

T4DT: TENSORIZING TIME FOR LEARNING TEMPORAL 3D VISUAL DATA

A PREPRINT

Mikhail Usvyatsov¹, Rafael Ballester-Ripoll², Lina Bashaeva³, Konrad Schindler¹, Gonzalo Ferrer³, and Ivan Oseledets³

¹ETH Zurich, Switzerland, {mikhailu, schindler}@ethz.ch

²IE University, Madrid, Spain, rafael.ballester@ie.edu

³Skolkovo Institute of Science and Technology, Moscow, Russia, {lina.bashaeva, g.ferrer, I.Oseledets}@skoltech.ru

August 3, 2022

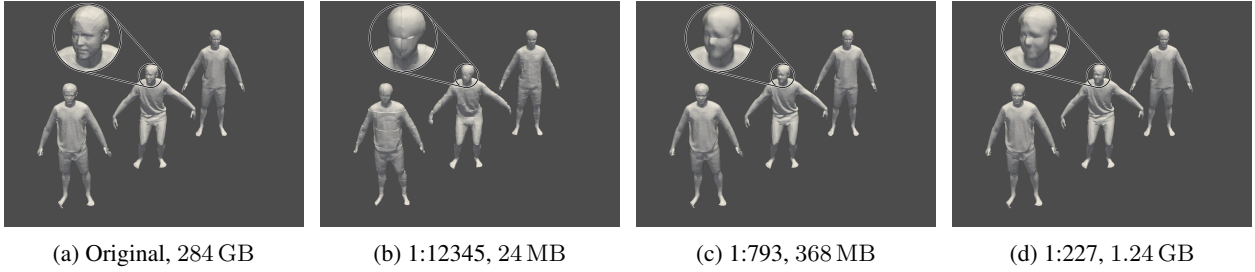


Figure 1: T4DT with different compression levels in OQTT format for a *longshort-flying-eagle* scene of resolution 512^3 with 284 time frames. Only frames 1, 142, and 284 are depicted. The compression ratio is different from the actual memory consumption due to the padding of the time dimension to 512. High compression is achieved with $r_{\max} = 400$, MSDM2 = 0.45 in Fig. 1b, medium compression with $r_{\max} = 1800$, MSDM2 = 0.32 in Fig. 1c, and low compression / high quality with $r_{\max} = 4000$, MSDM2 = 0.29 in Fig. 1d.

ABSTRACT

Unlike 2D raster images, there is no single dominant representation for 3D visual data processing. Different formats like point clouds, meshes, or implicit functions each have their strengths and weaknesses. Still, grid representations such as signed distance functions have attractive properties also in 3D. In particular, they offer constant-time random access and are eminently suitable for modern machine learning. Unfortunately, the storage size of a grid grows exponentially with its dimension. Hence they often exceed memory limits even at moderate resolution. This work explores various low-rank tensor formats, including the Tucker, tensor train, and quantics tensor train decompositions, to compress time-varying 3D data. Our method iteratively computes, voxelizes, and compresses each frame's truncated signed distance function and applies tensor rank truncation to condense all frames into a single, compressed tensor that represents the entire 4D scene. We show that low-rank tensor compression is extremely compact to store and query time-varying signed distance functions. It significantly reduces the memory footprint of 4D scenes while surprisingly preserving their geometric quality. Unlike existing iterative learning-based approaches like DeepSDF and NeRF, our method uses a closed-form algorithm with theoretical guarantees.

1 Introduction

Recent advances in hardware development have made it possible to collect rich depth datasets with commodity devices. For example, modern AR/VR hardware can record videos of 3D data, thus capturing their temporal evolution to yield

4D datasets. Nevertheless, working with 4D data presents computational challenges due to the curse of dimensionality. Furthermore, while 2D data mostly come in the form of raster images, there is an entire zoo of popular representations for 3D data, ranging from point clouds and meshes to voxel grids and implicit (neural) functions.

In this work, we address the case of representations based on regular grids. Such representations are highly structured and allow for efficient manipulation, e.g., they offer random access, slicing, and other operations in constant time. However, their storage requirements become a bottleneck: a grid of resolution I along each dimension D has I^D elements. We propose applying tensor decompositions to leverage the high temporal and spatial correlation in grids that arise from time-evolving 3D data. Besides exploring the general idea of compression via low-rank constraints, we compare different tensor decomposition schemes and their potential for 4D video. For example, although the Tucker format [Tucker \(1963, 1966\)](#) is known to yield good results for the 3D case, it still suffers from the curse of dimensionality since it requires $O(r^D)$ elements w.r.t. the Tucker rank r , while r must typically be chosen $\approx \frac{I}{\text{const.}}$ to achieve reasonable accuracy. Therefore, we explore hybrid low-rank formats based on the tensor train (TT) decomposition [Oseledets \(2011\)](#).

Widespread 3D scene representations include: the Signed Distance Function (SDF), sometimes called Signed Distance Field; a truncated version of the SDF known as Truncated SDF or TSDF; occupancy grids; and the recently popularized neural implicit fields [Mildenhall et al. \(2020\)](#); [Sitzmann et al. \(2021\)](#). We experimentally analyze the effects of the low-rank constraint, using sequences of 3D scenes from the CAPE dataset [Ma et al. \(2020\)](#); [Pons-Moll et al. \(2017\)](#) and problem-specific quality metrics. Our method can compress time-varying SDFs to a usable size that would take hundreds of GBs in uncompressed form.

The paper is organized as follows. In Section 2, we start with an overview of applicable tensor methods and techniques for compression of 3D and 4D data. Section 4 gives a detailed outline of our approach. Section 5 demonstrates its the performance in an experimental study, followed by concluding remarks in Section 6. Our contributions are:

1. The first tensor decomposition framework for compressing temporal sequences in 3D voxel space;
2. A benchmark and analysis of different decomposition schemes for both the spatial and temporal dimensions;
3. An open source implementation of our method based on the `tf-torch` framework [Usvyatsov et al. \(2022\)](#), available at <https://github.com/Aelphy/T4DT>.

2 Background and Related Work

In our work, we apply tensor methods to temporal sequences of 3D scenes. We consider grid representations of the 3D data, i.e., a tensor $\mathbf{A} \in \mathbb{R}^{I_1 \times \dots \times I_D}$ constitutes a discrete sampling of a D -dimensional space on a grid $\mathbb{I} = I_1 \times \dots \times I_D$, with I_d samples along dimension d .

