Adaptive Plane Projection for Video-based Point Cloud Coding

Eurico Manuel Rodrigues Lopes

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Dr. Fernando Manuel Bernardo Pereira

Prof. Dr. João Miguel Duarte Ascenso

Examination Committee

Chairperson: Prof. Dr. José Eduardo Charters Ribeiro da Cunha Sanguino

Supervisor: Prof. Dr. João Miguel Duarte Ascenso

Member of the Committee: Prof. Dr. Nuno Miguel Morais Rodrigues

November 2018

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank all my family and friends that have supported me throughout this degree with a special mention to my parents, for all the financial and emotional support they have provided me during these 5 years. I'd also like to thank my girlfriend for her love, and for her support and motivation throughout the most difficult challenges we have both surpassed together.

I would also like to thank my supervisors for their guidance, orientation and lessons that I take for life: Prof. Fernando Pereira, Prof. João Ascenso and Prof. Catarina Brites. Lastly, I'd like to thank to my colleagues at Instituto de Telecomunicações, namely Afonso Costa and Antoine Dricot, for the many discussions we have held, and for the help they have given me during the development of the Thesis.

Without these people, achieving this degree would never be possible.

Resumo

A recente evolução dos sistemas de aquisição 3D e dos métodos de processamento deste tipo de informação levaram ao desenvolvimento de novos modelos de representação 3D, nomeadamente *point clouds*, *holographic imaging* e *light fields*.

Uma *point cloud* é um conjunto de pontos com uma localização 3D associada, e opcionalmente alguns atributos, tais como cor, refletência e normais. Este modelo de representação viabiliza um novo conjunto de aplicações, desde a telepresença 3D a mapas 3D de larga escala. Contudo, a elevada imersão e o realismo acarretam consigo altos débitos binários, e por isso, soluções de codificação eficientes são essenciais. Como consequência, o MPEG lançou um projeto de normalização de soluções de codificação eficiente de *point clouds*. Este projeto deu origem a duas soluções de codificação: o Video-based Point Cloud Coding (V-PCC) para conteúdo dinâmico e o Geometry-based Point Cloud Coding (G-PCC) para conteúdo estático.

O objetivo desta Dissertação é o desenho, implementação e avaliação de uma solução de codificação que seguindo a estrutura do MPEG V-PCC seja capaz de melhorar o seu desempenho na codificação de point clouds estáticas (modo intra). Assim, este trabalho começa por abordar o estado-da-arte na área das *point clouds*, com especial foco nos conceitos mais relevantes, bem como nas soluções de codificação mais relevantes na literatura, incluindo as duas soluções de codificação do MPEG. A solução de codificação proposta segue a estrutura do MPEG V-PCC mas inova ao adaptar os planos de projeção às características específicas de cada point cloud. A análise de resultados demonstra um melhor desempenho de geometria relativamente ao MPEG V-PCC, especialmente para médias e altas qualidades.

Palavras-chave: Modelos de representação visual, sistemas de aquisição 3D, point clouds , soluções de codificação, *point clouds* estáticas.

Abstract

In recent years, the evolution of 3D capture systems and processing methods have unleashed the breakthrough of new 3D representation models, namely point clouds, holographic imaging and light fields.

A point cloud is a set of points with an associated 3D location and, optionally, some attributes, such as colour, reflectance, and normals. This visual representation model enables a more complete, realistic and immersive representation of the visual world, which allows a whole new set of applications, from 3D telepresence to large-scale 3D maps. However, the immersion and realism of the point cloud representation model comes at the cost of a very high bitrate, and thus efficient coding solutions are essential. As a consequence, MPEG has launched a standardization project on efficient point cloud coding solutions. This project has given rise to two coding solutions, notably Video-based Point Cloud Coding (V-PCC) to encode dynamic content and Geometry-based Point Cloud Coding (G-PCC) to encode static content.

This Thesis targets the design, implementation and assessment of a static point cloud coding solution following the MPEG V-PCC framework with increased compression performance. To achieve this ultimate goal, this Thesis starts by addressing the state-of-art on point clouds, with a special focus on the main relevant concepts and the most relevant coding solutions in the literature, including the two MPEG coding solutions. The coding solution follows MPEG V-PCC and innovates by adapting the projection planes to the specific characteristics of each point cloud. The assessment results show a solid improvement on the geometry performance with respect to the MPEG V-PCC, especially for medium to high rates.

Keywords: Visual representation models, 3D acquisition systems, point clouds, coding solutions, static point clouds.

Table of Contents

List of Figures

List of Tables

List of Acronyms

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
3 DoF	Three Degrees of Freedom
6 DoF	Six degrees of Freedom
7D	Seven Dimensional
AR	Augmented Reality
AP2V-PCC	Adapted Plane Projection for V-PCC
bpv	Bits per vertex
bpp	Bits per point
CAVE	Cave Automatic Virtual Environment
CfP	Call for Proposals
CTC	Common Test Conditions
DCT	Discrete-Cosine Transform
DPCM	Differential Pulse-Coded Modulation
DSIS	Double Stimulus Impairment Scale
EF	Expansion Factor
GOF	Group of Frames
HEVC	High Efficiency Video Coding
HMD	Head Mounted Display
IOU	Intersection over Union
JPEG	Joint Photographic Experts Group
K-NN	K-Nearest Neighbours
LiDAR	Light Detection and Ranging
LOD	Level of Detail
MCIC	Motion-Compensated Intra-Frame Coding
MMS	Mobile Mapping System
MPEG	Moving Picture Experts Group
MSE	Mean-Squared Error
MV	Motion Vector
P2Plane	Point-to-Plane
P2Point	Point-to-Point
PCA	Principal Components Analysis

PCADP	Point Cloud Average Dot Product
PCC	Point Cloud Compression
PCL	Point Cloud Library
PSNR	Peak-Signal to Noise Ratio
QoE	Quality of Experience
QP	Quantization Point
RAHT	Region-Adaptive Hierarchical Transform
RD	Rate Distortion
RGB	Red Green Blue
RGB-D	Red Green Blue - Depth
RLGR	Run-Length/Golomb-Rice
SAR	Synthetic -Aperture Radar
SGW	Spectral Graph Wavelets
SNR	Signal to Noise Ratio
TM	Test Model
ToF	Time-of-Flight
TriSoup	Triangle Soup
VR	Virtual Reality
V-PCC	Video-based Point Cloud Coding
YUV	Luminance and Chrominances

Chapter 1

1. Introduction

This chapter will introduce the topic of this work: Adaptive Plane Projection for Video-based Point Cloud Coding. To achieve this purpose, it will start by providing the context, motivation and objective of the work to present after the structure of this report.

1.1. Context and Motivation

Multimedia is a large field of knowledge including the content and technologies related to the combination of different content types/media, such as text, audio, images, videos, animations, etc. as well as the sensors, codecs and displays capturing, compressing and presenting information, respectively in the context of multiple applications. As the acquisition, representation and display models have been defined by the available cameras, displays and the available computational power, the adopted representation model for visual information has been traditionally a rather simple 2D model, consisting in rectangular sets of samples in a specific colour space, e.g. RGB primaries or luminance and chrominances (YUV). With the intention of providing more immersive user experiences, the image/frame spatial resolutions, the sample depth and the frame rate have been steadily increasing already in the digital era [1]. However, the 2D frame-based model has obvious limitations, as it does not offer a convincing enough replica of the real scene/world.

To achieve more realistic and immersive experiences, 3D representation models have gained popularity in multiple applications, as they provide additional information about the visual scene beyond the limited 2D texture. The most popular 3D representation model considers two different views as a stereo pair to provide stereo parallax, typically by asking the user to wear a pair of glasses. This solution has stagnated in recent years, since it did not provide the expected Quality of Experience (QoE). Notably, the immersion is rather poor since the user motion degrees of freedom (DoF) are limited and the use of glasses is sometimes associated with visual discomfort.

With the emergence of inexpensive consumer electronic systems for 3D capture and rendering, and the recent advances in 3D data acquisition systems, new 3D data representation models are emerging, namely point clouds, light fields and holographic imaging. Among these more recent 3D data models, point clouds are one of the most popular solutions. A point cloud is a set of points defined by their 3D coordinates (x,y,z), corresponding to the so-called *geometry*, which may be complemented with attributes, such as colour, normal vectors and reflectance, among others. Unlike traditional 2D video, where the position of each pixel is defined in a fixed grid and only its colour is to be coded, point cloud coding involves coding at least its (irregular) geometry, and optionally some attributes. Alike 2D imaging, point clouds coding may target still frames, i.e. static point clouds, or a sequence of point cloud frames, i.e. dynamic point clouds. This type of 3D model is able to efficiently represent the data while providing

the functionalities required notably the much ambitioned 6-DoF immersion where the user may look around as in the real world, this means exploiting the 3 translational directions and the 3 rotational axis. This is desirable for many applications such as geographic information systems, cultural heritage, immersive telepresence, etc. notably when object-based interaction is important. Some of the point cloud test sets used in the literature are presented in Figure 1. Moreover, point clouds may also enable a whole new set of applications, notably free-viewpoint video, advanced content manipulation (e.g. refocusing or relighting), natural user interaction (e.g. controlling devices by gestures) and interactive gaming.

Figure 1 – Typical point cloud sequences used for coding performance assessment; From left to right: 'Man'[2], 'Ricardo' (or 'Upper-Body') [3] 'Yellow Dress' [2] and 'Loot' [4].

As 3D data acquisition solutions can produce 3D+t spatio-temporal representation models with millions of 3D points per second, it is critical to use efficient point cloud coding solutions to enable the targeted applications considering the resources available. It is important to note that even though a considerable amount of work has been devoted to the coding of static point clouds, less work has been invested in the field of dynamic point clouds.

Recognizing the industry need to have efficient coding solutions, which should enable the storage and transmission of the previously mentioned forms of 3D data, JPEG and MPEG have launched standardization projects to specify new, appropriate coding formats. JPEG has launched the so-called JPEG PLENO project [5], which targets the development of a standard framework for the capture, representation and exchange of new imaging modalities such as 360 images, depth enhanced images, light fields, point clouds and holographic imaging. On the other hand, MPEG has issued, in January 2017, a Call for Proposals on Point Cloud Compression (PCC) [6], which targets the efficient representation of static objects and scenes, as well as dynamic and real-time environments, represented using point clouds. The development process which followed the Call for Proposals on Point Cloud Compression has given rise to two different coding solutions, notably Video-based Point Cloud Coding (V-PCC) for dynamic point clouds and Geometry-based Point Cloud Coding (G-PCC) for static point clouds. The V-PCC solution is based on the projection of the depth and texture data on some pre-defined projection planes to code after the projected maps using an advanced video coding solution.

1.2. Objectives and Structure of the Thesis

The final objective of this Thesis is the design, implementation and assessment of a lossy coding solution that improves the MPEG V-PCC RD performance for static point clouds by using projection

planes adapted to the point cloud characteristics. To achieve the final objective of the Thesis, the work was divided into three fundamental steps:

- **Revision of the State-of-the-Art –** The related concepts and most relevant solutions in the point cloud coding literature were reviewed, including V-PCC and G-PCC, to understand the point cloud coding landscape;
- **Design and Implementation –** The proposed static point cloud coding solution was designed and implemented, starting from the V-PCC architectural framework and software, and adding or changing some of its tools to achieve a better compression performance;
- **Performance Assessment –** The proposed static point cloud coding solution was evaluated with the MPEG metrics and benchmarked with both V-PCC and G-PCC.

Considering the mentioned objectives, this Thesis is structured as follows after this first chapter:

- **Chapter 2** – The state-of-the-art related to point cloud coding is reviewed, notably the main concepts, processing pipeline architecture, use cases as well as the most relevant coding solutions and test conditions, metrics and benchmarks by which the solutions are typically objectively and subjectively evaluated;
- **Chapter 3** – This chapter presents the MPEG standard named as Video-based Point Cloud Coding (V-PCC) in full detail, as this is the framework that the novel coding solution follows;
- **Chapter 4 –** This chapter presents the proposed point cloud coding solution architecture and its main tools;
- **Chapter 5 –** In this chapter, the performance of the proposed coding solution is assessed, and a benchmarking study is done regarding V-PCC and G-PCC;
- **Chapter 6 –** This chapter outlines the conclusions and proposes future steps to further develop the work described in this Thesis.

Chapter 2

2. Point Cloud Coding: Basics and Main Coding Solutions

This chapter will review the state-of-the-art on point clouds, notably the main concepts, end-to-end architecture, use-cases, codecs evaluation criteria, the most relevant coding solutions on the literature and the results from the Call for Proposals issued by MPEG on January 2017 [6].

2.1. Main Concepts

To reach a better understanding of the most important concepts related to 3D visual representation used in this report, this section provides some basic definitions for these main concepts.

- **Plenoptic Function** - Function representing the intensity of light seen from any point in space (V_x, V_y, V_z), for any direction (θ, φ), for each wavelength (λ) and for each instant in time (t), which may be expressed as:

$$P = P(\theta, \varphi, \lambda, t, Vx, Vy, Vz) \quad (1)$$

This function may also be parameterized with cartesian coordinates (x,y) instead of spherical coordinates as it is usually done in Computer Graphics [7]. This way, the function takes the form:

$$P = P(x, y, \lambda, t, Vx, Vy, Vz) \quad (2)$$

To visualize a graphical representation of the plenoptic function, see Figure 2. As this is a 7-dimensional function, it is in practice very important to reduce its dimensionality and to sample it in an efficient way to minimize the amount of data to store or to transmit when representing the relevant visual information.

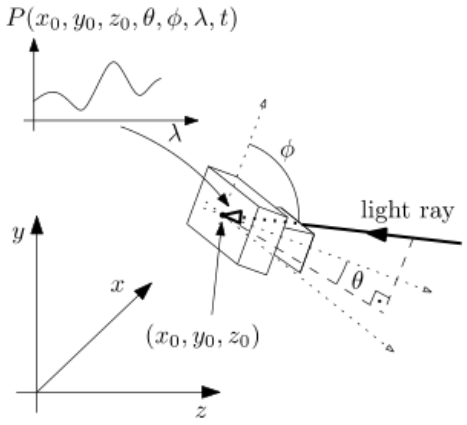


Figure 2- Graphical representation of the plenoptic function in spherical coordinates; (x_0,y_0,z_0) correspond to a specific viewing point in space [1].

- **Field of Light** - Set of directional light rays propagating from objects and filling the space; these light rays allow the Humans to see the world around them; this field can be mathematically represented by the Plenoptic Function [1].

- **Light Field Representation Model** – One of the ways to represent the field of light, notably its radiance. This type of model captures the field of light using multiple projections into 2D planar sensors. The most common practical light field acquisition processes involve acquisition with a lenslet camera, which includes an array of microlenses (narrow baseline light field data), or a multi-camera array (wide baseline light field data), see below. In addition to these two acquisition methods, Light field data can also be synthetically generated using mathematical models [5].
- **Lenslet Light Field Imaging** – Light field acquisition/representation model using a lenslet camera acquiring a 2D array of so-called micro-images, obtained using an array of microlenses placed in front of the main sensor. As the microlens array is placed in the optical path of a conventional monocular camera, the final lenslet light field image provides directional information for each sampled position [1]; in practice, each micro-image corresponds to a low resolution 2D image taken from a slightly different perspective. An example of a lenslet light field image is included in Figure 3 left).
- **Multi-Camera Array Light Field Imaging** – Light field acquisition/representation model using a high density, wide range, bi-dimensional array of monocular cameras, which captures a corresponding array of 2D views, possibly each of high resolution. The set of views may provide both horizontal and vertical parallaxes (or only one of them, typically only horizontal) with a camera linear or arc arrangement [1]. Parallax refers to the effect whereby closer objects seem to shift more than farther objects as the user's point of view changes; when a single view is used, no parallax effect is offered to the user as he/she sees the same information whatever his/her point of view. Stereo parallax is very well known as it corresponds to use two views to offer each eye a different perspective. Figure 3 right) also shows an example of a multi-camera array image.

