity and motion dissimilarity	As-rigid-as-possible
Global optimization	[DHV23]	Global	Maximum of motion dissimilarity	As-rigid-as-possible

Similarly to point clouds, the tracked centers are not connected. There is only a particular notion of a center neighbourhood. For this reason, they can represent surfaces of arbitrarily changing topology, as long as the surface provides a sufficient notion of the inside/outside distinction. The model was also designed with data footprint in mind. Since the ordering of the centers in each frame is consistent, we can treat such data globally and reduce the center trajectories using PCA and encode it efficiently, for example by the COBRA algorithm [VS09].

8.3 Proximity-based tracking

In the original method, we modelled the center neighbourhood using the spatial proximity in the given frame. For this reason, in our subsequent work, we denote this method the *Proximity-based tracking* (PBT). The method proceeds frame-by-frame, meaning that when computing the positions of centers in a certain frame, it only considers the given frame and the previous frame.

8.3.1 Algorithm description

First, the frame is converted into a dense regular square voxel grid by sampling the indicator function $IF^{(f)}(\mathbf{x})$, which returns 1 in the interior and 0 otherwise. To compute the indicator function, we originally used the ray-shooting technique, essentially evaluating the number of intersections with the surface along a certain ray. Choosing axis-aligned rays allows reusing the previously computed intersection points for evaluating a whole column of the regular sampling grid. Note that this is only possible for watertight models. Alternatively, the method can also accept the sequence of voxel grids directly as input, which means that it can be applied to any sequence of shapes for which it is possible to determine the inside/outside information with an acceptable amount of certainty (e.g., implicit representations, point clouds, etc.).

Center positions $\mathbf{c}_i^{(f)}$ are initially set by sampling n random occupied voxels. To achieve a uniform distribution of the centers in the volume, we perform Lloyd's algorithm [Llo82]: For each center, its Voronoi cell $V_i^{(f)}$ of occupied voxel positions is iteratively evaluated, such as

$$V_i^{(f)} = \left\{\mathbf{x} : IF^{(f)}(\mathbf{x}) = 1 \wedge \|\mathbf{x} - \mathbf{c}_i^{(f)}\| \leq \|\mathbf{x} - \mathbf{c}_j^{(f)}\|\right\}, \tag{8.1}$$

for every j, and the center is moved to the centroid $\bar{\mathbf{x}}_i^{(f)}$ of such cell (see Figure 8.2). The Lloyd's algorithm ends after a fixed number of iterations. For the first frame, the tracking already terminates.

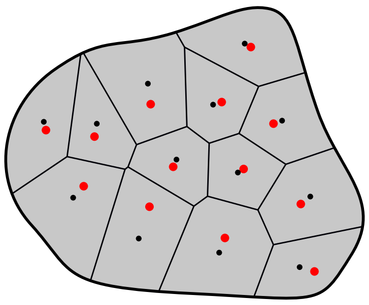


Figure 8.2: A single step of Lloyd's algorithm: each center (red) is moved to the centroid of its corresponding Voronoi cell (black).

For any subsequent frame, after distributing the centers uniformly in the volume, we continue by estimating correspondences between the centers of the current and the previous frame. Interpreting the correspondence estimation as an optimal assignment problem, we use the Kuhn–Munkres (also known as *Hungarian*) algorithm [Kuh55]. For the weights of the candidate pairs, we use squared distances. However, we also penalize any pairs for which the line connecting the two corresponding centers passes outside of the object (see Figure 8.3).

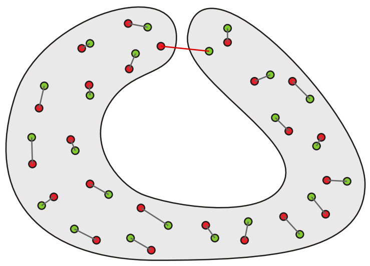


Figure 8.3: Red and green dots represent different samplings of the same grey domain. The red correspondence lies partially outside of the domain and thus should be penalized.

Finally, we apply an optimization process to ensure the consistency of the represented movement. Our energy $E^{(f)}$ consists of uniformity and smoothness terms: $E^{(f)} = E_u^{(f)} + \beta E_s^{(f)}$. The uniformity term is evaluated as a squared distance to the centroid $\bar{\mathbf{x}}_i^{(f)}$ of the Voronoi cell $V_i^{(f)}$ (see Eq. 8.1):

$$E_u^{(f)} = \frac{1}{2} \sum_{\mathbf{c}_i \in C} \|\mathbf{c}_i^{(f)} - \bar{\mathbf{x}}_i^{(f)}\|^2. \tag{8.2}$$

Similar to Lloyd's algorithm, it ensures uniform distribution of centers inside the volume. The movement between frames can be represented by a vector field $\mathbf{v}$ sampled at the centers in the previous frame. We measure the smoothness of the movement by a squared length of Laplacian $\Delta\mathbf{v}$ using the Laplace operator discretization proposed by Belkin [BSW08]:

$$E_s^{(f)} = \sum_{\mathbf{c}_i \in C} \|\Delta\mathbf{v}\left(\mathbf{c}_i^{(f-1)}\right)\|^2 = \frac{1}{|C|} \sum_{\mathbf{c}_i \in C} \| \sum_{\mathbf{c}_j \in C} H^t\left(\mathbf{c}_i^{(f-1)}, \mathbf{c}_j^{(f-1)}\right)\left(\mathbf{v}_j - \mathbf{v}_i\right)\|^2,$$

where $\mathbf{v}\left(\mathbf{c}_i^{(f-1)}\right) = \mathbf{c}_i^{(f)} - \mathbf{c}_i^{(f-1)} = \mathbf{v}_i$ is a displacement vector between the position of the center in the previous frame and its current position, and

$$H^t(\mathbf{x}, \mathbf{y}) = \frac{1}{(4\pi t)^{\frac{5}{2}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{4t}\right)$$

is a Gaussian kernel with parameter t. H^t can also be interpreted as a proximity weight capturing the notion of center neighbourhood. The closer the points $\mathbf{x}$ and $\mathbf{y}$ are, the higher the value of $H^t(\mathbf{x}, \mathbf{y})$.

To optimize this energy, we use gradient descent, i.e. in a series of steps, the centers are shifted in the direction of the sum of the gradients of the two energy terms. The procedure is terminated after a set maximum of iterations has been performed, or when the gradient is sufficiently small for each $\mathbf{c}_i^{(f)}$. However, the center positions obtained using the initial Lloyd's algorithm already represent the local optimum of the uniformity term $E_u^{(f)}$. Unfortunately, the gradient of $E_s^{(f)}$ is often not strong enough to exit this local optimum. To this end, we perform a few iterations considering only $E_s^{(f)}$ and then continue with the full energy $E^{(f)}$.

Tracking results of this method for three human mesh sequences (one TVM and two DMs) are shown in Figure 8.4. To colour the results, we use PCA projection coefficients in the first three principal directions. We normalize these coefficients and interpret them as an RGB value. Such colouring shows the center consistency, but can also serve as a way of detecting irregularly tracked centers (*irregularities*), i.e., centers that moved to parts of the volume, where they should not belong since such centers often have a different colour than their neighbours. Note that centers are represented in our visualizations as spheres of a certain radius chosen solely for clarity of results.

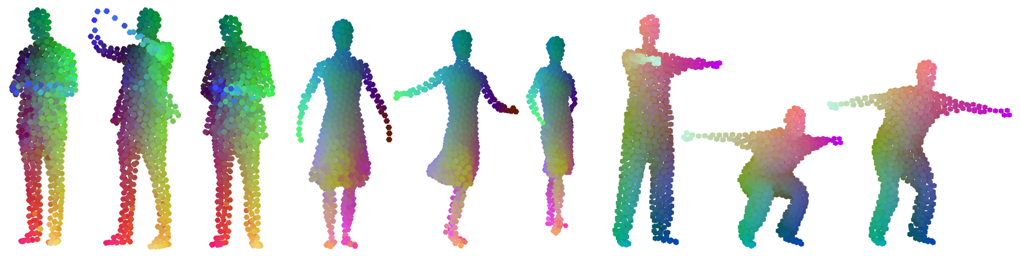


Figure 8.4: Results of the Proximity-based tracking on selected human performance datasets.

8.3.2 Limitations

Our initial work was innovative in the way of proposing the tracked centers temporal model and the objectives for obtaining the center trajectories. The algorithm we proposed in this paper is, however, quite limited.

Some limitations have been identified in the optimization process. By modelling the center movement smoothness with a vector field, the method often prefers local displacements over a global rotational movement. This can be seen in Figure 8.7b, which shows tracking results of half a revolution of a pentagonal prism. The proximity-based weights in the smoothness term ignore the topology of the volume, which means that two topologically distant parts influence each other when in near proximity (see Figure 8.5). Also, the optimization strategy of using a gradient descent turned out to be quite inefficient and required too many iterations without any guarantee of convergence.

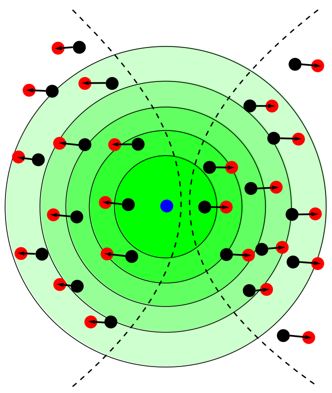


Figure 8.5: Influence of topologically distant parts in the smoothness energy term $E_s^{(f)}$ of proximity-based tracking.

Even without considering the optimization, the algorithm is considerably slow, mainly because of the uniform initialization of the centers in the volume and the correspondence estimation for each frame, which in our subsequent work, both turned out to be unnecessary. The errors in correspondence estimation were also one of the main causes of a large number of irregularities, alongside the proximity-based weights. These irregularities can be seen in Figure 8.4, mainly on the hands and legs.

8.4 As-rigid-as-possible volume tracking

In the following work, we have focused on improving the robustness of the process of obtaining the centers. Since the method also uses an *infinite-impulse-response* (IIR) filter to accumulate temporal information on center neighbourhood, we refer to this method in our experiments as IIR-based tracking.

8.4.1 Overview of the improvements

In this work, we significantly improve the quality of tracked centers, mainly by incorporating an *As-rigid-as-possible* (ARAP) movement energy term (see Section 8.4.2) and by modelling the center neighbourhood by a so-called *center affinity* (see Section 8.4.4). The ARAP energy permits a more efficient optimization strategy, which will be discussed in Section 8.4.3. We also propose two metrics, which together give a certain notion of the quality of tracking results.

We have also made some significant changes in the tracking pipeline. Firstly, we use a publicly-available implementation of *Fast winding number* [Bar+18] algorithm available in the IGL Library [JP+18] for evaluating the $IF^{(f)}(\mathbf{x})$. This way, not only the input sequence is not required to be watertight and without self-contacts, but we are also able to track mesh sequences containing non-manifold frames.

One of the most problematic parts of the original pipeline was the correspondence estimation. Combined with the initial distribution of centers, its goal was to provide an initial configuration for the optimization process. In our experiments, however, it turned out that much better results were obtained when the initial configuration was created by extrapolating the centers from the previous frame. Not only does this adjustment make the tracking more stable, but it also does not require the application of the Lloyd's algorithm to each frame – only to the first one.

8.4.2 ARAP energy

We incorporated a different coherent movement term $E_A^{(f)}$, inspired by the as-rigid-as-possible (ARAP) approach already used in various similar optimization scenarios [ACL00]. The ARAP energies ensure that points move rigidly or nearly rigidly with their neighbourhood. This energy is notably useful for representing surfaces undergoing piecewise rigid motion. We evaluate this energy as follows:

$$E_A^{(f)} = \frac{1}{2} \sum_{\mathbf{c}_i \in C} ||\mathbf{c}_i^{(f)} - \mathbf{p}_i^{(f)}||^2,$$

where $\mathbf{p}_i^{(f)}$ is a prediction of the center position obtained using a rigid transformation estimated from the movement of neighbouring centers and affinity weights from the previous frame $w^{(f-1)}$ (see Section 8.4.4):

$$\mathbf{p}_i^{(f)} = \mathcal{A}_{i|w^{(f-1)}}^{(f)} \left(\mathbf{c}_i^{(f-1)}\right) = \mathbf{R}_{i|w^{(f-1)}}^{(f)} \mathbf{c}_i^{(f-1)} + \mathbf{t}_{i|w^{(f-1)}}^{(f)}.$$

Considering center positions fixed, the rigid transformation $\mathcal{A}_{i|w}^{(f)} = \left(\mathbf{R}_{i|w}^{(f)}, \mathbf{t}_{i|w}^{(f)}\right)$ at a frame f given a certain set of weights w is found minimising

$$\left(\mathbf{R}_{i|w}^{(f)}, \mathbf{t}_{i|w}^{(f)}\right) = \underset{\mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3}{\arg\min} \sum_{w(i,j) \geq \mu} w(i,j) \left\| \mathbf{c}_j^{(f)} - \left(\mathbf{R}\mathbf{c}_j^{(f-1)} + \mathbf{t}\right) \right\|^2,$$

where $\mu = 0.001$ is a threshold parameter to speed up the computation process by considering only relevant weights. Such transformation can be found in closed form using singular-value decomposition [SR16].

8.4.3 Optimization strategy

The updated overall energy $E^{(f)} = E_A^{(f)} + \beta E_u^{(f)}$ incorporates the ARAP smoothness energy, but for uniformity energy $E_u^{(f)}$, we still use the same formulation as in Eq. 8.2.

The default value used in our experiments was $\beta = 1$. Finding an optimum in a closed form is non-trivial since both $\mathbf{p}_i^{(f)}$ and $\bar{\mathbf{x}}_i^{(f)}$ depend on the optimized center positions. However, if we consider them fixed, we can obtain the optimal center positions as follows:

$$\mathbf{c}_i^{(f)} = \frac{\mathbf{p}_i^{(f)} + \beta \bar{\mathbf{x}}_i^{(f)}}{1 + \beta}.$$

These, however, do not represent the optimum of the original energy. Nevertheless, we can now fix the $\mathbf{c}_i^{(f)}$ and recompute $\mathbf{p}_i^{(f)}$ and $\bar{\mathbf{x}}_i^{(f)}$. Similarly to Lloyd's algorithm [Llo82], we can alternate between these two steps. The optimization process is terminated when the change in $\mathbf{c}_i^{(f)}$ is sufficiently small or a fixed number of iterations has been reached.

8.4.4 Center affinity

In Section 8.4.2, we assumed that there is information available on the affinity of the tracked centers. This information is updated after each frame, and it captures the current confidence that two centers are physically connected and thus move together.

In the first frame, there is little information to work with, so we estimate the center affinity simply from the centers' proximity: using a Gaussian function, nearby centers are assigned a high affinity (up to a unit value), while for remote centers the affinity decreases to zero at a rate controlled by the σ_p parameter:

$$a_p^{(f)}(i,j) = \exp\left(-\sigma_p \cdot \left\|\mathbf{c}_i^{(f)} - \mathbf{c}_j^{(f)}\right\|^2\right).$$

In the subsequent frames, the center affinity should be estimated in a more sophisticated way, because proximity does generally not imply center affinity. Since the purpose of center affinity is to quantify whether or not the centers *move together*, we can exploit a by-product of the smoothness energy computation from the previous frame: after each step of the optimization, each center has a rigid transformation assigned, which describes the movement of the center and its neighbourhood. It seems reasonable to derive the center affinity from the dissimilarity $d_i\left(\mathcal{A}_{i|w^{(f-1)}}^{(f)}, \mathcal{A}_{j|w^{(f-1)}}^{(f)}\right)$ of their associated rigid transformations $\mathcal{A}_{i|w^{(f-1)}}^{(f)}$ and $\mathcal{A}_{j|w^{(f-1)}}^{(f)}$:

$$a_m^{(f)}(i,j) = \exp\left(-\sigma_m \cdot d_i^{(f)}\left(\mathcal{A}_{i|w^{(f-1)}}^{(f)}, \mathcal{A}_{j|w^{(f-1)}}^{(f)}\right)^2\right),$$

To allow a more intuitive control over the width of both Gaussian functions, the process is actually controlled by parameters ρ_p and ρ_m so that $\sigma_p = -\ln(0.5)/\rho_p^2$ and $\sigma_m = -\ln(0.5)/\rho_m^2$. These parameters determine at which distance the Gaussian function drops to 0.5. Values that worked well in our experiments were $\rho_p = 0.1861$ and $\rho_m = 0.00388$.

Although it is not easy to compare rigid transformations in general, feasible options have been discussed [Bot+07; Pot+06; HDV19]. The overall observation is that the transformation (dis)similarity can be meaningfully evaluated only with the knowledge of the locations that undergo the given transformation. Here, we can reuse another by-product of the previous algorithm steps: for each center, we can keep the set $V_i^{(f)}$ of voxel locations that form the discretization of the cell associated with the given center. With those locations, we can relate the transformation dissimilarity to the difference in the effect of the two transformations on the voxel center locations:

$$d_i^{(f)}(\mathcal{A},\mathcal{B}) = \frac{1}{|V_i^{(f)}|} \sum_{\mathbf{v}_k \in V_i^{(f)}} \|\mathcal{A}(\mathbf{v}_k) - \mathcal{B}(\mathbf{v}_k)\| \,.$$

A nice property of such a formulation is that after certain precomputations, it can be evaluated with complexity that is independent of the number of input voxels (see [HDV19] for details). As a result, it is computationally feasible to evaluate all pairwise center affinity values this way.

Finally, we observe that the center affinity notion must accumulate information from the previous frames in order to propagate information throughout the sequence processing. Using an IIR filter is a simple way to achieve this goal. The filtered transformation affinity $a_{\mathrm{IIR}}^{(f)}$ is computed as

$$a_{\mathrm{IIR}}^{(f)}(i,j) = \alpha a_m^{(f)}(i,j) + (1-\alpha) a_{\mathrm{IIR}}^{(f-1)}(i,j).$$

where the parameter $0 < \alpha < 1$ controls the response falloff ($\alpha = 0.01$ worked well with our data). The final weights used in Eq. 8.3 are computed as a product of proximity affinity $a_p^{(f)}(i,j)$ and filtered motion affinity $a_{\mathrm{IIR}}^{(f)}(i,j)$, since the conjunction of these two factors captures the desired notion of center affinity, i.e.

$$w^{(f)}(i,j) = a_p^{(f)}(i,j) \cdot a_{\mathrm{IIR}}^{(f)}(i,j). \tag{8.3}$$

where $w^{(f)}(i,j)$ is the affinity of the center j with the center i in the frame f. Note that the proximity affinity is not IIR filtered; this allows the algorithm to quickly react to situations when centers drift apart by reducing their final affinity.

After updating the center affinity, the algorithm moves on to process the next frame. Note that since the affinity is updated only once per frame, evaluating it for all pairs of centers is not a computational bottleneck for center counts such as 1000, as used in our experiments. If substantially more centers should be tracked, the potential performance impact could be mitigated by restricting some of the computation to pairs with high enough $a_p^{(f)}(i,j)$.

8.4.5 Volume tracking quality metrics

Quantifying the quality of the tracking result is not straightforward, as there are contradicting aspects that need to be considered. On the one hand, the result is expected to be a set of center trajectories that are smooth and simple in a certain sense; however, optimizing only for smoothness leads to a singular solution. In order to avoid this, it is necessary to also consider whether each frame has been fully and uniformly covered by the centers.

For this reason, we used a pair of statistics that captures these two aspects of volume tracking. First, *principal component analysis compactness* (PCAC) quantifies the complexity of the extracted trajectories. Each extracted center trajectory is represented by a vector of length $3n$ of concatenated x-, y-, z- coordinates. These vectors form a set of samples in a $3n$-dimensional space. Transforming them into a PCA basis provides another set of $3n$ decorrelated coordinates $\mathbf{a}_i = \left[a_i^{(0)}, a_i^{(1)}, \ldots, a_i^{(3n-1)}\right] \in \mathbb{R}^{3n}$ for each trajectory. Most of the variance will presumably be concentrated in the first few coordinates, and with the complexity of the tracked trajectories, this concentration is going to become less prominent. In general, the complexity of the trajectories is given by the complexity of the input movement itself, together with the redundant complexity caused by tracking errors.

The concentration can be quantified using the following statistic:

$$PCAC = \sum_{i=0}^{3n-1} i \sum_{j=0}^{k-1} \left|a_j^{(i)}\right|,$$

where k is the number of tracked centers. If all the variance is concentrated in the first principal component, then the value of PCAC will be zero, indicating the best possible compactness. With increasing complexity, the value of the PCAC statistic will increase, indicating the possible presence of tracking errors. Note that lower PCAC also implies less information to be stored during PCA-based dimensionality reduction if one desires to compress such data.

For each center, we may also compute the number of voxels $\left|V_i^{(f)}\right|$ that form their cell in each frame. For ideal uniform coverage, this number should be the same for all centers in each frame; however, due to pursuing other objectives as well, it often deviates. The average relative standard deviation can be used as a quantification of the *deviation from uniformity* (DFU):

$$\bar{V}^{(f)} = \frac{1}{k} \sum_{i=0}^{k-1} \left|V_i^{(f)}\right|,$$

$$DFU^{(f)} = \sqrt{\frac{1}{k} \sum_{i=0}^{k-1} \left(\left|V_i^{(f)}\right| - \bar{V}^{(f)}\right)^2}.$$

$$DFU = \frac{1}{n} \sum_{f=0}^{n-1} \frac{DFU^{(f)}}{\bar{V}^{(f)}}$$

To show the superiority of one method over another, the better one must provide better results in one criterion while also achieving better or comparable results in the other. To this end, we measured the performance of our approach and PBT with default parameters, lowering only the strength of the smoothness term of our optimized energy in cases in which the competing method provided a better DFU.

8.4.6 Experimental results

To demonstrate the contribution of our method, we compared our tracking results with the results of proximity-based tracking (PBT). To the best of our knowledge, this is the only method that we can objectively compare to since the output results are compatible. While some artificial scenarios could be considered (e.g. propagating volumetric elements using the deformation graph obtained from [Bož+21] or using the vector field of [Nie+19]), the results would be biased due to incompatible objectives.

8.4.6.1 Main experiment

The experiment was performed on three synthetic datasets of simple shapes under rigid or near rigid motion (see Figure 8.6). To simulate topological noise, we distorted the data by representing it by an implicit function (signed-distance function) sampled on a coarse grid (50 samples along the longest dimension of bounding box for *collision* dataset and 128 samples for *gears* dataset) and reconstructing it by the marching cubes algorithm [LC87]. This distortion yields merged objects when they are in contact, as can be seen for the *gears* dataset. The comparison was also done for five human performance datasets: a commercially available watertight time-varying mesh sequence *casual_man*, dynamic mesh sequence *samba* available for academic purposes [Vla+08] and three selected sequences from the *D-FAUST* dataset [Bog+17]. This dataset consists of multiple captured performances from multiple test subjects. Individual meshes are noisy, they contain holes, and some frames even have isolated points and walls of the surrounding environment. By filtering the meshes and the sampled indicator functions, we were able to obtain a volumetric representation that was robust enough to allow for consistent tracking performance. Although for the *samba* dataset, there are explicit surface correspondences between frames available, we treat the sequence as if this information was unknown. The resulting statistics are shown in Table 8.2.

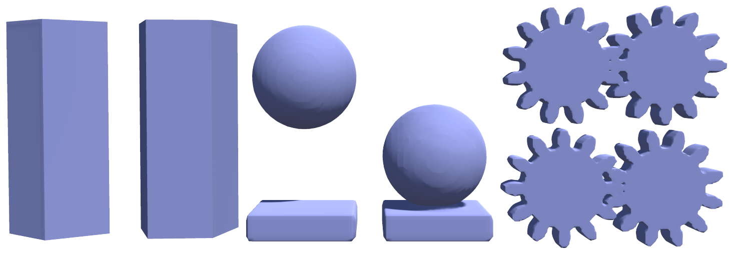


Figure 8.6: Synthetic datasets used in our experiments. From left to right: *pentagonal_prism*, a simple object undergoing rotation around the y-axis; *collision*, collision of a ball and a box; and *gears*, two gears rotating against each other.

Table 8.2: Comparison of measured quality of the volumetric tracking. Highlighted are the best results for a given dataset.

