WAVELET BASED APPROACH TO 3D GEOMETRY POINT CLOUD COMPRESSION

AT TEXAS A&M UNIVERSITY

A Thesis

by

MAX MICHAEL JAKOB LESSER

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Zixiang Xiong
Committee Members	Serap Savari
	Sunil Khatri
	Raktim Bhattacharya
Head of Department,	Miroslav Begovic

May 2023

Major Subject: Electrical Engineering

ABSTRACT

Proposed is a method of 3D point cloud geometry compression. Point clouds find applications in autonomous driving, education, and other fields. The approach proposed here hopes to improve on existing quantization techniques by segmenting each coordinate axis into continuous regions of arbitrary size and preforming a wavelet transformation on these regions. This Thesis describes a means of finding these regions, taking the transform, quantizing, concatenating acquired data in a bitstream, recovering data from the bitstream, and reconstructing the point cloud. Results presented on solid, dense, and sparse point clouds show improvements relative to Quantization-Inverse Quantization at low bitrates but exhibit poor performance at higher bitrates and lower densities. A better sorting approach is expected to increase performance, but the need for continuous regions, and possible latency incurred when sorting for them imposes higher quality requirements on this approach.

DEDICATION

Fur Hans, du frisst rostige Nägel und scheist fertige Lokomotiven aus, ich fress ganze objekte und scheiss einsen und nullen, mog good mien Fründ.

Special thanks go to Yalong Pi and Max Witek and Brent Matthews for their technical support, for their input, for their compassion and for sharing in my frustration and success during this work. To oscillating frequencies and bottoms of bottles, and may your cones always have splendid tippities. I want to thank my Mom and my Dad for everything they've done to put me into the position to get myself to where I am today. I want to thank the Askey clan, for being an amazing, loving family, and the Kimbell-Morans, for accepting me into their midst and of course Emma, for her love, support and compassion during this time. I want to thank John Travis, for being an all-around good friend and good person. Felix, Rustam ey bauma. Atze mach ma weiter so.

Many other people I want to thank, Andy, Collin, Q, Jack, Cezar, Edik, Roman, Dennis, Rachel, Ruby, Ryan there are far too many to list here, but all were part in getting me to where I am, and for that I am thankful.

To Emma: I love you more, it's in an official document now, I win.

"We're going all the way 'til the wheels fall off and burn"

ACKNOWLEDGEMENTS

I want to thank Dr. ZX for his continued support not just during this academic work, but also over the course of my entire graduate career and for motivating me to attend Grad School in the first place. Thank you.

I want to thank Xu Jizheng, for this support and guidance in this work, and for the wealth of knowledge he brought to this cooperation.

I'd like to thank the committee members for affording me this opportunity, for many valuable lessons learned in the classroom and outside, and for guiding me along this academic journey in the pursuit of knowledge.

I want to thank the ECEN Capstone Team, especially Dr. Kalafatis, for my teaching assistantship, for the flexibility to allow me an optimal usage of time, and their general support.

I'd finally like to thank the many wonderful faculty members in the ECEN department and College of Engineering whom I've had the pleasure to learn from during my time at Texas A&M. Every class and every interaction has pushed me closer to being able to complete this work, and for that I am thankful.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of my Advisor, Professor Zixiang Xiong and Professors Serap Savari, and Sunil Khatri of the Department of Electrical and Computer Engineering and Professor Raktim Bhattacharya of the Department of Aerospace Engineering.

The Point cloud data used to generate graphs and visualizations throughout this work was provided by Xu Jizheng and generated for MPEG by authors as cited in the Reference section. The GRASP framework used as an anchor was authored by J. Pang, M. Lodhi and D. Tian, as noted in the Reference section. All images, figures, graphs and tables used in this Thesis were generated by the author using datasets and publications cited in the reference section.

Funding Sources

Graduate study was supported by a Graduate Assistant position from Texas A&M University. The contents of this work are solely the responsibility of the authors and do not necessarily represent the official views of Texas A&M University.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND MOTIVATION

Figure 1: Sample Point Clouds. From left to right: Redandblack-vox10, soldier-vox10, boxer-vox12 and Staue-Klimt-vox12, where density decreases as we move right. These are the original, colored version of some of the point clouds we will use to test out compression scheme on.

One morning in the 17th century Rene Descartes laid in bed and observed a fly on the ceiling. In a letter he described the fly's location by taking steps along the orthogonal walls of the room, and thus invented cartesian coordinates. Or so goes the legend told to many young mathematicians, engineers, and scientists. Had Descartes described the position of two flies, he would have described the object of study in this Thesis, point clouds. Some sample point clouds are shown in figure 1. In the most basic view these are nothing but collections of points in 3D space. In a narrower view, the use of 3D-pointcluds as a computing tool have risen in parallel

with a means to capture them. Point clouds can be generated in several ways, one popular option being LIDAR sensors, as first invented by Hughes Aircraft Company [1]. In their 1972 paper on lunar laser altimetry Kaula et. al described the use of LIDAR Sensors aboard Apollo 15 to capture the spacecrafts altitude above the moons surface. Using earth-based tracking the spacecrafts location above the lunar surface could be determined [2]. Combining LIDAR altimeter data with spacecraft positional data yields a point cloud, although the term was not in use at the time. Smith et. al were able to create a topological map of the moon using this data [3]. Point clouds have been used in other areas since, for example by Slob et. al, who used 3D point clouds in their 1980 work to measure rock faces [4], by Sharr et. al. to store and process plants 3D structure in 1987 [5], and in 2001 Hoppe et. al. proposed an approach to use 3D point clouds to assist in surgery [6]. Given the ubiquity of 3 dimensional structures it is hard to point to a single origin of point clouds, but the above shows a computational use for over 50 years and one may surely find applications for 3D positional data reaching back much further. These applications listed above however are narrow use cases and scientific in nature, and hence can afford large data volumes. For point clouds to become broadly applicable efficient storage and representation is needed.

To provide standards for point cloud compression, we look to the ISO/IEC, and particularly MPEG. MPEG first started investigations into point clouds in 2014 when the 3D Graphics Coding Group began research into immersive technologies. This prompted a Call for Proposals in 2017 for 3D coding tools suitable to a wide range of applications culminating in the first 3D coding standards, V-PCC and G-PCC, published in 2020 [7]. However, as with any compression standard, work to find more efficient tools is ongoing, with PCGC published in 2021 [8] and grasp net published in 2022 [9]. Just as in Video Coding, where MPEG 1 was released in the

early 1990s [10], and nearly 30 years later new advances are still made, such as MPEGs H.266-VVC published in 2019 [11], Point Cloud compression can look forward to many years of development yet to come. We hope that this work can provide a small steppingstone on this path.

1.1 Introduction to Point Cloud Data

Point clouds are a primary means of representing 3D structures digitally. Point clouds consist of coordinate information and may include attribute information, for example color, opacity, or reflectivity. Point clouds may be captured in the real word using LIDAR or other distance sensors, or synthetically generated. Regardless of origins a point cloud consist of a collection of points ($\mathbf{x}_0$, $\mathbf{x}_1$,$\mathbf{x}_2$, . . ., $\mathbf{x}_N$) where each $\mathbf{x}_i$ consists of at least (x,y,z) coordinates, and may include any number of attributes $a_{i,0} – a_{i,K}$. As this work concerns itself with geometry compression, we will ignore attribute values going forward.

Given a set of captured coordinates in the world frame, we first transform them into a reference frame or internal frame system. To achieve this, we must fix a Geometry bit depth n, the number of bits we want to allocate to each axis element of the geometry. We next determine the maximum distance in the world frame from the origin and divide this length into 2^n segments. Thus, the resolution is set. We finally take the point volume in the world frame and rotate it such that the maximum distance is achieved along at least one coordinate axis. The resultant volume is then divided into cubes of size ($2^{-n}x2^{-n}x2^{-n}$), where all points falling within one such cube are associated with its center, and duplicates removed. We now can describe points via integer coordinates values indicating the cube in which they fall. We have thus voxelized the point

cloud. This transformation can be undone by storing the rotation information, geometry bit depth, and maximum distance in the world coordinate system.

Depending on the value of n, the total number of points and their distribution, point clouds are classified as either Solid, Dense, Sparse, or Scant. Typically, both geometry bit depth and number of points increase as point clouds get sparser, however the ultimate decision rests on density, number of points per smallest volume containing all points. Density is generally around 10^{-3} for dense point clouds and decreases by an order of magnitude or more for each sparser category [12]. This reference also lists MPEG approved datasets, which will be drawn from for our own testing.

Dense and solid point clouds typically represent a single object of focus, while sparser point clouds are more commonly used to represent scenes, or LIDAR scans for autonomous navigation. This distinction is not firm and can vary by use case. Next, we will give some information on possible applications of point clouds.

1.2 Use Cases and Motivation

1.2.1 Artifact Preservation

In cultural preservation, 3D imagining may be used to create realistic renderings of artifacts or scenes [13]. These can be more immersive and offer greater detail than simple images or video, and thus allow historians to study objects and scenes in detail without being present, allow us to preserve cultural heritage for future generation, and enable sharing of cultural goods, such as museum exhibits, over great distance and in greater detail.

1.2.2 Industry

The construction industry also has use for 3D point clouds, as explained in [14], 3D point clouds may be used to generate accurate 3D models of structures, as well as geometry quality inspection.

1.2.3 Education

Point clouds for emotional analysis of students, as presented in [15], show promise for recognizing emotions of learners. This technology could be used to inform an instructor when a topic is understood or more review is needed, which may be of particular use in remote learning applications where an instructor might not be able to see the learners.

1.2.4 Autonomous Navigation and Autonomous Vehicles

Use of 3D point clouds in autonomous vehicles is described in [16]. 3D point clouds may be used to create environment maps and localize a vehicle within an existing map, 3D point clouds may further be applied in environment perception, for example to locate pedestrians.

1.3 MPEG Standardization Efforts.

Having discussed possible applications of point clouds, the need for efficient storage and transmission becomes evident. As shown on the MPEG-pcc website [7] there is an ongoing effort to develop coding tools for 3D point clouds, some results, such as V-PCC and G-PCC, are briefly described below. Point clouds can range in size from less than 1 million data points at 10-bit geometry or less, up to many millions of points at bit depths of 16 or more bits per axis element. This can result in uncompressed sizes ranging from 3.07 MB for longdress_vox10_1300 up to 456.8 MB for ulb_unicorn_hires_vox20 for the geometry alone [12]. Compared to 23.73 MB for a 4K UHD RGB image with 8-bit color per channel. (3840x2160x24 bits per frame). Given the possibly very large sizes of point clouds, the need for efficient representation is clear.

1.4 Current Work and Related Approaches

1.4.1 Video Point Cloud Compression V-PCC

One approach for point cloud compression is to leverage existing 2D image and video compression tools. This has the advantage of drawing on decades long experience and work in these fields, however the translation layer of 3D into 2D objects may fail to exploit some of the structures of the problem that a native 3D approach could. None the less Video based point cloud coding (V-PCC) is an effective approach with good results.

In short V-PCC, as explained in [17], takes a 3D point cloud and projects it onto a 2D surface from different directions. The thus resulting projections, called patches, can then be coded using traditional 2D video or image coding standards. At the decoder the 2D coding is first undone to recover the patches, and finally using metadata about the captured patches the original point cloud can be recovered.

To lend more detail: The normal vector of each point is estimated and compared with the available projection directions to group points based on which projection direction best matches their normal vector. These groups of points are then projected onto a 2D plane where patches are generated to encode the distance of the point to the projection plane, while the other 2 coordinates of the point remain the same. To overcome auto-occlusions, as more than 1 point may be mapped to one location on the reference plane, this approach allows for multiple remedies: A near and far image may be created, encoding geometry points at different distance intervals from the reference plane. The number of projection directions may be increased, instead of for example using the 6 faces of a bounding cube, planes at 45-degree angles may be added. Additionally, any points that are still left uncoded at the end of the regular coding process may be added to auxiliary patches and coded directly. The exact choice of which tools to use depends on the desired encoding complexity and performance. Once the 2D patches are generated, they are packed into 2D images. For efficient 2D coding they must be packed as tight as possible. Empty space between Patches is filled in a manner to improve compression.

Once this is done an occupancy map is generated to indicate which pixels in the 2D image are valid and which were filled to aid 2D coding. Finally, Meta data about patch orientation and projection direction is encoded. If attribute information is to be transmitted as well, it can be mapped into 2D patches and images in a similar manner, taking advantage of the geometry

projections determined above. The resulting images can then be coded using any image or video coding standard.

1.4.2 Sparse Tensor Point Cloud Geometry Compression Sparse-PCGC

Point Cloud Geometry compression using Sparse Tensor based Multiscale Representation is proposed in [18]. This is a down sampling-based approach, where only level i-1 and i are considered at a time. Level i-1 is the down sampled version of level i by 2 in every direction. The approach considers Positively Occupied Voxels (POV), Negatively Occupied Voxels (NOV), and Most-Probable Positive Occupied Voxels (MP-POV). If a voxel at level i is a POV then the down sampled group of 8 voxels at level i-1 this voxel belongs to is also a POV. Likewise for NOVs.

The decoder receives the down-sampled version of the point cloud from level i-1, as well as a local neighborhood geometry feature stream. This information is passed to a sparse CNN where features from level i-1 are aggregated, the point cloud is up sampled from level i-1 to level i, converting every POV at level i-1 to 8 MP-POV at level i. Geometry features are again aggregated here, and occupation probability for MP-POVs is estimated, a threshold can be applied to finally decide on occupation status.

At the encoder, along with down sampling, a Sparse Local Neighborhood Estimation is preformed which attempts to encode the local neighborhood information into features to be sent to the decoder along with the down sampled point cloud.

Different variations exists where parts of MP-POV groups are treated differently, for example an 8-stage mode is presented that treats each of the 8MP-POVs in an up sampled block independently and uses information from previously decoded MP-POV for the next MP-POV. This improves results but increases complexity.

This method only requires 2 adjacent levels of a point cloud and uses the same architecture for all adjacent levels. This greatly improves efficiency compared to a model using all resolution layers of a point cloud but may fail to account for changing distributions at different scales.

1.4.3 Geometry Point Cloud Compression G-PCC and MPEGs TMC-13

G-PCC is the basis for the current MPEG test model for Geometry point cloud compression. It is implemented as MPEGs TMC-13 [19,20] We will briefly describe the geometry coding portion of G-PCC, both to give background, and as TMC-13 will perform the encoding and decoding of point clouds after processing via the proposed method.

Point cloud coordinates are first transformed from world coordinated to frame coordinates, as described above. Points are vowelized, that is a resolution, d, is chosen and the bounding box is divided into 2^d segments in each direction resulting in unit cubes. This forms the 3D analog to pixels. Coordinate points are then grouped with the center of the voxel into which they fall, if more than 1-point falls into a voxel it is removed, and we are effectively down sampling the point cloud.