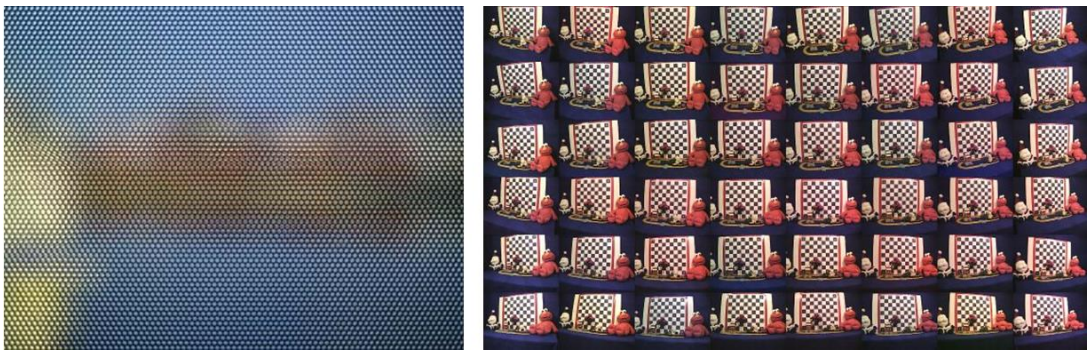


Figure 3 - left) Example of a lenslet light field image [8]; right) Example of a multi-camera array light field image [9].

- **Point Cloud Representation Model** – Alternative way of representing the field of light by using a collection of points in a 3D coordinate system, usually (*x,y,z*), that are intended to represent the external surface of one or more objects in a scene. Each point is characterized by at least the information corresponding to its position, designated as geometry information. The geometry information may be complemented with attributes for each point, such as colour, normal, etc. to achieve a more complete and powerful representation of the scene. Currently, most available point clouds have a single colour for each point, assuming Lambertian scenes, but ideally the colour of each point would be view-dependent [1], similarly to what a light field based representation does.
- **3D Mesh Representation Model** - Alternative way of representing the field of light by using a set of polygons defined by single points (vertices), the lines connecting them (edges) and the surface enclosed

by the edges (faces or polygons); as with the point clouds, meshes intend to represent the surface of one or more 3D object(s) in a visual scene [10]. Meshes are often derived from point clouds by adding connectivity (the line connecting the vertices) to the set of 3D points, see example in Figure 4. The colour of the resulting mesh faces is usually obtained by interpolating the colours of the vertices [1].

- **Hologram** – Photographic record of the interference pattern between the diffracted light waves by an object illuminated with coherent light, i.e. “light in which the phases of all waves at each point on a line normal to the beam direction are identical” [11] (as from a Laser, for example), and a phase-relative reference wave [12]. The display of the acquired hologram may be obtained by illuminating the stored hologram with the same reference wave, producing a realistic 3D image with full parallax and changing perspectives depending on the observer’s position, without any eye-wear.

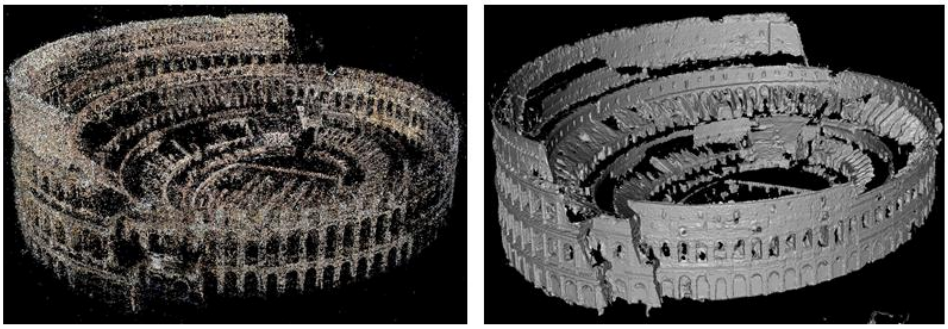


Figure 4 - Example of a point cloud and corresponding 3D mesh: left) Point cloud [13]; right) 3D mesh [13].

- **Virtual Reality** – Form of experience in which the user is immersed in a computer-generated environment or recreation of a real-life situation. This is typically achieved using a HMD device [14]; Figure 5 right) shows an example of VR content.
- **Augmented Reality** – Form of experience in which the real word around the user is enhanced with the addition of computer-generated content in a specific location or activity [15]. Compared with Virtual Reality, the user keeps his/her spatial references what may have benefits in terms of the Quality of Experience. Figure 5 left) shows an example of AR content.

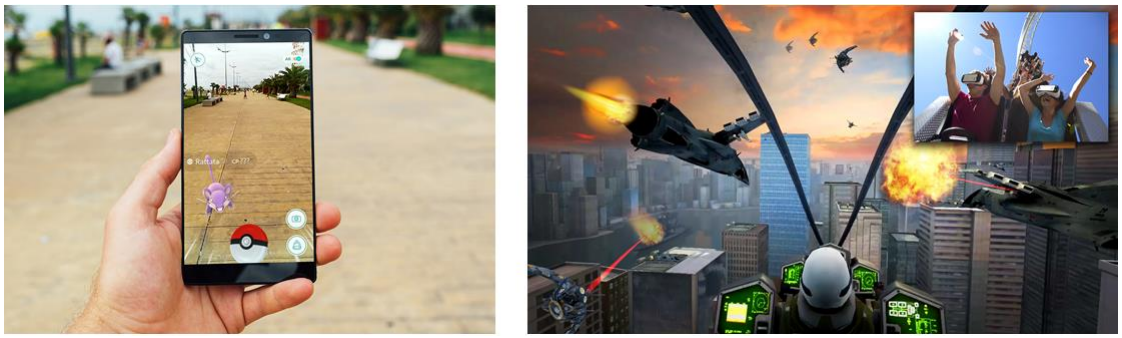


Figure 5 - Examples of AR and VR content/applications: left) The famous app Pokémon Go introduces computer-generated elements (Pokémon) in the real world [16]; right) Six Flags and Samsung partnered to bring Virtual Reality Rollercoasters [17].

- **Degrees of Freedom** – When moving in the real 3D world, a rigid body may do it exploiting 6 degrees of freedom (6 DoF). In fact, the body is free to change position translationally in three perpendicular axes: back/forward, left/right, up/down. It is also free to rotate about the said 3 axes: yaw (horizontally side-side), pitch (vertically up/down) and roll (gaze at an angle) [18], as it is depicted in Figure 6. It is

important to mention that only translational movements result in a parallax effect. All visual representations models targeting user immersion also target offering as many as possible motion degrees of freedom; however, the more degrees of freedom are offered, with the corresponding change on the visual information the user sees, the more data has to be acquired, processed, stored and transmitted. While traditional 2D systems offer limited angle rotations, 360 degrees video intend to offer the 3DoF rotations and light fields and point clouds target offering the full 6 DoF.

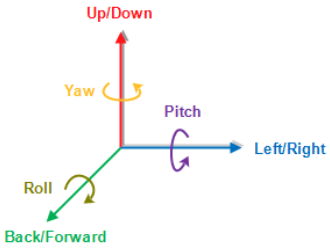


Figure 6 - Graphical representation of the 6 degrees of freedom.

2.2. End-to-End Point Cloud Processing Architecture

This section intends to provide an overview of the end-to-end architecture of a point cloud based system, notably briefly describing the main elements in the processing chain and the relations between them. The end-to-end processing architecture is presented in Figure 7.

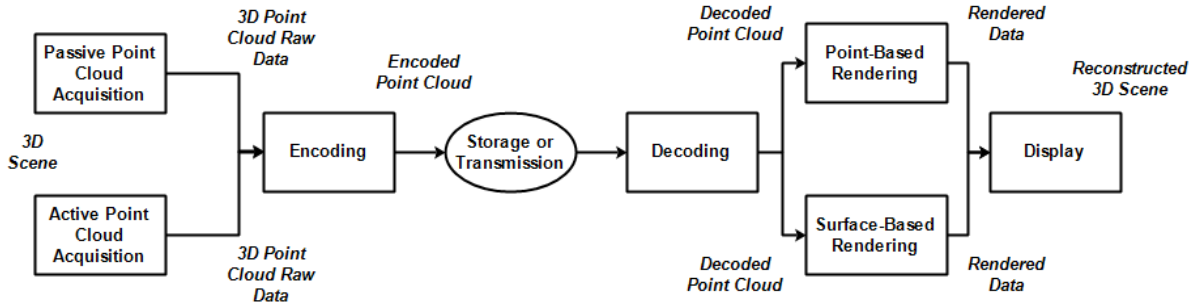


Figure 7 – Point cloud processing flow from acquisition to display.

A. Acquisition

To represent a 3D visual scene as a point cloud, the first step is the acquisition using a sensor or set of sensors that somehow map information defining the visual scene into some appropriate sampled data. This can be achieved with many different sensors and using different acquisition models, notably active depth/range (coupled with simple texture sensors) to directly acquire the point cloud or a set of texture sensors rearranged in different ways (e.g. linear or 2D array) that requires further processing to obtain a point cloud. This difference will be the basis for the classification of the point cloud acquisition systems adopted in this text, notably Active Point Cloud Acquisition systems and Passive Point Cloud Acquisition systems.

A.1. Geometry Acquisition

Active Point Cloud Acquisition

In the so-called Active Point Cloud Acquisition systems, the acquisition of the geometry component of a point cloud is achieved using devices that emit a signal and record some reflected signal that is returned from the visual scene. This can be accomplished following two main different strategies [19]:

- **Structured Light Projection -** This approach is based on the projection of a structured light pattern in the visible or infrared spectrum and the regular acquisition of the pattern illuminated scene from one or more points of view. The correspondences between the acquired pattern and the known projected pattern are then found. After, the 3D coordinates of the object/scene are obtained, e.g. by using triangulation methods and the correspondences [20]. The popular Microsoft Kinect uses this strategy to obtain depth maps, see Figure 8 left). The depth maps are 2D images that for each pixel position *(x,y)* express the distance of the surfaces of scene objects to the sensor. After, a function maps the raw depth value for position *(x,y)* into *(x,y,z)* 3D positions in the physical space, also using some camera calibration parameters. These 3D positions correspond to the geometry of the point cloud.
- **Range-Sensing** – This approach is based on the illumination of a scene with a modulated laser beam or a LED source and the following analysis of the backscattered signal. In this case, a scanning device emits a light signal to illuminate the object or objects around. The object surface reflects the light signal and the receiving unit of the scanning device receives a certain amount of the reflected light. The distance to the points in the illuminated scene is then inferred based on some comparison between the emitted and the received light signals information. This parameter is typically time, or phase difference [21]. For instance, the so-called Light Detection And Ranging (LiDAR) acquisition systems measure the time difference between the corresponding emitted and received laser pulses to obtain the distance to a certain point in the scene (generating depth maps), see Figure 8 right). On the other hand, the so-called Time-of-Flight (ToF) cameras may obtain the distance through the comparison of the time or phase difference between the emitted and reflected continuous light signal, resulting into depth maps [1]. Typically, these cameras are used for shorter ranged applications, i.e. distances lower than 5 meters, while LiDAR acquisition systems are appropriate for longer distances.

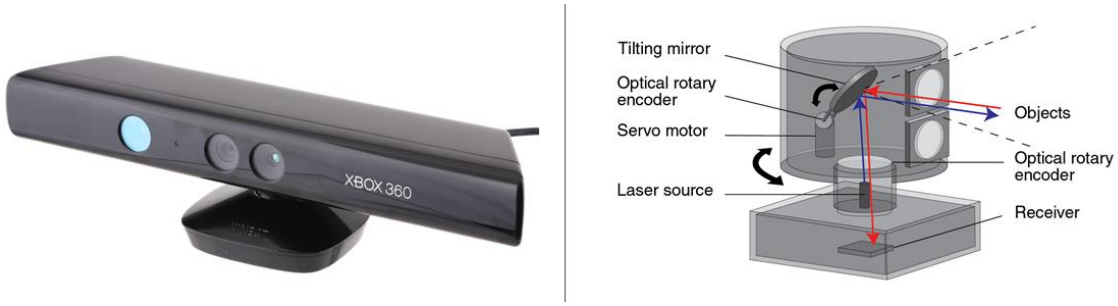


Figure 8 – left) Microsoft Kinect, an example of a structured light-based depth acquisition system [22]; right) LiDAR scanner, which is commonly used in self-driving cars [23].

Passive Point Cloud Acquisition

In the so-called *passive point cloud acquisition systems*, the point cloud is indirectly acquired by using a set of cameras (or one moving camera around an object) that produce a multi-dimensional array of images. Since the acquisition result is not directly a point cloud, it is necessary to create (reconstruct) the geometry from the multiple images acquired. While the reconstruction may be achieved in multiple ways, this text focus on the two most commonly applied solutions [19]:

- **Multi-View Stereo Techniques –** This strategy exploits two or more images of a scene, acquired with multiple cameras or a single moving camera around the object to obtain different perspectives. The reconstruction of the 3D scene structure is made by solving a correspondence problem. The system tries to match pixels in different views and convert their 2D locations into a point cloud using triangulation methods. In some cases, the 2D information may also be converted into depth maps and then converted into a point cloud [24]. One example of a point cloud generated using this type of strategy is depicted in Figure 9.
- **Structure from Motion (SfM)** - This strategy focuses on estimating simultaneously the camera pose and relationships between views by matching features in multiple views, which allows to project to the 3D space the visual information acquired. This is an iterative process as the features are tracked from image to image, enabling initial estimates of the object coordinates and the cameras positions to be refined using least-squares minimisation [25]. This technique may be applied to get 3D models from Internet pictures as it is described in [26]. First, the feature points in each image are detected and then these feature points are matched between pairs of images. After the matches are found, they are organized into the so-called tracks, i.e. "a connected set of matching key-points across multiple images" [26]. After, an image pair is selected using some criteria, the camera parameters (e.g., rotation, translation, and focal length) are estimated and the tracks are triangulated, estimating the 3D point locations. As this is an iterative technique, other cameras are then added to the process to finally obtain a richer 3D model [26].

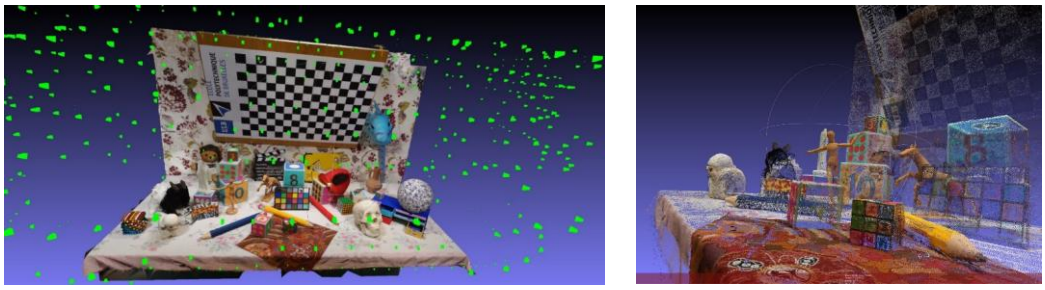


Figure 9 - Example of the very complex Unicorn point cloud used in MPEG which has been obtained with around 500 views in rather irregular camera positions to reach more obstructed areas [27].

A.2. Colour Acquisition

Active Point Cloud Acquisition

The colour acquisition is performed with an RGB camera associated to the geometry acquisition system. In the case of Microsoft Kinect, the RGB image typically has the same resolution of the depth map to make a 1 to 1 correspondence; otherwise, a more complex calibration is necessary. In the LiDAR system, for the colour attributes to be associated with the geometry, an RGB camera can be integrated with the geometry acquisition system, making sure that each scan line of the laser scanner is covered by texture, or the colour and the geometry information can be acquired separately, as in [28]. In addition, there are also ToF systems that integrate an additional colour camera synchronized with the ToF camera, in order to obtain RGB-D data that is then transformed into the colour and geometry of the point cloud [21].

Passive Point Cloud Acquisition

In the passive point cloud acquisition systems, the texture is most of times easily obtained as it comes directly from the matching of images, notably from the matching pixels or matching features. Thus, when the geometry information is associated to a given point, the colour associated to the pixels or matching feature must be also associated to that point.