		IIR		PBT	
	no. of frames	PCAC	DFU	PCAC	DFU
pentagon_prism	50	**0.703**	**0.050**	2.076	0.091
collision	60	**1.198**	**0.074**	1.444	0.093
gears	101	**1.816**	**0.071**	2.413	0.072
casual_man	546	**14.184**	**0.154**	19.756	0.158
samba	175	**6.539**	**0.218**	9.310	0.226
DF_50020_knees	515	**6.764**	0.179	9.202	**0.094**
DF_50009_chicken_wings	212	**3.583**	0.155	4.771	**0.099**
DF_50004_jumping_jacks	360	**7.826**	0.175	10.632	**0.084**

The IIR-based method yields the best measures for all the synthetic datasets. For the *casual_man* and *samba* datasets, we had to lower the smoothness term strength parameter β to 0.6 to obtain comparable DFU values to PBT, nevertheless; we believe that the default parameter provides visually more plausible results. For the D-FAUST dataset, the IIR-based tracking yielded visually the best results for smoothness strength $\beta = 0.8$.

Examining the results visually reveals the limitations of PBT: it cannot handle global rotations (see Figure 8.7) and the contact or proximity of two parts moving in different directions (see Figure 8.8). Figure 8.9 shows the tracking results for the *casual_man* dataset, with the β parameter lowered to obtain a comparable *DFU* measure. The performance of both methods was similar for the D-FAUST dataset and we were unable to find a configuration that yielded the best results for both measures. We believe that this is due to the limited occurrence of self-contact of disjoint parts or their proximity.

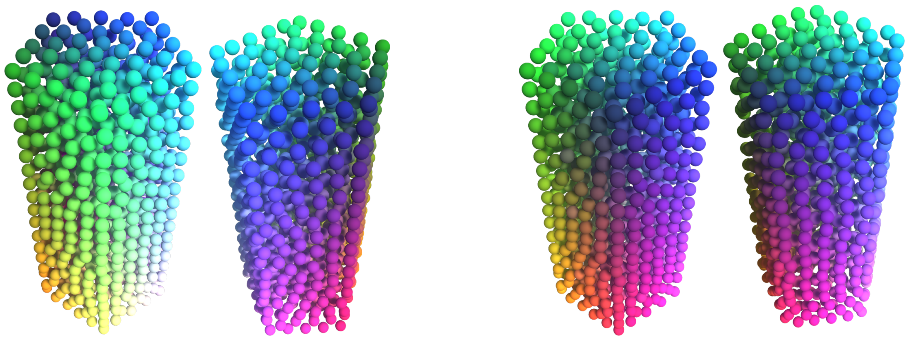

(a) IIR (b) PBT

Figure 8.7: Tracking results for the *pentagonal_prism* dataset. After half a revolution, the green centers tracked by the IIR method were correctly rotated with the corner of the prism. The movement of the centers tracked by PBT did not capture the global movement at all, and the positions of the green centers are similar to those in the initial frame.

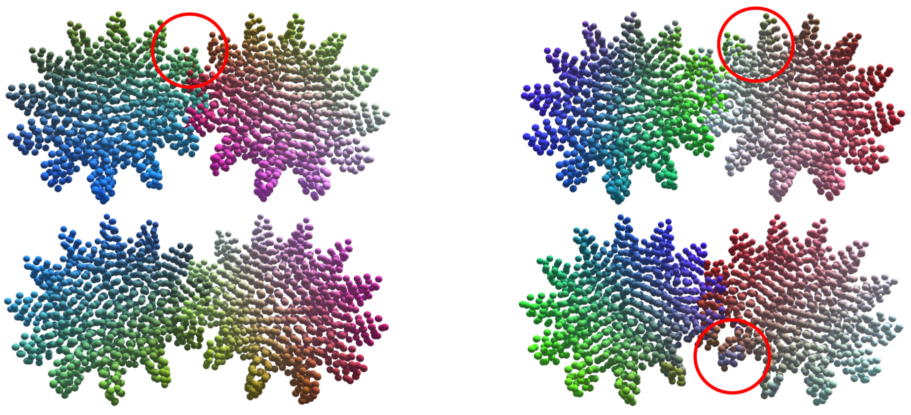

(a) IIR (b) PBT

Figure 8.8: Tracking results for the *gears* dataset; the top row shows the results for the first frame, and the bottom row shows the results for frame 81. Although some centers were incorrectly tracked by the IIR method in the first few frames (notice the inconsistencies in colouring), PBT tracks even more centers incorrectly, both centers in the first few frames and centers later in the sequence.

8.4.6.2 **Robustness**

We have also studied the robustness of both methods to various sampling artifacts. First, we measured the influence of temporal sampling. In this experiment, we have temporally subsampled the *casual_man* and *samba* datasets by selecting every second (resp. third) frame. Lowering the frame rate results in larger differences between consecutive frames. Results are shown in Table 8.3. For reference, we have also included the measures for the case, where the frames were subsampled in post-processing from the tracking results of original sequences.

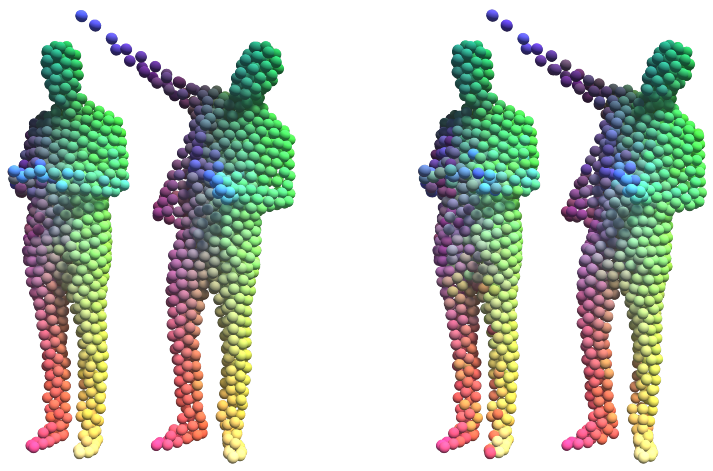


Figure 8.9: Tracking results for the *casual_man* dataset with similar *DFU* values. Notice the inconsistency in the colouring of the centers that are tracking the legs.

Table 8.3: Influence of temporal sampling on the tracking quality. Highlighted are the best values achieved between IIR-based tracking and PBT.

		IIR		PBT		Post. sub.	
	no. of frames	PCAC	DFU	PCAC	DFU	PCAC	DFU
casual_man	273	**11.777**	**0.227**	16.753	0.256	13.448	0.156
	182	**9.923**	**0.292**	14.524	0.344	12.593	0.154
samba	88	**4.965**	**0.294**	6.858	0.318	5.871	0.218
	59	**4.146**	*0.405*	5.418	**0.384**	5.295	0.218

When compared to PBT, the IIR approach achieved better PCAC values, while also achieving better or comparable DFU values. We believe that this is due to the assignment step of PBT, which is more prone to error when the difference between consecutive frames grows. Additionally, the IIR method is better suited for large movements.

Compared to the results subsampled in post-processing, the IIR approach achieved larger DFU values for the same configuration, which was expected, since this approach had to account for faster movement. Surprisingly, the PCAC values are smaller, although when compared visually, the results subsampled in post-processing are much more pleasing. This, however, merely points to a property of the PCAC metric - it measures the complexity of the movement, rather than its quality. While the low-frequency movement information might be similar, the results subsampled in post-processing capture additional higher-frequency information, which was not present in the subsampled data. Nevertheless, the PCAC values of PBT were higher than for reference data in all cases.

To demonstrate how both methods perform on noisy data, we have constructed datasets with different levels of noise from *collision* and *gears* sequences. The noise was introduced in the topological distortion step described in Section 8.4.6.1 by adding a random value $x \sim N(0, \sigma^2)$ of Gaussian distribution to each sampled SDF value, or by decreasing the grid resolution. Examples of resulting noisy data are shown in Fig. 8.10. Since we expected that the noise in sampled values would cause an increase in the difference of the neighbouring centers motions, we have increased the value of parameter $\rho_m = 0.05$, which controls the width of the Gaussian function for motion affinity. In the case of decreased grid resolution, we used the default value.

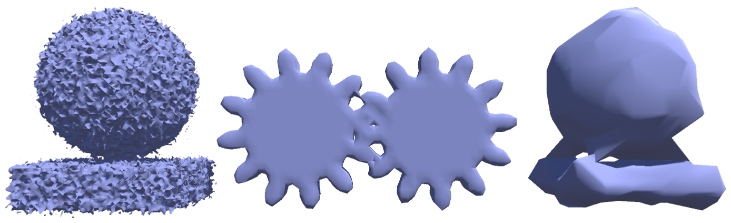


Figure 8.10: Examples of noise used in our experiments. From left to right: Distorted SDF samples, decreased marching cubes grid resolution and a combination of both.

Results for the *collision* dataset are shown in Table 8.4. For noise included only in the sampled implicit function, our method achieved better PCAC value with comparable DFU value relative to PBT. However, the tracked result contains a fast rotation of centers representing the upper ball (see Fig. 8.11). We believe that the rotation is caused by the noise interpreted as movement and by the fact that rigid sphere tracking is inherently ambiguous. The PCAC value is smaller since the rotation is captured by a few principal components, which is less penalized than the high-frequency movement present in the results of PBT. Our method performs better than PBT on data distorted by a coarser grid. Although we were unable to find a configuration that yields a better DFU value for the coarsest grid resolution, the results are visually much better, since the results of PBT involve a large number of centers moving between objects. When both distortion types were combined, we were also unable to find a configuration that yields a better DFU value. The DFU of IIR-based tracking result is higher because this model provides better rigidity and tracks the underlying objects rather than capturing the noise as much as PBT.

8.4.7 Limitations

In this work, we were able to significantly reduce the number of irregularities. However, we were still unable to fully prevent their occurrence. While we believe, that some are caused by the fact that the method only considers two subsequent frames, some of the irregularities are caused by the way how the temporal information is accumulated. Not only is there no temporal accumulation of spatial proximity, but the IIR filter in the motion affinity turned out to be inefficient in capturing the global temporal information.

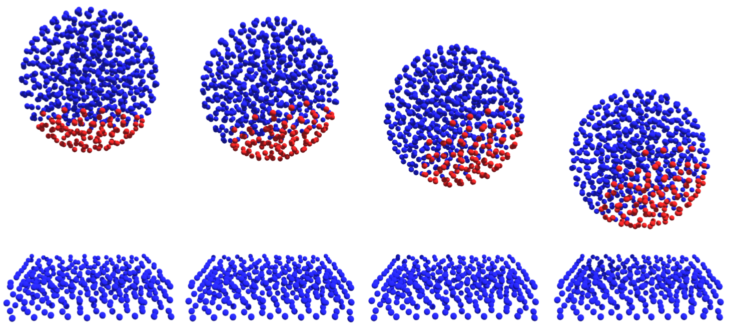


Figure 8.11: Incorrectly detected rotation induced by noise.

Table 8.4: Tracking results for the *collision* dataset distorted by noise. Highlighted are the best values achieved between the IIR-based method and PBT.

		IIR		PBT	
res.	σ	PCAC	DFU	PCAC	DFU
128	0.05	**2.265**	*0.066*	2.988	**0.063**
32	—	**1.154**	*0.054*	1.277	**0.054**
16	—	**1.248**	0.083	1.523	**0.067**
16	0.05	**4.409**	0.152	6.287	**0.105**

8.5 Maximum-distance-based affinity

In our most recent work, we have been able to further improve the quality of the frame-by-frame tracking pipeline by incorporating center affinity weights based on the maximal value encountered up to and including the currently processed frame.

Similarly to the original weight formulation in Eq. 8.3, the new weight is also computed as a product of spatial proximity and motion dissimilarity:

$$\tilde{w}^{(f)}(i,j) = \tilde{a}_p^{(f)}(i,j) \cdot \tilde{a}_m^{(f)}(i,j).$$

The difference is how the center proximity $\tilde{a}_p^{(f)}(i,j)$ and the motion dissimilarity $\tilde{a}_p^{(f)}(i,j)$ are formulated.

The previous method measured the spatial center proximity using the Euclidean distance of centers in a single frame, ignoring the information from previous frames. The main limitation of such an approach is the fact that as two topologically distant or separated parts come to near proximity, the affinity between their centers increases. Assuming the tracked sequence represents piecewise rigid objects, it could be more appropriate to use geodesic distance inside the volume, which, ideally, should be roughly constant throughout the sequence. However, due to self-contact present in real-world data, the topological information in a given frame might be incorrect, resulting in the introduction of an erroneous decrease in such a measure. Additionally, geodesic distance is computationally expensive to evaluate at the frequency required by the tracking pipeline. When examining the relative positions of a certain pair of centers in time, we observe that the Euclidean distance between them fluctuates, but it is never larger than their geodesic distance. Note that our goal is not to evaluate this quantity precisely, but to correctly differentiate between the truly connected neighbours of a center and the topologically distant centers in near proximity (see Figure 8.12). We therefore proposed to approximate this quantity by the largest Euclidean distance encountered in all frames up to and including the current frame:

$$\tilde{a}_p^{(f)}(i,j) = \exp\left(-\sigma_p \cdot \max_{0\leq l\leq f} \left\|\mathbf{c}_i^{(l)} - \mathbf{c}_j^{(l)}\right\|^2\right).$$

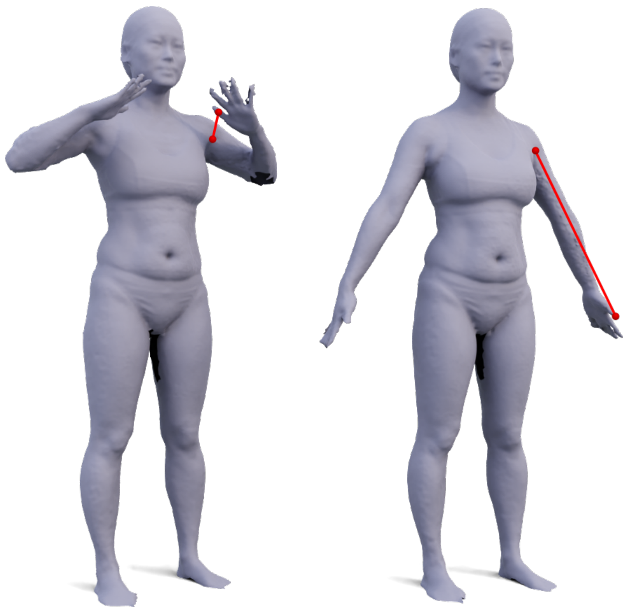


Figure 8.12: Spatial proximity in a single frame might not reflect the underlying topology of the represented object. Left: Two topologically distant points in near proximity. Right: Examining a different frame reveals that they should not be considered as neighbouring/affine.

An analogous observation can be made about the similarity of the movement. If a pair of centers moved significantly differently in the past, then they cannot both belong to the same rigid part, even when the movement has been almost identical in several previous frames. Instead of an IIR filter, we thus propose to also use the maximum dissimilarity over all already processed frames:

$$\tilde{a}_m^{(f)}(i,j) = \begin{cases} 1, & f = 0 \\ \exp\left(-\sigma_m \cdot \max_{1 \leq l \leq f} d_i^{(l)}\left(\mathcal{A}_{i|\tilde{w}^{(l-1)}}^{(l)}, \mathcal{A}_{j|\tilde{w}^{(l-1)}}^{(l)}\right)^2\right), & \text{otherwise} \end{cases} .$$

In the first frame, we have no information about the movement, therefore we assume there is no difference and instead solely rely on $\tilde{a}_p^{(0)}(i,j)$ when computing the affinity weights.

Setting Gaussian widths σ_m and σ_p (resp. ρ_m and ρ_p) to obtain satisfactory results is a task specific to the scale of the data and complexity of the motion. In our experiments, we have obtained the best results with $\rho_p = \rho_m = 0.125$ for human performance capture. For synthetic datasets, where the bounding box was significantly larger and the motion was mainly rigid, we have determined that the best results were obtained with $\rho_p = 0.2$ and $\rho_m = 0.05$.

The previous IIR-filter-based affinity depends on a falloff parameter α, which controls how the affinity reacts to occurring changes. Setting α too low results in slow reactions. On the other hand, too high α means that the affinity "forgets" the separation that occurred in the past faster. Our new formulation of affinity reacts more dynamically to changes and also captures every observed separation.

8.5.1 Experimental results

To evaluate how the previous frame-by-frame tracking pipeline benefits from the proposed maximum-distance-based affinity, we have compared the new tracking results with those reported previously using default configurations for both methods. The comparison included all the previously studied datasets (including selected sequences from D-FAUST dataset [Bog+17]) except for *pentagonal_prism* and *collision* datasets, which we believe were already tracked correctly with the previous method. The results are shown in Table 8.5.

Incorporating the newly proposed affinity results in a considerable improvement over the original affinity on all the sequences. Visually, the results contain fewer irregular centers, which can be seen when assigning each center a consistent colour, and the centers exhibit a better coverage over problematic parts (see Figure 8.13). One limitation of the maximum-distance-based affinity is its sensitivity to noise, since it may cause two affine centers to be treated as separated if the data contains erroneous movement even just for a single frame. In such cases, we advise using the original IIR-based affinity or computing the affinities from l-th largest encountered value instead of the maximum.

Table 8.5: Comparison of frame-by-frame-tracked results using proposed and original (IIR) affinity. Highlighted are the best results for a given dataset.

	n	Proposed		IIR	
		PCAC	DFU	PCAC	DFU
gears	60	**1.785**	**0.070**	1.816	0.071
casual_man	545	**10.587**	**0.127**	12.289	0.206
samba	175	**5.060**	**0.215**	5.547	0.262
DF_50020_knees	515	**5.117**	**0.111**	6.764	0.179
DF_50009_chicken_wings	212	**2.855**	**0.116**	3.583	0.155
DF_50004_jumping_jacks	360	**6.040**	**0.130**	7.826	0.175

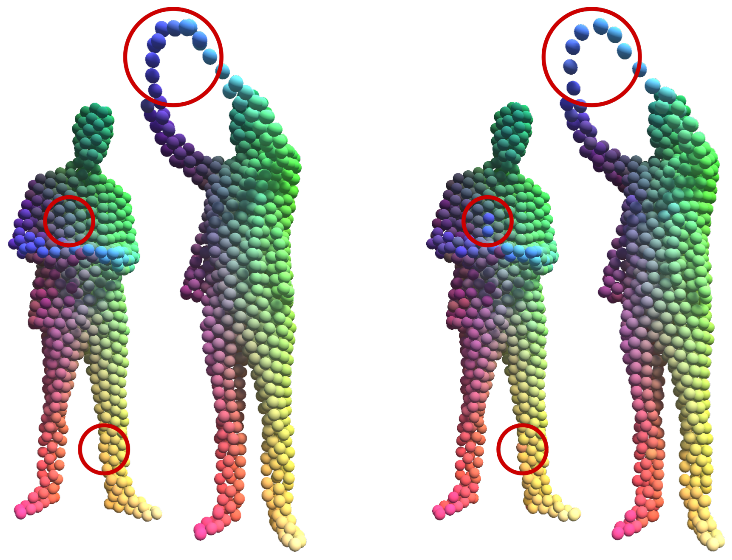

(a) Proposed (b) IIR

Figure 8.13: Tracking results for *casual_man* dataset.

8.6 Tracking improvement with global optimization

One of the drawbacks of the previous tracking pipeline is that it is a frame-by-frame procedure, gathering information about the nature of the objects captured in the data in chronological order. This approach prevents the information from frames that appear later in the sequence from influencing the tracking results (and the induced correspondence information) of preceding frames. This leads to certain artifacts in the tracking results, which in turn hinder the application of the tracking in scenarios that are sensitive to tracking errors, such as compression or time-consistent editing.

Optimizing the centers globally, however, depends strongly on initialization. To this end, we propose to first obtain tracked centers using the frame-by-frame procedure, followed by a post-processing, in which the results are improved using a global optimization. Adjusting the trajectories of irregular centers to revert tracking errors is quite a difficult task and might also lead to the introduction of additional irregularities into tracking results. We instead propose to detect and filter such centers out.

8.6.1 Irregular center detection

Once the frame-by-frame tracking is finished (regardless of the affinity weights used), we can analyse the achieved results and detect the *irregular centers*. To this end, we evaluate an irregularity measure $I_i = \min_j \left\|\mathbf{c}_i - \mathbf{c}_j\right\|_2^2$, where $\mathbf{c}_i$ is the center trajectory and $\|\cdot\|_2^2$ is the squared Euclidean norm. If a center is correctly tracked, there should exist another center with a similar trajectory in near proximity. Since an irregular center changes suddenly its relative position to its neighbouring centers, even the distance to the closest center to its trajectory is expected to be higher than for the correctly tracked centers (see Figure 8.14).

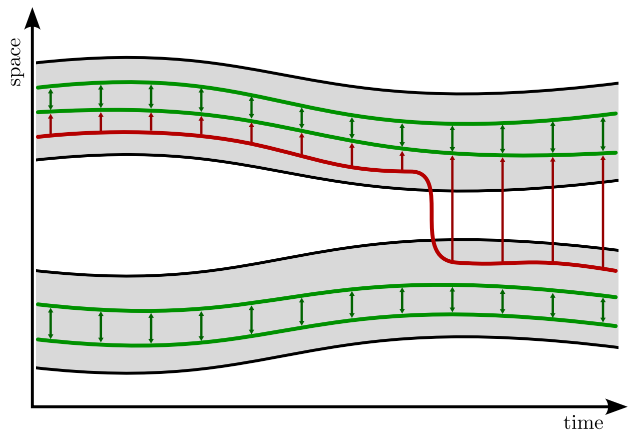


Figure 8.14: Irregular center detection using distance to closest trajectory. Arrows indicate distances that contributed to the computation. Red trajectory has a much higher I_i and is correctly detected as irregular.

The value of I_i must be considered in the context of the values of all centers, as it depends on various factors, e.g., center count, the scale of the data and the dynamics of the movement. We can also use this measure to quantify the success of tracking in terms of the presence of irregular centers, by sorting all the values in descending order and plotting them as a curve. By comparing the curves resulting from different tracking methods, we can determine which results are less affected by the presence of irregular centers (see Figure 8.15), as long as the results were tracked in the same input sequence and the center count is similar (although not necessarily equal). When attempting to improve the tracking results, one of our goals is to narrow or eliminate the part of the curve with I_i significantly higher than the correctly tracked centers, while not significantly increasing the irregularity of such centers.

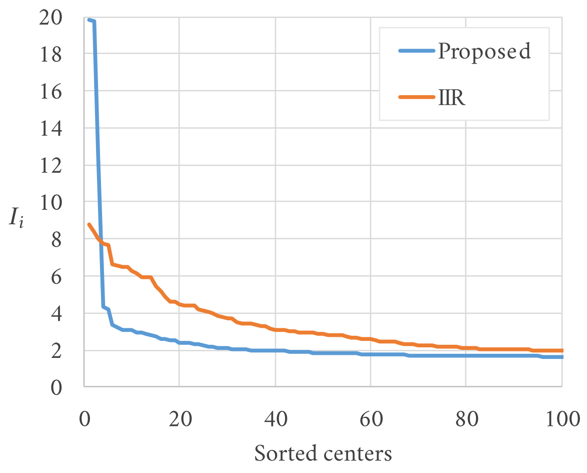


Figure 8.15: Comparison of largest 100 values of I_i for volume tracking results obtained using IIR-based affinity and max-based affinity on *casual_man* dataset.

8.6.2 Global optimization

Simply removing a certain number of centers with the highest I_i actually does not lead to an improvement in terms of flattening the irregularity curve. The uniformity of the centers distribution is violated since removed centers leave an uncovered volume. Re-running the frame-by-frame tracking with the irregular centers removed might still not prevent new irregular centers from appearing, even when the final affinity weights obtained in the initial tracking are utilised. Instead, we propose to follow the irregular center removal with adjustment of the remaining tracked trajectories of centers in a global optimization process.

The objectives of the global optimization are identical to the frame-by-frame tracking. We optimize a global energy $\hat{E}$ consisting of uniformity and motion smoothness energy terms $\hat{E} = \hat{E}_s + \hat{\beta}\hat{E}_u$.