2.1 SDF and Truncated SDF

The SDF at position $p \in \mathbb{R}^3$ is defined as:

$$\text{SDF}(p) = \begin{cases} \text{dist}(p, \partial\Omega) & \text{if } p \in \Omega, \\ -\text{dist}(p, \partial\Omega) & \text{otherwise,} \end{cases} \quad (1)$$

while the truncated SDF clamps the SDF as follows:

$$\text{TSDF}(p) = \begin{cases} -\tau & \text{if } \text{SDF}(p) \leq -\tau, \\ \tau & \text{if } \text{SDF}(p) \geq \tau, \\ \text{SDF}(p) & \text{otherwise.} \end{cases} \quad (2)$$

Depending on the available input data and the desired application, different formats may be more or less suitable. For example, colored images can be converted to NERF or NELF [Mildenhall et al. \(2020\)](#); [Sitzmann et al. \(2021\)](#) which optimizes the reconstruction error via differentiable rendering. Depth images can be fused into TSDFs as described in [Curless and Levoy \(1996\)](#); meshes can be converted into TSDFs, too, and also the inverse transformation is possible, with variants of marching cubes [Lorensen and Cline \(1987\)](#); point clouds can be turned into meshes or directly into TSDFs; etc.

2.2 Compression

Storing signed distance fields constitutes a computational and memory bottleneck, especially for time-varying data. In [Tang et al. \(2018\)](#), a real-time compression method was proposed that uses a learnable, implicit temporal TSDF

representation, combining independent color and texture encodings with learnable geometry compression. Multi-dimensional visual data compression based on the Tucker decomposition, combined with bit-plane coding, was applied to volumetric data in [Ballester-Ripoll et al. \(2019\)](#). Later, [Boyko et al. \(2020\)](#) proposed compressing single-frame TSDFs with a block-based neural network. They showed that, although TSDF tensors usually cannot be written exactly as a low-rank expansion, in practice, one can store them in the low-rank TT representation with low compression error. Here, we build on adopting tensor decompositions for high-dimensional visual data and address the task of compressing dynamic 3d scenes, whose 4D grid representations quickly exceed practical memory limits when stored in uncompressed form.

Compressing TSDF with tensor methods not only allows to work with data that otherwise would not fit the memory, but it also can be useful for:

1. Fusion, as demonstrated in [Boyko et al. \(2020\)](#);
2. Constant-time TSDF random access¹ which might be useful to test if a given point is inside or outside of a mesh or potentially, ray-casting;
3. Constant-time computation of the TSDF gradient can be used to compute surface normals [Sommer et al. \(2022\)](#);
4. Unlike neural-based learning approaches, tensor methods rely on SVD-based compression, which provides theoretical guarantees [Oseledets \(2011\)](#) and makes the framework more interpretable.

Throughout this work, we use the term *compression factor* (or just *compression*, if clear from the context) to denote the ratio between the number of coefficients used for the compressed format and the number of coefficients required to store the uncompressed tensor.

3 Tensor Formats

3.1 Notation

Tensor diagram notation To visualize tensor networks, we adopt Penrose graphical notation [Penrose \(1971\)](#). See Fig. 2 for visualization of vector, matrix and matrix-vector product.

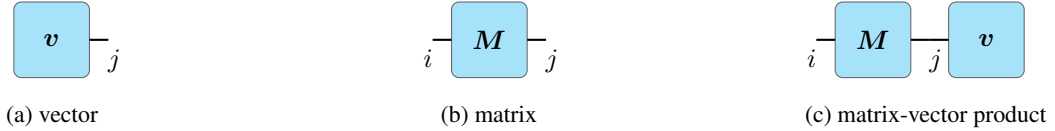


Figure 2: Each tensor is denoted as a node whose edges represent its dimensions. Whenever an edge is shared, tensor contraction is assumed: e.g., Fig. 2c depicts the matrix-vector product $\sum_j M_{ij} v_j$.

3.2 Tucker

The Tucker decomposition [Tucker \(1963\)](#) factors a tensor of dimension D into a D -dimensional *core* tensor $\mathbf{G} \in \mathbb{R}^{r_1 \times \dots \times r_D}$ and D matrices $\{\mathbf{A}_d\}_{d=1}^D$, $\mathbf{A}_d \in \mathbb{R}^{r_d \times I_d}$. The format is defined as

$$\mathbf{A}[i_1, \dots, i_D] = \mathbf{G}\mathbf{A}_1[:, i_1] \dots \mathbf{A}_D[:, i_D], \quad (3)$$

where r_d are the Tucker-ranks. The Tucker decomposition has $\mathcal{O}((\max_d[r_d])^D + \max_d[r_d] \cdot \max_d[I_d])$ storage cost. See Fig. 3a for visualization of Tucker format in graphical notation.

3.3 Tensor Train

The TT decomposition [Oseledets \(2011\)](#) factors a tensor of dimension D into a sequence of D 3-dimensional tensors. The format is defined as

$$\mathbf{A}[i_1, \dots, i_D] = \mathbf{Q}_1[0, i_1, :] \mathbf{Q}_2[:, i_2, :] \dots \mathbf{Q}_D[:, i_D, 0], \quad (4)$$

where the tensors $\{\mathbf{Q}_d\}_{d=1}^D$, $\mathbf{Q}_d \in \mathbb{R}^{r_{d-1} \times I_d \times r_d}$, are called *TT-cores*; and r_d are the *TT-ranks* ($r_0 = r_D = 1$). The TT decomposition has $\mathcal{O}(D \cdot (\max_d[r_d])^2 \cdot \max_d[I_d])$ storage cost, and leads to a linear tensor network; see Fig. 3b for a graphical example.

¹The complexity to access single element depends on the choice of tensor format and rank but is independent of the location of query point inside the tensor

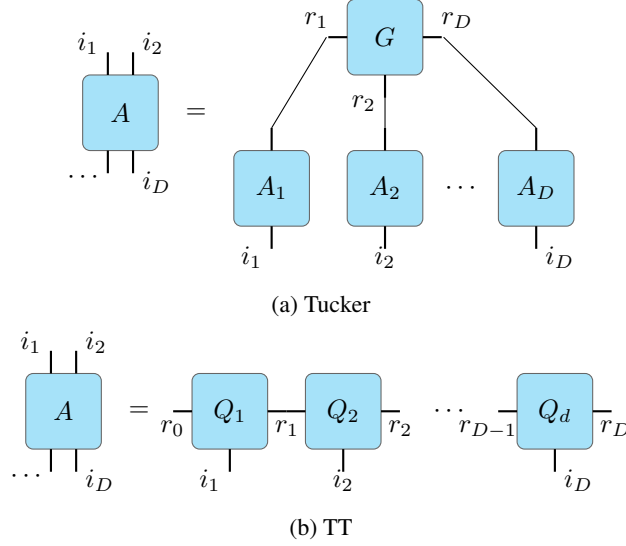


Figure 3: Graphical representation of the Tucker Fig. 3a and TT Fig. 3b decompositions.

3.4 QTT

Quantics TT (QTT) is an extension of TT that includes reshaping² the input into a tensor of shape $2 \times 2 \cdots \times 2 = 2^{\sum_{d=1}^D \log_2(I_d)}$, with I_j the size of the j -th mode [Kazeev et al. \(2017\)](#). The sub-dimensions x, y, z are then grouped side-by-side for each octet (imitating the traversal of a z -space filling curve), which makes QTT similar to a tensorized octree. Last, the resulting tensor is then subject to standard TT decomposition. This scheme is also connected to the wavelet transform [Kazeev and Oseledets \(2013\)](#).

