Graph-based Point Cloud Compression

Paulo Jorge Robert de Oliveira Rente

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Fernando Manuel Bernardo Pereira
Prof. João Miguel Duarte Ascenso
Prof. Catarina Isabel Carvalheiro Brites

Examination Committee:

Chairperson: Prof. José Eduardo Charters Ribeiro da Cunha Sanguino
Supervisor: Prof. Fernando Manuel Bernardo Pereira
Member of the Committee: Prof. Sérgio Manuel Maciel Faria

November, 2017

Acknowledgments

First of all, I would like to thank the closest persons in my life: my family and girlfriend; for their endless support, patient and motivation I am eternally grateful. Without them, it would be impossible to achieve this academic degree.

I would also like to show my deepest gratitude to my supervisors: Professor Fernando Pereira; Professor João Ascenso; and Professor Catarina Brites. Their orientation, availability and support throughout this process were a key component for its success.

Last but not least, I would like to thank all my friends.

Resumo

Os modelos tri-dimensionais (3D) de representação visual, como *light fields* e *point clouds*, estão a tornar-se cada vez mais populares devido à sua capacidade para representar o mundo real de uma forma mais completa e imersiva. Neste sentido, os modelos 3D abrem caminho para cenários aplicacionais novos e mais avançados. O modelo *point cloud* permite representar eficazmente a superfície de objetos/cenas através de pontos 3D com atributos associados. Tendo em conta as suas características, é já bastante utilizado em diferentes contextos, desde veículos autónomos a realidade aumentada. Recentemente, novos sensores de imagem vieram facilitar a aquisição de *point clouds* mais ricas e densas, constituídas por milhões de pontos. A enorme quantidade de dados associada a este tipo de informação torna o seu armazenamento e transmissão particularmente difíceis. Assim sendo, é necessário desenvolver soluções de codificação eficientes, que possibilitem experiências de multimédia imersivas e ofereçam uma maior qualidade de experiência ao utilizador.

O objetivo desta Dissertação é o desenho, implementação e avaliação de uma solução de compressão de informação geométrica de *point clouds* estáticas. Deste modo, este trabalho começa por introduzir alguns conceitos base, aplicações e estruturas gerais acerca de *point clouds* e revê soluções de codificação para *point clouds* disponíveis na literatura. A solução proposta usa um método baseado em *octrees* na camada base e a transformada de grafos na camada superior, onde um resíduo entre camadas é codificado. A análise de resultados mostra um aumento de desempenho comparativamente com o estado da arte, especialmente para baixos e médios débitos.

Palavras Chave: aquisição de point clouds, compressão de point clouds, rendering de point clouds, análise de point clouds, transformada de grafos.

Abstract

Recently, three-dimensional (3D) visual representation models such as light fields and point clouds are becoming popular due to their capability to represent the real world in a more complete and immersive way, paving the road for new and more advanced application scenarios. The point cloud representation model is able to efficiently represent the surface of objects/scenes by means of 3D points and associated attributes and is already widely used, from autonomous cars to augmented reality. Emerging advanced imaging sensors have made easier to perform richer and denser point cloud acquisition, notably with millions of points. However, the massive amount of data associated with this type of information makes its storage and transmission rather difficult and in some cases even impossible. Hence, there is a strong demand to develop efficient point cloud coding solutions which allow more powerful multimedia experiences and offer better quality of experience (QoE) to the user.

The objective of this Thesis is the design, implementation and assessment of a solution that compresses the geometry information of static point clouds. Considering this context, this Thesis introduces some basic concepts, applications and overall frameworks about the point cloud representation model and reviews relevant point cloud coding solutions available in the literature. The proposed coding solution uses an octree-based approach for a base layer and a graph-based transform approach for an enhancement layer where an Inter-layer residual is coded. The performance assessment shows very significant compression gains regarding the state-of-the-art, especially for the most relevant lower and medium rates.

Keywords: point cloud acquisition, point cloud compression, point cloud rendering, point cloud assessment, graph transform.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3D	Three Dimensional
QoE	Quality of Experience
PCL	Point Cloud Library
7D	Seven Dimensional
2D	Two Dimensional
DoF	Degrees of Freedom
6DoF	6 Degrees of Freedom
VAC	Vergence-accommodation Conflict
4D	Four Dimensional
RGB	Red Green Blue
LoD	Level of Detail
VR	Virtual Reality
AR	Augmented Reality
HMD	Head-mounted Displays
JND	Just-noticeable Distortion
HVS	Human Visual System
EPI	Epipolar-plane Images
IR	Infrared
LIDAR	Light Detection and Ranging
NIR	Near Infrared
DOE	Diffractive Optical Element
CPU	Central Processing Unit
RAM	Random Access Memory
LPC	Layered Point Clouds
GPU	Graphical Processing Unit
MPEG	Moving Picture Experts Group
CfP	Call for Proposals
PCC	Point Cloud Compression
C2C	Cloud to Cloud
C2S	Cloud to Surface
C2P	Cloud to Plane
sRMS	Symmetric Root Mean Square

sHausd	Symmetric Hausdorff
gPSNR	Geometry Peak Signal to Noise Ratio
PSNR	Peak Signal to Noise Ratio
cPSNR	Color Peak Signal to Noise
KLT	Kahunen-Loève Transform
JPEG	Joint Photographic Experts Group
DPCM	Differential Pulse-code Modulation
bpv	bits per vertex
SNR	Signal to Noise Ratio
ASP	Average Signal Power
RD	Rate-distortion
DCT	Discrete Cosine Transform
K-NN	K-Nearest Neighbors
SA-DCT	Shape-adaptive DCT
AVC	Advanced Video Coding
HEVC	High Efficiency Video Coding
PCA	Principal Component Analysis
LZMA	Lempel-Ziv Markov Algorithm
SCSM	Sparse Coded Surface Models
HSCSM	Hierarchical Sparse Coded Surface Models
GTGC	Graph Transform Geometry Coding
BL	Base Layer
EL	Enhancement Layer
U-BL	Upsampled Base Layer
RQCF	Rate-quality Control Factor
MLS	Moving Least Squares
QS	Quantization Step Size
bpp	bits per point
U-GTGC	Unrealistic Graph Transform Geometry Coding
BD	Bjøntegaard-delta

Chapter 1

1. Introduction

The goal of this chapter is to introduce the subject of this Thesis: point cloud coding. First, the context and motivation behind this topic are outlined, followed by the presentation of the Thesis objectives; finally, the structure of this document is detailed.

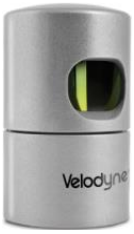
1.1. Context and Motivation

Multimedia plays a major role in our daily lives as its overwhelming presence vastly influences many important application areas, such as entertainment, sports, education, communication, art and medicine. Naturally, as our society endlessly targets improvement, there is a continuous effort to enhance the way multimedia content is experienced by the end users. In this context, recent technological progresses have allowed multimedia applications and services to improve the way visual information is acquired, compressed and displayed. Moreover, these technologies are the key enablers for new and more advanced application scenarios that shall enhance our society's lifestyle and working models.

It must be clear that the main subject of these technologies is visual information which means light which is mainly characterized by its position and direction and its intensity. One of the most popular ways to formalize the representation of light is the so-called plenoptic function which is a powerful seven-dimensional (7D) model. This model defines the intensity of light seen at any viewpoint, (x,y,z), from any angular viewing direction (θ,Φ), for every wavelength (λ) and over time (t) [1]. Therefore, all the information available to an observer from any viewpoint, in every direction, for any wavelength, in any moment in time, can be represented using this model.

The recent breakthroughs in terms of imaging sensors have been significantly changing the amount and type of visual information that may be acquired. For example Figure 1 (a) shows an example of a so-called light field camera, in this case the *Lytro Illum* [2] camera, and Figure 1 (b) shows an example of a depth sensor, in this case the *Velodyne's HDL-32E* [3]. These new sensors are allowing to perform richer plenoptic acquisitions both in terms of light intensity for multiple directions at the same position in space as well as in terms of three-dimensional (3D) geometry.

(a)

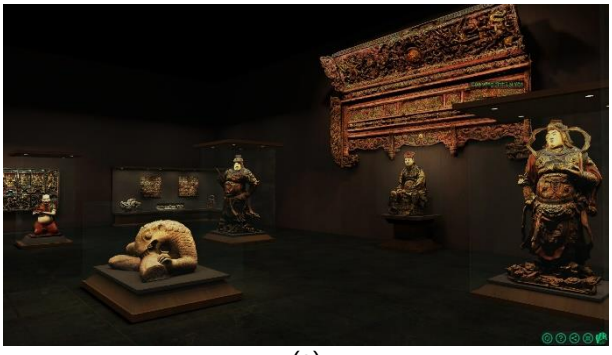


(b)

Figure 1: (a) *Lytro Illum* camera *[2]*; (b) *Velodyne's HDL-32E* depth sensor *[3]*.

In practice, the new imaging sensors allow a richer visual scene representation by acquiring the light intensity for each spatial position as a function of the incident direction very much like our eyes; moreover, the depth sensors allow obtaining depth information, i.e. the distance between the objects/scene and the sensor, which better characterizes the scene geometry. While the depth information may be directly acquired, it may also be estimated from light intensity, e.g. using multiple viewing perspectives and checking their disparity. In summary, these new sensors enable a better and more realistic representation of the visual scenes which should be stored and transmitted. On the other hand, they generate massive amounts of visual data, making the development of efficient coding solutions mandatory for scenarios such as online streaming and storage.

As it is critical to represent the visual information acquired by these sensors in the most faithful and realistic way, 3D visual representations have become increasingly more attractive as they should allow the full representation of the visual information by acknowledging what is obvious, we live in a 3D world. The 3D representation formats should be able to provide richer experiences to the end user, thus bringing benefits in terms of quality of experience (QoE) and should enable several application scenarios that require additional functionalities not offered by the traditional two-dimensional (2D) representation formats, e.g. free viewpoint viewing. Some examples of these applications are 3D free viewpoint broadcasting, cultural heritage, see Figure 2 (a), and real-time 3D immersive telepresence, see Figure 2 (b).



(a)

(b)

Figure 2: (a) 3D VR video frame of a museum *[4]*; (b) Real-time immersive 3D telepresence used for a presidential campaign *[5]*.

At this stage, the topic of this Thesis shall be introduced more directly: point clouds. A point cloud is a type of 3D representation consisting on a set of 3D positions, with a certain density, defined by its coordinates, and some corresponding attributes. The coordinates indicate the 3D spatial position (x,y,z) associated to objects in the scene whereas the attributes may consist of colors, normal vectors and other relevant features. As the 3D spatial positions of the points basically define the 3D geometry of the scene, the (acquired or estimated) depth information plays a major role in the process of creating a point cloud. It is also important to state that, under optimal conditions, a point cloud representation may include all the relevant visual information about the original or synthetic scene. In the past years this 3D representation format has gained much relevance and is already widely used in several fields such as medical imaging, computer graphics, 3D printing, robotics, architecture and manufacturing. Figure 3 (a) shows an example of point cloud acquisition used for manufacturing and Figure 3 (b) shows the Shipping Galleries point cloud before their decommission in 2012.

(a) (b)

Figure 3: (a) Example of point cloud acquisition for manufacturing *[6]*; (b) Point cloud of the Shipping Galleries at the Science Museum, London *[7]*.

There are some application scenarios where this particular form of visual representation format may bring QoE advantages when compared to other 3D representation formats, e.g. light fields which basically consist of multiple 2D acquisitions of the same scene from multiple perspectives.

Knowing that there are already numerous applications for point clouds and that this 3D representation format is associated with large amounts of information, there is an increasing necessity for efficient storage and streaming. Hence, suitable coding solutions, based on the target application characteristics and requirements, shall be designed, implemented and assessed. The main objective to be reached with this technology is to facilitate the emergence of more thrilling, realistic and interesting applications, by reducing the amount of information stored or streamed. Furthermore, it is important to state that point cloud coding involves both coding the geometric information, this means the 3D positions and the attributes; moreover, the point clouds may be static, i.e. a single frame for a single time instant, or dynamic, i.e. a sequence of frames for a lapse of time.

1.2. Objectives

In brief, the objective of this Thesis is the development of a lossy coding solution for the geometry information of static point clouds. More precisely, the target of this work is the design, implementation and assessment of a novel point cloud lossy coding solution, for the geometry of static point clouds, based on graph transforms which is an emerging tool in multimedia representation and processing. In order to achieve these objectives, the work developed was organized in the following steps:

1. **Revision** of the most relevant point cloud coding solutions in the literature targeting static point cloud coding; the coding solutions reviewed correspond to the state-of-the-art in this field of study that should be considered when developing more advanced solutions.
2. **Design and implementation** of a lossy point cloud geometric data coding solution, for static point clouds, using the recently popular graph-based transform.
3. **Evaluation** of the proposed point cloud coding solution, by presenting sustained justifications for some of the design choices adopted along the Thesis, and **benchmarking of** the novel coding solution with relevant state-of-the-art coding solutions.

1.3. Thesis Structure

To achieve the proposed objectives presented in the previous section, this Thesis is structured as follows:

- **Chapter 2:** In this chapter, the state-of-the-art on 3D visual representation is presented and the most relevant point cloud coding solutions are extensively reviewed.
- **Chapter 3:** In this chapter, the proposed coding solution architecture and main tools are presented.
- **Chapter 4:** In this chapter, the proposed coding solution performance is assessed and a benchmarking study is performed with state-of-the-art coding solutions.
- **Chapter 5:** In this chapter, the final conclusions are outlined and some future work regarding the topic of this Thesis is suggested.

Chapter 2

2. State-of-the-Art on 3D Visual Representation and Main Point Cloud Coding Solutions

This chapter intends to present the main concepts and tools involved in 3D visual representation as well as the most relevant point cloud data coding solutions in the literature. With that purpose in mind, this chapter starts by addressing the basic concepts on 3D representation, followed by the applications and requirements of 3D models. Afterwards, the focus will lie on the Thesis topic: point clouds; first, the point cloud system architecture and walkthrough is reviewed, the acquisition systems are introduced, as well as the rendering methods and the objective quality metrics. Finally, the main coding solutions of point cloud data present in the literature are introduced and reviewed.

2.1. Basic Concepts on 3D Representation

With the increasing popularity of 3D technologies, more realistic and powerful representations of the world visual information are emerging. A 3D visual representation allows to reproduce real and virtual scenes in such a way that the user can experience them with more Degrees of Freedom (DoF) in terms of his/her head and body movements. In this context, the DoF refer to the freedom of movement of a rigid body in space. In particular, the body is free to translate along three perpendicular axes, e.g. forward/backward, up/down, left/right, and also rotate around three axes, e.g. pitch, yaw and roll. All these movements result into 6 Degrees of Freedom (6DoF) [8], which when replicated with visual information provides the users highly immersive visual experiences, see Figure 4.

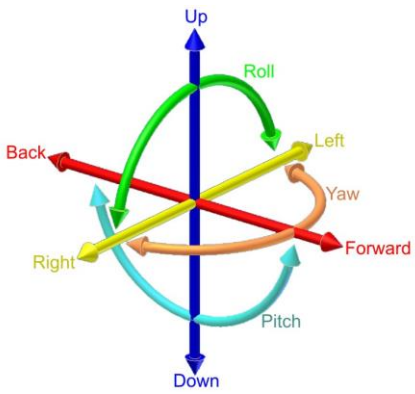


Figure 4: The six degrees of freedom – forward/back, left/right, up/down, yaw, pitch and roll *[8]*.

In contrast to traditional 2D multimedia, where the scene content is projected into a regular 2D grid of samples, 3D representations should replicate the scene as much as one can experience the real world, e.g. with horizontal and vertical (full) parallaxes, ability to refocus, etc.

There are two main approaches to represent 3D scenes: geometry-based and image-based. 3D meshes and point clouds are geometry-based solutions as they explicitly represent the geometry of the scene whereas light fields are image-based solutions as they just use 2D texture projections. At the end, all the approaches have advantages and drawbacks, notably depending on the application scenario targeted. Therefore, it is important to briefly review the main available 3D representation models and describe their characteristics to be able later to select the best one for a certain application and associated requirements.

The conceptual representation of the real world light can be achieved by the so-called plenoptic function, which expresses the intensity of light seen from any viewpoint, this means 3D spatial position (x,y,z), any angular viewing direction (θ,Φ), over time (t), and for each wavelength (λ) [1]. This powerful 7D model represents all the information available to an observer from any viewpoint, in every direction, over time. Every representation format, for image and video, 2D or 3D, can be directly or indirectly expressed in the framework provided by this function, using all or only some of its seven dimensions. With the emergence of new sensors such as the *Lytro* [2] *and Light* [9] cameras and the *Velodyne LIDAR* 3D scanner [10], richer plenoptic imaging acquisitions and representations are becoming more relevant. These types of sensors go beyond the classic 2D representation model to provide the users with more complete and immersive visual experiences. However, these new imaging sensors typically generate a massive amount of visual data and, thus, it is critical to develop appropriate representation formats and efficient coding solutions for scenarios such as online streaming and storage. Depending on the application characteristics and constraints, from acquisition to display, several 3D representation formats may be relevant:

1. **Light Field** – This is a 3D representation format where the scene light radiance is measured by projecting it into a high number and high density set of 2D sensors, generating many, so-called *views*. This model may involve both horizontal and vertical parallaxes and, ideally, a high angular resolution between the views. If the captured light density is high enough, the usual vergence-accommodation conflict (VAC) should be negligible, allowing to give the user the impression of experiencing a real light field. When there are many 2D sensors capturing the same scene, it is possible to measure the light coming from the same scene positions in different angular directions as expressed by the plenoptic function. This results into a 3D representation format where smooth motion parallax is obtained. In terms of light field data acquisition, there are two main approaches: using a (single) lenslet camera with many microlens such as Lytro [2] or an array of conventional cameras (in theory, it is also possible to use an array of lenslet cameras). The lenslet acquisition produces a four-dimensional (4D) spatial representation of the plenoptic function: the first two coordinates correspond to the position of each microlens within the camera and the other two coordinates correspond to the pixel position within each microlens image, the so-called *micro-image*. With these four coordinates, it is possible to measure the light for rays with different directions incident to a certain spatial position. The second type of acquisition, the array of (conventional) cameras, is characterized by the number of views, its density and their arrangement in space. Naturally, the higher the number of views and its density, the better the visual experience as typically more and better views are synthetized at the receiver based on the captured and coded views. The main drawback of this format is that when using lossy

compression, the synthesized images quality can be significantly affected. There is sometimes some confusion between the terms light fields, multiview and supermultiview representation models; their difference is mostly related to the number of cameras they involve which is much lower in multiview systems, meaning for example that the VAC is typically still a problem; on the contrary, supermultiview is rather equivalent to an array of light field cameras although typically limited to a linear, horizontal arrangement. For an insufficient number of cameras, discontinuous motion parallax and insufficient quality of the synthesized views may occur. After acquiring the light field data, there is always the option to convert it to a point cloud or a 3D mesh representation, or directly use it as a light field; this essentially depends on the application requirements and functionalities.

2. **Light Field Plus Depth** – This is a 3D representation format that adds depth maps to each of the views acquired in a light field. This is used to achieve more direct and reliable information about the scene geometry to enable encoders to know better the correlation between the multiple views and decoders to synthesize better virtual views [11]. In comparison with the pure light field format, this format allows to lower both the number of cameras and the coding bitrate as the availability of depth information allows the receiver to render better views even if the decoded views are farther apart (this means with lower density). Naturally, this format requires the availability of a new type of sensor to acquire real depth unless the depth is directly extracted from the texture.

3. **Point Cloud** – This representation format consists in a set of 3D positions with a certain density, defined by its coordinates, and some corresponding attributes; the attributes may include color information with more or less angular density, e.g. there may be several colors associated to one coordinate position, e.g. depending on the viewpoint. Examples of point clouds are shown in Figure 5, both containing color information. The coordinates indicate the 3D world location of a point (x,y,z), whereas the attributes may consist of colors (RGB), normal vectors, transparency, thermal information, etc. A normal vector, for a given point, is the perpendicular vector to the surface. Since the point cloud datasets represent the real surface through a set of point samples, it is possible to obtain normal vectors using normal estimation [12].

Figure 5: Examples of colored point clouds *[13] [14]*.

A point cloud may be defined as:

- **Static vs. Dynamic –** While a static point cloud consists of a single 3D *frame*, a dynamic point cloud evolves in time. In terms of compression, it is different compressing a dynamic and a static point cloud as dynamic point clouds have also redundancy in time which is critical to exploit. When dealing with dynamic point clouds, motion may be exploited to predict how the point cloud data changes over time [15].

- **Organized vs. Unorganized –** An organized point cloud is defined with 3D data structured as a 2D array, with rows and columns, resembling a matrix structure. If the relationship between adjacent points is known, nearest neighbor operations may be more efficient. On the other hand, unorganized point clouds include no relationship between the points, meaning that the dataset is not structured. Therefore, the computation cost of some tasks for unorganized point clouds may increase.
- **Single-rate vs. progressive coding -** When using a single-rate representation, the coding and decoding of both geometry data and attributes are performed as a whole [16] in the sense that no meaningful (full object) information may be decoded with less than full rate. In contrast, progressive coding allows transmission and rendering at different levels of detail (LoD) or quality depending on the consumed rate. In a streaming context, e.g. with varying rate, progressive coding may lead to better results as there is always a useful stream for any rate.

4. **3D Mesh** – This is a 3D representation format describing an object or scene by a set of polygons in the 3D space, each one with corresponding vertices and edges [17], see Figure 6 for examples. This representation is strongly related to point clouds since the vertices may correspond to points in a 3D domain, e.g. as in a point cloud. In terms of resolution, 3D meshes can provide a scalable representation from a coarse estimate of the object to a very fine LoD. One possible way to obtain texture for each polygon of a mesh is by interpolating the colors available for the vertices.

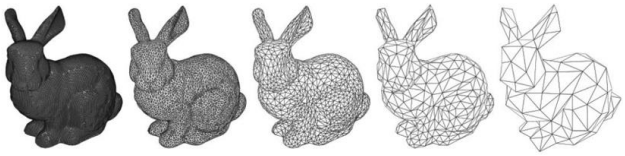


Figure 6: Triangle mesh examples with the number of polygons lowering from left to right *[18]*.

5. **Holography** – This is a 3D representation approach which measures the intensity of the interference fringes between a laser and its corresponding diffracted field by an object. This representation is called a hologram, and it reproduces everything that an observer can see, remarkably its size, shape, depth, texture and relative position [19], see Figure 7 for examples. Therefore, under ideal conditions, there should be no differences between the original object or scene and the hologram created. Nowadays, there are two main types of methods to acquire a hologram: the physical process with one or more lasers, and computer generated holograms which are synthetically created in a computer using appropriate propagation models applied to available 3D objects; this second process allows to overcome many of the constraints and limitations still associated to the physical creation process.

(a) (b)

Figure 7: (a) Physically generated hologram *[20]*; (b) Computer generated hologram *[21]*.

2.2. Applications and Requirements for 3D Representations Models

The representation formats introduced in Section 2.1 are relevant to several applications since they should enable more powerful user experiences. Moreover, all the applications to be introduced in this section vastly benefit from 3D representation formats in the sense that the QoE may be improved by providing additional functionalities impossible to achieve using traditional 2D representation formats. It is important to notice that some of these applications are already part of our daily lives, e.g. geographic information systems, whereas others have only being studied and developed in recent years, e.g. 3D immersive telepresence.

Some of the most promising applications considered in the scope of this Thesis are:

1. **Geographic Information Systems –** Applications used to store, view, manipulate, analyze, manage and display geographic and spatial information [22], such as maps and 3D models of buildings. This type of information is often represented as point clouds (or meshes for smaller datasets) so that the end-user is able to visualize the information with no viewpoint constrains, as opposed to 2D imaging modalities. Usually LIDAR systems are used to acquire these point clouds, which may result in massive amounts of data due to the high resolution of these devices. The key requirements for this type of applications are: *region selectivity*, the capability to query a portion of the entire point cloud; *lossless* coding to achieve the best possible representation when sensitive values are presented, e.g. water level, thermic (temperature) information; and *progressive* coding to enable efficient streaming of content since these datasets can cover a large geographical area with high precision.
2. **Cultural Heritage –** Applications dealing with cultural artifacts, e.g. paintings, drawings and sculptures, values and traditions that are passed down from previous generations and preserved in the present so that people can experience them now and in the future [23]. The 3D representation models, e.g. point clouds, are one of the ways to archive and visualize this type of information, e.g. art objects and collections. Moreover, they offer the advantage of enabling a powerful and immersive experience to the user without being physically present in the place, e.g. through Virtual Reality (VR), where these objects are exhibited, e.g. a museum. The main requirements that should be taken into account for this application are: *progressive* coding to enable increasing quality; and *lossless* coding for the best representation (if possible).
3. **3D Broadcasting –** This application is associated to the streaming, live and on demand transmission of an event such as a football game, a music concert or the New Year's Eve, while allowing free viewpoint interaction. This is an application for which the traditional 2D representation model cannot provide an immersive enough experience, e.g. being in the stadium. The primary requirements associated with 3D broadcasting are: *low delay* encoding/decoding (associated with *low complexity* compression) for real-time implementations, e.g. live broadcast; *lossy* coding as it is not essential to keep the fidelity of the information, with suitable bitrate control mechanisms; and *progressive coding* so the decoded data can be refined over time, depending on the bitrate available.
4. **3D Immersive Telepresence –** Application where VR technology can offer the user the sensation of being present at a different location other than the physical location [24]. The only way to achieve a realistic and immersive telepresence experience is by adopting 3D representation formats, e.g. 3D

meshes, to represent the scene to transmit and to integrate it into the physical world. This novel application has several advantages, notably reducing time spent on travel, reducing carbon footprint (environmental impact), and improving employee's quality of life and productivity [25]. In addition, there are a few essential requirements that must be considered: *low complexity* to allow *real-time* encoding/decoding; *lossy* coding, similarly to the scenario above; and *view dependence* (e.g. only query a subset of the point cloud, notably the portion that is being visualized by the user) to optimize the transmission process in streaming environments.

The point cloud 3D representation format seems rather promising for some of the applications above. For some of them, there are already products and datasets available, e.g. points clouds are very common in geographical information systems. However, there are many more applications that can benefit from 3D representation formats, notably light field photography, movie and video game production, biomedical imaging, depth photography, industrial imaging, etc.

In the context of point cloud representation, the applications above define a relevant set of requirements in terms of coding technology. Therefore, the codec to be designed must account for them in order to provide the best possible QoE. In this context, the main point cloud coding requirements are:

1. **Static and Dynamic Point Clouds –** Both static and dynamic (evolving in time) point clouds shall be coded.
2. **Efficient coding –** The coding process should be as efficient as possible for a target quality or target rate.
3. **Lossless Coding –** Decoding should output data mathematically identical to the original data, meaning that the fidelity of information is maintained. Using this type of compression, the decoded point cloud will be precisely the same as the original point cloud, notably with the same number of points, the same coordinates and exactly the same attributes.
4. **Lossy Coding –** To increase the compression efficiency, there may exist some distortion between the original and decoded point cloud data. For geometry coding, the number/position of points in the decoded cloud will not be precisely the same as in the original point cloud. For attribute coding, the values associated with each point will not be the same when compared to the original point cloud data. This type of coding intends to reduce the bitrate (for a target quality) which may be controlled using some appropriate rate control mechanism. Naturally, it introduces a trade-off between the perceived data quality and the bitrate.
5. **Error Resilience –** Defined as the ability to reduce the negative quality impact of packet losses introduced during the transmission process; this is especially important when the retransmission of lost data is not possible.
6. **Progressive Coding –** Useful decoding is possible for growing rates taken as subsets of the full rate while providing decoded data with growing quality, this means refining the decoded data over time.
7. **Low Complexity Coding –** The term complexity is associated with the computational effort, e.g. number of operations, memory accesses, etc. Typically, (low) encoding complexity is more critical than decoding complexity but this largely depends on the type of application.

8. **Low delay –** While there is delay associated with the computational complexity, and transmission, these delays may be reduced with more powerful platforms and more channel rate. This low delay requirement refers to the so-called *algorithmic delay*, which despite the platforms speed (where the codec is implemented) will always be present as this delay is associated to the coding algorithm itself, e.g. due to use of prediction frames. To allow efficient real-time implementation this type of delay should not be present meaning that there should be no reordering of point cloud frames between decoding and display.
9. **Parallel Coding –** Should be supported with the objective of spreading the encoding/decoding complexity associated to the point cloud data among several processors.
10. **View Dependent Coding –** Consists in coding only a subset of the point cloud, e.g. only what the user is viewing at a certain time using streaming.

The requirements above must be supported by a powerful point cloud coding framework, eventually considering more than one codec, e.g. to address both the lossless and lossy requirements. However, this does not mean that all applications require all requirements; for instance, a specific cultural heritage application may not benefit from dynamic or lossy point cloud coding. Furthermore, each application should only use the tools corresponding to the requirements that will offer some advantage.

Some of the applications previously described are strongly related to some key emerging technologies, notably VR and Augmented Reality (AR). Nowadays, these technologies offer a more immersive experience associated to the emerging VR/AR display devices. However, 3D technologies are not restricted to these devices as other devices such as auto-stereoscopic displays and free-viewpoint video systems can also benefit from richer 3D representation formats. Due to its current relevance, it is worthwhile to spend a few words to characterize these key technologies:

1. **Virtual Reality** is a technology allowing the user to experience, with all 6DoF or less, another world, real or computer-generated, and to interact with it, e.g. with certain objects of the environment, while not being physically present there. This definition of VR is based on several sources and tries to consider the identified key attributes in VR, e.g. the combination of interaction, immersion and simulation of physical presence. Despite the user not being physically present, the virtual world should be so realistic and immersive that the user can navigate, e.g. through user viewpoint dynamic control, and even interact with these environments, e.g. through sensory devices such as gloves, as if it is actually there. The main displays currently used to experience this technology are head-mounted displays (HMD), e.g. HTC Vive and Oculus Rift [26]. Nowadays, this type of VR glasses are opaque, e.g. do not let light rays through, making the user completely unware of the physical world where he/she is really located. Figure 8 (a) shows an example of this type of display.
2. **Augmented Reality** is a technology that layers computer-generated enhancements on top of the physical world environment (which is also visualized by the user), in real-time; this means that virtual (real or synthetic) data, e.g. sound, information and graphics, is harmoniously blended with the physical environment. Again, this definition of AR is based on several sources and tries to consider all the relevant aspects inherent to this technology, e.g. real-time, and mixture between physical and virtual worlds. Computer vision and visual search play a paramount role in AR since a good QoE can

only be obtained if the virtual data and the physical world are in perfect harmony, e.g. alignment and registration are usually needed. Nowadays, most of the AR devices are see-through glasses with a suitable projector to layer virtual data on top of real data, e.g. Microsoft Hololens (Figure 8 (b)) and Google Glasses.

(a) (b)

Figure 8: (a) Oculus Rift gear *[27]*; (b) Microsoft Hololens glasses *[28]*.

2.3. Point Cloud System Architecture and Walkthrough

As mentioned in Chapter 1, the 3D representation format adopted in this Thesis is the point cloud format. To better understand how a point cloud can be acquired/created, processed and displayed in a real world scenario, a simple although rather complete processing chain will now be presented and briefly explained. Since there are heavy interrelations between all modules in the architecture, the algorithmic design has to take into account these interrelations. The adopted point cloud system architecture is shown in Figure 9. The processing chain has the objective to recreate a captured real world scene in the most realistic and immersive way, thus providing a high QoE to the user.

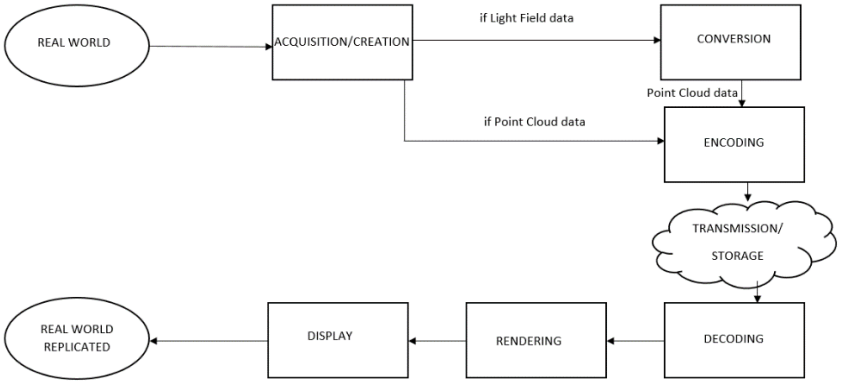


Figure 9: Point cloud system architecture.

1. **Acquisition/Creation** - The first step of the point cloud system is the acquisition of a real world scene or object. This process may happen using several acquisition systems, outputting either light field data, light field plus depth data or point cloud data. The way the 3D scene is acquired depends on the application requirements and the sensors/cameras available. For instance, to acquire point cloud data a sensor to compute object distances must be available. Moreover, it is also possible to create synthetic data using computer generated point clouds. This module will be thoroughly analyzed in the next section, where the point cloud acquisition systems will be distinguished into two main types: direct and indirect based acquisition.

2. **Conversion** - In the case the data acquired is not a point cloud (e.g. a light field was acquired), it is necessary to perform a format conversion. This module will convert the data from the adopted acquisition model (largely determined by the sensors) into point cloud data. After the data is properly converted, or if the acquisition is direct, the next module will code the point cloud.

3. **Encoding** - The encoding and transmission/storage are directly related in the sense that efficient encoding is needed for transmission and storage over limited bandwidth channels and storage devices. Moreover, the application requirements should drive the specifications regarding the encoding and transmission/storage modules. The main goal of the encoding module is to reduce the coding rate for a target quality by exploiting the redundancy and irrelevancy in the raw data. There are two main types of coding: lossless and lossy. When the raw point cloud data is lossless coded, the fidelity of the information is preserved; when the point cloud data is lossy coded, there are differences (distortion) between the original (raw) and the decoded point clouds. For lossy coding, just-noticeable distortion (JND) refers to the maximum distortion that the human visual system (HVS) cannot perceive, which means that the rendered content may still look subjectively the same as the original [29]. The purpose of this Thesis is to design solutions which make this module the most efficient and robust as possible. To do so, it is essential to understand how the encoder should deal with the different data characteristics that are inherent to point clouds, e.g. coordinates and attributes.