The uniformity term is the same as in the frame-by-frame tracking, except that it is evaluated for all the frames in the sequence at once:

$$\hat{E}_u = \frac{1}{2}\sum_{f=0}^{n-1} E_u^{(f)} = \frac{1}{2}\sum_{\mathbf{c}_i \in C}\sum_{f=0}^{n-1} \|\mathbf{c}_i^{(f)} - \bar{\mathbf{x}}_i^{(f)}\|^2.$$

If we consider the centroids $\bar{\mathbf{x}}_i^{(f)}$ fixed, we can approximate the gradient by these partial derivatives:

$$\frac{\partial \hat{E}_u}{\partial \mathbf{c}_i^{(f)}} \approx \mathbf{c}_i^{(f)} - \bar{\mathbf{x}}_i^{(f)}.$$

The global smoothness energy is evaluated as

$$\begin{aligned}
\hat{E}_s &= \frac{1}{2}\left(\sum_{\mathbf{c}_i \in C}\sum_{f=1}^{n-1}\left\|\mathbf{c}_i^{(f)} - \mathbf{p}_i^{(f)}\right\|^2 + \sum_{\mathbf{c}_i \in C}\sum_{f=0}^{n-2}\left\|\mathbf{c}_i^{(f)} - \mathbf{q}_i^{(f)}\right\|^2\right),\\
\mathbf{p}_i^{(f)} &= \mathcal{A}_{i|\omega}^{(f)}\left(\mathbf{c}_i^{(f-1)}\right),\\
\mathbf{q}_i^{(f)} &= \mathcal{A}_{i|\omega}^{(f+1)^{-1}}\left(\mathbf{c}_i^{(f+1)}\right),
\end{aligned}$$

where $\mathbf{p}_i^{(f)}$ and $\mathbf{q}_i^{(f)}$ are forward and backward rigid motion predictions of the center position at frame f, using rigid transformations estimated given overall movement-based affinity weights ω (see Eq. 8.4). Considering such predictions fixed, the partial derivatives, which form the approximated gradient, are as follows:

$$\frac{\partial \hat{E}_s}{\partial \mathbf{c}_i^{(f)}} \approx \begin{cases} \mathbf{c}_i^{(f)} - \mathbf{q}_i^{(f)}, & f = 0\\ \mathbf{c}_i^{(f)} - \mathbf{p}_i^{(f)} - \mathbf{q}_i^{(f)}, & 1 \leq f \leq n-2\\ \mathbf{c}_i^{(f)} - \mathbf{p}_i^{(f)}, & f = n-1 \end{cases}$$

The optimization process is iterative, working with a set of trajectories C, whose initial values are given by the original tracking results with irregular centers removed. In each iteration, we evaluate the energy $\hat{E}(C)$ and the approximated gradient $\nabla\hat{E}$, and construct a candidate set of trajectories $\bar{C}$, where the center positions are calculated as

$$\bar{\mathbf{c}}_i^{(f)} = \mathbf{c}_i^{(f)} - \lambda\left(\frac{\partial \hat{E}_s}{\partial \mathbf{c}_i^{(f)}} + \hat{\beta}\frac{\partial \hat{E}_u}{\partial \mathbf{c}_i^{(f)}}\right).$$

First, lambda is set to $\lambda = 0.1$ and then it is iteratively scaled by $\frac{1}{2}$ until $\hat{E}(\bar{C})$ is smaller than $\hat{E}(C)$, or a specified number of attempts has been reached. If an improvement in terms of energy is achieved, we set $C = \bar{C}$ and continue to the next iteration. Otherwise, the process is terminated and C is the resulting set of trajectories. The optimization process can also be terminated after a specified number of iterations (20 in our experiments).

Such an optimization strategy does not necessarily converge to a global optimum. If an irregular center was left in the initial set C, the local steps in the gradient direction will not straighten its trajectory in order to eliminate the transition between disconnected components. The locality of the changes is, however, also an advantage, since the local trajectory adjustments ensure that the objectives are met, while not introducing any large sudden changes, and therefore no new irregular centers can appear.

8.6.3 Global affinity based on movement

The affinity utilised in the global optimization process directly considers only the dissimilarity of motion:

$$\omega(i,j) = \exp\left(-\sigma_{\text{gm}} \cdot \max_{0 \le f < n} d_i^{(f)} \left(\mathcal{A}_{i|\omega_{\max}}^{(f)}, \mathcal{A}_{j|\omega_{\max}}^{(f)}\right)^2\right), \tag{8.4}$$

where σ_{gm} is a parameter controlling the tolerance of the affinity to dissimilar motions. The overall spatial proximity $\omega_{\max}(i,j)$ of centers is also considered, but only for estimating the transformations $\mathcal{A}_{i|\omega_{\max}}^{(f)}$:

$$\omega_{\max}(i,j) = \exp\left(-\sigma_{\text{gp}} \cdot \max_{0 \le f < n} \left\|\mathbf{c}_i^{(f)} - \mathbf{c}_j^{(f)}\right\|^2\right).$$

The motivation to mainly rely on the motion information instead of using both motion and proximity combined is the following: If a center belongs to a large rigidly moving part of the object, we want it to be influenced by the whole part rather than only a certain small neighbourhood around it. Having all the positions in each frame at hand, we can confidently rely solely on this information without worrying about the rigid part suddenly splitting in a later frame (see Figure 8.16).

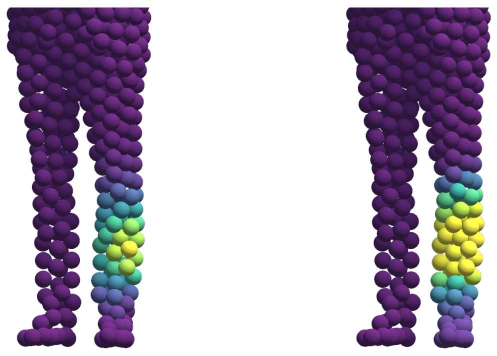

(a) Combined (b) Motion only

Figure 8.16: Affinity for global optimization. Combined spatial and movement information (a) leads to a fairly small center neighbourhood. The global affinity considering motion only (b) allows working with large rigidly moving parts as a whole.

8.6.4 Irregular center removal strategy

Removing a large number of detected irregular centers before the global optimization might result in large volume areas not covered by any center, so the optimization process might struggle to appropriately cover them. In our experiments, we have observed that it is more appropriate to iterate over the irregular center removal and optimization, with only a small number of centers removed in each iteration.

The user can specify, how many iterations will be performed and the number of the most irregular centers that will be removed in each iteration.

8.6.5 Experimental results

We first evaluate our post-processing on synthetic data for which the ground truth tracked centers trajectories are known. We show the benefits of running the global optimisation even without removing any center. This experiment also highlights additional limitations of PCAC and DFU metrics. Then, we also study the influence of various irregular center removal strategies.

8.6.5.1 Global optimisation without center removal

Evaluating tracking quality solely based on PCAC and DFU is limited since the optimum of both does not lie at zero (at least for a discretized volume in the case of DFU). To this end, we have created a synthetic dataset, for which, given the center positions in the first frame, we are able to obtain ground truth optimal center trajectories. The dataset consists of two rigidly moving objects (a unit cube and a unit tetrahedron) that first come in self-contact and then separate after a certain number of frames. The trajectories are obtained by assigning each center to the corresponding object and propagating it using rigid transformations determined from the motion of the vertices of the cube/tetrahedron. Having the ground truth trajectories C_{gt} and a tracking result C, $\mathbf{c}_{\mathrm{gt}_i}^{(0)} = \mathbf{c}_i^{(0)}$ for all i, we can now evaluate the tracking quality of C using mean-squared error of positions and mean-squared motion error:

$$MSE = \frac{1}{k(n-1)} \sum_{f=1}^{n-1} \sum_{i=0}^{k-1} \left\| \mathbf{c}_{\mathrm{gt}_i}^{(f)} - \mathbf{c}_i^{(f)} \right\|^2,$$

$$MSME = \frac{1}{k(n-1)} \sum_{f=1}^{n-1} \sum_{i=0}^{k-1} d_i^{(f)} \left(\mathcal{A}_{\mathrm{gt}_i}^{(f)}, \mathcal{A}_i^{(f)} \right)^2,$$

where $d_i^{(f)}$ is evaluated on voxel positions of Voronoi cells corresponding to ground truth center positions.

In this experiment, we have compared the tracking results obtained by the frame-by-frame tracking using IIR and the proposed affinity with their respective default configurations and both such results post-processed using the global optimization without removing any center. In the global optimization, we have set $\beta = 0.1$, since the uniformity of distribution was already close to the ground truth value and $\rho_{\mathrm{gm}} = 0.005$. The results are shown in Table 8.6.

The results demonstrate, that the frame-by-frame tracking with our proposed affinity already achieved satisfying results in terms of DFU and MSE. Nevertheless, we were able to achieve an improvement using global optimization in terms of PCAC and estimated transformations. Both frame-by-frame tracking and global optimization approaches using our proposed affinity achieved DFU lower than the ground-truth value. This implies that minimizing such a measure is not always desirable. IIR-based frame-by-frame tracking benefited the most from global optimization, however, it did not achieve the best results.

Table 8.6: Quantitative comparison of various tracking results and ground truth center trajectories. Highlighted are the best results.

	PCAC	DFU	MSE	MSME
Ground truth	1.078	5.65E-02	—	—
Proposed	1.492	**5.62E-02**	**1.93E-04**	5.521E-04
Proposed + global	**1.236**	5.60E-02	1.94E-04	**5.519E-04**
IIR	1.511	5.77E-02	2.08E-04	5.522E-04
IIR + global	1.249	5.70E-02	2.60E-04	5.5194E-04

8.6.5.2 Irregular center removal

In this experiment, we have studied the effects of various strategies for removing 20 irregular centers from tracking results obtained through frame-by-frame tracking with our proposed affinity on the *casual_man* dataset from Section 8.5.1. The strategies differed in the number of global optimizations performed n_{go} and in the number of removed centers in each optimization n_{rem}. The parameters of the global optimization were as follows: $\rho_{\text{gp}} = 0.125$, $\rho_{\text{gm}} = 0.03$ and $\hat{\beta} = 0.5$. Note that these values were selected empirically and slightly different values yield similar results. Table 8.7 shows the measured PCAC and DFU values and the irregularity curves are shown in Fig. 8.17. For comparison, we also include results for center removal without global optimization.

Table 8.7: Comparison of PCAC and DFU measures for various irregular center removal strategies on *casual_man* dataset frame-by-frame tracked using our proposed affinity.

n_{rem}	20	20	4	1
n_{go}	—	1	5	20
PCAC	10.154	8.760	**8.578**	8.869
DFU	0.160	0.149	0.144	**0.134**

With growing n_{go}, the improvement process achieves better coverage of volume, which is reflected in the DFU measure. However, we can also see a negative trend in terms of irregularity. Best values of PCAC were achieved with $n_{\text{go}} = 5$. This is also reflected in a visual inspection of results (see Fig. 8.18). Fig. 8.18a shows centers that were detected as irregular. It can be seen that the detected centers indeed travel across different body parts. The increased irregularity with growing n_{go} is reflected by certain number of centers oscillating to cover a larger volume, which is unfortunately visible only when the tracked centers are animated.

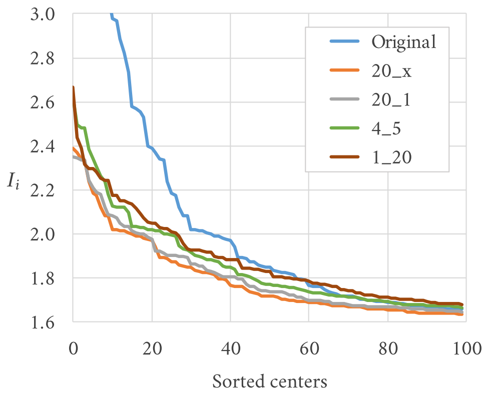
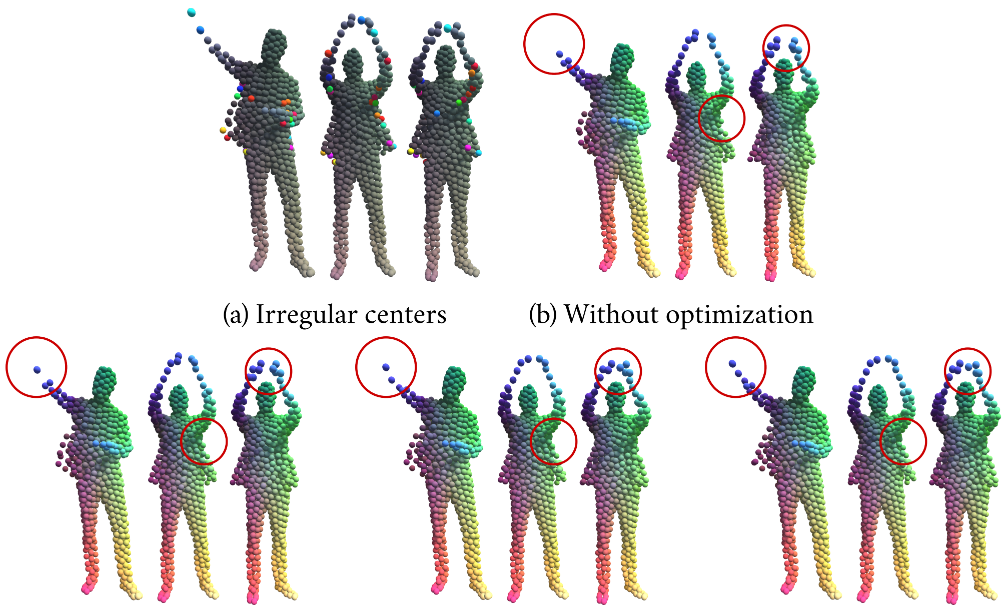


Figure 8.17: Comparison of first 100 I_i after center removal.



(a) Irregular centers (b) Without optimization

(c) $n_{\text{rem}} = 20, n_{\text{go}} = 1$ (d) $n_{\text{rem}} = 4, n_{\text{go}} = 5$ (e) $n_{\text{rem}} = 1, n_{\text{go}} = 20$

Figure 8.18: Results for various irregular center removal strategies for *casual_man* dataset with target of 20 centers to be removed. Highlighted are the areas with the most notable differences.

8.7 Applications

In this section, we discuss some examples of how the tracked centers could be utilized in various scenarios in mesh sequence processing. Although these applications are agnostic of the pipeline used to obtain the center trajectories, the usability of the model is directly connected to the tracking quality. Unless otherwise stated in the description of the individual application, we assume the tracking results to contain as few tracking irregularities as possible and to achieve regular coverage of the volume enclosed by the individual frames.

8.7.1 Surface approximation

Like many other temporal models (e.g., a tracked surface), the tracked centers can be used as a reduced representation of the dynamic surface they capture. There are various possibilities for how this could be achieved. We experimented in our initial work [DVV21] with an approach based on implicit surface representation, which represents a surface as an iso-surface $g(\mathbf{x}) = 0$ of an implicit function $g : \mathbb{R}^3 \rightarrow \mathbb{R}$. One of the most convenient implicit functions is the *signed-distance function* (SDF), which returns the distance of a point to the surface, multiplied by -1 if the point lies inside of the volume enclosed by the surface. The convenience stems from the ability to efficiently perform boolean operations over SDF-represented surfaces using min and max functions.

When examining the tracking results of our initial tracking pipeline with centers represented as spheres, we noticed that the silhouette of tracked results mostly resembled the silhouette of the original frames. This gave us an idea to approximate the original surface with sphere SDF primitives. SDF of a sphere $S = (\mathbf{c}, r)$ with center $\mathbf{c}$ and a radius r can be evaluated as

$$g_S(\mathbf{x}) = \|\mathbf{c} - \mathbf{x}\| - r.$$

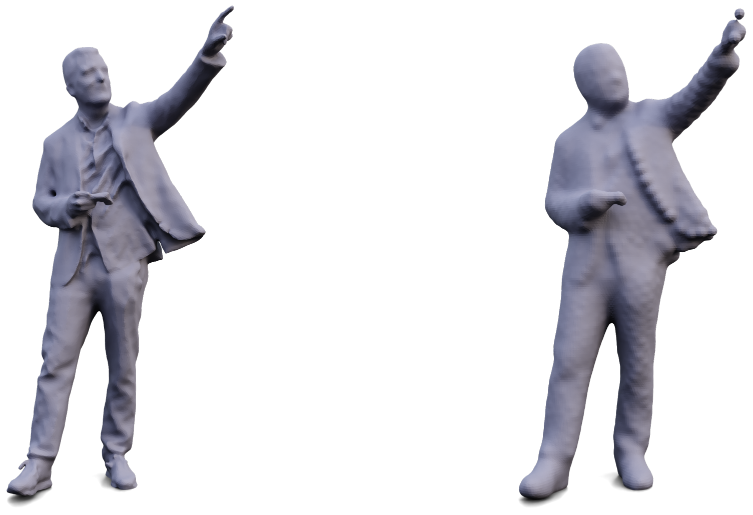
Instead of a trivial union of the spheres, which would cause sharp creases in places where the sphere primitives intersect, we compute the resulting implicit function $g^{(f)}(\mathbf{x})$ approximating a frame $\mathcal{M}_f$ using a smooth blending of these primitives:

$$g^{(f)}(\mathbf{x}) = -\frac{1}{\gamma} \log_2 \left(\sum_{\mathbf{c}_i \in C} \exp2 \left(-\gamma \cdot \left(\|\mathbf{c}_i^{(f)} - \mathbf{x}\| - r \right) \right) \right), \tag{8.5}$$

where γ is a smoothing parameter. Note that we use a single value of radius for all the centers, but it can be fine-tuned for each center separately. Additionally, a different SDF primitive could be utilized. For example, one could assign each center an ellipsoid skewed to reflect the shape of the corresponding Voronoi cell in each frame. The shape of the primitive could also be derived considering all the frames, however, such a shape would have to be rotated in each frame given the estimated rigid motion of centers.

A simple TVM compression method that does not preserve the structure of the frames can be designed using this formulation. Considering only sphere primitives of fixed radius, one can compress the TVM by encoding the center positions (e.g., using the COBRA algorithm [VS09]) and parameters γ and r. The surface can then be reconstructed using surface extraction algorithms such as marching cubes [LC87] or dual contouring [Ju+02]. Alternatively, the surface can be rendered directly using the raymarching algorithm [Har96].

Figure 8.19 shows an example surface approximation of a single frame from the *casual_man* dataset using 4000 centers tracked by a frame-by-frame algorithm with maximum-based affinity. The parameters γ and r were set empirically. This visualization reveals the main limitation of such a representation: lack of surface detail. For it to be preserved, a different representation, e.g., using centers as a deformation model (see the following section) for propagating a single frame, would have to be used. Such an approach is, however, much more sensitive to tracking irregularities. The surface approximation using Eq. 8.5, on the other hand, does not require perfect results, only a decent coverage of the volume and a consistent movement in terms of compressibility of centers, i.e., there can be any amount of irregularities, as long as the PCAC of such tracking result is reasonably low.



(a) Original (b) Approximation

Figure 8.19: Example surface approximation of a single frame selected from the *casual_man* dataset.

8.7.2 Deformation model

Given a certain frame $\mathcal{M}_f$ and the positions of the centers in such a frame, it is possible, similarly to skeletal animation, to deform such a frame by first moving the centers and then distributing this motion to the surface.

There are various ways of formulating such a deformation model, the simplest one being to assign each vertex of the frame a transformation of the closest center. This, however, creates blocking artifacts at the boundaries of regions influenced by different centers. A more sophisticated approach can be inspired by *embedded deformations* [SSP07]:

$$\tilde{\mathbf{v}}_i^{(f)} = \sum_{j \in N_k(v_i)} w_j\left(\mathbf{v}_i^{(f)}\right)\left[\mathbf{R}_j\left(\mathbf{v}_i^{(f)} - \mathbf{c}_j^{(f)}\right) + \mathbf{c}_j^{(f)} + \mathbf{t}_j\right], \tag{8.6}$$

where $N_k(v_i)$ is the set of k nearest centers to vertex v_i, the rotation matrix $\mathbf{R}_j$ and the translation vector $\mathbf{t}_j$ represent the input rigid motion of the center $\mathbf{c}_j^{(f)}$ and $w_j\left(\mathbf{v}_i^{(f)}\right)$ is a deformation weight calculated as

$$w_j\left(\mathbf{v}_i^{(f)}\right) = \left(1 - \|\mathbf{v}_i^{(f)} - \mathbf{c}_j^{(f)}\|/d_{\max}\right)^2, \tag{8.7}$$

where $d_{\max}$ is the distance to the $k+1$ nearest center. Example results of the centers being applied as a deformation model are shown in Figure 8.20.

8.7.3 Feature vectors

For a frame $\mathcal{M}_e$ and its corresponding center configuration, we can construct a function $\mathbf{f}^{(e)}(\mathbf{x})$ that assigns a certain point $\mathbf{x}$ a feature vector based on the relative position of the point to the individual centers. Assuming the relative positions of the centers to the surface in close proximity should not significantly change throughout the sequence, it might be possible to formulate $\mathbf{f}^{(e)}(\mathbf{x})$ so that feature vectors of corresponding points in different frames have fairly similar values. Having such feature vectors should be beneficial in various mesh sequence processing tasks, for example, in surface correspondence estimation (see Section 8.7.6) or in attribute mapping (see Section 8.7.7).

One example formulation that should achieve such a property assigns each point a vector with components evaluated as

$$f_i^{(e)}(\mathbf{x}) = \exp\left(-\sigma\|\mathbf{x} - \mathbf{c}_i^{(e)}\|^2\right),$$

where $\mathbf{c}_i$ is the position of the center corresponding to such a component and σ is a certain parameter controlling the falloff radius. The main limitation of such formulation is the high dimension of resulting feature vectors.

If one desires control over the dimension of feature vectors, it is possible to formulate $\mathbf{f}(\mathbf{x})$ using PCA over center trajectories:

$$\mathbf{f}^{(e)}(\mathbf{x}) = \sum_{j \in N_k(\mathbf{x})} w_j(\mathbf{x})\, \mathbf{a}_j,$$

where N_k is the set of k nearest centers to $\mathbf{x}$, $w_j(\mathbf{x})$ are weights defined in Eq. 8.7, and $\mathbf{a}_j = \left[a_j^{(0)}, a_j^{(1)}, \dots, a_j^{(l-1)}\right] \in \mathbb{R}^l$ is the vector of first l decorrelated coordinates of the trajectory of the center $\mathbf{c}_j$.

8.7.4 Temporally coherent editing

Static mesh editing is one of the most researched areas in the field of mesh processing. The attention it received is mainly due to its ability to support the creative process in design and animation. For more details about mesh editing of a single shape, we refer the reader to the survey of Yuan et al. [Yua+21].

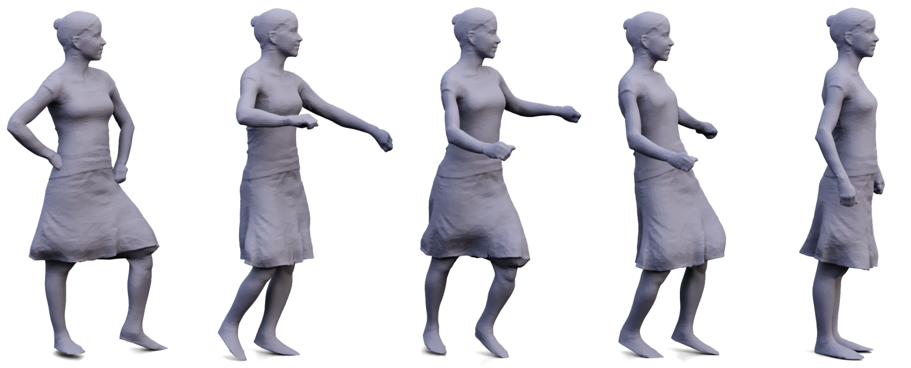
In some cases, it is desirable to edit mesh sequences. Applying static mesh editing techniques on individual frames is, however, difficult since one cannot automatically propagate the edits throughout the sequence and any editing discrepancy between subsequent frames is easily spotable by an observer [VS11]. There are already a few methods for temporally coherent editing of dynamic meshes [KG06; CH12; BSG12], however, for TVMs, to the best of our knowledge, there is only a single method, which is also only limited to detail enhancement [YXF14].