To code this information the point cloud is represented as an octree, the original bounding box containing all points is the root of the octree, it is then divided into 8 cubes, each of these 8 cubes

is again divided into 8, until a desired level of depth is reached. At each level occupancy of each cube can be signaled using 1 bit. A cube is empty if no occupied voxels are contained within it, it then receives 0 in the octree and its branch is terminated. This fashion proceeds until a certain level of depth, where the leaves indicate individual voxels. If the depth of the tree is equal to the resolution d chosen for voxelization then no down sampling is preformed, if it is less a down sampling occurs, and a tree depth larger then voxel resolution has no benefit.

The thus generated octree symbols are then entropy coded. The tree may be coded either in a tree-fashion, where occupancy at each level is coded sequentially, or in a Direct coding mode (DCM), where the location of individual points is signaled directly. DCM is useful to code outlying nodes in otherwise empty neighborhoods, as it reduces overhead for those points and allows for better estimation of distributions in areas where points are grouped densely. Entropy coding of Octrees is complex, and not directly at the heart of this work, therefore we will not go beyond this brief overview.

1.4.4 Geometric Residual Analysis and Synthesis for Point Clouds: GRASP

The approach proposed in [21] was selected as the framework into which this work will eventually be integrated, although our approach presented here may be useful in other frameworks or in a stand-alone fashion. Here we will introduce the GRASP-NET approach.

GRASP exhibits some similarity to intra prediction in video and image coding but adapts this to 3D geometry and applies ML to handle the irregular structure. In brief: an incoming Geometry point cloud (GPC) is first quantized, and the quantized version is transmitted to the decoder. At

the encoder the quantized version is dequantized to create a coarse version of the GPC. The coarse version is subtracted from the input point cloud to generate a residual. This residual is processed by an ML architecture, the output of which is then entropy coded and sent to the decoder as a refinement bitstream. Here the process is performed in reverse and the point reconstructed

To provide more detail: Consider an incoming geometry point cloud and quantize it. The quantized point cloud is entropy coded and sent to the decoder; at the encoder it is dequantized to obtain a coarse point cloud.

This coarse point cloud and the original point cloud are passed to a geometry subtraction module using kNN to associate groups of points in the input cloud with points in the coarse point cloud.

These sets of points are then passed to a Point Analysis Network to generate latent features for each point in the coarse point cloud. These latent features are down sampled using CNN architecture to remove redundancy, and finally the down sampled feature set is entropy coded and sent to the decoder.

At the decoder the coarse point cloud is constructed by dequantizing the quantized version, as was done in the encoder. Down sampled latent feature bitstream is passed to a CNN with de-convolutional layers and then passed through a Point syntheses network consisting of multiple MLPs to reconstruct an approximation of the residual obtained in the encoder. This approximated residual, along with the coarse point cloud are passed to a geometry addition model, the output of which is the reconstructed point cloud. For comparison purposes in this work we will use the quantized-dequantized point cloud generated by grasp-net as an anchor, as this is the component we aim the replace in later integration.

2. BACKGROUND

2.1 Notation and Point Cloud Structure

Geometry point clouds can be viewed as collections of length 3 vectors. Each such vector describes one point. In the frame coordinate system, all components are integers with the maximum value fixed by the bit depth. For a bit-depth n, the greatest value of any component is 2^n-1. We define the axes as X, Y and Z, where the first point in a vector is assigned the X-axis, the second the Y-axis and the 3rd the Z-axis. For the purpose of compression, this assignment is arbitrary so long as it is consistent.

Let us now define some notation: Input point clouds in the frame coordinate system will be denoted $\mathcal{X}$ with indexing space $I(\mathcal{X})$, its axis wise components we will denote as X, Y and Z.

Let $\mathcal{Y}$ be the point cloud after wavelet transform, with indexing space $I(\mathcal{Y})$, and denote its axis wise elements as x, y and z

Further consider axis wise transform blocks $\mathcal{B}_{a,i}$ where a denotes the axis and i the transform block number. Elements in each axis are split into transform blocks such that all elements in a Transform block are continuous, where continuity will be defined below.

Using this definition each Axis can be written as a collection of transform blocks, that is

$$X = \{\mathcal{B}_{x,0}, \mathcal{B}_{x,1}, \mathcal{B}_{x,2}, \dots\ \mathcal{B}_{x,K}\}, Y = \{\mathcal{B}_{y,0}, \mathcal{B}_{y,1}, \mathcal{B}_{y,2}, \dots\ \mathcal{B}_{y,K}\}, Z = \{\mathcal{B}_{z,0}, \mathcal{B}_{z,1}, \mathcal{B}_{z,2}, \dots\ \mathcal{B}_{z,K}\}$$

Where $\mathcal{B}_{x,i}, \mathcal{B}_{y,i}, \mathcal{B}_{z,i}$ contain the X, Y, Z coordinate components for all points in transform block i and are ordered. We denote transform blocks after wavelet transform as $\widehat{\mathcal{B}}_{x,0}$ where the letter subscript denotes the axis and the number the index of the transform block.

2.2 Wavelet Transform and Arbitrary Region Of Support Wavelet Transform

2.2.1 Wavelet Transform

We will begin this section by reviewing a standard wavelet transform and expand to the AROS wavelet transform used in this work. Sub Band coding schemes are described in detail in [22]. The 5/3 filters chosen for this work are developed in [23], where the authors note the 5/3 filters ability to create "visually pleasant and smooth outputs" and additionally point out that 5/3 filters may be implemented with a limited number of shifts and adds, resulting in a lower computational complexity, a valuable property for any compression approach. We will thus move forward with the 5/3 filters.

Consider a signal S of even length and define analysis low and high pass filters h and g.

The standard wavelet transform is performed by first odd symmetrically extending S, then filtering by both h and g, and finally down sampling the filtered results into Low-Pass and High-Pass components, where the Low-Pass filtered result retains elements on even indices and the High-Pass filtered result those on odd indices.

$$\text{Let: } S = (x_0, x_1, x_2, x_3, \ldots, x_N),\ h = (h_0, h_1, h_2, h_3, h_4),\ g = (g_0, g_1, g_2)$$

Where N is even, and h and g are the biorthogonal 5/3 filters analysis filters. First, we odd-symmetrically extend S by 2 samples on each side.

$$S_{ext} = (x_2, x_1, x_0, x_1, x_2, x_3, \ldots, x_{N-2}, x_{N-1}, x_N, x_{N-1}, x_{N-2})$$

Next, we filter by convolution

$$S_{lp-conv} = S_{ext} * h \text{ and } S_{hp-conv} = S_{ext} * g \text{ giving:}$$

$$S_{lp-conv}[i] = \sum_{n=0}^{5} S_{ext}[n+i]h[n]\,; i \in [0,1,\ldots,N]$$

$$S_{hp-conv}[i] = \sum_{n=0}^{3} S_{ext}[n+i+1]g[n]; i \in [0,1,\ldots,N]$$

We note first that we've used symmetry in the filters to simplify the convolution sum, secondly by adding an offset of 1 in the summation for the shorter high pass filter we can use the same extended signal for both convolutions. The result of the convolution will have the same length as the input.

After convolution we must down sample to obtain $S_{LP} = S_{lp-conv}[0::2]$ and $S_{HP} = S_{hp-conv}[1::2]$. Where we use the notation [a::b] to signify that we retain the sample at index a, and every b-th sample thereafter.

Reconstruction is the opposite of decomposition, we first up sample high and low pass signals, inserting 0s at odd indices for Low-Pass and at even indices for High-Pass. Next, we odd-symmetrically extend and filter by the Synthesis high and low pass filters. The results are added element wise, and the original signal is recovered.

To achieve compression in this approach we transmit and reconstruct using only the Low-Pass components. Thus, we only need to generate the Low-Pass component.

To describe the method used here: consider S_{ext} as the odd-symmetrically extended input signal, filter as described above and retain only those samples on even indices.

$$S_{lp} = (S_{ext} * h\,)[i];\;\; i\ even$$

S_{lp} can be transmitted or processed further. To recover the input from S_{lp} we must first up sample and then extend odd-symmetrically, and finally filter by the synthesis Low-Pass, denoted $\bar{h}$:

$$S_{lp-us}[i] = \begin{cases} S_{lp} & i\ even \\ 0 & i\ odd \end{cases}$$

$$S_{lp-ext} = \big(S_{lp-us}[2], S_{lp-us}[1], S_{lp-us}[0], S_{lp-us}[1], \ldots,$$

$$S_{lp-us}[N-2], S_{lp-us}[N-1], S_{lp-us}[N], S_{lp-us}[N-1], S_{lp-us}[N-2]\big)$$

$$S_{rec} = S_{lp-ext} * \bar{h}$$

S_{rec} is now an approximation of the input signal recovered from the Low-Pass component only. This manner of wavelet transform is suited for even length signals with 4 or more samples, in the next section we will extend it to arbitrary length signals.

2.2.2 AROS Wavelet Transform

Above we explained the standard wavelet transform. This transform is suitable for continuous signals, but if large discontinuities exist within a signal, intermediate points to bridge the discontinuity will be introduced by the filtering operation. For point cloud geometry compression

this would create outlying points where previously there were none. Hence, we need an approach to only consider signals with values that are close.

We will use the transform blocks $\mathcal{B}$ of continuous regions in each axis introduced in 2.1. To prevent creation of outlier points we will perform wavelet transforms on these Blocks independently, and then concatenate the results from all such blocks in an axis to create the transformed coordinate axis. This presents 2 immediate problems. Blocks may have odd or even length, and furthermore if we always started down sampling at the first element in a block, as we've done in the standard wavelet transform, the number of points in the low pass component can vary widely. To see this, consider the case where all transform blocks are length 3, the resultant Low-Pass signal would have 2/3 the length of the input signal instead of half.

Thus, we apply the AROS wavelet transform as described in [24], where down and up sampling indices are informed by the original index space $I(\mathcal{X})$ and not the index space $I(\mathcal{B})$ of transform blocks. In a sentence: we start down and up sampling at those points that occupy even indices in $\mathcal{X}$, regardless of their index parity in the transform block $\mathcal{B}$. The difference is then:

$$S_{lp}[j] = S_{conv}[i] \; if \;\; I(i) \; is \; even$$

$$S_{lp-upsample}[j] = \begin{cases} S_{lp} & I(j) \; even \\ 0 & I(j) \; odd \end{cases}$$

Where, in a slight abuse of notation we take $i \in I(\mathcal{B})$ and $I(i)$ to be the mapping of i to $I(\mathcal{X})$, that is i is the index within the transform block, and $I(i)$ maps this index to the original point cloud.

This new means of Down and Up sampling brings with it the need for an additional parameter we will call Transform Parity, or T-parity for short. In the standard wavelet transform the length

of the reconstructed signal can be inferred from the number of received Low-Pass components alone, however with the AROS up/down sampling scheme the number of received Low-Pass components does not fix the number of elements in the input signal. Depending on the parity of the start index in the original indexing space and the number of elements in the transform block 4 different cases exist for the same number of received Low-pass coefficients. They are listed in Table 1.

Table 1: Cases Resulting in 2 Received Symbols. the filtered signal shows the filtered transform block and the preceding / following values needed to explain the different cases, assume that this is the very start of the overall input signal. The transform block in question is indicated by using Capital, Bold letters, values not in the current transform block are lower case letters. The Down sampled column shows the result after down sampling the current transform block only. Ideal up sampled components shows the result we should obtain after up sampling for ideal reconstruction. T-parity of current block and T-parity of next block show the parity of the start index of the current and next block respectively, in the input indexing space. These quantities allow us to correctly infer reconstructed length.

Case	Filtered signal in original indexing space	Down sampled components	Ideal up sampled components	T-parity of current block	T-parity of next block
Even-start, Even-length	[**A, B, C, D**, e, f, . . .]	[A, C]	[A, 0, B, 0]	Even	Even
Odd-start Even-length	[a, **B, C, D, E,** f, g, . . .]	[C, E]	[0, C, 0, E]	Odd	Odd
Even-start Odd-length	[**A, B, C,** d, e, g, . . .]	[A, C]	[A, 0, C]	Even	Odd
Odd-start Odd-length	[a, **B, C, D, E, F,** g, h, . . .]	[C, E]	[0, C, 0, E,0]	Odd	Even

We see that the 4 cases listed all produce the same number of low pass coefficients, but that 3 different lengths of the input signal are possible. Furthermore, the placement of 0s in up sampling is not consistent either. If we track the parity of the start index of all transform blocks in the original index space, as shown in the table under the T-parity columns, we can infer the given case by looking at the current and next transform parity. Thus T-Parity must be either signaled or inferred at the decoder. Section 2.5 gives details on how this is done.

The nature of point clouds give rise to 2 special cases. Transform blocks are unrestricted in length. Since our longest filter is length 5, we need to extend at most by 2 elements on either side. Hence any signal of at least 3 elements can be odd-symmetrically extended in a normal fashion. Signals of length 2 or less cannot.

Length 1 signals cannot be filtered using wavelet transforms, as we require at least 1 Low-Pass and 1 High-Pass component, even if we do not send the High-Pass component. Hence the only way to treat isolated points is to scale and copy them directly to the High-Pass or Low-Pass components, depending on the parity in $I(\mathcal{X})$ they occur at, or remove them in pre-processing.

Length 2 signals can be filtered but need a special extension procedure. The extension used here is as follows: we first perform odd-symmetric extension by the 1 sample available to us, then repeat the procedure on the result. This is shown below, where the original signal is marked in bold

$$S_{in} = (\boldsymbol{x_0}, \boldsymbol{x_1})$$

$$S_{ext-1} = (x_1, \boldsymbol{x_0}, \boldsymbol{x_1}, x_0)$$

$$S_{ext-2} = (x_1, x_0, x_1, \boldsymbol{x_0}, \boldsymbol{x_1}, x_0, x_1, x_0)$$

Since we need at most an extension by 2 elements the result is:

$$S_{extended} = (x_0, x_1, \boldsymbol{x_0}, \boldsymbol{x_1}, x_0, x_1)$$

Which has the convenient property of being 3 concatenated copies of the input signal. Other filter choices beyond the 5/3 filter are possible, only the extension would have to be adjusted.

A final challenge in filtering point clouds is bit-expansion. Given the 5/3 filters a bit expansion may occur for higher value signals. The 5/3 filters in their original form are given in table 2 below.

Table 2: Original 5/3 Biorthogonal Filter Values.