The result of active and passive point cloud acquisition systems is a raw point cloud. As it is typical for a raw point cloud to have a considerable amount of noise, it is common to apply some type of filtering to the raw point clouds to remove unreliable/outlier points [19].

B. Encoding and Decoding

After obtaining the point cloud raw data, it is necessary to prepare this data for transmission or storage, by performing compression, exploiting the raw data redundancy but also irrelevancy, if lossy coding is targeted, and also the statistical characteristics of the produced coding symbols. After the storage or transmission using an efficient coding solution, it is time to recover a decoded point cloud from the encoded data. It is important to note that dense point clouds may include several millions of points, making it impossible to always rely on the raw data. The coding/decoding modules are highly dependent on the content and target application for the point cloud. In this context, it is essential to distinguish two main different types of coding:

- **Lossless coding** – The content is coded preserving all the original information, which means that the original and decoded content are mathematically the same;
- **Lossy coding** – The content is coded without preserving all the initial information, i.e. the original and decoded content are mathematically different, even though they may seem subjectively the same. The amount and type of losses are determined by the type of coding and its control parameters and the application scenario requirements.

Naturally, lossless coding implies lower compression factors and thus a larger amount of data to store and transmit, as only the samples redundancy is exploited while lossy coding exploits both the redundancy and irrelevancy. While most applications do not ask for lossless coding, this requirement may be relevant for some applications, e.g. for geographical information systems which is important to accurately measure distances between points.

As described in 2.1, each point in the point cloud is composed at least by geometry information and it may be complemented with some attributes, notably colour. Thus, there are several types of coding solutions, namely for geometry and attributes with these corresponding to colour, normal, reflectance, etc.

According to [1], the most popular point cloud coding solution at the end of 2017 is the so-called *Point Cloud Library (PCL*) [29], which addresses both geometry and attributes coding. While there are other codecs available in the literature, a standard is not yet available. However, this situation should change soon as MPEG launched, in January 2017, a Call for Proposals on Point Cloud Coding [6], which should quickly substantially change the point cloud coding landscape. For a deeper analysis of the currently available coding solutions, please refer to Section 2.5. In Section 2.6, the Call for Proposals and its results are detailed.

C. Rendering

After decoding, for the content to be displayed in some specific type of display, it is necessary to render the point cloud. Rendering is the computational process whereby the points of the point cloud are converted to a representation appropriate for the display at hand, most usually a 2D image-plane representation corresponding to the user requested point of view. Naturally, whatever the display, the rendering process should be designed to provide the user the best experience for the available display device.

The conditions in which the acquisition process was carried out or the acquisition system itself has also an impact on the rendering process, since it may lead to point clouds with artefacts, notably non-uniform sampling, noise, outliers, misalignment and missing data [30], which may lead to object holes and other visibility/occlusion problems on the visual result presented to the final user. The illustration of these different types of artefacts is depicted in Figure 10. To avoid these problems, it may be necessary to reconstruct the surface of the object or set of objects and to project them into a 2D image plane, in the so-called view creation process. It is also possible that some super-resolution tool is applied before the rendering to obtain more dense point cloud in all or only some selected regions. The order by which these two steps are done leads to two types of techniques:

- **Object-space artefact solving techniques** – The hole-free and the visibility problems are solved in the 3D domain (commonly called object-space), and the point cloud is afterwards rendered.
- **Image-space artefact solving techniques** – First, the points are projected into the 2D plane and afterwards the hole-free and other visibility problems are solved in the 2D domain (commonly called image-space).

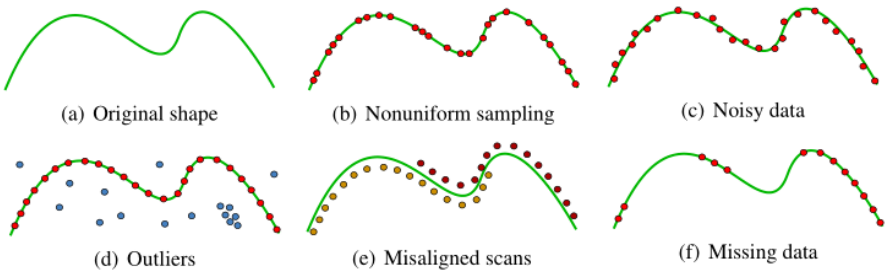


Figure 10 – Different types of point cloud artefacts [30].

Traditionally, the rendering of point clouds has been achieved by first converting to a more suitable representation model to be rendered, typically a 3D mesh model. However, efforts have been made to directly render surfaces from the cloud points or to render the points directly. This gives room to two different classifications, the so-called *point-based rendering* and *surface-based rendering*.

C.1. Point-Based Rendering

In point-based rendering, typically a surface is implicitly defined from the points of the point cloud to create the 2D view afterwards. To implicitly define the surface of the point cloud there are two main techniques:

- **Surface Splatting** – This method is based on radial or elliptical expansion tangential to the surface of each point and requires the computation of the surface normal for each point. This process generates

discs, the so-called splats, that overlap one another to cover all the surface of the scanned object [31]. Depending on the point cloud density, the size of the splats may have to be different to avoid holes but the final rendered point cloud may become less precise or realistic. There are multiple techniques that follow this approach, which may include an object-space based solution (splatting and the associated processing in the object-space until it is finally projected to a 2D plane) or an image-space based approach (splatting in the object-space and projecting to a 2D domain for the processing to take place in the image-space). Usually these techniques require some type of smoothing to obtain a more natural and realistic result. Figure 11 right) shows an example of a surface splatting process.

- **Point Set Surface** – The surface is locally reconstructed by making a polynomial fit of the set of points in a small vicinity, surrounding a given point. It is worth mentioning that the polynomial function is not required to pass through all the cloud points. Usually, this type of method is much more computationally demanding than the surface splatting but it typically produces more photo realistic renderings [32].

Point-based rendering does not necessarily need to involve the implicit definition of the surface. It can be achieved directly from the points, which has been more and more adopted since the acquisition systems are enabling denser point clouds. However, the results can be poor, notably if the point cloud is not sufficiently dense.

C.2. Surface-Based Rendering

In the surface-based rendering, the points of the point clouds are used to explicitly reconstruct a surface, typically a mesh:

- **Polygonal Mesh Generation –** The set of points is connected forming closed polygonal surfaces, typically triangles. As the number of points increases, the number of pixels covered by each polygon decreases. Thus, the generation of meshes may be a time consuming, difficult and costly process. One of the most popular mesh generation techniques is the so-called *Poisson reconstruction* [33]. A rendering example obtained with this technique is depicted in Figure 11 left).

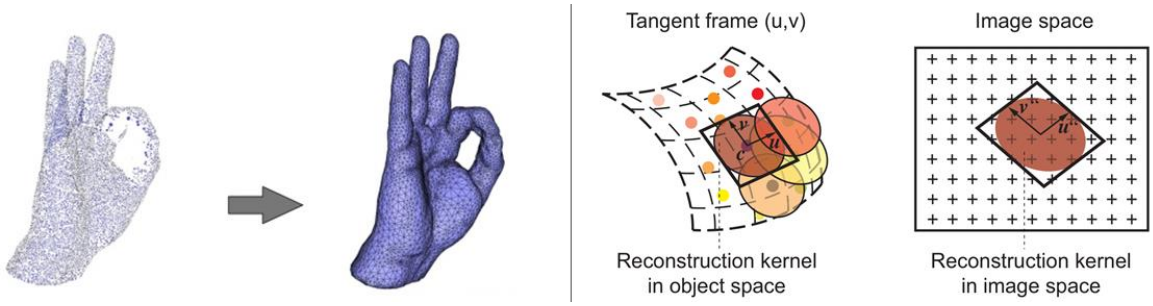


Figure 11 – left) Example of a Poisson reconstruction technique applied to a point cloud with missing data [33]; right) Splatting applied in the object-space and projected into the image-space [34].

The mesh rendering is a topic that has been widely explored in the Computer Graphics scientific community and thus, it is more developed than the point-based rendering, with multiple different solutions.

C.3. View-Creation Process

Another very important part of the rendering process is the view creation part, whereby the projection into the 2D domain is made to provide the user requested view. Even though there are more techniques, the following two are the most commonly used:

- **Raytracing** - Having the surface ready to be projected, a virtual camera inside the scene is placed in the location that will be used to create the rendered view. After this, a plane (image-plane) is formed and divided in a grid of pixels, upon which the image will be formed. This image-plane represents the field of view for the virtual camera and the purpose of the raytracing process is to figure out the colour of each pixel. Thus, for each pixel, a rendering engine casts a ray that starts at the camera, crosses the centre of the pixel and then extends off into the scene. The ray will reflect in the series of objects until it hits the light source or the maximum number of rebounds is reached. If the ray does not intersect any objects, the corresponding pixel takes the background colour. The final computed colour of the pixel is obtained considering the object colour, the reflections, refractions and shadows. Raytracing is illustrated in Figure 12 left). Typically, this has been a method applied to polygonal mesh surfaces, but in [34] and [35], ray-tracing is also applied to the implicit surface definition methods.
- **Rasterization** - From a high level, this view creation approach may be described as the simple identification of which pixels an object lies on from a certain view perspective and the respective colour computation. An example of the rasterization process can be found in Figure 12 right). While it is much less computationally demanding than raytracing for example, it may reduce the realism of the rendering result. Thus, rasterization is widely used in real-time content.

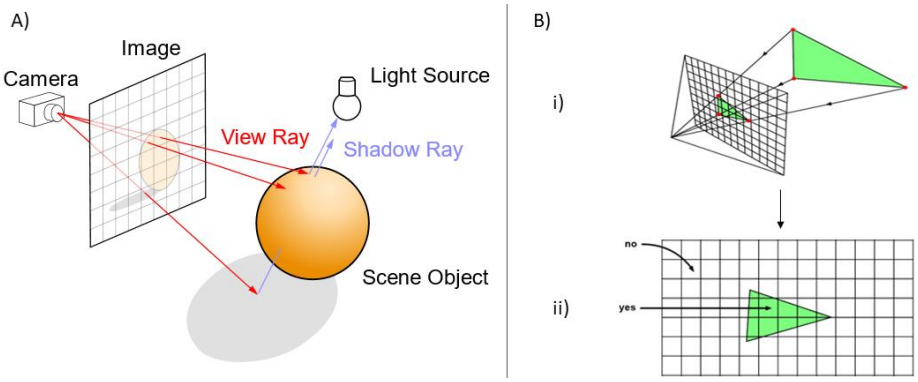


Figure 12 - left) Raytracing of an object scene with shadows and reflections [36]. right) Rasterization of a triangle divided into 2 steps: i) projection to a 2D-plane; ii) testing if the pixels are filled with the projected triangle [37].

D. Display

The last module in the end-to-end architecture is the display. The display is the physical device where the best user experience should be provided by exploiting the acquired/created information and eventually some associated metadata. [1] There are several types of display devices that may be used in multiple applications scenarios where point clouds representations are appropriate (in practice any display may be used after suitable rendering), notably:

- **Conventional 2D displays** – Displays using regular images and video, notably represented with three colour components, RGB or luminance and chrominance data, that are able to show a single 2D perspective of the scene [1].
- **Stereo Displays** – In this case, the user is provided with 2 different views (one for the left and another for the right eye) to offer stereo parallax; this usually involves using a pair of glasses, which may be associated to some user discomfort, especially for longer viewing times.
- **Autostereoscopic Displays** – Displays providing multiple views to the user to offer motion parallax, typically only horizontal; unlike stereo displays, usually autostereoscopic displays do not involve the use of any eye-wear.
- **Super Multiview Displays** – The user is provided with a very high number of views providing smooth motion parallax, typically only horizontal, but eventually also both vertical and horizontal parallaxes; similar to autostereoscopic displays, these displays do not the need any eye-wear.
- **Head Mounted Display (HMD)** – Display device to be used on the head or as part of a helmet with an optical display in front of one (monocular HMD) or both eyes (binocular), see Figure 13 left); this is the most common display device for Virtual Reality (VR) and 360 images/video.
- **Augmented Reality Display** – This is a see-though display device allowing to see real world views augmented with computer generated content in order to enhance the user's experience.
- **VR caves** – This is a rather complex type of display where multiple views are used to create an immersive lifelike environment. The most famous example is the so-called CAVE (cave automatic virtual environment), which makes use of multiple projectors directed to between three to six walls of a room-sized cube, where the displayed content is controlled by the user's position as the user is provided with a view corresponding to his/her real-time position; for a CAVE display example, see Figure 13 right).

Figure 13 - left) Oculus Rift, an example of a Head Mounted Display (HMD) [38]; right) CAVE display system [39].

2.3. Point Cloud Use Cases

This section intends to present relevant examples of point cloud use cases which should show the potential practical impact of developing point cloud coding solutions; this review is largely based on a similar review made within MPEG when this standardization group decided to launch a Call for Proposals on Point Cloud Compression [40].

- **Real-time 3D presence** - Traditionally, the most common method of presenting visual data to the users has been with 2D video representations. The development of more sophisticated devices able to capture and present 3D visual representations of the world, and the advances on 3D reconstruction,

have enabled the evolution of real-time 3D immersive telepresence systems. In this field, 3D point clouds play an important role, as they offer a representation that can be rendered and integrated for virtual reality purposes with a high subjective quality, thus enabling the introduction of real world elements in virtual realities and vice-versa.

In this use case, the visual data streams are acquired with multiple, calibrated colour plus depth cameras, which are afterwards processed to produce 3D point clouds. After the proper point cloud representation is obtained, compression and transmission take place, possibly after further conversion to a mesh representation. At the receiver side, the transmitted data is decoded and rendered in real-time in a 3D space or virtual world to be displayed to the user. At a high level, one might find that this end-to-end design is similar to the one used in traditional video-conferencing although much more immersive.

This type of systems is already being developed; for example, the so-called Microsoft Holoportation telepresence system [41] uses a Head Mounted Display (HMD) to allow the users to see, hear and interact remotely in real-time with other participants who are 3D reconstructed, see Figure 14.

Figure 14 – main frame) Conversation between two remote persons using Holoportation technology; left top) Footage of the "remote participant" room [42]; left bottom) Actual footage shot through the HMD – HoloLens - of the "local participant".

- **Virtual-Reality with Interactive Parallax -** Another application that is very much related to the previous one is Virtual Reality (VR) with Interactive Parallax. Using a HMD, the user is provided with an immersive experience through the visualization of 360 degrees virtual reality worlds. If the user changes its position and the perspective does not shift (or it shifts incorrectly), the immersion feeling is broken, and the experience brings visual discomfort. This issue can be improved with the update of the rendering viewport for each new position of the end-user head, also known as interactive parallax; parallax refers here to the effect whereby the position or direction of an object appears to differ when viewed from different positions.

The main difference between these first two use cases is that the latter does not have very demanding real-time requirements as the first one, thus enabling a longer and more complex content production workflow.

- **3D Free Viewpoint Sport Replays Broadcasting -** Point cloud representation is also useful in 3D Free Viewpoint Sport Replays Broadcasting. The representation of visual information, notably using a point cloud model, in sports events like baseball, basketball and football, enables the free viewpoint

playback and interaction on TV and mobile devices where the user may see the players from any perspectives he/she wishes. Intel® is already developing freeD™, a technology to provide the end-user with 360 degrees replays, creating a multi-perspective view of key moments, see Figure 15 [43].

Figure 15 - 360 degrees replay of an American football player using Intel® freeD™ technology [43].

- **Cultural Heritage -** Cultural objects and collections can be archived, through a point cloud scan, and later visualized. This allows the content to be experienced by the user in a very immersive way, beyond the physical installation. The most wide-spread point cloud based example in the literature is the Shipping Galleries at the Science Museum of London, see Figure 16. The entire point cloud model includes more than 2 billion points, taken in 276 scans [44].