4. **Decoding** - After the data is transmitted/stored, it is necessary to reconvert it into a format that can be rendered and displayed afterwards. The decoding is the process in which the bitstream obtained at the receiver is reconverted to the adopted representation format so it can be rendered and displayed. Consequently, this process will output point cloud data in a format similar to the one used for encoding in this case point cloud data. While, in conventional video, the decoded data is ready to be visualized, this is not the case now where the rendering module still has to extract the data to be (usually 2D) displayed.

5. **Rendering** - The process where the data to be visualized by the final user is created is denominated rendering. In this process, the data recovered by the decoder is converted to a format which can be visualized by the user, which naturally depends on the available display; if only a conventional 2D display is available, the decoded point cloud has to be rendered into conventional 2D frames. This module will be scrutinized in Section 2.5 addressing the rendering of point clouds. Note, however, that the input of the rendering module can be a hybrid 3D representation, e.g. with both point cloud and light field scene components. A hybrid representation may allow to generate a more complex and realistic rendered view of a scene by exploiting the advantages of the light fields and point clouds in different parts of the scene, e.g. light field for a detailed background and point clouds for interactive objects.

6. **Display** - At this stage, the content is ready to be displayed and experienced by the final user. For example, the user will be able to see a realistic world replica with VR glasses. In Figure 10 (a) a real-time holoportation system output can be visualized, where the object, e.g. a human body, is acquired using an array of cameras and depth sensors and then converted into a 3D mesh representation. In Figure 10 (b), there is a VR example using rendered point cloud data.

(a) (b)

Figure 10: (a) Holoportation using 3D meshes *[30]*; (b) VR using point clouds *[31]*.

2.4. Point Cloud Acquisition Systems

This section will review the several acquisition systems that are currently available to generate point clouds. It is important to notice that, usually, point cloud data consists of geometry and color. Notably, there are two distinct ways in which a point cloud may be acquired: by indirect acquisition, where the point cloud is obtained by estimation from a set of texture images, which represent the visual scene from different perspectives (views); and by direct acquisition, where the point cloud is obtained through two different types of sensors, one to measure the distance between the objects/scene and the sensors, the other to acquire scene texture. These types of acquisition procedures will be reviewed in the next subsections, ending with an interesting acquisition system example.

Regarding the acquisition systems that will be introduced, it is important to first define some key attributes related to them:

1. **Accuracy –** Indicates how close the device measurements are to actual values.
2. **Acquisition speed –** Indicates how frequently the device acquires information.
3. **Power consumption –** Indicates how much energy the device consumes over time.
4. **Measuring range –** Indicates the distance range for which the device can work within specifications.
5. **Image resolution –** Indicates the final quality/fidelity, in terms of number of pixels in the image, when compared to the real scene acquired.

2.4.1. Indirect Acquisition

As previously stated, this type of acquisition consists in creating a 3D point cloud from a set of images that represent different perspectives of the visual scene. Therefore, the scene geometry is indirectly obtained which may lead to a higher computational effort, i.e. an additional conversion module is needed to obtain a point cloud, and eventually lower accuracy. In addition, when dealing with low-brightness environments, these systems do not lead to good results. In this context, there are two different ways to implicitly acquire geometry information: stereo pairs and light fields:

1. **Stereo Pairs** – In this system, two traditional 2D texture cameras with a fixed distance between them, the baseline, are used to acquire luminance and chrominance data. To obtain the visual scene geometry, it is common to compute the disparity between the views. Thus, stereo matching is performed to find the correspondence between a set of points or features in one image with the

corresponding points or features in the other image. Afterwards, a depth map of the scene is computed based on the disparity values previously computed as the depth is inversely proportional to the disparity. Nowadays, this technology offers rather medium accuracy for small distances and low accuracy for larger ones. It is important to add that this type of acquisition is usually associated with a limited field of view when a single stereo pair is used.

2. **Light Fields –** As stated previously in Section 2.1, this type of data can be acquired using an array of conventional cameras or a single lenslet camera. In both cases, the output of these acquisition systems is multiple 2D images which represent different perspectives, containing luminance and chrominance data, with more or less disparity between views (rather small for the lenslet images). From this type of data, it is possible to compute depth maps. This is nowadays an important research area and several algorithms have been studied to estimate depth from light fields, notably multi-view stereo matching and epipolar-plane images (EPI)-based estimation [32]. These algorithms generate depth information for each of the multiple views previously acquired. In terms of the generic attributes (such as accuracy, power consumption and price), these light field based solutions largely depend on the number of cameras and the baseline associated with them.

As stated above, the output of these two acquisition systems are depth maps. Therefore, there is a need to convert this geometry information into a point cloud. As a depth map is an image whose pixel values represent the (perpendicular) distance from the sensor's plane origin to the surface of the scene objects, the depth map can be converted into a 3D point cloud if knowledge on the camera's intrinsic calibration parameters is available [33].

2.4.2. Point Cloud Direct Acquisition

For this type of acquisition, the output of the devices is point cloud data obtained by measuring the distance between the scene and the sensor and converting them into 3D points in space by applying a change of coordinates. That is, using the sensor device coordinates, the distance and the direction of the light, it is possible to obtain the corresponding 3D point in space. To add color to these 3D points, a texture sensor must complement the geometry acquisition. Therefore, this subsection will be divided into texture and distance sensors. By using both texture and distance sensors, the output of a direct acquisition system is a 3D colored point cloud.

Texture Sensors

In this context, there are two types of sensors that produce 2D images containing luminance and chrominance data: lenslet and traditional 2D texture cameras. The lenslet cameras output and concept was previously outlined on Section 2.1. Traditional 2D texture cameras produce a single 2D texture image. Both these sensors provide key information about the scene texture, and when combined with scene geometry, e.g. through distance sensors, the output is a point cloud with color associated to each 3D point in space.

Distance Sensors

There are three different types of sensors that can measure sensor-to-object distances, thus providing key information about scene geometry:

1. **Time-of-Flight** – This type of device (shown in Figure 11 (a)) measures the time it takes for a light signal to reach a certain target and then infers its distance. The distance between the camera and each 3D point is measured by projecting (usually infrared (IR)) light into an object and by measuring after the difference in phase between the transmitted and the reflected light, which allows the computation of the *time-of-flight* of the light signal [34]. Usually, time-of-flight devices are useful for shorter distances, e.g. less than ten meters. Moreover, these devices offer very high accuracy values.
2. **LIDAR** (Light Detection and Ranging) – This type of device relies on a sensing technology based on laser light (shown in Figure 11 (b)) and measures the distance between the emitting device and each point in the 3D space, usually by illuminating the object/scene with laser pulses and capturing the reflected light [35]. The time difference between the emitting signal and the reflected signal allows to measure the distance to the 3D point in the object. To measure the distances to several 3D points in space, the devices use beam steering, which is a method to change the direction of the main lobe in a radiation pattern [36]. Traditionally, LIDAR systems use mirrors mechanically spinning around while oscillating up and down. Therefore, most current LIDAR devices include these mechanical parts to achieve a useful field of view, which inflates its price and may limit their usage. Knowing the distance and direction for each point in 3D space, it is possible to produce a full 3D point cloud model of the visual scene. This type of device can capture objects at long distances, e.g. up to 20 km. These devices have low accuracy when capturing large distances and medium accuracy when dealing with smaller ones [37].
3. **Structured Light** – This type of device captures the geometry of the visual scene by projecting a reference light pattern, e.g. grids and horizontal bars, onto an object and capturing the reflected light using a regular video camera. Naturally, the reference light pattern appears distorted (according to the object shape) when captured by the camera, thus allowing to compute the 3D coordinates of the object geometry, i.e. by analyzing how the light pattern is distorted when the light is reflected [38], the 3D coordinates can be obtained using computer vision methods [39]. The light projected can be IR, e.g. wavelength range below 14nm, or visible, e.g. wavelength in the 390nm-700nm range. If visible light is used, the reflected pattern is sensitive to bright-light environments, making the system better suited for indoor captures. To extract full depth information about the target, e.g. different facets, successive projections are required, making these devices mainly suited for static objects [34]. Microsoft Kinect designed for Xbox 360, see Figure 11 (c), uses the structured light concept through an IR projector and an IR imaging sensor. This type of device has high accuracy, medium image resolution and low measuring range [40].

(a) (b) (c)

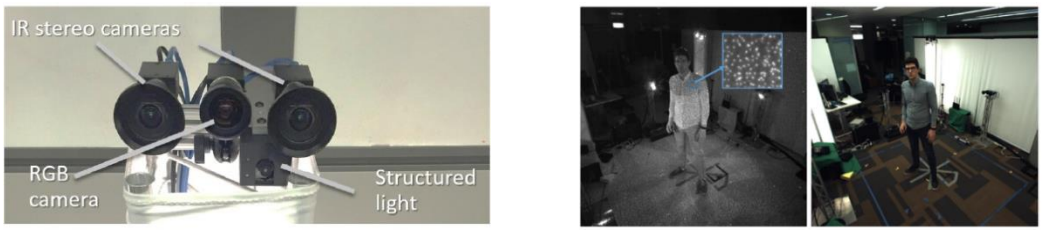
Figure 11: (a) ToF camera *[41]* ; LIDAR scanner *[42]*; Microsoft Kinect *[43]*.

Naturally, all three sensors introduced above output point cloud geometry information. Therefore, it is necessary to combine the output of the texture sensors previously introduced and this ones, so that each 3D point in space has a color associated. Having the texture image and the coordinates associated with both the distance and texture sensors, it is possible to determine for each pixel its associated geometry information [44]. This process results in a point cloud with color for each 3D point in space captured.

It is important to notice that, if the distance sensor and the object are fixed, only the points seen by it may be analyzed. Therefore, to obtain a realistic point cloud representation, e.g. no occlusions, the sensor must acquire all points of the scene, usually capturing it from different viewpoints (array of sensors for dynamic scenes or using the same sensor for static scenes). In this context, it will become necessary to perform 3D registration to integrate the various partial point clouds into a complete point cloud. This method consists on the alignment of different 3D point cloud data views to produce a full 3D point cloud model [45]. After this procedure is completed, the whole scene captured will be represented as a single set of 3D points (*xyz*) in space, e.g. a point cloud with multiple perspectives, with texture.

2.4.3. Interesting Acquisition System Example

The real-time holoportation system, designed by Microsoft, combines distance and texture sensors, to make a correspondence between each distance and color pixel [46]. The system uses 8 camera pods, to ensure full 360º capture. Each camera pod contains two Near Infrared (NIR) cameras and a color one, see Figure 12 (a), as well as a diffractive optical element (DOE) and a NIR projector to produce light patterns [46]. The NIR cameras are able to capture the light patterns that the NIR projector adds to the scene, whereas the color camera obtains a traditional texture image, see Figure 12 (b). This setup can acquire both NIR and texture images in real-time. The usage of NIR projectors to add more texture to the scene makes depth estimation more accurate. By having more texture, provided by the light patterns (which can be overlapped), the matching algorithm is able to disambiguate more easily patches across cameras. Therefore, the net result for depth estimation is an improvement, thus resulting in more accurate depth maps. To conclude, this system uses both indirect and direct acquisition, so the accuracy of the 3D point cloud is higher.



(a) (b)

Figure 12: (a) Trinocular Pod; (b) NIR and RGB images.

As a summary of this section, Table 1 shows a comparison between the various acquisition systems introduced above in terms of some key characteristics. For the cases with "*", it was not possible to accurately define a reliable, single label. The idea is that by using such table it should be possible to select, for different environments and applications, the most suitable point cloud generation method.

Table 1: Comparison between several point cloud acquisition systems *[47] [37] [43] [48]*.

Attributes	ToF	LIDAR	Structured Light	Stereo Pair	Light Field
Accuracy	Very High	Low - long distances High - short distances	High	Low - long distances Medium - short distances	*
Image Resolution	*	*	Medium	High	*
Acquisition Speed	Medium	Fast	Medium	Medium	*
Measuring Range	Low	High	Low	High	*
Power Consuption	Very Low	Medium	Low	Low	*
Price	High	Very High	Medium	Medium	*
Low-light Performance	Good	Good	Good	Weak	Weak
Bright-light Performance	Good	Good	Weak	Good	Good

2.5. Rendering of Point Clouds

The rendering of point clouds corresponds to the process of generating data from a point cloud in a format suitable for visualization on a specific display to be experienced by the final user. Naturally, the rendering process depends on the type of display the user intends to use to watch the content; some examples are HMDs, 2D displays, stereoscopic displays with glasses, glass-free autostereoscopic displays and even volumetric displays. The rendering of a high-quality point cloud into an HMD can be a complex process due to the device characteristics, e.g. low Central Processing Unit (CPU) speed and Random Access Memory (RAM). To achieve good QoE, the rendering can be performed by offloading to a computer (i.e. remote rendering), which should allow consistent frame rate, lower perceived latency and lower power consumption on the visualization device.

There are two major types of point cloud depending on the way point clouds are generated: computer generated, e.g. synthetic content, and real-world content, this means acquired using cameras and sensors. Nowadays, when dealing with synthetic content, e.g. for games and movies, the producer can select the position of the camera (or cameras for 3D displays) and the light sources characteristics (how many, type of illumination for each one, etc.) that will be used to render the content. Typically, this content is represented with meshes but point clouds can also be employed. Rendering of 3D meshes is a technology that has been around for a while and there are many rendering algorithms that may be used in this context [49] [50] [51]. For example, there are several computer graphics rendering techniques that perform shading, reflection, refraction and indirect illumination, and are able to improve (when properly applied) the quality of the rendered data. On the other hand, for real world content, the rendering techniques are mostly focused on two major issues: rendering quality, i.e. the fidelity of the rendered output in comparison to the captured scene, and rendering complexity, i.e. computational resources and memory necessary to recreate the 3D visual scene. Considering the solutions in the literature, there are two main ways to render a point cloud:

1. **Direct rendering:** Consists in directly rendering the point cloud data without converting to another 3D format, such as 3D meshes or light fields. This is a very efficient way, in terms of computational complexity, to visualize highly detailed 3D models (i.e. dense and complex geometry). Moreover, as many acquisition systems output large point clouds, surface reconstruction of these dense data sets can be highly complex or even impossible in some devices.
2. **Indirect rendering:** Consists in using an additional representation model conversion step where the point cloud is migrated to another 3D representation format. Most commonly, point clouds are converted to 3D meshes using a surface reconstruction algorithm; for instance, Poisson surface reconstruction is commonly used to create surfaces in the form of meshes [52]. One advantage of this indirect rendering approach is that a simple 3D shape can be represented by few polygons, while point-based representations, e.g. point clouds, do not account for the shape complexity, i.e. for low shape complexity, the point cloud density may not be lower [53]. However, for increasingly complex scenes, the mesh representation may lead to many polygons which are very small, sometimes smaller than a single pixel.

Since point clouds are the core format of this Thesis, direct point cloud rendering techniques are privileged as no other representation model is involved. One of the simplest ways to render a point cloud is to project the 3D points into 2D images, the so-called *point projection*. However, using only point projection, i.e. without any other methods to augment quality, the rendered quality and its readability (i.e. the perception of where the 3D points are placed) are very limited, thus making hard to understand the 3D content. For instance, point projection does not take into account the occlusions and all points are projected despite being visible or hidden, as illustrated in Figure 13 (a). There are three main concepts that need to be introduced when targeting the efficient rendering of a point cloud with high quality: LoD adaptivity, hole-free imaging and visibility culling.

1. **LoD Adaptivity** - LoD adaptivity involves reducing the complexity of the point cloud being rendered, while accounting, for instance, for the viewpoint position, the object distance and other factors. LoD adaptivity may increase the rendering efficiency, eventually at the cost of some lower quality. It is important to note that there are two different LoD algorithm modalities: those storing objects at discrete LoDs, e.g. Layered Point Clouds (LPC) [54], and those performing fine-grained LoD control, e.g. Qsplat [51].
2. **Hole-free** - Hole-free imaging consists in rendering point clouds in a continuous way, i.e. without holes between rendered points. Therefore, methods to close the gaps between rendered point cloud samples are employed. There are two main ways to create hole-free images: using object-space (i.e. by converting the points into new object-space samples) and screen-space (i.e. filling methods, see Figure 13 (b) for an example) techniques.
    - **Object-space:** In object-space methods, the hole-free imaging results from the conversion of points into object-space samples, e.g. splats, in the 3D domain. The last rendering step in these methods is the screen-projection where the obtained object-space samples are projected into the 2D domain.
    - **Screen-space:** In screen-space methods, the first operation of the rendering process is to project

the points into the 2D domain and only afterwards apply hole-filling methods to generate hole-free images.

(a) (b)

Figure 13: (a) Rendering with point projection only; (b) Rendering with visibility culling and hole-free imaging through screen-space method *[55]*.

3. **Visibility Culling** - The term culling means discarding (i.e. not rendered) and thus visibility culling regards the removal of points which are not visible for a given 2D viewpoint. Therefore, when visibility culling is employed, only 3D points that are visible from the viewpoint being rendered contribute to the final rendered image. In this context, there are four types of visibility culling:

   - **View frustum culling:** Only considers for rendering the 3D points that lie entirely inside the viewing frustum, i.e. the geometric representation of the volume visible to the projected 2D plane [56], as illustrated in Figure 14, discarding all the points lying outside.
   - **Back-face culling**: Only considers for rendering the 3D points that lie on a surface facing the camera, discarding all those that are on the back side of an opaque object.
   - **Occlusion culling:** Discards the 3D points that lie on objects that are completely occluded, i.e. in relation to the viewpoint being rendered, the 3D points lie behind opaque objects, therefore, should be discarded.
   - **Contribution culling:** Consists in discarding the 3D points that are representing an object so far away that do not contribute to the scene, e.g. the screen projection is too small. This may have an impact on the final quality of the rendered point cloud.

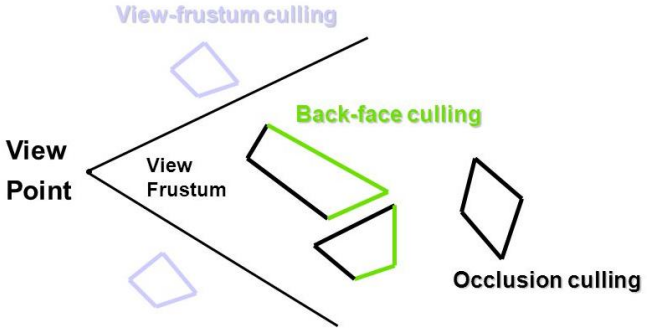
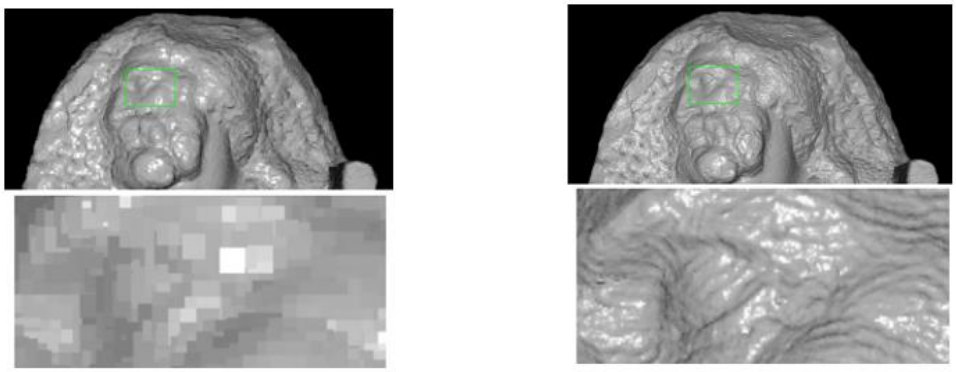


Figure 14: Examples of view frustum, back-face culling and occlusion culling on 2D space *[57]*.

In the following, two direct rendering systems, using the concepts introduced above, are briefly reviewed; the first one achieves hole-free imaging through object-space while the second uses screen-spacing.

1. **QSplat: A multiresolution point rendering system for large meshes** [51] **–** One common way to render point clouds using object-space is by converting the points into splats, see examples in Figure 15. A splat is a circular or elliptical surface element that, through assessment of the point arrangement, is aligned with the surface area that is representing. Although the Qsplat system was initially applied for 3D mesh rendering, it also allows the direct rendering of point clouds as it ignores any connectivity information.



(a) (b)

Figure 15: (a) Qsplat example, close-up below, with lower LoD; (b) Qsplat example, close-up below, with higher LoD *[51]*.

This method uses a tree hierarchy of bounding spheres, allowing object-space hole-free imaging, visibility culling and fine-grained LoD adaptivity. Each node of the tree contains several splat attributes such as sphere radius and center, a normal, the width of a normal cone and the color (optional). In this tree hierarchy of spheres, leaf nodes are created for each 3D point coordinate of the point cloud and the tree upper levels allow rendering a point cloud with lower levels of detail. During the rendering process, visibility culling is performed so that nodes that are not visible are removed from the rendering buffer. Three forms of visibility culling are used by this system: view frustum, back-face and occlusion culling. The first is achieved by evaluating, for the viewpoint being rendered, the splats that will lie entirely inside the viewing frustum. The second uses the normal vectors stored at each tree node to identify if that node is occluded, e.g. in the back of an opaque scene object. Occlusion culling is only employed in the last rendering step (i.e. the conversion from object-space to screen-space), so it will be outlined later. In terms of LoD adaptivity, this system allows to smoothly change the number of splats rendered throughout the scene, i.e. a higher number of splats results in an increased output quality. This implementation takes into account the user viewpoint variations, meaning that, if the user remains in the same viewpoint, the system will render the point cloud with successively smaller splats (until the splat size reaches the screen pixel size). On the other hand, if the user keeps changing viewpoint, the system will use a lower number of larger splats to render the point cloud, i.e. using a LoD with lower quality. The last step is the drawing of the splats in the screen domain, using the Z-buffer (or depth buffer) algorithm so that occlusion culling is implemented [58]. Moreover, the definition of the splat size is based on the projected diameter of the sphere being traversed. Recently, several enhancements to this method, in terms of memory management and rendering speed as well as faster graphical processing unit (GPU) implementations, have been proposed [54] [59] [60].

2. **Real-time rendering of massive unstructured raw point clouds using screen-space operators** [55] **–** The main objective of this system is to render unstructured point clouds in real-time using screen-space methods. This method assumes that no prior knowledge about the sampling density and normal vectors for each point is available, which doesn't occur for the previous work, e.g. normal vectors are required for a correct alignment of the splat. However, the main key difference is that the surface reconstruction of the 3D objects represented by the cloud points is performed in the 2D domain by first projecting points into the viewing plane ('screen') and then finding an accurate value for the pixels with missing information. Thus, the proposed technique works with multiple passes, notably: 1) projecting 3D points into screen-space; 2) performing visibility culling for efficient rendering; 3) performing anisotropic filling to fill existing holes; and 4) shading to effectively convey surface shape. A multi-resolution data representation is used to handle massive point clouds. The system is appropriate for instant visualizations of scanned datasets since it has low memory requirements and avoids pre-computation of any point cloud attributes such as normal vectors. This method proposes a visibility operator to compute which points lie on the viewing frustum of the rendering perspective and are not occluded, given the orientation and position of the camera. In this method, a point is visible if it is possible, from the rendering viewpoint, to create a point-to-camera line. The points that are not visible, the so-called *hidden points*, are removed from the screen space, meaning they will not be rendered. In terms of filling, to make the rendered image hole-free, this system uses the hidden points, from the visibility step previously described, to perform an anisotropic filling. Afterwards, a shade depiction method is implemented with three types of shading contributions: directional light, ambient occlusion and line drawing shape abstraction. The objective of the shape depiction step is to provide a fast way to analyze the scene objects shape, thus increasing the model readability. The rendering pipeline is illustrated in Figure 16.

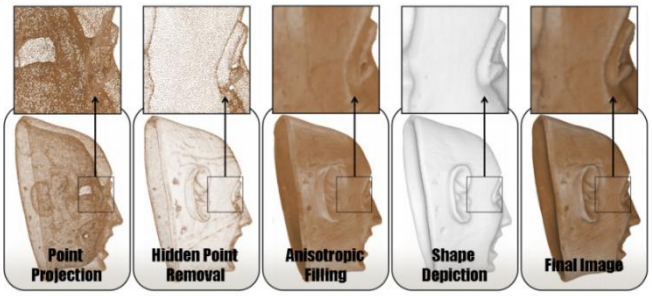


Figure 16: Rendering pipeline overview *[55]*.

2.6. Point Cloud Objective Quality Metrics

As for image and video codecs, a point cloud codec also needs to have its performance evaluated, notably in terms of its quality and compression efficiency, i.e. how much bitrate is required for a certain quality. Therefore, some objective quality evaluation metrics for point clouds are described in this section. Although there are several metrics available in the literature, the metrics described here are those adopted by the Moving Picture Experts Group (MPEG) group to evaluate the upcoming point cloud coding proposals to be submitted to the issue Call for Proposals (CfP) on Point Cloud Compression (PCC) [61]. The primary goal of this type of quality assessment metric is to compare the original point cloud (original raw data) with the decoded point cloud, which should have some artifacts (or errors) when compared to

the original point cloud, if a lossy codec is used. The quality evaluation procedure is illustrated in Figure 17. All the presented metrics are full reference quality metrics meaning that the decoded point cloud texture or geometry data (V_{dec}) is directly compared with the original one (V_{or}), with no knowledge about the encoding or decoding process in between. Naturally, these quality metrics may assess not only the quality of decoded data but also any data produced using the original data with any type of processing. In Figure 17, Q_{point_cloud} represents the texture or geometry quality metric and, therefore, should express the similarity in terms of geometry or color (depending on which type of metric is being computed), between the original and decoded point clouds. These quality metrics will be outlined later in this section. Moreover, to efficiently design the full reference quality metric, it is necessary to express V_{or} and V_{dec} in some formal way.

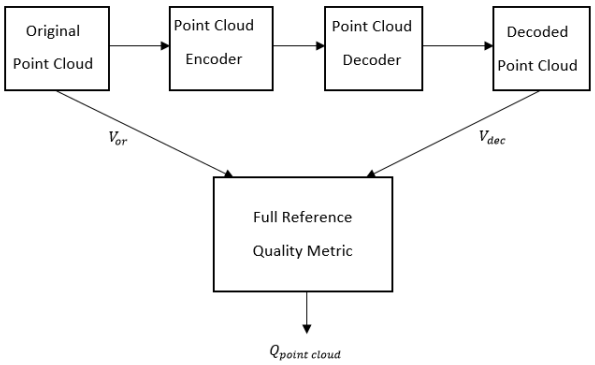


Figure 17: Full reference quality assessment for a decoded point cloud.

Consider that a point cloud point *v* can be expressed with a position in the 3D space (*xyz*) and a color attribute *c* (RGB or *YUV*) and other optional attributes, e.g. a normal vector. Thus, each cloud point can be formally defined as:

$$v = \big(((x,y,z),[c],[a_0..a_A]\big):\ x,y,z \in \boldsymbol{R}\ ,[c\ \in\ (R,G,B)|R,G,B\ \in N],[a_i\ \in [0{,}1]]) \quad (1)$$

Therefore, a full point cloud, e.g. the original 3D point cloud, can be simply defined as a set of *K* points:

$$V_{or} = \{(v_i): i = 0 \ldots\ldots K-1\} \quad (2)$$

The original cloud, V_{or}, will be the reference to establish the quality/distortion of a processed/ decoded cloud, V_{dec}, which consists of *N* points (where *N* can be different from *K*). This point cloud can result from lossy V_{or} encoding followed by decoding, thus resulting into

$$V_{dec} = \{(v_i): i = 0 \ldots\ldots N-1\} \quad (3)$$

With V_{or} and V_{dec} available, it is possible to compute the adopted quality metric (Q_{point_cloud}), as illustrated in Figure 17.

While accepting a full reference approach, there are three types of quality metrics that may be relevant and eventually have a different correlation with the perceptual rendered quality:

1. **Cloud to Cloud (C2C) quality assessment metrics –** These metrics compare the original and decoded clouds point to point with a focus to determine point correspondences between the clouds and compare them in terms of geometry and attributes similarities. The main disadvantage of this type

of quality metrics is that they do not consider the surfaces associated to the pair of points being compared what may be relevant, e.g. from a perceptual point of view.

2. **Cloud to Surface (C2S) quality assessment metrics –** These metrics assess the decoded point cloud quality by considering some reference surface associated to the original point cloud. In this context, the original point cloud must be associated to a surface, e.g. in the form of a 3D mesh or some other type of model. The objective is to 'compare' the decoded point cloud, both geometry and attributes, to the corresponding original surface using the obtained model. Naturally, this type of metric also considers points lying on the determined surfaces/meshes and, thus, a major disadvantage of this type of quality metric is that it depends on the specific surface reconstruction algorithm used.

3. **Cloud to Plane (C2P) quality assessment metrics –** These metrics assess the decoded point cloud quality by considering a reference plane associated to the original point cloud, e.g. the plane orthogonal to the normal at the closest point in the original cloud. In some sense, these C2P metrics lie in between the C2C and C2S metrics as they consider more than points but not yet the object surface as above. This method does not rely on any surface construction method as it only considers local plane properties. An example of this type of metric for point cloud geometry is illustrated in Figure 18; the main goal of this specific metric is to measure a so-called *projected error vector distance* which is the distance between a decoded point in the cloud and the plane determined by the normal of the closest point in the original cloud; this may be different from just computing the error vector distance between pairs of points, as done in C2C, as this type of metric may better express the perceived rendered quality [62]. An eventual disadvantage of this type of metric is that it requires the availability of the normal directions for each point in the reference cloud, which may require using some appropriate normal definition algorithm.

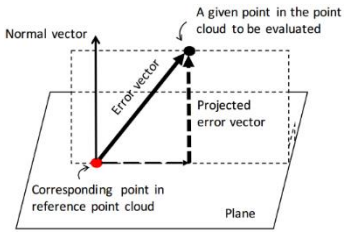


Figure 18: Example of a point to plane distance, the so-called projected error vector *[62]*.

In practice, as a point cloud considers geometry and texture, two types of metrics are needed to assess the quality of a point cloud: a geometry quality metric and a texture/color quality metric. The evaluation metrics currently adopted by MPEG [63] follow the C2C and C2P approaches. Only C2C metrics are reviewed below as C2P metrics have only recently been adopted. However, C2P metrics may show a higher correlation with the perceived rendered quality.

Besides the geometry and color quality metrics, which are mainly used to assess the quality of the decoded point cloud, there are other metrics that may also help in the overall compression performance assessment. For instance, a comparison between the number of points in the original and decoded clouds, which can be evaluated for various LoD. Moreover, the bitrate is also essential, if beyond the distortion/quality also the compression performance is to be measured; naturally, the bitrate has to be divided between color and geometry as their compression performance is typically independently

assessed. In this way, key information about the two types of compression is provided.

C2C Quality Evaluation Metrics: Geometry

From a geometry point of view, there are three important C2C metrics which target providing an assessment of the geometric quality of the decoded point cloud. These metrics are defined in the following:

1. **Symmetric Root Mean Square (sRMS) Distance –** Geometry distance based metric computed between two point clouds that measures the maximum of the average positioning errors of the decoded cloud using as reference the original cloud and vice-versa (thus symmetric). The sRMS distance is computed as:

$$sRMS(V_{or}, V_{dec}) = \max(d_{rms}(V_{or}, V_{dec}), d_{rms}(V_{dec}, V_{or})) \quad (4)$$

Where d_{rms} is the average distance between closest points, the so-called *neighbors*, from one cloud to the other. This distance is calculated for all points in the original point cloud and after for all points in the decoded point cloud as it is necessary to compute the symmetric distance, meaning that d_{rms} should be also computed from the decoded point cloud to the original one, e.g. $d_{rms}(V_{dec}, V_{or})$.

$$d_{rms}(V_{or}, V_{dec}) = \sqrt{\frac{1}{K} \sum_{\substack{v\_o \in Vor \\ v\_d \in Vdec}} ||v\_o(i) - v\_d(k)||^2} \quad (5)$$

Where *K* represents the number of points in the original point cloud, $v\_o(i)$ is a point in the original point cloud with index *i* in the point cloud data, and $v\_d(k)$ is a point in the decoded cloud, with index *k* in the point cloud data, which is the nearest neighbor of $v\_o(i)$ in the decoded point cloud. In this context, $||\ldots||$ represents the Euclidean distance between two cloud points.

2. **Symmetric Hausdorff Distance (sHausd) –** Geometry distance based metric that represents the worst geometric distance between neighbors in the two point clouds being assessed. First, for all points in the original point cloud, the neighbor distance is computed. The so-called *Hausdorff distance* (*Hausd*) is the maximum of these distances. To calculate *sHausd*, the *Hausd* metric must also be computed in a symmetric way, e.g. from the decoded cloud to the original one, as illustrated in Figure 19 (a). The distances associated to the dotted lines in this example follow the definition adopted for the Hausd distance: while the distance on the left corresponds to the *Hausd* distance from the P set (blue) to the Q set (red), the distance on the top-right is the *Hausd* distance from the Q set to the P set.

(a) (b)

Figure 19: (a) Example of Hausdorff distances for two 2D point sets *[64]*; (b) Example of a 2D point set axis-aligned bounding box *[65]*.

The *sHausd* metric is then given by the maximum between the two distances previously calculated as:

$$sHausd\left(V_{or}, V_{deg}\right) = \max(Hausd(V_{or}, V_{dec}), Hausd(V_{dec}, V_{or})) \quad (6)$$

With *Hausd* [62] corresponding to

$$Hausd(V_{or}, V_{dec}) = max(\ ||v\_o(i) - v\_d(k)||) \quad (7)$$

Where $v\_o(i)$ and $v\_d(k)$ are defined as above.