A proof-of-concept method for temporally coherent editing based on our temporal model was presented in the master's thesis of Zuzana Káčereková [Káč23]. The method consists of four steps. First, the user selects and moves a center $\mathbf{c}_i^{(f)}$ (denoted effector) in a frame f of choice. The motion of $\mathbf{c}_i^{(f)}$ is then distributed to other centers in the given frame by applying a displacement of the effector weighted by a Gaussian over spatial proximity. These displacements are distributed into the center positions of the rest of the frames by rotating them given the estimated rigid motion of the centers. Finally, the motion of the centers is used to deform the surface in each frame. To compute the resulting position of a vertex $\bar{\mathbf{v}}_i^{(e)}$, the method uses a simple formula similar to Eq. 8.6, however, considering only the displacement motion:

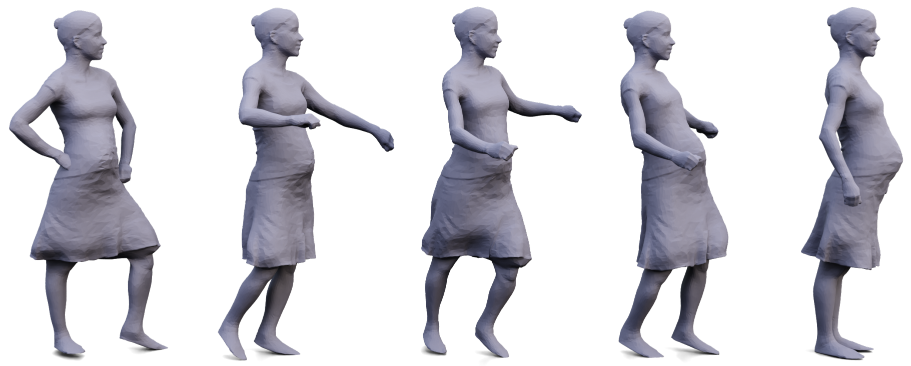
$$\bar{\mathbf{v}}_i^{(e)} = \mathbf{v}_i^{(e)} + \sum_{j \in N_k(v_i)} w_j \mathbf{t}_j^{(e)}, \tag{8.8}$$

where $\mathbf{t}_j^{(e)}$ is the displacement vector of center $\mathbf{c}_j^{(e)}$.

Example results are shown in Figure 8.20. The method is quite limited, but it was only designed to serve as a baseline for our further efforts. Since then, our team has been able to achieve some improvements, but the work is still in progress.



(a) Original



(b) Edited

Figure 8.20: Example results of proof-of-concept method for temporally coherent editing [Káč23]. Results are courtesy of Zuzana Káčereková.

8.7.5 Structure-preserving TVM compression

A model-based structure-preserving TVM compression method following the pipeline shown in Figure 4.3 on page 33 can be built around the tracked centers. It first constructs the temporal model considering the whole sequence and encodes it using, for example, the COBRA algorithm [VS09]. Then, the first frame is encoded by an intra-only method, such as weighted parallelogram [VB13].

Each subsequent frame $\mathcal{M}_i$ is then encoded as follows: The model is used to obtain a reference shape $\mathcal{R}_i$ (a mesh or point cloud) which should be fairly aligned with the coded frame. The geometry coding approach proposed by Yamasaki et al. [YA10] is then used. Each vertex of $\mathcal{M}_i$ is assigned to the closest vertex (resp. point) of $\mathcal{R}_i$. For each vertex (resp. point) of $\mathcal{R}_i$, the number of corresponding coded vertices is encoded, followed by the matching correction vectors. The connectivity of the frame can be encoded by our priority-based approach, which will be presented in Chapter 9.

The most crucial part of the method is the process of obtaining the reference shape $\mathcal{R}_i$. We are currently considering two approaches to address this, each having its own challenges and limitations. One such approach is using the centers as a deformation model to deform the previous reference frame $\bar{\mathcal{M}}_{\text{ref}}$ given the motion of centers between the frames. Unfortunately, in our preliminary experiments, such an approach did not achieve satisfactory results, regardless of the formula used to propagate the motion from centers to the surface. We believe this is due to the deformation not guaranteeing the alignment between the $\mathcal{R}_i$ and $\mathcal{M}_i$ since the process of obtaining tracked centers does not take into account that they should serve for such a purpose. Incorrect alignment causes an increase in the length of correction vectors, which implies a larger entropy of the encoded values.

The second approach obtains $\mathcal{R}_i$ by extracting the iso-surface from the surface approximation presented in Section 8.7.1. However, the $\mathcal{R}_i$ obtained this way might have a different vertex density than $\mathcal{M}_i$, which may also lead to inefficient geometry coding - having a too small number of vertices relative to the coded frame, the length of the correction vectors is expected to increase; having too many vertices means that the method must indicate for all the redundant vertices that they are not used to predict any coded vertex. This even applies to local vertex density - if the coded frame was adaptively sampled so that parts of higher detail have a higher vertex density, the reference shape should reflect this as well, thus we cannot simply uniformly sample $|V_i|$ points on $\mathcal{R}_i$. Note that this is not expected to be an issue in the case of the deformation model, since we expect that all the frames were obtained using an identical technique, thus two subsequent frames should have a relatively (even locally) similar vertex density.

We have decided to prefer the deformation-model-based approach over the surface approximation and to attempt to address its limitations in the future. The reason is that solving its alignment problem implies additional applications of the tracked centers model (see Section 8.7.6). To address this, we first plan to select an appropriate mechanism for the deformation model, and then to incorporate it in an additional energy term in the optimization of center positions, which measures the alignment of consecutive frames using the deformation model. As a consequence, to this date, we still were not able to propose an efficient general and structure-preserving compression method.

8.7.6 Surface correspondences

The tracked centers can be interpreted as sparse volume correspondences between the frames. We believe that if the deformation model alignment limitation presented in the previous section is addressed, it would be possible to use them to estimate the temporally consistent surface correspondence information. Temporally consistent correspondences are such that contain correspondence information from any frame to all the remaining frames, where the corresponding surface point exists. Obtaining such information is a fairly different problem from pairwise shape matching.

This information could be derived using the deformation model, similarly to the approach proposed by Eisenberger et al. [ELC19]: The frames are aligned given the motion of centers, as in structure-preserving TVM compression. Then, the correspondences are estimated from the spatial proximity of vertices. Alternatively, this information could be derived by matching the temporally coherent feature vectors $\mathbf{f}^{(e)}(\mathbf{x})$ between the frames. The latter approach would also benefit from considering the deformation model during optimization since it would make the relative positions of centers to the surface in near proximity much more temporally stable.

8.7.7 Attribute mapping on mesh sequences

Given an input mesh sequence and an attribute function $a^{(i)} : \mathcal{M}_i \to \mathbb{R}$, which assigns all the points on the surface of a particular frame $\mathcal{M}_i$ an attribute value, the goal of attribute mapping on mesh sequences is to generalize the function $a^{(i)}$ to all the frames, so that corresponding points are assigned identical values.

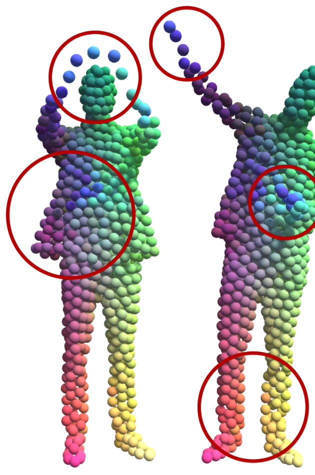
Although we have not yet verified this, such a problem can be addressed using the tracked centers. First, a set of points $P = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ is sampled from the surface of $\mathcal{M}_i$. For each of the sampled points $\mathbf{p}_j$, an attribute value $a_j = a^{(i)}(\mathbf{p}_j)$ and a feature vector $\mathbf{f}^{(i)}\left(\mathbf{p}_j\right)$ (see Section 8.7.3) are obtained. To generalize the attribute map to all the frames, it is possible to estimate a function $a : \mathbb{R}^l \to \mathbb{R}$, where l is the dimension of input feature vectors so that $a\left(\mathbf{f}^{(i)}\left(\mathbf{p}_j\right)\right) \approx a_j$ for any of the sampled points. This can be done, for example, by assigning an input vector the attribute value of the nearest sampled feature vector, or by regression (e.g., using radial-basis functions or a regression neural network).

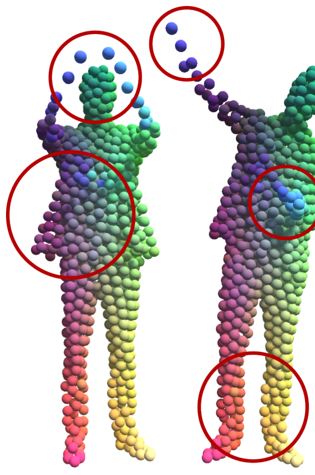
This approach can be easily extended to multi-dimensional attributes (e.g. RGB colours) by treating each component of the attribute vectors separately. The formulation through feature vectors also allows for attribute mapping with input attribute maps defined for more than a single frame. This means it can be used in a related scenario - TVM attribute compression. Instead of the attribute maps for each frame, only the centers and the parameters of the model representing the attribute function a are stored.

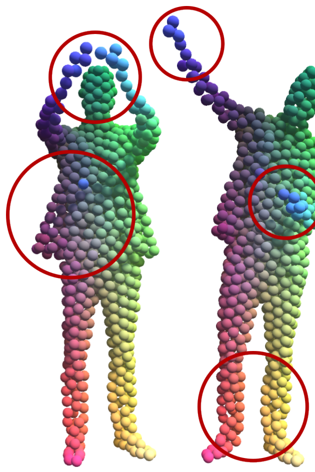
The representation capability of such mapping is influenced by multiple contributing factors: the number of samples, the representation capability of the model representing the function and the temporal stability of the feature vectors.

8.8 Summary

We have presented a novel temporal model for representing the motion of dynamic surfaces based on tracking of volume elements denoted *tracked centers*. The model is agnostic of the underlying structure of the surface and can even handle sequences where surface correspondences between the frames are non-bijective.


(a) PBT


(b) IIR-based affinity


(c) Max-based affinity

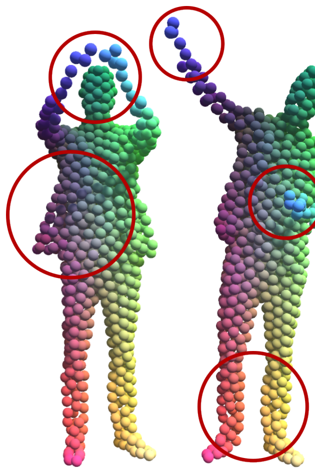

(d) Global optim.

Figure 8.21: Comparison of results obtained with all the published versions of our tracking algorithm. Highlighted are the areas with the most notable differences.

Although the original tracking pipeline was very limited, we have achieved considerable improvements. Initially, the main issue of the tracking approach was the presence of tracking irregularities. Realizing that these may never be entirely prevented from occurring, we proposed a post-processing step, in which we aim to remove these from the tracked results. There is still a very small number of irregular centers that were undetected by our method, mainly those, which became irregular in the first (resp. last) few frames. Figure 8.21 shows the progress we achieved in terms of irregularities. Currently, however, the main issue of obtained tracking results is that they are not designed to work as a deformation model, which still limits their applicability in structure-preserving TVM compression.

Regardless of the issues, the tracked centers have already some applications in different areas. We find the centers particularly helpful in temporally coherent editing.

Priority-based Connectivity Coding 9

To address the limitations of connectivity coding in TVM compression described in Section 4.5.1, we have proposed a novel method for connectivity coding with known geometry, which is similar to the Distance-ranked approach of Marais et al. [MGS07]. The method was published in *Computer graphics forum* journal [Dvo+22b]. It was later selected as one of the CGF papers to be presented at *Eurographics 2023* conference. The source code of the method was released under the MIT License and is available at:
`https://gitlab.kiv.zcu.cz/jdvorak/priority-based-connectivity-coding`.

9.1 Algorithm overview

Our algorithm (as well as the Edgebreaker and the Distance-ranked algorithm of Marais et al. [MGS07]) follows the general pipeline of traversal-based algorithms for connectivity encoding. The procedure starts with a single arbitrarily chosen triangle, which immediately divides the mesh into a processed part (the selected triangle) and an unprocessed part (the rest of the mesh). Both the encoder and the decoder then enter a loop, where a single edge on the border separating the processed and unprocessed parts is selected (denoted *gate*), and a single triangle is attached in a certain way. The task of the encoder is to emit the information needed by the decoder to attach the new triangle correctly.

This is done by identifying the third vertex (*tip vertex*) that forms the new triangle together with the gate. The tip vertex may lie on the processed/unprocessed border, or it may be a vertex that has not been visited by the traversal yet. The distance-ranked algorithm [MGS07] determines a point in space where the tip vertex is most likely to be located (*prediction*) using the location of the gate and the vertex completing the triangle that is incident with the gate within the already processed part of the mesh (this vertex will be denoted *base vertex*), and the available vertices are ordered by their distance to it. The encoder then emits a symbol identifying the actual tip vertex within this sorted list. The assumption is that the true tip vertices will be located at the beginning of their respective lists, yielding a stream of integer indices with a distribution that is strongly skewed towards zero and thus of low entropy. The typical situation is depicted in Figure 9.1. Our algorithm also uses a sorted list of candidate vertices, but these are ordered employing their suitability of forming a feasible triangle, which we will denote a *candidate vertex quality* (see Section 9.3).

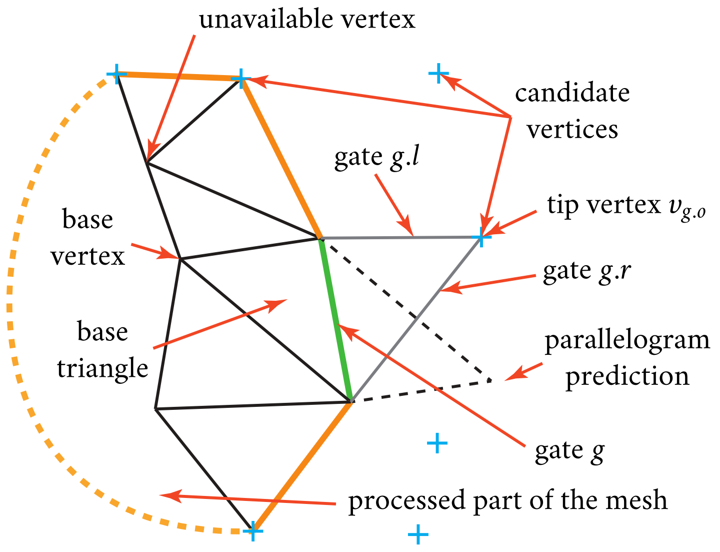


Figure 9.1: A schematic of a typical situation in each step of the main algorithm loop. All blue crosses represent candidate vertices, the task is to identify which one is the true tip vertex.

The distance-ranked algorithm continues by selecting the next gate using certain simple local rules, so that no additional data must be transmitted, and by processing the gates until the whole mesh is covered. Our main observation is that a *different traversal order* that provides a *lower code entropy* can be found. At each step of the main algorithm loop, there are several possible gates available to the decoder, and we conjecture that it should choose one where the identity of the tip vertex can be estimated most reliably, based on the available (already decoded) data. The encoder merely mimics the decoder's reasoning and therefore the two stay in sync, even though no additional data is sent to drive the traversal order. As a result, the main loop of the proposed algorithm is driven by a *priority queue* (PQ) of available gates, rather than by implicit rules for selecting the next gate. The process of determining which gates should be processed first will be discussed in Section 9.2.

Additionally, the previous method [MGS07] reserved the symbol 0 to identify a boundary edge. To handle these edges, we develop a simple prediction rule, which will be discussed in Section 9.4, that allows further data reduction.

9.2 Priority-driven traversal

The objective of priority-driven traversal is to first process the edges where the encoder is the most certain it will make a correct guess of the tip vertex. The priority cannot simply reflect the quality of the best possible candidate: a gate with a single, albeit lower-quality candidate that outperforms all other candidates by a large margin should be preferred over a gate with two or more candidates of similarly high quality since the probability of generating a low-magnitude code is higher in the first case. To address this, for each gate, the two highest candidate qualities $q_{\max}$ and $q_{\max2}$ are determined, and the priority of the gate is determined as $p = q_{\max} - q_{\max2}$. After the calculation, the highest quality value is also stored to be used later when performing the boundary prediction.

Having the traversal order reflect the certainty means that ambiguous configurations are often ignored until the algorithm arrives at the same place from a different location, where the situation might be clearer (see Figure 9.2). The distance-ranked algorithm would be forced to guess the tip vertex, which is more likely to be guessed incorrectly, thus hurting the compression rate.

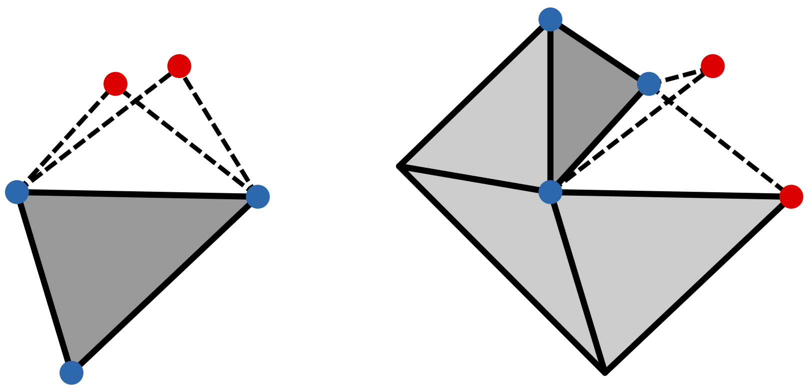
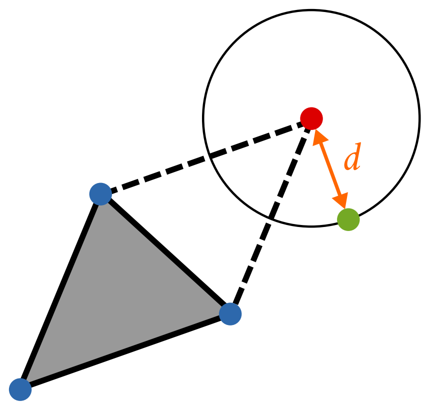


Figure 9.2: Example of priority-driven traversal. Highlighted triangle with blue-coloured vertices is the base triangle. Candidate vertices are coloured in red. Leaving ambiguous situations (left) for later means there is a possibility to arrive at the same place from a different direction, where the encoder might be more certain it will take a correct guess (right).

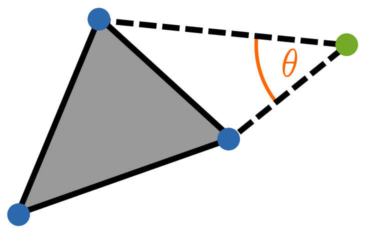
The priority-driven traversal also has an additional benefit. Since the probabilities of symbols are expected to be of the exponential distribution, it is helpful to encode them using unsigned exp-Golomb code. This assumption should also hold for the distance-ranked approach. However, if our method estimates the priority correctly, we expect that it emits smaller values first, as the more ambiguous scenarios are treated later. This yields a stream of values in an order, which can be then exploited by context-adaptive coding algorithms such as CABAC [MSW03] to achieve data rates smaller than the overall entropy of coded symbols.

9.3 Vertex candidate quality

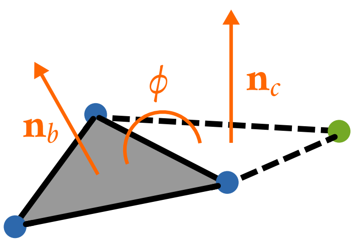
While encoding vertex indices, the algorithm attempts to determine their quality in such a way that the best candidates consistently achieve the highest ranking qualities. In order to achieve this, several properties of the *base triangle* (the known triangle adjacent to the active gate) and the *candidate triangle* (consisting of the gate and a candidate vertex) are considered. These are shown in Figure 9.3.



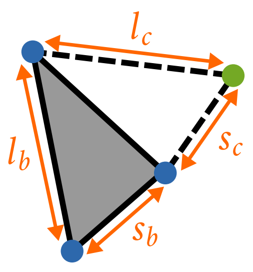
(a) Distance to parallelogram prediction



(b) Inner angle



(c) Dihedral angle



(d) Triangle similarity

Figure 9.3: Geometric criteria contributing to our vertex candidate quality computation.

The first of these properties is the *distance d* of a candidate vertex from the parallelogram prediction produced using the gate (see Figure 9.3a). In order to maintain cohesion between meshes, the distance value is divided by the *average gate length* l_{avg} of the mesh, which can be determined before encoding.

Next, the *inner angle θ* at the tip of the candidate triangle is considered (as shown in Figure 9.3b). Larger inner angles are expected to correspond to higher candidate quality, as sliver-like triangles with very small angles at the tip are generally avoided, while obtuse tip angles indicate the proximity of the vertex to the gate. This may not apply to meshes modelled manually for 3D printing and manufacturing, but by adjusting parameter weights, it is possible to achieve good compression results even for this class of meshes.

The *dihedral angle* ϕ between the base triangle and the candidate triangle is also considered. Flatter surfaces can be considered more likely and therefore correspond to a higher quality. The algorithm evaluates the dihedral angle as

$$\phi = \pi - \arccos(\mathbf{n}_b \cdot \mathbf{n}_c), \tag{9.1}$$

where $\mathbf{n}_b$ and $\mathbf{n}_c$ are the unit normals of the base and candidate triangles respectively (see Figure 9.3c). This way, the angle approaches π for flat predictions and decreases when bending to either side. The quality can thus be linked directly to ϕ.

Finally, the *similarity of the base and the candidate triangles* S is considered. With b_s and b_l denoting the shorter and the longer non-gate edge lengths in the base triangle respectively, and c_s and c_l denoting the shorter and the longer non-gate edge lengths in the candidate triangle respectively (see Figure 9.3d), a measure of triangle similarity can be expressed as:

$$\begin{aligned} r_s &= b_s/c_s, r_l = b_l/c_l \\ r &= (r_s + r_l)/2 \\ S &= -(|r - r_s| + |r - r_l|)/2. \end{aligned}$$

Note that S is zero when the ratios of shorter and longer non-gate edges are equal, otherwise, it is negative.

The complete formula for candidate quality q is then as follows:

$$q = \theta - \frac{w_1}{l_{\text{avg}}} \cdot d + w_2 \cdot \phi + w_3 \cdot S, \tag{9.2}$$

where w_1, w_2 and w_3 are weights that control the influence of the individual properties. Since all the terms are assumed to be positively proportional to the quality of a candidate, all the weights are chosen positive.

9.3.1 Search for relevant vertices

To allow fast encoding (resp. decoding) of the mesh, it is essential to always limit the number of vertices to be considered when obtaining the list of candidates for the tip vertex or when computing the gate priority. For the distance-ranked algorithm, this can be achieved efficiently, for example, by using a k-d tree: when encoding, the method performs a radial search between the prediction and the ground truth vertex; the decoder then performs a k-nearest neighbours search around the prediction, where k is the rank of the tip vertex.

Having candidate vertices sorted by the quality formulated in Eq. 9.2, the list of candidates can no longer be obtained this way, since for any candidate vertex of quality $q = q_c$, we cannot guarantee that all vertices of quality $q > q_c$ lie closer to the prediction. Nevertheless, we can deduce the search bound which contains all the relevant vertices by considering the possible ranges of values that can be input to Eq. 9.2.

For positively chosen w_1, w_2 and w_3, and a quality limit q_c, there is a particular *maximum distance* $d_{\max}$ from the parallelogram prediction up to which a better candidate can exist:

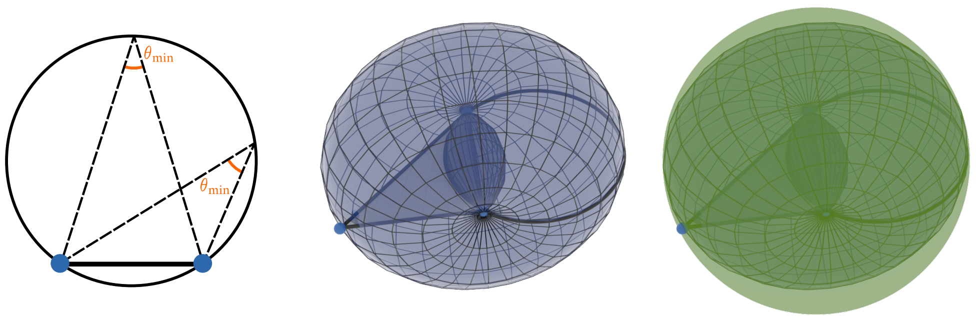
$$d_{\max} = (w_2 \cdot \pi + \pi - q_c) \cdot \frac{l_{\text{avg}}}{w_1}. \tag{9.3}$$

A potential candidate with $d = d_{\max}$ and all other quality terms as good as possible, i.e., $\theta = \phi = \pi$ and $S = 0$, is going to have $q = q_c$, therefore any candidates with $d > d_{\max}$ must have $q < q_c$.