Octet QTT We also propose a variant of QTT with a base dimension of size eight instead of two by merging the three sub-dimensions of each octet into one. This way, the resulting tensor has shape $8 \times 2 \times \cdots \times 8 \times 2$. We call this novel format OQTT for octet QTT, and we found it to reduce discontinuity artifacts (Section 5).

4 Proposed Method

We introduce T4DT, a method to compress high-resolution temporal TSDF fields with tensor decompositions discussed in Sections 3.3 - 3.4. We exploit that individual TSDF frames can be compressed into a low-rank TT decomposition with reasonable error [Boyko et al. \(2020\)](#). Under the low-rank constraint, a zero-level set can be reconstructed with sufficiently good quality. However, this becomes more challenging when considering time-evolving data since uncompressed 4D grid scenes at fine spatial resolutions can range in the hundreds of GBs. Section 4 gives a high-level overview of our pipeline.

4.1 Framework Decomposition

The first technical choice is the base decomposition selection at the frames' level. We considered three variants: Tucker, TT, and QTT. In 3D, Tucker is an attractive choice since it is equivalent to TT plus an additional compression step along the second dimension. For the 4D case, we propose to combine the best of both worlds by using the TT-Tucker blend, where spatial dimensions are stored in Tucker formatted cores, and the temporal dimension is represented with a TT core. Each core or factor is responsible for a single dimension in TT and Tucker formats. However, the Tucker model shares the core tensor between all dimensions.

Many real-world datasets are sparse in the sense that most of the occupied volume is empty. An octree is an excellent choice to pack such sparse data into a compact representation. We argue that QTT is a tensor analog of an octree data structure. Indeed, it is easy to see that padded and reshaped to $2^{\sum_{d=1}^D \log_2(I_d)}$ scene becomes a piece-wise separated reshaped set of octets after permutation of the corresponding sub-dimensions. Rank truncation is a way to reduce the number of coefficients to encode each octet sub-dimension.

²With zero-padding where needed

Algorithm 1 T4DT pipeline. Since the whole 4D scene is not expected to fit into memory, we first compress each frame individually. Next, the collection of individually compressed frames is assembled into a single compressed scene using a tree-like merge procedure $\mathbf{X}'_0 \cup \mathbf{Y}$. See Section 4.2 for the merge procedure details.

$\mathbb{I} = I_1 \times I_2 \times I_3 \times I_4$ – temporal 3D grid
 $\mathbf{X} \in \mathbb{R}^{\mathbb{I}}$ – input temporal TSDF
 R_s – maximal rank along spatial dimensions
 R_t – maximal rank along time dimension
 \mathbf{X}' – Collection of compressed TSDF frames
 \mathbf{Y} – Compressed temporal TSDF

Require:

for $i = 1, 2, \dots, I_4 - 1, I_4$ **do**
 $\mathbf{X}'_i \leftarrow \text{truncate}(\mathbf{X}[\dots, i], R_s)$
end for
while $\mathbf{X}' \neq \emptyset$ **and** $|\mathbf{Y}| \neq 1$ **do**
 $\mathbf{Y} \leftarrow \text{truncate}(\mathbf{X}'_0 \cup \mathbf{Y}, R_t)$
 $\text{remove}(\mathbf{X}'_0)$
end while

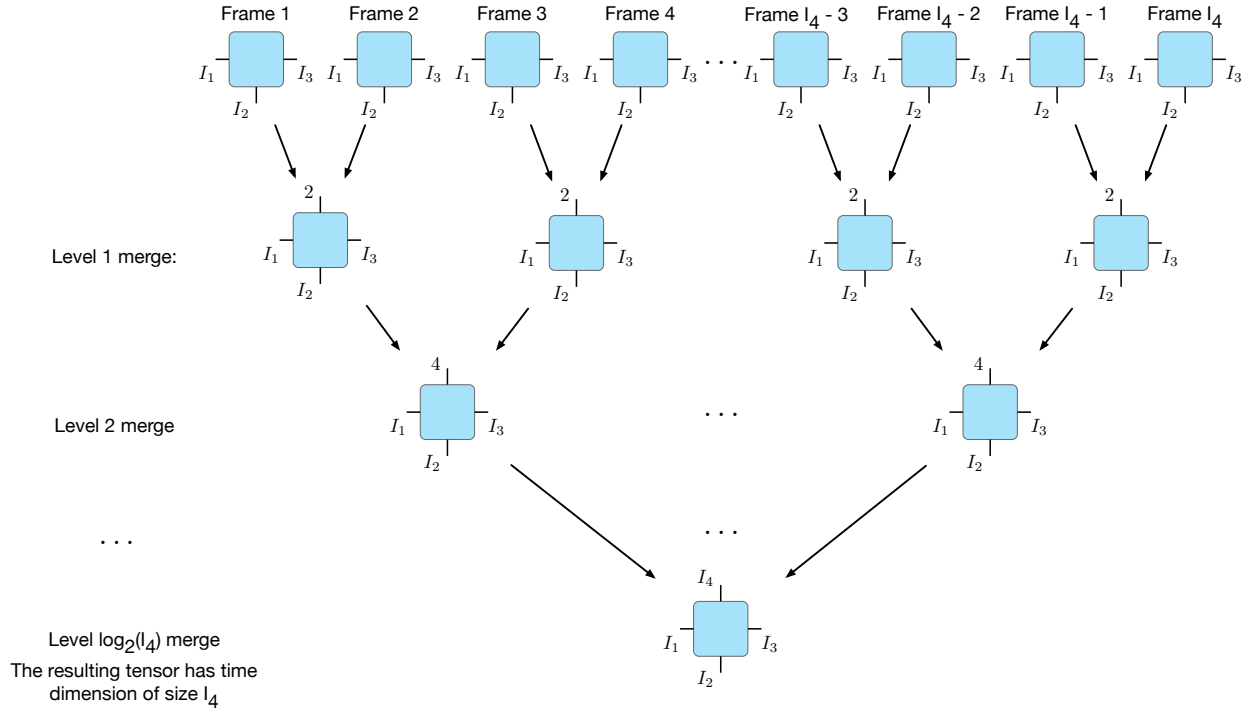


Figure 4: We compress and merge individual frames progressively into a compressed scene (for simplicity, all tensors are shown in uncompressed form). Two of the previous level tensors are merged at each tree level using the concatenation procedure (which increases tensor rank), followed by rank truncation. On the first level, an additional core for the temporal dimension is inserted into each frame.