3. **Geometry Peak Signal To Noise Ratio (gPSNR) –** This is the geometry peak signal to noise ratio and represents the geometry quality of the decoded point cloud, considering the bounding box maximum width of the original point cloud, i.e. BB_{width}, and the sRMS which represents the average geometric distortion as defined above. In other words, this metric assesses the decoded cloud quality in terms of positioning errors (geometry) when compared to the original point cloud. Therefore, to compute this metric the following expression is used:

$$gPSNR = 20log_{10}(BB_{width}/\ sRMS(V_{or}, V_{dec})) \quad (8)$$

The bounding box is the smallest enclosing axis-aligned box within which all 3D points lie, see Figure 19 (b) for a 2D example. The so-called *maximum width* is given when computing the maximum value between: i) length, given by calculating the difference between maximum and minimum values of x; ii) width, difference between maximum and minimum values of y; and iii) height, difference between maximum and minimum values of z. In (8), BB_{width} represents the maximum width of the original cloud bounding box and is given by:

$$BB_{width} = \max((x_{max} - x_{min}), (y_{max} - y_{min}), (z_{max} - z_{min})) \quad (9)$$

C2C Quality Evaluation Metrics: Color

Regarding color, a similar quality metric to gPSNR is introduced, where each color component of the original point cloud is compared to the color of the nearest point in the decoded cloud, which leads to a PSNR for each component in the YUV color space.

1. **Color Peak Signal To Noise Ratio (cPSNR) –** PSNR is a common metric to assess 2D image quality and is usually computed for each component of the YUV color space. In the case of cPSNR, the color attributes between points of the original and decoded point clouds are compared. To compute this metric in the YUV color space, it may be necessary to convert the color from the RGB color space (which is often the color space of the original point cloud) to the YUV color space. Thus, the cPSNR is computed as:

$$cPSNR_y = 20log_{10}(255/\ d_y(V_{or}, V_{dec})) \quad (10)$$

The formula uses 255 as the peak signal since 8-bit color representations are assumed. The variable d_y represents the difference in terms of average color errors between the original and decoded clouds as expressed by:

$$d_y(V_{or}, V_{dec}) = \sqrt{\frac{1}{K} \sum_{\substack{v\_o \in Vor \\ v\_d \in Vdec}} ||y(v\_o(i)) - y(v\_d(k))||^2} \quad (11)$$

Where $v\_o(i)$ and $v\_d(k)$ are defined as above. The variable *y*(…) is the luminance value for each point, *K* is the number of points in the original point cloud and || … || represents the Euclidean distance between two values. The expressions above can be used for all YUV components, e.g. replacing *y* by *u* and *v*, to compute the peak signal to noise ratio for the chrominance components.

2.7. Point Cloud Coding: Reviewing Main Solutions

Point cloud data requires a massive amount of storage/transmission resources, and thus, it is paramount to develop efficient coding solutions to enable point cloud based applications. In this section, the most relevant point cloud coding solutions available in the literature will be reviewed. These solutions were selected considering their technical approach, performance and impact, so that a diverse set of solutions is reviewed. The Intra-frame coding solutions (i.e. without exploiting temporal correlation) are reviewed in more detail as static point clouds are the main target of this Thesis. Since a point cloud is made of 3D positions (geometry) and attributes, point cloud coding solutions may be divided into two classes, notably geometry and attributes coding, depending on the type of data being coded.

Geometry Data Coding

In this case, the focus is on the coding of the 3D coordinates of each cloud point, this means the point cloud geometry. In this context, two main classes of solutions appear in the literature:

- **Octree-based:** The octree-based coding method organizes the point cloud data using a tree-based data structure where every branch node represents a bounding volume in the 3D space, the so-called *voxel*. To create an octree, a top-down approach is usually employed where the original point cloud bounding box is divided into eight 3D subspaces, the so-called *octants,* and then each octant is recursively divided into eight smaller octants, as shown in Figure 20. In practice, octants are voxels at different resolutions. An octant containing one or more cloud points is classified as *occupied*. Afterwards, only occupied octants are recursively divided. There are several octree-based coding schemes for geometry data in the literature, notably: statistical average, i.e. each point, in a given octant, is analyzed and a statistical average is computed; polynomial approximation, i.e. each point, in a given octant, is represented by a high-order polynomial function; point differences, i.e. for each point, in a given octant, the coordinate differences of the point under analysis regarding the center of the octant (or other designated point such as the smallest *xyz* coordinates) are computed and coded. This will be the approach adopted for the novel coding solution to be proposed in this Thesis.

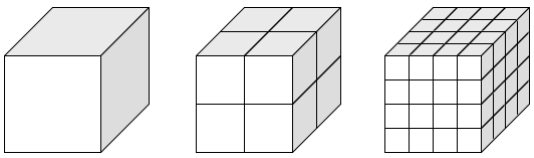


Figure 20: Left to right: 1) initial bounding box to be divided; 2, 3) each octant is further subdivided finally resulting into 64 final octants *[66]*.

- **Patch-based:** The patch-based coding methods create a set of 2D bitmap images, designated as *patches*, which represent the point cloud. By representing the point coud geometry using patches, it is possible to apply standard image codecs. The corresponding processing pipeline includes three main operations: i) partitioning, i.e. the set of points is divided into a suitable number of voxels that are decomposed into point clusters; ii) parametrization, i.e. 2D parametrizations of the point clusters obtained in the previous step, the so-called *patches*; the geometry attributes of the point clusters are projected onto the patch, creating height and occupancy maps; and iii) coding, i.e. the created symbols are transformed into bits. Moreover, using a suitable set of height and occupancy maps (properly placed in the 3D space), it is possible to represent a point cloud [67]. In this context, a height map is a 2D matrix including the geometric elevation of points over a surface. An occupancy map is a 2D matrix that represents the presence of points, i.e. data to be shown, as a field of boolean values.
- **Graph-based:** In the context of point cloud coding, a graph is a suitable structure to represent a set of 3D points, the so-called *vertices.* In this structure, nearby points are correlated and, therefore, connections between points may be created, the so-called *graph edge*s. Usually, a graph structure is created by connecting nearby 3D points, and assigning a weight to each graph edge. The graph transform is then learned using this graph structure; after, it can be applied to any signal that resides on the vertices of the graph. By using this transform, the signal energy is efficiently compacted, thus making it ideal for compression. In terms of compression efficiency, the graph transform is comparable to the Kahunen-Loève Transform (KLT) [68]. Although there are no solutions yet in the literature where the graph-based approach is applied to geometry data coding, it shows great potential. In fact, the graph-based approach is part of the state-of-the-art on attribute point cloud coding (as it will be seen when reviewing graph-based coding solutions). Hence, the coding solution proposed in this work wonderfully fills the existing gap by adopting this powerful signal processing tool to compress point cloud geometry information.

Attribute Data Coding

The focus here is on the coding of the attributes available for each cloud point. Most of the solutions available in the literature target the coding of color attributes, notably one single color per cloud point. Three main classes of solutions appear in the literature:

- **Octree-based**: The octree-based approach previously outlined for geometry coding can also be employed for color coding. There are several octree-based color coding schemes, notably: statistical average, i.e. the color of the points inside a voxel is represented by their average; color map, i.e. the octree is mapped into an image so that standard image codecs can be applied; and color differences, i.e. the residual between each point color and the color average of the corresponding voxel is coded.
- **Patch-based**: The patch-based approach for color coding is based on color maps which are obtained using the same process introduced earlier for geometry, i.e. partitioning and parametrization. As the color maps are computed analogously to the height and occupancy maps, they are taken as 2D images and thus standard image codecs can be applied.
- **Graph-based**: The graph-based approach for color coding is similar to the approach for geometry

coding, described above. The main difference is that the graph transform learned is applied to the color information residing on the graph vertices and not to its coordinates.

2.7.1. Octree-based Coding: the Point Cloud Library Coding Solution

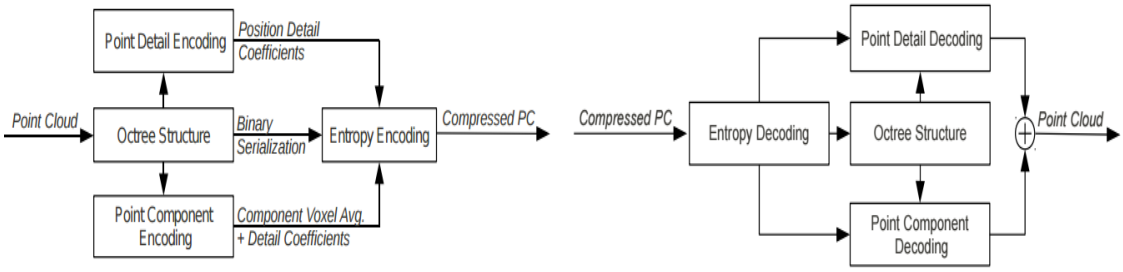
Context and Objective

The solution proposed in [69] is octree-based and was designed to achieve real-time compression of both geometric and attribute data of dynamic point clouds for network streaming, i.e. real-time transmission. This solution has been implemented in the popular PCL and can handle unorganized point cloud data of arbitrary and varying size and density. Thus, this coding solution is suitable for many types of sensors and acquisition strategies.

The objective of this solution is to reach high compression efficiency by exploiting both the spatial and temporal correlations in point cloud data, for geometry and attribute (color) data, using a lossy compression approach.

Technical Approach and Coding Architecture

The encoder and decoder architectures are illustrated in Figure 21. At the encoder, all points of the point cloud are organized in an octree structure and the binary serialization of this structure is created. The relative position of each point inside each octree leaf octant is encoded in the point detail encoding module. To encode the point cloud geometry, it is only necessary to combine the binary serialization of the octree structure and the point detail data. The additional point attributes such as color are also encoded, namely by encoding the voxel component (i.e. attribute) average and the corresponding differential detail coefficients. The final module in the encoding architecture is entropy coding, which processes each of the three different types of data separately and multiplexes them into the final output bit stream, thus obtaining the compressed point cloud stream to transmit to the decoder. At the decoder, an inverse process relative to the encoding process is used. Thus, the decoder is able to recover the points and respective attributes with some error regarding the original point cloud, thus obtaining the decoded point cloud.



(a) Encoder (b) Decoder

Figure 21: (a) Encoding and (b) decoding architectures *[69]*.

The various modules in the encoder architecture are presented next:

- **Octree Structure:** In this module, an octree structure is created, which is basically a tree structure with some depth, where each node can have eight children nodes (i.e. forming a branch) or none (implying this is a leaf node/octant). The octree structure can be serialized for transmission, i.e. a

binary representation of the octree structure can be obtained. The octree depth (i.e. number of octree levels) defines the number of possible divisions in the structure and can be limited either directly (i.e. with a coding parameter) or indirectly by specifying the leaf octant resolution, this means the *octree resolution*. Figure 22 (a) contains a binary tree example illustrating a root node, 4 leaf nodes and an octree depth of 3. The Octree Structure representation module can be organized into two sequential steps, as follows:

1. **Octree creation:** To perform the octree decomposition, the bounding box that includes all the cloud points is recursively divided into eight octants, each represented by a unique bounding box in the 3D space. The main task of the octree decomposition is to determine the correct leaf octant for each cloud point. The first step is to divide the root node into eight octants, i.e. thus creating child nodes, to determine to which octree octant a cloud point belongs. Afterwards, it is necessary to check if the target leaf octant resolution already corresponds to the resolution of the current octant. If not, the creation of additional child nodes in the tree continues until the octant resolution is equal to the target leaf octant resolution. If no further subdivisions should be made, the corresponding octant is called *leaf octant* and the point is added to the list of points associated to that leaf octant. The process continues so that every cloud point is added to the octree and belongs to a leaf octant. In Figure 22 (a), six octree structures for the *Stanford Bunny* point cloud, with different octree depths, are shown where each leaf octant contains one or more cloud points. Notice that only leaf octants with cloud points are created.

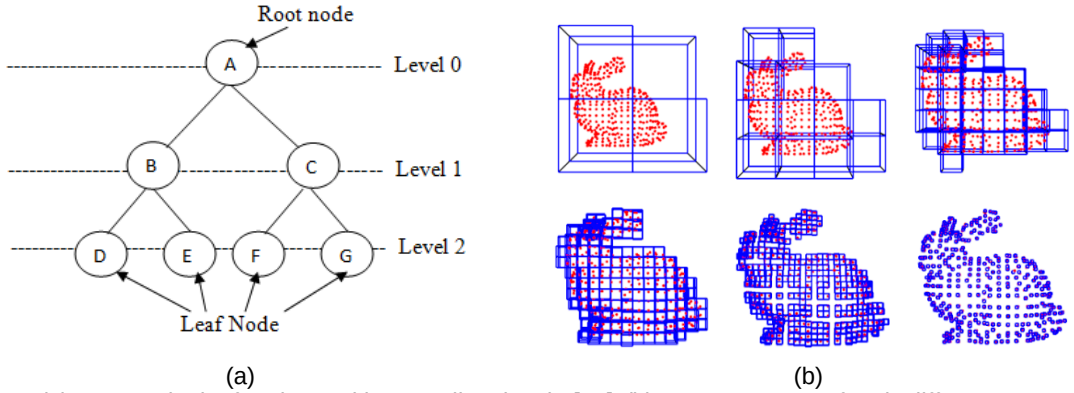
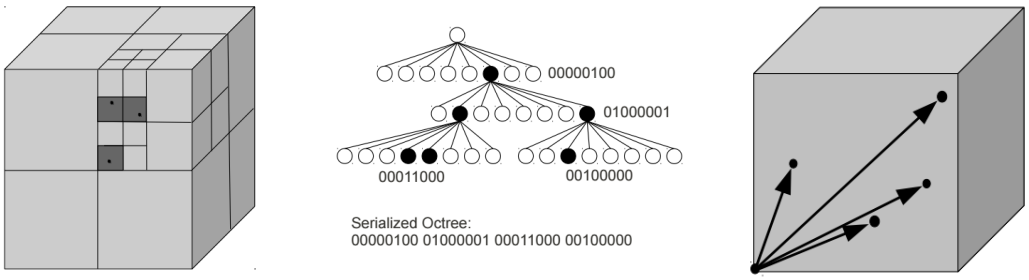


Figure 22: (a) Root node, leaf nodes and intermediate levels *[70]*; (b) Octant structure for six different octree depths *[69]*.

2. **Octree Binary Serialization:** The objective of this step is to create a byte stream that represents the octree structure obtained in Step 1, previously outlined. In this case, every branch of the octree is represented by a single byte (i.e. eight bits correspond to the eight octants), the so-called *child node configuration*, which indicates if the octant is further divided or not and how. The encoded byte stream is obtained by traversing the tree in breadth-first order, meaning that the serialization starts at the root node and progresses by moving first to the neighbors in the same octree level before moving after to the next octree level nodes. The output is every child node configuration byte associated to each tree node. An example of the relation between an octree and an encoded bit stream is given in Figure 23 (a) and (b). The serialized bit stream has to convey some additional parameters such as the root node bounding box volume and the octree depth, so that it is possible to fully reconstruct the point cloud.

- **Point Detail Encoding:** To improve the precision of the point cloud without increasing the octree depth (i.e. without using smaller leaf octants, thus increasing the octree resolution), an additional tool called *point detail encoding* is proposed. In the context of this technique, the point precision of the point cloud is defined as the maximum difference, i.e. the error, between each decoded point location and the corresponding original point location. This tool can avoid increasing the computational complexity when very small size leaf octants are created and coded (i.e. large octrees with many levels); in this case, every leaf octant may contain more than one point, which position is coded using a reference point within the corresponding leaf octant. In case this point detail information is not present, all the points belonging to one particular leaf octant would be represented by the center of that octant, thus only allowing the precision of the point cloud to be, at best, the precision of the octree leaf octants. In this case, the number of decoded points will be smaller than the number of points in the original cloud. The point detail information is computed as follows:

  1. **Octant Local Point Location:** For each occupied leaf octant, the distances between the leaf octant origin (i.e. the corner with smallest *xyz* coordinates) and each octant local point (i.e. points belonging to that octant) are computed, see Figure 23 (c) for example. This operation computes three distances (i.e. in *x*, *y*, *z*), the so-called *octant local point location*, for each point in the leaf octant, using as reference the leaf octant origin. This process allows increasing the point resolution and, thus, the geometry decoded quality without increasing the octree resolution.



(a) (b) (c)

Figure 23: (a) Octree structure example; (b) Encoding bit stream for the octree structure on the left; (c) Example of octant point location *[69]*.

  2. **Point Detail Coefficients:** Each octant local point location is discretized into a positive integer representation (i.e. point detail coefficients) for posterior encoding. To obtain these point detail coefficients, it is necessary to discretize each point location with respect to the octant it belongs; this discretization depends on the octant size and the target point precision. For example, to achieve a 1 mm point precision, when the leaf octant resolution is 9 mm, the 3 x, y, z distances defining the point location should correspond to an integer value in the interval [0,8]. Note that, for each point, the three distances representing the octant local point location must be independently discretized. Therefore, this module outputs a stream of (integer) point detail coefficients.

- **Point Component Encoding:** The principle of point detail encoding for geometry is also applied to the color attributes. For example, to encode the colors, two steps are followed:

  1. **Average Color Computation:** The average color of all the points belonging to a leaf octant, for each leaf octant in the octree, is computed. Naturally, this step results in an average color value

for each leaf octant in the octree structure.

2. **Color Detail Coefficients:** Color differences for each point are computed using the average color value previously obtained as reference, analogously to the point detail differences for geometry coding. This results in the attribute (component) detail coefficients, i.e. the distances from each point color to the corresponding average color for the corresponding octant leaf.

This module outputs the average color for each leaf octant complemented with the residual data (distance from each point color to the average) for each cloud point in each leaf octant.

- **Entropy Encoding:** The main goal of this module is to create a bit stream exploiting the statistics of all the data (output by the previous modules) that needs to be transmitted to the decoder. This entropy encoder takes into account the specific (very likely non-uniform) symbol frequencies for the three data streams to be entropy coded. In this solution, the entropy coder is a range coder [71], which is a variant of an arithmetic coder. Besides the entropy encoded bit stream, the corresponding symbol frequency tables are transmitted so that it is possible for the decoder to properly decode the three coded data streams, i.e. binary serialization, point component stream and point detail coefficients stream.

The proposed coding solution also exploits the temporal correlation in dynamic point clouds by using a novel double buffered octree scheme that enables the comparison of the point clouds captured at successive time instances, thus exploiting their temporal redundancy. While the first point cloud frame is Intra coded as described above, the next point cloud frames are differentially encoded (i.e. using Inter coding). If the bounding box of the point cloud increases between adjacent frames, it is necessary to either provide additional information or to perform Intra frame coding again. Naturally, it is possible to perform Intra frame coding after a certain number of point cloud frames are Inter coded to provide random access and address synchronization requirements.

Performance Evaluation

To assess the performance of the proposed codec, a set of point clouds, each 15 seconds long (i.e. dynamic content), are used. In [69], only geometry compression results are provided, notably using Intra and differential (Inter) coding.

Table 2 shows the geometry compression performance in terms of bytes per point and compression ratios for different point precisions, when using Intra and Inter coding for the set of 15-seconds point clouds, both without using point detail encoding (implying that the point resolution is the same as the octree resolution). For Intra coding (‘static octree’ in Table 2), the frames are compressed independently, while for Inter coding (‘differential octree’ in Table 2) the point cloud frames are differentially coded with the exception of the first point cloud frame which is Intra coded. The Intra coding results show compression ratios from 1:7 for the higher point precisions to 1:33 for the lower point precisions. For the higher point precisions, the Inter coding compression ratios have a gain regarding Intra coding of approximately 40% (i.e. from 1:7 to 1:11) while, for the lower point precisions, the gains in percentage are not so high. Moreover, as the compressed data size decreases, the impact of the fixed size frequency tables in the expended rate increases as a higher percentage of the rate is used for the tables, thus contributing to reduce the overall efficiency.

Table 2: Comparison between Intra (static) and Inter (differential) coding approaches for different precisions *[69]*.

Compression Method	Precision	Bytes / Point	Compression
uncompressed	∞	12 bytes	**1:1**
static octree	0.5 mm	1.73 bytes	**1:7**
	1 mm	1.34 bytes	**1:9**
	3 mm	0.75 bytes	**1:16**
	5 mm	0.48 bytes	**1:25**
	7 mm	0.37 bytes	**1:32**
	9 mm	0.36 bytes	**1:33**
differential octree	0.5 mm	1.14 bytes	**1:11**
	1 mm	0.87 bytes	**1:14**
	3 mm	0.63 bytes	**1:19**
	5 mm	0.45 bytes	**1:27**
	7 mm	0.43 bytes	**1:28**
	9 mm	0.30 bytes	**1:40**

Another coding solution recently proposed in [72] extends the solution here reviewed [69] by using standard image codecs for the color attributes, notably Joint Photographic Experts Group (JPEG), achieving improved coding efficiency. To create the image data, the octree is traversed in depth-first order, i.e. starting from the root node, the octree is traversed as deep as possible along each branch before backtracking [73], so that the color attributes are mapped into a regular image grid. The PCL codec performance presented above is now complemented with some additional experimental results as the solution in [72] uses the same Intra coding engine for the point cloud geometry. The datasets used are RGB realistic tele-immersive reconstructed data from [74] with approximately 300K points.

Figure 24 shows the geometry compression efficiency in terms of PSNR versus the geometry rate for various qualities of the decoded point cloud, this means for various levels of geometry precision, this means different LoD and, thus, different octree depths. To assess the geometry quality, the metric defined in (8), Section 2.6, has been used as a function of the bitrate in Kbytes per frame. As expected, both the bitrate and quality increase when higher LoD are coded.

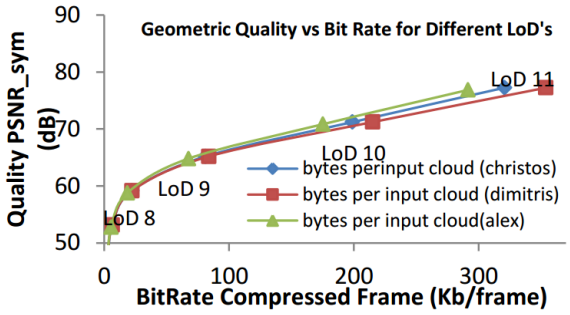
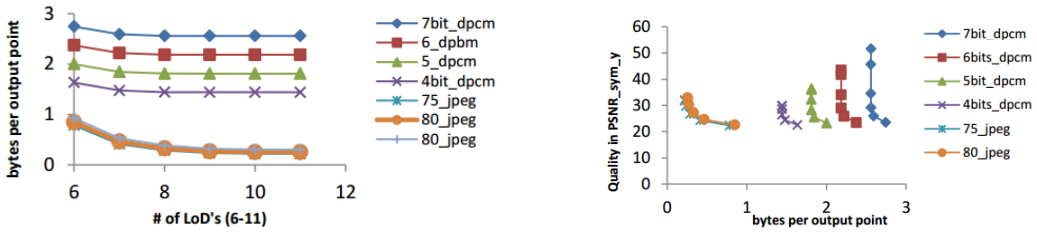


Figure 24: Geometry compression efficiency: quality versus rate for various LoD *[72]*.

Figure 25 (a) shows the rate spent in terms of bytes per output/decoded point for different LoD. Moreover, Figure 25 (b) shows the color compression efficiency in terms of luminance PSNR versus bytes per output point. Since the color compression is being assessed, the rates correspond only to color coding, i.e. without geometry information. To assess the color quality, the color quality metric defined in (10), Section 2.6, has been used. In Figure 25 (b), the color coding schemes compared are the differential pulse-code modulation (DPCM) PCL solution reviewed above with different quantization and several JPEG codings with different quality factors (highest quality corresponds to 100). For approximately the same quality, the JPEG solutions provide a much lower bitrate, i.e. up to 10 times lower, when compared to the PCL DCPM solution.



(a) (b)

Figure 25: (a) Rate versus geometry precision[1]; (b) Color quality versus rate *[72]*.

The reviewed point cloud coding solution strengths and weaknesses are outlined as follows:

1. **Strengths:** It handles unorganized point cloud data of arbitrary and varying size while exploiting the spatial and temporal redundancies; it has low memory requirements and it is not computationally expensive (encoding can be performed in real-time).
2. **Weaknesses:** The color encoding solution does not exploit the correlation between the color attributes of points in adjacent octants; as there is no rate control mechanism, it is not possible to define a specific data rate to be used.

In the following, a color coding solution exploiting the color correlation between adjacent octree points will be reviewed.

2.7.2. Point Cloud Attribute Compression with Graph Transform

Context and Objective

In the context of point cloud compression, a realistic decoded point cloud can only be obtained if also color attributes are coded and transmitted. Compared to traditional images, these attribute components are unstructured since they are associated to different locations in the 3D space, thus making their compression process more challenging.

The objective of the solution proposed in [68] is to efficiently compress point cloud attributes, for static models, using a graph transform. The main goal is to reduce the bitrate associated to the alternative octree-based schemes as the one reviewed in 2.7.1. Using a graph-based representation and treating the attributes as signals over the graph, a graph transform can be used to obtain a higher compression efficiency when compared to octree-based compression.

Technical Approach and Coding Architecture

The encoding architecture is illustrated in Figure 26. First, it is assumed that the point cloud is represented using an octree-based structure, like the one described in Section 2.7.1, so that it can be easily partitioned into the so-called *octree blocks*, which are subsets of the original octree structure representing a small branch of the octree at a certain depth level. The geometry data coding process is performed separately from the attribute color coding process, as illustrated in Figure 26. To code the color attributes, it is necessary to create the graph structure, which means to connect (i.e. defining the edges in the graph) points from nearby occupied voxels (i.e. the vertices in the graph). For each of the edges, a

[1] In the figure caption, there are two 80_jpeg curves. The last one (i.e. grey curve corresponding to 80_jpeg) should be 85_jpeg as the curve is slightly above the other two (i.e. 75_jpeg and 80_jpeg) in terms of bpp.

weight is assigned expressing the similarity between the corresponding edge attributes. After obtaining a graph-based representation with the edges, vertices and associated weights, it is possible to compute the transform coefficients using a graph transform. These transform coefficients are after quantized to efficiently represent the three YUV color components available for each cloud point. The quantization process is independently applied to each of the three vectors of coefficients, thus resulting into three quantized vectors of coefficients. These coefficients are then entropy encoded to obtain the final output bitstream. The decoder receives both the color and geometry data streams to decode first the geometry and after the subsequent color, exploiting the previous knowledge on the decoded geometry information.

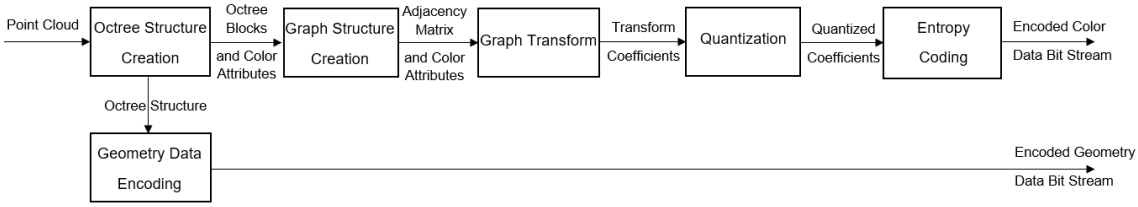
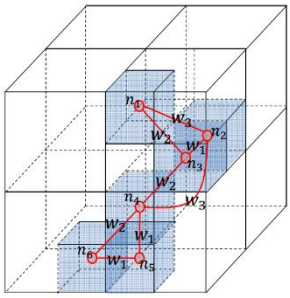


Figure 26: Graph transform based encoding architecture.

The various encoding modules are now briefly reviewed:

- **Octree Structure Creation:** In this module, the point cloud is organized as an octree with a single point per leaf octant, represented by the center of the voxel/octant. For this specific solution, the octree structure will be partitioned into several octree blocks, each containing K×K×K adjacent voxels where N are occupied (i.e. voxels containing a point). Naturally, it is possible to obtain the original octree structure by simply concatenating the octree blocks.

- **Graph Structure Creation:** To obtain the graph structure for a given block, nearby occupied points are connected, thus forming the graph edges. A weight is then assigned to each edge as illustrated in Figure 27 (a) for an octree block with 4×4×4 voxels and N=6 occupied voxels. The weights on the graph are set as inversely proportional to the geometrical distances between the voxels center. Moreover, to mathematically describe the graph structure, an adjacency matrix can be defined with the weights set as illustrated in Figure 27 (b).



(a)

$$\mathbf{A} = \begin{pmatrix} 0 & w_3 & w_2 & 0 & 0 & 0 \\ w_3 & 0 & w_1 & w_3 & 0 & 0 \\ w_2 & w_1 & 0 & w_2 & 0 & 0 \\ 0 & w_3 & w_2 & 0 & w_1 & w_2 \\ 0 & 0 & 0 & w_1 & 0 & w_1 \\ 0 & 0 & 0 & w_2 & w_1 & 0 \end{pmatrix}$$

(b)

Figure 27: (a) Graph structure for an octree block of 4×4×4 voxels, with $n_1 \ldots n_6$ being nodes/points of the graph and $w_1 \ldots w_3$ weight values defined for the edges. The blue shaded cubes represent the occupied voxels in the octree block *[68]*; (b) Adjacency matrix for the graph structure defined in (a) *[68]*.

- **Graph Transform:** As the neighboring elements on a graph structure are usually highly correlated, a so-called *graph transform* is applied to decorrelate the data associated to these elements. In this module, the following steps are used to obtain the graph transform for a certain graph structure with

N occupied voxels:

1. **Graph Laplacian Matrix Creation:** Given an adjacency matrix A describing the graph structure, it is possible to construct a graph Laplacian matrix Q (12) by:

$$Q = \alpha(D - A) \tag{12}$$

where α is a scalar related to the graph attributes variance and D is the diagonal matrix computed as:

$$D = diag(d_1, \dots, d_N) \tag{13}$$

$$d_i = \sum_j a_{ij} \tag{14}$$

where a_{ij} are elements of the adjacency matrix A corresponding to the weights assigned to each edge.

2. **Eigenvalue Decomposition:** The eigenvalue decomposition of Q is then performed. The eigenvector matrix and the diagonal matrix containing the eigenvalues are decomposed according to:

$$Q = \Phi\Lambda\Phi^{-1} \tag{15}$$

$$\Lambda = diag(\lambda_1, \dots, \lambda_N) \tag{16}$$

where Φ represents the eigenvector matrix of Q, which is used as a transform for the color attributes residing on the graph points. The diagonal matrix Λ contains the eigenvalues of the Laplacian matrix Q. Each of these eigenvalues, which are associated to an eigenvector, represents the inverse of the expected variance of the respective associated transform coefficient. Φ is the N×N matrix graph transform, which contains the transform defining coefficients.

3. **Color Components Transform Coefficients Computation:** The transform coefficients for each color component, for the *N* occupied voxels, will be computed as three different N×1 vectors (denoted as y, u, v). For example, the transform coefficients for the y component are calculated as:

$$f_y = \Phi y \tag{17}$$

Afterwards, it is necessary compute the remaining color components (U and V) coefficients. To do so, the same expression as above is used by replacing y with u and v.

- **Quantization:** The quantization process is applied to each of three coefficients vectors obtained from the previous module. The quantized coefficients (f_{y_q}), are obtained using a quantization step size denoted as q, as follows:

$$f_{y\_q} = round\left(\frac{f_y}{q}\right) \tag{18}$$

The quantized coefficients vector representing the color component Y for an octree block is then entropy coded. As for the previous module, it is necessary to quantize the coefficients for the three

color components by using the same formula while replacing f_y, with f_u and f_v. The process is done for each of the point cloud blocks, until all octree blocks (representing the full octree structure and, therefore, the full point cloud) are entropy encoded, thus creating the output stream.

- **Entropy Encoding:** After the graph transform is applied to a given block, a DC term and a set of corresponding AC terms are associated to each sub-graph in the block. For a given octree block, *m* disconnected subsets of points correspond to *m* sub-graphs for that block. Therefore, many DC terms may be associated to a block with many block sub-graphs. The DC and AC terms are treated differently during entropy coding as follows:

  1. **AC coefficients:** The AC coefficients are encoded using an arithmetic encoder based on an adaptive Laplacian probability distribution, under the assumption that the f_{y_q} coefficients have zero mean. The probability tables are obtained from the probability density function that follows an adaptive Laplacian distribution, which has a diversity parameter (variance of the Laplacian) that is updated every time a new AC term is encoded.

  2. **DC coefficients:** The first step to encode a DC term is to subtract an appropriate estimate based on the previously coded DC term, thus resulting in a difference signal. This procedure takes into account information, i.e. value and number of connected voxels, regarding the DC term under analysis and the last coded DC coefficient. The probability tables are computed using the difference signal probability density function that also follows a Laplacian distribution and has a diversity parameter (representing the variance) that is updated every time a new DC term is encoded.

Performance Evaluation

To assess the performance of the proposed coding solution, six different point clouds associated to human upper bodies, see two of them in Figure 28, containing approximately 500-750k points with color attributes, were used. All the results to be presented correspond to averages for the six point clouds.

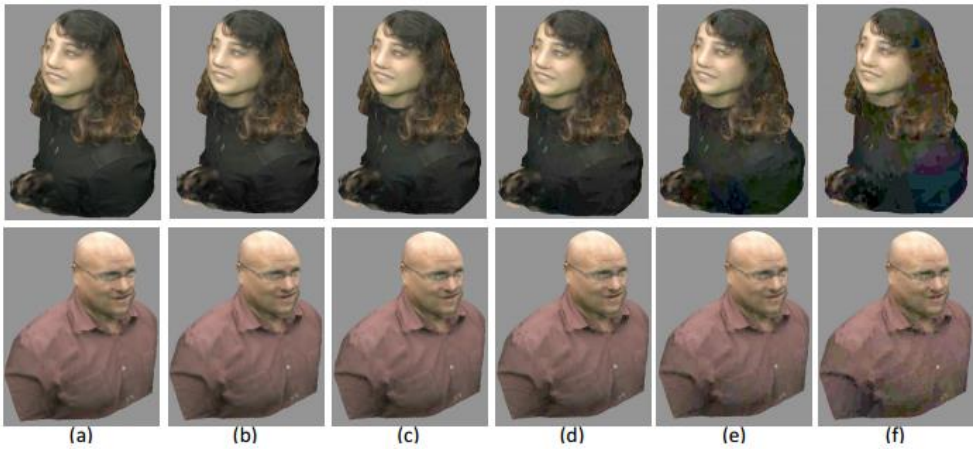


Figure 28: Results for two point clouds compressed using the graph transform based codec, with K=8 and five different quantization steps q. (a) original point cloud; (b) q=4; (c) q=8; (d) q=16; (e) q=32; (f) q=64 *[68]*.