Additionally, given a candidate with $q = q_c$, it is certain that any candidates with $q > q_c$ must have the inner angle $\theta > \theta_{\min}$ where

$$\theta_{\min} = q_c - w_2 \cdot \pi.$$

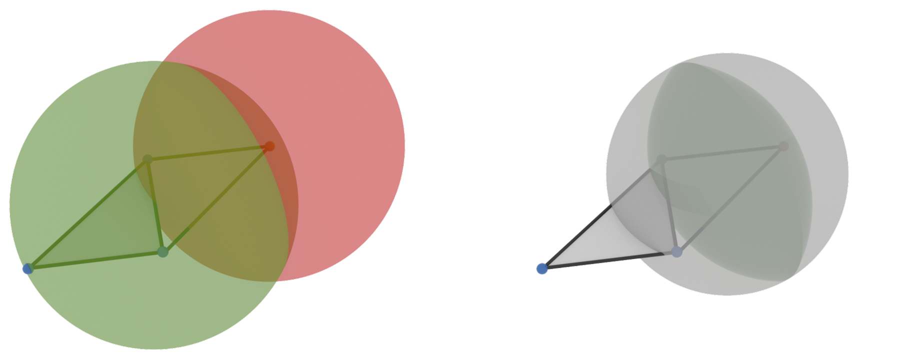
Such points lie within circles of a certain radius that depends on $\theta_{\min}$, of which the gate is a chord (Figure 9.4a). For acute $\theta_{\min}$, all such circles make up a torus (Figure 9.4b), which is in turn contained within a ball centered at the midpoint of the gate (Figure 9.4c). The radius of the ball is $r_t = (\|g\|/2)/\tan(\theta_{\min}/2)$, where $\|g\|$ is the length of the gate. For obtuse $\theta_{\min}$, the radius is simply $\|g\|/2$.



(a) 2D (b) 3D (c) Ball $\mathcal{B}_{\theta_{\min}}$

Figure 9.4: Candidate vertex search bound derived from inner angle $\theta_{\min}$.

The intersection of this ball with the ball defined by the condition of Eq. 9.3 can be enclosed within an even smaller ball centered at a certain point on the line connecting the gate midpoint and the parallelogram prediction, as shown in Figure 9.5. This ball provides a tight search space limitation when searching for a better candidate.



(a) Combined criteria (b) Final area for a radial search

Figure 9.5: All the candidates with $q > q_c$ lie within the intersection of the two balls derived from $d_{\max}$ and $\theta_{\min}$.

The candidate list is then further reduced by removing all the vertices that cannot represent the tip vertex: the vertices of the base triangle and the vertices having all the incident faces already encoded.

9.3.2 Finding optimal weights

One limitation of quality formulated as a weighted sum is the need for fine-tuning the weights w_1, w_2 and w_3 to optimize the data rate function $bpf(w_1, w_2, w_3)$. Not only does the global optimum of bpf lie at different points for each individual mesh, but it is also difficult to find such an optimum. Due to various contributing factors (e.g., a slight change in one of the weights can lead to a significant change in the traversal order), the data rate function is noisy with lots of local optima, although a trend can be seen towards a range of parameters which achieve satisfactory rates, as shown in Figure 9.6. This unfortunately means that we cannot use gradient-based methods to find the global optimum.

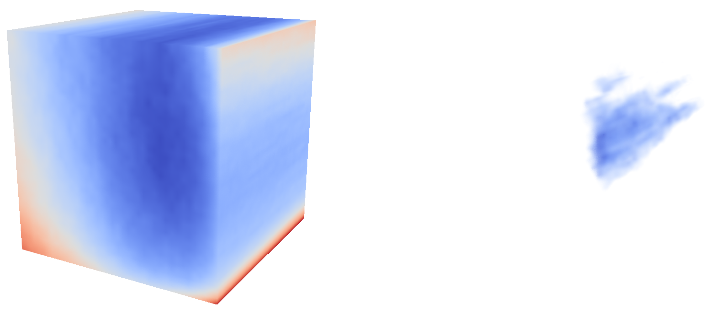


Figure 9.6: Data rate visualized as a function of the configuration space (w_1, w_2, w_3) with two different transfer functions. The basin of the minima (right) has a noisy, disconnected shape due to unpredictable effects in encoding.

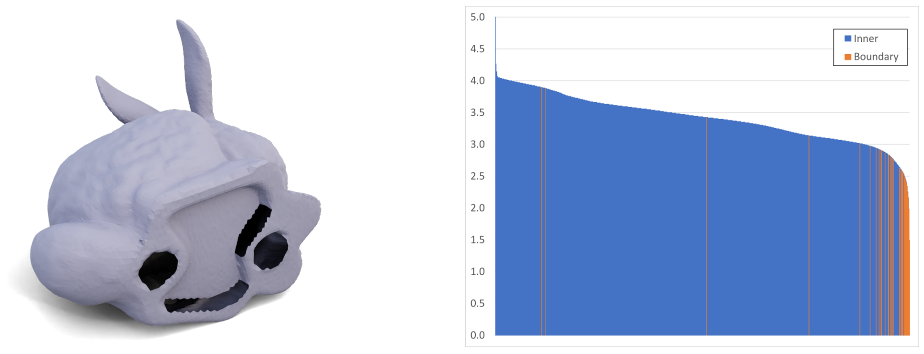
Using an exhaustive search over a range of feasible values is computationally expensive since the data rate function cannot be evaluated without first encoding the data. Although this approach tests a large number of configurations, it is still limited by the sampling resolution of the parametric domain, which means that it is not expected to find a global optimum.

Later in our experiments, we also used simulated annealing with steps of decreasing size, which significantly reduces the number of tested configurations and is not limited to the original parametric grid resolution, which means it can find a better configuration than the exhaustive search in fewer steps. However, the best configuration obtained is still not guaranteed to represent the global optimum.

In case one does not desire optimal data rates, but satisfactory ones, we have also selected 40 meshes of varying properties on which we found a single default configuration for our method. These included a few meshes of each dataset used in our experiments (see Section 9.5.1) accompanied by selected commonly used meshes in computer graphics research (e.g., Stanford Bunny, Armadillo, Igea, Max Planck). In Section 9.5.1, we discuss how much the results differ between default and fine-tuned parameters.

9.4 Boundary prediction

To identify a boundary edge in the output stream of symbols, we adopt the following strategy: based on the quality of the candidates, the decoder makes a prediction for each gate whether or not there is a triangle attached to it. The prediction is based on the assumption that for a border edge, there are probably no candidates of high quality available. This holds at least for meshes with large holes, e.g. the *Stanford bunny* as shown in Figure 9.7. The encoder mimics the prediction process and emits a code that contains a confirmation/rejection of the decoder's prediction.



(a) Mesh boundary (b) Sorted qualities $q_{\max}$ for each edge

Figure 9.7: The connection between the gate being a boundary edge and the quality of its best candidate $q_{\max}$ demonstrated on the *Stanford bunny* model.

In order to facilitate the border gate prediction, we introduce a border quality threshold q_t. Its value is determined in a simplified encoding-like traversal of the input mesh, which does not use the kd-tree to find the index of the tip vertex, instead using the known, true tip vertex, and using only nearest-neighbour queries while calculating gate priorities. The algorithm also determines up front which edges lie on the mesh boundary, if any. During the calculation of a gate's priority, the quality of the best candidate vertex $q_{\max}$ is determined. The best candidate qualities for border edges are stored in a set Q_b, while the best candidate qualities for inner edges are stored in another set Q_i.

Expecting that qualities in Q_b are predominantly low, as there are no triangles attached to the corresponding gates, the algorithm finds a threshold quality q_t that best separates Q_i from Q_b. This is achieved by testing all qualities $q_c \in Q_b \cup Q_i$ as candidate thresholds and evaluating the separating ability of each q_c as the number of items $q \in Q_b,\ s.t.\ q < q_c$ plus the number of items $q \in Q_i,\ s.t.\ q > q_c$, i.e., the total number of correctly predicted border/inner gates, if q_c were selected as threshold. The q_c yielding the largest sum is finally chosen as q_t.

The index encoding and decoding processes are then modified to accommodate boundary edges. The decoder can predict whether a gate represents a border edge by comparing the quality of the gate's best tip vertex candidate with q_t. The encoder must in turn produce a symbol based on the outcome of this prediction. It is also necessary to aim for low-magnitude output integers in order to make the output sequence well suited for arithmetic coding, i.e., in the most probable case, we would like to encode the symbol zero, or a low, non-negative integer. Four situations can arise while making the prediction, which we list in Table 9.1.

Upon encountering a gate that represents a true boundary edge, the encoder replicates the decoder's prediction. If the prediction correctly indicates a border, the encoder emits the symbol 0 to confirm it. If the prediction indicates an inner gate, the encoder must encode a special *border code* so that the decoder corrects its prediction and decodes the gate as a boundary edge. Temporarily, the encoder uses the symbol -1. Upon completing the encoding process, the largest encoded integer is found, and its value incremented by one is used as the border code. The border code is then used to replace all the temporary -1 symbols, resulting in a set of unsigned codes. The particular choice of border code is transmitted to the decoder as part of auxiliary data.

In the remaining case, i.e., when the gate represents an inner edge, two possibilities may occur. First, the decoder makes a correct prediction. The encoder can encode the index of the tip vertex directly. If a wrong prediction is made, i.e., the decoder predicts a border edge, it expects a zero code as confirmation. Since zero is also a viable identifier of the true tip vertex, the encoder must produce the true index incremented by one. This way, upon predicting a boundary edge, the decoder can always distinguish between a border confirmation and a correction, serving simultaneously as the identification of the tip vertex.

Table 9.1: Encoded symbol depending on the predicted and actual type of the gate, in order of usual occurrence from most to least common. Symbol i denotes the index of the tip vertex in the list of candidates. For typical datasets, the bottom two rows (wrong predictions) amount to less than 0.3% of the cases.

Prediction	Actual	Encoded symbol
Inner	Inner	i
Boundary	Boundary	0
Boundary	Inner	$i + 1$
Inner	Boundary	initially -1, finally $\max(i) + 1$

9.5 Experimental results

To evaluate the proposed connectivity compression algorithm, we compare the quantitative results against those of the Distance-ranked algorithm proposed by Marais et al. [MGS07]. Since there is no public implementation, we have reimplemented the algorithm following the original paper. The results of our implementation generally match those reported in the paper, and the remaining discrepancy can be reasonably explained by using a different entropy coder in the current experiments. To eliminate the influence of the particular choice of entropy coding, we are also reporting the entropy of the output symbol sequence.

In all experiments, we use an adaptive entropy coder based on the CABAC scheme [MSW03]. The encoder uses an unsigned exp-Golomb code, which is well suited for data of exponential distribution, which can be roughly observed with the encoded symbol sequences.

9.5.1 Connectivity compression

First, we report the results of a pair of "sanity check" experiments. When a mesh has a very regular geometry and connectivity, such as a sphere constructed by subdividing an icosahedron (see Figure 9.8), it is expected that the best candidate will always be the true tip vertex, identified by a sequence of identical symbols. Indeed, both the proposed algorithm and the Distance-ranked algorithm [MGS07] reach exact zero entropy for such a mesh with 2562 vertices. Since the entropy coder uses a few auxiliary bytes, the resulting actual data rates are slightly larger than zero: 0.048 bpf (bits per face) for the proposed algorithm vs. 0.078 bpf for the Distance-ranked algorithm [MGS07].

To test the other end of the regularity spectrum, we have created a mesh with the connectivity of a subdivided icosahedron with 18000 faces, however, the vertex positions were chosen at random within a unit sphere (see Fig. 9.8). Both algorithms have reached comparable entropy of 12.42 and 12.18 bits per symbol for the proposed and the Distance-ranked [MGS07] algorithms respectively. The actual data rate was more than 15 bpf since the data do not exhibit the exponential distribution expected by the entropy coder because the prediction fails in each case and therefore the symbols are roughly uniformly distributed.

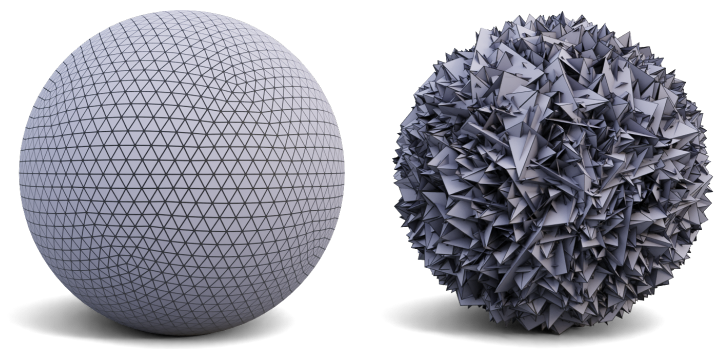


Figure 9.8: Meshes used in "sanity check" experiments. Left: A subdivided icosahedron yielding perfect compression. Right: A mesh with random positions within the unit sphere representing the worst case.

Note that this translates to more than 30 bits per vertex. The Edgebreaker [Ros99] algorithm is known to provide a guarantee of reaching less than 4 bpv for any data. In our setting, the vertex indexing is fixed by the input geometry and cannot be altered without encoding additional data. An arbitrary permutation map for a sequence of N elements costs an additional $\log_2(N!)/N$ bits per element. With 9000 vertices of the test mesh, this translates to 11.69 additional bpv. Together with the 4 bpv needed by the Edgebreaker, we have 15.69 bpv, which is better than what the two tested algorithms have produced, but by a smaller margin. With more realistic data, the tested algorithms reach much lower data rates, while the need for reindexation (and the related additional storage cost) when using Edgebreaker or a similar algorithm remains.

For experiments with real-world data, we have selected a group of datasets of varying character. From each dataset, we have randomly selected up to 10 000 meshes with manifold connectivity, and we have unified the face orientation using the PyMeshlab software [MC21].

There were six datasets in total:

1. the *ABC_regular* dataset has been obtained from a set of 1 million CAD models [Koc+19], selecting from the subset of regular triangulations,
2. the *ABC_irregular* dataset has been obtained from the same set of CAD models [Koc+19], selecting from the subset of irregular triangulations,
3. the *thingi10k* dataset has been selected from the set of models published by Zhou et al. [ZJ16],
4. the *tosca* dataset has been constructed from a set of nonrigid 3D shapes in a variety of poses used for non-rigid shape similarity and correspondence experiments [BBK09],
5. the *mcGill* dataset has been constructed from a set of meshes used for testing of shape retrieval algorithms [Zha+05],
6. and the *casual_man* dataset is a proprietary scanned human Time-varying mesh

Representative meshes of each dataset with details illustrating the regularity of triangulation are shown in Figure 9.9. The best performance is expected on the *ABC_regular*, the *mcGill* and the *casual_man* datasets which all exhibit regular triangulations induced by certain meshing algorithms. The *thingi10k* has varying regularity, as it also contains, e.g., 3D scans [ZJ16]. However, the majority of models are irregular. The worst performance is expected on the *ABC_irregular* and the *tosca* datasets, since all their meshes have irregular triangulation typical for models created by 3D modelling tools.

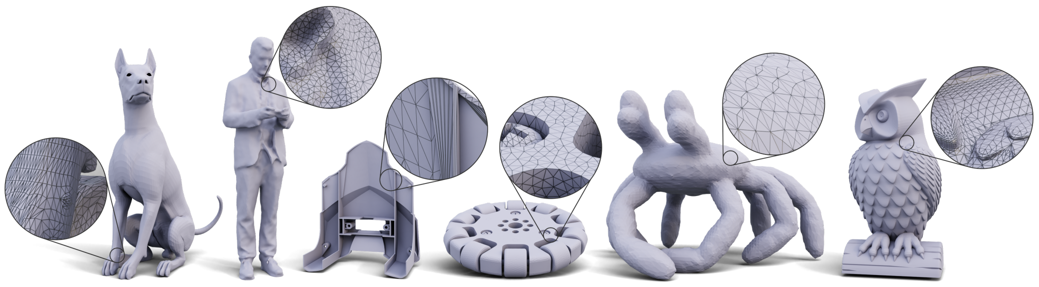


Figure 9.9: Representative models of each dataset used in our experiments. Details show the triangulations of models. From left to right: *tosca, casual_man, ABC_irregular, ABC_regular, mcGill, thingi10k*.

From each dataset, 10 meshes were randomly selected for optimising the parameters of the quality function. The optimal parameters were found using an exhaustive search over a range of feasible values, and the best-performing configuration has been used for the rest of each dataset without change. We also measured the results for the default configuration. The selected parameters for each dataset and the default parameters are shown in Table 9.2.

dataset	w_1	w_2	w_3
default	**0.4094**	**0.7920**	**0.1350**
abc_regular	0.252	0.654	0.151
abc_irrgegular	0.84	1.8125	0.089
thingi10k	0.24	0.905	0.000015
tosca	2.18	1.05	0.0005
mcGill	0.362	0.116	0.845
casual_man	0.1406	0.8781	0.0272

Table 9.2: Configuration for each dataset used in our experiments. The default configuration is highlighted.

Additionally to data rate and entropy, we have also measured encoding and decoding times. The experiment was conducted on a machine with AMD Ryzen 9 5950X 3.4 GHz 16 core processor, 64 GB of memory and NVIDIA GeForce GTX 1650 graphics card. The results are summarized in Table 9.3. The data rates and entropy listed in the table are averages weighted by vertex count.

Table 9.3: Compression performance. Bits per face (bpf) counts and entropy (H) values are averaged using vertex counts as weights. Encoding and decoding times (t_e, t_d) are measured as average time per vertex (μs). Best values are highlighted.

		Default parameters				Optimized parameters				[MGS07]			
dataset	# M	bpf	H	t_e	t_d	bpf	H	t_e	t_d	bpf	H	t_e	t_d
abc_regular	10000	0.191	0.296	40.2	42.8	**0.181**	**0.282**	36.7	39.5	0.331	0.381	**34.6**	**28.0**
abc_irrgegular	10000	1.059	1.225	135.4	186.0	**1.050**	**1.195**	119.7	170.4	2.261	2.324	**111.0**	**120.4**
thingi10k	8133	0.988	1.148	84.8	102.9	**0.919**	**1.079**	103.6	120.3	1.523	1.552	**40.1**	**36.2**
tosca	80	1.129	1.261	116.7	83.5	**1.112**	**1.247**	68.9	55.3	1.343	1.317	**27.1**	**25.9**
mcGill	458	0.487	0.582	36.0	38.8	**0.448**	**0.521**	31.3	34.3	0.708	0.693	**18.4**	**13.4**
casual_man	546	0.165	0.183	37.5	40.2	**0.152**	**0.171**	35.9	38.2	0.264	0.232	**18.9**	**13.7**

It can be seen that the proposed algorithm produces data rates that are consistently considerably lower than those achieved by the state-of-the-art algorithm. The largest reduction in bpf of 53.5% has been achieved with the *abc_irregular* dataset. Generally, the lowest data rates are achieved with regular datasets (abc_regular, mcGill and casual_man), while models with irregular sampling, such as those created in CAD systems, result in higher data rates.

The results obtained using optimised parameters are only slightly better than those obtained using the default configuration. The largest reduction in bpf of 8.2% has been achieved with the *casual_man* dataset, while the smallest of 0.8% has been achieved with *abc_irregular* dataset. It seems that regular datasets benefit more from the parameter optimisation, while for irregular datasets, the default parameters generally suffice. Nevertheless, on all the datasets, both configurations yield better results than the current state of the art.

While the Distance-ranked algorithm achieves similar values of bpf and entropy, in the results of the proposed algorithm the bpf is consistently lower than the entropy (by 9.5% – 35.9%). This indicates that the priority-based traversal indeed emits symbols in order that is exploitable by the CABAC-based encoder we use.

As expected, the runtimes of our method are higher than those of the Distance-ranked approach, which is not required to perform the extended search when determining a vertex rank. The highest difference between the two methods occurs on the *thingi10k* dataset, where our method performs 3.3 times slower in decoding. In the case of pure CAD datasets (the *abc_regular* and the *abc_irregular* datasets), the running times are comparable. This is because, in the Distance-ranked algorithm, the means of adjusting the prediction by stretching sometimes leads to the processing of all model vertices for a single gate [MGS07, Section 3.2].

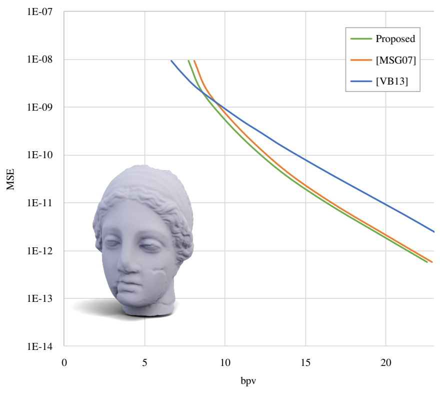
To illustrate, how the encoding and decoding runtimes of both methods are affected by mesh size, we also show times measured at different orders of magnitude of vertex counts (see Table 9.4) given fixed parameters (optimised). The reported times are averages over 10 selected models with the number of vertices closest to the given order of magnitude. All the models were selected from the *abc_regular* dataset, which exhibits low variance in mesh regularity, but large variance in mesh sizes. The results show that with an increasing number of vertices, the difference between the two methods gets smaller and, eventually, our method performs better for large meshes. We again believe that this is due to the inefficiency of the prediction adjustment in the Distance-ranked algorithm.

Table 9.4: Encoding and decoding times (ms) at different orders of magnitude of numbers of vertices for the *abc_regular* dataset.

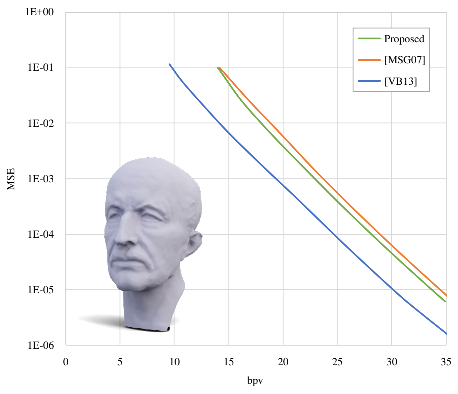
	Proposed		[MGS07]	
$\sim \lvert V \rvert$	t_e	t_d	t_e	t_d
1000	31.02	34.12	15.78	13.03
10000	391.45	384.83	307.09	256.40
100000	3878.33	3776.72	4885.15	4102.01
1000000	61644.31	60679.80	246979.09	225822.83

9.5.2 Mesh compression

Inspired by similar experiments of Marais et al., we have also evaluated the performance of our method in terms of static (intra-only) mesh compression. To this end, we first encode the geometry with a static point cloud geometry coder by Merry et al. [MMG06], which constructs a minimum spanning tree over the vertices and uses the structure of the tree for prediction and then we apply our method on connectivity. We compared this method to the Distance-ranked algorithm [MGS07] combined with the same geometry coder and a connectivity-first Edgebreaker-based weighted parallelogram [VB13] (WP) mesh coding scheme, which can be considered a state-of-the-art method for single rate manifold mesh compression. Unlike Marais et al., who originally considered only the performance on a single level of precision, we compared the Rate-Distortion (RD) performance using Mean-Squared Error (MSE) as a distortion measure over all levels of precision permitted by the used point cloud coder. Experiments were done on *Igea* (regular triangulation) and *Max Planck* (irregular triangulation) models. Different parameters were selected for each model, however, equal for all the levels of precision, found by using simulated annealing with undistorted original data. The results are shown in Figure 9.10.



(a) Igea



(b) Max Planck

Figure 9.10: RD curve comparison of the proposed method with Distance-ranked compression [MGS07] and Weighted parallelogram [VB13]

Our method achieves better data rates than the Distance-ranked [MGS07] compression on both models at all levels of precision. For the Max Planck model, the WP [VB13] outperforms both the geometry-first approaches. At the same amount of distortion, it achieves data rates of around 3.75 bpv less than our proposed method. For the Igea model, the geometry-first approach performs better at data rates higher than circa 8.5 bpv. This is because the regular and dense sampling of vertex positions allows efficient geometry coding which does not depend on the connectivity. Our method also benefits from such mesh properties. However, with increasing distortion, the reconstructed geometry that drives the connectivity coding gradually loses regularity. As a result, the difference between the geometry-first approaches and WP decreases. We believe that this effect could be slightly attenuated by optimising the parameters of the connectivity compression for each level of precision, however, it cannot be fully eliminated.

9.6 Summary

We have presented a connectivity compression algorithm that encodes a set of triangles for a known set of vertex positions available at both the encoder and the decoder. By controlling the traversal by a geometric priority, the algorithm is able to outperform the current state of the art by up to 53.5% (39.2% on average per dataset) for connectivity coding.

The performance gain depends mostly on the characteristics of the input data. Naturally, for a connectivity that is random and cannot be predicted, there is no gain; however, even for practical connectivities, there are differences in the algorithm's performance. In particular, the algorithm works well for evenly and densely sampled surfaces, such as those obtained by means of scanning real-world objects, while it struggles with artificial models created using standard means of computer-aided design.