Filter name	Values
Analysis Low pass h	$\sqrt{2}\left(\frac{-1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, \frac{-1}{8}\right)$
Analysis High pass g	$\sqrt{2}\left(\frac{1}{4}, \frac{-1}{2}, \frac{1}{4}\right)$
Synthesis Low Pass $\bar{h}$	$\sqrt{2}\left(\frac{1}{8}, \frac{1}{4}, \frac{-3}{4}, \frac{1}{4}, \frac{1}{8}\right)$
Synthesis High Pass $\bar{g}$	$\sqrt{2}\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$

This produces a gain of more than 1 for some signals in the low pass components. This is problematic for higher values. An example of a constant signal k will illustrate the problem.

$$\text{Let } \mathcal{B}_{x,l} = (k, k, k)$$

$$then\ \widehat{\mathcal{B}}_{x,l} = \sqrt{2}\left[\left(\frac{-1}{8}k + \frac{1}{4}k + \frac{3}{4}k + \frac{1}{4}k + \frac{-1}{8}k\right), \left(\frac{-1}{8}k + \frac{1}{4}k + \frac{3}{4}k + \frac{1}{4}k + \frac{-1}{8}k\right), \left(\frac{-1}{8}k + \frac{1}{4}k + \frac{3}{4}k + \frac{1}{4}k + \frac{-1}{8}k\right)\right] = \sqrt{2}(k, k, k)$$

where we omitted to show the odd-symmetric extension for simplicity.

if $k = 2^n-1$, the largest possible value for the given geometry bit depth, the resultant transform has values exceeding the geometry bit depth. This is problematic in entropy coding. The solution is simple: Since the condition we want to enforce is DC-gain of 1 in the reconstructed point cloud, we can apply our scaling in either analysis or synthesis. If we push all scaling to the decoder, we can keep a scaling of at most 1 in the Low-pass component on the synthesis side, and DC-gain of 1 in the reconstructed point cloud. This can be achieved by multiplying the filter coefficients by $1/\sqrt{2}$ in the encoder, and by $\sqrt{2}$ in the decoder. The filter values used here are shown in table 3.

Table 3: 5/3 Biorthogonal Filters as Used in this Work.

Filter name	Values
Analysis Low pass h	$\left(\frac{-1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, \frac{-1}{8}\right)$
Analysis High pass g	$\left(\frac{1}{4}, \frac{-1}{2}, \frac{1}{4}\right)$
Synthesis Low Pass $\bar{h}$	$2\left(\frac{1}{8}, \frac{1}{4}, \frac{-3}{4}, \frac{1}{4}, \frac{1}{8}\right)$
Synthesis High Pass $\bar{g}$	$2\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$

2.3 Quantization

To achieve variable rate, we must quantize our filtered results. The quantization approach used here is a version of the approach from [9] adapted to work with our code. If we consider the Transformed point cloud $\mathcal{Y}$ the quantization applies a scaling of less than 1 to two of the three coordinates. Suppose we have a point

$$\boldsymbol{y}\epsilon\mathcal{Y}, \boldsymbol{y} = (y_1, y_2, y_3)$$

and wish to quantize if by a factor s, the resultant point will be

$$Q(\boldsymbol{y}) = \tilde{\boldsymbol{y}} = (y_1, sy_2, sy_3), s \leq 1.$$

We do this for all points. Inverse quantization is achieved by

$$Q^{-1}(\tilde{\boldsymbol{y}}) = \bar{\boldsymbol{y}} = \left(y_1, \frac{1}{s}y_2, \frac{1}{s}y_3\right),$$

where s is shared across the encoder and decoder. In addition to creating duplicate points that can be removed this approach forces points closer to each other, which is beneficial in entropy coding, as the next section will explain.

2.4 Geometry Entropy Coding

The entropy coder used here is MPEGs TMC-13 software, source code obtained from [20] and complied locally, details on the approach are listed in [19]. TMC-13 is an Oct-tree encoding scheme that successively subdivides the space into 8 cubes, and signals occupancy for each cube

using 1 bit. Thus, a tree is formed, where the leaf nodes represent individual points, and the root captures the whole space. The value of the Quantization approach in combination with this entropy coding scheme is that points are concentrated in fewer cubes as scaling by less than 1 “pushes” cubes toward the origin in the y and z directions. This reduces the number of occupied cubes per level.

Given that at each level we subdivide all cubes into 8 octants, each eliminated cube reduces the number of bits in all subsequent layers by up to 8^d where d is the depth wise distance from the removed node. This saving can be substantial and reduces the entropy under the given coding scheme compared with uniform down sampling or other approaches that remove points in a less systematic fashion. This approach is also described in 1.4.3.

2.5 Symbol Stream Parsing

As stated in 2.2.2, we need the parity of start indices in the original index space to reconstruct to the proper length, and with proper up sampling. Additionally, since we concatenate all the transform blocks after the AROS transform, we need to determine where one block ends and the next begins. We do this by explicitly sending start indices and T-parity symbols. Depending on the chosen continuity condition these quantities may or may not be inferable from the bitstream. The strictest continuity condition is to allow adjacent values to be none decreasing only and differ by at most one.

$$\forall x_i, x_{i+1} \in \mathcal{B}_i : \quad x_{i+1} \epsilon \{x_i, x_i + 1\}$$

For this strict definition start indices of transform blocks cannot be inferred from the Low-Pass symbol stream alone, to show this consider 2 adjacent transform blocks

$$\mathcal{B}_0 = (n, n+1, n+2, n+3, n+4, n+4)$$

$$\mathcal{B}_1 = (n+6, n+6, n+7, n+8, n+9, n+10)$$

$$n \in \mathbb{Z}^+$$

And their Low-Pass Wavelet transforms

$$\widehat{\mathcal{B}_0} = \left(\ldots, n+2, n+\frac{25}{8}, n+4, n+\frac{17}{4}\right)$$

$$\widehat{\mathcal{B}_1} = \left(n+\frac{21}{4}, n+\frac{23}{4}, n+\frac{55}{8}, n+8, \ldots\right)$$

Consider first the case where $\mathcal{B}_0$ has even T-parity, that is the first value occurs on an even index in the original index space. The Low-Pass symbol stream after down sampling would then be:

$$\mathcal{X}_{T-even} = \left(\ldots, n+2, n+4, n+\frac{21}{4}, n+\frac{55}{8}, \ldots\right)$$

and the element wise difference is

$$\mathcal{X}_{diff-even} = \left(\ldots, 2, \frac{\mathbf{5}}{\mathbf{4}}, \frac{13}{8}, \ldots\right) = (2, \mathbf{1.25}, 1.625)$$

For Odd T-parity we obtain

$$\mathcal{X}_{T-odd} = \left(\ldots, n+\frac{25}{8}, n+\frac{17}{4}, n+\frac{23}{4}, n+8, \ldots\right)$$

$$\mathcal{X}_{diff-odd} = \left(\ldots, \frac{9}{8}, \mathbf{\frac{3}{2}}, \frac{9}{4}, \ldots\right) = (1.125, \mathbf{1.5}, 2.25)$$

Where the difference value at the block boundary is marked in bold for both cases.

In both parity cases we see that we have differences internal to the blocks that are greater than the difference at the block boundary. This means we cannot infer block boundaries using a lower bound on the differences at block boundaries. Since differences across block boundaries can be large for far separated blocks, we cannot use an upper bound here either. Hence block boundaries are not distinguishable by difference alone for all cases. As we have no other way of knowing how long a transform block is at the decoder, we cannot in general tell which value is at the boundary of a transform block. Therefore, we must signal this quantity explicitly, at least for the continuity definition described above. A similar counter example can be constructed to show Transform parity cannot be inferred in all cases either, so we choose to signal explicitly. The continuity condition chosen was ultimately less strict than the one used in this example. However, we tested several definitions, and explicitly signaling Start Index and T-Parity simplified testing greatly. Once an optimal continuity condition is decided the ability to infer these quantities from the symbol stream should be tested again.

2.6 Objective Quality Assessment

Objective quality measurements for point clouds are defined by the ISO in [25] and further listed on the reporting template [26]. MPEG currently considers two different quality metrics and one rate metric, rate, PSNR D1 and PSNR-D2.

The rate is defined as

$$Rate = \frac{number\ of\ bits\ produced\ by\ the\ encoder}{number\ of\ points\ in\ the\ input\ point\ cloud}$$

measured in bits per point (bpp). Here the number of bits produced by the encoder should include all information needed to reconstruct the point cloud. For this approach that includes the bits in the base geometry bitstream as well as 2 streams of SEI, whose generation will be explained in section 3. As the base geometry bitstream is saved after entropy coding its size can be read directly from the file or gathered via pythons os library. SEI information will be estimated at the Shannon entropy limit, details are provided in section 3.6.

In addition to bpp metrics to assess the accuracy of reconstruction are also defined. In [25] the ISO proposed Common Training and Test Conditions to evaluate point cloud compression schemes submitted. For geometry only 2 objective quality metrics are proposed: PSNR-D1 and PSNR D2. We will briefly explain both and then justify our choice.

PSNR-D1 considers a point cloud to be evaluated, P2, and a reference point cloud P1. Using the nearest neighbor algorithm every point in P2 is associated with a point in P1. Euclidian distance between the points is measured as the difference vector between them and denoted $V(i,j)$. The MSE is calculated as

$$e_{P1,P2}(i) = \left|\left|V(i,j)\right|\right|_2^2;\ mse_{P1,P2} = \frac{1}{N}\sum_{k=0}^{N} e_{P1,P2}(k);$$

This process is repeated with the roles of P1 and P2 reversed, to find $mse_{P2,P1}$. PSNR-D1 is finally defined as

$$PSNR - D1 = 10\log_{10}\left(\frac{3 * (2^n)^2}{\max\left(mse_{P1,P2}, mse_{P2,P1}\right)}\right)$$

where n is the geometry bit-depth. This definition is chosen such that quality results are consistent regardless of submission order of point clouds.

A second metric, called PSNR-D2, or point to plane metric exists. As PSNR-D1 it considers a test point cloud and a reference point cloud and associates each point in the test-cloud with a point in the reference cloud using the nearest neighbor algorithm. However instead of calculating the error between these points directly a surface plane in the reference point cloud is defined in the local neighborhood of the identified point and the planes normal vector calculated. The distance vector between the reference point and the test point is then projected onto the normal vector. The resultant vector forms the basis for MSE calculation, and the remainder of the procedure is as in D1 metric.

The quality measurement tool used here is pc_error, it is used in PCGC [8] and Grasp net [9]. To produce PSNR-D2 pc_error requires a file with surface normals. The data we have available does not include these normal vectors, and hence we are limited to presenting RD results using the D1 metric only.

Additional Objective Metrics are proposed in [30]. Four basic groups of features are considered, Geometry based, Normal based, Curvature based, and Color based, where each feature type considers 6 different statistical measures to assess content quality, applied on group sizes of 6,12,24 and 48. Of all considered features Color based features correlate best with Subjective opinion scores, however our work does not include color, making these results of little use to us. The authors of [30] tested their methods using 2 datasets, and the performance of different metrics varies. Overall curvature-based metrics are successful, as is the standard deviation within a local group. Assessing the quality of the proposed result via these metrics may lead to

additional insights, however given only the consideration of MSE based metrics in standardization work we have elected to present only those.

3. WAVE-GRASP

This section will first give an overview of WAVE-GRASP and then explain the different components in detail. Wave GRASP is a lossy wavelet geometry compression approach that aims to improve on the quantization-inverse quantization approach for coarse point cloud compression used in [9]. The core principle is to replace simple quantization-inverse quantization with an Arbitrary Region of Support (AROS) wavelet transform along coordinate axes followed by quantization. The wavelet transform not only reduces the number of input points overall, but also causes an energy concentration, allowing us to cover more energy of the point cloud in fewer points. This comes at the cost of higher latency and requires us to sort the geometry to create continuous regions for good results.

3.1 Overview

The input to wave-grasp is the geometry component of a point cloud transformed to the frame coordinate system. This gets passed to a pre-processing block to facilitate the AROS transform. Here the cloud is sorted to reduce to total number of transform blocks and isolated points can be removed. Next, transform blocks are identified jointly in all 3 axes. The values within each transform block for each axis are then transformed using the AROS wavelet transform where Supplemental Encoder Information (SEI) regarding start indices in the symbol stream of the different transform blocks, as well whether the transform blocks start on indices with odd or even

parity is generated. The result is concatenated back into a point cloud and quantized by scaling 2 of the 3 axis components by a factor less than 1. Duplicate values thus created need to be removed and the SEI updated accordingly. Once this is done, we can entropy code the result. The base geometry will be encoded using the TMC-13 codec. To report RD results, we estimate the bits needed for SEI as the entropy limit. The decoder first decodes the bitstream, then inverse-quantizes the result, and finally performs the inverse wavelet transform, resulting in a Reconstructed point cloud. An overview of this process is provided in figure 2, and the individual steps will be explained in the following sections.

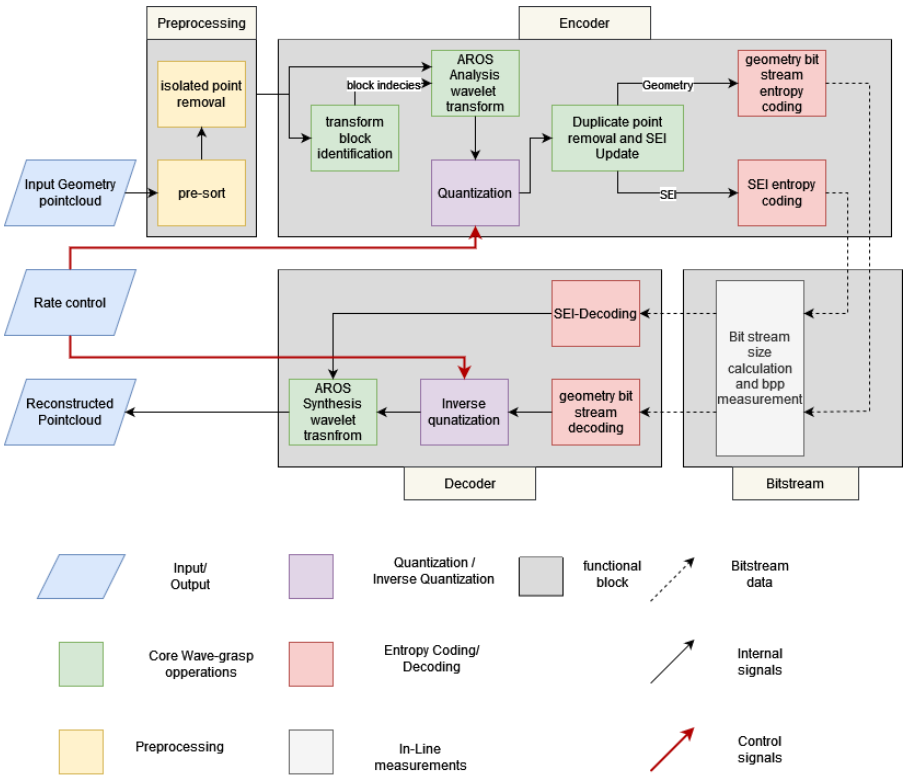


Figure 2 WAVE-GRASP System Overview. Green blocks are core components of WAVE-GRASP, while quantization (purple) and entropy coding(red) can be replaced with any quantization and entropy coding scheme respectively, so long as input-output relationships are maintained. Yellow blocks are pre-preprocessing blocks. Input and output blocks are colored blue, and the theoretical bitstream size estimation occurs in the grey block.