Table 3 shows a comparison, in terms of quality/distortion of the color data (Y, U and V) for different bitrates, between the proposed codec and the PCL codec described in the previous section. The bpv (bits per vertex) represents the bitrate, in this case computed as the average number of bits spent by vertex/voxel. The distortion of the various color components is expressed with the Signal to Noise Ratio

(SNR), a quality metric similar to the one defined in (10), Section 2.6. The only difference regarding PSNR is that SNR does not always represent the signal with its peak value, i.e. 255 when 8-bit value are used. Instead of the peak value in the numerator, the SNR uses the Average Signal Power (ASP), for each of the color components, computed as follows for the luminance:

$$ASP_y = \left(\frac{1}{K} \sum_{v\_d \in Vdec} y(v\_d(i)) \right)^2 \tag{19}$$

The bitrate for the graph transform based codec is always much smaller, for similar color components distortion values, which means that the PCL codec is significantly outperformed.

Table 3: RD performance for the graph transform based codec and the PCL codec for color data *[68]*.

bitrate(bpv)	SNR_Y(dB)	SNR_U(dB)	SNR_V (dB)
PCL encoder			
14.15	52.0	54.6	54.5
11.34	44.3	51.2	50.8
8.40	38.0	47.0	46.3
5.70	32.1	41.9	41.4
3.30	26.9	37.7	36.9
1.48	23.0	34.0	33.3
Proposed Graph Transform encoder			
5.36	52.1	54.7	54.6
1.74	44.3	51.4	50.8
0.36	38.1	47.1	46.4
0.16	28.4	38.2	38.2

Figure 29 shows a rate-distortion (RD) performance comparison between the proposed codec and a coding solution using a 1D Discrete Cosine Transform (DCT) as the transform. The 1D DCT is a special case of the graph transform where the graph is constructed as a 1D chain and only 1D neighboring connections are made [75]. Therefore, when the graph connects points in 3D space as in the graph-based approach proposed, the compression performance improves compared to the case where a 1D DCT approach is used as less correlation is exploited. The RD performance experimental results are illustrated in Figure 29 for four different octree block sizes: level 5 has K=16, level 6 has K=8, level 7 has K=4 and level 8 has K=2. Therefore, the charts in Figure 29 correspond to different octree block sizes for the two coding methods, i.e. graph transform and 1D-DCT, e.g. DCT5 corresponds to the 1D-DCT method for level 5, this means K=16. The graph transform compression performance outperforms the 1D DCT compression performance, notably the PSNR gain goes from 1 to 3 dB for the same bitrate. Naturally, using larger block sizes leads to better compression performance as more voxels (color attributes) are decorrelated together in each block.

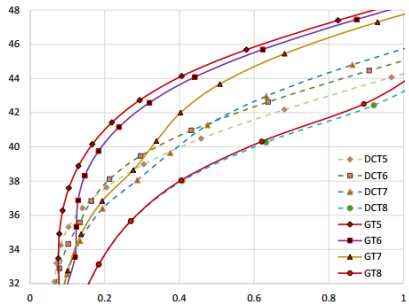


Figure 29: RD performance for the graph based and the 1D DCT transform coding solutions for different octree blocks levels *[68]*.

The reviewed point cloud coding solution strengths and weaknesses are outlined as follows:

3. **Strengths:** Significantly outperforms traditional octree-based methods in terms of color attribute coding; for unstructured point clouds, there is no overhead incurred in the graph construction since

the geometry information is encoded prior to the color data.

4. **Weaknesses:** For sparse point clouds, the graph structure may have many isolated sub-graphs, thus making the graph transform less efficient; intra block prediction, i.e. prediction among points belonging to different blocks, is not considered in this solution.

Next, a color coding solution that extends this one to obtain better compression efficiency when dealing with sparse point clouds is presented.

2.7.3. Graph Transform Attribute Compression for Sparse Point Clouds

Context and Objective

In the context of point cloud attribute compression, the solution proposed in [76] is an extension of the previously presented graph transform based codec for sparsely populated static point clouds. The main objective of the codec presented here is to efficiently compress color information in sparse-populated blocks using a graph transform [76]. To do so, the authors propose two alternative methods: i) block compaction so that each graph-transformed block has only one associated DC coefficient; and ii) K-Nearest Neighbors (K-NN) driven graph structure creation to obtain more efficient graphs, i.e. with more points connected.

Technical Approach and Coding Architecture

The proposed encoding architecture is illustrated in Figure 30. As the focus of this work lies on attribute coding, it is assumed here that the geometry is previously coded, independently from the attributes. The novel module in this solution, when compared to the solution reviewed in Section 2.7.2, is the graph structure creation, which corresponds to the second module in the encoding architecture. Similarly to the solution reviewed in Section 2.7.2, the first module is the creation of the octree structure so that it can be easily partitioned into blocks. In this coding solution, there are two alternative methods proposed to create the graph structure: *blocks compaction* and *K-NN neighbors*. This coding solution will be assessed using the two proposed alternative methods independently. After using one of the two proposed graph structure creation methods, the graph transform is applied and the transform coefficients are quantized and subsequent entropy coded, as outlined in Figure 30 and Section 2.7.2.

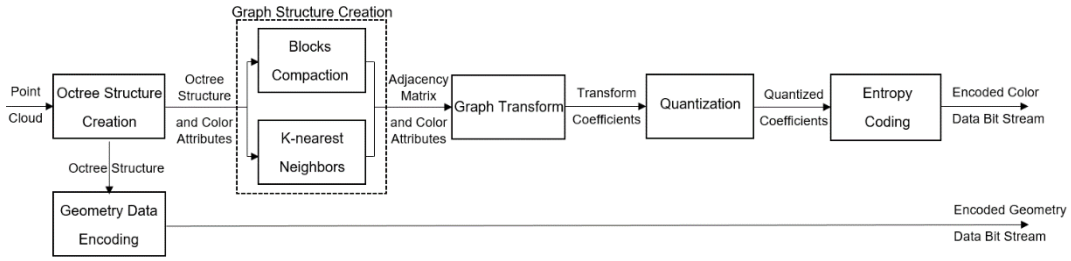


Figure 30: Encoding architecture for sparse point clouds using graph transforms, where the novel coding tools correspond to the *Graph Structure Creation* module.

The novel graph structure creation methods are now briefly reviewed:

- **Blocks Compaction:** This method guarantees that all points, and respective color values, are compacted towards one corner of the block, thus ensuring that a single graph structure is created for each block. Therefore, when the graph transform is applied to the created graph structure, it only

produces one DC coefficient, along with a single set of AC coefficients. The process is illustrated in Figure 31 and resembles the shifting procedure used in the so-called *shape-adaptive DCT* (SA-DCT) [77]. Firstly, for a given K×K×K block, all cloud points (i.e. voxels that are marked as occupied) are shifted to the top, i.e. along the *z* axis, of the K×K×1 sub-block. The second step shifts all cloud points to the left, i.e. along the *x* axis, of the obtained sub-block. The final step recursively repeats the first and second steps along the third dimension, i.e. along the *y* axis of the block, so that all occupied voxels of the K×K×K block are compacted towards the block corner. This procedure must be performed for all K×K×K blocks. The compacting procedure involves a trade-off between the compression efficiency and the exploitation of the spatial correlation among the color of the points. A graph structure similar to the one created for the solution reviewed in Section 2.7.2 is created where edges between neighboring nodes only one unit away are formed, so that the graph transform can be applied; in the following, this type of graph will be called *unmodified* to distinguish from those created with the novel tools proposed in this section. To undo this compacting process at the decoder side, it is necessary to guarantee that each decoded attribute is assigned to its corresponding 3D position. Unlike SA-DCT, there is no need to compute and signal the boundary shape of the attributes as the "shape" of the point cloud is defined by the voxel occupancy of the blocks, the point cloud geometry, previously independently coded.

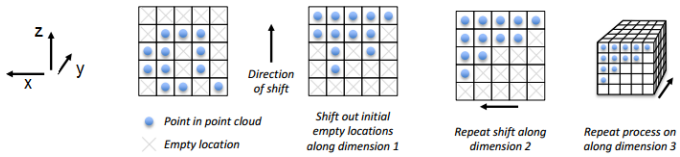


Figure 31: Example of the blocks compaction procedure *[76]*.

- **K-NN Neighbors:** The graph structure created with the K-NN neighbors method allows connections between more distant points, which when applied to a sparse block critically contribute to reduce the number of sub-graphs. Figure 32 illustrates the procedure for constructing the K-NN graph structure with K=3. In the K-NN neighbors method, each point in the block must be connected to at least K neighbors. Notice that, to guarantee the K neighbors connectivity for all points, it may happen that after the procedure is complete and duplicated edges are pruned, some points are connected to more than K neighbors, as illustrated in Figure 32 for P7 (which has 5 associated edges while only 3 were requested). This may occur because the nearest neighbor computation is done independently for each point, e.g. while P2 has P7 as nearest neighbor, P7 doesn't consider P2 as a nearest neighbor (for K=3, P7 has closer points). Therefore, P7 is a nearest neighbor for P2 and P5, in addition to its own nearest neighbors. The weight associated to each graph edge is a function of a distance, *d*, e.g. $e^{-\frac{d}{2\sigma^2}}$, where *d* is the distance between the two connected 3D points and σ is the variance of relevant point characteristics. Unlike the blocks compaction method outlined above, the K-NN neighbors method does not ensure that a single graph per block results, i.e. several sub-graphs may result.

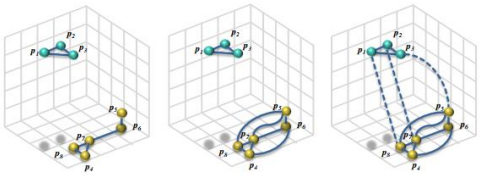


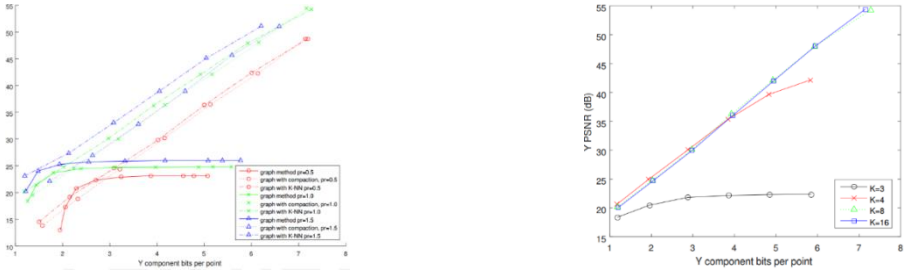
Figure 32: Example of the K-NN process evolution for K=3 *[76]*.

Performance Evaluation

To assess the performance of the proposed solution, a specific point cloud has been used, in this case *Statue_Klimt_PointCloud.ply* described in [78], with 3000 cloud points. Each 3D cloud point has a 3D coordinate and a corresponding RGB value. For the experiments reported, the RGB color attributes were converted to the YUV color space and the 8-bit luminance value, *Y*, was the single adopted target attribute to code. The octree decomposition used a target resolution of 0.0081, which ensured that each leaf octant contains only one cloud point. Afterwards, the point cloud is divided into blocks with a specific partition resolution, *pr*. This parameter describes the octree blocks resolution and is inversely proportional to the point precision. For the experiments, three different values for this parameter were used: *pr* = 0.5, 1.0 and 1.5. For all the *pr* values, the Intra-block prediction defined in [79] was implemented, which exploits directional similarities in the Intra prediction process such as in the Advanced Video Coding (AVC) and High Efficiency Video Coding (HEVC) video coding standards. The main idea behind this tool is to perform block predictions in the *xyz* directions, when adjacent blocks are available at the encoder, to choose the prediction direction yielding the smallest distortion. If the block under analysis yields the lowest distortion, there is always the option to code it without any prediction.

The results presented here intend to compare the compression performance of three graph transform based methods, this means *unmodified*, *blocks compaction* and *K-NN neighbors*. The K-NN neighbors performance is assessed for several values of K.

Figure 33 (a) and (b) illustrate two different compression performances, both in terms of luminance PSNR vs. the luminance rate in bytes per point. Figure 33 (a) assesses the performance for three *pr* values and the three different graph transform based coding methods: unmodified, K-NN neighbors with $K = 8$ and blocks compaction. The results show that the unmodified graph transform based method PSNR saturates as the rate increases. On the other hand, for the K-NN neighbors and blocks compaction methods, the PSNR continues to improve as the rate increases. In this experiment, the K-NN neighbors method with $K = 8$ only produces one DC coefficient as it happens for the blocks compaction method. Further experiments should be made to compare the blocks compaction and K-NN neighbors performances when the latest produces more than one DC coefficient. Figure 33 (b) represents the K-NN neighbors compression performance for different values of K while using a fixed value of *pr* = 1.0. For $K = 3$, the compression performance is rather poor. For the lower rates, $K = 4$ performs rather well, whereas $K \geq 8$ shows the best compression performance for all the other rates.



(a) (b)

Figure 33: (a) RD performance for the three graph transform based methods; (b) K-NN neighbors RD performance for various values of K and *pr* = 1.0 *[76]*.

The common strengths and independent weaknesses of the two novel methods reviewed above are:

1. **Common Strengths:** Outperform the previously presented coding solutions in terms of compression efficiency.
2. **Blocks Compaction Weaknesses:** Spatial correlation between neighboring points may not be exploited with the blocks compaction procedure; if the original block, i.e. without performing the blocks compaction method, can be represented with one single graph, then this method does not bring any compression efficiency gain.
3. **K-NN Neighbors Weaknesses:** Does not guarantee to encompass all points in a given block in one single graph, i.e. does not ensure coding one single DC coefficient.

In the next section, a patch-based coding solution for compressing both attributes and geometry information will be reviewed.

2.7.4. Patch-based Attribute and Geometry Compression

Context and Objective

The coding solution proposed in [80] features two important characteristics that are relevant for many applications: the capability to code incrementally acquired data, and the capability to just decode a subset of the full point cloud. The proposed technical solution is able to exploit the spatial correlation, in terms of color and geometry, between cloud points, using a lossy patch-based approach.

Technical Approach and Coding Architecture

The encoding architecture is illustrated in Figure 34. The main idea of this coding solution is the creation of patches, so that standard image codecs can be applied to code them. In this context, a patch is understood as a 2D parametrization of a voxel, basically corresponding to the mapping of the voxel points attributes onto a 2D plane.

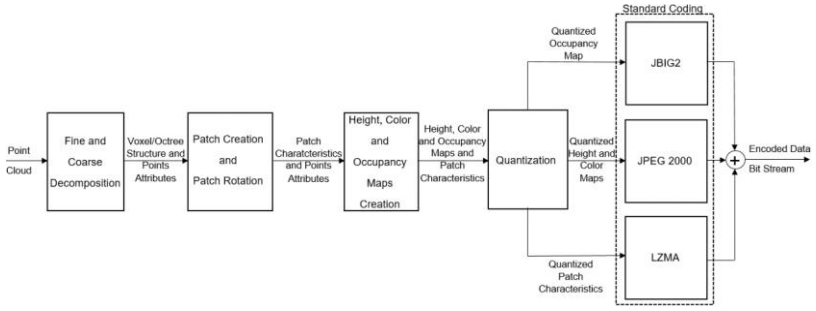


Figure 34: Patch-based encoding architecture.

The various modules in the encoder architecture are presented next:

- **Fine and Coarse Decomposition** First, the point cloud data is split into *compression chunks*, i.e. voxels of a user-specified size to be independently compressed, so that a coarse decomposition of the point cloud is created. Such coarse decomposition makes the coding solution able to support incrementally acquired data and also partial decoding; in practice, the compression chunks are basically smaller point clouds that derive from the original point cloud. By having this compression chunks, it is possible: i) to code them independently, thus offering partial decoding, and; ii) to handle incrementally acquired data, e.g. in an online scenario, it is possible to incrementally receive data from a robot navigating a given area, with the new data successively creating new compression chunks. Afterwards, a fine decomposition of the coarse compression chunks is performed in two different ways: with a voxel grid or with an octree (similarly to Section 2.7.1) which should represent the point cloud. For the fine decomposition into a voxel grid, a voxel resolution parameter has to be chosen; for the octree, a leaf octant resolution has to be chosen.
- **Patch Creation and Points Rotation:** Independently of the fine decomposition method used, each voxel created in the previous module includes a set of cloud points called a *cluster*. For each cluster, the point coordinates and associated colors are parametrized into two dimensions, thus creating a so-called *patch*. Each patch is defined by its characteristics, notably its position, orientation and size. To achieve efficient compression, it is important to obtain as much as possible *well behaved patches*, this means for example with edges parallel to the voxel coordinate system as the following wavelet bases and the oriented DCT are also oriented along the voxel coordinate system. The voxel coordinate system is defined by the centroid of the voxel and the voxel edges orientation. The patch position is defined through the geometrical center of mass of the points cluster while the patch size is directly proportional to the voxel size associated to the cluster. The next patch characteristic is its orientation: All points inside a voxel undergo a rotation, to project them into a rectangular shaped plane. This will allow projecting all points into this plane and just use the height of each point with respect to this plane to express its position. To estimate the rotation, the following procedure is applied:
  1. **Patch Normal Computation:** For the computation of the patch normal, n, a principle component analysis (PCA) of the corresponding cluster is performed.
  2. **Patch Orientation Computation:** First, the patch position is chosen as the patch coordinate system origin. Next, three different orientation vectors, corresponding to the patch orientation, are computed considering the normal obtained from the previous step.
  3. **Patch Rotation:** To rotate the patch, it is necessary to rotate the voxel coordinate system so that it becomes aligned with the patch coordinate system; the cluster of points should be now ready for mapping. The patch coordinate system is obtained by combining the patch orientation and position computed in the two previous steps.
- **Height, Occupancy and Color Maps Creation:** For each patch created, it is necessary to create maps, which are 2D images with a fixed resolution, with the target to represent the points attributes. At this stage, height and occupancy maps are created to represent the geometry, and color maps to represent the color information. As the voxels may vary in size and the maps resolution is fixed, this coding approach is data-adaptive; while sparsely populated regions are represented by larger voxels, densely populated regions are represented with smaller voxels. The height and occupancy maps are

created as follows: for each 3D point, the corresponding height (i.e. *z* coordinate) is stored in the appropriate 2D position (i.e. *x*, *y* coordinates) of the height map and the corresponding 2D position of the occupancy map is defined as occupied. Color maps are computed in a similar way to the height maps. Note that, when dealing with the height and color maps, it is possible that several points are projected onto the same pixel of the 2D patch. In this case, the value of that pixel is set to the average value of all points mapping to that same pixel.

- **Quantization:** The quantization process is applied to the height and color maps values which can assume non-integer values after their creation. Both maps are quantized to 8 bits so that any standard image codec can be applied. For the patch characteristics, the size and the three different orientation values are quantized to 8 bits, whereas the patch position 3D coordinates are quantized to 16 bits. The quantization step size is applied independently for each compression chunk. To perform the quantization of the height and color maps values, two steps are performed:
  1. **Minimum and Maximum Computation:** For each map being quantized, both the minimum value *m* and the maximum value *M* are computed.
  2. **Quantization:** A given value *v* is quantized as $v' = \frac{v-m}{M-m}$. Afterwards, all maps of the same type, e.g. all height maps, are combined into a single large map to be coded.
- **Standard Coding:** As the quantized maps computed in the previous step correspond to 2D matrices that have the same structure as 2D images (with values with a precise bit depth), it is possible to apply any standard image codec. Three different image coding standards have been used, depending on the type of data to be coded, as follows:
  1. **JPEG and JPEG 2000:** For the height and color maps, both JPEG and JPEG 2000 are used. In terms of compression efficiency, JPEG 2000 achieves the best performance while using the Cohen-Daubechies-Feauveau 9/7 wavelet transform [81].
  2. **JBIG2:** Binary occupancy maps are coded with the lossless JBIG2 algorithm [82].
  3. **LZMA:** The patch information is coded with the lossless Lempel-Ziv-Markov chain algorithm (LZMA) [83] and an extension of the LZ77 algorithm [84].

  Finally, the elementary output encoded bit streams from each codec above are multiplexed into a joint encoded data bit stream and transmitted to the decoder.

Performance Evaluation

To assess the performance of the proposed coding solution, two different point clouds were compressed: i) the *David* statue dataset containing 28.2 million points; and ii) the *fr1/room* dataset [85] containing 315.5 million points and a corresponding raw input stream size of 2.7 GB. The *fr1/room* dataset has been obtained from a sequence of RGB-D images, which were processed and fused into a single point cloud [86].

Figure 35 shows the geometry RD performance using the PSNR as geometry quality metric for the *David* statue dataset considering performance results both for coding solutions available in the literature [87] [88] [89] [90] and the proposed patch-based coding solution. The geometry distortion metric used in this assessment is similar to the one outlined in Section 2.6 with the only difference that the authors chose the bounding box diagonal as the geometry peak value instead of BB_{width} used in (8). For a fixed bitrate,

the proposed solution outperforms all the alternative solutions in terms of quality, thus offering the best geometry RD performance.

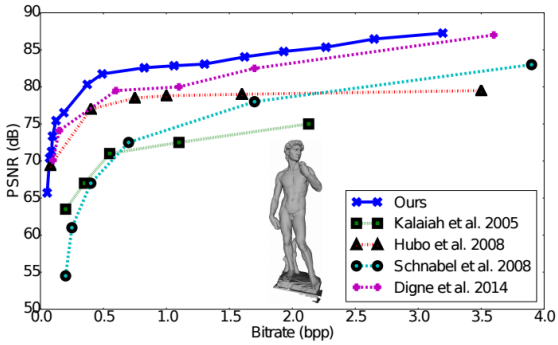


Figure 35: Geometry RD performance *[80]*.

Table 4 shows a geometry and color compression performance comparison between several alternative coding solutions for the *fr1/room* dataset, notably: i) the proposed coding solution using both voxel and octree decompositions; ii) Octomap [91], which is similar to the solution reviewed in Section 2.7.1 in terms of Intra coding; iii) Sparse Coded Surface Models (SCSM) solution [92], which is based on a decomposition of the point cloud into a set of local surface patches using sparse coding to compactly describe the attributes and geometry of each patch; and iv) Hierarchical Sparse Coded Surface Models (HSCSM) [86], which is an hierarchical extension of iii) leading to more compact models by using LoD capabilities. The performance assessment is made in terms of: i) compressed file size, s_c; ii) color and geometry distortion, respectively, $RMSE_{rgb}$ and $RMSE_D$; iii) encoding time, t_c, and decoding time, t_d, in seconds and; iv) compression speed, p/t_c, in points per second. For the geometry distortion metric, the authors used the following expression:

$$RMSE_D(V_{or}, V_{dec}) = \sqrt{0.5 d_{rms}(V_{or}, V_{dec}) + 0.5 d_{rms}(V_{dec}, V_{or})} \tag{20}$$

where d_{rms} is the root mean square distance (5) introduced in Section 2.6.

The color distortion metric considers all color components and is computed as:

$$RSE_{rgb}(V_{or}, V_{dec}) = \sqrt{0.5 MSE_{rgb}(V_{or}, V_{dec}) + 0.5 MSE_{rgb}(V_{dec}, V_{c}} \tag{21}$$

where $MSE_{rgb}(V_{or}, V_{dec})$ is the RGB mean square error which can be obtained after computing each component mean square error. $MSE_r(V_{or}, V_{dec})$ is the R component mean squared error computed as:

$$MSE_r(V_{or}, V_{dec}) = \frac{1}{K} \sum_{\substack{v_o \in Vor \\ v_d \in Vdec}} ||r(v_o(i)) - r(v_d(k))||^2 \tag{22}$$

and $r(v_o(i))$ is the *i*th R component value of the original point cloud, $r(v_d(k))$ is the *k*th R component value of the decoded point cloud and *K* is the number of points in the original point cloud. $MSE_g(V_{or}, V_{dec})$ and $MSE_b(V_{or}, V_{dec})$ are computed is a similar way. The results obtained show that the proposed coding solution obtains smaller compressed file sizes, for both the octree and voxel decomposition methods, when compared to previous solutions, for approximately the same color and geometry distortion, respectively $RMSE_{rgb}$ and $RMSE_D$.

Table 4: Geometry and color compression performance for several coding solutions *[80]*.

Method	s_c	RMSE (D/RGB)	t_c	t_d	p/t_c
OctoMap	45 MB	0.006 m / 39.6	120 s	n.a.	$2.63 \cdot 10^6$
SCSM	8.1 MB	0.005 m / 29.9	1860 s	n.a.	$0.17 \cdot 10^6$
HSCSM	7.2 MB	0.005 m / 30.0	2160 s	n.a.	$0.14 \cdot 10^6$
Ours, Voxel, JPEG	7.07 MB	0.005 m / 26.6	43 s	2.7 s	$7.33 \cdot 10^6$
Ours, Octree, JPEG 2000	2.97 MB	0.005 m / 27.2	71.5 s	4.7 s	$4.41 \cdot 10^6$

The proposed coding solution strengths and weaknesses are outlined as follows:

1. **Strengths:** It is able to code point clouds with arbitrary and varying size; supports local compression and decompression; supports parallel coding; has low complexity allowing real-time compression; it also outperforms the compression efficiency of the octree solution presented in Section 2.7.1.
2. **Weaknesses:** It does not exploit the spatial correlation between adjacent patches.

After the literature review made in this chapter, the next chapter will proposed a novel point cloud geometry coding solution based both on octrees and graph transforms.

Chapter 3

3. Proposing a Graph Transform-Based Point Cloud Geometry Compression Solution

This chapter describes the proposed point cloud geometry coding solution, the so-called *graph transform geometry coding* (GTGC), which consists in a two-layer scalable approach that adopts octree-based base layer (BL) coding and graph transform-based enhancement layer (EL) coding. To improve the compression efficiency, some appropriate Inter-layer prediction tools are proposed. To introduce the GTGC, this section starts by presenting the encoder and decoder architectures and their corresponding walkthroughs, followed by the detailed presentation of the GTGC main tools.

3.1. Architecture and Walkthrough

The GTGC encoding and decoding architectures are illustrated in Figure 36 and Figure 37, respectively. The focus of this solution lies on geometry coding for static point clouds; the main idea is to combine graph transform and Inter-layer prediction tools with the well-established point cloud octree-based structure, used in the PCL codec, to reach more efficient point cloud geometry coding. The general coding approach is the following:

1. The point cloud is coded in two layers, the BL and EL. In the BL, a version of the original point cloud with a coarser level of detail is obtained and coded. In the EL, the original/full point cloud is coded.
2. While the BL uses conventional PCL coding, the EL uses a novel coding approach that exploits graph-based transform and Inter-layer prediction.
3. Based on the BL, a so-called *upsampled base layer* (U-BL) cloud is created, which is a coarse approximation (without a fine level of detail) of the original point cloud. This U-BL point cloud is used, both at the encoder and the decoder, for the graph transform learning step.
4. To learn and apply the graph transform, it is necessary to create clusters for both the EL and U-BL point clouds.
5. Using the EL and U-BL clusters, a spatial Inter-layer prediction process is applied, so that only appropriate residuals are transformed and coded.
6. For each U-BL cluster, a graph transform is learned and applied to the previously obtained residuals, obtaining the transform coefficients.
7. Finally, some transform coefficients are selected, quantized and entropy coded, thus creating the EL point cloud bitstream.

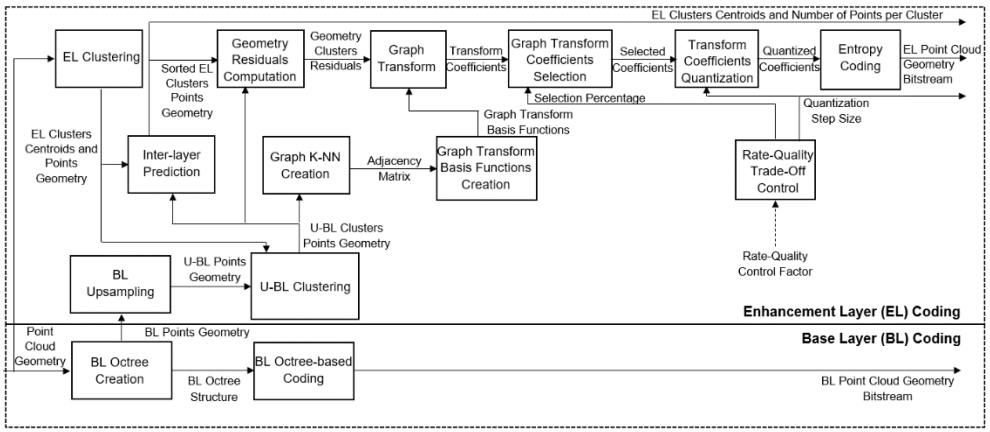


Figure 36: GTGC encoding architecture.

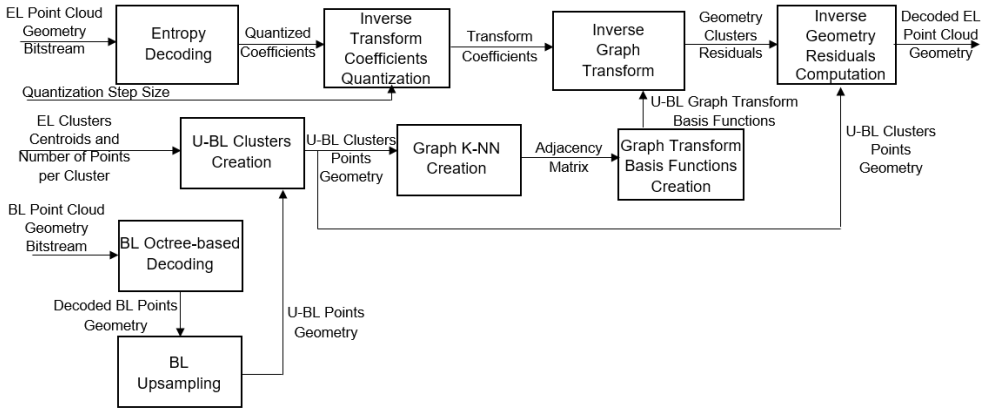


Figure 37: GTGC decoding architecture.

The main objectives for the modules in the GTGC solution are now outlined:

BL Coding

- **BL Octree Creation:** This module creates the two layers that will be used for coding: one with a coarse level of detail, denominated BL; and another with the full point cloud, denominated EL. This division is made by creating an octree structure for the BL up to a certain number of points. The octree is iteratively extended until the pre-defined ratio between the number of points in the BL and EL clouds, the so-called *BL-EL number of points ratio*, is achieved.
- **BL Octree-based Coding:** This module codes the previously created BL octree structure which is sent to the decoder, thus making the BL point cloud available at both the encoder and decoder sides; in this case, the PCL codec is used for BL point cloud coding, thus providing compatibility with PCL in the BL.

EL Coding

- **BL Upsampling:** In this module, an U-BL point cloud is created given the decoded BL point cloud. The main target is to upsample the BL point cloud to obtain a higher density point cloud with approximately the same density, i.e. the same number of points, as the full EL point cloud. The output of this step is the U-BL cloud, which will be used after to create the U-BL clusters.
- **EL Clustering:** The main objective of this module is to create the clusters for the EL cloud, this means

the full cloud, to which different graph transforms learned from the BL cloud will be applied. This step creates clusters with a K-Means procedure for a given input point cloud with a user-specified number of clusters and a similar (although not necessarily exactly the same) number of points in each cluster. The output of this step is the set of EL clusters point coordinates and corresponding cluster centroids. Both the centroids and the clusters dimensions must be sent to the decoder, thus making it possible to perform both the U-BL clustering and apply the graph transform related modules at the decoder.

- **U-BL Clustering:** The purpose of this module is to create U-BL clusters, as similar as possible to the EL clusters created in the previous module, using the U-BL cloud previously created. Note that the decoder must be able to reproduce this same clustering to make it possible to learn the exact same graph transform basis functions at both the encoder and decoder sides. This approach ensures that, for each EL cluster centroid received from the EL clustering module, a corresponding U-BL cluster is created with equal dimension, i.e. number of points, thus creating a rather good match between the EL and U-BL clusters.

- **Inter-layer Prediction:** The main target of this module is to guarantee that the residuals energy, i.e. the EL to U-BL point-to-point distance, is minimized. As each EL and U-BL clusters are stored as 3D points vectors, it is necessary to sort the EL vector so that the correspondence distances between 3D points with the same index are minimized. To do so, an Inter-layer point matching algorithm is applied to the previously created U-BL and EL clusters; this step is key to reduce the residuals for the following module. Notice that, as the decoder does not have both cloud layers available, this Inter-layer point matching procedure is only done at the encoder side. This Inter-layer prediction module was designed to provide benefits in terms of RD performance, notably it is very critical that the U-BL and EL clusters are as similar as possible. Thus, it is important to create a high similarity between the U-BL and EL layers clusters with a suitable BL upsampling procedure. The output of this module are the sorted EL vectors per cluster, thus representing each EL cluster geometry in terms of correspondences to the U-BL, the so-called *Inter-layer point matching correspondences*.

- **Geometry Residuals Computation:** The main goal of this module is to compute the geometry residuals for each pair of EL and U-BL clusters, so that the overall compression efficiency is improved; in practice, this step should lower the energy of the signal to be after transformed.

- **Graph K-NN Creation:** In this module, a graph K-NN structure is created at both the encoder and decoder for each U-BL cluster. In the graph structure, each graph vertex corresponds to a point with certain 3D coordinates. Moreover, it is necessary to ensure that each graph vertex is connected to at least K nearest neighbors, where K is obtained using a percentage of the cluster number of points. Also, a suitable weight expression shall be adopted for the adjacency matrix creation. This module outputs the adjacency matrix for each U-BL cluster.

- **Graph Transform Basis Functions Creation:** In this module, the graph transform basis functions are obtained based on the graph previously created for each U-BL cluster. To do so, given the adjacency matrix for each cluster, the Laplacian matrix and the corresponding eigen-decomposition are computed, thus obtaining the so-called *graph transform basis functions*. These graph transform basis functions will be used in the graph transform module applied to the geometry residuals.