The main motivation for our method was its application in TVM compression, where the geometry can be encoded separately to exploit the temporal coherence between the frames. However, it turns out it is also applicable for the compression of static regular meshes, where it can be combined with a point cloud codec [MMG06] to form a method that outperforms conventional connectivity-driven approaches such as weighted parallelogram [VB13]. It could also be utilized in a progressive compression pipeline, where the first few levels of detail are represented by a point cloud only, and then, at some level of detail, the connectivity is transferred using our priority-driven approach.

Currently, the method is limited to 2-manifold triangle meshes as it is built upon the Edgebreaker algorithm. The ideas of the approach (vertex candidate quality, priority-driven traversal and boundary prediction) could possibly be applied to a different algorithm, e.g., the TFAN [MZP09]. However, in such a case, it will be much more difficult to deduce whether a certain vertex has already all of its incident faces processed and can be safely removed from the vertex candidate list.

Although the default configuration suffices for irregular data, fine-tuning of parameters is, nevertheless, required to obtain the best compression performance. Even with the simulated annealing, the process is still quite costly. In our future work, we would like to analyze whether there might be a connection between the optimal configuration and certain mesh properties such as vertex degrees, inner angles etc. Alternatively, the weights might be adjusted on the go during the traversal.

In the future, we would also like to investigate other possible formulations of the priority, aiming at improving the performance with CAD models. Even such models exhibit certain properties that could be exploited in order to improve the prediction accuracy and in turn reduce data rates. In particular, candidate dihedral angles of magnitude $\pi/2$ may indicate candidate quality. Also, certain regularity of inner angles stemming from the sampling of basic primitives, such as cylinders or cones, could be exploited as well in estimating candidate quality.

Part IV
Epilogue

Conclusions 10

10.1 Summary

The focus of this thesis was the problem of compression of dynamic data, mainly the compression of sequences of triangle meshes with variable connectivity (time-varying meshes), which is still considered an open problem in contrast to the compression of dynamic meshes, where all the frames share the same connectivity. Current methods either discard the original structure of the data, have limited versatility (e.g., can handle only sequences of constant topology), or have a limited compression performance. We have pointed out the challenges one must face to address these issues. Additionally, some inspiration can be also drawn from methods in the related field of dynamic point cloud compression.

Our first contribution was in the field of static and dynamic mesh compression, where we proposed an algorithm to control error propagation in Laplacian mesh compression by adjusting encoded values on the fly in forward substitution when solving a linear system. Our method achieves compression performance on par with state-of-the-art under both mechanistic and perceptual criteria and also allows faster decoding times than the original High-pass coding approach.

We have also proposed a temporal-model-based method for compression of molecular dynamics trajectories. The method utilizes a temporal model denoted *canonical molecule*, which captures the relative positions of atoms and angles between incident edges representing atom bonds. Results show that our method performs particularly well on large molecular systems (e.g. proteins).

One of our two main contributions is a temporal model for representing surfaces evolving in time denoted *tracked centers*. We have proposed multiple improvements since publishing the original work, each substantially reducing the number of tracking irregularities. There are already various applications of the model, although its current limitations prevent it from being applicable in the structure-preserving compression of TVMs.

Our second main contribution was a method for compressing mesh connectivity for known geometry. It uses a priority-driven connectivity traversal, which emits symbols in an order that can be exploited by context-adaptive coding. The method can be integrated into any structure-preserving TVM compression pipeline that encodes geometry separately from connectivity, as long as the input frames are 2-manifold meshes.

Except for the algorithm for error propagation control in Laplacian compression and the initially proposed volume tracking pipeline, the source codes of all our contributions were released to the public under permissive licences.

10.2 Discussion

There are still issues to be solved before we will be able to combine our main contributions to form a complete structure-preserving TVM compression pipeline. Currently, we are researching better ways of using the tracked centers as a deformation model and also the ways of taking such a deformation model into account when optimizing the center positions. Addressing this problem should result in a much better alignment of subsequent frames for geometry compression, while we expect almost no changes in the data footprint of the model. Alternatively, the surface approximation presented in Section 8.7.1 might also be used to predict frame geometry, however, different challenges would have to be solved. It might also be possible that these challenges cannot be overcome. In such a case, we plan to investigate different temporal models for versatile data, likely based on deep learning. Nevertheless, the tracked centers already show promising results in different applications (e.g., surface approximation and temporally-coherent editing).

For practical reasons, it is also important to adjust our connectivity compression method to non-manifold meshes in the future. Although we have access to TVMs with all the frames being 2-manifold meshes, in real-world scenarios, the non-manifold frames are to be expected, particularly if the compression is to be performed immediately after the process of obtaining the data.

The area of TVM compression is expected to further evolve in the future, in part also due to the current efforts of the MPEG. Although the method that is currently developed to be the future MPEG V-DMC standard does not preserve mesh structure, it is very likely that in the future, it will be adapted to preserve this information.

To date, the performance of all the TVM compression methods has been evaluated using only the mechanistic error measures. To the best of our knowledge, there is currently no perceptual metric designed specifically to work with the geometry of time-varying meshes. For this reason, we also plan to focus on this research area. Proposing such a metric will very likely require some notion of temporal coherence, for which, in theory, it might be possible to incorporate our tracked centers.

Activities

Publications in International Conferences

- DVOŘÁK, Jan; VANĚČEK, Petr; VÁŠA, Libor. Towards Understanding Time Varying Triangle Meshes. *Computational Science – ICCS 2021.* Cham: Springer International Publishing, 2021, pp. 45–58. ISBN 978-3-030-77977-1. doi:10.1007/978-3-030-77977-1_4
- DVOŘÁK, Jan; HÁCHA, Filip; VÁŠA, Libor. Global Optimisation for Improved Volume Tracking of Time-Varying Meshes. In: *Computational Science – ICCS 2023.* Springer Nature Switzerland, 2023, pp. 113–127. doi:10.1007/978-3-031-36027-5_9

Publications in Impacted Journals

- VÁŠA, Libor; DVOŘÁK, Jan. Error propagation control in Laplacian mesh compression. *Computer Graphics Forum*. 2018, vol. 37, no. 5, pp. 61–70. doi:10.1111/cgf.13491 (**IF 2.373**, Citations in WOS: 1)
- DVOŘÁK, Jan; MAŇÁK, Martin; VÁŠA, Libor. Predictive compression of molecular dynamics trajectories. *Journal of Molecular Graphics and Modelling*. 2020, vol. 96, p. 107531. issn 1093-3263. doi:10.1016/j.jmgm.2020.107531 (**IF 2.518**, Citations in WOS: 4)
- DVOŘÁK, Jan; KÁČEREKOVÁ, Zuzana; VANĚČEK, Petr; HRUDA,Lukáš; VÁŠA, Libor. As-rigid-as-possible volume tracking for time-varying surfaces. *Computers & Graphics*. 2022, vol. 102, pp. 329–338. ISSN 0097-8493. doi:10.1016/j.cag.2021.10.015 (**IF 1.821**, Citations in WOS: 2)
- DVOŘÁK, Jan; KÁČEREKOVÁ, Zuzana; VANĚČEK, Petr; VÁŠA, Libor. Priority-based encoding of triangle mesh connectivity for a known geometry. *Computer Graphics Forum*. 2022, vol. 42, no. 1, pp. 60–71. doi:10.1111/cgf.14719 (**IF 2.5**)

Participation in Scientific Projects

- SGS-2019-016, Synthesis and Analysis of Geometric and Computing Models, Ministry of Education, Youth and Sports
- SGS-2022-015, New Methods for Medical, Spatial and Communication Data, Ministry of Education, Youth and Sports
- 17-07690S, Methods of Identification and Visualization of Tunnels for Flexible Ligands in Dynamic Proteins, Czech Science Foundation
- 20-02154S, Representation and processing methods for three-dimensional dynamic shapes, Czech Science Foundation
- 22-04622L, Data compression paradigm based on omitting self-evident information - COMPROMISE, Czech Science Foundation

Stays Abroad

- Research stay at *TU Munich*, Computer Vision Group, Feb. 2020 - May 2020, first month in person, continued online due to the COVID-19 pandemic.

Oral Presentations

- Towards Understanding Time-Varying Triangle Meshes, 18. 6. 2021, English, *ICCS 2021*, online
- As-rigid-as-possible volume tracking for time-varying surfaces, 15. 11. 2021, English, *SMI 2021*, online
- Priority-based encoding of triangle mesh connectivity for a known geometry, 11. 5. 2023, English, *EUROGRAPHICS 2023*, Saarbrücken, Germany
- Global Optimisation for Improved Volume Tracking of Time-Varying Meshes, 2. 7. 2023, English, *ICCS 2023*, Prague, Czech Republic

Unpublished work

- HÁCHA, Filip; DVOŘÁK, Jan; KÁČEREKOVÁ, Zuzana; VÁŠA, Libor. Editing Mesh Sequences with Varying Connectivity. Submitted to: *Eurographics Conference 2024*
- VANĚČEK, Petr; HÁCHA, Filip; DVOŘÁK, Jan; VÁŠA Libor. A Predictor for Triangle Mesh Compression Working in Tangent Space. Submitted to: *GRAPP 2024*

Other publications

- HRUDA, Lukáš; DVOŘÁK, Jan. Estimating approximate plane of symmetry of 3D triangle meshes. In: *Proc. Central European Seminar on Computer Graphics*. 2017
- HRUDA, Lukáš; DVOŘÁK, Jan; VÁŠA, Libor. On evaluating consensus in RANSAC surface registration. *Computer Graphics Forum*. 2019, vol. 38, no. 5, pp. 175–186. doi:10.1111/cgf.13798 (**IF 2.116**, Citations in WOS: 2)

Bibliography

[ACL00] ALEXA, Marc; COHEN-OR, Daniel; LEVIN, David. As-Rigid-as-Possible Shape Interpolation. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 157–164. SIGGRAPH '00. ISBN 1581132085. Available from DOI: `10.1145/344779.344859` (cit. on p. 88).

[AK15] ALEXA, Marc; KYPRIANIDIS, Jan Eric. Error diffusion on meshes. *Computers & Graphics*. 2015, vol. 46, pp. 336–344. ISSN 0097-8493. Available from DOI: `10.1016/j.cag.2014.09.010`. Shape Modeling International 2014 (cit. on pp. 61–63).

[Alf+22] ALFACE, Patrice Rondao et al. V3C-based Coding of Dynamic Meshes. In: *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2022. Available from DOI: `10.1109/euvip53989.2022.9922839` (cit. on pp. 29, 38, 40, 43, 54).

[ACO16] ANIS, Aamir; CHOU, Philip A.; ORTEGA, Antonio. Compression of dynamic 3D point clouds using subdivisional meshes and graph wavelet transforms. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016. Available from DOI: `10.1109/icassp.2016.7472901` (cit. on pp. 45, 46).

[AG04] ANUAR, Nizam; GUSKOV, Igor. Extracting Animated Meshes with Adaptive Motion Estimation. In: *VMV*. 2004, pp. 63–71 (cit. on p. 63).

[ALM19] ARVANITIS, Gerasimos; LALOS, Aris S.; MOUSTAKAS, Konstantinos. Adaptive representation of dynamic 3D meshes for low-latency applications. *Computer Aided Geometric Design*. 2019, vol. 73, pp. 70–85. Available from DOI: `10.1016/j.cagd.2019.07.005` (cit. on p. 27).

[ALM21] ARVANITIS, Gerasimos; LALOS, Aris S.; MOUSTAKAS, Konstantinos. Fast Spatio-temporal Compression of Dynamic 3D Meshes. In: *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2021. Available from DOI: `10.1109/mmsp53017.2021.9733486` (cit. on pp. 26, 27).

[BSG12] BARBIČ, Jernej; SIN, Funshing; GRINSPUN, Eitan. Interactive editing of deformable simulations. *ACM Transactions on Graphics*. 2012, vol. 31, no. 4, pp. 1–8. Available from DOI: 10.1145/2185520.2185566 (cit. on p. 111).

[Bar+18] BARILL, Gavin; DICKSON, Neil G.; SCHMIDT, Ryan; LEVIN, David I. W.; JACOBSON, Alec. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics*. 2018, vol. 37, no. 4, pp. 1–12. Available from DOI: 10.1145/3197517.3201337 (cit. on p. 87).

[BSW08] BELKIN, Mikhail; SUN, Jian; WANG, Yusu. Discrete laplace operator on meshed surfaces. In: *Proceedings of the twenty-fourth annual symposium on Computational geometry*. ACM, ACM, 2008, pp. 278–287. Available from DOI: 10.1145/1377676.1377725 (cit. on p. 85).

[BM92] BESL, P. J.; MCKAY, Neil D. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1992, vol. 14, no. 2, pp. 239–256. Available from DOI: 10.1109/34.121791 (cit. on pp. 30, 34, 46, 49).

[Bis+20] BISWAS, Sourav; LIU, Jerry; WONG, Kelvin; WANG, Shenlong; URTASUN, Raquel. MuSCLE: Multi Sweep Compression of LiDAR Using Deep Entropy Models. *Advances in Neural Information Processing Systems*. 2020, vol. 2020-December. ISSN 1049-5258. Available from DOI: 10.48550/arXiv.2011.07590. Publisher Copyright: © 2020 Neural information processing systems foundation. All rights reserved.; 34th Conference on Neural Information Processing Systems, NeurIPS 2020 ; Conference date: 06-12-2020 Through 12-12-2020 (cit. on pp. 44, 45, 48, 52–54).

[BS07] BOBENKO, Alexander I; SPRINGBORN, Boris A. A discrete Laplace–Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry*. 2007, vol. 38, no. 4, pp. 740–756. Available from DOI: 10.1007/s00454-007-9006-1 (cit. on p. 60).

[Bog+17] BOGO, Federica; ROMERO, Javier; PONS-MOLL, Gerard; BLACK, Michael J. Dynamic FAUST: Registering Human Bodies in Motion. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. Available from DOI: 10.1109/CVPR.2017.591 (cit. on pp. 7, 92, 99).

[BLW12] BOJSEN-HANSEN, Morten; LI, Hao; WOJTAN, Chris. Tracking Surfaces with Evolving Topology. *ACM Trans. Graph.* 2012, vol. 31, no. 4. ISSN 0730-0301. Available from DOI: 10.1145/2185520.2185549 (cit. on pp. 17, 43, 81, 82).

[Bot+07] BOTSCH, Mario; PAULY, Mark; WICKE, Martin; GROSS, Markus. Adaptive Space Deformations Based on Rigid Cells. *Computer Graphics Forum*. 2007, vol. 26, no. 3, pp. 339–347. Available from DOI: 10.1111/j.1467-8659.2007.01056.x (cit. on p. 90).

[Bož+21] BOŽIČ, Aljaž et al. Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction. *CVPR*. 2021 (cit. on pp. 44, 82, 92).

[Bri+03] BRICEÑO, Hector M.; SANDER, Pedro V.; MCMILLAN, Leonard; GORTLER, Steven; HOPPE, Hugues. Geometry Videos: A New Representation for 3D Animations. In: San Diego, California: The Eurographics Association, 2003, pp. 136–146. SCA '03. ISBN 1581136595. Available from DOI: 10.2312/SCA03/136-146 (cit. on p. 35).

[BBK09] BRONSTEIN, Alexander M.; BRONSTEIN, Michael M.; KIMMEL, Ron. *Numerical Geometry of Non-Rigid Shapes*. Springer New York, 2009. Available from DOI: 10.1007/978-0-387-73301-2 (cit. on p. 128).

[Bud+12] BUDD, Chris; HUANG, Peng; KLAUDINY, Martin; HILTON, Adrian. Global Non-rigid Alignment of Surface Sequences. *International Journal of Computer Vision*. 2012, vol. 102, no. 1-3, pp. 256–270. Available from DOI: 10.1007/s11263-012-0553-4 (cit. on p. 82).

[CBI10] CAGNIART, Cedric; BOYER, Edmond; ILIC, Slobodan. Free-form mesh tracking: A patch-based approach. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010. Available from DOI: 10.1109/cvpr.2010.5539814 (cit. on p. 81).

[Cao+20] CAO, C.; TULVAN, C.; PREDA, M.; ZAHARIA, T. Skeleton-based motion estimation for Point Cloud Compression. In: *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2020, pp. 1–6. Available from DOI: 10.1109/MMSP48831.2020.9287165 (cit. on pp. 45, 51, 55).

[CH12] CASHMAN, Thomas J.; HORMANN, Kai. A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. *Computer Graphics Forum*. 2012, vol. 31, no. 2pt4, pp. 735–744. Available from DOI: 10.1111/j.1467-8659.2012.03032.x (cit. on p. 111).

[CGR09] CHAINE, Raphaëlle; GANDOIN, Pierre-Marie; ROUDET, Céline. Reconstruction Algorithms as a Suitable Basis for Mesh Connectivity Compression. *IEEE Transactions on Automation Science and Engineering*. 2009, vol. 6, no. 3, pp. 443–453. Available from DOI: 10.1109/TASE.2009.2021336 (cit. on p. 42).

[CK12] CHAMPAWAT, Yadvendar; KUMAR, Subodh. Online point-cloud transmission for tele-immersion. In: *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*. ACM, 2012. Available from DOI: `10.1145/2407516.2407540` (cit. on pp. 45, 48, 53).

[Che+18] CHEN, Chengju; XIA, Qing; LI, Shuai; QIN, Hong; HAO, Aimin. High-Fidelity Compression of Dynamic Meshes with Fine Details Using Piece-Wise Manifold Harmonic Bases. In: *Proceedings of Computer Graphics International 2018*. Bintan, Island, Indonesia: Association for Computing Machinery, 2018, pp. 23–32. CGI 2018. ISBN 9781450364010. Available from DOI: `10.1145/3208159.3208163` (cit. on p. 26).

[Che+19] CHEN, Chengju; XIA, Qing; LI, Shuai; QIN, Hong; HAO, Aimin. Compressing animated meshes with fine details using local spectral analysis and deformation transfer. *The Visual Computer*. 2019, vol. 36, no. 5, pp. 1029–1042. Available from DOI: `10.1007/s00371-019-01650-5` (cit. on p. 26).

[Cho+22a] CHOI, Yi-Hyun; JEONG, Jong-Beom; LEE, Soonbin; RYU, Eun-Seok. MPEG Dynamic Mesh Coding (DMC) standardization trend for volumetric video. *Proceedings of the Korea Broadcasting Media Engineering Conference Conference*. 2022, pp. 225–228 (cit. on pp. 38, 39).

[Cho+22b] CHOI, Yi-Hyun; JEONG, Jong-Beom; LEE, Soonbin; RYU, Eun-Seok. Overview of the Video-based Dynamic Mesh Coding (V-DMC) Standard Work. In: *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2022. Available from DOI: `10.1109/ictc55196.2022.9952734` (cit. on pp. 38, 39).

[Col+15] COLLET, Alvaro et al. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics*. 2015, vol. 34, no. 4, pp. 1–13. Available from DOI: `10.1145/2766945` (cit. on p. 81).

[Cor+13] CORSINI, M. et al. Perceptual Metrics for Static and Dynamic Triangle Meshes. *Computer Graphics Forum*. 2013, vol. 32, no. 1, pp. 101–125. Available from DOI: `10.1111/cgf.12001` (cit. on p. 15).

[Dar+12] DARIBO, Ismael et al. Efficient rate-distortion compression of dynamic point cloud for grid-pattern-based 3D scanning systems. *3D Research*. 2012, vol. 3, no. 1, p. 2. Available from DOI: `10.1007/3DRes.01(2012)2` (cit. on pp. 45, 48).

[Dou+15] DOU, Mingsong; TAYLOR, Jonathan; FUCHS, Henry; FITZGIBBON, Andrew; IZADI, Shahram. 3D scanning deformable objects with a single RGBD sensor. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 493–501. Available from DOI: `10.1109/CVPR.2015.7298647` (cit. on p. 49).

[Dou+14a] DOUMANOGLOU, Alexandros; ALEXIADIS, Dimitrios; ASTERIADIS, Stylianos; ZARPALAS, Dimitrios; DARAS, Petros. On human Time-Varying Mesh compression exploiting activity-related characteristics. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014. Available from DOI: `10.1109/icassp.2014.6854785` (cit. on pp. 34, 39, 42).

[Dou+14b] DOUMANOGLOU, Alexandros; ALEXIADIS, Dimitrios S.; ZARPALAS, Dimitrios; DARAS, Petros. Toward Real-Time and Efficient Compression of Human Time-Varying Meshes. *IEEE Transactions on Circuits and Systems for Video Technology*. 2014, vol. 24, no. 12, pp. 2099–2116. Available from DOI: `10.1109/TCSVT.2014.2319631` (cit. on pp. 8, 29, 34, 39, 41, 42, 44, 81).

[Dvo18] DVOŘÁK, Jan. *Application of data-dependent discrete Laplacian*. Pilsen, Czech Republic, 2018. Master's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor Doc. Ing. Libor Váša, Ph.D. (cit. on p. 167).

[DHV23] DVOŘÁK, Jan; HÁCHA, Filip; VÁŠA, Libor. Global Optimisation for Improved Volume Tracking of Time-Varying Meshes. In: *Computational Science – ICCS 2023*. Springer Nature Switzerland, 2023, pp. 113–127. Available from DOI: `10.1007/978-3-031-36027-5_9` (cit. on pp. 9, 81, 83).

[Dvo+22a] DVOŘÁK, Jan; KÁČEREKOVÁ, Zuzana; VANĚČEK, Petr; HRUDA, Lukáš; VÁŠA, Libor. As-rigid-as-possible volume tracking for time-varying surfaces. *Computers & Graphics*. 2022, vol. 102, pp. 329–338. ISSN 0097-8493. Available from DOI: `10.1016/j.cag.2021.10.015` (cit. on pp. 9, 81, 83).

[Dvo+22b] DVOŘÁK, Jan; KÁČEREKOVÁ, Zuzana; VANĚČEK, Petr; VÁŠA, Libor. Priority-based encoding of triangle mesh connectivity for a known geometry. *Computer Graphics Forum*. 2022, vol. 42, no. 1, pp. 60–71. Available from DOI: `10.1111/cgf.14719` (cit. on pp. 9, 117).

[DMV20] DVOŘÁK, Jan; MAŇÁK, Martin; VÁŠA, Libor. Predictive compression of molecular dynamics trajectories. *Journal of Molecular Graphics and*

Modelling. 2020, vol. 96, p. 107531. ISSN 1093-3263. Available from DOI: 10.1016/j.jmgm.2020.107531 (cit. on pp. 9, 67).

[DVV21] DVOŘÁK, Jan; VANĚČEK, Petr; VÁŠA, Libor. Towards Understanding Time Varying Triangle Meshes. In: PASZYNSKI, Maciej; KRANZLMÜLLER, Dieter; KRZHIZHANOVSKAYA, Valeria V.; DONGARRA, Jack J.; SLOOT, Peter M. A. (eds.). *Computational Science – ICCS 2021*. Cham: Springer International Publishing, 2021, pp. 45–58. ISBN 978-3-030-77977-1. Available from DOI: 10.1007/978-3-030-77977-1_4 (cit. on pp. 9, 81, 83, 108).

[ELC19] EISENBERGER, Marvin; LÄHNER, Zorah; CREMERS, Daniel. Divergence-Free Shape Correspondence by Deformation. In: *Computer Graphics Forum*. Wiley, Wiley Online Library, 2019, vol. 38, pp. 1–12. No. 5. Available from DOI: 10.1111/cgf.13785 (cit. on p. 114).

[FJB20] FARAMARZI, E.; JOSHI, R.; BUDAGAVI, M. Mesh Coding Extensions to MPEG-I V-PCC. In: *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2020, pp. 1–5. Available from DOI: 10.1109/MMSP48831.2020.9287057 (cit. on pp. 29, 37, 40, 43, 54).

[Fen+18] FENG, Xiang et al. A perceptual quality metric for 3D triangle meshes based on spatial pooling. *Frontiers of Computer Science*. 2018, vol. 12, no. 4, pp. 798–812. ISSN 2095-2236. Available from DOI: 10.1007/s11704-017-6328-x (cit. on p. 15).

[FLZ20] FENG, Yu; LIU, Shaoshan; ZHU, Yuhao. Real-Time Spatio-Temporal LiDAR Point Cloud Compression. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*. 2020. Available from DOI: 10.1109/iros45743.2020.9341071 (cit. on pp. 45, 50, 53).

[Fer+10] FERREIRA, Renan U.; HAN, Seung-Ryong; YAMASAKI, Toshihiko; AIZAWA, Kiyoharu. Mixed spatial and SNR scalability for TVM geometry coding. In: *2010 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video*. IEEE, 2010. Available from DOI: 10.1109/3dtv.2010.5506299 (cit. on pp. 32, 40).