3.2 Preprocessing: Sorting and Isolated Point Removal

Geometry components of point clouds are in general order agnostic, to visualize a point cloud we simply occupy the signaled points. However, to successfully implement an AROS-wavelet transform we require continuous segments. The continuity will be defined in section 3.3, for now it suffices to say that each continuous segment requires 2 symbols of SEI, and furthermore segments of length 1, i.e. isolated points, pose issues in transform. Therefore, we need to sort the point cloud to minimize the total number of continuous segments and isolated points.

The sorting algorithm used here first sorts all points in the Point cloud jointly, first sorting in ascending order by the X-component, then Y then Z. We will call this sorting "Ascending sort", as it will be used throughout this process. Ascending sort creates an array where the X-component is non-decreasing, Y-components are non-decreasing within each constant X-segment, and Z-components are non-decreasing within each constant Y-segment. This approach exhibits good continuity in the X and Y axes, but very poor continuity in the Z-axis.

The next step in sorting we will refer to as "Sort for Transform" here we take the first point in the ascending point cloud and find all points where the Y and Z components are continuous. Off all these points we then select the first one found.

In practice, since we sort by X-first, even if we only check the Y and Z components for continuity the first selected point is likely to also be continuous in the X-axis, but computational cost is reduced by only checking 2 of the 3 components. Experiments show that this approach yields results comparable to checking all 3 axes explicitly. In figure 3 below we can see that checking all 3 axes instead of just the Y and Z has little effect, as shown by the blue and grey

curves, but allowing a decrease between adjacent points does yield some benefit. Note that base grasp can achieve perfect reconstruction at 1.02 bpp for this test case, so the far-right side of this graph is not relevant.

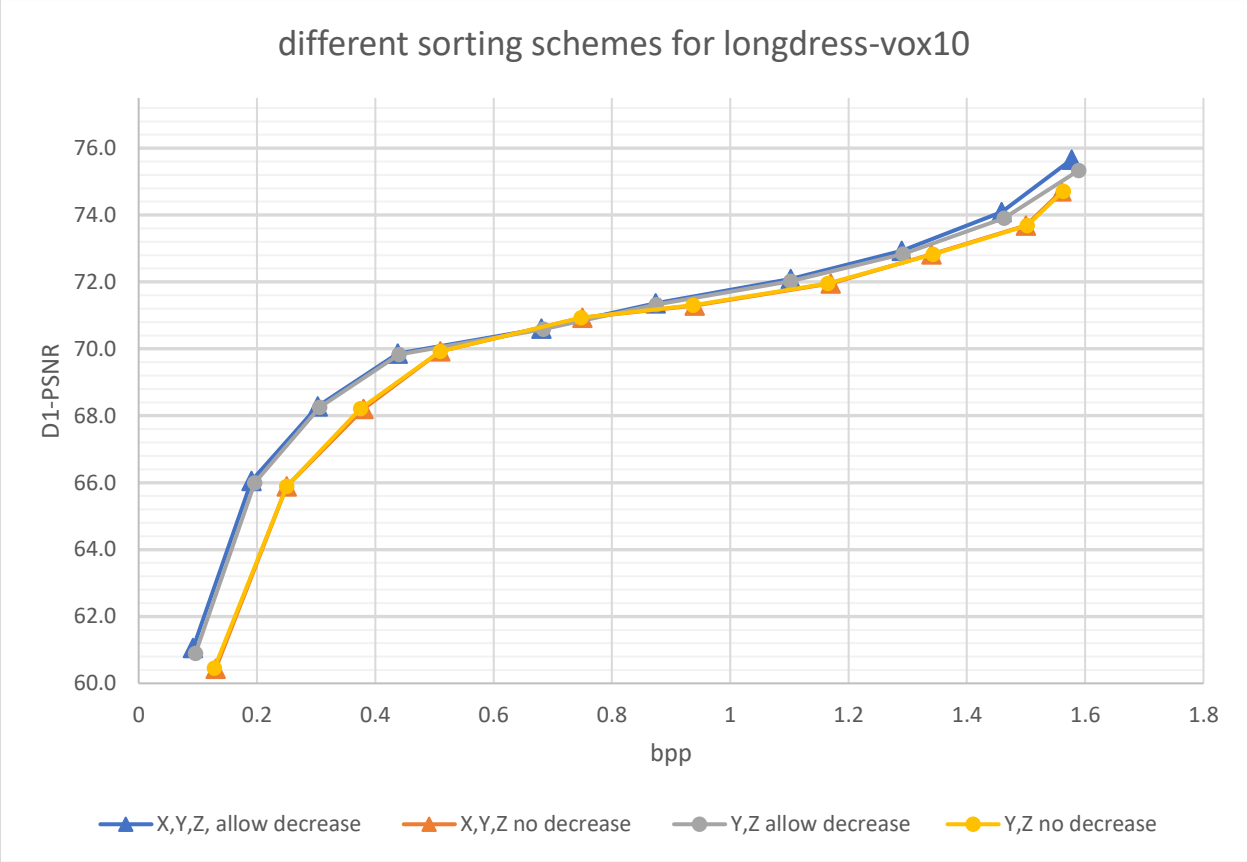


Figure 3 RD at Different Sorting Schemes for Longdress_vox10_1080. The blue curve sorts by X, Y and Z axis, and allows adjacent values to decrease by 1. The Orange curve sorts X, Y and Z as well but does not allow decreases, Grey, and Yellow repeat this pattern but only consider the Y and Z axis. Under the proposed approach sorting by X in addition to Y and Z has little effect, however allowing decreases between adjacent points yields gains around 2dB at low bit rates.

Once we have selected a point that is continuous, we append it to the current transform block and delete it form the input array. We then search for a point continuous to the point we just found, in the same fashion. This approach continuous until we can no longer find continuous points. Now the current transform block is complete and gets written out to the sorted array. We resort the input array with all processed points removed using Ascending sort described above and repeat the Sort for transform process. This continues until all points in the input have been processed.

After the sorted point cloud is obtained in the above manner, we remove all isolated points. An isolated point is defined as a continuous segment of length 1. These are disadvantageous to the AROS transform, as they incur large SEI overhead and increase geometry bitstream size for little improvement. Two examples, one for the sparse staue-klimt (sic) and another for the dense soldier vox-12 Point Clouds are shown in figures 4 and 5 respectively. These plots show that including isolated points leads to worse results. This is caused not only by increased SEI, but also by the increased size of the base geometry bitstream, as the entropy coder now needs to contend with a number of isolated points that are inefficient to represent.

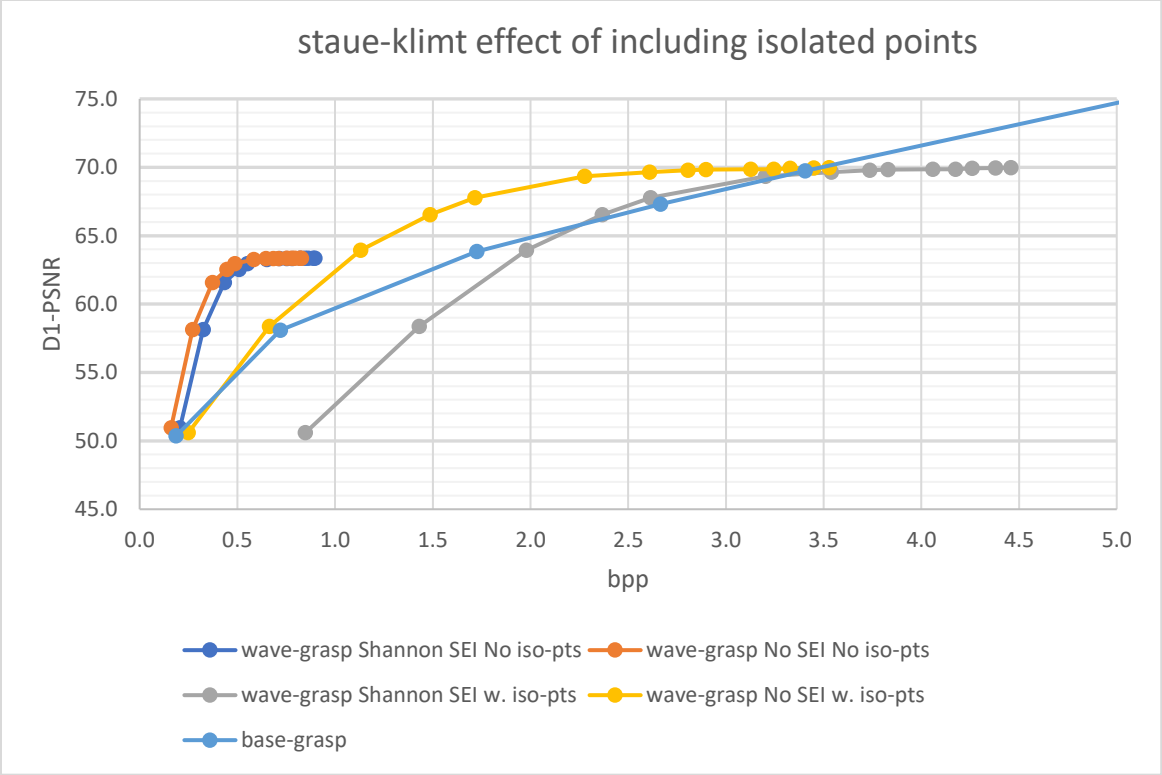


Figure 4 Staue-klimt With and Without Isolated Points in the Low Pass Component. Neither case is superior to base-grasp here, the SEI the case with isolated points yields nearly no benefit, while excluding isolated points is at least beneficial in the low bitrate regime. Thus, we conclude removing isolated points is beneficial here. Note that iso-pts stands for isolated points

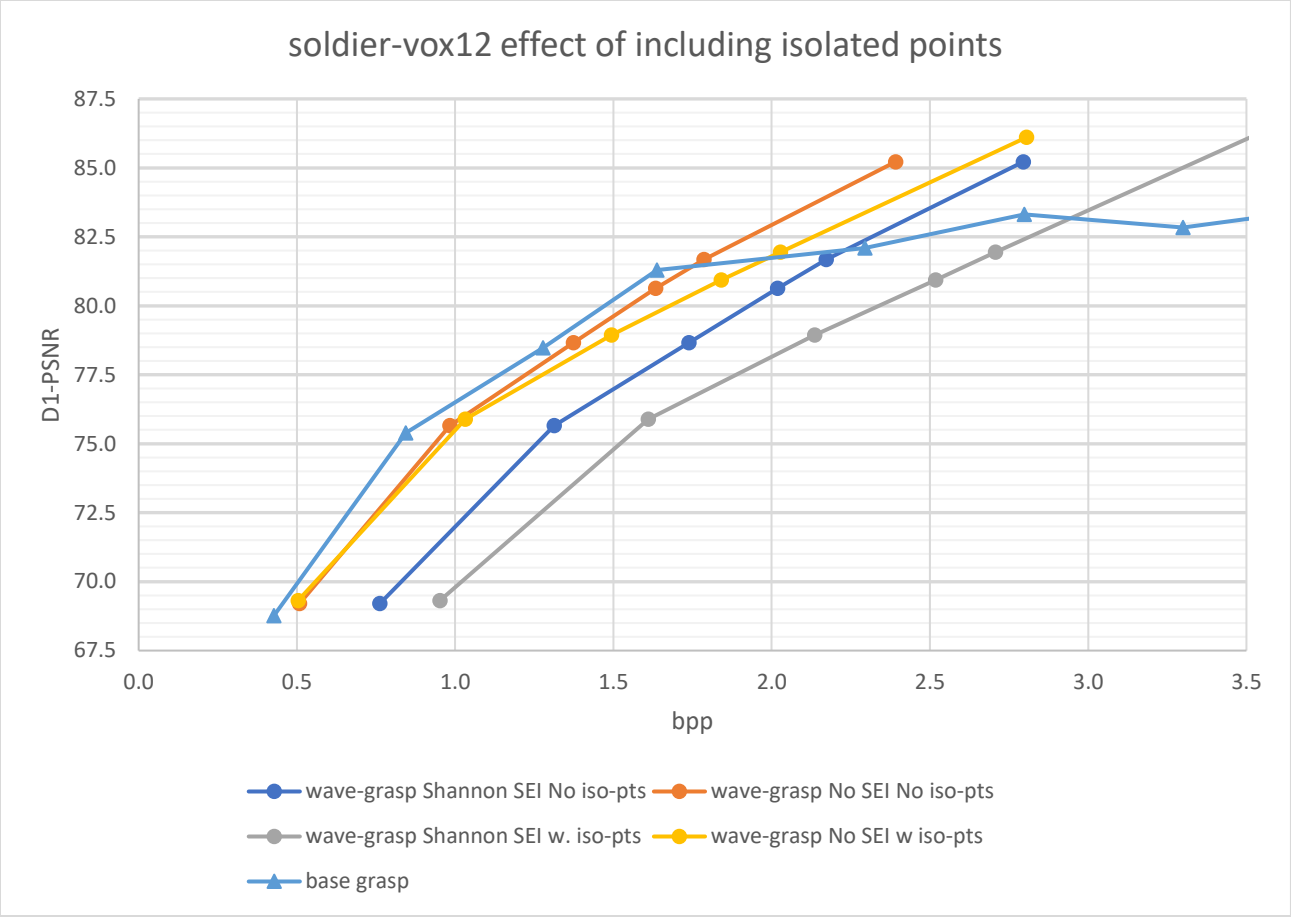


Figure 5 Soldier vox-12 With and Without Isolated Points in the Low Pass Component. This plot clearly shows that removing isolated points is beneficial. Note that this graph only considers quantizer scales between 0.1 and 1, while the results graph presented below considers quantizer scales as low as 0.01. We choose to include less scale points here to create an easier to follow graph.

The sorting applied here is not necessarily optimal but is reasonably computationally efficient and provides satisfying results for Solid point clouds. Future iterations of this work should first focus on improving the sorting. Furthermore, sorting can be computationally expensive, but this step may be pushed to the capture side and removed from the encoding process if desired. Once a sorted point cloud is obtained it does not have to be sorted again.