- **Graph Transform:** This module applies the graph transform, defined by the graph transform basis functions learned in the previous module, to each of the components of the *xyz* vectors corresponding to the residuals for each cluster. The input to this module are the geometry clusters residuals, to be transformed, and the graph transform basis functions computed in the previous module; the output are the transform coefficients for the residuals of each cluster.
- **Graph Transform Coefficients Selection:** The objective of this module is to select the most relevant graph transform coefficients, zeroing the least relevant ones. The relevance of the coefficients is mainly based on their energy. This selection procedure is based on the ordered graph transform basis functions previously learned.
- **Transform Coefficients Quantization:** This module applies a dead-zone uniform quantizer to the previously computed selected coefficients, thus obtaining the quantized coefficients.
- **Rate-Quality Trade-Off Control:** This module controls the rate-quality trade-off for the GTGC solution. To do so, a control parameter is created, the *so-called* rate-quality control factor (RQCF), that controls both the graph transform coefficients selection and the transform coefficients quantization.
- **Entropy Coding:** The last encoder module is the entropy coding, which generates the bitstream to be sent to the decoder side while exploiting the quantized coefficients statistics; here an arithmetic encoder based on a Laplacian distribution has been adopted.

Although the presented architecture only uses two coding layers, it is possible to increase the number of coding layers by recursively predicting each layer based on the previous layer, thus augmenting the potential benefit introduced by the Inter-layer prediction module. As increasing too much the number of layers may penalize the overall compression efficiency, this should only be done following well identified application scenario requirements.

3.2. Main Tools Detailed Description

The purpose of this section is to detail the main coding tools designed and implemented for the GTGC solution. To do so, the various modules introduced in the walkthrough above will be thoroughly described and the respective design decisions motivated.

3.2.1. BL Octree Creation and Coding

The objective of this module is to create and code the BL point cloud using an octree structure. The main control parameter for this module is the *BL-EL number of points ratio*, defined as the target ratio between the number of points in the BL and EL (full) point clouds; in practice, this parameter defines the boundary between the EL and BL point clouds. In this module, the first step is to determine the octree voxel size value which ensures that the desired BL-EL number of points ratio is achieved. To do so, the following steps are performed:

1. **Octree Creation and Division:** Using the EL point cloud geometry, the corresponding octree structure is created starting with a voxel size equal to the bounding box previously obtained, this mean with only one point. This module successively divides the occupied voxels in the octree structure until

the desired BL-EL number of points ratio is obtained assuming that the number of BL points is the number of occupied voxels. For each division, the number of points in the current octree structure, this means the number of BL points, is used to compute the so-called *current number of points ratio* (CNPR) using the following expression:

$$CNPR = \frac{N_{BL\_points\_current}}{N_{EL\_points}} \tag{23}$$

2. **BL-EL Number of Points Ratio Checking:** In this step, the CNPR computed in the previous step is compared with the target BL-EL number of points ratio to determine if the octree division process should continue or stop. There are two options, as follows:

   - If the CNPR is lower than the BL-EL number of points ratio the division process continues as in Step 1 to further divide the octree structure.
   - If the CNPR is larger than the BL-EL number of points ratio, the division process should stop, thus obtaining the octree voxel size defining the BL and the corresponding octree structure.

At this stage, the final step is the encoding and decoding of the BL octree structure created using the octree voxel size previously obtained. In order to do so, the PCL codec is used with the voxel resolution and point precision parameters both equal to the octree voxel size found. This guarantees that the point precision method is not used and the resolution obtained from the BL octree creation algorithm is the same as the decoded BL. Notice that this step ensure the BL cloud, which will be posteriorly used for the graph transform learning, is available at both the encoder and decoder sides. Figure 38 (a) shows a BL cloud using a BL-EL number of points ratio of 20% for the *Egyptian Mask* point cloud.

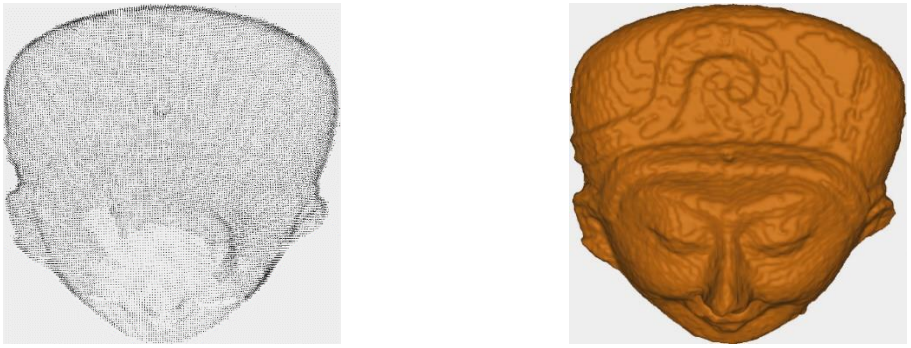
3.2.2. BL Upsampling

The main target for this module is to perform the upsampling of the BL point cloud to match the EL point cloud density. To do so, two steps are followed: i) a surface reconstruction method is adopted to compute a 3D mesh for the BL point cloud; and ii) using the 3D mesh faces, i.e. the mesh triangles, an upsampling algorithm is used to add more points to the BL cloud, that is, to create the U-BL point cloud. It is important to state that the motivation to use an upsampling algorithm emerged from the need to produce an U-BL cloud as uniform as possible, this means with points rather regularly spaced. This implies that there should be no significantly large holes present in the U-BL cloud if they are not present in the EL cloud.

Surface Reconstruction

The method adopted to obtain the 3D mesh is the so-called *Poisson reconstruction*. The idea behind this approach is to use a so-called *indicator function* to determine which points better define the surface, hence making it possible to extract the desired reconstructed surface [93]. The main reason for this choice is that the Poisson reconstruction method creates a smooth, noise-free and watertight surface. To apply this reconstruction method, it is necessary to estimate the normals for the cloud points. Both the normal computation and the Poisson reconstruction were done using the *CloudCompare* software [94]. For the normal computation, the local surface model (for the surface approximation) adopted was the plane (which is robust to noise but weak on approximating edges/corners) and the normals orientation were

obtained using the minimum spanning tree method. Figure 38 (b) shows the BL point cloud (from Figure 38 (a)) Poisson reconstructed 3D Mesh; it is clear that the reconstructed 3D mesh rather accurately represents the BL cloud surface.



(a) (b)

Figure 38: (a) Downsampled *Egyptian Mask* point cloud with 20% of the EL cloud points; (b) Reconstructed surface using the Poisson reconstruction method for the BL *Egyptian Mask* point cloud.

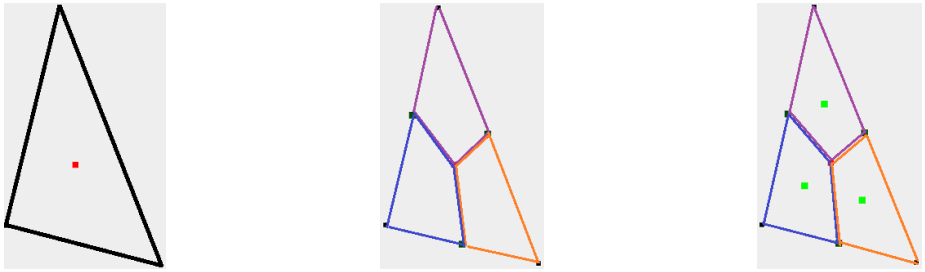
At this stage, it is possible to use the reconstructed 3D mesh as input for the upsampling algorithm described below to create the U-BL point cloud.

BL Upsampling

The purpose of the BL upsampling algorithm is to create the U-BL cloud using the 3D mesh triangles obtained from the Poisson reconstruction method introduced above. Considering this objective, the following process is proposed:

1. **BL Points Adding:** First, all points in the BL point cloud are added to the U-BL cloud.
2. **3D Mesh Triangles Counting:** The objective of this step is to determine the number of points to be added to the U-BL cloud, so that the U-BL cloud matches the EL cloud density, i.e. number of points in the EL cloud. To do so, the number of triangles in the 3D mesh is computed. If the number of triangles present in the 3D mesh is equal or higher than the desired number of points to be added, only steps 3 and 6 are applied. Otherwise, all steps are applied.
3. **3D Mesh Triangles Centroid Computation:** For each triangle, this step computes its corresponding centroid using the coordinates of the triangle vertices. The centroid is represented by the arithmetic mean of the triangle vertices 3D coordinates. Figure 39 (a) shows, for a given triangle, the three triangle vertices and the corresponding centroid.
4. **3D Mesh Triangle Mean Points Computation:** For each triangle, this step computes the mean point of each edge connecting two vertices of a triangle. Similar to the centroid computation described in the previous step, the arithmetic mean is computed using the coordinates of the two vertices.
5. **Polygon Creation and Centroids Computation:** The objective of this step is to define new points inside the 3D mesh triangle to add more points to the U-BL point cloud. This step uses the mean points computed in the previous step and the 3D mesh triangle original vertices to create three new polygons. Each polygon is represented using a mean point, the two closest vertices of the selected mean point and the triangle centroid, see Figure 39 (b). After, the centroid for each polygon just created is computed, thus creating three new points to be added to the U-BL cloud. To obtain the centroid, the arithmetic mean is computed using the corresponding polygon vertex coordinates. Figure

39 (c) illustrates the three smaller polygons and the respective centroids.

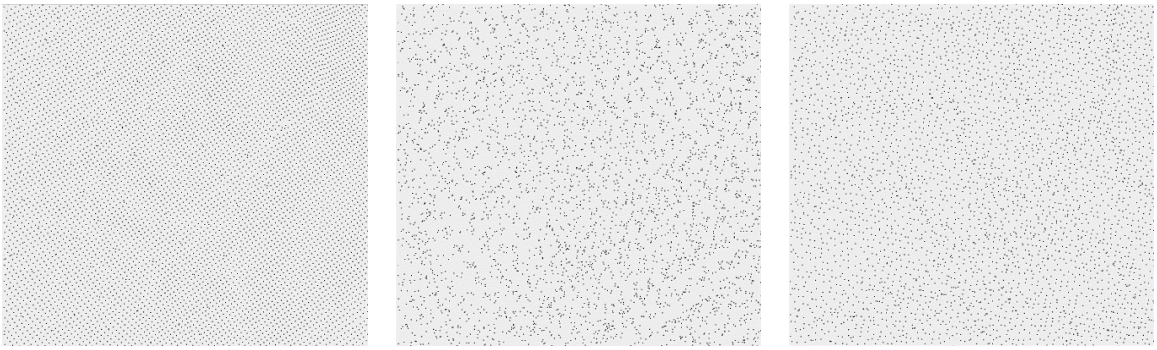


(a) (b) (c)

Figure 39: (a) Triangle vertices and corresponding centroid; (b) Smaller polygons obtained using the proposed method (c) Smaller polygons centroids.

6. **Selecting and Adding Points:** In this final step, the centroids defined either at step 3 (if the number of triangles in the mesh is equal or higher than the number of points desired for the U-BL cloud) or at step 5 (if the number of triangles in the mesh is lower than the number of points desired for the U-BL cloud) are added to the U-BL cloud. The points are added one by one; the first points to be added are those with a higher distance to the closest point already in the U-BL cloud. After adding a point to the U-BL, the distances are computed once again to determine the next point to be added; therefore, this procedure is iterative and the algorithm stopping criterion is the target number of points in the U-BL cloud; if the number of points in the U-BL cloud is already equal to the number of points in the EL, the algorithm stops. Notice that this point selection process guarantees a rather uniform U-BL cloud.

Figure 40 a) shows an EL cloud area while Figure 40 b) and Figure 40 c) show the U-BL clouds obtained with the Moving Least Squares (MLS) upsampling method available in the PCL, and the proposed upsampling method. The cloud used for this example is the *Egyptian Mask* cloud and the BL-EL number of points ratio used is 10%. As it may be observed, the main goals for the upsampling process are achieved with the proposed method as the corresponding U-BL cloud is more uniform and represents more closely the desired EL cloud density. Moreover, when compared to the MLS upsampling method, the proposed method provides an U-BL with less holes (areas without any points).



(a) (b) (c)

Figure 40: (a) EL point cloud. (b) MLS upsampling method. (c) Proposed upsampling method.

3.2.3. EL Clustering

The purpose of this module is to perform a suitable clustering of the EL point cloud by dividing its points into a user-specified number of clusters; note that the clusters obtained shall have a similar (although not necessarily precisely the same) number of points and represent a local volume of the point cloud. To limit the time complexity of the graph transform learning, i.e. to have a computable eigen

decomposition, the size of the clusters cannot be very high (at maximum a few hundred of points). The implementation of this algorithm was adopted from [95].

In the GTGC, the popular K-Means algorithm, which attempts to cluster the points into sets with equal variance, is used. The K-Means algorithm divides the number of points in the EL cloud (N_{points_EL}), the so-called *samples*, into an user-specified number of disjoint clusters ($N_{clusters}$), where each of these clusters is defined by the corresponding samples mean, the so-called *cluster centroid*; typically, these clusters centroids are not points that belong to the EL cloud.

The K-Means algorithm main objective is to select cluster centroids that minimize the *inertia* of the clustering, *J* , i.e. the sum of distances of each point to its nearest cluster centroid:

$$J = \sum_{j=1}^{N_{clusters}} \sum_{n \in S_j} \left\| x_n - \mu_j \right\|^2 \tag{24}$$

where x_n is the nth point in cluster S_j, μ_j is the centroid of the points in cluster S_j and $N_{clusters\_enh}$ is the user-specified number of disjoint clusters. Notice that the inner sum in (24) does not have an upper limit as the clusters do not have the same dimensions. To achieve the desired minimization of the clustering inertia, *J*, the K-Means algorithm performs the following three steps:

1. **Initialization:** In this step, the positions of the clusters' centroids are initialized. There are several methods to do this centroids initialization, such as the random initialization and the *kmeans++* initialization [96]. In this Thesis, the *kmeans++* method was adopted as it ensures that the initial centroids are equally distant from each other; this type of initialization is relevant to make sure that the algorithm converges to a suitable division and consumes less time.
2. **Points Assignment:** After having the initial centroids (or a set of centroids), each EL point is assigned to the corresponding cluster, notably each EL point is added to the corresponding closest cluster. The closest cluster corresponds to the cluster which centroid is closer to the EL point being processed.
3. **Clusters Centroids Computation:** After having assigned all the EL points to the corresponding clusters (step 2), the clusters centroids positions are computed again; for each cluster, the new cluster centroid corresponds to the mean value of the 3D points assigned to that cluster in the previous step. Afterwards, the distance between the old and new centroids is computed to determine if the algorithm has converged; the K-Means algorithm converges when the average distance between the old and new centroids is small, in this case smaller than a pre-defined threshold. Thus, if the average distance computed is lower than a pre-defined threshold, the algorithm stops; otherwise, the algorithm goes to Step 2.

After performing the K-Means clustering, each point in the EL point cloud is associated to a cluster and each EL cluster is defined by the corresponding centroid; the clusters centroids will be used in the following module. It is important to note that each cluster obtained using the K-Means algorithm will have its own size; that is, the K-Means does not guarantee that the clusters have the same number of points. Figure 41 shows the clusters obtained after running the K-Means algorithm for three different point clouds: (a) *Egyptian Mask* cloud with 1362 clusters; (b) *Loot* cloud with 4026 clusters; and (c) *Longdress* cloud

with 4289 clusters. To obtain these clusters values, the average number of points in each cluster was set to 200. The clusters obtained appear to have similar areas and a hexagonal shape. Naturally, the number of clusters used is directly related to the number of points in each cluster and its corresponding area. Therefore, the K-Means parameter $N_{clusters}$ will impact the GTGC solution RD performance.

(a) (b) (c)

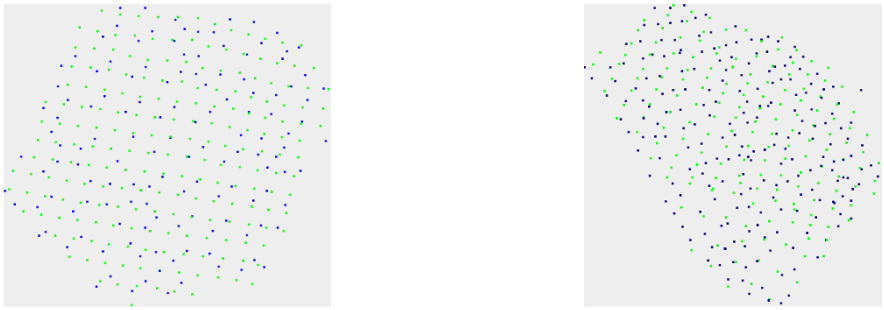
Figure 41: K-Means clustering for: (a) *Egyptian Mask* point cloud; (b) *Loot* point cloud; (c) *Longdress* point cloud.

It is important to notice that, from now on, the EL clusters will be independently processed until the quantization module is reached. Also, the EL centroids must be sent to the decoder; the estimation for this overhead was made considering 30 bits per centroid, corresponding to 10 bits per component.

3.2.4. U-BL Clusters Creation

The main goal of this module is to create the U-BL clusters based on the U-BL point cloud and the EL cluster centroids. This procedure intents to create U-BL clusters similar in geometry to the corresponding EL clusters. Moreover, the U-BL and the corresponding EL clusters must have the same number of points as the graph transform basis functions will be learned from the U-BL cluster and applied to the corresponding residuals; thus, the EL, U-BL and their corresponding residuals must have the same sizes so that the Inter-layer prediction and graph transform steps can be performed. To achieve the goals previously outlined, the following steps are performed:

1. **U-BL Points Assignment**: Using the EL clusters centroid, a K-Means points assignment step (described in the previous section) is applied to the U-BL point cloud. This means that each point in the U-BL point cloud is assigned to the closest EL centroid. Figure 42 depicts two U-BL clusters (blue points) obtained using this step, the *so-called* U-BL temporary clusters, and the corresponding EL clusters (green points).



(a) (b)

Figure 42: EL cluster (green) and corresponding U-BL temporary cluster (blue) for: (a) lower and (b) higher number of points in the U-BL temporary cluster.

From the results obtained, it was possible to conclude that the created U-BL temporary clusters have similarities in terms of point locations when compared to their corresponding EL clusters.

2. **U-BL Temporary Clusters Resampling:** As the previous step does not ensure that the U-BL temporary cluster has the same number of points as the corresponding EL cluster, it is necessary to perform some resampling as follows:

   - **U-BL Points Removal:** In case the U-BL temporary cluster has a number of points higher than the corresponding EL cluster, some points from that U-BL temporary cluster have to be removed. To select the points to be discarded, the distance between each temporary U-BL cluster point and the corresponding three closest U-BL cluster points is computed and stored in a distance vector; afterwards, the points associated with the lowest distance in this vector are progressively discarded. There is a natural restriction in this process preventing that the BL cloud points present in the U-BL temporary cluster being processed are discarded.

   - **U-BL Points Addition:** In case the U-BL temporary cluster has a number of points lower than the corresponding EL cluster, some points have to be added to the U-BL temporary cluster. To do so, the K-Means algorithm is used to iteratively divide the U-BL temporary cluster; thus obtaining temporary sub-clusters. The temporary sub-clusters centroids, which are associated to the previously obtained sub-clusters, are added to the U-BL temporary cluster. Note that the points to be added are not present in the U-BL cloud. The algorithm obtains the number of sub-clusters based on the current number of points present in the U-BL temporary cluster; notably using the following expression:

$$N_{sub-clusters} = ceil(\frac{N_{points\_U-BL\_temp}}{8}) \quad (25)$$

     where $N_{sub-clusters}$ is the number of sub-clusters, $ceil(\dots)$ is the ceil function that rounds upward the input value, i.e. returns the lowest integer that is higher than the input value, and $N_{points\_U-BL\_temp}$ is the current number of points present in the U-BL temporary cluster. The denominator in (25) defines the target number of points in the sub-clusters; several values were assessed and the one adopted, i.e. eight points, showed the best results.

The resampling algorithm stopping criterion is the U-BL temporary cluster number of points; the addition of U-BL points stops when it reaches the dimension desired, i.e. the corresponding EL cluster dimension. Figure 43 (a) and (b) show a U-BL cluster obtained using the resampling with addition of points, from two different viewpoints. Figure 43 (c) and (d) show a U-BL cluster obtained using the resampling with removal of points, also from two different viewpoints. Here, the U-BL clusters are shown as red points, the corresponding EL clusters as green points and the corresponding EL cluster centroid as a black point. The BL point cloud was obtained using a *BL-EL points ratio* of 15%. From the examples, it is possible to observe that the proposed technique for the U-BL creation can rather efficiently represent the surface of the EL clusters; this is clear for all examples presented in Figure 43. Naturally, there are still some error in terms of the points location which correspond to the geometry residuals that will be coded in the EL, using appropriate transform and quantization tools. Note that the U-BL clusters created here (at both encoder and decoder) will be posteriorly used in the graph transform related module to learn the graph transform basis functions.

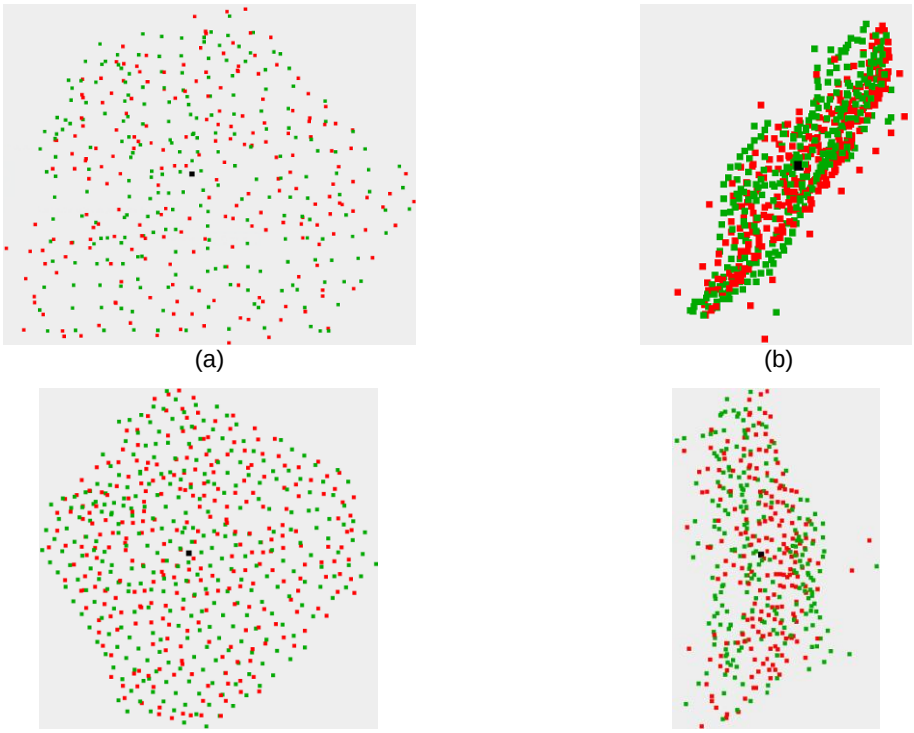


Figure 43: Examples of EL clusters (green points), the corresponding EL centroid (black point) and the corresponding U-BL clusters (red points) with the BL-EL number of points ratio of 15%: (a) and (b) Points addition step for different viewpoints; (c) and (d) Points removal step for different viewpoints.

3.2.5. Inter-layer Prediction

The objective of the Inter-layer prediction module is to exploit the redundancy between layers, this means to avoid the EL points to be coded directly and instead coding a residual distance between each EL point and some corresponding point in the U-BL. To obtain high compression efficiency, the residual energy, i.e. the EL to U-BL points distance, should be minimized. As each cluster is stored as a vector of 3D points and each point has an index associated, the distance between points with the same index in the EL and the U-BL vectors must be minimized. For each cluster, the proposed Inter-layer prediction algorithm performs the following steps:

1. **Sorted EL Vector Creation:** First, an empty vector with the same size of the EL cluster, called *sorted EL vector*, is created; this vector will be used to store the EL cluster points sorted according to some criteria.
2. **Sorted EL Vector Point Addition:** For each point in the U-BL cluster previously obtained, starting at index zero, the closest point in the corresponding EL cluster is found and added to the sorted EL vector.
3. **Original EL Point Removal:** The point included in the sorted EL vector in the previous step must be removed (after) from the EL vector so that there are no duplicate points in the sorted EL vector. The algorithm iterates Steps 2 and 3 until all points in the U-BL cluster have been processed. This ensures that all points in the EL cluster have been added to the sorted EL vector as the U-BL and EL clusters should have the same dimensions; the main goal here is that the Inter-layer point matching correspondence distances are minimized.

The output of the algorithm previously detailed is a sorted EL vector, which has the same points as the EL vector but now ordered accordingly to the minimum U-BL vector distance. Figure 44 shows the Inter-layer point matching correspondences between a sorted EL cluster (green points) and its corresponding U-BL cluster (red points). Figure 44 (a) shows the correspondences for a U-BL cloud using 5% of BL-EL number of points ratio while Figure 44 (b) shows the correspondences for an U-BL cloud using 20% of BL-EL number of points ratio.

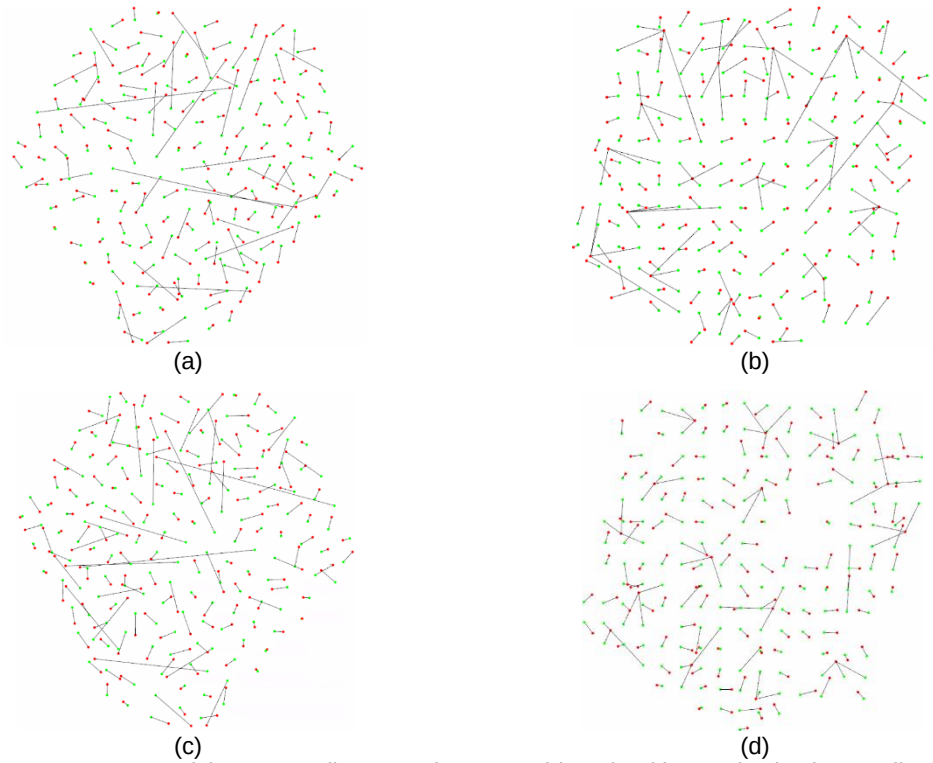


Figure 44: Assessment of the upsampling, Inter-layer matching algorithm and selective EL discarding for: (a) 5% BL-EL number of points ratio with no discarding; (b) 20% BL-EL number of points ratio with no discarding; (c) 5% BL-EL number of points ratio with discarding; and (d) 20% BL-EL number of points ratio with discarding. The EL clusters are shown as green points and the U-BL clusters are shown as red points.

As expected, the distance associated to the Inter-layer point correspondences decreases as the *BL-EL number of points ratio* increases. If the Inter-layer point correspondence distance values are shorter, a lower residual can be computed; thus, when the graph transform is applied, a high level of compactness may be obtained since the U-BL learned basis function were learned with a set of points that are highly correlated with the EL set of points to be coded.

From Figure 44, it is possible to observe that the distance associated to some Inter-layer correspondences is rather large; those large distance correspondences correspond to the last indices added to the sorted EL vector which had less candidates for the correspondences. As these large distance correspondences have a high negative impact on the RD performance, a points discarding method is here proposed that naturally slightly reduces the EL decoded number of points, which by the way more intensively happens when coding with the PCL codec. The PCL also reduces the rate by reducing the number of points to be coded.

Selective EL Points Discarding

The objective of this method is to remove some Inter-layer correspondences of points that are too far

away, i.e. have high correspondence distances. Since the Inter-layer matching algorithm starts by establishing correspondences with small distances, it is expected that the last correspondences to be made have higher distances. The proposed method exploits this fact and works as follows:

1. **Inter-layer Matching Mean Distance Computation:** For each index of the U-BL and EL clusters, the Inter-layer point matching distance is computed and stored onto a distance vector. Afterwards, the mean distance $meanDist_{IL}$ for the vector previously obtained is computed.
2. **EL Points Discarding:** A discarding threshold $D_{threshold}$ parameter is defined based on the mean distance computed in the previous step, e.g. $D_{threshold} = 2 \times meanDist_{IL}$. Then, the last indices of the sorted EL vector are discarded, until the Inter-layer point matching distance does not exceed the $D_{threshold}$ value. Those EL points will not be coded.
3. **U-BL Points Discarding:** Knowing the size of the EL cluster after the EL points discarding procedure, the last corresponding indices of the U-BL vector are also discarded until the same cluster dimension is achieved. The U-BL discarding procedure is replicated at the decoder after receiving the corresponding EL vector size. This step ensures that the dimension of the U-BL vector after the discarding procedure will be the same as the EL vector obtained from Step 2.

Note that this method introduces some distortion in the EL decoded point clouds as some EL cloud points will not be coded. Figure 44 (c) and (d) show the clusters correspondences after applying the proposed EL points discarding method using $D_{threshold} = meanDist_{IL}$ to the initial correspondences presented in, respectively, Figure 44 (a) and (b). As the Inter-layer mean distance is different for different BL-EL point ratios, the resulting number of discarded points is also different. For the cluster presented using a 5% BL-EL point ratio, the discarding procedure removes approximately 10% of the points; with a 20% BL-EL point ratio, the discarding procedure removes approximately 20% of the points using a $D_{threshold} = meanDist_{IL}$. From the examples in Figure 44 (c) and (d), it is possible to conclude that the proposed EL points discarding method allows reducing the larger distance Inter-layer correspondences despite some distortion introduced. Naturally, the EL points discarding parameter $D_{threshold}$ will impact the GTGC solution RD performance.

To estimate the rate necessary to code the size of each cluster, after the discarding procedure is applied, the following expression is used:

$$bitrate_{cluster\_size} = N_{clusters} \times log_2\left(max_{cluster\_size} - min_{cluster\_size}\right) + log_2(min_{cluster_{size}}) \quad (26)$$

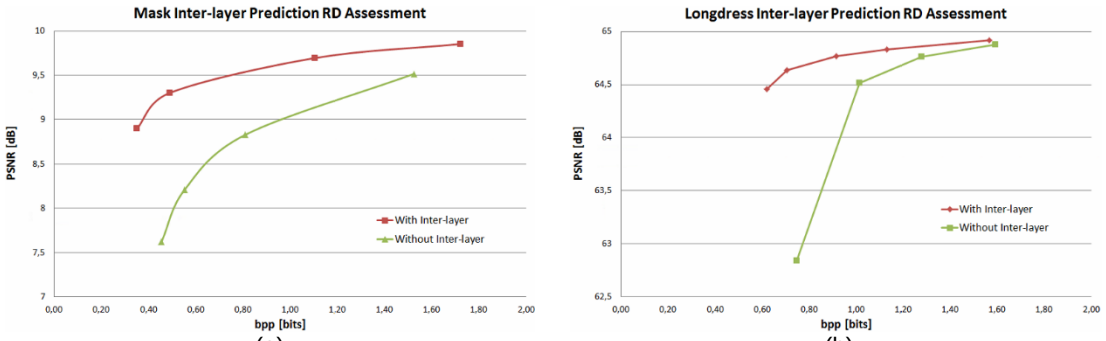
where $bitrate_{cluster\_size}$ is the estimated bitrate overhead for the cluster size, $N_{clusters}$ is the number of clusters and $max_{cluster\_size}$ and $min_{cluster\_size}$ are, respectively, the maximum and minimum number of points present in all the clusters.

3.2.6. Geometry Residuals Computation

The objective of this module is to compute the geometry residuals, thus ensuring compression efficiency gains by taken advantage of the strong similarity between the U-BL and EL clusters. In order to do so, the geometry residuals for each cluster coordinate value are computed as follows

$$x_{residual} = x_{EL} - x_{U-BL} \quad (27)$$

where x_{EL} and x_{U-BL} are, respectively, the EL and U-BL coordinate values (for x), and $x_{residual}$ is the residual coordinate computed that will be transformed in the following modules (see Figure 36). Naturally, it is necessary to compute the residual coordinate values for *y* and *z*, changing *x* in (27) for, respectively, *y* and *z*. Figure 45 shows the RD performance of the GTGC solution, with and without exploiting the Inter-layer prediction method, for the *Egyptian Mask* and *Longdress* clouds. The coding approach not exploiting the Inter-layer prediction tool simply consists in transforming the full geometry of the EL clusters, instead of transforming the geometry residuals.



(a) (b)

Figure 45: Inter-layer assessment for: (a) *Egyptian Mask* and (b) *Longdress* point clouds.

As it can be observed from Figure 45, the proposed Inter-layer prediction method leads to significant RD performance gains regarding coding without Inter-layer prediction.

3.2.7. Graph K-NN Creation and Transform Basis Functions Creation and Application

The objective of this set of modules is to learn the graph transform basis functions for each cluster and then apply the learned transform to the residual coordinates, thus computing the transform coefficients; the specific architecture for this sequence of modules is illustrated in Figure 46.

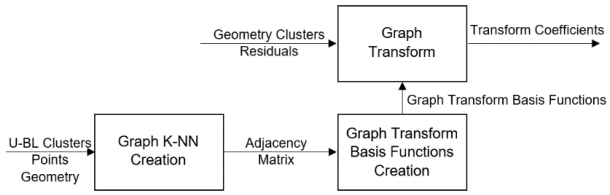


Figure 46: Transform related modules.