4.2 Frame Concatenation

Since the original scene can exceed by far the available memory, we developed a progressive procedure to merge compressed frames into a single compressed scene. The algorithm is akin to the *pairwise summation* method to reduce round-off errors in numerical summation [Higham \(2002\)](#): we stack the frames by pairs along a new temporal dimension, recompress each pair via tensor rank truncation [Oseledets \(2011\)](#), and repeat the procedure recursively in a binary-tree fashion to ultimately yield a single 4D compressed tensor. See Fig. 4 for an illustration.

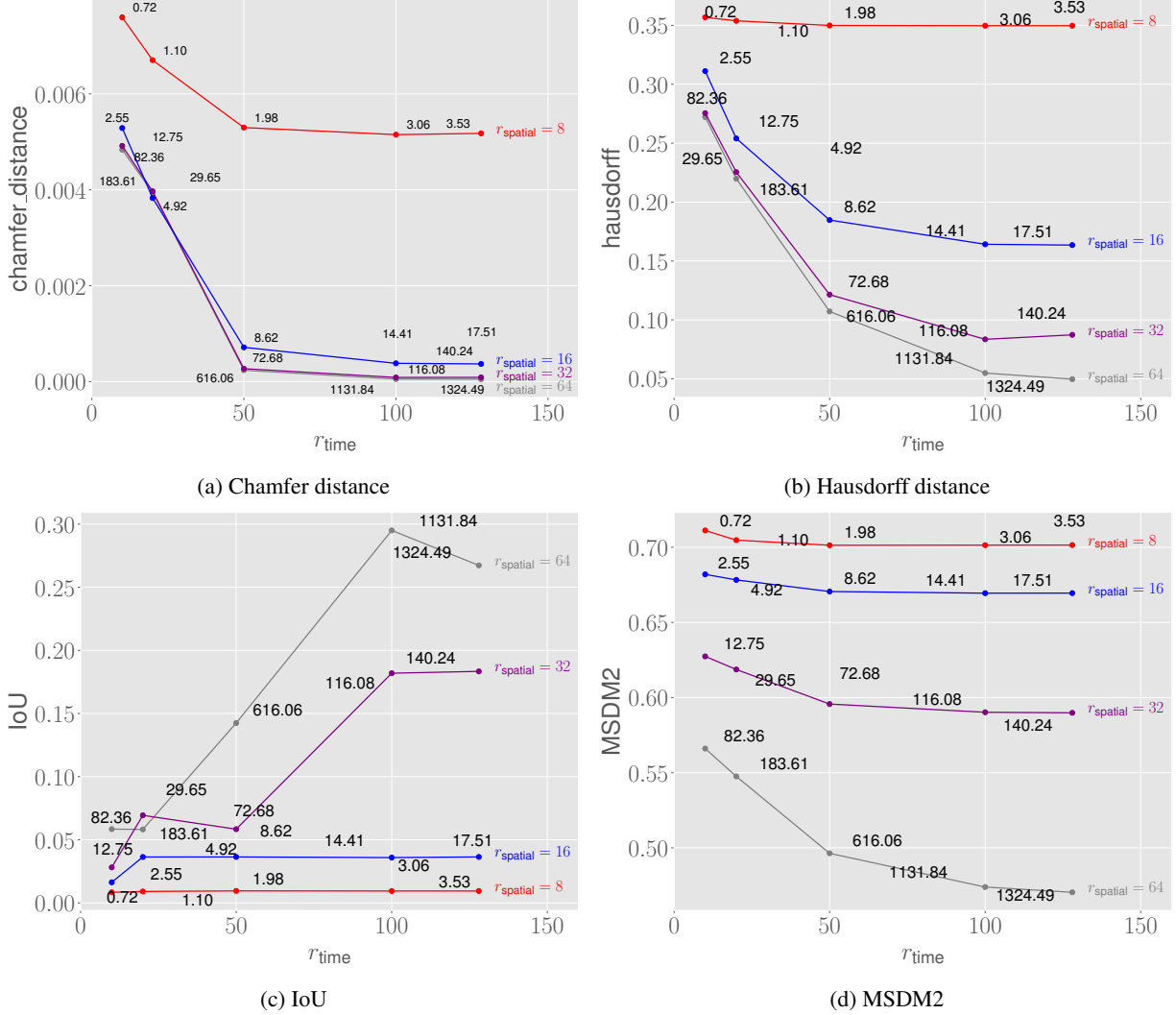


Figure 5: Ranks vs. performance for TT-Tucker compressed *longshort-flying-eagle* scene of resolution 512^3 with 284 time frames. Metrics are averaged between frames 1, 142, and 284. Each data point is annotated with the corresponding compression factor scaled with 10^6 .

Note that this procedure is compatible with any format that supports rank truncation, which includes the TT and Tucker formats and mixtures thereof that allow for concatenation in compressed format. They are implemented using the concatenation of the cores, increasing the rank of the resultant cores to a sum of the ranks of the operands. Afterward, rank truncation allows reducing rank back to the desired maximal value using an SVD-like algorithm [Oseledets \(2011\)](#).

QTT scene. The scene stored in QTT format has the shape $x_1 \times y_1 \times z_1 \times t_1 \times x_2 \times y_2 \times z_2 \times t_2 \times \dots \times x_k \times y_k \times z_k \times t_k$, where $x_i = y_i = z_i = t_i = 2$ and $\prod_{i=1}^{\lceil \log_2 I_1 \rceil} x_i = W$, $\prod_{i=1}^{\lceil \log_2 I_2 \rceil} y_i = D$, $\prod_{i=1}^{\lceil \log_2 I_3 \rceil} z_i = H$, $\prod_{i=1}^{\lceil \log_2 I_4 \rceil} t_i = T$, with uncompressed scene shape $W \times D \times H \times T$. In order to reconstruct the i -th frame from a scene compressed this way, one must first compute the binary representation of i and then decompress the sub-tensor at $[:, :, :, i_{\text{bit}_k}, :, :, :, i_{\text{bit}_{k-1}}, \dots, :, :, i_{\text{bit}_0}]$.

OQTT scene. The scene stored in OQTT format has the shape $x_1 y_1 z_1 \times t_1 \times x_2 y_2 z_2 \times t_2 \times \dots \times x_k y_k z_k \times t_k$, where $x_i = y_i = z_i = t_i = 2$ and $\prod_{i=1}^{\lceil \log_2 I_1 \rceil} x_i = W$, $\prod_{i=1}^{\lceil \log_2 I_2 \rceil} y_i = D$, $\prod_{i=1}^{\lceil \log_2 I_3 \rceil} z_i = H$, $\prod_{i=1}^{\lceil \log_2 I_4 \rceil} t_i = T$, with uncompressed scene shape $W \times D \times H \times T$. In order to reconstruct the i -th frame from a scene compressed this way, one must first compute the binary representation of i and then decompress the sub-tensor at $[:, i_{\text{bit}_k}, :, i_{\text{bit}_{k-1}}, \dots, :, i_{\text{bit}_0}]$.