3.3 Transform Block Identification

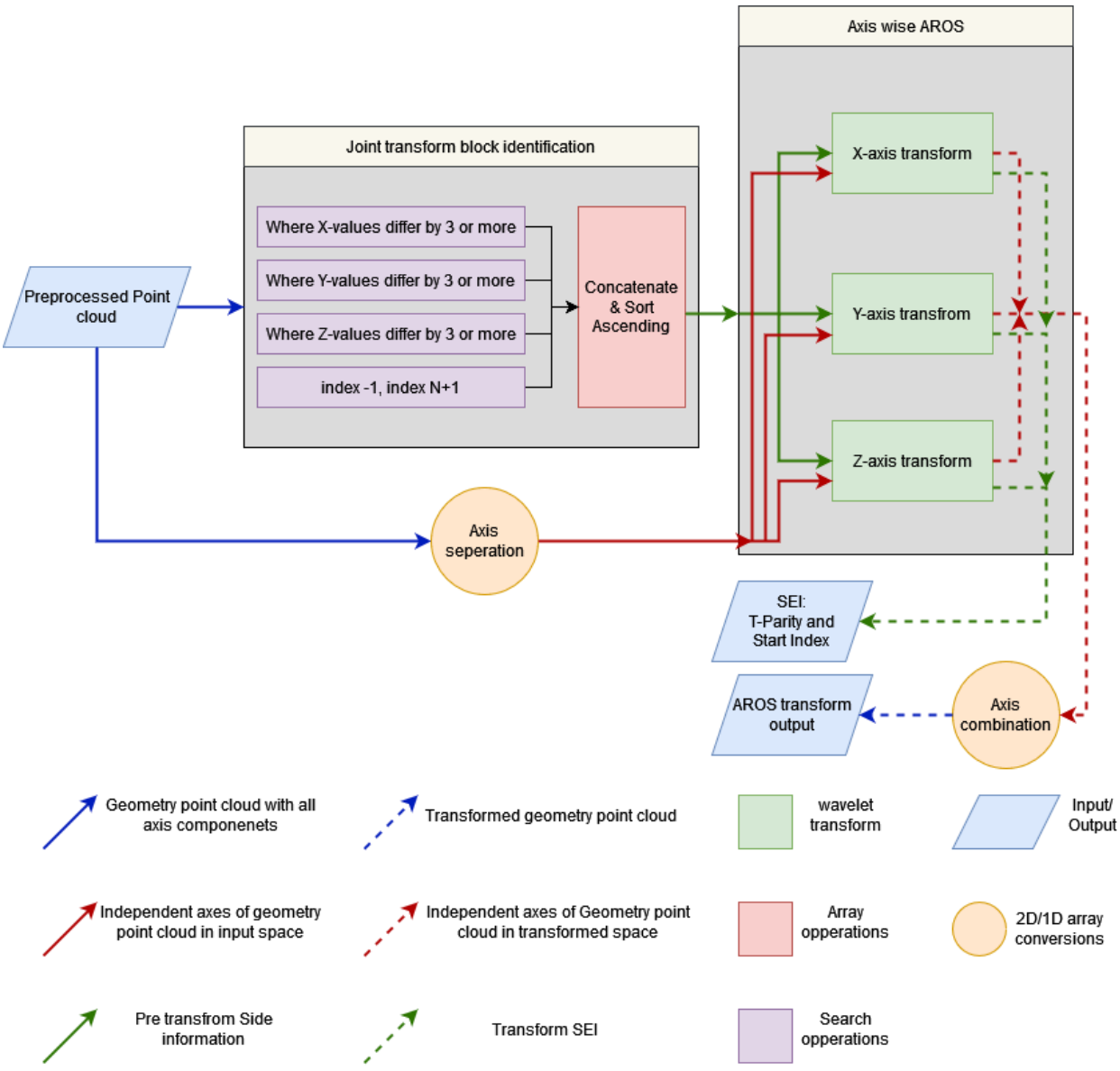


Figure 6 Transform Block Identification and AROS Transform Closeup.

Once the point cloud has been sorted, we can identify transform blocks. The AROS transform preforms an independent wavelet transform on blocks of arbitrary length; this unit, shown on the

left side of figure 6, is intended to furnish those blocks. As long as points within transform blocks do not differ substantially in value and the transform block is longer than 1 element the reconstruction quality from the block is not greatly affected by length. However, the amount of SEI needed and the length of the geometry bitstream itself may increases with number of transform blocks. Hence, we want as few as possible.

Components will be transformed axis wise, as shown in the right half of figure 6. It may be tempting to define different transform blocks for each axis individually as a discontinuity in one axis does not necessarily imply a discontinuity in all axes. This is not advisable as it causes an index drift in the SEI update function.

Transform blocks are identified by finding all indices where adjacent values in any axis differ by more than 3. While a stricter definition of continuity would lead to a more accurate wavelet transform, it would also increase the number of continuous sections and isolated points, which adversely affects bpp. Having a continuity threshold of +-3 allows points that would otherwise be isolated to be sorted into transform blocks. Figure7 shows the effects of different continuity definition on the RD curve for a 10-bit dense point cloud. Tested are continuities of [+3, -3], [+2, -2], [+1, -1] and {0,1}. We see that the continuity of {0,1} performs significantly worse than others. We chose the +-3 continuity condition in the hopes that it would help reduce isolated points in lower density point clouds, while it achieves comparable performance in in solid point clouds compared to +-1 and +-2 conditions.

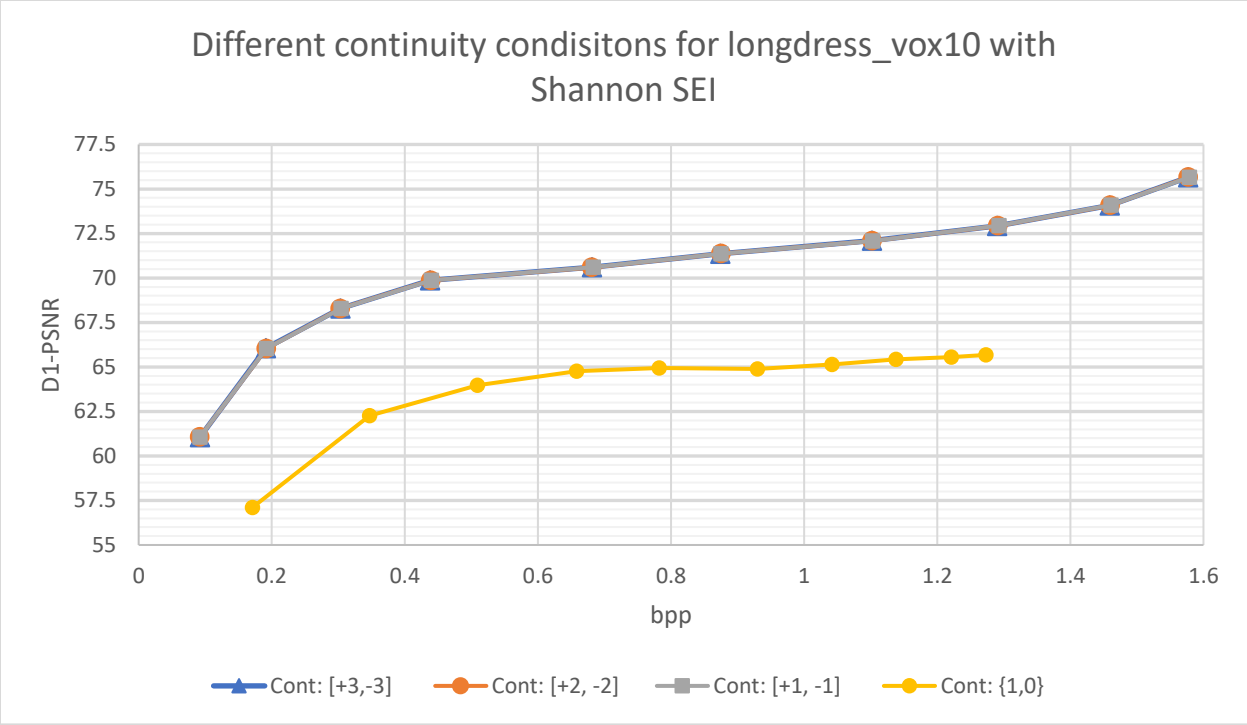


Figure 7 Longdress-vox10 With Different Continuity Conditions. A discontinuity is found if the difference between points is greater than the indicated values. For example, for the blue curves any points with absolute difference of greater than 3 are discontinuous. Note that the blue, orange and grey curves are overlapping.

Once we have identified all breaks in continuity in any axis, we concatenate the indices where these occur into one array. We also insert indexes that point to the first element and the N+1th element. This array then gets sorted in ascending order. The result is start indices for all transform blocks, where we can simply select the portion of the point cloud bounded by 2 consecutive entries in the start index array to obtain a transform block. This information along with the presorted point cloud is passed to the AROS transform unit.

3.4 AROS Wavelet Transform

AROS transform is performed axis wise; we first need to separate the point cloud into its X, Y and Z axes. We then select transform blocks using the start indices found above. We will explain the process in detail here for one axis, but note that all 3 axes operate identically and only differ in the values of the transform block they take in. Figure 8 below shows detail.

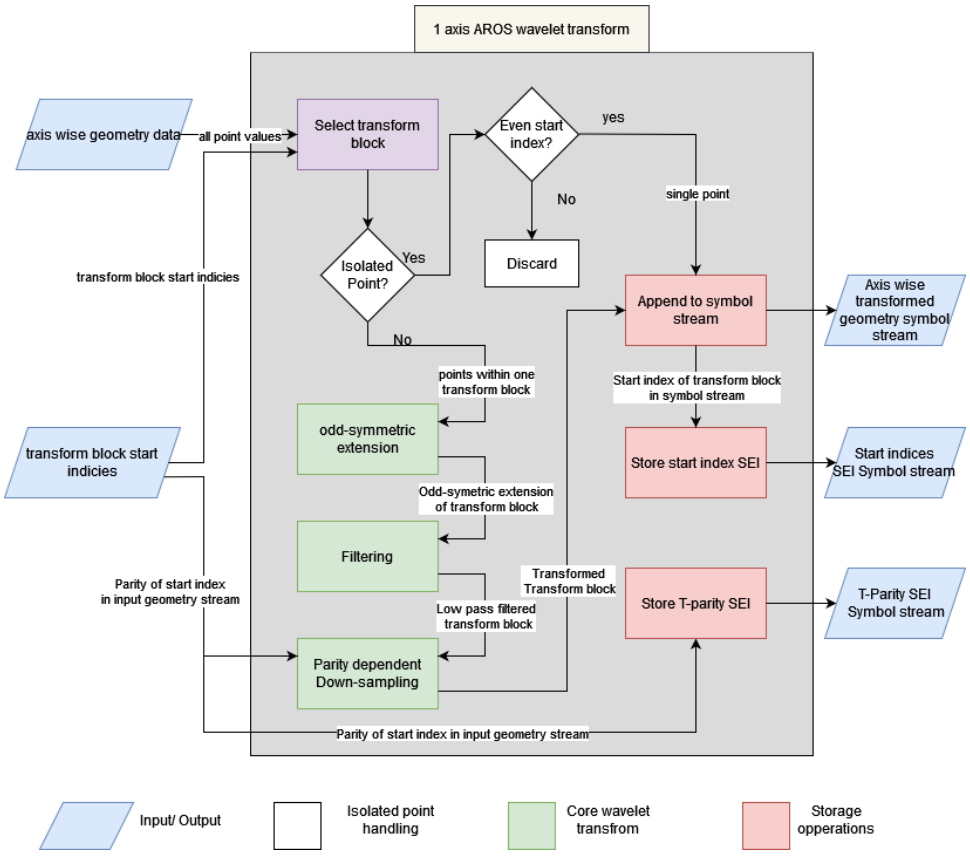


Figure 8 Single axis Wavelet Transforms. Note that we include a portion to handle isolated points. Above we show that it is beneficial to remove isolated points in preprocessing, but we retain the ability to handle them here should we want to.

Once a transform block is selected, we first determine if we have an isolated point. As discussed in 3.2 isolated point removal is an optional pre-processing step. Hence, we need to be able to accommodate isolated points if they occur. Points at even indices are copied to the Low-Pass symbol stream, while those occurring at odd indices in the input signal index space are discarded. This is in accordance with the AROS transform described in section 2.2.2. If the current block is not an isolated point, we first perform odd-symmetric extension. We then filter the result with only the Low-Pass filter, as High-Pass components are discarded we can omit High-Pass filtering.

After we have obtained the Low-Pass version of the transform block we need to down sample. Here we retain all Low-Pass coefficients that occur at even indices in the input point cloud indexing space. The resulting Low-Pass signal is appended to the Low-Pass symbol stream. We save the start index of the current block in the Low-Pass symbol stream as start index SEI, and the parity in the input index space of the first element of the block in T-parity SEI. Since all axes use the same transform blocks only SEI for one axis needs to be produced and sent. Once all axes are transformed, we concatenate them together to form the Low-Pass point cloud.

3.5 Quantization, Duplicate Point Removal and SEI Update

From 3.4 we receive the Low-Pass symbol streams concatenated into a Low-Pass point cloud, and 2 arrays with SEI information. The Low-Pass point cloud will be quantized as described in section 2, with the result containing duplicate points. Naturally we must remove these for efficient coding, but to successfully reconstruct we must also update the SEI.

To do so, we find all the indices of the first occurrence of unique points in the Quantized point cloud, these are the indices of points to be retained. We also take the compliment of these indices to find all the points that must be removed.

We then loop over the old start indices in the cloud with duplicates and consider one transform block at a time. By comparing the indices of points to be removed, we can determine how many points in the current transform block will be removed. Three cases exist here, they and the needed actions are listed below

1. No points to remove in the current block: update start index if needed and skip to next block
2. All points removed in the current block: Remove the start index and T-parity symbol for the current block, track number of removed points to update all following start indices
3. Some points removed in the current block: Update T-parity and all following start indices, update current start index if needed and track number of removed points.

Case 1 requires only an update of the start index if points have been removed in preceding blocks, in case 2 we need to remove the start index and T-parity symbol corresponding to the current block. In case 3 we may need to change the T-parity if an odd number of points was removed, otherwise it remains the same. If points in preceding blocks were removed, we must shift the start index forward by the number of removed points. If the start index is a removed point, we need to assign the next retained point as the new start index for this block. To update start indices, we must also track to total number of points removed thus far as we progress through the SEI-list. The output of this block is the quantized point cloud without duplicates, and SEI information updated such that it is relevant to the quantized point cloud.

3.6 Entropy Coding

Entropy coding of the geometry is done using MPEGs TMC-13 [19,20]. This is an Octree coding algorithm where the input space is repeatedly separated into octants. Occupancy of each octant is signaled using 1 bit, resulting in 8 bits for the first level, 8x8 bits for the 2nd level and so on, until we reach the leaf nodes. Therefore, concentrating points into fewer high depth octants increases coding efficiency, which motivates the use of grasp style quantization.

SEI is not directly coded, but the number of bits needed are estimated as the Shannon entropy limit.