The processing of each cloud cluster by the transform related modules proceeds as follows:

- **Graph K-NN Creation:** First, a K-NN graph is created for each of the U-BL clusters computed in Section 3.2.4, after the discarding procedure; for a user-specified K, the graph K-NN creation module computes the graph representing the connectivity and similarity of the input cluster points, i.e. the adjacency matrix with size $N_{points\_cluster} \times N_{points\_cluster}$. For this, the following sequence of steps is implemented:

  1. **Connectivity Matrix Creation:** First, it is necessary to compute the nearest K neighbors for each cluster point. To do so, the *nearestKSearch* function from the PCL is used. After, the K nearest neighbors indices for each of the U-BL cluster points are obtained, it is possible to construct the corresponding connectivity matrix as:

$$Conn_{ij} = Conn_{ji} = \begin{cases} 1\ , if\ i\ and\ j\ are\ neighbors \\ 0, otherwise \end{cases} \quad (28)$$

Note that the connectivity matrix is symmetric as the connections are bidirectional, i.e. if the point with index *i* is connected to the point with index *j,* then both cells $Conn_{ij}$ and $Conn_{ji}$ of the connectivity matrix must be set to one.

2. **Distance Matrix Creation:** Next, using the graph connectivity and points geometry, it is possible to compute the distance between each connected pair of points. Afterwards, with the connectivity information available, the distance matrix with size $N_{points\_cluster} \times N_{points\_cluster}$ is determined as follows:

$$Dist_{ij} = Dist_{ji} = \begin{cases} distance(P_i, P_j), if\ Conn_{ij} = 1 \\ 0, otherwise \end{cases} \quad (29)$$

where $distance(P_i, P_j)$ is the Euclidean distance between the points with indexes *i* and *j* computed as:

$$distance(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \quad (30)$$

At this stage, the desired Euclidean distance for the distance matrix is obtained.

3. **Mean and Variance Computation:** The mean and variance of the distances between connected cloud points are computed. The mean distance is computed as

$$meanDist = \frac{1}{N_{non\_zero\_dist}} \sum_{i=0}^{N\_rows} \sum_{j=0}^{N\_columns} Dist_{ij} \quad (31)$$

where $N_{non\_zero\_dist}$ is the number of cells in the distance matrix with a non-zero distance value $Dist_{ij}$. Then, the variance of the distance between connected cloud points is computed as:

$$varDist = \frac{1}{N_{non\_zero\_dist}} \sum_{i=0}^{N\_rows} \sum_{j=0}^{N\_columns} (Dist_{ij} - meanDist)^2 \quad (32)$$

where $N_{non\_zero\_dist}$ and $Dist_{ij}$ are the same as in (31) and $meanDist$ is the result of (31).

4. **Adjacency Matrix Creation:** This step creates the desired adjacency matrix according to the distance between each connected pair of points and the distance matrix variance. The weight adopted to express the K-NN points similarity is the result of a series of experiments to be presented in the next chapter, thus ensuring that the expression used produces the best RD results. The desired adjacency matrix is obtained by:

$$Adj_{ij} = Adj_{ji} = \begin{cases} weight_{ij}\ , if\ Conn_{ij} = 1 \\ 0, otherwise \end{cases} \quad (33)$$

where $weight_{ij}$ is a weight computed using a suitable weighting expression, such as $\frac{Dist_{ij}}{\sqrt{varDist}}$.

- **Graph Transform Basis Functions Creation:** This module targets the creation of the graph

transform basis functions given the adjacency matrix computed in the previous step and the geometry residuals, for each cluster; the internal architecture of this specific module is illustrated in Figure 47.

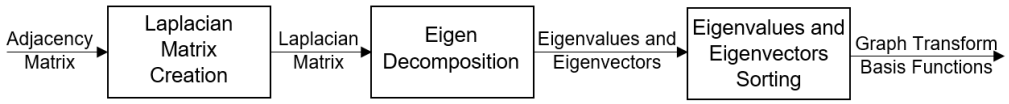


Figure 47: Graph transform basis functions creation architecture.

For each of the U-BL adjacency matrices computed in the Graph K-NN Creation module, this module performs the following processing steps:

1. **Laplacian Matrix Creation:** Using the adjacency matrix previously obtained, a Laplacian matrix is computed. The Laplacian matrix will be later used to compute the graph transform basis functions. The first step in this process is to compute the diagonal matrix for the available adjacency matrix where the diagonal matrix, *Diag*, is expressed as:

$$Diag_{ij} = \begin{cases} \sum_{j=0}^{N\_columns} Adj_{ij}, if\ i = j \\ 0, otherwise \end{cases} \quad (34)$$

   Having both the adjacency and diagonal matrices, it is possible to compute the Laplacian matrix, *Lap,* as:

$$Lap_{ij} = Diag_{ij} - Adj_{ij} \quad (35)$$

2. **Eigen Decomposition:** To obtain the eigenvalues, i.e. a $N_{points\_cluster}$ vector, and the eigenvectors, i.e. a $N_{points\_cluster} \times N_{points\_cluster}$ matrix, the eigen decomposition of the Laplacian matrix obtained in the previous step is performed as:

$$Lap = \Phi \Lambda \Phi^{-1} \quad (36)$$

   where *Lap* is the Laplacian matrix, Φ is the eigenvector matrix and Λ is a diagonal matrix containing the eigenvalues. Each of the eigenvector matrix columns corresponds to a graph transform basis function and the absolute value of the associated eigenvalue is proportional to the basis functions frequency, i.e. if the eigenvalue has a small absolute value the associated eigenvector corresponds to a low frequency transform basis functions and vice-versa.

3. **Eigenvalues and Eigenvectors Sorting:** This step sorts the eigenvalues ascendingly, in terms of their absolute value, and applies the same reordering permutations to the eigenvector matrix columns. At this stage, the eigenvector columns are ascendingly ordered in terms of their frequency, that is, the first eigenvector column is the lowest-frequency basis functions and will originate the DC value after the transform is applied. The obtained sorted eigenvector matrix defines the set of graph transform basis functions to be used for the computation of the transform coefficients for each geometry residuals cluster which is basically a vector of (residual) coordinates.

It is important to state that despite the fact that the graph transform basis functions are learned using the graph vertices geometry information, notably their neighboring distances and connectivity, these basis functions can be applied independently to any type of signal that “lives” on the graph

vertices. For instance, it is possible to apply these graph transform basis functions not only to the *xyz* geometry but also to RGB and to normals information.

- **Graph Transform:** The final step corresponds to the computation of the transform coefficients for the residual geometry coordinates. Naturally, the graph transform basis functions computed in the previous step are used. For each of the geometry clusters residuals components, *x, y* and *z*, a transform coefficient vector is obtained by multiplying the graph transform basis functions matrix by the relevant residual coordinates vector as follows

$$coeff_x = x_{residual} \times \Phi \tag{37}$$

where x is the $N_{points\_cluster}$ residual coordinates vector corresponding to the $x_{residual}$ component, Φ is the graph transform basis functions previously learned and $coeff_x$ is the resulting transform coefficients vector (for the *x* coordinate). Thus, for each cluster, the transform is applied three times to compute three different transform coefficients vectors; this is performed by replacing $x_{residual}$ with $y_{residual}$and $z_{residual}$ in (37), and computing $coeff_x$, $coeff_y$ and $coeff_z$. Note that it is possible to eliminate (zeroing) transform coefficients, making the transform process conceptually lossy. As the graph transform basis functions are ascendingly ordered in terms of frequency, the first transform coefficients correspond to lower frequencies, which should not be discarded since the quality of the point cloud obtained after the inverse graph transform could be severely degraded in that case. The dimensions of the graph transform basis functions and the residuals coordinates vector need to have the same size so that the transform can be applied, i.e. the matrix-vector multiplication can be performed. Therefore, the number of residual coordinate values is equal to the number of points in each U-BL cluster, after the discarding procedure, and must be the same as the number of rows in the graph transform basis functions matrix.

Notice that the graph transform basis functions, learned and applied to compute the transform coefficients, play a major role in the rate and quality of the proposed coding solution. Naturally, the quantization and entropy encoding modules vastly benefit from a good signal decorrelation. As previously stated, these graph transform basis functions are learned using the U-BL clusters. The transform basis functions 'quality' is proportional to the quality of the U-BL clusters, which is related to the corresponding Inter-layer prediction distances. Hence, it is safe to conclude that the density of the BL cloud (controlled by the *BL-EL number of points ratio* parameter introduced in Section 3.2.1) has a major impact on the rate-quality trade-off of the GTGC solution.

The output of this module is a transform coefficients matrix with size $N_{clusters} \times max_{cluster\_size}$, where $N_{clusters}$ is the total number of clusters computed in Section 3.2.3 and $max_{cluster\_size}$ is the dimension of the cluster with the largest number of points, after the discarding procedure introduced in Section 3.2.5. From now on, each of the coefficients matrix columns will be denominated a *subband* since all coefficients in that column correspond to the same frequency. In practice, not all the computed transform coefficients will be coded as some will not be selected.

3.2.8. Graph Transform Coefficients Selection

The main goal of this module is to perform an appropriate selection of the graph transform coefficients,

exploiting the characteristics of the graph-based transform previously applied. This selection process controls the number of transform coefficients to be discarded, i.e. whose corresponding values are set to zero, in order that no rate is spent in coefficients bringing relatively small quality improvements. Taking into account the graph transform basis functions ordering explained in the previous section, the coefficients selected should correspond to coefficients in the first indices of the coordinate vector (i.e. first subbands), which hold the highest amount of energy (i.e. higher square value). The user is referred to Figure 49, in Section 3.2.10, that depicts a chart which shows the coefficients energy per subband and for each coordinate *xyz*.

The control parameter for this module is the *coefficients selection percentage*, defined independently for each transform coefficients vector computed in the previous module; this procedure is equally and independently applied to the various transform coefficients vector components, i.e. *xyz*. This module discards a percentage of the coefficients vector values and, since each vector corresponds to a different cluster, the number of coefficients discarded may be slightly different depending on the size of the cluster (as the vectors may have different dimensions, as explained in sections 3.2.3 and 3.2.5). Knowing that the basis functions created in Section 3.2.7 are ascendingly sorted in terms of their frequency, the selection procedure will only set to zero the last indices of the coefficients vector; this method ensures that the first coefficients to be discarded are those holding the lowest amount of energy. The following expression illustrates the selection procedure for a given transform coefficients vector

$$coeff\_selected_i = \begin{cases} coeff_i, if\ i < floor\left(N_{points\_cluster} \times S_{percentage}\right) + 1 \\ 0, \quad otherwise \end{cases} \tag{38}$$

where $S_{percentage}$ is the selection percentage applied to the coefficients vector $coeff$, i is the coefficients vector index, $N_{points\_cluster}$ is the number of points present in the corresponding cluster, $floor(\dots)$ is the floor function that rounds downward the input value, i.e. returns the highest integer that is not higher than the input value, and $coeff\_selected$ is the resulting coefficients vector, after the selection is applied. Naturally, the larger the selection percentage applied, the lower the number of coefficients discarded and thus the higher the resulting quality (as less coefficients will be set to zero); thus, the coefficients selection module has naturally an impact on the final rate-quality trade-off.

3.2.9. Transform Coefficients Quantization

The objective of this module is to quantize the transform coefficients selected in the previous module. The quantization process aims to map a set of continuous or discrete values to a smaller set of discrete values, thus introducing some distortion and reducing the quality; such procedure is the key element of lossy compression as it reduces the transform coefficients precision, hence lowering the bit rate. It is important to notice that this process is not invertible and introduces distortion, the so-called *quantization error*. This quantization error is the difference between the input and the quantized values and shall be controlled using the quantization step size. Naturally, the lower the rate, the lower the quality; thus, the quantization, in combination with the coefficients selection percentage, the EL discarding threshold and the BL-EL number of points ratio, regulates the rate-quality trade-off.

The quantization method adopted in the GTGC solution is a dead-zone uniform scalar quantizer as shown in Figure 48. The control parameter for this module is the quantization step size, which controls

both the number of quantization bins (given a maximum coefficient value), as well as the quantization bin width of the bins in which the selected coefficients may lie.

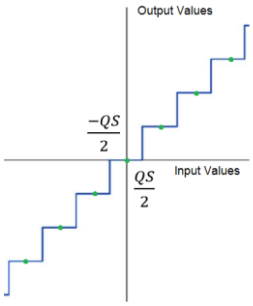


Figure 48: Uniform scalar quantizer with a dead-zone centered around the origin.

The expression used to obtain the quantized coefficients, based on the quantization step size, is as follows

$$coeff_{quant} = floor(\frac{coeff\_selected}{QS} + \frac{1}{2}) \tag{39}$$

where QS is the quantization step size, $floor$ is the floor function, $coeff\_selected$ is the input coefficient to quantize and $coeff_{quant}$ is the resulting quantized coefficient; the sum of $\frac{1}{2}$ inside the floor function is responsible for the dead-zone placement, i.e. from $\frac{-QS}{2}$ to $\frac{QS}{2}$, and ensures that both the negative and positive values are rounded towards their closest integer. Notice that all coefficients lying in the dead-zone will be set to zero, hence its denomination. In addition, it is important to state that this quantization method ensures equal width across all quantization bins, thus a uniform quantization. To obtain the reconstructed coefficients at the decoder side, it is necessary to apply the inverse quantization expression below

$$coeff_{rec} = QS \times coeff_{quant} \tag{40}$$

Naturally, the lower the step size applied, the higher the resulting quality as the number of bins will be larger and the bin width narrowed. In other words, the lower the step size, the lower the quantization error.

3.2.10. Rate-Quality Trade-Off Control

As mentioned before, the coefficients selection and the coefficients quantization processes have a key role on the control of the rate-quality trade-off, this means RD performance. It is, therefore, important to define the best values for the coding parameters which have a high impact on the rate-quality trade-off: the coefficients selection percentage, $S_{percentage}$, and the quantization step size QS. This is precisely the goal of the rate-quality trade-off control module described in this section. As these control parameters should not be defined independently from each other, i.e. in principle, the quantization step size should be lower for higher selection percentages, it is rather mandatory to control the two parameters in a joint way to achieve the best possible RD performance. Hence, an additional control parameter is proposed, the so-called *rate-quality control factor* ($RQCF$). With this control parameter, it will be possible to determine the best pair of quantization step size and coefficients selection percentage values for a target distortion/quality; here lower $RQCF$ values correspond to lower distortions, i.e. higher qualities, while

higher $RQCF$ values correspond to higher distortions, i.e. lower qualities. The remaining coding parameters will be studied in the following chapter, so that the best coding parameters configuration shall be obtained.

Defining the Quantization Step Size

First, the relationship between $RQCF$ and the quantization step size is addressed, as the resulting expression is used later on to obtain the relationship between $RQCF$ and the coefficients selection percentage. With the objective of determining an appropriate quantization step size for a specified $RQCF$ value, the following expression is used

$$QS_x = \frac{coeffx_{max} - coeffx_{min}}{2^{14-RQCF}} \tag{41}$$

where $coeffx_{max}$ and $coeffx_{min}$ are, respectively, the maximum and minimum values of the selected coefficients matrix (for the *x* coordinate) , i.e. already considering all clusters, $RQCF$ is the target rate-quality control factor, QS_x is the resulting quantization step size (for the *x* coordinate). The quantization step sizes for the *y* and *z* coordinates, i.e. QS_y and QS_z, are computed in the same way by replacing in (41) *x* by *y* and *z*, respectively. The motivation behind the expression in (41) comes from the need to ensure that the selected quantization step size leads to a number of quantization levels symbols that does not exceed the maximum number of symbols, the alphabet size, allowed by the entropy encoder; the denominator in (41) controls the maximum number of quantization bins adopted in the solution, which is set to 2^{13} (the entropy codec uses unsigned short precision, i.e. 2^{16}). In other words, with $RQCF$ equal to one the number of bins is 2^{13} and corresponds to the highest quality possible. Finally, the common quantization step size to be used for the three spatial coordinates is obtained using

$$QS = \max(QS_x, QS_y, QS_z) \tag{42}$$

where $\max()$ is the maximum function and QS is the final quantization step size to be applied to the selected coefficients matrix for the *xyz* coordinates.

It is important to state that other approaches have been studied to obtain the quantization step size (or step sizes) as a function of $RQCF$ but the one presented in (42) was the one yielding the best RD performance; for this reason, the expression in (42) has been adopted in the GTGC solution. The possibility to apply different quantization step sizes to each 3D coordinate coefficients matrix as an alternative to the solution in (42) was also considered. Figure 49 shows the sum of the absolute coefficients at the subbands level, for each of the 3D coordinates, for all clusters; the sum of the absolute values is an estimation of the energy present in each subband. As the lower frequency subbands hold larger energy, a zoom-in for the medium to higher frequency subbands is also displayed in Figure 49 to better observe this behavior. It is perceptible that there is no major difference between the estimated energy for the three coordinates, at each subband level. This observation suggests that using a different quantization step size for each of the 3D coordinates should not bring significant benefits in terms of RD performance; for this reason, the solution in (42) has been adopted for the GTGC solution.

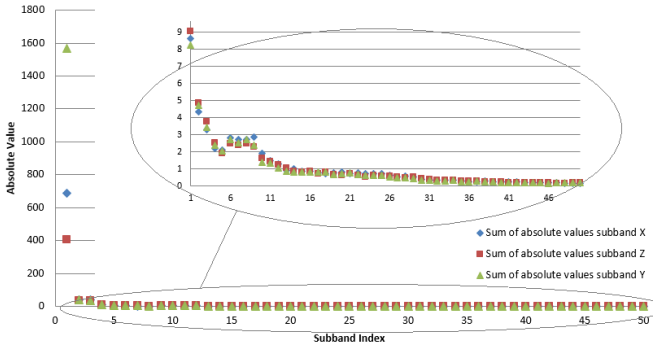


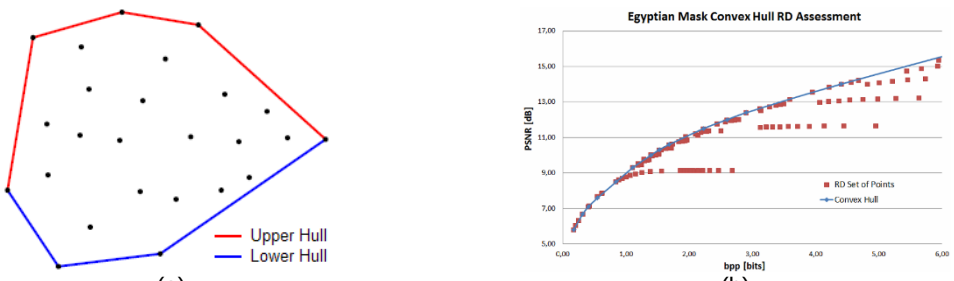
Figure 49: Sum of the absolute values of the (x,y,z) coordinates coefficients subbands for the *Bunny* point cloud with 180 clusters (with a zoom-in for the AC coefficients).

It is important to state that an adaptive quantization step size approach, i.e. different QS for different groups of subbands, was also studied. For instance, it was experimented to use a smaller QS for the first ten bands, i.e. those holding larger estimated energy, and a higher QS for the remaining subbands, i.e. those holding lower estimated energy. This adaptive quantization step size approach did not show promising RD performance when compared to the solution adopted (42). In fact, there is a perceptual study to make related to the perceptual importance of each of these subbands to somehow replicate the subjective-driven selective quantization which happens in image coding.

Defining the Coefficients Selection Percentage

At this stage, the relationship between $RQCF$ and the coefficients selection percentage shall be defined. For a given $RQCF$, there is only one pair of quantization step size and selection percentage that should lead to the best RD performance and, therefore, a reliable relationship between the selection percentage and $RQCF$ must be determined.

To identify this relationship, it is necessary first to determine several RD points, using different values for the parameters $RQCF$ and $S_{percentage}$; after, the convex hull of the resulting set of RD points is computed. In this context, the convex hull defines the set of RD points yielding the best RD performance and, for each selected RD point, the corresponding $RQCF$ and $S_{percentage}$ pair may be selected. In theory, the convex hull for a set of points *S* is defined as the smallest convex polygon encapsulating all the points present in S, as illustrated in Figure 50 (a). Figure 50 (b) shows the set of RD points obtained for the *Egyptian Mask* point cloud and the corresponding upper convex hull, computed using a similar convex hull algorithm to the one in [97]; the pairs of $RQCF$ and $S_{percentage}$ values used to obtain the set of RD points were, respectively, a combination of [1, 10], with increments of 1, and [0, 90%], with increments of 10%.



(a) (b)

Figure 50: (a) Set of RD points and corresponding upper and lower convex hulls *[98]*; (b) Upper RD performance convex hull for the *Egyptian Mask* point cloud.

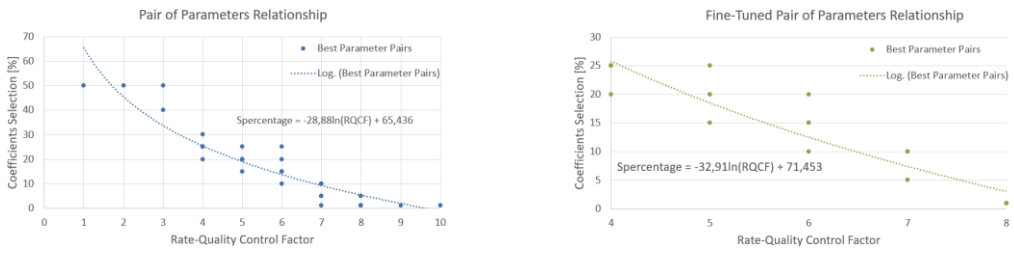
With the objective of determining the relationship between $RQCF$ and $S_{percentage}$, the following sequence of steps was performed:

1. **Upper Convex Hull Determination:** Since the density of the point clouds impacts the obtained transform coefficients values and, therefore, the quantization step size used for a given $RQCF$, a set of five different points clouds with different characteristics was used to determine the target relationship. First, the upper convex hull of the RD points associated to each point cloud, was identified. Based on these upper convex hull RD points, the set of $RQCF$ and $S_{percentage}$ pairs leading to the best RD performance were extracted.

2. **Fitting Curve Computation:** By concatenating the five different sets of parameter pairs obtained in Step 1, a full set of parameter pairs was obtained; this way, it is guaranteed that all the pairs have the same impact on the estimation. By observing the full set of $RQCF$ and $S_{percentage}$ pairs (see Figure 51 (a)), it is possible to notice that $S_{percentage}$ decreases rapidly at the beginning (i.e. for lower $RQCF$ values) and then it gradually decreases as the $RQCF$ value increases; this data trends suggest that a logarithmic function should be a good fitting curve. Hence, a logarithmic function was used as the fitting curve for the full set of parameters pairs, i.e. to define the relationship between $RQCF$ and $S_{percentage}$.

3. **Refined Fitting Curve Computation:** To obtain a better fitting curve, the so-called *fine-tuned* fitting curve, this step targets to more accurately represent the relationship between $RQCF$ and $S_{percentage}$. Having the full set of parameter pairs and the fitting curve computed in Step 2, it is possible to discard some of these pairs to obtain a new fine-tuned fitting curve. The discarding decision works as follows: knowing that, for the same cloud, a selection percentage may have two or more corresponding quantization step sizes, i.e. two or more pairs of parameters, one of these pairs may be discarded; the discarded pair is the one that deviates the most, in terms of the coefficients selection percentage value difference, from the one obtained using the fitting curve computed in the previous step. Moreover, for the lower $RQCF$ values (1 to 3), the highest value for the selection percentage has been set to 50% as the resulting bitrate would increase drastically for higher percentages; for the highest $RQCF$ values (9 and 10), it is beneficial to apply the lowest coefficients selection percentage, which has been set to 5%, as lower percentages may discard coefficients with high relevance and, therefore, compromise the resulting quality severely.

Figure 51 (a) illustrates the full set of convex hull parameter pairs (Step 1) and the corresponding logarithmic fitting (Step 2) while Figure 51 (b) illustrates the refined fitting curve (Step 3).



(a) (b)

Figure 51: (a) Logarithmic fitting for the $RQCF$ and $S_{percentage}$ relationship; (b) Logarithmic fitting after refinement.

The resulting refined fitting expresses the coefficients selection percentage as a function of *RQCF* as:

$$S_{percentage}(RQCF) = \begin{cases} 50\%, if\ RQCF \leq 3 \\ -32.9\ln(RQCF) + 71.453\ if\ 3 < RQCF < 9 \\ 5\%, if\ RQCF \geq 9 \end{cases} \quad (43)$$

where $S_{percentage}$ is the coefficients selection percentage to be used to obtain the best RD performance and $RQCF$ is the user-defined, target rate-quality control factor.

3.2.11. Entropy Coding

The purpose of this module is to exploit the statistical redundancy of the quantized transform coefficients, here the *symbols* to be coded, computed in the coefficients quantization module described in Section 3.2.9. In other words, the main goal of an entropy encoder is to losslessly and efficiently represent a set of symbols into a sequence of bits while considering its statistics. The efficiency of an entropy encoder is measured by how close the average number of bits per symbol is to the theoretical entropy value.

The entropy encoder used in the GTGC solution is an arithmetic encoder [99]. To better understand the arithmetic encoding procedure, please refer to [99] as it contains an example and source code implementation. The main idea behind arithmetic coding is to represent the set of symbols to be encoded, the so-called *message*, using an interval of real numbers between 0 and 1. For this, the symbol probabilities, collected in the so-called *probability tables,* are used to encode and decode the symbols. As these probability tables contain the corresponding probability of each unique symbol, they must be present at both the encoder and decoder. In order to determine the most efficient method (in terms of RD performance) to entropy encode the quantized coefficients matrix, two approaches were considered:

1. **Subbands Approach:** Here each column of the quantized coefficients matrix, i.e. each quantized coefficients subband, is independently entropy encoded.
2. **Clusters Approach:** Here each row of the quantized coefficients matrix, i.e. all the quantized coefficients associated to a given cluster, is independently entropy encoded.

Figure 52 shows the RD results obtained for both entropy coding approaches above: while the green curve represents the subbands approach, the red curve corresponds to the clusters approach.

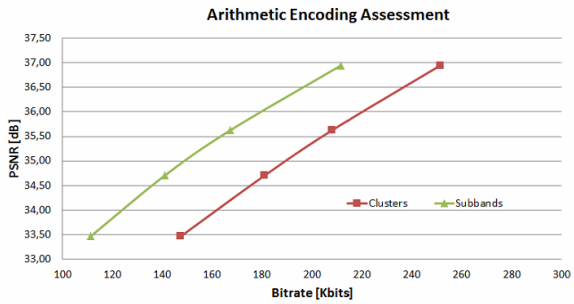


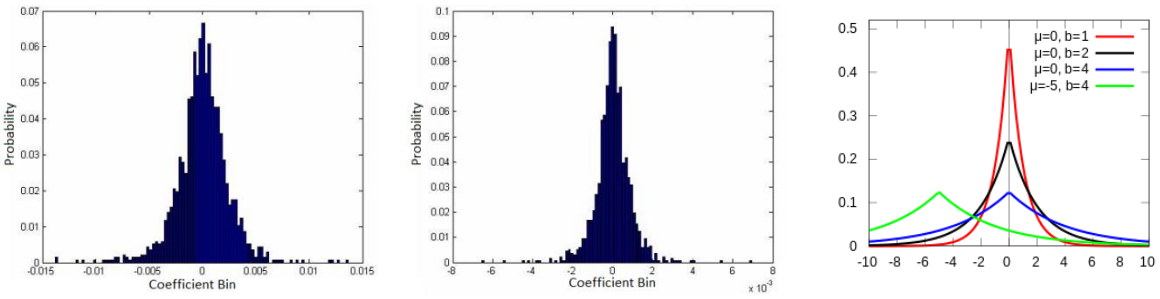
Figure 52: Arithmetic encoding of the quantized coefficients: subbands approach (green curve) versus the clusters approach (red curve).

As it can be observed from Figure 52, the subbands approach yields the most promising RD performance results and, hence, the subbands approach was the one adopted in the GTGC.

After determining the entropy encoder input data model, the next step consists in statistically characterizing the input data so that the probability tables, necessary for entropy coding, are created. Figure 53 a) and b) depict relative frequency histograms for two different quantized coefficients subbands, notably subbands 5 and 70. By analyzing the histograms in Figure 53 a) and b), showing the relative frequency of each symbol, it is possible to conclude that the quantized coefficients subbands follow rather well a Laplacian distribution; the same conclusion holds for the remaining subbands. The Laplacian distribution is a continuous probability distribution characterized by the following probability density function

$$f(x|\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}} \tag{44}$$

where x is a random variable, μ is the mean of the distribution and b is a parameter related to the distribution's variance ($\sigma^2 = 2b^2$). Figure 53 c) depicts a set of Laplacian probability density functions, for different values of the mean and variance parameters.



(a) (b) (c)

Figure 53: (a) Relative frequency histogram for the subband with index 5; (b) Relative frequency histogram for the subband with index 70; (c) Laplacian probability density functions for different values of the mean and variance parameters *[100]*.

As it can also be observed from Figure 53 a) and b), the mean value of the coefficients subbands distribution is zero (or rather close to zero). Thus, the entropy encoder method proposed in the GTGC solution is an arithmetic encoder with frequency (or probability) tables derived from a zero-mean Laplacian distribution, i.e. with $\mu = 0$.

To compute the probability table associated to the arithmetic codec input data, the following sequence of steps is implemented:

1. **Laplacian Parameter Computation:** First, the Laplacian parameter *b* must be estimated for each subband as different subbands have different variances (see Figure 53 a) and b)). Naturally, the *b* parameters (one per subband) have to be sent to the decoder to guarantee that both the encoder and decoder can create exactly the same frequency tables. The procedure to estimate the parameter *b*, for each subband, is as follows:

   - **Variance Computation:** The variance of the selected coefficients subband is computed as $E[(coeff\_selected)^2]$, where $E[\dots]$ is the expected value and $coeff\_selected$ are the coefficients after the selection procedure.

   - ***b* Parameter Estimation:** After computing the variance of the coefficients for each subband, the Laplacian parameter *b* is computed as

$$b = \sqrt{\frac{\sigma^2}{2}} \tag{45}$$

Note that after estimating the *b* parameters (one for each subband), they are converted from floating precision (double) to a fixed precision (unsigned short); this conversion makes it possible to reduce the rate associated to their transmission, as only 2 bytes per *b* value are needed in fixed precision.

2. **Set of Possible Values Determination:** Next, the set of possible values for the reconstructed coefficients, $coeff_{rec}$, is obtained (the decoder must also replicate this procedure) as follows

$$coeff_{rec} = QS \times coeff_{quant} \tag{46}$$

where QS is the quantization step size and $coeff_{quant}$ are all possible quantized coefficient values determined for the selected $RQCF$ value (which is also made available at the decoder); in this case, they are represented by all integer values in the $[-2^{13-RQCF}, 2^{13-RQCF}]$ interval.

3. **Probability Computation:** Once the Laplacian parameter *b* is known, the probability of each reconstructed coefficient $coeff_{rec}$ is computed from:

$$f(coeff_{rec}|b) = \frac{1}{2b} e^{-\frac{|coeff_{rec}|}{b}}. \tag{47}$$

Followed by a normalization step that ensures the probabilities sum is equal to one. The normalized probability of each reconstructed coefficient is then associated to the corresponding quantized coefficient value, as in practice the quantized coefficients will be the arithmetic coder input.

At the end, all $coeff_{quant}$ values shall have a corresponding normalized probability so that it is possible to apply the arithmetic encoding procedure. The output of the entropy coding module corresponds to the EL point cloud bitstream, which, together with the BL point cloud bitstream, and bitstream associated to the transmission of the EL centroids and the post-discarding clusters dimensions, constitutes the full bitstream of the GTGC solution to be sent to the decoder side.

Chapter 4

4. Proposed Coding Solution Performance Assessment

The key goal of this chapter is to assess the performance of the proposed GTGC solution, previously presented. First, the test material is introduced followed by the test conditions. Next, the objective quality metrics used for RD performance assessment are detailed. Afterwards, the benchmark coding solutions are described and the corresponding coding parameters values listed. Then, a study targeting the GTGC coding parameters optimization is detailed. Finally, a comparison between the GTGC and the alternative benchmark solutions is presented. To better compare the coding solutions, not only objective quality metrics are used but also some informal subjective assessment of the decoded point clouds is performed.

4.1. Test Material

The main purpose of this section is to present the set of point clouds used along this chapter, notably detailing their main characteristics. To assess the GTGC performance, the test material was selected from the MPEG repository created for the MPEG CfP on PCC [101]. Figure 54 shows the point cloud test material, with and without color attributes, notably: (a) *Egyptian Mask*; (b) *Statue Klimt;* c) *Loot;* and d) *Longdress*. The selected point clouds include two different types of point clouds depending on the way they were acquired and their general characteristics, notably inanimate objects and people. The inanimate objects dataset has acquisition noise, i.e. noisy surface, holes (regions with no points) and non-uniform regions in terms of density (that include sparser and denser regions). The people dataset has smoother surfaces and more uniform regions with no obvious hole regions.

The unique characteristics of each selected point cloud, which will be critical to better analyze the coding performance, are now detailed:

1. ***Egyptian Mask*:** Inanimate object point cloud, with 272 689 3D points, representing an Egyptian mask; this point cloud has large holes, the non-uniform regions in terms of density, and a rather noisy surface.
2. ***Statue Klimt:*** Inanimate object point cloud, with 499 886 3D points, representing the statue of a woman holding a baby; this point cloud also has a noisy surface and non-uniform regions in terms of density of points.
3. ***Loot*:** People type point cloud, with 805 285 3D points, representing a man; the man arms (high amplitude and distant from each other and from the body) and the fingers are the most difficult parts to be coded.
4. ***Longdress*:** People type point cloud, with 857 966 3D points, representing a woman wearing a long dress; the hair details and the dress surface are the most difficult parts to be coded.

(a) (b) (c) (d)

Figure 54: Test material point clouds, with and without color attributes: (a) *Egyptian Mask*; (b) *Statue Klimt*; (c) *Loot*; and (d) *Longdress*.

4.2. Test Conditions

This section presents the most relevant test conditions that were applied to the test material.

Pre-processing the Test Material

To ensure that the test material is in the appropriate format to be coded, some pre-processing steps had to be applied; these pre-processing steps are presented in the following.