Algorithm 2 Conversion of a single 3D volumetric frame into OQTT format.

```

 $\mathbb{I} = I_1 \times I_2 \times I_3$  – 3D grid
 $\mathbf{X} \in \mathbb{R}^{\mathbb{I}}$  – input 3D volumetric frame
 $R$  – maximal rank
 $\mathbf{Y}$  – Compressed 3D volumetric frame
Require:
  for  $d = 1, 2, 3$  do
     $\mathbf{X} \leftarrow \text{pad}(\mathbf{X}, 2^{\lceil \log_2 I_d \rceil})$ 
  end for
 $\mathbf{X} \leftarrow \text{reshape}(\mathbf{X}, 2^{\sum_{d=1}^3 \lceil \log_2(I_d) \rceil})$ 
 $\mathbf{X} \leftarrow \text{permute}(\mathbf{X}, (x_1, y_1, z_1, \dots, x_{\lceil \log_2 I_1 \rceil}, y_{\lceil \log_2 I_2 \rceil}, z_{\lceil \log_2 I_3 \rceil}))$ ,
 $I_1 = \prod_{i=1}^{\lceil \log_2 I_1 \rceil} x_i, I_2 = \prod_{i=1}^{\lceil \log_2 I_2 \rceil} y_i, I_3 = \prod_{i=1}^{\lceil \log_2 I_3 \rceil} z_i$ 
 $\mathbf{Y} \leftarrow \text{TT}(\mathbf{X}, R)$ 

```

5 Results

5.1 Data

We use the selected scenes from the CAPE dataset introduced in [Ma et al. \(2020\)](#); [Pons-Moll et al. \(2017\)](#). The dataset consists of 284 3D mesh registrations of 15 (10 male, 5 female) clothed people in motion.

For each scene frame, we compute its TSDF and discretize it at resolution 512^3 using the PySDF library. We use $\tau = 0.05$ in Eq. (2) to allow ~ 10 voxels with distinct levels near the TSDF zero level set. Since all scene frames must share a common frame of coordinates, we first compute the scene’s bounding box and compute the SDF of each frame within that box.

We also demonstrate the performance of T4DT on selected scenes from Articulated Mesh Animation dataset [Vlasic et al. \(2008\)](#). That real-world dataset contains 10 mesh sequences depicting 3 different humans performing various actions.

5.2 Low-rank influence

We provide error metrics computed for the TT-Tucker format and averaged between the first, middle, and last frames of the *longshort-flying-eagle* scene for different spatial and temporal ranks in Fig. 5. See Appendix A.1 for the definition of each metric. Note that the time dimension is far from full-rank in the TT-Tucker format as the error metrics saturate far below 284, which is the size of the temporal dimension.

See Fig. 7b for error metrics for the TT format. The best compression/performance was obtained for the OQTT base format. See qualitative results in Fig. 1 for OQTT, Fig. 8 for QTT, Fig. 6 for TT-Tucker, and Fig. 7 for the TT format.

In the cases of TT and TT-Tucker, severe rank reduction smoothens the reconstructed surface and erodes smaller features like fingers or facial details. In contrast, QTT preserves more details at the cost of discontinuities due to octet sub-dimensions separation and their independent compression. OQTT reduces the discontinuity artifacts by encoding the octets as a single dimension without compressing sub-dimensions inter ranks. See Fig. 1 for qualitative results for OQTT. We also present quantitative OQTT compression results optimized for performance in Table 1.

6 Conclusions

We have presented T4DT, a novel scalable and interpretable compression pipeline for temporal 3D data based on tensor decomposition. We improved over the related method TT-TSDF in two ways:

1. We are able to process temporally varying data. We can do so even though the TSDF field takes hundreds of GBs, and our method works fully in-memory;
2. We tested alternative tensorization schemes such as a TT-Tucker hybrid, the QTT format, and the new OQTT variant and found OQTT to outperform previous decompositions.

We argue for the effectiveness of the special reshaping of a scene into OQTT formatted tensor and provide quantitative and qualitative justifications.





	Crane	Swing	Handstand	Samba
				
	$512^3 \times 174$	$512^3 \times 174$	$512^3 \times 174$	$512^3 \times 149$
Metric				
L2 ↓	2.67	2.07	2.45	1.71
Chamfer distance ↓	$5e^{-5}$	$4e^{-5}$	$5e^{-5}$	$4e^{-5}$
Hausdorff distance ↓	0.19	0.015	0.012	0.013
MSDM2 ↓	0.36	0.38	0.35	0.36
IoU ↑	0.41	0.4	0.64	0.54
Compression	1:954	1:1356	1:938	1:935
Size	549 MB	386 MB	558 MB	560 MB

Table 1: Metrics of OQTT optimized for performance ($r_{\max} = 4000$) for selected scenes from Articulated Mesh Animation dataset [Vlasic et al. \(2008\)](#). Results are averaged between the first, the middle, and the last frames.

References

- Ballester-Ripoll, R., Lindstrom, P., and Pajarola, R. (2019). TTHRESH: Tensor compression for multidimensional visual data. *IEEE Transactions on Visualization and Computer Graphics*, 26(9):2891–2903. 3
- Boyko, A. I., Matrosov, M. P., Oseledets, I. V., Tsetserukou, D., and Ferrer, G. (2020). TT-TSDF: Memory-efficient TSDF with low-rank tensor train decomposition. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10116–10121. IEEE. 3, 4, 11
- Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. 2
- Fuji Tsang, C., Shugrina, M., Lafleche, J. F., Takikawa, T., Wang, J., Loop, C., Chen, W., Jatavallabhula, K. M., Smith, E., Rozantsev, A., Perel, O., Shen, T., Gao, J., Fidler, S., State, G., Gorski, J., Xiang, T., Li, J., Li, M., and Lebedev, R. (2022). Kaolin: A pytorch library for accelerating 3d deep learning research. <https://github.com/NVIDIAGameWorks/kaolin>. 11
- Higham, N. J. (2002). *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, USA, 2nd edition. 5
- Jacobson, A., Panozzo, D., et al. (2018). libigl: A simple C++ geometry processing library. <https://libigl.github.io/>. 11
- Jakob, W., Rhineland, J., and Moldovan, D. (2017). pybind11 – seamless operability between c++11 and python. <https://github.com/pybind/pybind11>. 11
- Kazeev, V. and Oseledets, I. (2013). The tensor structure of a class of adaptive algebraic wavelet transforms. *Preprint*, 28. 4
- Kazeev, V. A., Oseledets, I., Rakhuba, M., and Schwab, C. (2017). Qtt-finite-element approximation for multiscale problems i: model problems in one dimension. *Advances in Computational Mathematics*, 43:411–442. 4
- Lavoué, G. (2011). A multiscale metric for 3d mesh visual quality assessment. *Computer Graphics Forum*, 30. 11
- Lavoué, G., Gelasca, E. D., Dupont, F., Baskurt, A., and Ebrahimi, T. (2006). Perceptually driven 3d distance metrics with application to watermarking. In *Applications of Digital Image Processing XXIX*, volume 6312, pages 150–161. Spie. 11
- Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169. 2
- Ma, Q., Yang, J., Ranjan, A., Pujades, S., Pons-Moll, G., Tang, S., and Black, M. J. (2020). Learning to Dress 3D People in Generative Clothing. In *Computer Vision and Pattern Recognition (CVPR)*. 2, 7
- Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer. 2
- Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317. 2, 3, 5, 6
- Penrose, R. (1971). Applications of negative dimensional tensors. *Combinatorial mathematics and its applications*, 1:221–244. 3
- Pons-Moll, G., Pujades, S., Hu, S., and Black, M. J. (2017). Clothcap: Seamless 4D clothing capture and retargeting. *ACM Transactions on Graphics (ToG)*, 36(4):1–15. 2, 7
- Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666. 11
- Sitzmann, V., Rezkikov, S., Freeman, B., Tenenbaum, J., and Durand, F. (2021). Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems*, 34:19313–19325. 2
- Sommer, C., Sang, L., Schubert, D., and Cremers, D. (2022). Gradient-sdf: A semi-implicit surface representation for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6280–6289. 3
- Tang, D., Dou, M., Lincoln, P., Davidson, P., Guo, K., Taylor, J., Fanello, S., Keskin, C., Kowdle, A., Bouaziz, S., et al. (2018). Real-time compression and streaming of 4d performances. *ACM Transactions on Graphics (TOG)*, 37(6):1–11. 2