To find this value for T-parity we simply calculate the binary entropy in the T-parity array and multiply the result by the number of symbols. For the start indices we use a differential coding scheme. We send the first start index directly, and for all subsequent indices we only send the difference to the previous index. Entropy is calculated by finding all unique differences and their respective probabilities. From here we can calculate the entropy per symbol and the total number of bits in the start index SEI as

$$\begin{aligned} &startindex\ bits \\ &\quad = Symbol\ Entropy * Number\ of\ Difference\ Symbols \\ &\quad + \lceil \log_2 total\ number\ of\ points \rceil \end{aligned}$$

$$total\ bits = startindex\ bits + T - parity\ bits + geometry\ bitstream\ bits$$

MPEGs TMC-13 is not order preserving. After encoding-decoding the points may be in a different order. Clearly this is detrimental to the proposed scheme. For testing purposes, we create a bitstream using TMC-13 and use its size for bit per point calculations as we feel this is a

good estimate for achievable base-geometry bitrate. For reconstruction we use a version of the point cloud saved in code just prior to entropy coding. The encoded-decoded cloud and the saved cloud are equivalent except for the order of points. If this approach is to be fully implement an order preserving entropy coder would be required. This may be achieved by differentially coding points in each axis, as the high correlation should lead to good results given a well sorted point cloud. We feel that an investigation into the compression performance should precede work to tailor an entropy coder to this approach, as such no detailed order preserving entropy coding approach is proposed or implemented at this time.

3.7 Decoder Steps

The decoder operates in reverse of the encoder, except we do not need a duplicate point removal or SEI update unit here. We simply decode and dequantize the geometry bitstream then perform the inverse AROS using the SEI from the Encoder. The result is an approximated geometry point cloud. No post processing is proposed at this time, however there may exist beneficial post processing schemes.

3.8 Quality Measurement with PC-Error

To compare the quality of the reconstructed point cloud with the input point cloud we use the PC-error software provided by Nanjing University Vision Lab in their GitHub [8]. This is the same software Grasp-net uses. It takes 2 .ply files as inputs and computes MSE and D1- PSNR

using both files as reference and a final symmetric version. The symmetric version is reported for all wave-grasp and base grasp results. Details on quality measurement are provided in 2.6.

4. RESULTS

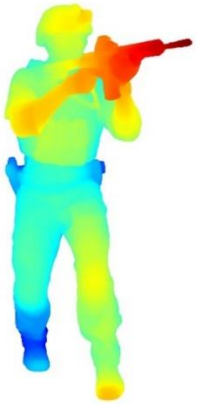
The following section will present the achieved results divided by Point cloud class. As stated above point clouds are divided into Solid, Dense, Sparse and Scant Point clouds, where the number of points per unit volume decrease as we move to sparser classes [12].

Tested here are Solid, Dense and Sparse point clouds. Due to this approach requiring continuous segments we expect the performance to decrease as point clouds get sparser. This suspicion largely holds, but we observe good gains in the low bitrate regime. Scant point clouds are not tested as performance on Sparse Point Clouds suggests that further decreases in density are not worth testing.

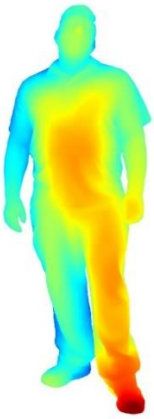
4.1 Dataset Overview

Figure 9 below shows the datasets used to generate the results presented here. We show both a colored version and a version with arbitrary coloring applied. The Color is for illustrative purposes as it is difficult to properly visualize 3D structures with uniform color. However, no color attributes were processed in this approach. All results presented below show our RD performance alongside with using base-grasp quantization alone. Dataset classification follows [12] datasets are obtained from [27-29].

Redandblack_vox10_1460



soldier_vox10_0690



Boxer_viewdep_vox12

Staue-klimt-vox12

Shiva_00035_vox12

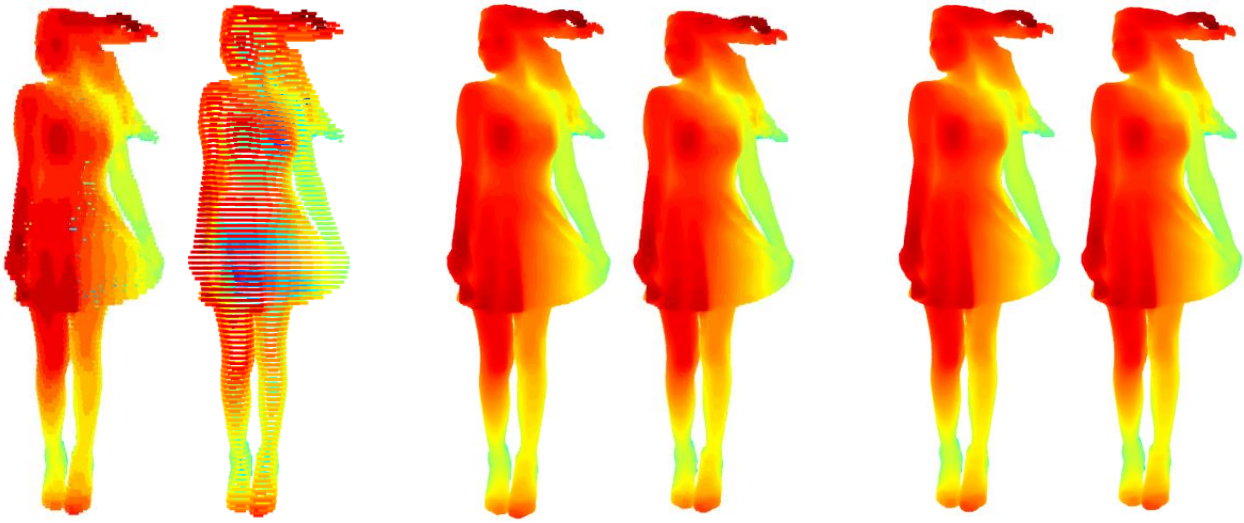
Figure 9 Point Clouds Used for Testing. Presented both in full color and with artificial color scheme. Color is presented only for visualization purposes and was not coded or processed in any way.

4.2 Solid Point Clouds

The following results were produced on Solid Point clouds soldier_vox10 and redandblack_vox10 [28]. The RD curve for Soldier-vox10 is shown in figure 10, figure 11 redandblack-vox10 reconstructed at different scales and figure 12 shows the RD curve for the redandblack-vox10 dataset. For illustrative purposes we show RD curves of the geometry bitstream only, along with those where the SEI is estimated at the Shannon limit and added to the geometry bitstream. No-SEI curves are not decodable and only included to show how the amount of SEI varies as the rate changes, and to give a theoretical upper limit.

Figure 10 RD Curve for 10-bit Soldier Point Cloud. The figure shows the decodable base grasp and Wave-grasp with SEI estimated at the Shannon limit, and the non-decodable reference curve for the wave-grasp geometry component only.

Original geometry



56.8dB wave 52.6dB base 64.2dB wave 61.9dB base 68.1dB wave 66.6dB base

Scaling factor of 0.1 Scaling factor of 0.3 Scaling factor of 0.7

Figure 11 Side by Side Geometry Compression Results for Redandblack-vox10-1460. Wave grasp on left and base grasp on right at each scale factor, for Quantizer scales of 0.1, 0.3 and 0.7 (left to right), with original at the top. Wave Grasp has 56.8dB, 64.2dB and 68.1dB to the Original while base grasp has 52.6dB, 61.9dB and 66.6dB for the scale points respectively. Note that we present results at a fixed quantizer scale to show reconstruction behavior, but the achieved bpp rate at those scales may differ for the two approaches.

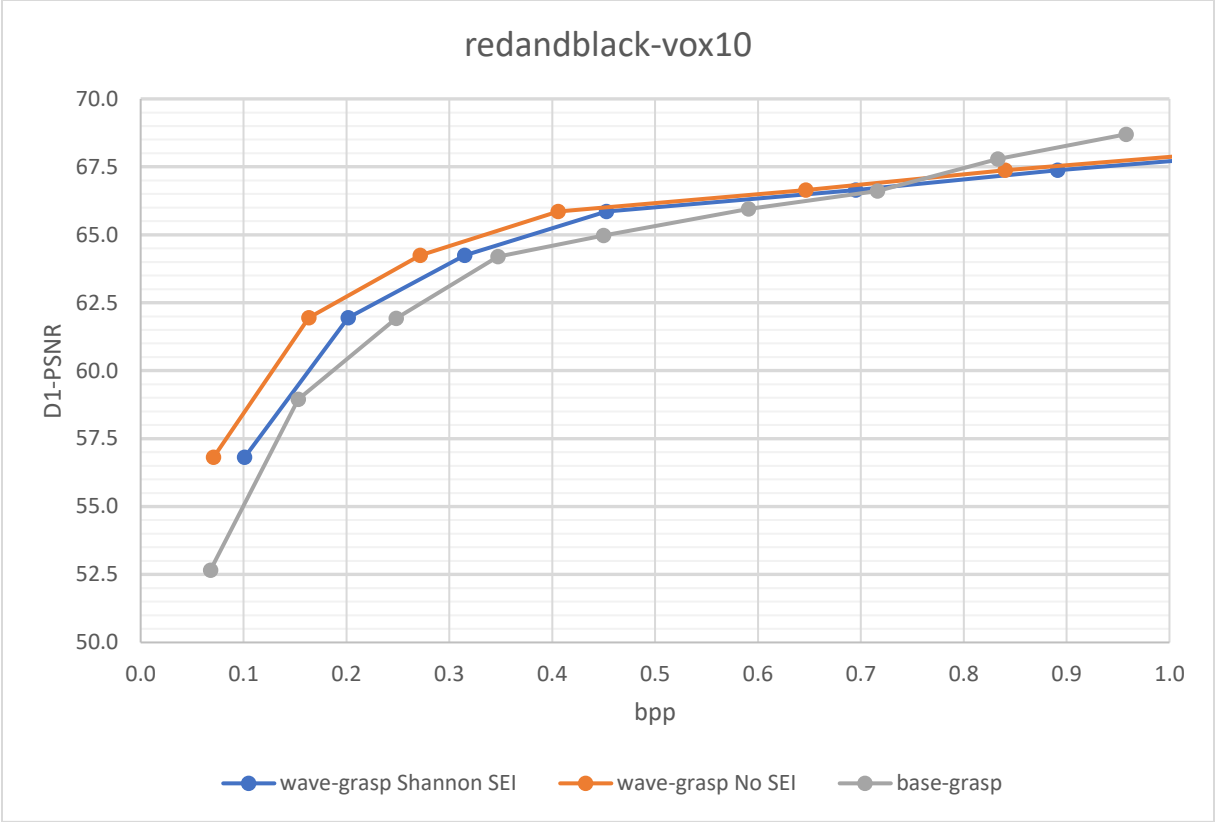


Figure 12 RD Curve for 10-bit Redandblack-vox10-1460. The grey and blue curves are decodable, the orange curve shows wave-grasp with the base geometry component only. This is not decodable but intended to show the trend of SEI as the rate changes.

4.3 Dense Point Clouds

These results were produced on 12-bit dense point clouds soldier-vox12 and boxer-vox12 [27]. Presented in figures 13 and 15 are concave rate points for soldier-vox12 and boxer-vox12 respectively, while figure 14 shows the reconstructed boxer-vox12 dataset at different quantization rates. Plotting all generated datapoints leads to a convex shape, however we can improve coding performance with timesharing between all points that form a concave line. This concave line is presented here. We again show undecodable No-SEI curves and decodable curves

with SEI estimated at the Shannon limit, along with base-grasp quantization. We see that as density decreases the amount of SEI needed increases and stays relatively constant even at higher bit rates. This is due to Dense point clouds having more discontinuities than Solid Point clouds after the input sorting, and discontinuities are inherent to the data and not primarily caused by quantization. We also see an interesting behavior here that Wave-grasp performs worse at low-bitrates but begins to exceed base-grasp at higher bit rates. More experiments are needed to investigate this behavior. However isolated point removal combined with strong quantization may not leave enough data at low bit rates for accurate reconstruction, while at higher bit rates the effects of quantization are not as extreme and enough distinct Low-Pass points are transmitted to make their higher energy density valuable.

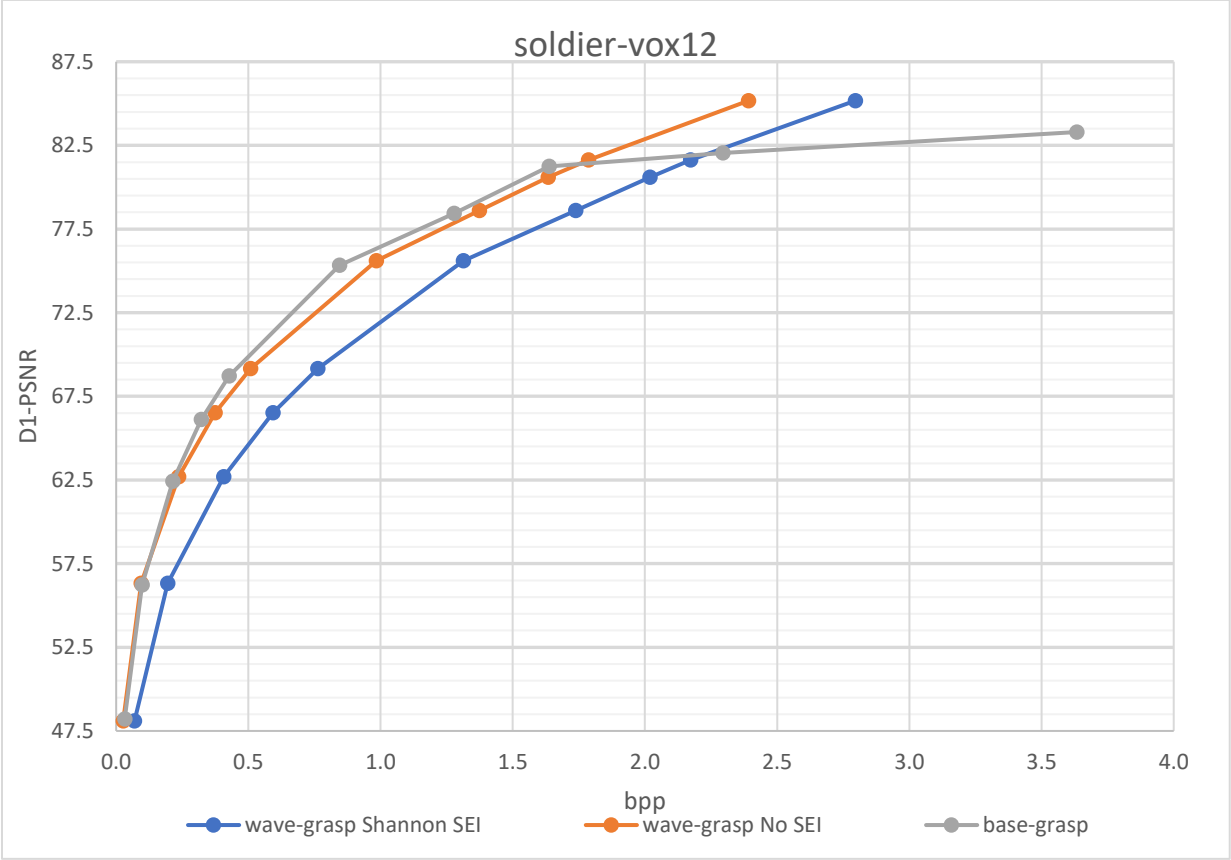


Figure 13 RD Curve for 12-bit Soldier Point Cloud. Grey and blue curves are decodable, the orange curve is provided for reference and shows the wave-grasp base geometry component only, omitting any SEI. Note that the amount of SEI is relatively constant at all displayed rate points.