[Flo03] FLOATER, Michael S. Mean value coordinates. *Computer Aided Geometric Design*. 2003, vol. 20, no. 1, pp. 19–27. ISSN 0167-8396. Available from DOI: 10.1016/s0167-8396(03)00002-5 (cit. on p. 60).

[Gal+18] GALLIGAN, Frank; HEMMER, Michael; STAVA, Ondrej; ZHANG, Fan; BRETTLE, Jamieson. *Google/Draco: a library for compressing and decompressing 3D geometric meshes and point clouds*. 2018 (cit. on pp. 38, 39).

[GD02] GANDOIN, Pierre-Marie; DEVILLERS, Olivier. Progressive Lossless Compression of Arbitrary Simplicial Complexes. *ACM Trans. Graph.* 2002, vol. 21, no. 3, pp. 372–379. ISSN 0730-0301. Available from DOI: `10.1145/566654.566591` (cit. on p. 41).

[Gar+19] GARCIA, Diogo C.; FONSECA, Tiago A.; FERREIRA, Renan U.; QUEIROZ, Ricardo L. de. Geometry Coding for Dynamic Voxelized Point Clouds Using Octrees and Multiple Contexts. *IEEE Transactions on Image Processing*. 2019, vol. 29, pp. 313–322. Available from DOI: `10.1109/TIP.2019.2931466` (cit. on pp. 45, 47, 52, 54).

[GQ17] GARCIA, Diogo C.; QUEIROZ, Ricardo L. de. Context-based octree coding for point-cloud video. In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, IEEE, 2017, pp. 1412–1416. Available from DOI: `10.1109/ICIP.2017.8296514` (cit. on pp. 45, 46, 52, 54).

[Gar+13] GARCIA, I.; XIA, Jiazhi; HE, Ying; XIN, Shi-Qing; PATOW, G. Interactive Applications for Sketch-Based Editable Polycube Map. *IEEE Transactions on Visualization and Computer Graphics*. 2013, vol. 19, no. 7, pp. 1158–1171. Available from DOI: `10.1109/tvcg.2012.308` (cit. on p. 36).

[GH97] GARLAND, Michael; HECKBERT, Paul S. Surface simplification using quadric error metrics. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*. ACM Press, 1997, pp. 209–216. Available from DOI: `10.1145/258734.258849` (cit. on pp. 33, 38).

[GZT22] GRAZIOSI, Danillo; ZAGHETTO, Alexandre; TABATABAI, Ali. *Video based mesh compression*. Google Patents, 2022. No. US20220108483A1. Available also from: `https://patents.google.com/patent/US20220108483A1`. US Patent App. 17/322,662 (cit. on pp. 38, 41, 43, 54).

[Gra21] GRAZIOSI, Danillo Bracco. Video-Based Dynamic Mesh Coding. In: *2021 IEEE International Conference on Image Processing (ICIP)*. 2021, pp. 3133–3137. Available from DOI: `10.1109/ICIP42928.2021.9506298` (cit. on pp. 37, 41, 43).

[GRO18] GROMACS DEVELOPMENT TEAM. *GROMACS File Formats*. 2018. Available also from: `http://manual.gromacs.org/documentation/2018/user-guide/file-formats.html` (cit. on pp. 68, 77).

[GGH02] GU, Xianfeng; GORTLER, Steven J.; HOPPE, Hugues. Geometry Images. In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. San Antonio, Texas: Association for Computing Machinery, 2002, pp. 355–361. SIGGRAPH '02. ISBN 1581135211. Available from DOI: 10.1145/566570.566589 (cit. on p. 35).

[Guo+15] GUO, Kaiwen; XU, Feng; WANG, Yangang; LIU, Yebin; DAI, Qionghai. Robust Non-rigid Motion Tracking and Surface Reconstruction Using L0 Regularization. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015. Available from DOI: 10.1109/iccv.2015.353 (cit. on p. 81).

[GSK03] GUPTA, Sumit; SENGUPTA, Kuntal; KASSIM, Ashraf. Registration and partitioning-based compression of 3-d dynamic data. *IEEE Transactions on Circuits and Systems for Video Technology*. 2003, vol. 13, no. 11, pp. 1144–1155. ISSN 1051-8215. Available from DOI: 10.1109/tcsvt.2003.817625 (cit. on pp. 29, 30, 40, 42, 53).

[HKM04] HABE, Hitoshi; KATSURA, Yosuke; MATSUYAMA, Takashi. Skin-off: Representation and compression scheme for 3D video. In: *Proceedings of Picture Coding Symposium (PCS '04)*. 2004, pp. 301–306 (cit. on pp. 29, 35, 40, 43).

[HZP16] HACHANI, Meha; ZAID, Azza Ouled; PUECH, William. Segmentation-based compression scheme for 3D animated models. *Signal, Image and Video Processing*. 2016, vol. 10, no. 6, pp. 1065–1072. Available from DOI: 10.1007/s11760-015-0859-0 (cit. on pp. 25, 27).

[HE15] HAJIZADEH, Mohammadali; EBRAHIMNEZHAD, Hossein. Predictive compression of animated 3D models by optimized weighted blending of key-frames. *Computer Animation and Virtual Worlds*. 2015, vol. 27, no. 6, pp. 556–576. Available from DOI: 10.1002/cav.1685 (cit. on p. 25).

[HE17] HAJIZADEH, Mohammadali; EBRAHIMNEZHAD, Hossein. Eigenspace compression: dynamic 3D mesh compression by restoring fine geometry to deformed coarse models. *Multimedia Tools and Applications*. 2017, vol. 77, no. 15, pp. 19347–19375. Available from DOI: 10.1007/s11042-017-5394-2 (cit. on p. 26).

[HE19] HAJIZADEH, Mohammadali; EBRAHIMNEZHAD, Hossein. NLME: a nonlinear motion estimation-based compression method for animated mesh sequence. *The Visual Computer*. 2019, vol. 36, no. 3, pp. 649–665. Available from DOI: 10.1007/s00371-019-01645-2 (cit. on p. 27).

[HYA07] HAN, Seung-Ryong; YAMASAKI, Toshihiko; AIZAWA, Kiyoharu. Time-Varying Mesh Compression Using an Extended Block Matching Algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*.

2007, vol. 17, no. 11, pp. 1506–1518. ISSN 1051-8215. Available from DOI: 10.1109/tcsvt.2007.903810 (cit. on pp. 29, 31, 40, 43, 53).

[HYA08] HAN, Seung-Ryong; YAMASAKI, Toshihiko; AIZAWA, Kiyoharu. Geometry compression for time-varying meshes using coarse and fine levels of quantization and run-length encoding. In: *2008 15th IEEE International Conference on Image Processing*. IEEE, IEEE, 2008, pp. 1045–1048. Available from DOI: 10.1109/ICIP.2008.4711937 (cit. on pp. 29, 31, 32, 40, 43, 54).

[Han+19] HANOCKA, Rana et al. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics*. 2019, vol. 38, no. 4, p. 90. Available from DOI: 10.1145/3306346.3322959 (cit. on p. 21).

[Har96] HART, John C. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*. 1996, vol. 12, no. 10, pp. 527–545. Available from DOI: 10.1007/s003710050084 (cit. on p. 109).

[Hou+13a] HOU, Junhui; CHAU, Lap-Pui; HE, Ying; MAGNENAT-THALMANN, Nadia. Expression-invariant and sparse representation for mesh-based compression for 3-D face models. In: *2013 Visual Communications and Image Processing (VCIP)*. IEEE, 2013. Available from DOI: 10.1109/vcip.2013.6706442 (cit. on pp. 36, 40, 42).

[Hou+14a] HOU, Junhui; CHAU, Lap-Pui; HE, Ying; MAGNENAT-THALMANN, Nadia. A novel compression framework for 3D time-varying meshes. In: *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, IEEE, 2014, pp. 2161–2164. Available from DOI: 10.1109/ISCAS.2014.6865596 (cit. on pp. 29, 36, 37, 39, 40, 42).

[Hou+12] HOU, Junhui; CHAU, Lap-Pui; HE, Ying; QUYNH, Dao T. P.; MAGNENAT-THALMANN, Nadia. Dynamic 3-D facial compression using low rank and sparse decomposition. In: *SIGGRAPH Asia 2012 Technical Briefs*. ACM, 2012. Available from DOI: 10.1145/2407746.2407768 (cit. on pp. 36, 40, 42).

[Hou+13b] HOU, Junhui; CHAU, Lap-Pui; HE, Ying; ZHANG, Minqi; MAGNENAT-THALMANN, Nadia. Rate-Distortion Model Based Bit Allocation for 3-D Facial Compression Using Geometry Video. *IEEE Transactions on Circuits and Systems for Video Technology*. 2013, vol. 23, no. 9, pp. 1537–1541. Available from DOI: 10.1109/tcsvt.2013.2248971 (cit. on pp. 36, 40, 42).

[Hou+15] HOU, Junhui; CHAU, Lap-Pui; MAGNENAT-THALMANN, Nadia; HE, Ying. Compressing 3-D Human Motions via Keyframe-Based Geometry Videos. *IEEE Transactions on Circuits and Systems for Video Technology*. 2015, vol. 25, no. 1, pp. 51–62. Available from DOI: `10.1109/tcsvt.2014.2329376` (cit. on pp. 37, 40, 42).

[Hou+17] HOU, Junhui; CHAU, Lap-Pui; MAGNENAT-THALMANN, Nadia; HE, Ying. Sparse Low-Rank Matrix Approximation for Data Compression. *IEEE Transactions on Circuits and Systems for Video Technology*. 2017, vol. 27, no. 5, pp. 1043–1054. Available from DOI: `10.1109/tcsvt.2015.2513698` (cit. on pp. 25, 60).

[Hou+14b] HOU, Junhui; CHAU, Lap-Pui; ZHANG, Minqi; MAGNENAT-THALMANN, Nadia; HE, Ying. A Highly Efficient Compression Framework for Time-Varying 3-D Facial Expressions. *IEEE Transactions on Circuits and Systems for Video Technology*. 2014, vol. 24, no. 9, pp. 1541–1553. Available from DOI: `10.1109/tcsvt.2014.2313890` (cit. on pp. 36, 40, 42).

[HDV19] HRUDA, L.; DVOŘÁK, J.; VÁŠA, L. On evaluating consensus in RANSAC surface registration. *Computer Graphics Forum*. 2019, vol. 38, no. 5, pp. 175–186. Available from DOI: `10.1111/cgf.13798` (cit. on p. 90).

[Hua+22] HUANG, Chao; ZHANG, Xiang; TIAN, Jun; XU, Xiaozhong; LIU, Shan. Boundary-Preserved Geometry Video for Dynamic Mesh Coding. In: *2022 Picture Coding Symposium (PCS)*. IEEE, 2022. Available from DOI: `10.1109/pcs56426.2022.10018051` (cit. on pp. 29, 38, 40, 43, 54).

[HBI13] HUANG, Chun-Hao; BOYER, Edmond; ILIC, Slobodan. Robust Human Body Shape and Pose Tracking. In: *2013 International Conference on 3D Vision*. IEEE, 2013. Available from DOI: `10.1109/3dv.2013.45` (cit. on p. 81).

[Hua+16] HUANG, Chun-Hao et al. Volumetric 3D Tracking by Detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. Available from DOI: `10.1109/cvpr.2016.419` (cit. on p. 82).

[Hua+18] HUANG, Chun-Hao Paul et al. Tracking-by-Detection of 3D Human Shapes: From Surfaces to Volumes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2018, vol. 40, no. 8, pp. 1994–2008. Available from DOI: `10.1109/tpami.2017.2740308` (cit. on p. 82).

[Huf52] HUFFMAN, D. A. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*. 1952, vol. 40, no. 9, pp. 1098–1101. Available from DOI: `10.1109/JRPROC.1952.273898` (cit. on pp. 11, 73).

[Huw+16] HUWALD, J.; RICHTER, S.; IBRAHIM, B.; DITTRICH, P. Compressing molecular dynamics trajectories: Breaking the one-bit-per-sample barrier. *Journal of Computational Chemistry*. 2016, pp. 1897–1906. Available from DOI: `10.1002/jcc.24405` (cit. on pp. 68, 77).

[ISO21a] ISO/IEC 23090-5:2021. *Information technology — Coded representation of immersive media — Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)*. 2021. International Organization for Standardization, Geneva, Switzerland (cit. on p. 38).

[ISO21b] ISO/IEC JTC 1/SC 29/AG 03. *White paper on MPEG Immersive video*. 2021. MPEG/N59 (cit. on p. 38).

[ISO17] ISO/IEC JTC 1/SC 29/WG 11. *Call for proposals for point cloud compresssion V2*. 2017. MPEG2017/N16763 (cit. on p. 50).

[ISO19] ISO/IEC JTC 1/SC 29/WG 11. *V-PCC Codec Description v8*. 2019. MPEG/N18892 (cit. on pp. 37, 38, 45, 50, 52, 55).

[ISO20] ISO/IEC JTC 1/SC 29/WG 7. *G-PCC Codec Description v9*. 2020. MPEG/N0011 (cit. on pp. 49, 50).

[ISO21c] ISO/IEC JTC 1/SC 29/WG 7. *CfP for Dynamic Mesh Coding*. 2021. MPEG/N231 (cit. on pp. 38, 39).

[JP+18] JACOBSON, Alec; PANOZZO, Daniele, et al. *libigl: A simple C++ geometry processing library*. 2018. https://libigl.github.io/ (cit. on p. 88).

[JJ81] JAIN, Jaswant; JAIN, Anil. Displacement Measurement and Its Application in Interframe Image Coding. *IEEE Transactions on communications*. 1981, vol. 29, no. 12, pp. 1799–1808. ISSN 0096-2244. Available from DOI: `10.1109/tcom.1981.1094950` (cit. on p. 31).

[Ju+02] JU, Tao; LOSASSO, Frank; SCHAEFER, Scott; WARREN, Joe. Dual contouring of hermite data. *ACM Transactions on Graphics*. 2002, vol. 21, no. 3, pp. 339–346. Available from DOI: `10.1145/566654.566586` (cit. on p. 109).

[Káč23] KÁČEREKOVÁ, Zuzana. *A system for editing triangle mesh sequences with time-varying connectivity*. Pilsen, Czech Republic, 2023. Master's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor Doc. Ing. Libor Váša, Ph.D. (cit. on pp. 111, 112).

[Kam+12] KAMMERL, Julius et al. Real-time compression of point cloud streams. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE, IEEE, 2012, pp. 778–785. Available from DOI: `10.1109/ICRA.2012.6224647` (cit. on pp. 22, 45–47, 49).

[KG04] KARNI, Zachi; GOTSMAN, Craig. Compression of soft-body animation sequences. *Computers & Graphics*. 2004, vol. 28, no. 1, pp. 25–34. ISSN 0097-8493. Available from DOI: `10.1016/j.cag.2003.10.002` (cit. on pp. 15, 63).

[Kat+17] KATHARIYA, B.; KARTHIK, A.; LI, Z.; JOSHI, R. Embedded binary tree for dynamic point cloud geometry compression with graph signal resampling and prediction. In: *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2017, pp. 1–4. Available from DOI: `10.1109/VCIP.2017.8305130` (cit. on pp. 45, 49).

[KT22] KAYA, Emre C.; TABUS, Ioan. Lossless Compression of Point Cloud Sequences Using Sequence Optimized CNN Models. *IEEE Access*. 2022, vol. 10, pp. 83678–83691. ISSN 2169-3536. Available from DOI: `10.1109/access.2022.3197295` (cit. on pp. 45, 48, 52–55).

[KH13] KAZHDAN, Michael; HOPPE, Hugues. Screened poisson surface reconstruction. *ACM Transactions on Graphics*. 2013, vol. 32, no. 3, pp. 1–13. Available from DOI: `10.1145/2487228.2487237` (cit. on p. 46).

[Kim+20] KIM, J.; IM, J.; RHYU, S.; KIM, K. 3D Motion Estimation and Compensation Method for Video-Based Point Cloud Compression. *IEEE Access*. 2020, vol. 8, pp. 83538–83547. Available from DOI: `10.1109/ACCESS.2020.2991478` (cit. on pp. 45, 51, 55).

[KG06] KIRCHER, Scott; GARLAND, Michael. Editing arbitrarily deforming surface animations. *ACM Transactions on Graphics*. 2006, vol. 25, no. 3, pp. 1098–1107. Available from DOI: `10.1145/1141911.1142000` (cit. on p. 111).

[Koc+19] KOCH, Sebastian et al. ABC: A Big CAD Model Dataset for Geometric Deep Learning. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. Available from DOI: `10.1109/cvpr.2019.00983` (cit. on p. 128).

[Kog81] KOGA, T. Motion-compensated interframe coding for video conferencing. *Proc. Nat. Telecommunications Conf., 1981*. 1981, G5.3.1–G5.3.3 (cit. on p. 49).

[Kuh55] KUHN, Harold W. The Hungarian method for the assignment problem. *Naval research logistics quarterly*. 1955, vol. 2, no. 1-2, pp. 83–97. Available from DOI: `10.1007/978-3-540-68279-0_2` (cit. on p. 85).

[Lal+17] LALOS, Aris S.; VASILAKIS, Andreas A.; DIMAS, Anastasios; MOUSTAKAS, Konstantinos. Adaptive compression of animated meshes by exploiting orthogonal iterations. *The Visual Computer*. 2017, vol. 33, no. 6-8, pp. 811–821. Available from DOI: 10.1007/s00371-017-1395-4 (cit. on pp. 26, 27).

[LF19] LASSERRE, Sébastien; FLYNN, David. *Point Cloud Compression in MPEG and beyond*. Inria, Rennes, France, 2019 (cit. on p. 50).

[Lav11] LAVOUÉ, Guillaume. A Multiscale Metric for 3D Mesh Visual Quality Assessment. *Computer Graphics Forum*. 2011, vol. 30, no. 5, pp. 1427–1437. Available from DOI: 10.1111/j.1467-8659.2011.02017.x (cit. on p. 15).

[LJF13] LEANDRO, Jorge De Jesus Gomes; JR, Roberto Marcondes Cesar; FERIS, Rogerio Schmidt. Shape Analysis Using the Spectral Graph Wavelet Transform. In: *2013 IEEE 9th International Conference on e-Science*. IEEE, 2013. Available from DOI: 10.1109/escience.2013.45 (cit. on p. 26).

[Lew+06] LEWINER, Thomas et al. GEncode: Geometry-driven compression for General Meshes. *Computer Graphics Forum*. 2006, vol. 25, no. 4, pp. 685–695. Available from DOI: 10.1111/j.1467-8659.2006.00990.x (cit. on p. 41).

[Li+09] LI, Hao; ADAMS, Bart; GUIBAS, Leonidas J.; PAULY, Mark. Robust Single-View Geometry and Motion Reconstruction. *ACM Trans. Graph.* 2009, vol. 28, no. 5, pp. 1–10. ISSN 0730-0301. Available from DOI: 10.1145/1618452.1618521 (cit. on pp. 32, 81).

[Li+20] LI, L.; LI, Z.; ZAKHARCHENKO, V.; CHEN, J.; LI, H. Advanced 3D Motion Prediction for Video-Based Dynamic Point Cloud Compression. *IEEE Transactions on Image Processing*. 2020, vol. 29, pp. 289–302. Available from DOI: 10.1109/TIP.2019.2931621 (cit. on pp. 45, 51, 52).

[LKB10] LIEN, Jyh-Ming; KURILLO, Gregorij; BAJCSY, Ruzena. Multi-camera tele-immersion system with real-time model driven data compression. *The Visual Computer*. 2010, vol. 26, no. 1, pp. 3–15. Available from DOI: 10.1007/s00371-009-0367-8 (cit. on pp. 45, 46).

[LHS10] LIN, Sheng-Fuu; HSIN, Hsi-Chin; SU, Chien-Kun. Hybrid Image Compression Based on Set-Partitioning Embedded Block Coder and Residual Vector Quantization. *Journal of Information Science and Engineering*. 2010, vol. 26, no. 3, pp. 1011–1027. Available from DOI: 10.6688/JISE.2010.26.3.18 (cit. on p. 26).

[Liu+19] LIU, Jianqiang; YAO, Jian; TU, Jingmin; CHENG, Junhao. Data-Adaptive Packing Method for Compression of Dynamic Point Cloud Sequences. In: *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019. Available from DOI: `10.1109/icme.2019.00160` (cit. on pp. 45, 51, 52).

[Llo82] LLOYD, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory*. 1982, vol. 28, no. 2, pp. 129–137. Available from DOI: `10.1109/TIT.1982.1056489` (cit. on pp. 84, 89).

[LV14] LOBAZ, P.; VÁŠA, L. Hierarchical Laplacian-based compression of triangle meshes. *Graphical Models*. 2014, vol. 76, no. 6, pp. 682–690. ISSN 1524-0703. Available from DOI: `10.1016/j.gmod.2014.09.003` (cit. on p. 61).

[LC87] LORENSEN, William E.; CLINE, Harvey E. Marching cubes: A high resolution 3D surface construction algorithm. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. ACM, 1987. Available from DOI: `10.1145/37401.37422` (cit. on pp. 92, 109).

[Luo+21] LUO, Guoliang; ZHAO, Xin; CHEN, Qiang; ZHU, Zhiliang; XIAN, Chuhua. Dynamic data reshaping for 3D mesh animation compression. *Multimedia Tools and Applications*. 2021, vol. 81, no. 1, pp. 55–72. Available from DOI: `10.1007/s11042-021-10629-1` (cit. on p. 27).

[Luo+19] LUO, Guoliang et al. 3D Mesh Animation Compression Based on Adaptive Spatio-Temporal Segmentation. In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. Montreal, Quebec, Canada: ACM, 2019. I3D '19. ISBN 9781450363105. Available from DOI: `10.1145/3306131.3317017` (cit. on p. 26).

[Luo+20] LUO, Guoliang et al. Spatio-Temporal Segmentation Based Adaptive Compression of Dynamic Mesh Sequences. *ACM Transactions on Multimedia Computing, Communications, and Applications*. 2020, vol. 16, no. 1, pp. 1–24. ISSN 1551-6857. Available from DOI: `10.1145/3377475` (cit. on p. 27).

[MYA08] MAEDA, Takashi; YAMASAKI, Toshihiko; AIZAWA, Kiyoharu. Model-Based Analysis and Synthesis of Time-Varying Mesh. In: PERALES, Francisco J.; FISHER, Robert B. (eds.). *Articulated Motion and Deformable Objects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 112–121. ISBN 978-3-540-70517-8. Available from DOI: `10.1007/978-3-540-70517-8_12` (cit. on pp. 29, 32, 34, 40, 42).

[Mag+15] MAGLO, Adrien; LAVOUÉ, Guillaume; DUPONT, Florent; HUDELOT, Céline. 3D Mesh Compression: Survey, Comparisons, and Emerging Trends. *ACM Comput. Surv.* 2015, vol. 47, no. 3, pp. 1–41. ISSN 0360-0300. Available from DOI: `10.1145/2693443` (cit. on p. 25).

[Mam+22] MAMMOU, Khaled; KIM, Jungsun; TOURAPIS, Alexis M.; PODBORSKI, Dimitri; FLYNN, David. Video and Subdivision based Mesh Coding. In: *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2022. Available from DOI: `10.1109/euvip53989.2022.9922888` (cit. on pp. 29, 39, 40, 43, 44, 54).

[MZP09] MAMMOU, Khaled; ZAHARIA, Titus; PRÊTEUX, Françoise. TFAN: A low complexity 3D mesh compression algorithm. *Computer Animation and Virtual Worlds*. 2009, vol. 20, no. 2-3, pp. 343–354. Available from DOI: `10.1002/cav.319` (cit. on pp. 37, 40, 41, 132).

[MGS07] MARAIS, P.; GAIN, J.; SHREINER, D. Distance-Ranked Connectivity Compression of Triangle Meshes. *Computer Graphics Forum*. 2007, vol. 26, no. 4, pp. 813–823. Available from DOI: `10.1111/j.1467-8659.2007.01026.x` (cit. on pp. 41, 117, 118, 126, 127, 129–131).

[MSW03] MARPE, D.; SCHWARZ, H.; WIEGAND, T. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*. 2003, vol. 13, no. 7, pp. 620–636. Available from DOI: `10.1109/tcsvt.2003.815173` (cit. on pp. 12, 119, 126).

[Mar+22] MARVIE, Jean-Eudes et al. Compression of Time-Varying Textured Meshes using Patch Tiling and Image-based Tracking. In: *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2022. Available from DOI: `10.1109/euvip53989.2022.9922890` (cit. on pp. 38, 40, 43, 54).