- Tucker, L. R. (1963). Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15:122–137. 2, 3
- Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311. 2
- Usvyatsov, M., Ballester-Ripoll, R., and Schindler, K. (2022). tntorch: Tensor network learning with PyTorch. *Journal of Machine Learning Research*, 23. 2
- Vlasic, D., Baran, I., Matusik, W., and Popović, J. (2008). Articulated mesh animation from multi-view silhouettes. *ACM SIGGRAPH 2008 papers*. 7, 8

A Appendix

A.1 Metrics

Throughout this work, we convert temporal 3D data from mesh format to tensor format for compression using sampled truncated SDF and from tensor format back to temporal 3D data in the form of meshes for quantitative tests using the marching cubes algorithm.

We selected a tensor-based l2 reconstruction metric for the comparison in tensorial form. We use the metrics below to evaluate the quality of the obtained mesh.

A.1.1 Intersection over Union (IoU)

IoU is a commonly used metric for comparing 3D shapes [Rezatofighi et al. \(2019\)](#). For two given 3D volumes \mathbf{A} and \mathbf{B} :

$$\text{IoU}(\mathbf{A}, \mathbf{B}) = \frac{\|\mathbf{A} \cap \mathbf{B}\|}{\|\mathbf{A} \cup \mathbf{B}\|}. \quad (5)$$

3D mesh if converted to occupancy grid, and the operation is easily performed with implementation from [Fuji Tsang et al. \(2022\)](#).

A.1.2 Hausdorff distance

One-sided Hausdorff distance is computed as:

$$d_h(\mathbf{A}, \mathbf{B}) = \max\{\min\{\|a - b\| : a \in \mathbf{A}\} : b \in \mathbf{B}\}. \quad (6)$$

Note, that d_h is not symmetric. The symmetric version is defined as :

$$d_H(\mathbf{A}, \mathbf{B}) = \max\{d_h(\mathbf{A}, \mathbf{B}), d_h(\mathbf{B}, \mathbf{A})\}. \quad (7)$$

We use the implementation from [Jacobson et al. \(2018\)](#).

A.1.3 Chamfer distance

Symmetric Chamfer distance is defined as:

$$d_{CD}(\mathbf{A}, \mathbf{B}) = \sum_{a \in \mathbf{A}} \min_{b \in \mathbf{B}} \|x - y\|^2 + \sum_{b \in \mathbf{B}} \min_{a \in \mathbf{A}} \|x - y\|^2. \quad (8)$$

Following [Boyko et al. \(2020\)](#) we sample 30000 points from each mesh and compute sampled symmetric Chamfer distance. Finally, we use the implementation from [Fuji Tsang et al. \(2022\)](#).

A.1.4 Mesh Structural Distortion Measure (MSDM2)

Introduced in [Lavoué \(2011\)](#) MSDM2 metric is used to estimate the correlation with human visual perception. The metric assumes one mesh to be original and the second distorted. See Algorithm 3 for an algorithmic view of MSDM2 computation. Refer to [Lavoué et al. \(2006\)](#); [Lavoué \(2011\)](#) for the more details on metric formulation. We use original implementation from [Lavoué \(2011\)](#) and self written pybind11 [Jakob et al. \(2017\)](#) interface to Python.

Algorithm 3 MSDM2 high-level computation scheme

M_o – original mesh, M_d – distorted mesh, $\{v_i \in M\}$ – set of mesh vertices

Require:

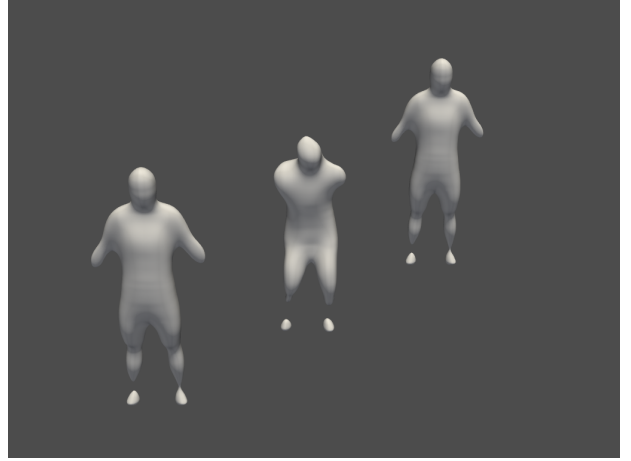
```

for  $v_i \in M_o$  do
    Find  $v_j = \min_{v_k \in M_d} \text{dist}(v_i, v_k)$ 
end for
for  $v_i \in M_o$  do
    calculate curvature of  $v_i$ 
end for
for  $v_i \in M_d$  do
    calculate curvature of  $v_i$ 
end for
interpolate curvature per face for  $M_o$  and  $M_d$ 
for  $v_i \in M_d$  do
    calculate MSDM2 for  $v_i$  based on curvature features and nearest neighbours  $v_j$  from  $M_o$ 
end for
Integrate global asymmetric MSDM2