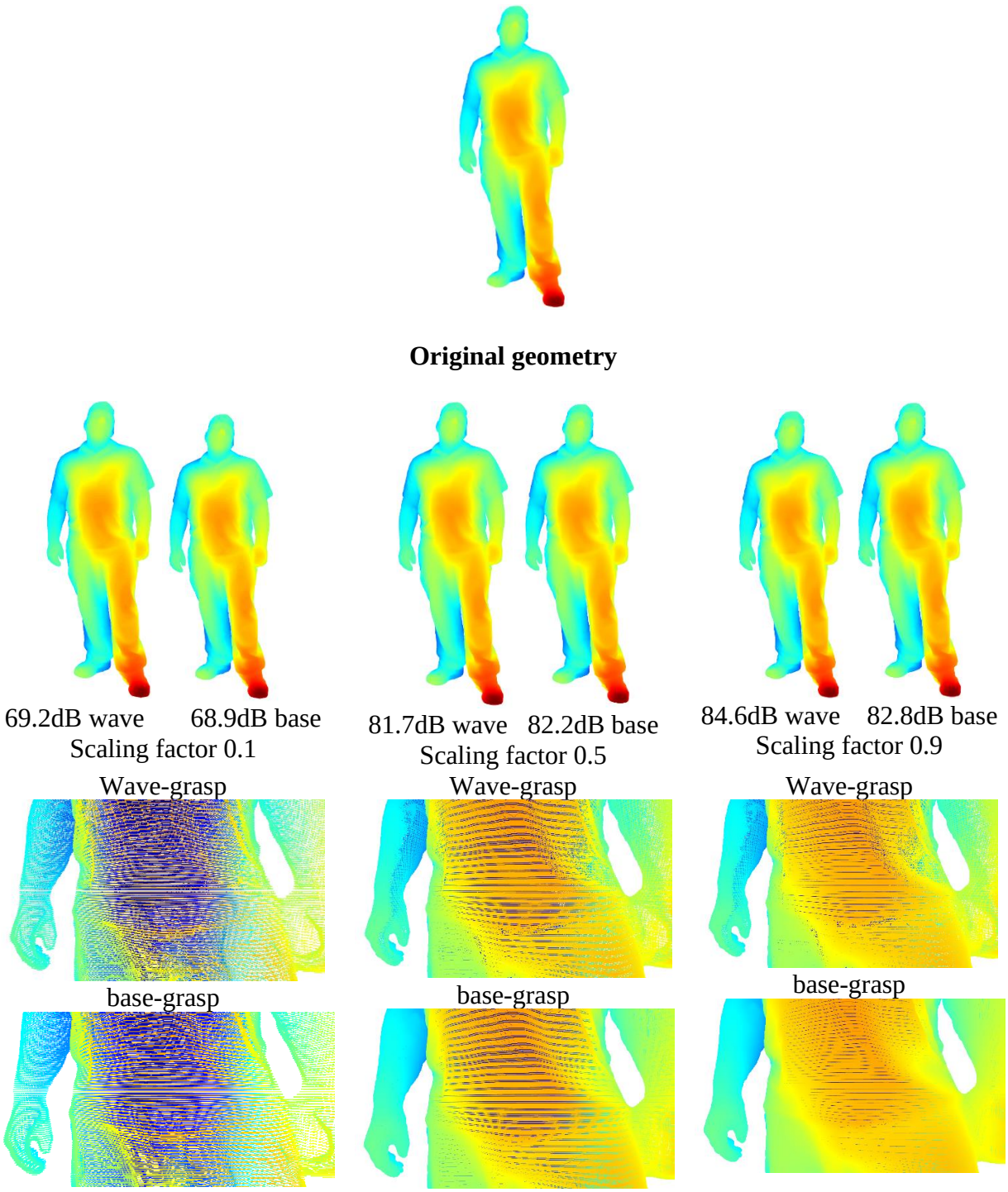


Figure 14 Boxer_viewdep_vox12 Compression Results. The Top shows the original input geometry, the center row shows the full reconstructed point clouds, with Wave Grasp and the left and base grasp on the right for each rate point. The bottom row shows detail, where wave grasp is on top and base grasp on the bottom. Tested quantizer scales are 0.1, 0.5 and 0.9 where wave grasp has a PSNR to the original of 69.2dB, 81.7dB and 84.6dB respectively, while base grasp has PSNRs of 68.9dB, 82.2dB and 82.8dB. Note that we may have different bpp for a given quantizer scale for the 2 approaches.

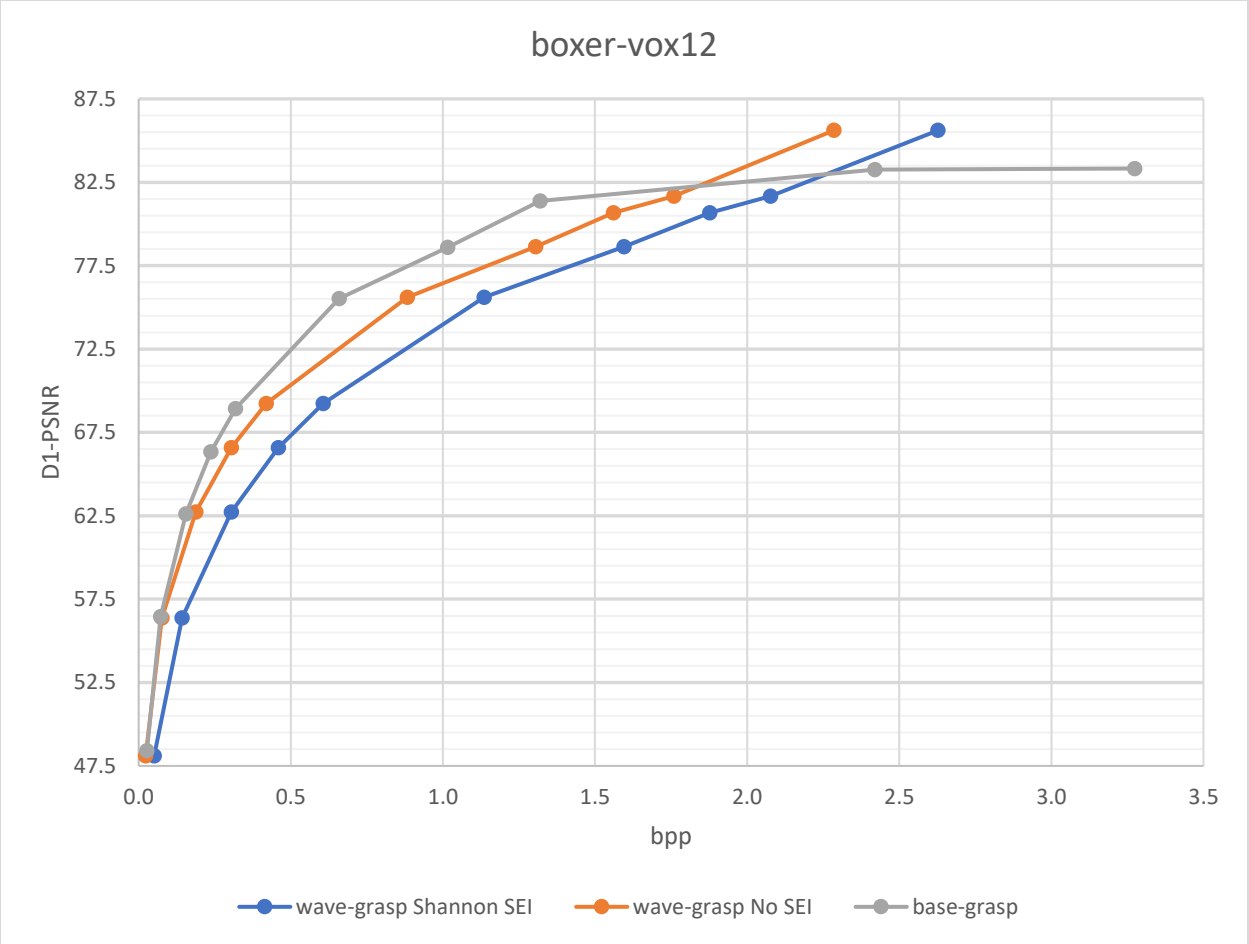


Figure 15 RD Curve for 12-bit Boxer Point Cloud. Grey and blue curves are decodable, the orange curve is provided for reference and shows the wave-grasp base geometry component only, omitting any SEI. Note that the amount of SEI is relatively constant at all displayed rate points.

4.4 Sparse Point Clouds

Results on Sparse point clouds are presented on staue-klimt-vox12 (sic) and Shiva-00035-vox12 [29], shown in figures 16 and 18 respectively. Figure 17 shows the reconstructed Shiva-00035-vox12 dataset at different quantizer rates. Sparse point clouds are very detrimental to the proposed approach due to the high number of isolated points which constitute upward of 80% of

all points. Wave-grasp still outperforms base-grasp at very low bitrates. As in the Dense case more testing is needed to thoroughly explain this behavior, and especially why the trend from Dense point clouds reverses. An initial suspicion is that now the number of isolated points is so large that base-grasp quantization is removing significant detail at low bit rates, and energy concentration of the wavelet-based approach becomes useful again as both base and wave grasp lose significant detail at the lower rates. Here we only present decodable RD curves and elected not to include a No-SEI version. The difference between wave-grasp and base grasp is sufficiently strong here that we feel additional data does not yield more insight only makes the graph harder to read as the difference between No SEI and Shannon SEI is negligible compared to the distance to base-grasp.

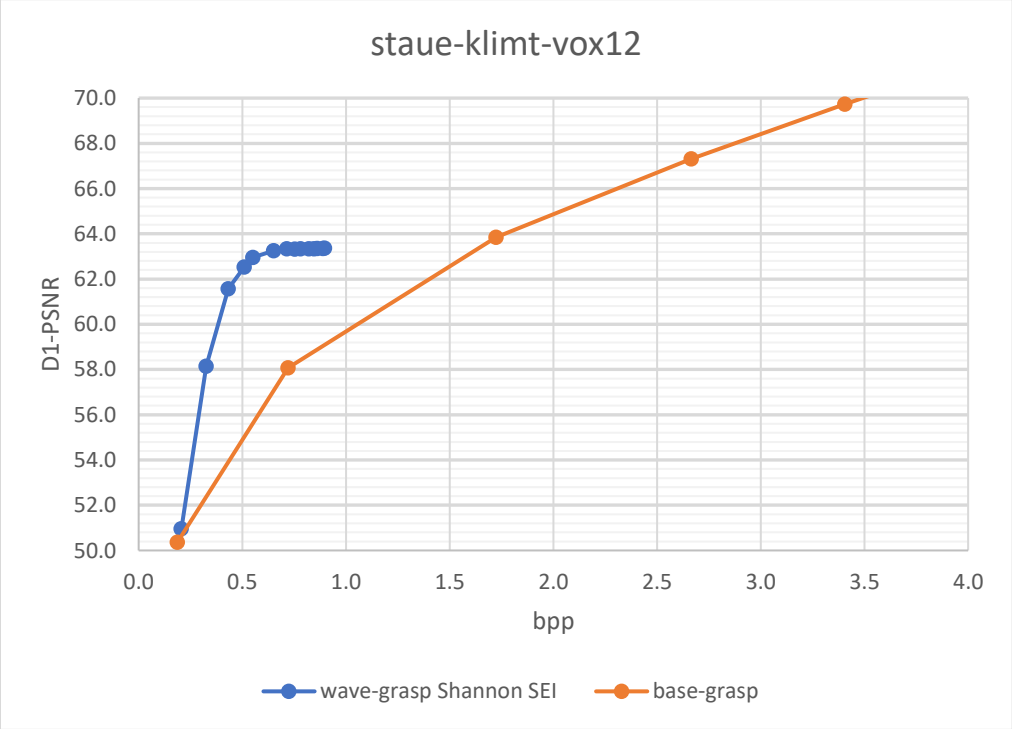


Figure 16 RD Curve for 12-bit Staue-klimt Point Cloud.

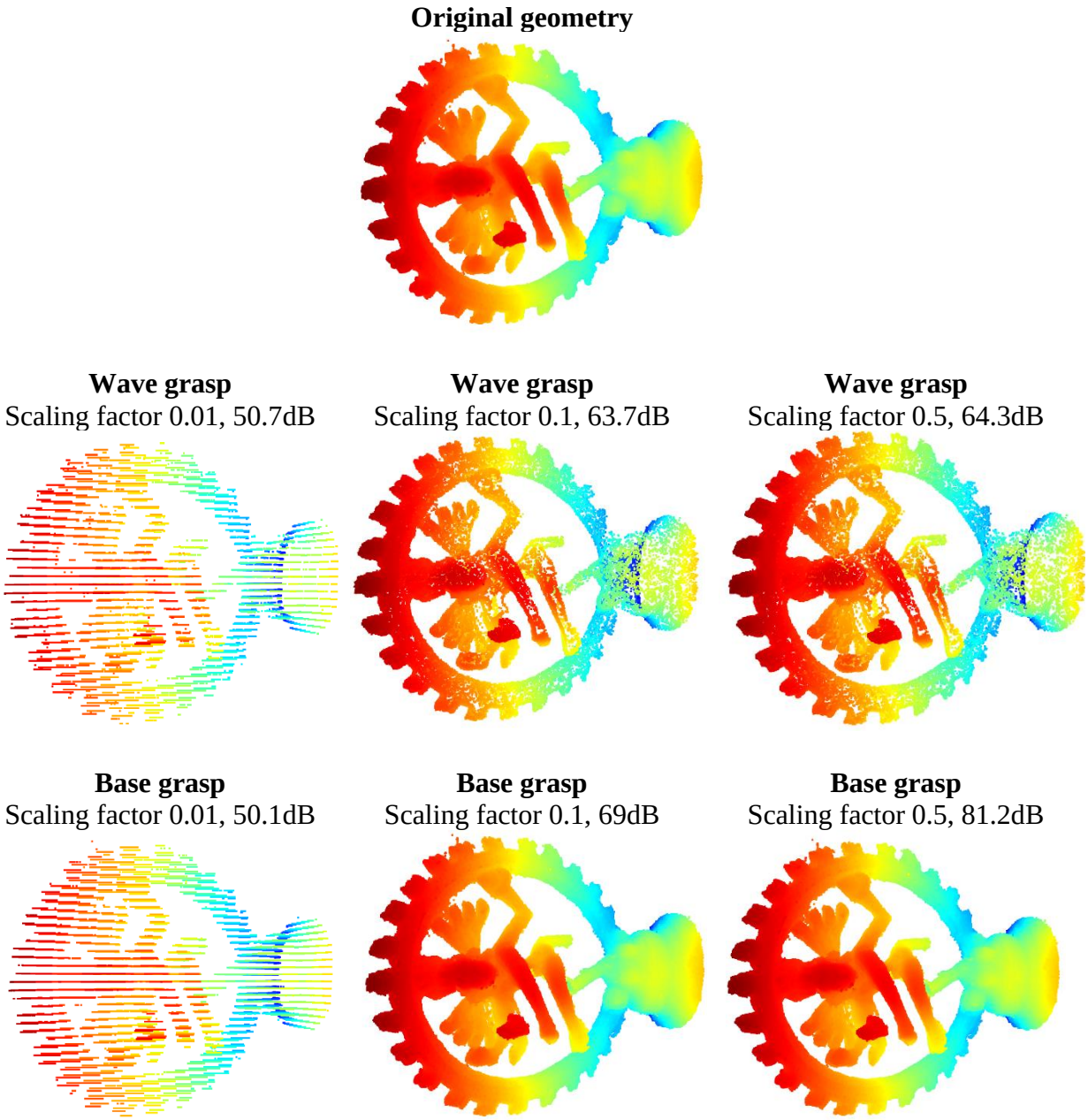


Figure 17 Reconstructed point clouds for Shiva_00035. Original (Top), Wave-grasp (Middle), and Base-Grasp (bottom), Quantizer scale for both Wave and Base grasp were 0.01, 0.1 and 0.5. Where wave grasp has PSNRs of 50.7dB, 63.7dB and 64.3dB respectively while base grasp has 50.1dB, 69dB and 81.2dB. Note that to illustrate the reconstruction behavior we present images at the same quantizer scale point, the bpp for each approach at a given quantizer scale can vary widely. One can clearly see Wave-grasp suffering from isolated point removal, especially at higher rates.