The objective of the pre-processing procedure is to remove the point cloud attributes and normalize the resulting geometry information. To do so, the following pre-processing steps are performed:

1. **Attribute Extraction:** This step has the objective of obtaining only the geometry information of the point cloud to be coded, since the GTGC only targets geometry compression. After applying this step, the resulting point cloud only contains geometry information, i.e. no color or other attributes, as presented in Figure 54.
2. **Normalization:** This step ensured that all point clouds being coded have similar bounding boxes, thus guaranteeing that the geometry information is always within the same range of values, i.e. the same dynamic range. This pre-processing step is important to enforce that the several point clouds to be coded have transform coefficients in a similar dynamic range; this should allow to find optimal coding parameters, such as the quantization step size and the coefficients selection percentage, transversal to all clouds. To normalize a point cloud, the following steps are performed:
    - **Maximum and Minimum:** For the point cloud being coded, the maximum and minimum values, for each 3D coordinate, are obtained.

- **Absolute Maximum:** The maximum of the absolute values of the maxima and minima obtained in step 1 is computed as

$$abs_{max} = \max(abs(max_x), abs(min_x), abs(max_y), abs(min_y), abs(max_z), abs(min_z)) \quad (48)$$

where $abs()$ computes the absolute value, $\max()$ computes the maximum value, min_x and max_x are, respectively, the minimum and maximum value for the *x* coordinate (similar for the *y* and *z* coordinates), and abs_{max} is the resulting absolute maximum to be used in the next step.

- **Normalization:** This step normalizes all points coordinates to the interval $[-1, 1]$ as follows, using as example the *x* coordinate:

$$x_{normalized} = \frac{x}{abs_{max}} \quad (49)$$

where x is the initial coordinate value, abs_{max} is the absolute maximum computed in (48) and $x_{normalized}$ is the resulting normalized x coordinate. Naturally, this normalization process has to be applied to all *xyz* coordinate values.

It is important to state that, when assessing the coding performance using objective quality metrics and following the specifications defined in the MPEG CfP [63], it is necessary to perform the inverse operation to the point clouds as the objective quality assessment is performed between denormalized original and decoded point clouds.

Defining the Appropriate Bitrate Range

To perform the appropriate GTGC assessment, it is important to adopt a bitrate range which is perceptually meaningful; for example, it does not make sense to consider rate ranges where the resulting quality is already so high that no improvements are perceptually relevant, even if the objective quality keeps improving. This reasoning is especially relevant for application scenarios, such as 3D immersive telepresence, 3D broadcasting and video games, where lossless point cloud reconstruction is not required. Thus, the main goal here is to select a bitrate range which avoids qualities that are very similar to each other. To select these rates, an informal subjective quality assessment has been performed with rendered point clouds since this is what the users see and also including the color attributes, again because this is what is done for the addressed application scenarios. Moreover, the subjective quality analysis is rather difficult, i.e. the coding artefacts are hard to perceive, when presenting only geometry information. In fact, several visualizations of the rendered point clouds with and without color were performed and the final conclusion was that the color attributes shall be included to aid the subjective assessment. However, it is important to stress that no color attributes coding was performed and thus all the bits per point (bpp) results presented only refer to the geometry.

With the objective of defining a proper bitrate range, Figure 55 shows two point clouds, one for each type of selected clouds: (a) original *Egyptian Mask* point cloud with 3D 272 689 points; (b) *Egyptian Mask* decoded point cloud after coding with PCL at 2 bpp, resulting in 167 794 3D points; (c) original *Loot* point cloud with 805 285 3D points; (d) *Loot* decoded point cloud after coding with PCL at 2 bpp, resulting in the same number of points as the original point cloud (805 285 3D points). The rendering was performed using the Technicolor software also used in MPEG [102]; this rendering software has two key control

parameters: the point size and the rendering primitive. Different point size values were tested, for each point cloud presented, and the size producing no holes in the rendered point cloud was selected. The rendering primitive used for the four point clouds was cubes, in detriment of points or splats.

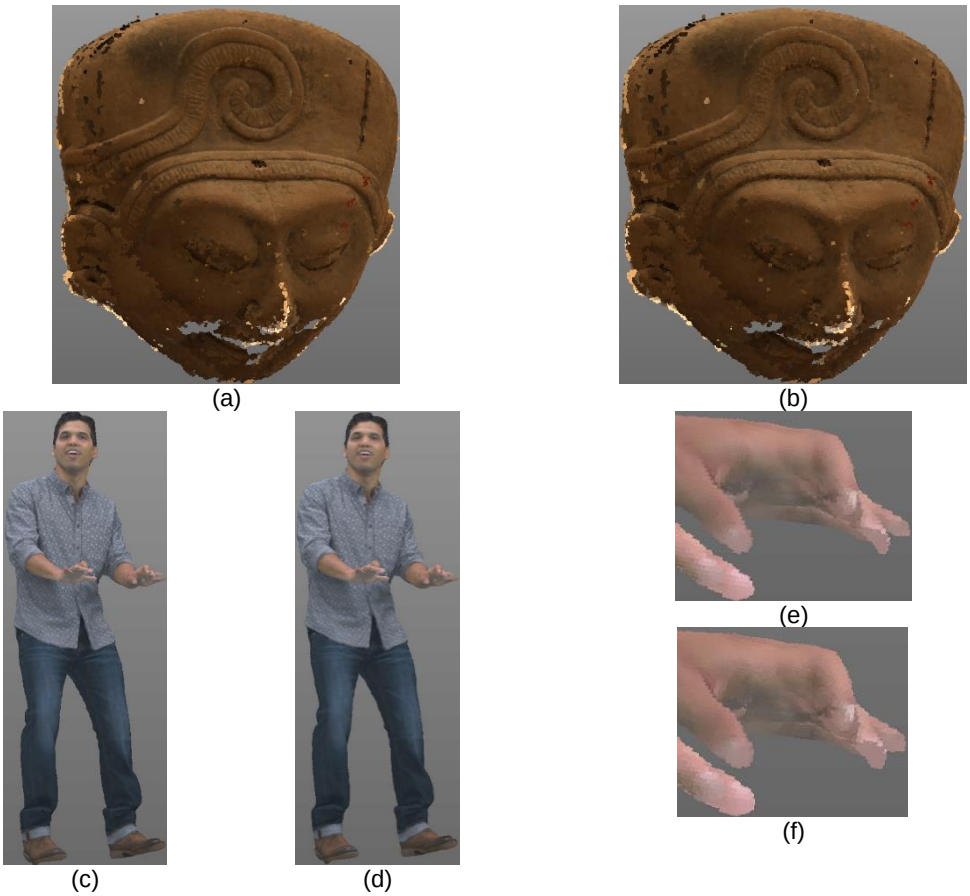


Figure 55: Bitrate range for performance assessment: (a) *Egyptian Mask* original point cloud; (b) *Egyptian Mask* PCL coded at 2 bpp; (c) *Loot* original point cloud; (d) *Loot* PCL coded at 2 bpp; (e) *Loot* original point cloud zoom-in region; and (f) *Loot* PCL coded at 2 bpp zoom-in region.

From Figure 55, it is perceptible that, at 2 bpp coding, the quality of the PCL decoded point clouds is already very high, both in terms of the clouds shape and its resolution. Based on this evidence, it was concluded that the appropriate performance assessment of the coding solutions may be done using a bitrate range of $[0, 2]$ bpp. Although recommending this bitrate range, it is important to stress that the 2 bpp maximum value may be adjusted for the specific target point cloud as some datasets achieve rather high qualities for lower bpp values.

4.3. Objective Quality Metrics

This section outlines the objective quality metrics used for the assessment to be performed. Following the MPEG CfP [63], the objective performance assessment will be conducted based on the RD performance. In this context, the rate will be reported as bpp, computed by dividing the total rate by the number of points in the original point cloud, while the distortion will be reported in terms of the PSNR of a point-to-point error and the PSNR of a point-to-plane error, i.e. C2C and C2P metrics. The number of decoded points is also an important metric, especially because it may be substantially lower than the number of points in the original cloud, notably for PCL coding. The general PSNR expression used for the geometric distortion assessment is:

$$gPSNR = 10\log_{10}\left(\frac{3peak^2}{sRMS}\right) \quad (50)$$

where $sRMS$ is the symmetric root mean square distance; symmetric because this distance is computed first for all points of the original point cloud and after for all points of the decoded point cloud as explained in Section 2.6. The $peak$ parameter is the adopted peak signal value (presented in Table 5) and $gPSNR$ is the resulting geometric PSNR value. It is important to state that the $sRMS$ computation is performed differently for the C2C and C2P metrics.

Since both the coding solutions being assessed do not guarantee that the decoded point clouds have the same number of points as the original point clouds, a label will be introduced in the RD performance plots specifying the number of points present in the decoded clouds. While the PSNR is important, it is important to know if the same PSNR value is reached with a similar or different number of points from the original cloud.

Table 5 contains important information about the test point clouds, namely their dataset, the number of points, the peak value used to compute the $gPSNR$ and the coordinate values precision (in bits). Note that both the peak and precision values here defined are those adopted in the MPEG CfP.

Table 5: Test material information: dataset, number of points, peak value and precision.

Point Cloud	Dataset	Number of Points	Peak Value	Precision [bits]
Egyptian Mask	Inanimate objects	272 689	0.142735	32
Statue Klimt		499 886	0.30491	32
Loot	People	805 285	1023	10
Longdress		857 966	1023	10

The reasoning behind the different peak values presented above is related to the coordinate precision values of the different types of clouds, notably:

1. **Inanimate Objects:** This is a dataset where coordinate values are represented using floating point values with 32 bits; for this dataset, the peak value is computed as the maximum of the nearest neighbor distance for all points in the original point cloud.
2. **People:** This is a lower precision dataset where the coordinate values are represented using integers in the $[0, 1023]$ range (10 bits); for this dataset, the peak value corresponds to the maximum possible range of the coordinate values.

4.4. Benchmarking

This section intends to introduce the adopted coding benchmarks and present the set coding parameters values. In summary, two coding benchmarks are considered:

1. **PCL Intra solution:** The PCL Intra solution corresponds to the PCL codec used for static point clouds; the PCL Intra coding parameters are the point precision and voxel resolution.
2. **The so-called unrealistic graph transform geometry coding (U-GTGC) solution:** The U-GTGC solution basically corresponds to the GTGC solution using a set of graph transform basis functions learned using the EL clusters geometry information, instead of the U-BL clusters geometry information.

While this approach is unrealistic as the EL clusters geometry information is not available at the decoder side, it allows to assess the RD performance of the proposed codec using ideal transforms, thus assessing the impact of using practical transforms, learned from an upsampled cloud. Also, instead of the residuals, this solution codes the EL clusters geometry information. The coding parameters for this solution consist on the adjacency matrix weight expression, the K value per cluster and the number of points per cluster; all these coding parameters are also present in the GTGC solution and are introduced in the following section.

Table 6 depicts the coding parameter values used for the PCL solution; as previously stated, the point precision tool was disabled by setting its parameter value as the same as the voxel resolution. Using these values, the corresponding RD curves for each cloud were obtained and will be shown in Section 4.6, dedicated to the RD performance assessment.

Table 6: Coding parameter values for the PCL solution.

	PCL Coding Parameters Values				
Test Material Point Cloud	Point Precision/Voxel Resolution				
Egyptian Mask	0.011	0.014	0.017	0.02	0.03
Statue Klimt	0.06	0.08	0.1	0.125	0.2
Loot	1.25	1.9	2.25	3	5
Longdress	1.25	1.8	2.25	3	5

The coding parameter values used to create the RD performance curves for the U-GTGC solution are the same as for the GTGC, defined in the following section; the only difference is that the BL-EL points ratio is set to 0% (there is no BL cloud) and the EL discard is disabled. Therefore, the number of points present in the U-GTGC solution is the same as the number of points in the original point cloud.

4.5. GTGC Parameters Optimization

The main goal of this section is to present the performance assessment impact of the GTGC coding parameters and to define the best coding parameters configuration. To obtain this best coding configuration, it is important to perform a thorough study of the performance impact of each coding parameter. With this objective in mind, the target is to list the main coding parameters and their corresponding values or expressions used in the course of the following assessment. The coding parameters sensitivity study has been performed as follows: for each single coding parameter, the set of relevant values were evaluated and the best value in terms of performance is selected for the next experiments. Naturally, it is important to establish a suitable order for this coding parameters study as this order has an impact on the associated selection. The order used for the coding parameters study is the order corresponding to the rows in Table 7 and also in the list below. A brief description of each coding parameter will now be outlined:

1. **Adjacency Matrix Weight:** This expression represents the possible values for the weight in the adjacency matrix which is obtained from the graph K-NN creation (detailed in Section 3.2.7). Three different expressions were studied: a ‘transparent’ weight equal to one, making the weight independent from the points edge distance which represents the topology of the U-BL cluster; one inversely proportional to the points edge distance, i.e. higher distances lead to lower weights; and a last one proportional to the points edge distance, i.e. higher distances lead to higher weights.

2. **K Value per Cluster:** For each U-BL cluster, this parameter expresses the K value for the graph K-NN creation, obtained using a percentage of the total number of cloud points in each cluster. When assessing the different values for this coding parameter, the adjacency matrix weight expression used will already be the best. The values that were tested corresponded to the $[5, 60]$% interval; an increment of 5% was used to avoid testing too similar values.

3. **Target Number of Points per Cluster:** This parameter defines the target number of points in the EL and U-BL clusters. The input parameter to the K-Means clustering module, i.e. the number of EL and U-BL clusters, is obtained by dividing the total number of points in the original cloud by this parameter value. For this coding parameter experiment, the parameters previously described are used with their corresponding best values. The values for the target number of points per cluster were in the $[200, 500]$ interval, with increments of 50.

4. **BL-EL Points Ratio and EL Points Discarding Threshold:** The first parameter expresses the BL-EL points ratio necessary for the BL octree creation and coding module and defines the number of points in the BL point cloud. The study of this parameter was jointly done with the EL discard threshold parameter, as these parameters are strongly correlated. The second parameter is used in the U-BL clusters creation and matching module and expresses the number of EL points to be discarded, i.e. which are not present in the decoded cloud; in practice, it defines how lossy is the adopted coding configuration (in terms of number of decoded points). The values for the BL-EL points ratio were 1, 2, 5, 10 and 15 % and for the EL points discarding threshold were 0.05, 0.1, 0.25, 0.5, 0.75, 1, 2 and 3 times the Inter-layer mean distance. Notice that it is possible to disable the discarding procedure, thus coding all original points, i.e. not discarding any. In this case, the number of decoded points is the same as the number of original points.

Table 7: List of GTGC coding parameters and corresponding test values used in the study of the best coding solution configuration.

Coding Parameter	**Values Tested**									
Adjacency Matrix Weight	1		$e^{-(\frac{Dist_{ij}}{\sqrt{variance}})}$				$\frac{Dist_{ij}}{\sqrt{variance}}$			
K Value per Cluster (% of the number of points in a cluster)	$[5, 60]$% , with 5% increments									
Target Number of Points per Cluster	$[200, 500]$, with 50 increments									
BL-EL Points Ratio (%)	1%		2.5%		5%		10%		15%	
EL Points Discarding ($\mathrm{MeanDist_{IL}}$)	Disabled	Enabled:	0.05	0.1	0.25	0.5	0.75	1	2	3

At this stage, it is possible to perform the necessary coding parameters study that targets the identification of the best GTGC configuration. To limit the complexity associated to this optimization process, a single point cloud was selected, in this case the *Egyptian Mask* point cloud. Table 8 presents the coding parameters used in the course of this assessment, before the best configuration for each parameter is selected.

Table 8: Coding parameter values used before computing the best configuration.

Coding Parameter	Value
K Value per Cluster (% of the number of points in the cluster)	10%
Target Number of Points per Cluster	200
BL-EL Points Ratio (%)	5%
EL Points Discarding ($\mathrm{MeanDist}_{\mathrm{IL}}$)	Disabled

For example, when assessing the best values for the target number of points per cluster parameter, and as at that stage the K value per cluster is already obtained, only the BL-EL points ratio and the EL points discarding will use the values from Table 8. To plot each RD curve in the following sections, five different RQCF, introduced in Section 3.2.10 with the purpose of controlling the rate-quality trade-off, are used in the [6,10] interval.

4.5.1. Adjacency Matrix Weight

To define the adjacency matrix entries for the GTGC solution, several expressions were assessed for the computation of its values. The assessment was performed using the coding parameter values present in Table 8.

Figure 56 shows the RD performance results for the three different weight expressions tested, depicted in Table 7.

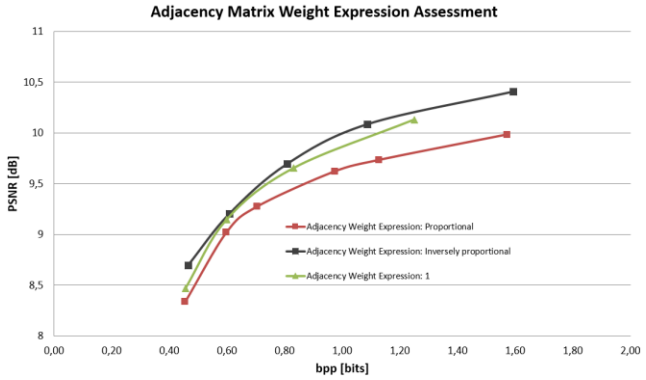


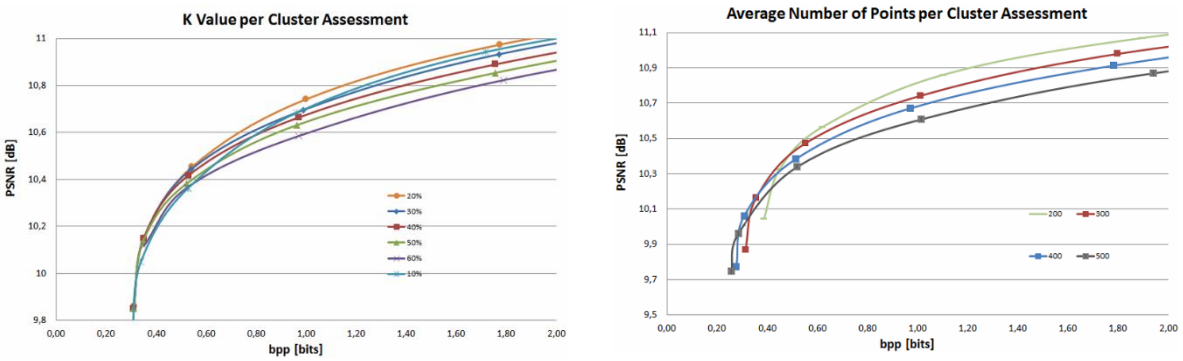
Figure 56: Adjacency matrix weight RD performance impact for the *Egyptian Mask* point cloud.

The adjacency matrix weight expression selected is the one inversely proportional to the points edge distance ($e^{-(\frac{Dist_{ij}}{\sqrt{variance}})}$) since it shows the best RD performance.

4.5.2. K Value per Cluster

To select the K value per cluster as a function of the percentage of points present in the corresponding cluster, the following assessment is performed using $e^{-(\frac{Dist_{ij}}{\sqrt{variance}})}$ for the adjacency matrix weight expression and the parameter values presented in Table 8 for the remaining coding parameters.

Figure 57 (a) shows the RD performance results for the set of K values. To better analyze the coding parameter behavior in terms of RD performance, only six curves are presented in the RD plots, representing the [10,60]% interval with increments of 10% (instead of 5%).



(a) (b)

Figure 57: (a) K value per cluster RD performance assessment impact; (b) Target number of points per cluster RD performance assessment results. Both (a) and (b) for the *Egyptian Mask* point cloud.

From Figure 57 (a) above, it is possible to conclude that there is a correlation between the RQCF value and the points percentage for the K-NN value. In other words, the K value per cluster depends on the target bitrate. Thus, using the results obtained from this study it was possible to define a K value per cluster for each RQCF.

4.5.3. Target Number of Points per Cluster

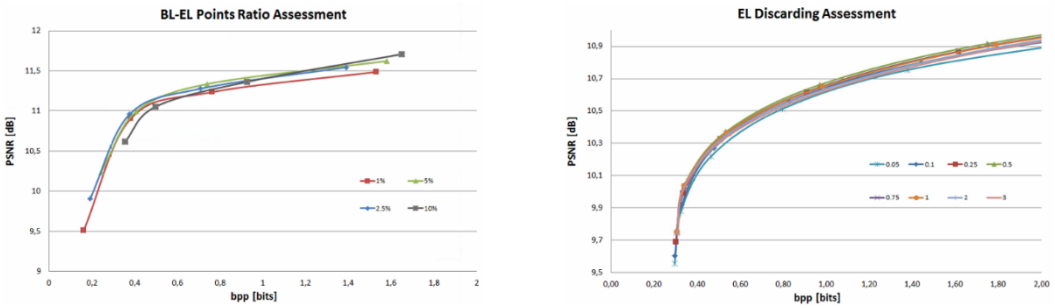
With the goal of selecting the target number of points per cluster, which is used to define the number of EL clusters computed by the K-Means module, the following study was performed, using the set of values depicted in Table 7.

Figure 57 (b) shows the RD performance results for the target number of points per cluster assessment; however, only four curves are plotted to better analyze the parameter behavior. From the assessment results, it is possible to conclude that the best target number of points per cluster depends on the target bitrate, a similar conclusion with respect to the last assessment. Therefore, this coding parameter is adjusted according to the RQCF value. This adaptation allows to select a lower value for this parameter (higher number of clusters) when targeting higher qualities and vice-versa. It is important to state that, as each test point cloud has a different number of points, the target number of points per cluster will result into a different number of clusters across the test material; thus, the area of each cluster will also vary across the test material.

4.5.4. BL-EL Points Ratio and EL Points Discarding

To define the BL-EL points ratio and EL points discarding pair to be used in the GTGC solution, a combinatorial study was performed; this should allow to select the best pairs of parameter values for each RQCF. Thus, all combinations of values present in Table 7, for the BL-EL points ratio and EL points discarding, are evaluated.

Figure 58 shows the RD assessment performance results for: (a) the EL discarding value fixed, disabled; and (b) BL-EL points ratio value fixed at 5%. This has been done to facilitate the visualization of the results; despite the plot being done with fixed values, the assessment was done with the full combination of parameters.



(a) (b)

Figure 58: BL-EL points ratio and EL points discarding RD performance assessment using the *Egyptian Mask* point cloud for: (a) several BL-EL points ratio values; and (b) several EL discarding values.

From Figure 58 it can be concluded that there is a strong relation between the BL-EL points ratio and the RQCF value. Therefore, it is necessary to determine a suitable value for each RQCF adopted. Using the results of this study, it was possible to select pairs of BL-EL points ratio and EL points discarding, for each RQCF value. At this stage, all coding parameter values have been selected and the final coding solution performance results may be obtained; Table 9 shows the coding parameter values used to obtain the final RD performance assessment results.

Table 9: Best configuration of coding parameter values used for the final RD performance assessment.

RQCF	Adjacency Matrix Weight	K Value per Cluster (% of the number of points in the cluster)	Target Number of Points per Cluster	BL-EL Points Ratio (%)	EL Points Discarding ($\mathrm{MeanDist}_{\mathrm{IL}}$)
6	$e^{-(\frac{Dist_{ij}}{\sqrt{variance}})}$	20	200	10	0.5
7		20	300	5	
8		20	300	5	
9		50	500	1	
10		50	500	1	

In the following section, the proposed codec will be evaluated against the PCL coding solution presented in Section 2.7.1, and the U-GTGC solutions using the objective quality metrics previously outlined; moreover, some informal subjective assessment will be performed for both GTGC and PCL coding solutions.

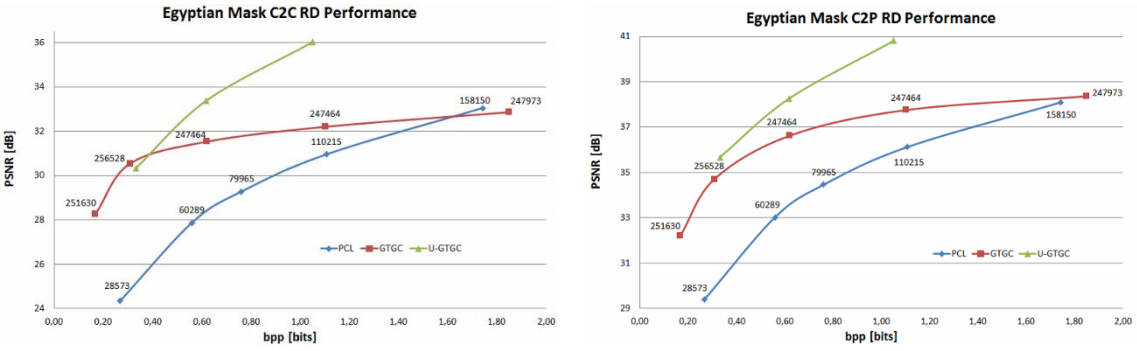
4.6. RD Performance Assessment

The main goal of this section is to present the final RD performance for the GTGC solution and to benchmark it with the PCL and the U-GTGC solutions. First, the RD performance for each solution is obtained using the objective quality metrics previously outlined. After, for the GTGC and PCL solutions, an informal subjective assessment is done to better understand their perceptual strengths and drawbacks.

4.6.1. Formal Objective Assessment

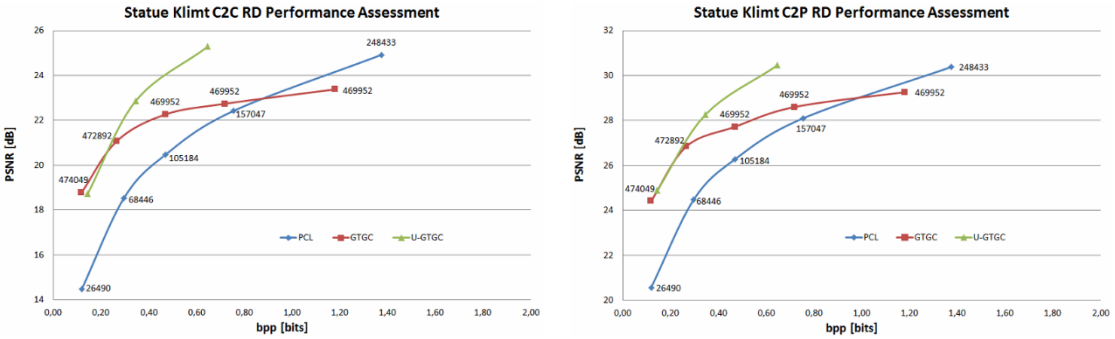
First, for each test point cloud, the RD performance is presented, for both the C2C and C2P metrics. Next, the Bjøntegaard-delta (BD) metrics are used to compute the bitrate/PSNR gains between the two realistic coding solutions, i.e. GTGC and PCL. Finally, the BD-PSNR and BD-bitrate results for each point cloud are used to compute a mean to express the overall GTGC gains over PCL.

Figure 59 to Figure 62 (a) and (b) show the RD performance using, respectively, the C2C and C2P objective quality metrics, for the *Egyptian Mask, Statue Klimt, Loot and Longdress* point clouds.



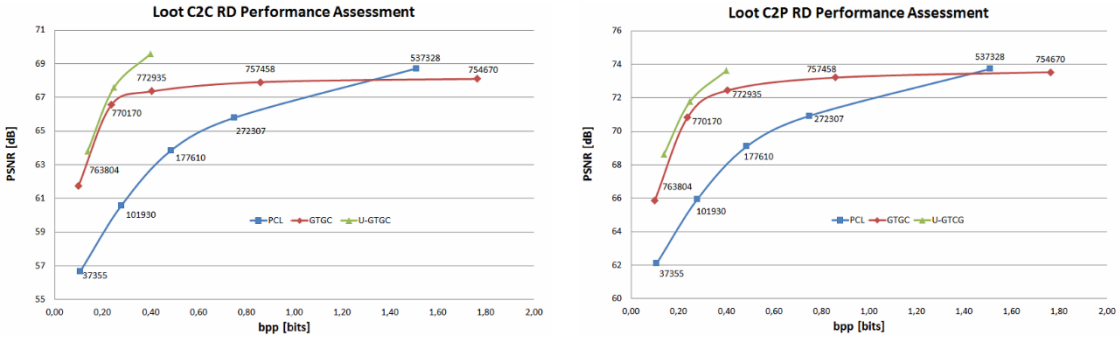
(a) (b)

Figure 59: (a) C2C and (b) C2P RD performance for the *Egyptian Mask* point cloud.



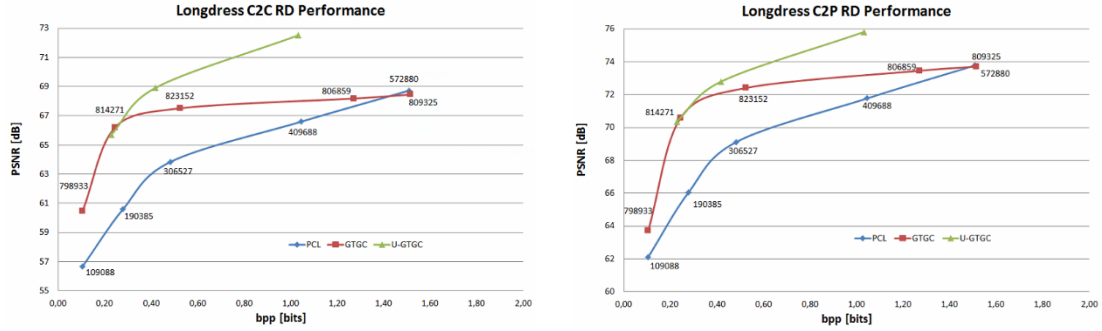
(a) (b)

Figure 60: (a) C2C and (b) C2P RD performance for the *Statue Klimt* point cloud.



(a) (b)

Figure 61: (a) C2C and (b) C2P RD performance assessment for the *Loot* point cloud.



(a) (b)

Figure 62: (a) C2C and (b) C2P RD performance for the *Longdress* point cloud.

At this stage, it is possible to use the BD metrics to assess the GTGC performance regarding the PCL solution. To do so, the BD-PSNR and BD-bitrate gains, for both the C2C and C2P metrics, and for each point cloud, are computed and depicted in Table 10. Also, the mean of the BD values is computed to estimate the overall gain. The BD metrics shown were computed using the four lower bpp values for each solution.

Table 10: BD-PSNR and BD-Rate considering both C2C and C2P metrics.

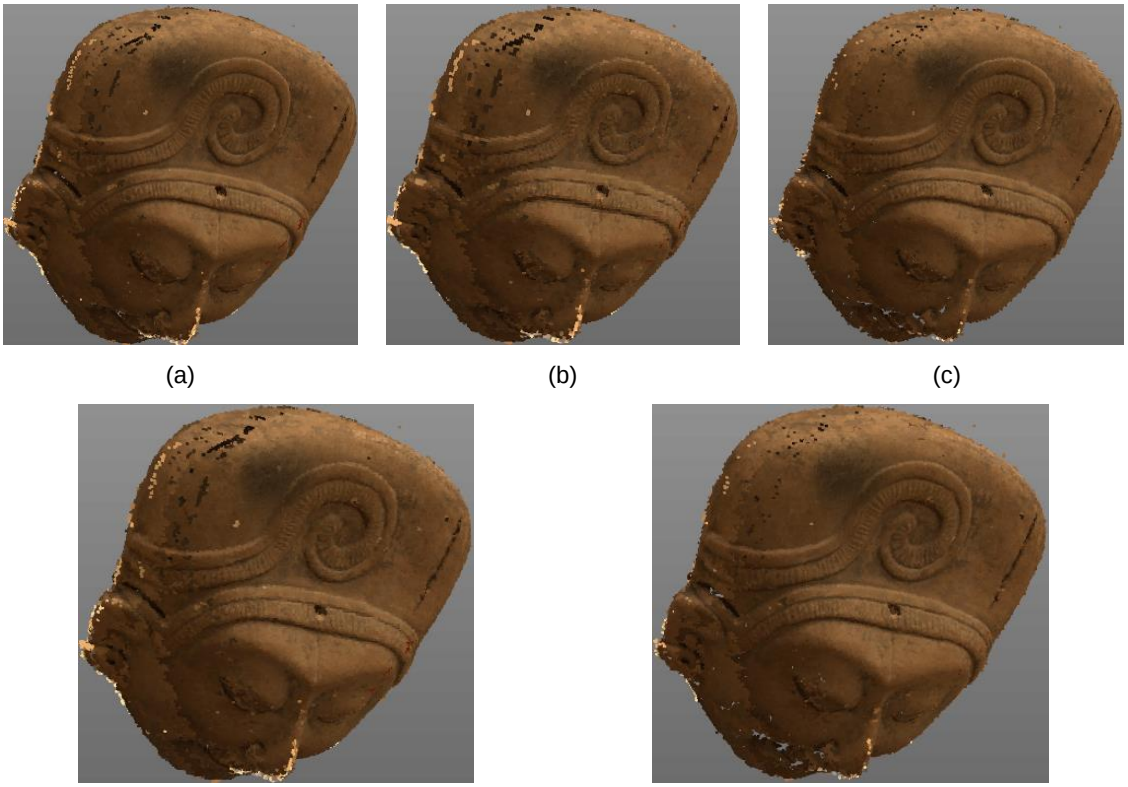
	Bjøntegaard-delta	**C2C**	**C2P**
Egyptian Mask	PSNR [dB]	3.63	3.37
	Bitrate [%]	-73.6	-61.9
Statue Klimt	PSNR [dB]	2.69	2.48
	Bitrate [%]	-50.6	-55.9
Loot	PSNR [dB]	5.46	4.72
	Bitrate [%]	-46.5	-43.9
Longdress	PSNR [dB]	4.43	3.55
	Bitrate [%]	-40.5	-45.8
Overall average	PSNR [dB]	4.05	3.53
	Bitrate [%]	-52.8	-51.9

From the charts and the results in Table 10, it is possible to conclude that the proposed GTGC solution brings substantial RD performance gains regarding the PCL solution. The average rate gains are around 50% for a similar quality which is a very substantial gain. Moreover, these gains are obtained with decoded point clouds including a significantly higher number of decoded points. However, it is also clear that the proposed GTGC solution RD performance saturates for the higher bitrates, which does not occur for the unrealistic graph transform solution. This behavior seems to indicate that the higher frequency coefficients are far from ideal and do not allow a continuous quality increase with the rate for the higher qualities. The largest gains happen for the *Egyptian Mask* point cloud what was somehow expected as the parameters optimization was made with that cloud. The worse gains were obtained for *Longdress* because this is the cloud with more geometry details, e.g. the hair and the dress.