```



(a) Original, 284 GB



(b) Compression 1:769230, 0.2 MB

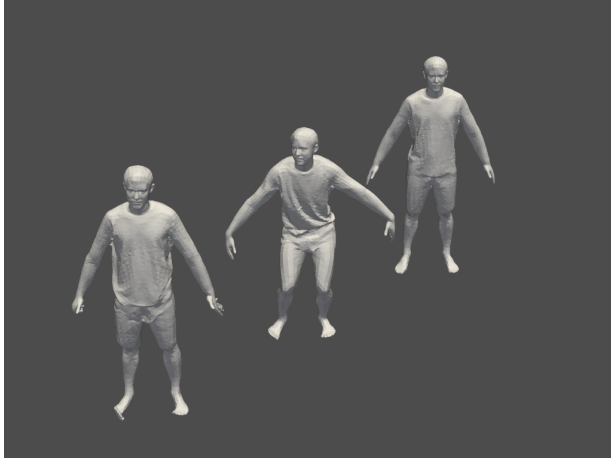


(c) Compression 1:1333, 33 MB



(d) Compression 1:116, 385 MB

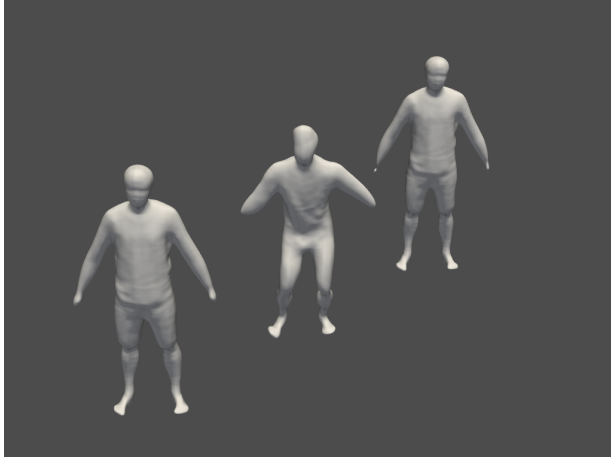
Figure 6: T4DT with different compression levels in TT-Tucker format for a *longshort-flying-eagle* scene of resolution 512^3 with 284 time frames. Only frames 1, 142, and 284 are depicted.



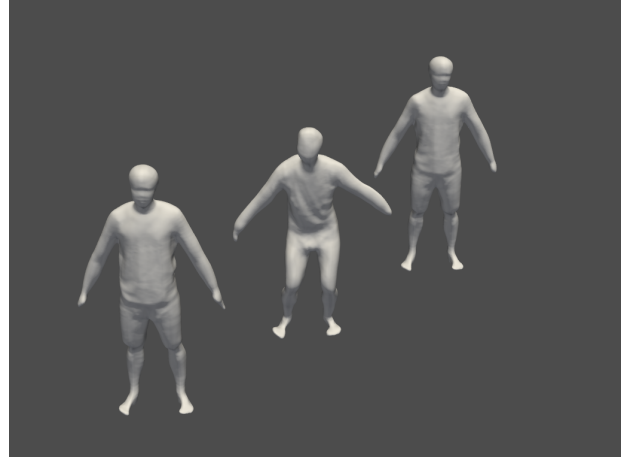
(a) Original, 284 GB



(b) compression 1:476190, 0.61 MB



(c) compression 1:6250, 16 MB



(d) compression 1:2000, 48.5 MB

Figure 7: T4DT with different compression levels in TT format for a *longshort-flying-eagle* scene of resolution 512^3 with 284 time frames. Only frames 1, 142, and 284 are depicted. The compression ratio is different from the actual memory consumption due to the padding of the time dimension to 512.

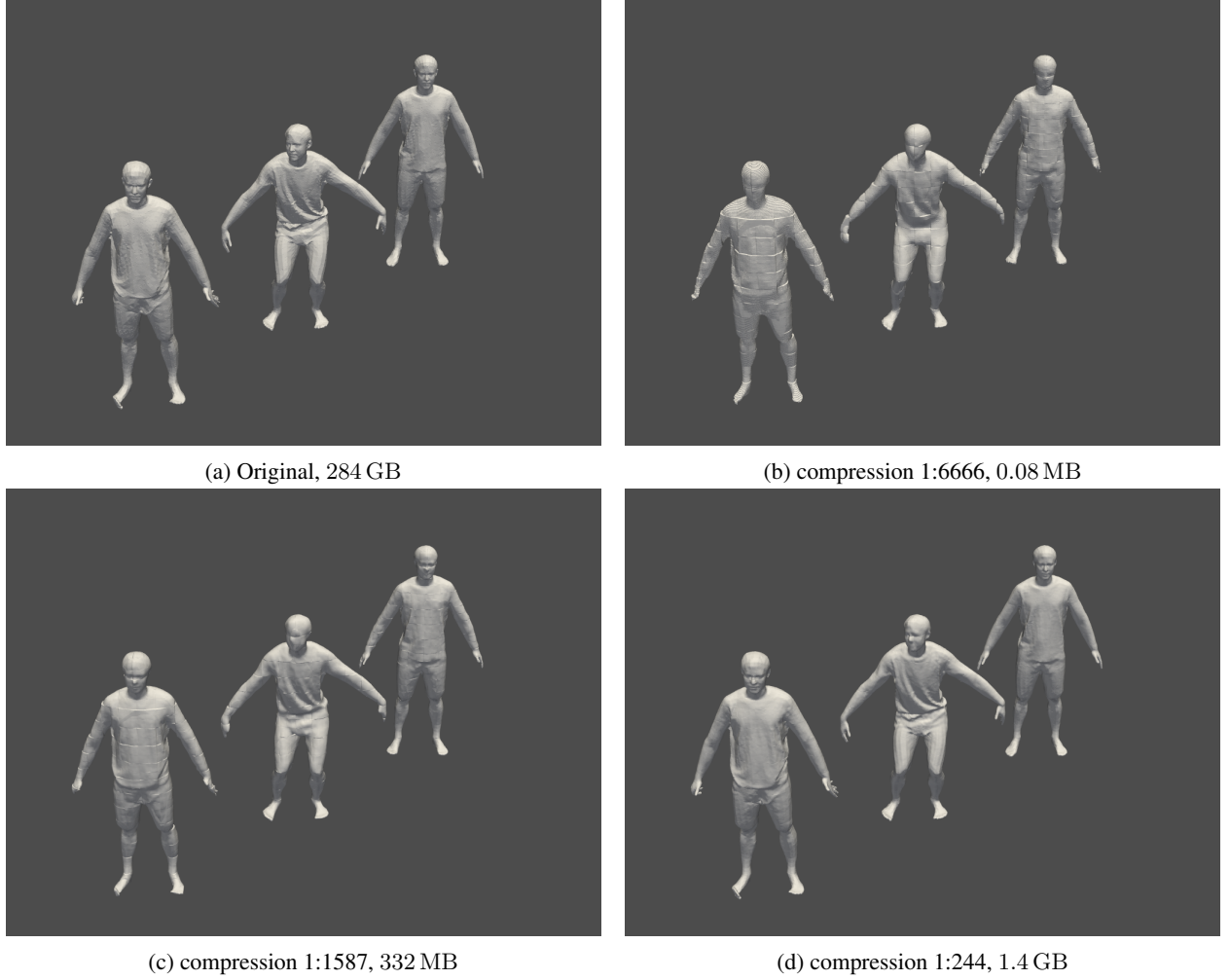


Figure 8: Different levels of compression in QTT format for a *longshort-flying-eagle* scene of resolution 512^3 with 284 time frames. Only frames 1, 142, and 284 are depicted. The compression ratio is different from the actual memory consumption due to the padding of the time dimension to 512.