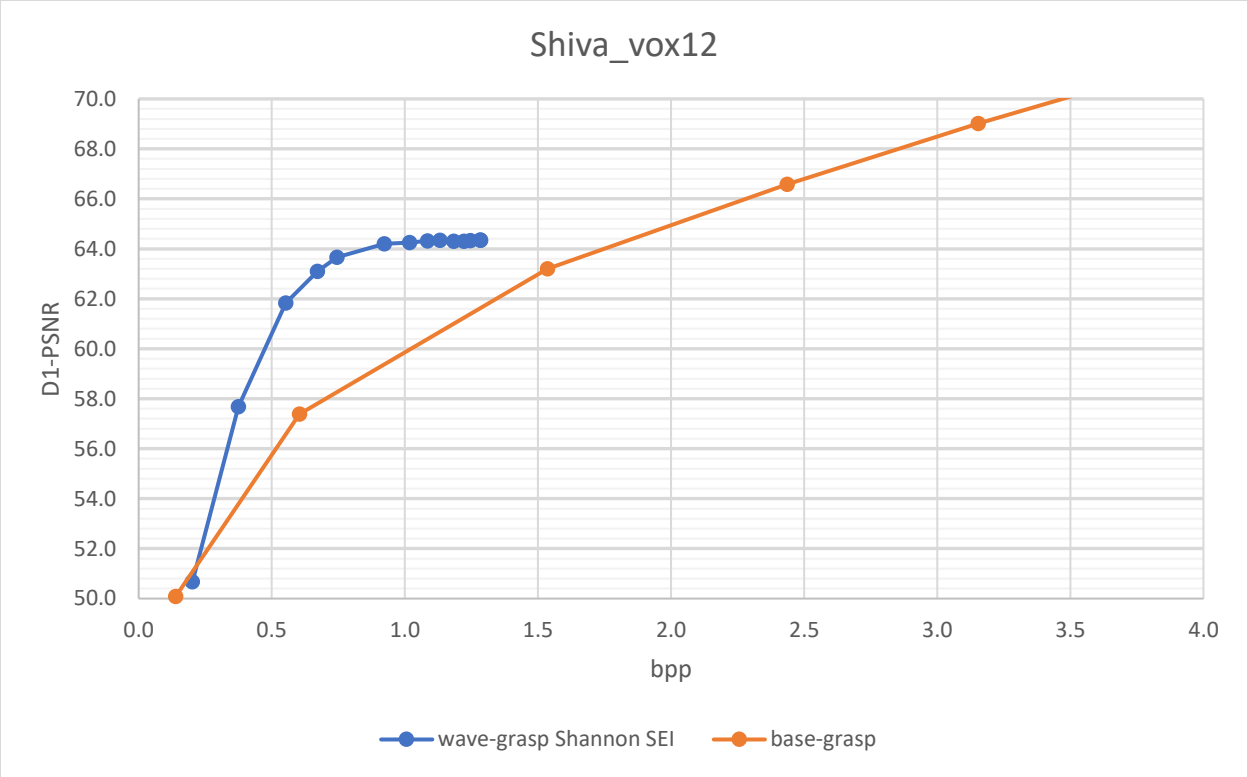


Figure 18 RD Curve for 12-bit Shiva_00035 Point Cloud.

4.5 Number of Isolated Points Removed

Presented in table 4 below are the number of input points, non-isolated points and percentage of points removed in wave-grasp pre-processing. We see that as density decreases number of removed points increases drastically, until the majority of points are removed as isolated for sparse clouds. Section 3 shows that sending isolated points does not improve RD performance, as any gain in quality is absorbed by the increases in bpp.

Table 4 Number and Percentage of Points Removed as Isolated.

Dataset name	Number of points in input cloud	Number of not isolated points	Number of removed isolated points	Percentage of points removed
Solider-vox10-0690	1089091	1085914	3177	0.29%
Redandbalck-vox10-1460	710964	708192	2772	0.39%
Solider-vox12-viewdep	4001754	2769601	1232153	30.8%
Boxer-vox12-viewdep	3493085	2690014	803071	23.0%
Staue-klimt-vox12	499660	55856	443804	88.8%
Shiva-00035-vox12	1009132	184959	824173	81.7%

5. DISCUSSION

5.1 Results Discussion

We immediately notice that our approach is best suited to denser point clouds. This is intuitive due to our need for continuous segments. Our performance gains at low bitrates are believed to be due to energy concentration, when sending few points wave-grasp excels as its points contain more energy. At higher bitrates energy concentration becomes less useful compared to sending the non-transformed points as we simply have enough points available such that missing points are of reduced importance.

Interesting is also that wave grasp performs better at high bit rates for Dense point clouds but favors low bit rates for all other point cloud classes. The immediate cause for this phenomenon is not clear. A first suspicion is that Dense point clouds form an intermediate regime where simple grasp-style quantization still produces good results at lower bit rates, but the number of isolated points is large enough to prevent wave grasp from performing well, while at higher rates and lower quantization scales, enough unique points are produced to allow energy concentration in wave grasp to be effective. Points in Solid point clouds are close enough together that they are shifted more as a unit at all scales, allowing base-grasp to exceed energy concentration benefits at high rates. Finally for Sparse point clouds the vast majority of points are isolated and removed in wave-grasp, in order to achieve any gain over base-grasp we need to test at extremely low bitrates, such that the number of removed points in base-grasp is large enough for energy

concentration to offset the effects of isolated point removal in wave grasp. These suspicions may be confirmed by modifying the approach to reduce the number of isolated points.

5.2 Possible Improvements and Future Work

As can be seen in the results section, the largest detriment of wave grasp is it's need for (long) continuous sections. The fewer continuous sections the better the entropy performance and the less SEI we incur. Additionally sparser point clouds lead to more isolated points, which can only be coded at high SEI costs for even indices, and not at all for odd indices. This suggests an immediate area of improvement: Sorting. A more sophisticated sorting algorithm that finds an optimal solution in terms of either minimum number of isolated points, minimum number of continuous sections, or a combination thereof may lead to significant improvements. Hand in Hand with this are possibly different definitions of continuity, some of which were investigated in section 3. Coupled with more sophisticated sorting this should be a primary area of investigation for future work. Additionally, once an optimal continuity condition is found, renewed attempts can be made to infer start indices and T-parity from the bitstream itself and not explicitly signal them.

5.3 Conclusion

We have shown a wavelet-based method for geometry point cloud compression. Our proposed approach shows gains in the low bitrate regime but exhibits poor performance on low density

point clouds. Nonetheless Low bitrate applications may benefit from this approach and future work may yield modifications more suitable for low density point clouds.

REFERENCES

[1] Geo-Plus inc., "A short story about LIDAR technology - geo-plus: LIDAR, Land Surveying and Civil Engineering Software Solutions," Geo-Plus, Jun. 1, 2022. [Online]. Available: https://geo-plus.com/blog-a-short-story-of-lidar-technology/. [Accessed: Mar. 6, 2023].

[2] W. M. Kaula, et al., "Analysis and interpretation of lunar laser altimetry," Proceedings of the Lunar Science Conference, vol. 3, p. 2189, 1972.

[3] D. E. Smith, et al., "Topography of the Moon from the Clementine lidar," Journal of Geophysical Research: Planets, vol. 102, no. E1, pp. 1591-1611, 1997.

[4] R. Hack, A. Turner, and S. Slob, "An approach to automate discontinuity measurements of rock faces using laser scanning techniques," in EUROCK2002, Runchal, Maderia Island, Portugal, 1980, pp. 87-94.

[5] H. Scharr, et al., "Fast Visual Hull Extraction at High Resolution," Computer Vision, Graphics, and Image Processing, vol. 40, no. 1, pp. 1-29, 1987.

[6] H. Hoppe, et al., "Projector based intraoperative visualization of surgical planning data," Proceedings of ISRACAS, 2001.

[7] "Introduction to the MPEG-PCC project," MPEG Point Cloud Compression. [Online]. Available: https://mpeg-pcc.org/. [Accessed: Jan. 18, 2023].

[8] Njuvision, "NJUVISION/pcgcv2: Multiscale point cloud geometry compression," GitHub. [Online]. Available: https://github.com/NJUVISION/PCGCv2. [Accessed: Jan. 18, 2023].

[9] J. Pang, M. A. Lodhi, and D. Tian, "GRASP-Net: Geometric Residual Analysis and Synthesis for Point Cloud Compression," in ACM MM Workshop on APCCPA, 2022.

[10] "ISO/IEC 11172-2:1993," International standards organization. [Online]. Available: https://www.iso.org/standard/22411.html.

[11] "Information technology — Coded representation of immersive media — Part 3: Versatile video coding," ISO/IEC JTC 1/SC29/WG11, N18692, 2019. [Online]. Available: https://mpeg.chiariglione.org/standards/mpeg-i/versatile-video-coding/text-isoiec-cd-23090-3-versatile-video-coding.

[12] A. Zaghetto, D. Graziosi, and A. Tabatabai, "Dataset Split for AI-PCC," ISO/IEC JTC 1/SC 29/WG 7, m58754, Virtual, Jan. 2022.

[13] Elkhrachy, "3D structure from 2D dimensional images using structure from motion algorithms," MDPI, Apr. 30, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/9/5399. [Accessed: Jan. 18, 2023].

[14] Q. Wang and M.-K. Kim, "Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018," Sciencedirect, 15-Feb-2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034618304683?casa_token=f1NipUOXxUwAAAAA%3A-aa71Qyx6IIXsMSLK8eTe8Dq8OMOnFcJa30KSCnp61XyfI51UBcqhhh-6JNDN0BLRkMpFvPoegY. [Accessed: 18-Jan-2023].

[15] S. Chickerur and K. Joshi, "3D face model dataset: Automatic detection of facial expressions and emotions for educational environments," British Educational Research Association, 13-Aug-2015. [Online]. Available: https://bera-journals.onlinelibrary.wiley.com/. [Accessed: 18-Jan-2023].

[16] S. Chen, B. Liu, C. Feng, C. Vallespi-Gonzalez, and C. Wellington, "3D Point Cloud Processing and Learning for Autonomous Driving: Impacting Map Creation, Localization, and Perception," IEEExplore, 01-Jan-2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9307334/references#references. [Accessed: 18-Jan-2023].

[17] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC): APSIPA Transactions on Signal and Information Processing," Cambridge Core, 03-Apr-2020. [Online]. Available: https://www.cambridge.org/core/journals/apsipa-transactions-on-signal-and-information-processing/article/an-overview-of-ongoing-point-cloud-compression-standardization-activities-videobased-vpcc-and-geometrybased-gpcc/56FCAF660DD44348BCB1BCA9B5EC56CF. [Accessed: 18-Jan-2023].

[18] J. Wang, Z. Ma, H. Wei, Y. Yu, V. Zakharchenko, and D. Wang, "Point Cloud Geometry Compression using Sparse Tensor-based Multiscale Representation," ISO/IEC JTC1/SC29/WG7, m59035, Virtual, 1/2022.

[19] MPEG, "G-PCC codec description," ISO/IEC JTC1/SC29/WG7, output document N0151, Virtual, 7/2021, Sep. 24, 2021.

[20] MPEGGroup, "MPEGGroup/MPEG-PCC-TMC13: Geometry Based Point Cloud Compression (G-PCC) test model," GitHub. [Online]. Available: https://github.com/MPEGGroup/mpeg-pcc-tmc13. [Accessed: 18-Jan-2023].

[21] J. Pang, M. A. Lodhi, and D. Tian, "Geometric Residual Analysis and Synthesis for PCC," ISO/IEC JTC 1/SC 29/WG 7m59649, Apr-2022.

[22] M. Vetterli and J. Kovačević, Wavelets and Subband Coding, Prentice Hall, Englewood Cliffs, NJ, 1995.

[23] D. Le Gall and A. J. Tabatabai, "Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques," in Proc. ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing, 1988, vol. 2, pp. 761-764.

[24] G. Minami, Z. Xiong, A. Wang, and S. Mehrotra, " 3-D Wavelet Coding of Video With Arbitrary Regions of Support" IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no. 9, pp. 1063-1068, Sep. 2001.

[25] L. Cruz, "JPEG Pleno Point Cloud Coding Common Training and Test Conditions v1.3," ISO/IEC JTC1/SC29/WG1. N100276, 2022.

[26] A. Zaghetto, D. Graziosi, and A. Tabatabai, "AI-PCC Evaluation Framework," ISO/IEC JTC1/SC29/WG7 m58753, 2022.

[27] M. Krivokuća, P. A. Chou, and P. Savill, "8i Voxelized Surface Light Field (8iVSLF) Dataset," ISO/IEC JTC1/SC29 WG11 (MPEG) input document m42914, Ljubljana, July 2018, Boxer-viewdep-vox12, soldier-viewdep-vox12.

[28] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva, January 2017, Longdress_vox10, soldier_vox10, redandblack_vox10.

[29] C. Tulvan, A. Gabrielli, and M. Preda, "Datasets update on Point Cloud compression for cultural objects," ISO/IEC JTC1/SC29/WG11 (MPEG/JPEG) input document m38678, Geneva, 2016, Shiva and Staue Klimt datasets.

[30] E. Alexiou and T. Ebrahimi, "Towards a Point Cloud Structural Similarity Metric," Multimedia Signal Processing Group.