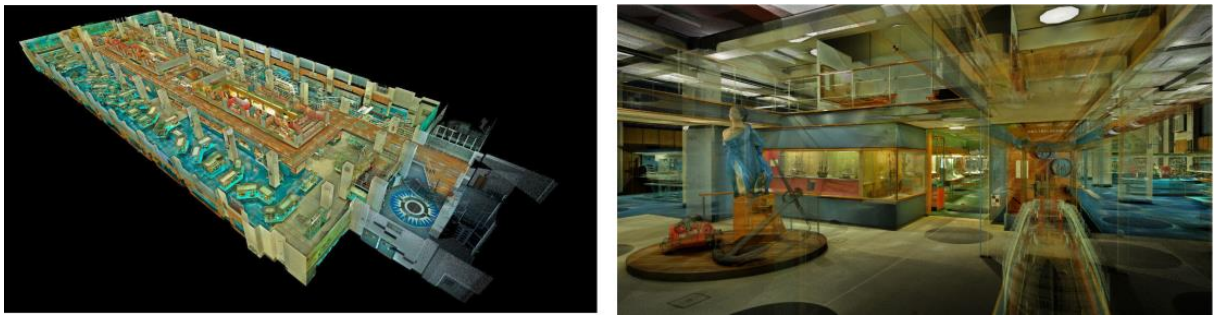


Figure 16 - left) A view over the entire Shipping Galleries digital model [44]; right) A view taken from inside the Shipping Galleries digital model [44].

- **Geographic Information Systems** - Geographic information can also be represented as a point cloud. In this case, the acquisition can be achieved using an airborne Synthetic-Aperture Radar (SAR) or a Light Detection And Ranging (LiDAR) system, among other techniques. This data is often stored in servers that are able to provide services, such as remote rendering and geographic information querying. However, the information is still too massive to be handled efficiently by the Point Cloud Data Management Systems. A standardized coding solution, offering significant compression capabilities, would help to attenuate this issue and to reduce the traffic exchange between the server and the clients. An example of this use case is the AHN2 dataset with the height map of Netherlands, containing around 640 billion points, with 6 to 10 points per square meter [5].
- **Large Scale 3D Maps** – The last point cloud use case regards large scale 3D maps: these maps can be created using localization devices in combination with depth and colour measurements of the involving environment. Combining these maps with other features, such as road marking and lane information, will enable the autonomous navigation of vehicles. The 3D maps can also be used for other purposes such as space planning, public surveying of infrastructures and roads, among many other services [5]. One good example is the Mitsubishi Electric Mobile Mapping System (MMS) [45], which

creates high-accuracy 3D maps of urban areas, by combining GPS antennas, laser scanners, colour cameras and other devices on a car, see Figure 17.

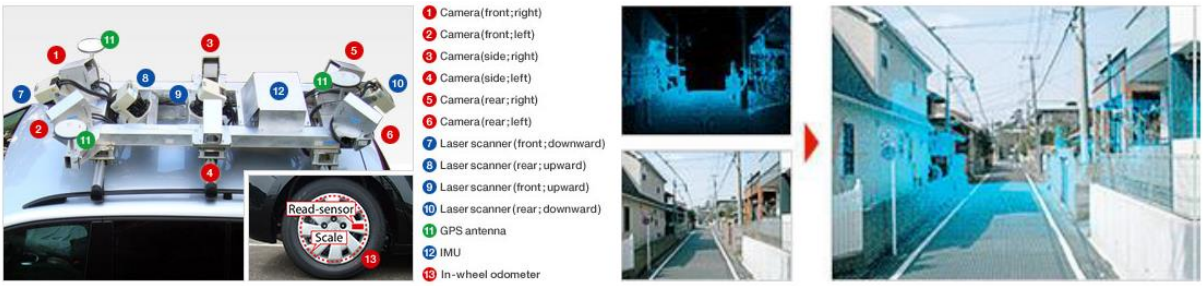


Figure 17 – left) Mitsubishi Electric MMS Acquisition System [46]; right) map created by superimposing high-density point cloud data onto camera images using MMS (right) [46].

2.4. Point Cloud Quality Assessment

As the quality of experience (QoE) is of the utmost importance in point cloud systems, it is crucial to determine which coding solutions provide the best experience to the final user. This is typically achieved by an evaluation procedure, which assesses the quality of the decoded point cloud via an objective or subjective method. Thus, this chapter briefly reviews two different approaches for the evaluation of the quality associated to the point cloud coding solutions, the so-called *objective quality evaluation metrics* and the *subjective quality evaluation protocols*.

A. Objective Quality Evaluation Metrics

The most common type of objective quality assessment is based on the comparison between the original (raw) point cloud and the decoded point cloud, where the original point cloud acts as the reference to determine the fidelity/quality of the decoded point cloud. This type of metric is called *full reference quality metric* as there is original data taken as reference.

With the purpose of objectively evaluating the quality of different point cloud coding solutions under the same test conditions, MPEG has adopted a set of objective quality assessment metrics presented in [6]. The MPEG adopted metrics are divided into so-called Point-to-Point (P2Point) metrics that define a point to point distance between the points of the original and the degraded point clouds and Point-to-Plane (P2Plane) metrics that define a distance between a point in the decoded point cloud and the plane orthogonal to the normal of the closest point in the original point cloud. The idea is that this plane approaches the surface of the object associated to the points and thus this type of distance may be perceptually more meaningful. Figure 18 right) shows the difference between the P2Point and P2Plane distances/errors.

For better understanding, it is important to present some basic terminology that will be used:

- **Point (v)** – Basic unit of a point cloud; each point v is composed of geometry *(x,y,z)*, an optional colour (*c*), with Red, Green, Blue *(r,g,b)* or luminance and chrominances *(y,u,v)* components, and other optional attributes a_i, such as normals, reflectances, etc.
- **Original Point Cloud (V_{or})** – Initial (before being encoded) set of *K* points (v_i) without any relevant order.

- **Degraded/decoded Point Cloud (V_{deg})** – Set of *N* points (v_i) which result from encoding and decoding the original point cloud; often *N* is different from *K*, typically smaller

The quality assessment process takes as input the original point cloud (V_{or}) and the degraded point cloud (V_{deg}) to output the objective quality score ($Q_{pointcloud}$), as illustrated in Figure 18 Left).

A.1. Geometry Quality Evaluation Metrics

This section introduces the objective quality metric for geometry.

Point-to-Point Metrics

For the P2Point geometry assessment, some basic metrics are relevant, notably:

- d_{min} – Euclidian distance between a point v_a of a point cloud V_a and its nearest neighbour in the point cloud V_b, formally defined as follows:

$$d_{min}(v_a, V_b) = \min_{\forall v_b \in V_b} \|v_a - v_b\| \quad (3)$$

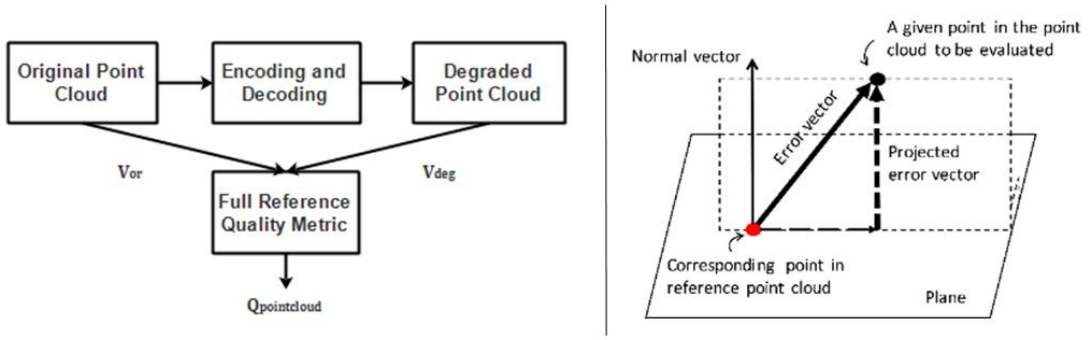


Figure 18 – left) Illustration of the objective quality metric processing chain [47]; right) P2Point distance (error vector distance) and P2Plane distance (projected error vector distance) [48].

- d_{rms} – Root mean square of the distances between the points of the two point clouds defined by d_{min}, which is formally presented as follows:

$$d_{rms}(V_a, V_b) = \sqrt{\frac{1}{K} \sum_{v_a \in V_a} d_{min}(v_a, V_b)^2} \quad (4)$$

Where *K* is the number of cloud points in V_a. This distance metric involves computing first the minimum distance between all the points of V_a and the closest point of V_b.

- $d_{hausdorff}$ – Maximum of the distances between the points of one point cloud V_a to a full point cloud V_b:

$$d_{hausdorff}(V_a, V_b) = \max_{\forall v_a \in V_a} d_{min}(v_a, V_b) \quad (5)$$

Since the Hausdorff distance corresponds to a maximum distance, this metric is determined by the worst distance case, e.g. some outlier points may drastically increase this metric.

$d_{rms}(V_a, V_b)$ and $d_{hausdorff}(V_a, V_b)$ may be interpreted as average (root mean square to be precise) and maximum errors in terms of geometry**,** respectively, notably when estimating/representing V_a with V_b. However, d_{rms} and $d_{hausdorff}$ are typically not symmetrical, i.e. $d_{rms}(V_a, V_b) \neq d_{rms}(V_b, V_a)$ and $d_{hausdorff}(V_a, V_b) \neq d_{hausdorff}(V_b, V_a)$. Since the purpose is to compute the average and the maximum

errors between the two point clouds, making a 'one-sided' computation may lead to the underestimation of these errors. This is the reason for the definition of symmetric rms and Hausdorff distances as quality metrics, instead of only a one-direction rms or Hausdorff distances.

Having covered the basic definitions, the objective quality metrics for point clouds typically adopted in the literature to measure the P2Point error are:

- $\boldsymbol{d_{symmetric\ rms}}$ – Symmetric rms distance between point clouds, formally defined as:

$$d_{symmetric\ rms}(V_{or}, V_{deg}) = max\left(d_{rms}(V_{or}, V_{deg}), d_{rms}(V_{deg}, V_{or})\right) \quad (6)$$

- $\mathbf{d_{symmetric\ hausdorff}}$ – Symmetric Hausdorff distance between point clouds, formally defined as:

$$d_{symmetric\ hausdorff}(V_{or}, V_{deg}) = max\left(d_{hausdorff}(V_{or}, V_{deg}), d_{hausdorff}(V_{deg}, V_{or})\right) \quad (7)$$

It is important to note that these two metrics are referred to as D1 metrics in the official MPEG documents [6], where the former metric is used to measure the P2Point error for all test conditions excepting the near-lossless conditions, where the latter metric is used.

Point-to-Plane Metrics

The main motivation for the definition of the so-called Point-to-Plane metrics is to increase the correlation between the objective quality results and the subjective scores. While the Point-to-Point measure the distance between a point and a point cloud through the error vector $E(v_a, v_b)$, depicted in Figure 18 right), that connects v_a to its closest neighbour v_b, the projected error vector along the normal direction (N_b) of the point (v_b) is given by $(E(v_a, v_b).N_{vb})$. In summary, the projected error vector distance is defined:

$$d_{P2Plane}(V_{deg}, V_{or}) = \sqrt{\frac{1}{N}\sum_{v_{deg}\in V_{deg}} \left\|E(v_{or}, v_{deg})\cdot N_{or}\right\|^2} \quad (8)$$

Similarly to the P2Point metrics, the symmetric distance is also computed for $d_{P2Plane}(V_{deg}, V_{or})$, which is referred by MPEG as D2 metric [6].

PSNR Computation

The metrics are usually presented in the form of geometry peak signal to noise ratio, formally defined as:

$$psnr_{geom} = 10\ log_{10}\frac{\sqrt{3.peak^2}}{MSE} \quad (9)$$

where MSE is the P2Point (D1) or P2Plane (D2) error and $peak$ is the peak constant value defined for each reference point cloud and specified in [6]. If the point clouds are voxelized to *D* bits of precision, $peak = 2^D - 1$. The '3' comes from the fact that the MSE regards three spatial components.

A.2. Colour Quality Evaluation Metrics

It is also important to review the most important basic definitions and metrics related to colour data:

- $\boldsymbol{v_{b\ closest\ neighbour}}$ – Closest point of the point cloud V_b to a specified point of V_a, notably:

$$\boldsymbol{v_{a\ closest\ neighbour}(v_a, V_b) = v_b \in V_b : \min_{\forall v_b \in V_b} \|v_a - v_b\|} \quad \textbf{(10)}$$

- $\boldsymbol{d_x}$ – Root mean square of the difference between the value of a colour component x for a point v_a, denoted as $x(v_a)$, and the corresponding value for the same colour component of the closest point in a point cloud V_b:

$$d_x(V_a, V_b) = \sqrt{\frac{1}{K}\sum_{v_a \in V_a} \left\| x(v_a) - x\left(v_{a\ closest\ neighbour}(v_a, V_b)\right)\right\|^2} \quad (11)$$

Where *K* is the number of points in V_a.

- $\boldsymbol{d_{symmetric\ x}}$ – As for the geometry evaluation, d_x is not symmetrical and thus it makes sense to define a corresponding symmetric metric:

$$d_{symmetric\ x}(V_a, V_b) = max\left(d_x(V_a, V_b), d_x(V_b, V_a)\right) \quad (12)$$

Having covered the basic definitions, the only colour quality metric that is applied to the three colour components in the colour space is now presented:

- $\boldsymbol{psnr_x}$ – The peak signal to noise ratio of the colour component x computes a ratio between the maximum signal value (255 for 8-bit per colour components) for the colour component and the error taken by approximating the original point cloud V_{or} with the degraded point cloud V_{deg}; if x is one of the colour components for a specific colour space, the metric is defined as:

$$psnr_x = 10\log_{10}\frac{255}{d_{symmetric\ x}(V_{or}, V_{deg})} \quad (13)$$

A.3. Compression Evaluation Metrics

The aforementioned objective quality metrics assess the geometry and colour errors between the original and degraded point clouds. However, the rate and complexity costs of the quality have also to be evaluated as this is what defines how good a codec is. Naturally, if the point cloud does not have attributes associated to it, the related quality metrics are not used. The rate performance may be evaluated using the following metrics:

- **Compressed size** – Number of bytes used for the compressed point cloud;
- **Geometry compressed size** – Number of bytes used for the compressed geometry data;
- **Colour compressed size** – Number of bytes used for the compressed colour attributes;
- **Attributes compressed size** – Number of bytes used for the compressed attributes;
- **Output point count** – Total number of points in the degraded/decoded point cloud.

The compression performance is defined in terms of a rate-distortion (RD) curve where the rate associated to each quality level is represented.

Besides the rate cost, it is also important to assess the complexity of a codec, usually by means of the following metrics:

- **Encoder running time** – Encoding time in milliseconds to run a specific coding test on a well-defined hardware and software platform;
- **Decoder running time** – Decoding time in milliseconds to run a specific decoding test on a well-defined hardware and software platform.

Altogether, the metrics above allow to rather well characterize the overall performance of a point cloud codec.

B. Subjective Quality Evaluation Protocols

The subjective quality assessment protocols are used to anticipate the evaluation to be made by those who may view or interact with the decoded content, the users. Usually, subjective assessment is more cumbersome, time-consuming and expensive, as it involves a statistically significant number of people, performing their assessments under well-defined conditions for the results to be statistically relevant [49]. Even though the purpose of objective quality assessment is largely the same, often the objective and subjective quality assessment results do not correlate as well as desired [50] as the objective quality metrics are not able to fully express the quality assessment process performed by a Human through the human visual system. This is even more important for the point cloud quality metrics as the available metrics are still very 'young' and rather simple. This is the reason why, despite its downsides, subjective quality assessment is very important and much used.

The subjective quality assessment experiments with point clouds reported in the literature are rather limited. In [49] and [50], a subjective quality assessment protocol method originally tailored for 2D video is used, the so-called *Double Stimulus Impairment Scale (DSIS)*. DSIS is a cyclic method, whereby the evaluator first watches the rendered original point cloud taken as reference, and afterwards the rendered impaired point cloud. The references and degraded points clouds are rendered for a succession of user perspectives according to a pre-defined view navigation path and a video is created with these successive renderings. At this stage, the point cloud quality is assessed in terms of video quality. Thus, the evaluator scores the quality of the second point cloud video, in terms of how impaired it is, taking as reference the first reference point cloud video. The scores are in a 1-5 scale according to the following quality levels: 1- very annoying (degradation); 2- annoying; 3- slightly annoying; 4- perceptible, but annoying; 5- imperceptible. Naturally, the quality scale may consider more levels if appropriate, notably if it is possible to accurately distinguish them. The test sessions typically last up to half an hour (to avoid the subjects to be too tired), and a series of 'reference' and 'degraded/decoded/denoised' point cloud pairs are presented in random order for different levels of degradation. At the end, the mean quality score for each point cloud and for each test condition, e.g. quantization level, is calculated.