[MBC16] MEKURIA, Rufael; BLOM, Kees; CESAR, Pablo. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Transactions on Circuits and Systems for Video Technology*. 2016, vol. 27, no. 4, pp. 828–842. Available from DOI: `10.1109/TCSVT.2016.2543039` (cit. on pp. 45, 49, 50, 53).

[MMG06] MERRY, Bruce; MARAIS, Patrick; GAIN, James. Compression of Dense and Regular Point Clouds. In: *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. Cape Town, South Africa: Association for Computing Machinery, 2006, pp. 15–20. AFRIGRAPH '06. ISBN 1595932887. Available from DOI: `10.1145/1108590.1108593` (cit. on pp. 131, 132).

[Mes+19] MESCHEDER, Lars; OECHSLE, Michael; NIEMEYER, Michael; NOWOZIN, Sebastian; GEIGER, Andreas. Occupancy Networks: Learning 3D Reconstruction in Function Space. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4455–4465. Available from DOI: `10.1109/CVPR.2019.00459` (cit. on p. 34).

[MPL20] MILANI, S.; POLO, E.; LIMUTI, S. A Transform Coding Strategy for Dynamic Point Clouds. *IEEE Transactions on Image Processing*. 2020, vol. 29, pp. 8213–8225. Available from DOI: `10.1109/TIP.2020.3011811` (cit. on pp. 45, 48, 54).

[Moy+21] MOYNIHAN, Matthew; RUANO, Susana; PAGES, Rafael; SMOLIC, Aljosa. Autonomous Tracking For Volumetric Video Sequences. In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2021. Available from DOI: `10.1109/wacv48630.2021.00170` (cit. on p. 81).

[MPE22] MPEG. *MPEG 138* [online]. 2022-04. [visited on 2023-05-30]. Available from: `https://www.mpeg.org/meetings/mpeg-138/` (cit. on p. 39).

[MC21] MUNTONI, Alessandro; CIGNONI, Paolo. *PyMeshLab*. Zenodo, 2021. Available from DOI: `10.5281/ZENODO.4438750` (cit. on p. 127).

[MS10] MYRONENKO, Andriy; SONG, Xubo. Point Set Registration: Coherent Point Drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2010, vol. 32, no. 12, pp. 2262–2275. Available from DOI: `10.1109/tpami.2010.46` (cit. on p. 81).

[NYA10] NAKAGAWA, S.; YAMASAKI, T.; AIZAWA, K. Deformation-based data reduction of Time-Varying Meshes for displaying on mobile terminals. In: *2010 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video*. IEEE, 2010, pp. 1–4. Available from DOI: `10.1109/3DTV.2010.5506509` (cit. on pp. 34, 40, 42).

[NO13] NARANG, Sunil K.; ORTEGA, Antonio. Compact Support Biorthogonal Wavelet Filterbanks for Arbitrary Undirected Graphs. *IEEE Transactions on Signal Processing*. 2013, vol. 61, no. 19, pp. 4673–4685. Available from DOI: `10.1109/tsp.2013.2273197` (cit. on p. 46).

[Nie+19] NIEMEYER, Michael; MESCHEDER, Lars; OECHSLE, Michael; GEIGER, Andreas. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019. Available from DOI: `10.1109/ICCV.2019.00548` (cit. on pp. 44, 82, 92).

[Ort+16] ORTS-ESCOLANO, Sergio et al. Holoportation: Virtual 3D Teleportation in Real-Time. In: Tokyo, Japan: Association for Computing Machinery, 2016, pp. 741–754. UIST '16. ISBN 9781450341899. Available from DOI: `10.1145/2984511.2984517` (cit. on pp. 8, 44).

[PMR20] PEIXOTO, E.; MEDEIROS, E.; RAMALHO, E. Silhouette 4d: An Inter-Frame Lossless Geometry Coder Of Dynamic Voxelized Point Clouds. In: *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 2691–2695. Available from DOI: `10.1109/ICIP40778.2020.9190648` (cit. on pp. 48, 52, 54).

[PK05] PENG, Jingliang; KUO, C.-C. Jay. Geometry-Guided Progressive Lossless 3D Mesh Coding with Octree (OT) Decomposition. In: *ACM SIGGRAPH 2005 Papers*. Los Angeles, California: Association for Computing Machinery, 2005, pp. 609–616. SIGGRAPH '05. ISBN 9781450378253. Available from DOI: `10.1145/1186822.1073237` (cit. on p. 41).

[PP93] PINKALL, Ulrich; POLTHIER, Konrad. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics*. 1993, vol. 2, no. 1, pp. 15–36. Available from DOI: `10.1080/10586458.1993.10504266` (cit. on pp. 14, 60).

[Pot+06] POTTMANN, Helmut; HUANG, Qi-Xing; YANG, Yong-Liang; HU, Shi-Min. Geometry and Convergence Analysis of Algorithms for Registration of 3D Shapes. *International Journal of Computer Vision*. 2006, vol. 67, no. 3, pp. 277–296. Available from DOI: `10.1007/s11263-006-5167-2` (cit. on p. 90).

[Pra+17] PRADA, Fabián; KAZHDAN, Misha; CHUANG, Ming; COLLET, Alvaro; HOPPE, Hugues. Spatiotemporal atlas parameterization for evolving meshes. *ACM Transactions on Graphics*. 2017, vol. 36, no. 4, pp. 1–12. Available from DOI: `10.1145/3072959.3073679` (cit. on p. 81).

[QC17] QUEIROZ, Ricardo L. de; CHOU, Philip A. Motion-Compensated Compression of Dynamic Voxelized Point Clouds. *IEEE Transactions on Image Processing*. 2017, vol. 26, no. 8, pp. 3886–3895. Available from DOI: `10.1109/TIP.2017.2707807` (cit. on pp. 45, 49, 53).

[Que+18] QUEIROZ, Ricardo L. de; GARCIA, Diogo C.; CHOU, Philip A.; FLORENCIO, Dinei A. Distance-Based Probability Model for Octree Coding. *IEEE Signal Processing Letters*. 2018, vol. 25, no. 6, pp. 739–742. Available from DOI: `10.1109/lsp.2018.2823701` (cit. on pp. 45, 47, 52, 54).

[RPM21] RAMALHO, Evaristo; PEIXOTO, Eduardo; MEDEIROS, Edil. Silhouette 4D With Context Selection: Lossless Geometry Compression of Dynamic Point Clouds. *IEEE Signal Processing Letters*. 2021, vol. 28, pp. 1660–1664. Available from DOI: `10.1109/lsp.2021.3102525` (cit. on pp. 45, 48, 52, 54, 55).

[Ros99] ROSSIGNAC, J. Edgebreaker: connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*. 1999, vol. 5, no. 1, pp. 47–61. Available from DOI: `10.1109/2945.764870` (cit. on pp. 37, 40, 41, 62, 127).

[SP96] SAID, A.; PEARLMAN, W. A. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*. 1996, vol. 6, no. 3, pp. 243–250. Available from DOI: `10.1109/76.499834` (cit. on p. 26).

[San+03] SANDER, P. V.; WOOD, Z. J.; GORTLER, S. J.; SNYDER, J.; HOPPE, H. Multi-Chart Geometry Images. In: *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. Aachen, Germany: Eurographics Association, 2003, pp. 146–155. SGP '03. ISBN 1581136870 (cit. on p. 38).

[San+21] SANTOS, Cristiano; GONCALVES, Mateus; CORREA, Guilherme; PORTO, Marcelo. Block-Based Inter-Frame Prediction For Dynamic Point Cloud Compression. In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021. Available from DOI: `10.1109/icip42928.2021.9506355` (cit. on pp. 45, 49, 53, 55).

[SC17] SAWHNEY, Rohan; CRANE, Keenan. Boundary First Flattening. *ACM Transactions on Graphics*. 2017, vol. 37, no. 1, pp. 1–14. Available from DOI: `10.1145/3132705` (cit. on p. 38).

[Sch+19] SCHWARZ, Sebastian; SHEIKHIPOUR, Nahid; FAKOUR SEVOM, Vida; HANNUKSELA, Miska M. Video coding of dynamic 3D point cloud data. *APSIPA Transactions on Signal and Information Processing*. 2019, vol. 8, no. 1, e31. Available from DOI: `10.1017/ATSIP.2019.24` (cit. on pp. 45, 51).

[SSC19] SHARP, Nicholas; SOLIMAN, Yousuf; CRANE, Keenan. The Vector Heat Method. *ACM Transactions on Graphics*. 2019, vol. 38, no. 3, pp. 1–19. Available from DOI: `10.1145/3243651` (cit. on p. 53).

[SL22] SHI, Haoyu; LI, Fan. Patch Re-Segmentation and Packing for Dynamic Point Cloud Compression via Back-and-Forth Structure. *IEEE Open Journal of Signal Processing*. 2022, vol. 3, pp. 155–168. Available from DOI: `10.1109/ojsp.2022.3160392` (cit. on pp. 45, 51, 52).

[SBI17] SLAVCHEVA, Miroslava; BAUST, Maximilian; ILIC, Slobodan. Towards Implicit Correspondence in Signed Distance Field Evolution. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, 2017. Available from DOI: 10.1109/iccvw.2017.103 (cit. on p. 82).

[SCT03] SORKINE, Olga; COHEN-OR, Daniel; TOLEDO, Sivan. High-Pass Quantization for Mesh Encoding. In: *Symposium on Geometry Processing*. The Eurographics Association, 2003, vol. 42. Available from DOI: 10.2312/SGP/SGP03/042-051 (cit. on pp. 15, 59, 61–63).

[SR16] SORKINE-HORNUNG, Olga; RABINOVICH, Michael. *Least-Squares Rigid Motion Using SVD*. 2016. Technical note (cit. on p. 88).

[SLS11] SPÅNGBERG, Daniel; LARSSON, Daniel S. D.; SPOEL, David van der. Trajectory NG: portable, compressed, general molecular dynamics trajectories. *Journal of Molecular Modeling*. 2011, vol. 17, no. 10, pp. 2669–2685. Available from DOI: 10.1007/s00894-010-0948-5 (cit. on pp. 68, 77).

[Str97] STROBACH, Peter. Fast recursive orthogonal iteration subspace tracking algorithms and applications. *Signal Processing*. 1997, vol. 59, no. 1, pp. 73–100. Available from DOI: 10.1016/s0165-1684(97)00039-x (cit. on p. 26).

[Sul+12] SULLIVAN, G. J.; OHM, J.; HAN, W.; WIEGAND, T. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology*. 2012, vol. 22, no. 12, pp. 1649–1668. Available from DOI: 10.1109/TCSVT.2012.2221191 (cit. on p. 51).

[SP04] SUMNER, Robert W.; POPOVIĆ, Jovan. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*. 2004, vol. 23, no. 3, pp. 399–405. Available from DOI: 10.1145/1015706.1015736 (cit. on p. 26).

[SSP07] SUMNER, Robert W.; SCHMID, Johannes; PAULY, Mark. Embedded Deformation for Shape Manipulation. In: *ACM SIGGRAPH 2007 Papers*. San Diego, California: Association for Computing Machinery, 2007, 80–es. SIGGRAPH ’07. ISBN 9781450378369. Available from DOI: 10.1145/1275808.1276478 (cit. on p. 110).

[Tev+12] TEVS, Art et al. Animation cartography—intrinsic reconstruction of shape and motion. *ACM Trans. Graph.* 2012, vol. 31, no. 2, pp. 1–15. ISSN 0730-0301. Available from DOI: 10.1145/2159516.2159517 (cit. on p. 18).

[TCF16] THANOU, Dorina; CHOU, Philip A.; FROSSARD, Pascal. Graph-based compression of dynamic 3D point cloud sequences. *IEEE Transactions on Image Processing*. 2016, vol. 25, no. 4, pp. 1765–1778. Available from DOI: `10.1109/TIP.2016.2529506` (cit. on pp. 45, 49, 54).

[TWC14] TORKHANI, Fakhri; WANG, Kai; CHASSERY, Jean-Marc. A Curvature-Tensor-Based Perceptual Quality Metric for 3D Triangular Meshes. *Machine GRAPHICS & VISION*. 2014, vol. 23, no. 1/2, pp. 59–82. Available from DOI: `10.22630/MGV.2014.23.1.4` (cit. on p. 15).

[TG98] TOUMA, Costa; GOTSMAN, Craig. Triangle Mesh Compression. In: *Proceedings of the Graphics Interface 1998 Conference, June 18-20, 1998, Vancouver, BC, Canada*. 1998, pp. 26–34 (cit. on pp. 14, 60).

[TM12] TUNG, T.; MATSUYAMA, T. Topology Dictionary for 3D Video Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2012, vol. 34, no. 8, pp. 1645–1657. Available from DOI: `10.1109/tpami.2011.258` (cit. on pp. 29, 34, 40, 42, 43).

[TM13] TUNG, Tony; MATSUYAMA, Takashi. Invariant Surface-Based Shape Descriptor for Dynamic Surface Encoding. In: LEE, Kyoung Mu; MATSUSHITA, Yasuyuki; REHG, James M.; HU, Zhanyi (eds.). *Computer Vision – ACCV 2012*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 486–499. ISBN 978-3-642-37331-2. Available from DOI: `10.1007/978-3-642-37331-2_37` (cit. on pp. 29, 36, 37, 40, 42).

[TS05] TUNG, Tony; SCHMITT, Francis. The Augmented Multiresolution Reeb Graph Approach for Content-based Retrieval of 3d Shapes. *International Journal of Shape Modeling*. 2005, vol. 11, no. 01, pp. 91–120. Available from DOI: `10.1142/s0218654305000748` (cit. on p. 33).

[TSM07] TUNG, Tony; SCHMITT, Francis; MATSUYAMA, Takashi. Topology matching for 3D video compression. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007. Available from DOI: `10.1109/cvpr.2007.383294` (cit. on pp. 29, 33, 40, 42, 43).

[VL08] VALLET, B.; LÉVY, B. Spectral Geometry Processing with Manifold Harmonics. *Computer Graphics Forum*. 2008, vol. 27, no. 2, pp. 251–260. Available from DOI: `10.1111/j.1467-8659.2008.01122.x` (cit. on p. 26).

[VB13] VÁŠA, L.; BRUNNETT, G. Exploiting Connectivity to Improve the Tangential Part of Geometry Prediction. *IEEE Transactions on Visualization and Computer Graphics*. 2013, vol. 19, no. 09, pp. 1467–1475. ISSN 1941-0506. Available from DOI: `10.1109/TVCG.2013.22` (cit. on pp. 39, 60, 62, 63, 112, 131, 132).

[Váš+14] VÁŠA, L.; MARRAS, S.; HORMANN, K.; BRUNNETT, G. Compressing dynamic meshes with geometric laplacians. *Computer Graphics Forum*. 2014, vol. 33, no. 2, pp. 145–154. Available from DOI: 10.1111/cgf.12304 (cit. on pp. 60, 63).

[VP11] VÁŠA, L.; PETŘÍK, O. Optimising Perceived Distortion in Lossy Encoding of Dynamic Meshes. *Computer Graphics Forum*. 2011, vol. 30, no. 5, pp. 1439–1449. Available from DOI: 10.1111/j.1467-8659.2011.02018.x (cit. on pp. 15, 60, 63).

[VS09] VÁŠA, L.; SKALA, V. COBRA: Compression of the Basis for PCA Represented Animations. *Computer Graphics Forum*. 2009, vol. 28, no. 6, pp. 1529–1540. Available from DOI: 10.1111/j.1467-8659.2008.01304.x (cit. on pp. 13, 83, 109, 112).

[VD18] VÁŠA, Libor; DVOŘÁK, Jan. Error propagation control in Laplacian mesh compression. *Computer Graphics Forum*. 2018, vol. 37, no. 5, pp. 61–70. Available from DOI: 10.1111/cgf.13491 (cit. on pp. 8, 59).

[VR12] VÁŠA, Libor; RUS, Jan. Dihedral Angle Mesh Error: a fast perception correlated distortion measure for fixed connectivity triangle meshes. *Computer Graphics Forum*. 2012, vol. 31, no. 5, pp. 1715–1724. Available from DOI: 10.1111/j.1467-8659.2012.03176.x (cit. on pp. 15, 62).

[VS07] VÁŠA, Libor; SKALA, Vaclav. CODDYAC: Connectivity Driven Dynamic Mesh Compression. In: *2007 3DTV Conference*. IEEE, 2007, pp. 1–4. Available from DOI: 10.1109/3DTV.2007.4379408 (cit. on p. 18).

[VS11] VÁŠA, Libor; SKALA, Vaclav. A Perception Correlated Comparison Method for Dynamic Meshes. *IEEE Transactions on Visualization and Computer Graphics*. 2011, vol. 17, no. 2, pp. 220–230. Available from DOI: 10.1109/TVCG.2010.38 (cit. on pp. 15, 63, 111).

[Vla+08] VLASIC, Daniel; BARAN, Ilya; MATUSIK, Wojciech; POPOVIĆ, Jovan. Articulated Mesh Animation from Multi-View Silhouettes. In: *ACM SIGGRAPH 2008 papers*. Los Angeles, California: Association for Computing Machinery, 2008. SIGGRAPH '08. ISBN 9781450301121. Available from DOI: 10.1145/1399504.1360696 (cit. on pp. 18, 63, 92).

[WTM12] WANG, Kai; TORKHANI, Fakhri; MONTANVERT, Annick. A fast roughness-based approach to the assessment of 3D mesh visual quality. *Computers & Graphics*. 2012, vol. 36, no. 7, pp. 808–818. ISSN 0097-8493. Available from DOI: 10.1016/j.cag.2012.06.004. Augmented Reality Computer Graphics in China (cit. on p. 15).

[Wuh+15] WUHRER, Stefanie; LANG, Jochen; TEKIEH, Motahareh; SHU, Chang. Finite element based tracking of deforming surfaces. *Graphical Models*. 2015, vol. 77, pp. 1–17. Available from DOI: `10.1016/j.gmod.2014.10.002` (cit. on p. 82).

[Xia+10] XIA, Jiazhi; HE, Ying; QUYNH, Dao P. T.; CHEN, Xiaoming; HOI, Steven C. H. Modeling 3D facial expressions using geometry videos. In: *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010. Available from DOI: `10.1145/1873951.1874010` (cit. on pp. 29, 35, 39, 40, 42).

[Xia+12] XIA, Jiazhi; QUYNH, Dao Thi Phuong; HE, Ying; CHEN, Xiaoming; HOI, Steven C. H. Modeling and Compressing 3-D Facial Expressions Using Geometry Videos. *IEEE Transactions on Circuits and Systems for Video Technology*. 2012, vol. 22, no. 1, pp. 77–90. Available from DOI: `10.1109/tcsvt.2011.2158337` (cit. on pp. 36, 40, 42).

[YA10] YAMASAKI, Toshihiko; AIZAWA, Kiyoharu. Patch-based compression for time-varying meshes. In: *2010 IEEE International Conference on Image Processing*. IEEE, IEEE, 2010, pp. 3433–3436. Available from DOI: `10.1109/ICIP.2010.5652911` (cit. on pp. 29, 31, 34, 40, 43, 53, 112).

[Yan+18a] YANG, Bailin et al. Compressed dynamic mesh sequence for progressive streaming. *Computer Animation and Virtual Worlds*. 2018, vol. 30, no. 6. Available from DOI: `10.1002/cav.1847` (cit. on pp. 26, 27).

[Yan+19] YANG, Bailin et al. Motion-Aware Compression and Transmission of Mesh Animation Sequences. *ACM Transactions on Intelligent Systems and Technology*. 2019, vol. 10, no. 3, pp. 1–21. ISSN 2157-6904. Available from DOI: `10.1145/3300198` (cit. on pp. 26, 27).

[Yan+18b] YANG, Chenguang; WANG, Zunran; HE, Wei; LI, Zhijun. Development of a fast transmission method for 3D point cloud. *Multimedia Tools and Applications*. 2018, vol. 77, no. 19, pp. 25369–25387. Available from DOI: `10.1007/s11042-018-5789-8` (cit. on pp. 45, 46).

[YKL06] YANG, Jeong-Hyu; KIM, Chang-Su; LEE, Sang-Uk. Semi-regular representation and progressive compression of 3-D dynamic mesh sequences. *IEEE Transactions on Image Processing*. 2006, vol. 15, no. 9, pp. 2531–2544. Available from DOI: `10.1109/TIP.2006.877413` (cit. on pp. 29, 33, 40, 43, 54).

[YXF14] YANG, Long; XIAO, Chunxia; FANG, Jun. Multi-scale geometric detail enhancement for time-varying surfaces. *Graphical Models*. 2014, vol. 76, no. 5, pp. 413–425. Available from DOI: `10.1016/j.gmod.2014.03.010` (cit. on p. 111).

[Yu+14] YU, Wuyi; ZHANG, Kang; WAN, Shenghua; LI, Xin. Optimizing polycube domain construction for hexahedral remeshing. *Computer-Aided Design*. 2014, vol. 46, pp. 58–68. Available from DOI: `10.1016/j.cad.2013.08.018` (cit. on p. 42).

[Yua+21] YUAN, Yu-Jie; LAI, Yu-Kun; WU, Tong; GAO, Lin; LIU, Ligang. A Revisit of Shape Editing Techniques: From the Geometric to the Neural Viewpoint. *Journal of Computer Science and Technology*. 2021, vol. 36, no. 3, pp. 520–554. Available from DOI: `10.1007/s11390-021-1414-9` (cit. on p. 111).

[ZGT23] ZAGHETTO, Alexandre; GRAZIOSI, Danillo; TABATABAI, Ali. *Task-oriented dynamic mesh compression using occupancy networks*. Google Patents, 2023. No. US20230016302A1. Available also from: `https://patents.google.com/patent/US20230016302A1`. US Patent App. 17/861,033 (cit. on pp. 29, 33, 34, 39, 40, 43, 44).

[Zha+05] ZHANG, J.; KAPLOW, R.; CHEN, R.; SIDDIQI, K. *The McGill Shape Benchmark*. 2005. Available also from: `http://www.cim.mcgill.ca/~shape/benchMark/` (cit. on p. 128).

[ZJ16] ZHOU, Qingnan; JACOBSON, Alec. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797*. 2016. Available from DOI: `10.48550/ARXIV.1605.04797` (cit. on pp. 62, 63, 128).

[Zhu+21] ZHU, Wenjie; MA, Zhan; XU, Yiling; LI, Li; LI, Zhu. View-Dependent Dynamic Point Cloud Compression. *IEEE Transactions on Circuits and Systems for Video Technology*. 2021, vol. 31, no. 2, pp. 765–781. Available from DOI: `10.1109/tcsvt.2020.2985911` (cit. on pp. 35, 45, 51, 55).

Authorship contribution statement

Error propagation control in Laplacian mesh compression.
Váša 50%, Dvořák 50%
I implemented and experimented with a novel discretization of Laplace-Beltrami operator, which I presented in my master's thesis [Dvo18]. Its application to Laplacian mesh compression was planned to be one of the major contributions of the paper, but, in the end, we rejected this idea due to the poor conditionality of its Laplacian matrix.

Predictive compression of molecular dynamics trajectories.
Dvořák 60%, Maňák 20%, Váša 20%
I implemented the method, performed the majority of experiments, wrote method description and rendered the majority of the figures in the paper.

Towards Understanding Time Varying Triangle Meshes.
Dvořák 40%, Vaněček 20%, Váša 40%
I formulated the energy function to measure the smooth movement of centers, came up with the idea to use a blending of signed-distance functions to approximate the original surface with centers, wrote related work and parts of the paper describing my contributions.

As-rigid-as-possible volume tracking for time-varying surfaces.
Dvořák 50%, Káčereková 15%, Vaněček 15%, Hruda 5%, Váša 15%
I came up with the idea of using ARAP energy and measuring center affinity based on movement, implemented these contributions, wrote a significant part of the paper, performed all the experiments, created the majority of the figures in the paper and rendered the results for accompanying video.

Priority-based encoding of triangle mesh connectivity for a known geometry.
Dvořák 40%, Káčereková 40%, Vaněček 10%, Váša 10%
Although not an author of the initial idea of the paper, I proposed to measure gate priority based on more than just the quality of the best candidate, performed all the experiments and created some of the figures. I did almost all the work revising the paper (code refactoring, reformulation of some of the geometric criteria for candidate quality, improved candidate vertex search bound, simulated annealing for parameter optimisation, and additional experiments)

Global Optimisation for Improved Volume Tracking of Time-Varying Meshes.
Dvořák 80%, Hácha 10%, Váša 10%
I came up with the initial idea, implemented all the contributions, performed all the experiments, created all the figures and wrote the majority of the paper.