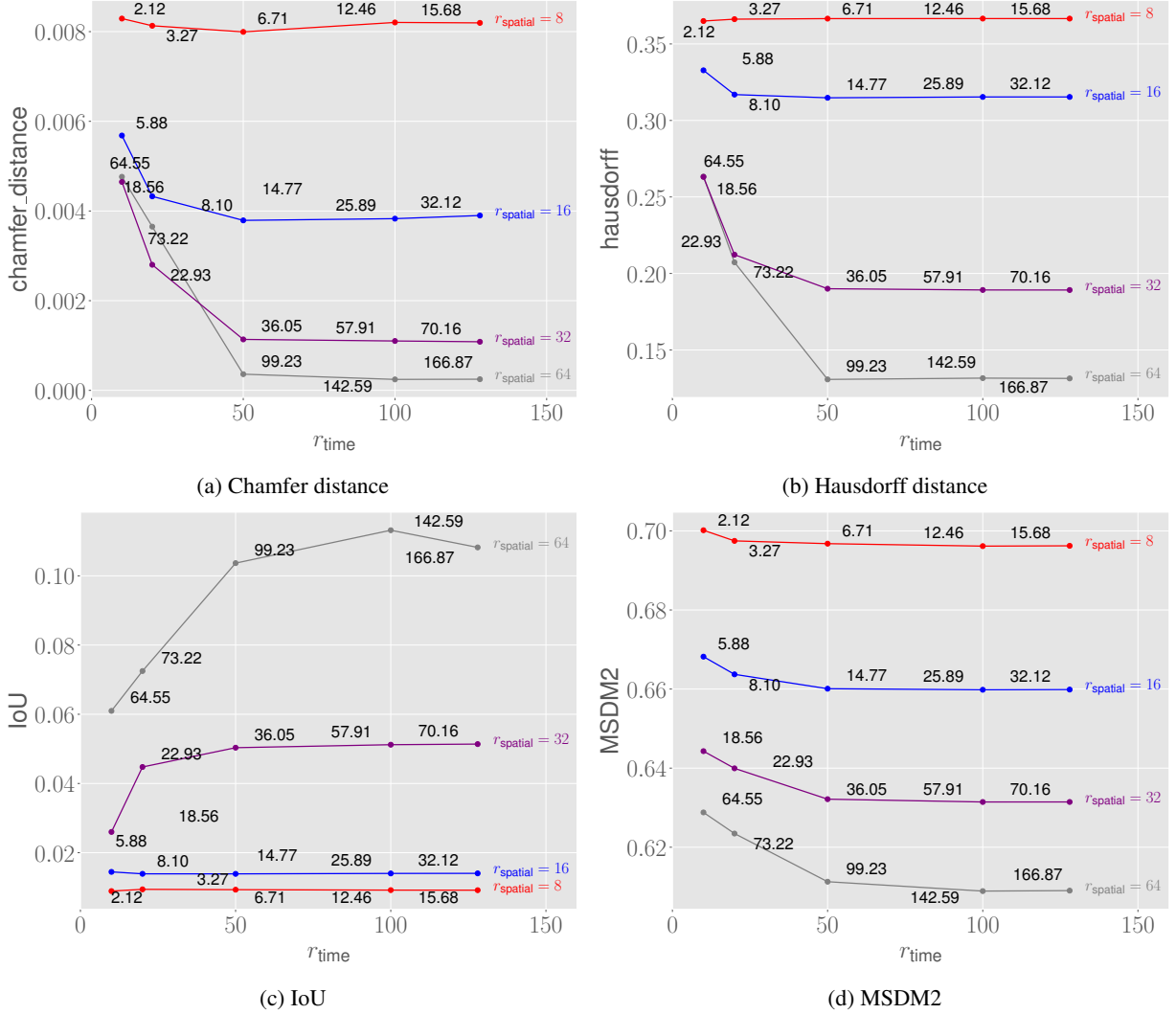
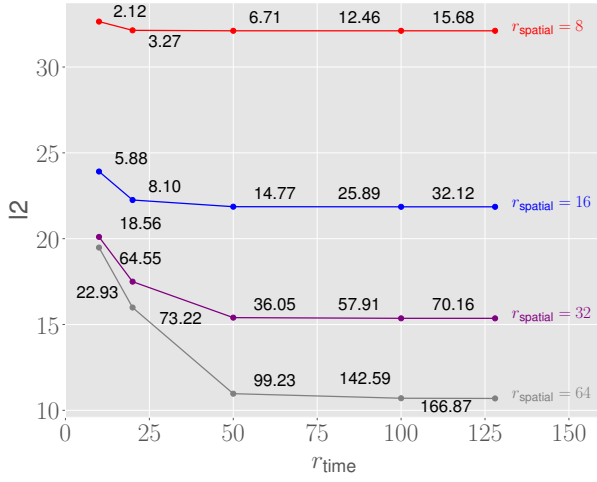
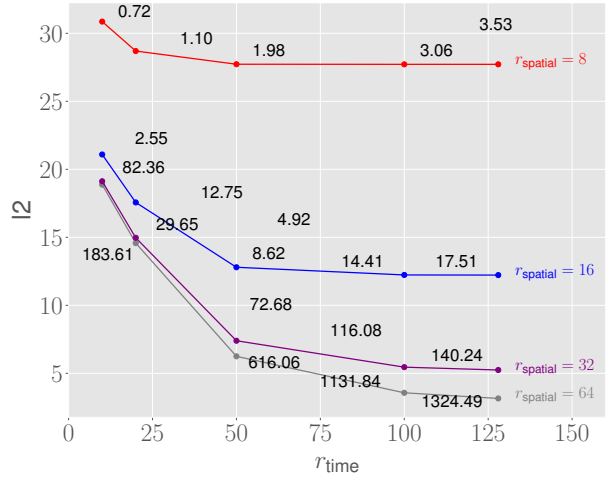


Figure 9: Ranks vs. performance for TT compressed *longshort-flying-eagle* scene of resolution 512^3 with 284 time frames. Metrics are averaged between frames 1, 142, and 284. Each data point is annotated with the corresponding compression factor scaled with 10^6 .



(a) TT



(b) TT Tucker

Figure 10: Ranks vs. L_2 for *longshort-flying-eagle* scene of resolution 512^3 with 284 time frames. L_2 is averaged between frames 1, 142, and 284. Each data point is annotated with the corresponding compression factor scaled with $1e6$.