In the process above, the rendering process, which has to be applied to the reference (non-degraded) and degraded point clouds to create the video frames, has a major impact as it influences the quality shown to the test subjects.

2.5. Main Point Cloud Coding Solutions

After having introduced many relevant concepts and tools related to point clouds, this section will briefly review the main coding solutions in the literature.

2.5.1. Point Cloud Library Solution

The Point Cloud Library (PCL) point cloud coding solution [51] is very likely the most popular solution in the literature as of the end of 2017, largely due to the availability of the PCL software implementation.

It is a lossy, octree-based coding solution that considers both the geometry and attributes for static and dynamic point clouds. The set of 3D points is organized as an octree which is a tree data structure, where each branch node has up to eight children – one for each sub-octant (also called *voxel*), starting from the initial node associated to the full point cloud bounding box. This data structure is used to partition the 3D space by recursively subdividing it up to the required precision. An example of an octree structure is depicted in Figure 19.

The octree itself only identifies the leaf octants (up to a certain resolution) where one or more points of the point cloud lie, and the smallest occupied voxels are represented with its central point, independently of the number of points that fall in it. If further spatial resolution is required, it is possible to represent the exact position of one or more points within the same leaf voxel by coding the distance between the point in question and the octant/voxel origin, generating the so-called *point detail coefficients*. Naturally, this increased level of resolution may also be achieved by increasing the octree depth/resolution.

A similar octree-based procedure is applied for colour coding. After the octree decomposition, the colour for each leaf voxel is coded as the average of the colours of the points in the voxel. If further point detail precision is required, the colour difference for each point in the voxel regarding the average voxel colour is coded. After, the three complementary streams, i.e. the octree structure, the point details coefficients and the colour coefficients, are separately entropy coded by a range encoder and multiplexed into a final, single output bit stream.

This is a rather basic coding solution for the colour components which does not take into account the spatial redundancy among different octants, nor the temporal redundancy among temporally adjacent point clouds. The encoding of other attributes is not mentioned but according to [51], it makes sense to analyse and fine tune the coding procedures for each attribute, taking into account their specific characteristics, instead of relying on a single generic approach. This very simple point cloud colour coding solution will be a relevant benchmark for more sophisticated ones to come.

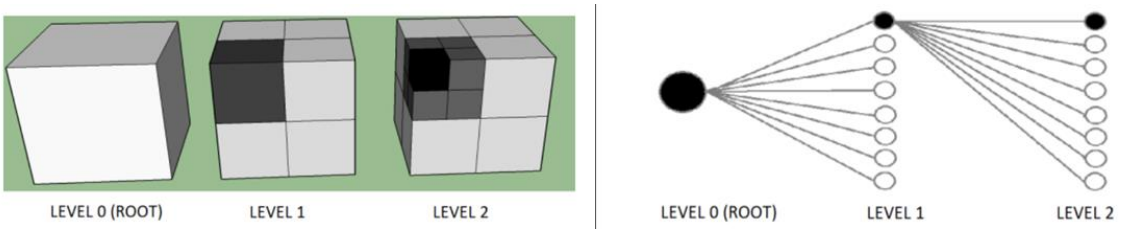


Figure 19- left) Recursive sub-division of 3D space into octants; right) Corresponding octree structure (based on [52]).

2.5.2. JPEG-Based Point Cloud Colour Coding

A. Objective and Main Approach

Since octrees are probably the most adopted data structures to represent point clouds, this section reviews another octree-based coding solution, now targeting dynamic point clouds. This solution has been authored by Mekuria *et al.* [52] and proposes a lossy codec that is claimed to be suited for 3D immersive real-time video, where point clouds are acquired at a rather fast rate. Following this requirement, the geometry is progressively coded with an octree subdivision and an inter-prediction

algorithm is proposed to exploit inter-frame dependencies. The colour attributes are encoded by mapping the vertex colour into an image grid and using after the most popular image coding standard, i.e. JPEG. As the main novelty of this coding solution is the colour encoding process, particular attention will be devoted to it.

B. Architecture and Walkthrough

The proposed architecture for the colour coding process is presented in Figure 20.

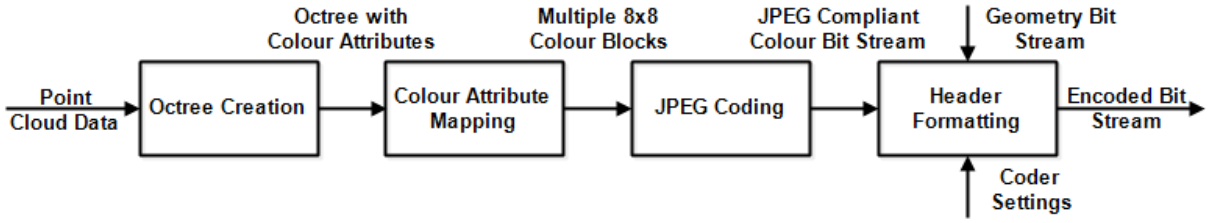


Figure 20 – End-to-end architecture for the colour coding process.

To start the colour encoding process, it is necessary to have available the geometry octree with the associated colour attributes, which is obtained from the point cloud data after the so-called *octree creation* [52]. While the octree creation is a core module for geometry coding, it is also essential for colour coding and thus it will be also included in the following colour coding walkthrough:

- **Octree Creation** - First, a bounding box is computed as the smallest rectangular box which may include the full point cloud. This bounding box will be the root level of the octree. In case the bounding box changes from frame to frame within a certain limit, the algorithm may expand it to use a common axis and range representation between subsequent frames and a consistent bit assignment for the octree. This will make inter-prediction easier. In addition, an outlier filter is included to increase the effectiveness of the bounding box by avoiding 'noisy' points. After, the bounding box for each frame is recursively subdivided into eight children until the target octree depth level is achieved. The position of each voxel is represented by its centre and the colour is set to the average of the points in the voxel. If the octree decomposition proceeds until no voxel has more than one point, there is no averaging for the colour.
- **Colour Attribute Mapping** – The octree voxels colour values are mapped into an image-like regular grid, using a depth first octree traversal process which means that the octree is traversed as deep as possible on each child (until it gets to a leaf node) before going to the next child [53]. The order for this colour mapping into 8×8 blocks is defined in Figure 21 left). The depth first traversal scanning order is chosen as it allows creating 8×8 blocks with the advantage that the mapped neighbouring points in the image grid correspond to points that are adjacent in the point cloud and thus very likely with well correlated colours. As output, one gets the octree-organized colour values mapped into 8x8 blocks of colour samples.
- **JPEG Coding** – The module applies JPEG coding to the mapped 8x8 blocks in order to encode the various components of the colour attributes. This module starts by applying the DCT to each 8x8 block, to explore the spatial redundancy between the data samples in the blocks of each colour component. After, the DCT coefficients are quantized to explore the spatial irrelevancy. Lastly, the quantized

coefficients are converted into symbols and subsequently entropy encoded, taking into consideration their statistical distribution.

- **Header Formatting -** After exploiting the spatial and statistical redundancies and also the irrelevancy in the points colour, the coded colour attributes, the coding settings and the coded geometry individual bitstreams are combined into a single bitstream using a common header format.

Along the end-to-end processing flow, some coding control criteria and tools are necessary to select the best colour bit allocation, mapping mode, etc., but in the presented coding solution they are omitted for simplicity.

0	1	2	3	4	5	6	7	64
15	14	13	12	11	10	9	8	79
16	17	18	19	20	21	22	23	80
31	30	29	28	27	26	25	24	…
32	33	34	35	36	37	38	39	…
47	46	45	44	43	42	41	40	…
48	49	50	51	52	53	54	55	…
63	62	61	60	59	58	57	56	…

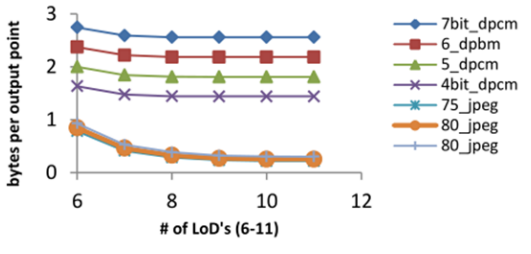


Figure 21 – left) Mapping order for the octree colour values into the image grid [52]; right) Compressed size (in Bytes) for colour coding per output point, for different LoDs, using several colour coding solutions [52].

C. Performance Assessment

This section reviews the experimental results obtained by Mekuria *et al.* in [52]. Three different performance dimensions have been evaluated, namely objective quality, subjective quality and, finally, complexity.

Objective Quality Evaluation

For objective quality evaluation, some tests were performed, notably to compare the just reviewed JPEG-based colour coding solution to the differential pulse-coded modulation (DPCM) coding solution used by PCL that was reviewed in the previous section; the performance comparison has been done for several Levels of Detail (LoDs), i.e. different octree depths.

The number of bytes for colour coding per input point for the DPCM and JPEG solutions while using different LoDs is shown in Figure 21 right). In general, as the LoD increases, the compression efficiency of both coding solutions increases, as more points exist in the same space and the correlation between neighbour points is thus increased. In general, the JPEG solution provides lower bitrates for the same LoD.

The second test consisted in comparing the colour PSNR for each colour component, notably YUV, for the DPCM and JPEG coding solutions, again for different LoDs, see Figure 22. Overall, as the LoD increases, the PSNR also increases and the number of bytes per output point decreases. The JPEG-based solutions (with different compression factors) perform similarly in terms of PSNR to the 4, 5 and sometimes 6 bits DPCM solutions with a rate that is up to 10 times lower.

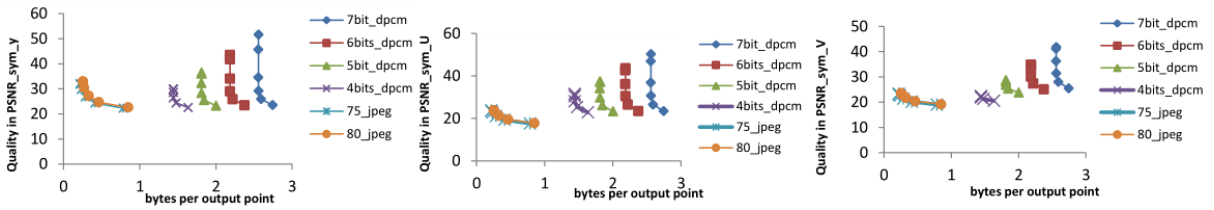


Figure 22 – Rate-quality performance for colour coding, for Y,U and V components, respectively [52].

Subjective Quality Evaluation

The subjective quality assessment was performed using 20 subjects, who were presented a realistic 3D tele-immersive system in a virtual 3D room scenario [52]. The subjects were exposed to three different codecs: intra-coding of geometry with DPCM colour encoding, intra-coding of geometry with JPEG colour encoding and predictive inter-coding of geometry with JPEG colour encoding. The assessment targeted 8 different quality aspects to be evaluated in a 1-5 scale, ranging from Bad to Excellent. One of the quality aspects to be assessed was specifically the colour quality. The results are represented in Figure 23 right), where the three coding solutions are labelled as *Or. (corresponding to DPCM)*, *JPEG* and *Pred*., respectively. The results show that JPEG coding does not introduce significant degradation, implying that JPEG colour coding will achieve bit rates up to 10 times lower than the 8 bits DPCM coding solution only at the cost of a negligible subjective distortion.

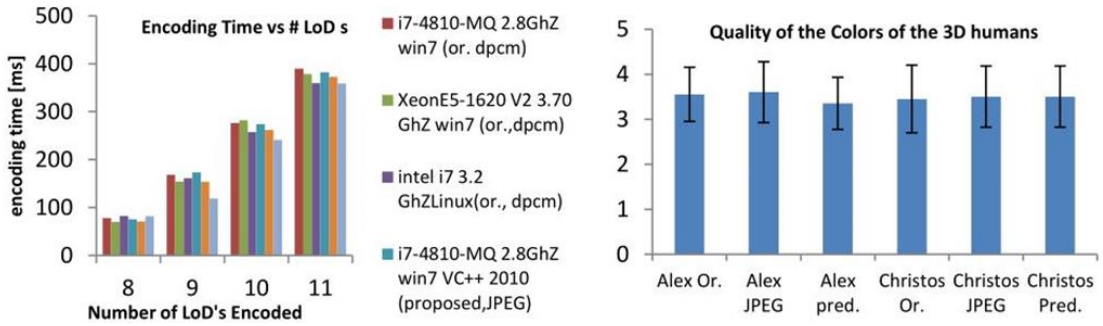


Figure 23 - left) Encoding time for different coding solutions and LoDs with different set-ups; right) Subjective quality scores for colour quality assessment using two datasets: Alex and Christos [52].

Complexity Evaluation

For a dynamic point cloud, it is very important to also assess the encoding complexity, typically through the encoding time. In this case, the encoding time was measured for the DPCM and JPEG solutions using varying LODs, see Figure 23 left). Overall, the encoding time decreases for lower LoDs; for the higher LoDs, the highest encoding time for one frame was around 40ms, which means that these codecs run in real-time or near real-time for intra-coding. For inter-coding, the frame encoding time is about 1 second [52]. The encoding time difference between the DPCM and JPEG solutions was not significant, notably due to the usage of a fast-optimized JPEG implementation.

The main strength of this codec is the usage of JPEG compliant coding for the colour data, with all interoperability advantages this option brings, and the good compression performance regarding the selected benchmarks, e.g. reaching rate reductions up to 10 times when compared to more traditional methods, without significant subjective quality degradation. However, this codec fails to explore the temporal correlation between temporally adjacent point clouds when dynamic point clouds are to be coded.

2.5.3. Graph-based Transform Coding Solution

A. Objective and Main Approach

Graph-based transforms are linear transformations determined by an underlying graph structure and applied to some associated signal(s). This type of transformations have been successfully used for the coding of visual data using several representation models, notably meshes, depth maps, images and videos [54]. The graph-based coding solution proposed by *Zhang et al* [54] here reviewed performs the coding of attributes of static point clouds. It is a lossy coding solution, where the 3D positions of the point cloud are first organized into an octree data structure, from which the graph is constructed and after the transform created.

B. Architecture and Walkthrough

The architecture of this coding solution is depicted in Figure 24.

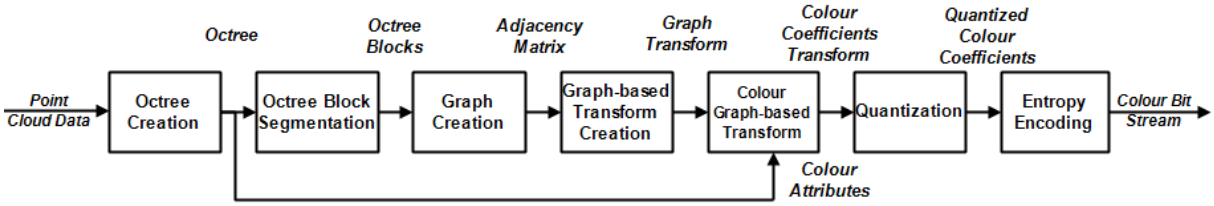


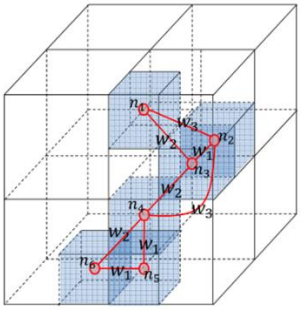
Figure 24 – End-to-end architecture of the Graph-based Transform solution.