4.6.2. Informal Subjective Assessment

To better understand the strengths and drawbacks of the proposed GTGC solution, some informal subjective assessment was performed for two different RD points, notably approximately 0.5 and 1 bpp for both the PCL and GTGC decoded point clouds with the main objective of performing some subjective comparison for the lower and higher qualities. The decoded clouds were rendered in the same way as previously done for the bitrate range definition, see Section 4.2.

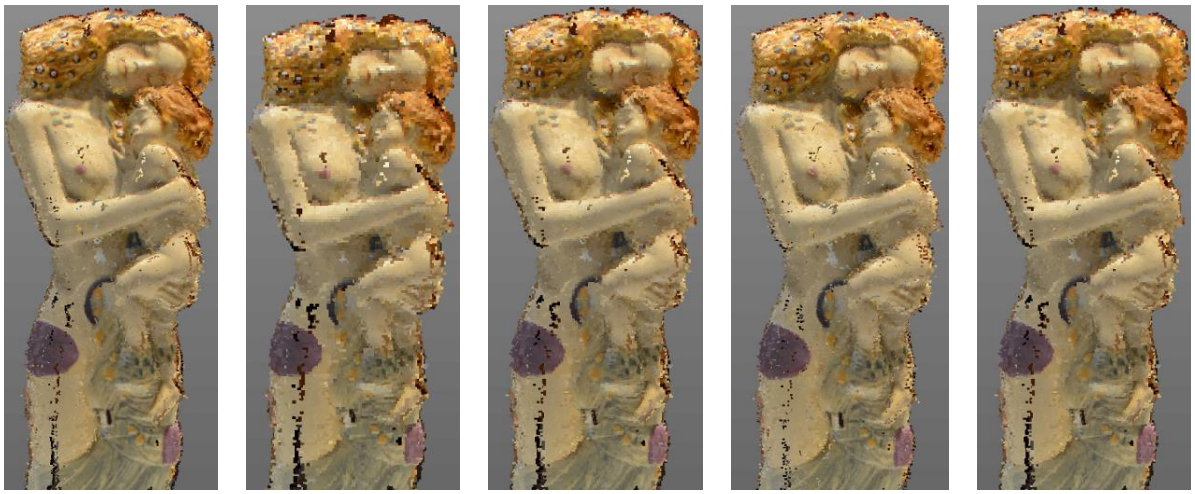
Figure 63 to Figure 66 shows the *Egyptian Mask, Statue Klimt, Loot and Longdress* point clouds for both the PCL and the GTGC solutions. While a) depicts the original point cloud, (b) and (c) shows the corresponding decoded clouds for approximately 0.5 bpp and (d) and (e) for approximately 1 bpp, first for PCL and after for the GTGC solution.



(a) (b) (c)

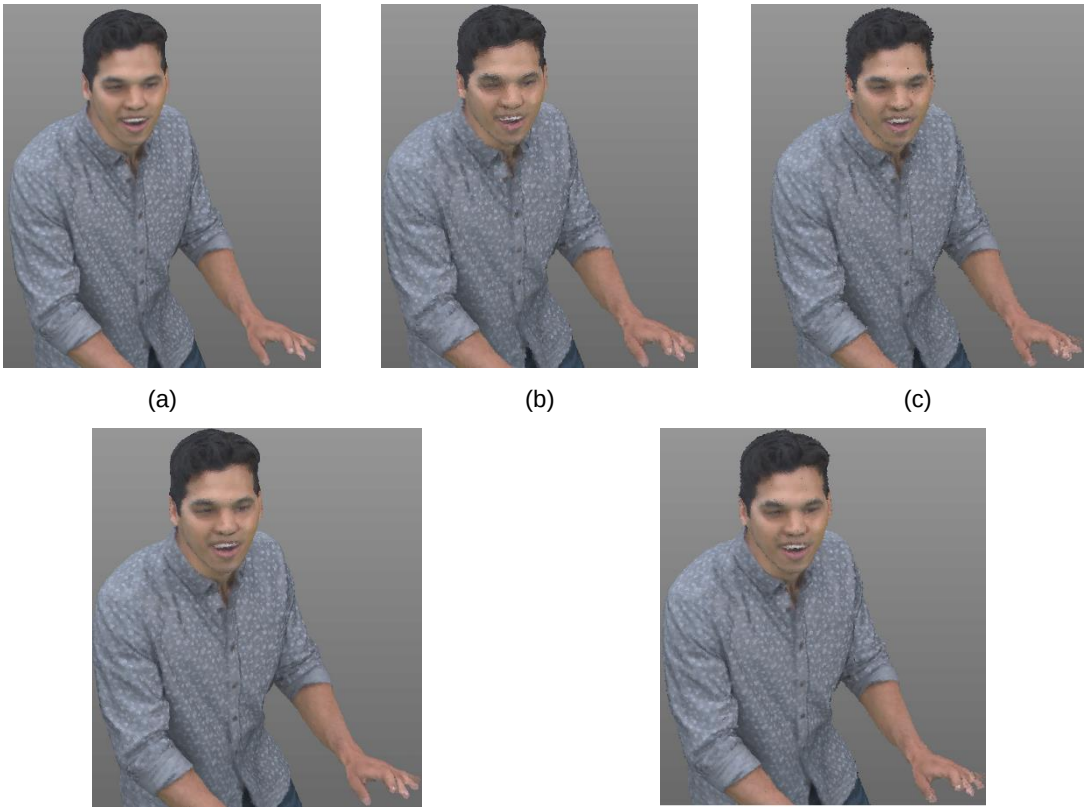
(d) (e)

Figure 63: *Egyptian Mask* point cloud: (a) Original; (b) PCL for approximately 0.5 bpp; (c) GTGC for approximately 0.5 bpp; (d) PCL for approximately 1 bpp; and (e) GTGC for approximately 1 bpp.



(a) (b) (c) (d) (e)

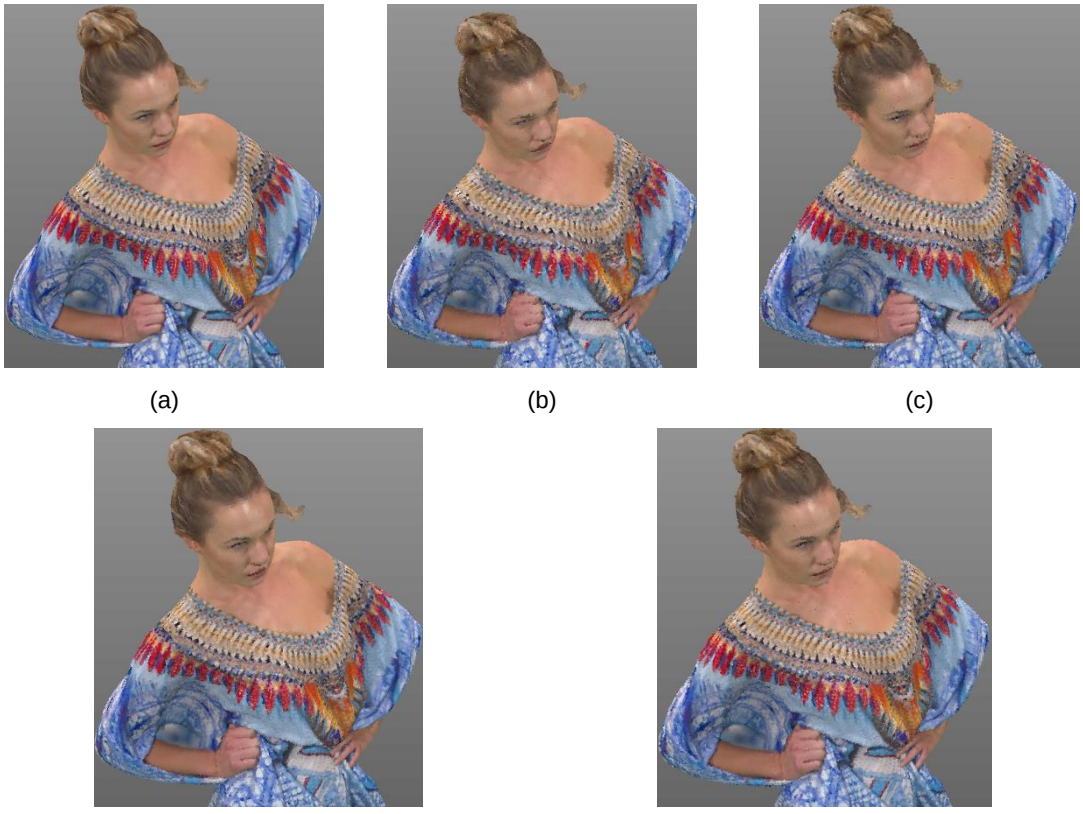
Figure 64: *Statue Klimt* point cloud: (a) Original; (b) PCL for approximately 0.5 bpp; (c) GTGC for approximately 0.5 bpp; (d) PCL for approximately 1 bpp; and (e) GTGC for approximately 1 bpp.



(a) (b) (c)

(d) (e)

Figure 65: *Loot* point cloud: (a) Original; (b) PCL for approximately 0.5 bpp; (c) GTGC for approximately 0.5 bpp; (d) PCL for approximately 1 bpp; and (e) GTGC for approximately 1 bpp.



(a) (b) (c)

(e) (d)

Figure 66: *Longdress* point cloud for: (a) Original; (b) PCL for approximately 0.5 bpp; (c) GTGC for approximately 0.5 bpp; (d) PCL for approximately 1 bpp; and (e) GTGC for approximately 1 bpp.

It is important to state that the informal subjective assessment was performed playing with clouds using the renderer software; from the figures presented above it is much harder to fully inspect the clouds quality. During the informal subjective assessment it was perceptible that all GTGC decoded clouds targeting lower bitrates, i.e. below 1 bpp, showed significantly higher quality when compared to the PCL ones. On the other hand, for higher bitrates, i.e. over 1 bpp, the GTGC advantages were not that clear; while some GTGC decoded clouds showed better qualities, e.g. *Longdress, Egyptian Mask* and *Statue Klimt*, for the *Loot* cloud the quality seemed lower when compared to the PCL decoded cloud, especially on the finger details. It is also important to state that during the informal subjective assessment a clustering effect was perceived, for all GTGC decoded point clouds; this effect is shown in Figure 67 for the *Egyptian Mask* decoded cloud with approximately 1 bpp. However, in the figures presented above it is not perceptible as the rendered point size is adjusted to fill the holes; this procedure is done for both the PCL and GTGC solution and has already been detailed in Section 4.2.

Figure 67: GTGC decoded *Egyptian Mask* cloud zoomed-in for approximately 1 bpp.

At this stage, the GTGC strengths and weaknesses can be outlined as:

1. **Strengths:** GTGC significantly outperforms the compression efficiency of the PCL coding solution, especially when targeting lower to medium point cloud qualities; it shows considerable gains in terms of the number of decoded points as it does not introduce the PCL downsampling effect; for all point clouds assessed, a rather low number of transform coefficients was enough to faithfully represent the surface of the point clouds with a number of decoded points rather similar to the original number.
2. **Weaknesses:** For higher rate, there is a noticeable saturation in terms of RD performance, notably due to the transform coefficients quality; there is a perceptible clustering effect, very similar to the blocking artifacts experienced in the DCT-based image coding solutions, that should be possible to overcome with proper post-processing tools (thus augmenting the GTGC solution performance); there is a clear drawback when representing local areas with small details, such as the fingers details on the *Loot* cloud.

In summary, the proposed GTGC solution is the first static point cloud geometric coding solution designed based on graph transforms. Its performance is very promising, thus deserving further research investment to address the identified weaknesses.

Chapter 5

5. Conclusions and Future Work

This final chapter presents the conclusions of this Thesis and suggests some ideas for future work. But before, it starts by listing the key contributions of this work.

5.1. Key Contributions and Conclusions

The target of this work was to design and propose a point cloud coding solution that exploits the graph transform to code geometry information of static point clouds. From the results obtained, it can be concluded that the proposed graph transform geometry coding (GTGC) solution is efficient and thus, the target of this work is achieved. Also, the RD performance evaluation shows that the GTGC brings significant improvements in comparison to the state-of-the-art, especially for application scenarios that require lower to medium point cloud quality. The main contributions of this Thesis are listed as follows:

- **Design**, **implementation** and **assessment** of a scalable point cloud codec with two layers targeting static point cloud geometry information. When only the BL is decoded, a coarser level of detail can be obtained; the EL provides higher quality levels.
- The GTGC solution proposes a novel **hybrid approach** using octree-based and graph-based coding. While the first is rather popular the latest has never been exploited for geometry coding of point clouds.
- The GTGC achieves **better performance** with respect to the state-of-the-art, which is the pure octree-based PCL coding solution; especially for lower to medium bitrates.

The RD performance gains of the GTGC solution are explained by the fact that the proposed graph transform learning procedure is very efficient when targeting low to medium qualities as the first transform coefficients are very representative of the original point cloud surface. On the other hand, for higher bitrates the performance gains are lower since the number of coefficients needed to represent the precise point location of the original point cloud is very high. Due to this fact, the RD performance seems to saturate as the coefficients bring low quality gain despite being expensive in terms of bitrate. Besides, the BL-EL points ratio (relationship between the number of points at the BL and EL layers) needs to increase to achieve higher qualities. This increases the overhead associated with the octree-base layer which makes the solution less efficient. It is also shown that when the graph transform basis are learned with the EL data (which is not possible to do in a realistic scenario) higher RD performance is achieved for high bitrate ranges.

From the informal subjective assessment performed, it is clear that the GTGC and PCL coding solutions introduce different coding artifacts: i) the GTGC introduces a clustering effect (similar to the

blocking artifacts experienced in standard DCT image codecs) that can be easily overcome with some suitable post-processing tool, which should allow even higher quality gains; ii) the PCL introduces a downsampling effect which is more clear for lower bitrates; after rendering, a low number of points leads to visible artifacts such as the appearance of the rendering primitives (e.g. cubes).

5.2. Future Work

Since point cloud geometry coding using graph-based transform is a relatively new topic, there are several interesting future directions that may lead to a coding solution with even higher performance. Considering that the proposed solution obtains lower RD performance gains (and in some cases losses) for higher bitrates, the following future work is proposed:

- **Adaptive EL Clustering:** This topic targets the improvement of the EL clustering module so that the clusters have characteristics that allow to achieve higher compression ratios. For example, the clustering can exploit the normals of the point cloud to group points and therefore obtain surfaces that can be efficiently represented with a small set of graph transform coefficients which allows the improvement of the proposed solution performance. Also, it can be beneficial to consider a clustering method adapted to the different densities of the point cloud, which means that the clustering approach may consider clusters with higher dimensions for higher density areas of the point cloud and vice-versa.

- **Inter-layer Matching Algorithm:** The objective of this topic is to improve the Inter-layer prediction module, thus lowering the geometry residuals energy to be coded at the enhancement layer. Let's assume that for each pair of U-BL and EL clusters, a bipartite graph is created with the points of the U-BL and EL clusters [103]; consider also, that each possible edge between the *i*th U-BL and all *j*th EL clusters points has an associated weight equal to their distance. To find the best match between points in the U-BL and EL layers, the Kuhn-Munkres algorithm [104], also known as Hungarian algorithm, can be used. This combinatorial optimization algorithm may solve this matching problem rather efficiently. In the end, the Inter-layer matching distance between corresponding points of the U-BL and EL layers would be minimized without the need to send signaling information to the decoder side.

- **Upsampling Algorithm:** This topic aims the enhancement of the BL upsampling module. With this target in mind, the current algorithm could be enhanced by considering the BL point cloud holes. Thus, two possible options could be pursued: i) using other surface reconstruction methods (such as marching cubes, surfel based geometry, etc.) more efficient than the current Poisson surface reconstruction; and ii) a suitable post-processing algorithm applied to the mesh surface obtained using the Poisson reconstruction method so that the vertices belonging to holes in the BL cloud are discarded. In this last case, the BL octree could be used to check if each point to be added to the upsampled point cloud also belongs to an occupied voxel of the BL octree. In both cases, a better Inter-layer prediction would be obtained and thus a lower Inter-layer residual could be coded.

To conclude, there is still room for significant improvements in the point cloud compression of geometry following the graph-based transform approach. Thus, it is expected that with more research

effort, significant efficiency improvements could be achieved to efficiently store and transmit point cloud information over bandwidth limited channels.

References

[1] F. Pereira and E. Silva, "Efficient plenoptic imaging representation: why do we need it?," in *IEEE international conference on multimedia and expo*, Seattle, Washington, United States, July 2016.

[2] "Lytro," [Online]. Available: https://www.lytro.com/. [Accessed 23 August 2016].

[3] "Velodyne LiDAR - HDL-32E," [Online]. Available: http://velodynelidar.com/hdl-32e.html. [Accessed 16 September 2017].

[4] "Youtube - First VR 3D virtual museum online," [Online]. Available: https://www.youtube.com/watch?v=bQcpZkJneO0. [Accessed 11 September 2017].

[5] "Musion 3D," [Online]. Available: http://www.musion3d.co.uk/melenchon-presidential-campaign-gaining-momentum-after-use-of-musion-telepresence-technology/. [Accessed 11 September 2017].

[6] "Zeiss - New benchmark in robot-based 3D in-line metrology," [Online]. Available: https://www.zeiss.nl/metrologie/over-ons/press-releases.html?id=ZEISS_AIMax_cloud. [Accessed 12 September 2017].

[7] "UCL 3DIMPact - Shipping galleries," [Online]. Available: http://www.ucl.ac.uk/3dimpact/projects/research-heritage/#shippinggalleries. [Accessed 16 September 2017].

[8] "Wikipedia on six degrees of freedom," [Online]. Available: https://en.wikipedia.org/wiki/Six_degrees_of_freedom. [Accessed 24 August 2016].

[9] "Light," [Online]. Available: http://www.light.co/camera. [Accessed 23 August 2016].

[10] "Velodyne LiDAR," [Online]. Available: http://velodynelidar.com/. [Accessed 23 August 2016].

[11] T. Maugey, A. Ortega and P. Frossard, "Graph-based representation for multiview image geometry.," *IEEE transactions on image processing,* vol. 17, no. 11, pp. 1573-1586, May 2015.

[12] "Point cloud library - estimating surface normals in a point cloud," [Online]. Available: http://pointclouds.org/documentation/tutorials/normal_estimation.php. [Accessed 2 October 2016].

[13] "GI science," [Online]. Available: http://k1z.blog.uni-heidelberg.de/files/2014/03/blogcompilations-1.png. [Accessed 26 July 2016].

[14] "Autodesk 3DS max 2014 - point cloud object," [Online]. Available: http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-Help/index.html?url=files/GUID-49CE0ACB-1345-4D50-B6E5-361DBFDB5B33.htm,topicNumber=d30e158270. [Accessed 6 October 2016].

[15] P. Frossard, D. Thanou and P. Chou, "Graph-based compression of dynamic 3D point cloud sequences," *IEEE transactions on image processing,* vol. 25, no. 4, pp. 1765-1778, April 2016.

[16] J. Peng, C. S. Kim and C. C. Jay Kuo, "Technologies for 3D mesh compression: a survey.," *Journal of visual communication and image representation,* vol. 16, no. 6, pp. 688-733, December 2005.

[17] G. Stockman and L. G. Shapiro, "3D models and matching," in *Computer vision*, Washington, United States, Prentice Hall, 2001, pp. 517-571.

[18] "IEEE image repository," [Online]. Available:

http://ieeexplore.ieee.org/ieee_pilot/articles/01/ttg2009010006/assets/img/article_1/fig_12/large.gif. [Accessed 26 July 2016].

[19] "Holophile," [Online]. Available: http://www.holophile.com/about.htm. [Accessed 26 July 2016].

[20] "Hyper physics," [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/optmod/imgopm/chesr21.jpg. [Accessed 26 July 2016].

[21] "Computer graphics," [Online]. Available: http://graphics.tu-bs.de/static/people/magnor/thumbs/bunny_CGH.png. [Accessed 26 July 2016].

[22] "Wikipedia on geographic information systems," [Online]. Available: https://en.wikipedia.org/wiki/Geographic_information_system. [Accessed 9 September 2016].

[23] "Wikipedia on cultural heritage," [Online]. Available: https://en.wikipedia.org/wiki/Cultural_heritage. [Accessed 9 September 2016].

[24] "The free dictionary on telepresence," [Online]. Available: http://www.thefreedictionary.com/telepresence. [Accessed 10 September 2016].

[25] "Wikipedia on telepresence," [Online]. Available: https://en.wikipedia.org/wiki/Telepresence. [Accessed 10 September 2016].

[26] "Wareable - the best VR headsets," [Online]. Available: http://www.wareable.com/headgear/the-best-ar-and-vr-headsets. [Accessed 19 September 2016].

[27] "Oculus Rift," [Online]. Available: https://www3.oculus.com/en-us/gear-vr/. [Accessed 23 August 2016].

[28] "Microsoft Hololens," [Online]. Available: https://www.microsoft.com/microsoft-hololens/en-us/hardware. [Accessed 23 August 2016].

[29] F. Pereira, Comunicações audiovisuais: tecnologias, normas e aplicações, Lisbon, Portugal: IST Press, July 2009.

[30] "Microsoft," [Online]. Available: https://www.microsoft.com/en-us/research/project/holoportation-3/. [Accessed 30 July 2016].

[31] "Point cloud data in VR design studio," [Online]. Available: https://www.youtube.com/watch?v=aI21rZf0OZs. [Accessed 30 July 2016].

[32] Y. Zhang, H. Lv, Y. Liu, H. Wang, X. Wang and Q. Huan, "Light field depth estimation via epipolar plane image analysis and locally linear embedding," *IEEE transactions on circuits and systems for video technology,* vol. PP, no. 99, pp. 1-1, April 2016.

[33] "Point cloud library - walkthrough," [Online]. Available: http://pointclouds.org/documentation/tutorials/walkthrough.php. [Accessed 21 September 2016].

[34] "Texas Intruments - Time of flight camera: an introduction," [Online]. Available: http://www.ti.com/lit/wp/sloa190b/sloa190b.pdf. [Accessed 1 September 2016].

[35] "Spectrum IEEE," [Online]. Available: http://spectrum.ieee.org/tech-talk/semiconductors/optoelectronics/mit-lidar-on-a-chip. [Accessed 1 September 2016].

[36] "Wikipedia on beam steering," [Online]. Available: https://en.wikipedia.org/wiki/Beam_steering. [Accessed 21 September 2016].

[37] "Velodyne LiDAR - Puck," [Online]. Available: http://velodynelidar.com/docs/datasheet/63-9286_Rev-D_Puck%20LITE_Spec%20Sheet_Web.pdf. [Accessed 7 September 2016].

[38] "Creaform3D - an introduction to 3D scannning," [Online]. Available: http://www.creaform3d.com/sites/default/files/assets/technological-fundamentals/ebook1_an_introduction_to_3d_scanning_en_26082014.pdf. [Accessed 8 September 2016].

[39] "Wikipedia on structured light 3D scanner," [Online]. Available: https://en.wikipedia.org/wiki/Structured-light_3D_scanner. [Accessed 8 September 2016].

[40] C. Wang, Z. Zhu, S.-C. Chan and . H.-Y. , "Real-time depth image acquisition and restoration," *Journal signal processing systems,* vol. 79, no. 1, pp. 1-18, April 2015.

[41] "3D time of flight cameras," [Online]. Available: https://larrylisky.com/2013/12/07/fun-with-3d-time-of-flight-cameras/. [Accessed 23 August 2016].

[42] "Velodyne LiDAR products," [Online]. Available: http://velodynelidar.com/hdl-64e.html. [Accessed 23 August 2016].

[43] "Microsoft developer network," [Online]. Available: https://msdn.microsoft.com/en-us/library/jj131033.aspx. [Accessed 6 September 2016].

[44] N. Jianhui, H. Ying, M. Zi and L. Qingmeng, "Online colored point cloud acquisition by reprojection," in *International conference on intelligent systems design and engineering application*, Sanya, China, January 2012.

[45] "Point cloud library - registration API," [Online]. Available: http://pointclouds.org/documentation/tutorials/registration_api.php. [Accessed 21 September 2016].

[46] S. Orts, C. Rhemann and S. R. Fanello, "Holoportation: virtual 3D teleportation in real-time," in *29th ACM User interface software and technology symposium*, Tokyo, Japan, October 2016.

[47] "PMD technology," [Online]. Available: http://pmdtec.com/picoflexx/downloads/PMD_RD_Brief_CB_pico_flexx_V0201.pdf. [Accessed 7 September 2016].

[48] "Point grey," [Online]. Available: https://www.ptgrey.com/bumblebee2-stereo-vision-08-mp-color-firewire-1394a-6mm-sony-icx204-camera. [Accessed 6 September 2016].

[49] K. Pulli, H. Hoppe, M. Cohen, L. Shapiro, T. Duchamp and W. Stuetzle, "View-based rendering: visualizing real objects from scanned range and color data," in *Rendering techniques*, Vienna, Springer, 1997, pp. 23-34.

[50] P. Heckbert and M. Garland, "Multiresolution modeling for fast rendering," in *Proceedings of graphics interface*, Alberta, Canada, May 1994.

[51] S. Rusinkiewicz and M. Levoy, "QSplat: A multiresolution point rendering system for large meshes," in *Proceedings of ACM SIGGRAPH 2000*, New Orleans, USA, July 2000.

[52] "Meshlabstuff - meshing point clouds," [Online]. Available: http://meshlabstuff.blogspot.pt/2009/09/meshing-point-clouds.html. [Accessed 14 October 2016].

[53] M. Botsch, A. Wiratanaya and L. Kobbelt, "Efficient high quality rendering of point sampled geometry," in *Proceedings of the 13th eurographics workshop on rendering*, Pisa, Italy, July 2002.

[54] E. Gobbetti and F. Marton, "Layered point clouds," *Proceeding of the eurographics symposium on point-based graphics,* vol. 28, no. 6, pp. 113-120, June 2004.

[55] R. Pintus, E. Gobetti and M. Agus, "Real-time rendering of massive unstructured raw point clouds using screen-space operators," in *IEEE visual analytics science and technology*, Rhode Island, USA, October 2011.

[56] "Microsoft - what is view frustrum?," [Online]. Available: https://msdn.microsoft.com/en-us/library/ff634570.aspx. [Accessed 26 September 2016].

[57] "Slideplayer - visibility culling slideshow from Ohio Sate university," [Online]. Available: http://slideplayer.com/slide/5909768/. [Accessed 8 October 2016].

[58] "Institute of data analysis and visualization - depth buffer," [Online]. Available: http://idav.ucdavis.edu/education/GraphicsNotes/. [Accessed 26 October 2016].

[59] M. Wimmer and C. Scheiblauer, "Instant points: fast rendering of unprocessed point clouds," in *Symposium on point-based graphics*, Massachusetts, USA, July 2006.

[60] C. Dachsbacher, C. Vogelgsang and M. Stamminger, "Sequential point trees," *ACM transactions on graphics,* vol. 22, no. 3, pp. 657-662, July 2003.

[61] R. Mekuria, Z. Li and C. Tulvan, "Evalutation criteria for PCC (point cloud compression)," in *MPEG 115*, Geneva, Switzerland, February 2016.

[62] D. Tian, H. Ochimizu, C. Feng, R. Cohen and A. Vetro, "Evaluation metrics for point cloud compression," in *JPEG 73rd meeting*, Chengdu, China, October 2016.

[63] "Call for proposals for point cloud compression," in *ISO/IEC JTC1/SC29 WG11*, Hobart, Australia, April 2017.

[64] "Mathworks - Hausdorff distance," [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/26738-hausdorff-distance. [Accessed 7 October 2016].

[65] "Wordpress - the infinite loop," [Online]. Available: https://geidav.files.wordpress.com/2014/01/aabb_vs_ombb.png?w=820. [Accessed 9 October 2016].

[66] "Wikipedia on Octree," [Online]. Available: https://en.wikipedia.org/wiki/Octree. [Accessed 25 October 2016].

[67] T. Ochotta and D. Saupe, "Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields," *Proceedings of the eurographics symposium on point-based graphics,* vol. 28, no. 6, pp. 103-112, June 2004.

[68] C. Zhang, D. Florêncio and C. Loop, "Point cloud attribute compression with graph transform," in *IEEE international conference on image processing*, Paris, France, October 2014.

[69] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz and E. Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE internation conference on robotics and automation*, St. Paul, Minnesota, USA, May 2012.

[70] "Basicofy - Root node image," [Online]. Available: http://www.basicofy.com/wp-content/uploads/2016/09/rootnode.png. [Accessed 4 November 2016].

[71] G. Nigel and N. Martin, "Range encoder range encoding: an algorithm for removing redundancy from a digitalized message," in *Proceedings of the Vvdeo and data recording conference*, Southampton, UK, July 1979.

[72] R. Mekuria, K. Blom and P. Cesar, "Design, implementation and evaluation of a point cloud codec for 3D tele-immersive video," *IEEE transactions on circuits and systems for video technology,* vol. 27, no. 4, pp. 828-842, April 2017.

[73] "Wikipedia on depth first search," [Online]. Available: https://en.wikipedia.org/wiki/Depth-first_search. [Accessed 20 November 2016].

[74] "Visual computing labs - reconstruction," [Online]. Available: http://vcl.iti.gr/reconstruction/. [Accessed 3 November 2016].

[75] C. Zhang and D. Florêncio, "Analyzing the optimality of predictive transform coding using graph-based models," *IEEE signal processing letters,* vol. 20, no. 1, pp. 106-109, January 2013.

[76] R. A. Cohen, D. Tian and A. Vetro, "Attribute compression for sparse point clouds using graph transforms," in *2016 IEEE international conference on image processing*, Phoenix, Arizona, USA, September 2016.

[77] T. Sikora and B. Makai, "Shape-adaptive DCT for generic coding of video," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 5, no. 1, pp. 59-62, February 1995.

[78] V. Scurtu, C. Tulvan, A. Gabrielli and M. Preda, "Reconstruction platform and datasets for 3D point cloud compression," in *document MPEG2015/m36523, ISO/IEC JTC1/SC29/WG11, coding of moving pictures and audio*, Warsaw, Poland, June

2015.

[79] R. A. Cohen, D. Tian and A. Vetro, "Point cloud attribute compression using 3D intra prediction and shape-adaptive transforms," in *Proceedings 2016 Data Compression Conference*, Snowbird, UT, USA, March-April 2016.

[80] T. Golla and R. Klein, "Real-time point cloud compression," in *International conference on intelligent robots and systems*, Hamburg, Germany, September-October 2015.

[81] A. Cohen, I. Daubechies and J. C. Feauveau, "Biorthogonal based of compactly supported wavelets," *Communications on pure and applied mathematics,* vol. 45, no. 5, pp. 485-560, June 1992.

[82] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer e W. J. Rucklidge, “The emerging jbig2 standard,” *IEEE transactions on circuits and systems for video technology,* vol. 8, nº 7, pp. 838-848, November 1998.

[83] "7-zip," [Online]. Available: http://7-zip.org/. [Accessed 24 November 2016].

[84] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE transactions on information theory,* vol. 23, no. 3, pp. 337-343, May 1977.

[85] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A benchmark for the evaluation of rgd-d slam systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, October 2012.

[86] M. Ruhnke, L. Bo, D. Fox and W. Burgard, "Hierarchical sparse coded surface models based on sparse coding," in *IEEE international conference on robotics and automation*, Hong Kong, China, May-June 2014.

[87] J. Digne, R. Chaine and S. Valette, "Self-similarity for accurate compression of point sampled surfaces," *Computer Graphics Forum,* vol. 33, no. 2, pp. 155-164, May 2014.

[88] E. Hubo, T. Mertens, T. Haber e P. Bekaert, “Self-similarity based compression of point set surfaces with application to ray tracing,” *Computer & Graphics,* vol. 32, nº 2, pp. 221-234, April 2008.

[89] A. Kalaiah e A. Varshney, “Statistical geometry representation for efficient transmission and rendering,” *ACM Transactions on Graphics,* vol. 24, nº 2, pp. 348-373, April 2005.

[90] R. Schnabel, S. Möser e R. Klein, “Fast vector quantization for efficient rendering of compressed point-clouds,” *Computers & Graphics,* vol. 32, nº 2, pp. 246-259, April 2008.

[91] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proceedings of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, Anchorage, Alaska, USA, May 2010.

[92] M. Ruhnke, L. Bo, D. Fox and W. Burgard, "Compact rgbd surface models based on sparse coding," in *Twenty-seventh AAAI conference on artificial intelligence*, Bellevue, Washington, USA, July 2013.

[93] M. Kazhdan, M. Bolitho and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth eurographics symposium on geometry processing*, Cagliari, Sardinia, Italy, June 2006.

[94] "CloudCompare - 3D point cloud and mesh processing software," [Online]. Available: http://www.cloudcompare.org/. [Accessed 3 Obtober 2017].

[95] "Rosetta code - k-means++ clustering," [Online]. Available: rosettacode.org. [Accessed 3 Obtober 2017].

[96] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Eighteenth annual ACM-SIAM symposium on discrete algorithms*, New Orleans, Louisiana, USA, January 2007.

[97] J. S. Gary and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE signal processing magazine,* vol. 15, no. 6, pp. 74-90, July 1998.

[98] "Algorithmist - Monotone chain convex hull," [Online]. Available: http://www.algorithmist.com/index.php/Monotone_Chain_Convex_Hull. [Accessed 29 June 2017].

[99] I. H. Witten, R. M. Neal and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM,* vol. 30, no. 6, pp. 520-540, June 1987.

[100] "Wikipedia on Laplace distribution," [Online]. Available: https://en.wikipedia.org/wiki/Laplace_distribution. [Accessed 5 August 2017].

[101] "MPEG point cloud datasets," [Online]. Available: http://mpegfs.int-evry.fr/MPEG/PCC/DataSets/pointCloud/CfP. [Accessed 17 August 2017].

[102] "Technicolor point cloud rendered," in *ISO/IEC JTC1/SC29 WG11*, Hobart, Australia, April 2017.

[103] "Wolfram MathWorld - bipartite graph," [Online]. Available: http://mathworld.wolfram.com/BipartiteGraph.html. [Accessed 5 October 2017].

[104] "The Hungarian algorithm for weighted bipartite graphs," [Online]. Available: https://www.cc.gatech.edu/~rpeng/18434_S15/hungarianAlgorithm.pdf. [Accessed 5 October 2017].