The walkthrough for this coding solution is as follows:

- **Octree Creation** - The point cloud data is organized into an octree structure, where each leaf node is a voxel corresponding to a point in the point cloud.
- **Octree Block Segmentation** - The octree is then subdivided into regular blocks of *k*×*k*×*k* voxels, where *N* voxels are occupied with points from the point cloud (*N* may differ from block to block).
- **Graph Creation** – Having the octree blocks, the graph is formed by connecting the neighbouring voxels and assigning a weight to the formed edges. These weights should represent the similarity (or correlation) between the nodes, and thus they are here inversely proportional to the distance between voxels. For instance, in Figure 25 left), an octree block with the occupied voxels and their connections is represented. In this case, $k=4$ and $N=6$, and the assigned weights are $w_1 = 1$, $w_2 = 1/\sqrt{2}$ and $w_3 = 1/\sqrt{3}$. An adjacency matrix, *A,* where each entry corresponds to the weight between the node in the row and the node in the column, is then defined, see Figure 25 right).
- **Graph-based Transform Creation** – Having defined the adjacency matrix, it is possible to define now a diagonal matrix *D* as $diag(d_1,\ldots,d_n)$, where $d_i = \sum_j a_{ij}$. a_{ij} are the elements of the adjacency matrix, *A*. It is also possible to create the so-called *precision matrix*, Q, defined as $Q = \delta\,(D - A)$, where δ is a scaler related to the variance of the signal lying on the graph. Afterwards, the eigendecomposition of *Q* is performed, thus obtaining a diagonal matrix containing its eigenvalues and the eigenvector matrix. The latter is defined as Φ and it is used to transform the signal associated to the graph nodes, in this case the colour components, which is able to very efficiently compact the energy of the signal. The graph transform must be computed for each block of the octree as it depends on the geometry of each block. This implies that the encoder and decoder can construct the same precision matrix for each

block of the octree, without any overhead, given that all the information is on the geometry structure assumed to be already transmitted.

- **Colour Graph-based Transform** – The graph transform, Φ, is a *N*×*N* matrix, computed for each block, considering the occupancy of the voxels. Having the *N* occupied voxels, the colour signal of each node is divided into three *N*×*1* vectors, corresponding to the YUV colour components. The transform is then applied to each vector, resulting into $f = \Phi y$, where here y is the vector derived from the Y component. The same transform is applied to the other two colour components.



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

Figure 25 – left) Octree block with k=4 and N=6 used to form a graph with the corresponding nodes and interconnecting edges; right) Corresponding adjacency matrix [54].

- **Quantization** - Each transform coefficients vector is quantized as $f_q = \mathrm{round}(\mathrm{f}/Q_s)$, where Q_s is the quantization step. After, the quantized coefficients vector is entropy coded.
- **Entropy Coding** – The three resulting transform coefficients vectors are separately entropy coded.

It is important to consider the way the transform Φ is constructed as this impacts the coefficients vector, f, corresponding to each block. For instance, for *m* disconnected subsets of points in a block, the corresponding *m* first f coefficients represent the respective DC terms. Naturally, the rest of the coefficients are AC terms, and the entropy coder should treat the DC and AC terms differently as they have different statistical properties. In this coding solution, an arithmetic encoder is used to encode the AC elements of f_q, assuming that the underlying probability distribution is a zero mean Laplacian distribution. The probability tables are derived from an adaptive Laplacian distribution, where the base diversity parameter (variance of the distribution) is updated as a new coefficient is encoded. On the other hand, for each DC term, a residual signal is computed as the difference between the DC coefficient and an estimated DC mean, while considering the number of connected voxels related to that DC term and the decoded value of the previous DC coefficient. The residue signal is assumed to follow a Laplacian distribution, with an adaptive diversity parameter (variance of the distribution), which is updated every time a new DC coefficient is encoded. The probability tables are constructed using the residue signal probability density function.

C. Performance Assessment

This section will review the experimental results presented in [54], notably in terms of objective quality and complexity, mainly by comparing this new solution with the PCL encoder and another codec using a N-point 1D DCT as transform, where the graph is constructed as a 1D chain and then a 1D DCT is applied [54].

Objective Quality Evaluation

For the objective quality assessment, the octree was limited to 9 depth levels, which corresponds to 512x512x512 voxels.

First, the proposed graph-based transform codec was compared to the PCL codec, which has already been reviewed in this report. The results are presented in Figure 26 left), where the graph-based transform codec is shown to largely outperform the PCL codec. For a SNR of 38 dB, the PCL codec uses 8.40 bits per vertex (bpv), while the graph-based transform codec uses around 0.36 bpv, which implies a compression improvement of 23 times.

After, the graph-based transform codec is compared to another encoder also using a transform in this case a N-point 1D DCT. The PSNR is evaluated while changing the bitrate, for octree depth levels of 5, 6, 7 and 8, which correspond to 16x16x16, 8x8x8, 4x4x4 and 2x2x2 size voxels, respectively. The results presented in Figure 26 right) show that the graph-based transform codec outperforms the N-point 1D DCT for all octree depth levels. For the same bitrate, the 1D DCT transform codec achieves a PSNR which is 1 to 3 dB lower, while for the same PSNR, the 1D DCT transform codec may spend twice the bitrate of the graph-based transform codec.

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

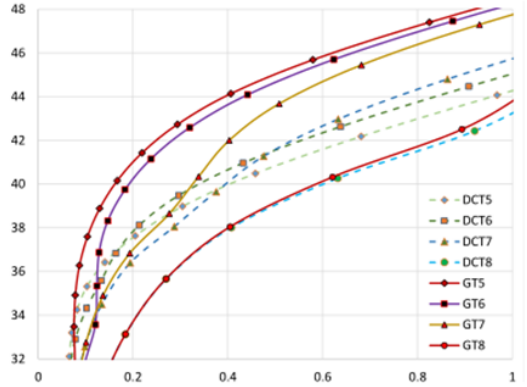


Figure 26 – left) Comparison between the PCL and graph-based transform coding solutions, namely in terms of bitrate and colour components SNR; right) PSNR versus bits per vertex (bpv) rate for different octree levels.[54]

Complexity Evaluation

The complexity of the precision matrix Q eigendecomposition is cubic, i.e. $O(N^3)$. This may yield large encoding times as the octree depth increases, i.e. the number of voxels increases. On a single core 3.0 Ghz CPU, for a point cloud with 520k points, the level 8 graph-based transform takes about 0.85 seconds, level 7 takes around 8.2 seconds, level 6 takes 141 seconds and level 5 takes 109.8 minutes. The authors also refer that the encoding times will be very much reduced if the graph-based transforms of different blocks are run in parallel, on a GPU for example.

The biggest strength of the graph-based transform coding solution is its improved RD performance in comparison with PCL. On the opposite side, one of the biggest weaknesses is its significant computational complexity, notably encoding complexity, which may hinder its application to dynamic point clouds (although the complexity may be largely solved with the evolution of implementation platforms). The redundancy between temporally-adjacent point cloud blocks is also not considered.

Lastly, the adopted solution to create the graphs may generate many isolated sub-graphs, which may imply a lower compression.

D. Coding Solution Extension

As the solution above may generate multiple sub-graphs when sparse static point clouds are targeted, thus becoming rather inefficient, *Cohen et al.* [55] have proposed an extension of the previous coding solution to make it more efficient under those conditions. With this target, two alternative methods are proposed in [55]:

- **Compacting Blocks** - After the block segmentation above described, a compacting process is applied to the points, where all the points (each one corresponds to a voxel) are shifted towards one corner of the block, as depicted in Figure 27 left). This process should ensure that all points will be part of a single graph, implying that when the graph-based transform is applied only one DC coefficient will be produced; naturally, this means that all the remaining coefficients will be AC coefficients. Even though this process aims at a more efficient coding, it may also imply some reduction of the spatial correlation among the points what may be negative in terms of compression. It is worth mentioning that the decoder will reverse this compacting process, as the geometry information is not affected by the proposed compacting process.
- **K-Nearest-Neighbours (K-NN)** – In this case, the block segmentation is the same as for the coding solution presented above [54]. The K-NN solution here presented diverges from the latter, when the graph is being constructed as the graph will not limit the neighbours to one unit apart and each point will be connected to its K nearest neighbours. The resulting duplicate edges will be pruned. This method allows establishing connections between distant points, which implies the integration of more points in a single graph, thus reducing the total number of sub-graphs in a sparse block, see Figure 27 right). However, a single sub-graph for each block is not ensured, unlike the compacting blocks method above. The weight of each graph edge is given by $e^{-d/2\sigma^2}$, where d is the distance between the connected points and σ is the statistical variation of the points.

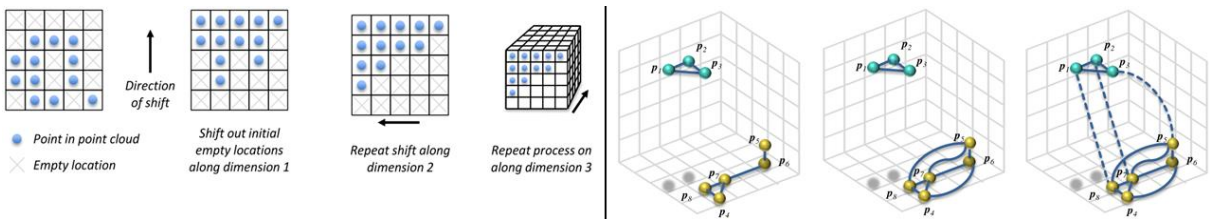


Figure 27 – left) Example of the compacting process; right) Process of aggregating different sub-graphs into one graph using K-NN with K=3.[55]

Figure 28 left) compares the PSNR versus rate in bits per point performance for the *Statue_Klimt_PointCloud* dataset [56], with an octree resolution (voxel edge size) of 0.0081, using three coding solutions: the unmodified graph transform ([54]), the blocks compaction extension and the K-NN extension(with K=8) is plotted. The performance study is made for a varying partition resolution (block edge size - *pr*) parameter, notably *pr* values of 0.5, 1.0 and 1.5 which correspond to maximum block sizes of approximately 61, 123 and 185 voxels, respectively. The results show that, for the unmodified graph transform, the PSNR saturates as the rate increases, while the performance of both the K-NN

and block compaction extensions keeps increasing. Between the two proposed extensions, the K-NN extension outperforms the block compaction extension for all tests. Figure 28 right) illustrates the K-NN extension performance while varying *K* for a *pr* of 1.0. For $K=3$, the performance is rather bad, while for $K=4$ the performance is good for the lower rates. The best compression performance is achieved for $K \geq 8$.

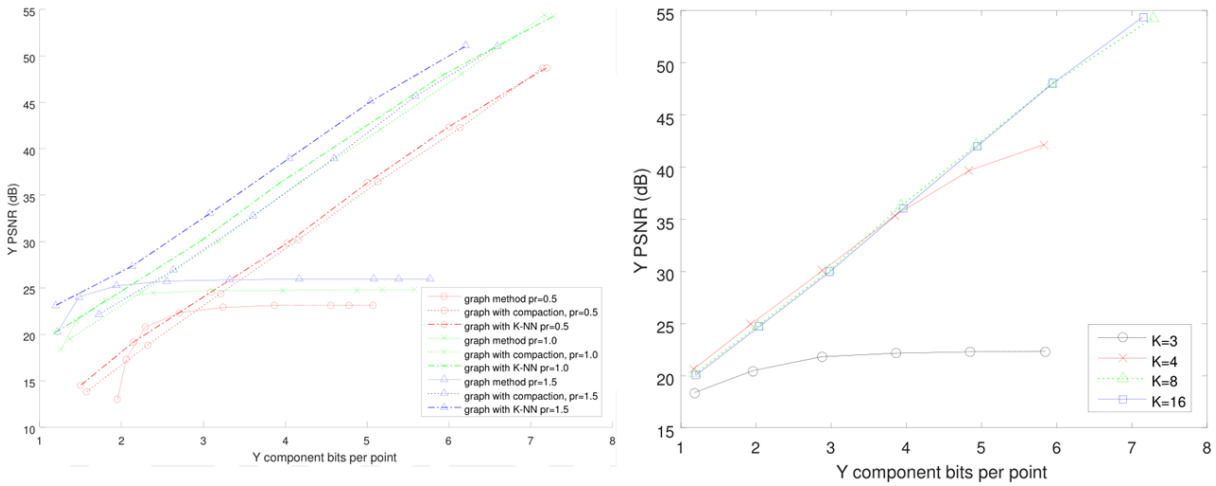


Figure 28 – left) Rate distortion for different coding solutions; right) Compression performance of the K-NN solution for different Ks. [55]

These two new coding methods introduced by *Cohen et al.* allow increasing the compression efficiency of the initial graph-based transform coding solution. However, the blocks compaction process may bring potential loss of spatial correlation among the points, and thus some degree of inefficiency, while the K-NN extension does not always ensure a single sub-graph for each block. Both these solutions still do not consider the redundancy between blocks in the same frame nor redundancy between subsequent point cloud frames.

2.5.4. Graph-based Compression of Dynamic Point Clouds

A. Objective and Main Approach

While the previously reviewed coding solutions adopt the intra-coding paradigm, this section presents a coding solution which also exploits the temporal correlation between frames to compress both the colour and geometry of dynamic point cloud sequences. This codec was designed by *Thanou et al* [2] and proposes to represent the point cloud sequence as a set of graphs, from which motion is estimated following a feature matching approach. After, geometry and colour prediction are performed and, finally, a differential coding scheme is used to compress the geometry and colour, exploiting the estimated motion information.

B. Architecture

The architecture of this coding solution is depicted in Figure 29. It is important to note that the colour of the first frame is coded using the algorithm proposed by *Zhang et al* [54] already reviewed on this report in Section 2.5.3, where each octree is divided in blocks of *k*×*k*×*k* voxels and the graph Fourier

transform is computed for each block. For the remaining frames, only the colour residual is coded with the same transform.

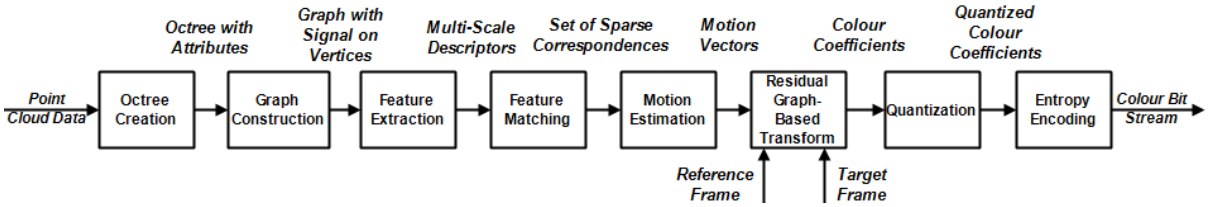


Figure 29 – End-to-end architecture of the Graph-based Compression of Dynamic Point Clouds codec.

The walkthrough for the coding process is as follows:

- **Octree Creation** – First, the point cloud data is partitioned into voxels and an octree is constructed by traversing the tree in a depth first order, for a given depth. At the maximum depth of the octree, all points are mapped to leaf voxels.
- **Graph Construction** – Having the voxel set, each occupied voxel is mapped to a node in a graph. The connectivity between nodes is defined based on a K-NN method, with K=26, which is the maximum number of adjacent neighbours for a node. The weight for each connection is inversely proportional to the distance between the corresponding voxel centres.
- **Feature Extraction** – To find correspondences between the irregular point cloud structures, graph information and the signals residing on the vertices are used to compute feature descriptors for each vertex. In this case, the signals considered are the 3D position coordinates and the colour components. First, features are defined in each node by using the so-called *octant indicator functions*, which will provide the geometry variation in different parts of the nodes neighbourhood. After, the graph spectral features are obtained by using the so-called *spectral graph wavelets* (SGW) computed on the geometry and colour signals and octant indicator functions. The SGW are a family of functions that are used to decompose a signal residing on the vertices of a graph simultaneously on frequency and time (or space) [57]. These set of functions differ from each other by a scaling parameter, *s*. The result of the SGW computation is a multi-scale descriptor for each node, which consists of wavelet coefficients that will be used to determine the correspondences between different frames.
- **Feature Matching** - After having the descriptors, one must find the best match, in this case the node m_n in the reference frame (G_t), for the node n in the target graph (G_{t+1}). For this, a matching score is computed, in this case the Mahalanobis distance, which depends on the features extracted above. The best match for node n is node m_n in the graph G_t for which this distance is minimized. Only significant matches between both point clouds are kept, i.e. the correspondences with the best scores. It is also important to keep correspondences for all areas of the 3D space, and thus the vertices of G_{t+1} are clustered into different regions, keeping only one correspondence – one representative vertex per region. The clustering is performed using K-means, where *N* vertices are partitioned into *K* clusters, by making each vertex belong to the cluster with the nearest position centroid. To belong to the final sparse set of correspondences, the matching distance must be lower than a certain threshold, which will avoid inaccurate matches.
- **Motion Estimation** – Having the set of sparse correspondences, the motion is estimated by computing some motion vectors, i.e. the 3D position difference between the correspondence nodes. It

is assumed that vertices close in 3D positions undergo similar motion. The objective of this module is to estimate a dense motion field, v_t, for all the nodes $m \in G_t$, which is achieved by first calculating the motion vector between the vertex $n \in G_{t+1}$ to its correspondence in G_t, m_n. To calculate the motion vectors for the remaining nodes $m \in G_t$, these motion vectors are considered to be a vector-valued signal that lives on the vertices of G_t. After, the sparse set of already calculated motion vectors are interpolated, by treating each component independently. The motion interpolation is cast as a regularization problem that estimates the motion values for all the vertices by requiring the motion signal to vary smoothly across the vertices connected by an edge. As a result of this process, reshaped motion vectors, denoted as v_t^*, are obtained with a dimensionality of $3 \times N_t$, where N_t is the number of vertices of G_t and the '3' term corresponds to the 3D coordinates. Naturally, $v_t(m)$ denotes the 3D motion vector for the node $m \in G_t$. The motion vectors are then coded in the graph Fourier domain and uniformly quantized, generating $\widehat{v_t^*}$.

- **Residual Graph-based Transform** – From the reference frame, denoted as I_t, and its quantized motion vectors ($\widehat{v_t^*}$), which are both signals living on G_t, it is possible to predict the 3D positions and colour components of the target frame (I_{t+1}) points. However, these frames do not typically have the same size, and thus it is necessary to warp I_t to I_{t+1}, generating $I_{t,mc}$. While the 3D positions of the warped frame, denoted as $p_{t,mc}$, are estimated by exploiting the quantized motion vectors as $p_{t,mc}(m) = p_t(m) + \widehat{v_t^*}(m)$, the colour can be transferred directly according to $c_{t,mc}(m) = c_t(m)$, where $c_t(m)$ is the colour of the node m in the frame I_t. At this stage, the points in $I_{t,mc}$ are used to predict the colour of each point in G_{t+1}, which will be referred to as $\widetilde{c_{t+1}}(n)$, which corresponds to the colour prediction. For each position $p_{t+1}(n)$, $\widetilde{c_{t+1}}(n)$ is obtained by averaging the colour value of the nearest points of $I_{t,mc}$. These set of points are the so-called *nearest neighbours* (NN_n) . Thus, $\widetilde{c_{t+1}}(n) = \frac{1}{|NN_n|}\sum_{m \in NN_n} c_{t,mc}(m)$. The number of nearest neighbours is usually set to 3. Excepting for the first frame, the Graph-based transform [54] is applied to the residual (Δc_{t+1}) of the target frame with respect to the colour prediction highlighted above, computed as $\Delta c_{t+1} = c_{t+1} - \widetilde{c_{t+1}}$.

The quantization and entropy encoding modules are the same as applied in [54], previously reviewed on this report.

C. Performance Assessment

This section will briefly summarize the performance assessment presented in [2], that only focus on objective quality evaluation.

Objective Quality Evaluation

For the colour compression evaluation, two types of tests were performed. In the first one, the performance of the motion estimation and compensation process (warping) is assessed. The coding rate of the motion vectors was fixed to 0.1 bpv and the SNR, computed as $20 \log \frac{\|c_{t+1}\|}{\|c_{t+1} - \widetilde{c_{t+1}}\|}$. Three different prediction cases were considered: (i) SNR_{mc}: the colours of points in the target frame are predicted from their nearest neighbours in the warped frame $I_{t,mc}$; (ii) $SNR_{previous}$: the colours of points in the target frame are predicted from their nearest neighbours in the referemce frame I_t and (iii)

SNR_{avg}: the colours of points in the target frame are predicted as the average colour of all points in the reference frame I_t. The results are depicted in Figure 30 left), for three different point cloud sequences, notably *Yellow dress*, *Man* and *Upper body* [2]. The first method is able to reduce the prediction error, with an average gain in terms of colour prediction of 2.5 dB and 8-10 dB when compared to the second and third methods, respectively.

In the second test, the prediction obtained using the motion estimation and compensation scheme previously described is used in the context of the entire colour compression part of the point cloud codec. The rate and PSNR are measured when different levels of quantization are applied to the colour information for independent (Intra) and differential (Inter) coding. The results can be found in Figure 30 right) for the three selected sequence sets. Each point represents the average PSNR of the RGB components across the first ten frames of each sequence, for a quantization step of 32, 64, 256, 512 and 1024, respectively. For the lower bit rates (larger quantization steps), differential coding provides a gain in PSNR of approximately 10dB when compared to independent coding. For the higher bit rates (smaller quantization steps), the difference between independent and differential coding becomes smaller.

Note that the prediction path has a high impact on the compression performance gain. As the number of successively predicted frames increases, the PSNR is expected to be gradually degraded as there is some error propagation. This problem is mitigated by a periodic insertion of reference frames or optimizing the frequency of this Intra-coded frames.

Sequence	SNR_{mc}	$SNR_{previous}$	SNR_{avg}
Yellow dress	17	15	6.5
Man	13	10.5	4
Upper body	9.8	7.5	1.2

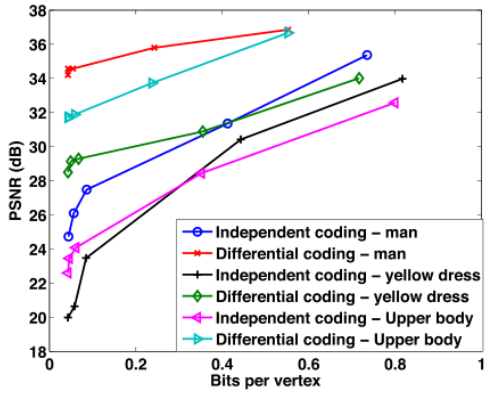


Figure 30 – left) Colour prediction error (SNR in dB); right) Rate-distortion for independent and differential coding [2].

This codec is the first presented which is able to explore the colour redundancy between temporally-adjacent point cloud frames. The differential coding enables gains up to 10 dB for the same rate when compared to independent coding. The biggest downside of this codec is the fact that the graph-based transform applied in the colour coding does not explore inter-block redundancy.

2.5.5. Dynamic Point Cloud Compression with Coding-Mode Decision

A. Objective and Main Approach

This section reviews another dynamic point cloud coding solution proposed by *Queiroz et al.* [3] which adopts a motion-compensation approach to exploit the temporal redundancy. First, the point

clouds are divided into blocks of voxels and, for each block, the encoder decides if intra-coding (I-frames) or motion-compensated coding (P-frames) from the past frame should be performed. As no prediction residue is coded, P-frames are degraded I-frames with an extra distortion that was found to be "worth" in an RD performance sense [3]. Thus, this codec includes a coding mode decision tool to evaluate the trade-off between distortion and coding bitrate using two proposed quality metrics which jointly consider the geometry and colour distortions.

B. Architecture

The basic architecture of this coding solution is depicted in Figure 31. It is worth mentioning that the highlighted modules of Figure 31 are the core innovation of this coding solution.

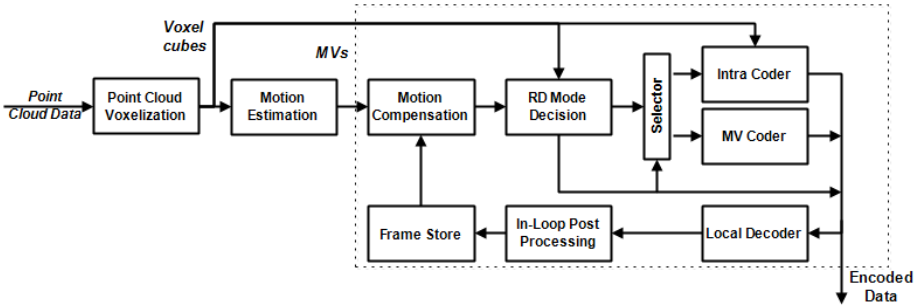


Figure 31 – End-to-end architecture of the coding solution, based on [3].

In terms of colour coding, this solution includes the following steps:

- **Point Cloud Voxelization** – The 3D data is represented by voxelized point clouds. The voxelization process is achieved by quantizing the point positions to a regular lattice $\mathbb{Z}_W^3$, where $\mathbb{Z}_W = \{0, \dots, W-1\}$, being $W = 2^D$ the point cloud width and D an integer related to the representation precision. A voxel is considered to be occupied if it contains a point of the point cloud and unoccupied, otherwise. The authors assume that each occupied voxel has $(\mathrm{x,y,z})$ coordinates that are D-bit unsigned integers, and YUV are 8-bit unsigned integers colour components. Thus, the voxelized point cloud may be described as an arbitrarily indexed list of occupied voxels $\{\mathrm{v_i}\}$. Since these are dynamic point clouds, at every discrete time t, there is a frame $F(t) = \{v_{i,t}\}$, where $v_{i,t} = [x_{i,t}, y_{i,t}, z_{i,t}, Y_{i,t}, U_{i,t}, V_{i,t}]$. It is important to note that there is no direct relationship between voxels from different frames in the same list position, i.e. $v_{i,t}$ and $v_{i,t+1}$, from $F(t)$ and $F(t+1)$, respectively, have no direct relationship. Moreover, different frames often have a different number of occupied voxels. The frame $F(t+1)$ is partitioned into blocks of $\mathrm{N{\times}N{\times}N}$ voxels, generating a list of occupied blocks. Each occupied block at frame instant $\mathrm{t{+}1}$ in the integer position (b_x, b_y, b_z) is composed by the occupied voxels within the block boundaries, i.e. without loss of generality for the 3D coordinates, $b_x N \leq x_{i,t+1} < b_x N + N$.
- **Motion Estimation –** This coding solution exploits the temporal redundancy by using a motion field between every two successive frames, which is considered available. For the motion estimation, the authors use the correspondence/motion fields created by using the solution proposed in [58]. With these correspondences, each voxel in the frame to be coded is first associated with a motion vector pointing to a voxel in the previous, decoded frame. However, for compression efficiency only one MV is coded

for each block (with $N \times N \times N$ voxels). Thus, the MV that is the closest to the MVs average of the block is assigned to each block.

- **Motion Compensation –** Let Ω denote the set of occupied voxels for a given block in frame $F(t+1)$. This set Ω can be predicted from (decoded) $F(t)$, using the motion vector MV (M_x, M_y, M_z), associated to each block such that, without loss of generality for the 3D coordinates, $b_x N - M_x \leq x_{i,t} < b_x N + N - M_x$. Thus, Ω may be predicted by Ω_p, which denotes the set of voxels in $F(t)$ which predict the block in $F(t+1)$ with the motion vector MV (M_x, M_y, M_z), such that $v_{i,t+1} = [x_{i,t} + M_x, y_{i,t} + M_y, z_{i,t} + M_z, Y_{i,t}, U_{i,t}, V_{i,t}]$, where the colour values are equal for both the "reference voxel" $v_{i,t}$ and $v_{i,t+1}$. To evaluate the quality of the prediction of Ω by Ω_p, a local distortion metric δ, which may be computed as correspondence or projection-based, is computed. The correspondence-based distortion is computed by associating the closest voxels in the sets Ω and Ω_p, and summing all their geometric (δ_g) and colour distances (δ_c), i.e. for a pair of Ω and Ω_p, $\delta = \sum(\delta_g + \beta \cdot \delta_c)$. The colour distance is multiplied by a constant, β, which defines the relative importance of these two distortions in the joint distortion metric. On the other hand, for the projection-based distortion, the voxels in Ω and Ω_p are projected onto the six "walls" of the block and then the MSE between the two corresponding images (original and reconstructed) for the Y channel of the individual projections is computed and summed.
- **RD Mode Decision** – Two types of frames exist in this coding solution:
  - I-frames, where all blocks are intra-coded using octree coding for the geometry and the Region-Adaptive Hierarchical Transform (RAHT) proposed in [59] for the colour components;
  - P-frames, where blocks may be intra-coded or inter-coded (motion compensated), with Ω substituted by Ω_p (without residual coding), using a coding mode decision tool which acts at the block level.

To select the coding mode for the blocks in the P-frames, the encoder must test which coding mode is more efficient in an RD sense, which is achieved by comparing the geometry and colours rates and distortions for the intra and inter-coding modes, i.e. $(R_g^{intra}, R_c^{intra}, D_g^{intra}, D_c^{intra})$ and $(R_g^{inter}, R_c^{inter}, D_g^{inter}, D_c^{inter})$, respectively. Assuming that octrees encode geometry at 2.5 bits/voxel [3], one may make the approximation $R_g^{intra} \simeq 2.5\|\Omega\|$. Thus, the intra-coding rate takes the form: $R_{intra} = R_g^{intra} + R_c^{intra} = 2.5\|\Omega\| + R_c^{intra}$. As there is no colour residue in the inter-coding mode, this inter rate corresponds only to the MV encoding rate, where R_{MV} is the average rate to encode one MV. Thus, $R_{inter} = R_g^{inter} + R_c^{inter} = R_{MV}$. Since there is no geometry distortion for octree-based coding, the intra-coding distortion only depends on the colour distortion. Therefore, $D_{intra} = D_g^{intra} + \beta D_c^{intra} = \beta D_c^{intra}$, where β is the same constant presented above. The inter-coding distortion is basically the correspondence-based or projection-based distortion presented above, i.e. $D_{inter} = D_g^{inter} + \beta D_c^{inter} = \delta$. After, the Lagrangian costs for each mode are computed. A block is intra-coded if $D_{intra} + \lambda \cdot R_{intra} < D_{inter} + \lambda \cdot R_{inter}$, for any fixed $\lambda > 0$, and inter-coded, otherwise. To provide a meaning to λ, one may compute $\lambda^* = \frac{D_{inter} - D_{intra}}{R_{intra} - R_{inter}}$, where now λ is a globally advertised threshold, such that a block is intra-coded if $\lambda^* > \lambda$, and inter-coded, otherwise. Finally, the quantization step (Q) of the RAHT coefficients, which is only used in the case of block intra-coding, will also trade-off rate and distortion, as both are

controlled by this key parameter. In fact, both λ and Q influence the RD performance: As λ increases, the number of inter-coded blocks will increase and thus one gets higher distortion and lower bitrates; as Q increases, the distortion for the intra-coded blocks increases and the bitrates decrease. Even though both parameters have the same overall effect on (R,D), the variations caused by fixing one of them and varying the other are very different. Thus, it is important to map (λ,Q) onto (R,D), which is typically done by fixing λ and varying Q, for multiple values of λ and Q, to get the pairs of values (λ,Q) that minimize both the bitrates and distortions. If possible, the relation $\lambda = f_\lambda(Q)$ should be identified.

At the end of the coding process, the coded stream is locally decoded, post-processed to avoid blocking artefacts and to smooth the surfaces, and stored in the Frame Store to serve as the reference for the motion compensation and prediction processes of the following frame.

C. Performance Assessment

This section will briefly review the performance results presented in [3]. Three different point cloud sequences were used for the tests: *Man* and *Loot*, which are both a full-body 3D sequence and *Ricardo*, which is a frontal upper-body in 3D. The chosen lattice width was $W = 512$ and the Group of Frames (GOF) length used was 8 (7 P-frames between each I-frame). Q is varied from 5 to 45 for the above described coding scheme, here referred as Motion-Compensated Intra-Frame Coding (MCIC), and from 10 to 50 for the intra-coding of all the frames.

The first test compares the RD performance of applying MCIC and intra-coding of all the frames to the three sequences. For the correspondence-based distortion metric, it was possible to obtain the relation between λ and Q, derived from one single frame, as $\lambda = Q^2/60$, and it produces very good results. Figure 32 shows that, for all the three sequences, MCIC has a superior performance for every bitrate. On the other hand, for the projection-based distortion metric, it was only possible to find a well performing $f_\lambda(Q)$ relation for the sequence *Man*. This was derived from a single frame with a simple least-squares fitting in the log domain, obtaining $\lambda = e^{5.6329\times10^{-5}\cdot Q^3 - 7.6525\times10^{-3}\cdot Q^2 + 0.35998\cdot Q - 71873}$. The solution only yielded good results for *Man*. The authors also add that, for the sequence *Ricardo*, it is not easy to find a $f_\lambda(Q)$ relation that works well for most frames, as it contains a certain level of noise, it is a rapidly changing sequence and it is only a front-faced sequence, which makes the distortion estimation from the local cube projections more difficult. Ideally, one could fix a λ and choose the optimal Q for each frame through exhaustive search, but this would be complex, notably hindering the real-time application of the encoder.

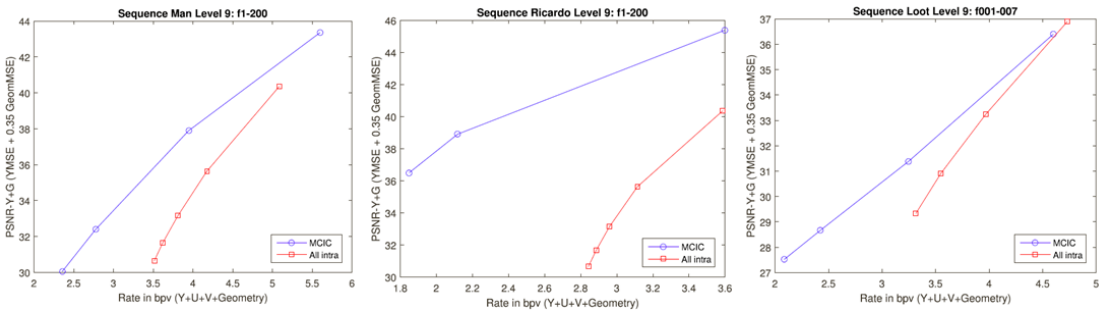


Figure 32 – RD performance results for Man, Ricardo and Loot using Motion-Compensated Intra-Frame Coding (MCIC) with correspondence-based distortion metrics, and full Intra-coding.

This is a novel scheme for dynamic point clouds coding, which main strength is the joint geometry and colour RD optimized selection of the inter and intra-coding modes, achieving superior performance for all bitrates, as depicted in Figure 32. The main improvement areas are the inclusion of B-frames, similarly to traditional 2D video, the development of an optimized motion estimation encoder and the design of an optimized way to obtain $f_\lambda(Q)$. It is worth stressing that, for the inter-coding mode, the colour is precisely the same for the motion-compensated block in $\mathrm{F(t+1)}$ and the $\mathrm{F(t)}$ block taken as prediction as there is no residual coding. Thus, the inclusion of a colour residue, which implies a rate cost and a certain complexity, may reduce the colour distortion and eventually more blocks may be inter-coded for the same λ. By using three coding modes, intra, inter and inter with residue, it may be possible to achieve a better RD performance at the cost of some additional complexity.

2.6. MPEG Geometry-based Point Cloud Coding

When the industry has a need for efficient coding solutions, notably for an emerging type of data as the point clouds, and there is enough prior evidence that there are coding solutions available which may fulfill the minimum relevant set of requirements, MPEG may launch a so-called *Call for Proposals* which basically requests companies, universities, research institutions, etc. to answer with their best technical solutions. At this stage, all interested parties should submit their best coding solutions to be compared among each other and with some anchor coding solutions, e.g. previous standards, using pre-defined objective evaluation metrics and often subjective evaluation protocols.

The best solution or a solution combining the best tools of the best solutions, typically with the core architecture based on the best one, for each pre-defined category, will then become the so-called *Test Model*, which is the first stage of the coding solution, that should become the international standard after some collaborative development. The Test Model solution will then be iteratively improved through the so-called *core experiments*, in which potential improvements to the current Test Model are tested to check if better performance regarding some performance dimension may be achieved. In that case, the Test Model will be updated, and the core experiment tool will be added or some previous tool substituted. Thus, to design the final international standard, the solution in the Test Model goes through several stages of maturity, which have to be approved by the MPEG members and later, more formally, by the National Bodies.

In the case of point clouds, MPEG has launched, in January 2017, a Call for Proposals (CfP) on Point Cloud Coding [6]. Three different categories of point clouds were addressed by this CfP, notably:

- **Category 1:** Static Objects and Scenes;
- **Category 2:** Dynamic Objects;
- **Category 3:** Dynamic Acquisition, which targets content produced by a moving acquisition system, e.g. urban scenarios, autonomous vehicles, etc., which is typically a frame-based laser scan.

The codecs proposed following the CfP, were evaluated with a predefined set of point cloud data sets, where some were intentionally more complex than others. Three fundamentally different test conditions existed, notably: i) Lossless Geometry and No Attributes; ii) Lossless Geometry and Lossy Attributes; and finally, iii) Lossy Geometry and Lossy Attributes. The three test conditions were applied to Category 1, while only the two last test conditions were applied to Category 2. Each proposed coding

solution encoded the data sets (for its category) with a predefined set of bitrates. The results were then objectively and subjectively evaluated. The objective evaluation was based on RD performance, using *PSNR* as the quality measure, using the definition in Section 2.4. The PSNR was computed for both P2Point and P2Plane geometric distortions and also for colour distortion. For the subjective evaluation, the decoding result was rendered with a predefined viewing path, generating a video sequence. All video sequences had around ten seconds and were evaluated by the test subjects using the double stimulus subjective (DSIS) evaluation protocol as defined in Rec. ITU-T P.910 [60].

By the end of October 2017, MPEG has approved the first versions of the Test Models for each point cloud category. Later, in January 2018, significant commonalities between the categories 1 and 3 Test Models (TMC1 and TMC3) were observed, notably in terms of high-level architecture, what led to the creation of a Core Experiment on the convergence of TMC1 and TMC3 [61]. Following the confirmation of these similarities, a joint Test Model was named *TMC13* and later G-PCC (Geometry-Point Cloud Coding), which is able to fulfil the requirements of both categories 1 and 3 point clouds [62].

The Test Model for Categories 1 and 3 (TMC13 or G-PCC) [62] is briefly reviewed here. In the following chapter, the Test Model for Category 2 (TMC2 or V-PCC from Video based-Point Cloud Coding) will be reviewed in more detail, as it has a higher impact on the work reported in this Thesis.

A. MPEG G-PCC Architecture and Walkthrough

The G-PCC Test Model (TM) architecture is depicted in Figure 33. This coding solution is composed by three main modules: i) the geometry encoder/decoder, which uses an octree structure to encode/decode the geometry; ii) the recolouring module, which transfers the colour (or other relevant attributes) to the reconstructed geometry; and iii) a colour encoder/decoder, which may use a spatial transformation based on the so-called Region-Adaptive Hierarchical Transform (RAHT), highlighted in green, in alternative to the Predicting and Lifting Transform tools, highlighted in orange, which will be explained later. It is worth mentioning that the green and orange modules comprise techniques that are typically used for Category 1 and Category 3 point clouds, respectively. It is important to note that even though the most relevant attribute is colour, the architecture modules that operate on colour may also operate on other different attributes with the appropriate modifications.

The walkthrough of G-PCC Test Model is as follows:

- **Geometry Encoder/Decoder** – The geometry encoder receives the original set of point cloud positions and starts by transforming the geometry data from possible application-specific domains, which may have floating-point values, to finite-resolution internal spaces, on which all the remaining modules will operate. The transformed 3D positions are then quantized and, as a consequence, multiple points may have the same position. These are the so-called duplicate points and may be optionally removed. Following this step, the quantized 3D positions are encoded using an octree coding approach, where one can either choose to code the octree structure with full precision, i.e. from the root all the way down to the leaves with minimum possible size voxels, or to code using a pruned-octree, i.e. from the root down to leaves of blocks larger than the minimum possible size voxels, and approximate the geometry within each block using a parametrized surface model. The first technique is known as simple

octree coding, while the second, which will be explained in more detail later, is known as TriSoup, an amalgam for Triangle Soup [63].

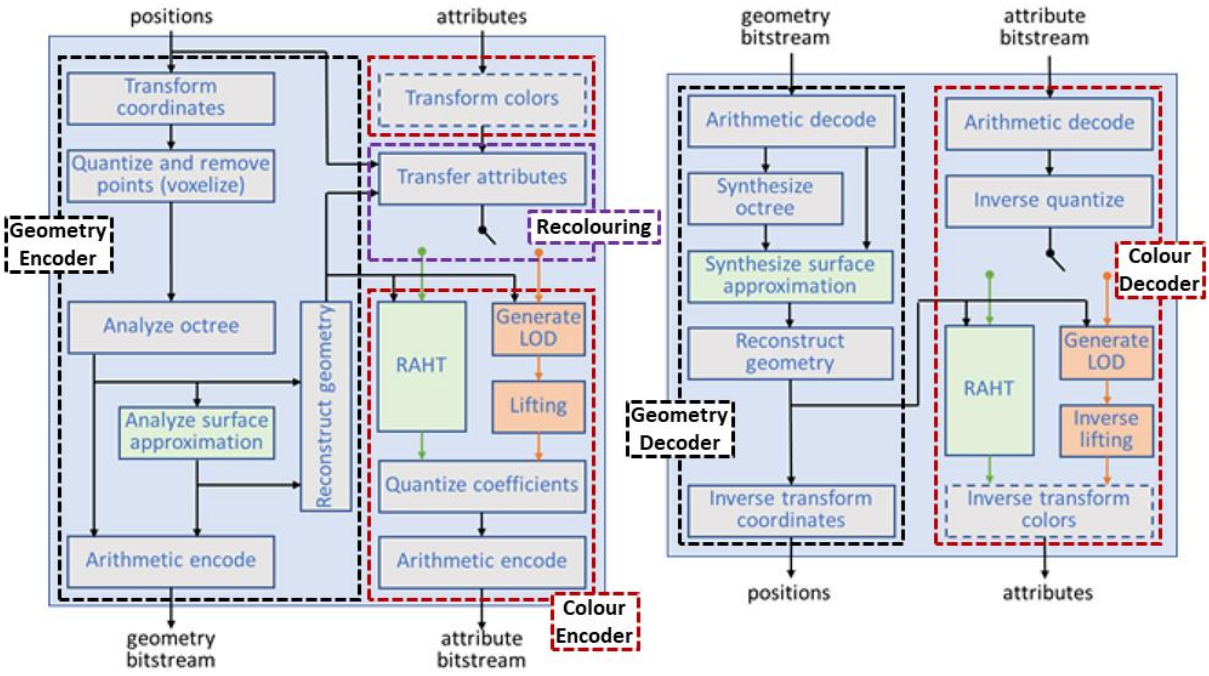


Figure 33 – G-PCC coding architecture, based on [62] – the geometry encoder/decoder, recolouring, and colour encoder/decoder modules are highlighted in black, purple and red-dashed rectangles, respectively.

- **Recolouring** - After encoding, a decoded/reconstructed version of the geometry is made available at the encoder side, since the colour will be coded for this reconstructed, and in fact degraded, version of the original geometry. Since the decoded geometry may be lossy (notably in terms of number of points and their positions), the attribute transfer module has the task to 'migrate' the attributes, namely the colour (thus the name recolouring), from the original to the decoded/reconstructed geometry with the aim of minimizing the attributes distortion as it does not make much sense to be coding attributes/colour for points that do not even exist.
- **Attribute/Colour Encoder/Decoder** – The attribute encoder and decoder may be used for many types of attributes, but it is currently mostly used for colour. Alike the geometry encoder, the colour encoder receives the point cloud original set of colours and starts with an optional conversion from the commonly used RGB to YCbCr colour spaces, which are then provided to the recolouring module. After the recolouring process, the set of colours associated to the reconstructed geometry may be encoded with one of the three alternative colour coding tools, namely Region-Adaptive Hierarchical Transform (RAHT), interpolation-based hierarchical nearest-neighbour prediction (Predicting Transform), and interpolation-based hierarchical nearest-neighbour prediction with update/lifting step (Lifting Transform). Typically, the first and third tools are used for Category 1 point clouds, while the other tool is used for Category 3 point clouds [62].

B. Main MPEG G-PCC Tools

Among the main G-PCC tools, it is worthwhile to detail the following:

B.1. Triangle Soup (TriSoup)

As mentioned before, one may choose to code the geometry with a pruned-octree, where the geometry of each block, which must be at least 2×2×2 minimum size voxels, is approximated by a surface using a poligonal model. When this tool is applied, an initial surface intersecting each edge of the block at most once, is fitted to the points in the block. Each block has 12 edges and each edge may be shared by the 4 adjacent blocks. The intersections are called *vertices*, and they are only detected if there is at least one occupied voxel adjacent to the edge among all blocks sharing the edge. The position of the intersecting vertex along an edge is coded as the average position for the edge of all blocks sharing the edge. With the collection of vertices coded, it is possible to reconstruct a surface corresponding to a non-planar polygon passing through the vertices a collection of triangles. TriSoup uses triangles for the surface reconstruction, as they are particularly friendly to standard graphics processors.

B.2. Region-Adaptive Hierarchical Transform (RAHT)

The RAHT architecture is depicted in Figure 34. If the RAHT is chosen to code the colour (or other attributes), the points are grouped into voxels (at a certain resolution) and the point colours belonging to the same voxel are averaged and assigned to that voxel. These colours are spatially transformed with the RAHT [59], thus obtaining the transform coefficients $(TY_n, TU_n, TV_n), n = 1, \dots, N_{vox}$, where N_{vox} is the number of voxels. Then, the transform coefficients are uniformly quantized with a given stepsize that must be provided to the decoder. Finally, the quantized coefficients are entropy coded using an arithmetic encoder.

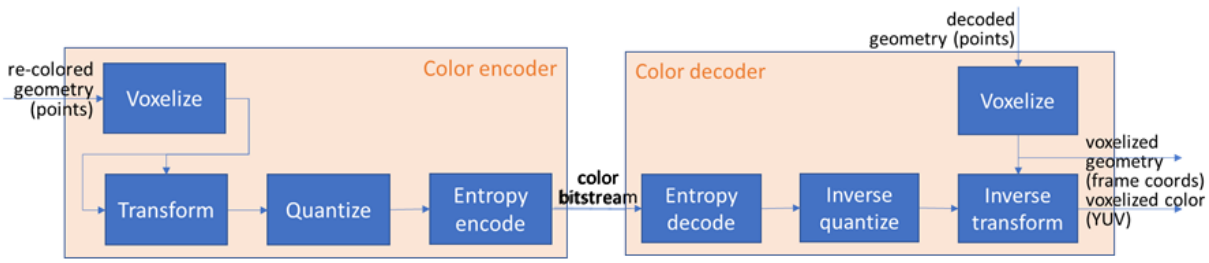


Figure 34 – RAHT-based colour encoder and decoder architecture, based on [63].

B.3. Predicting Transform

For the Predicting Transform colour coding alternative, the colours (or other attributes) are encoded with a prediction scheme, where at each coding step only the already encoded colours are considered for prediction. Since, the points encoding order is very important for this specific coding tool, the first step is the so-called *Level of Details* (LOD) generation process, targeting the definition of an encoding order that is directly related to the segmentation of the points into different levels of details. Naturally, this process intends to improve the prediction efficiency as opposed to applying the predicting transform directly to the points ordered according to the octree decoding process. The predicting transform corresponds to a linear interpolation process based on the distances to the (already encoded) k-nearest neighbours of the point which colour is being coded. After applying this transform, the colour prediction residuals, i.e. the difference between the colour values to code and the predicted ones, must be computed, quantized and entropy coded with an arithmetic encoder.

B.4. Lifting Transform

The Lifting Transform tool is an enhancement of the Predicting Transform previously described, where two main tools are added: adaptive quantization and update operator. Both tools consider the fact that, for any prediction scheme, the encoding order makes the first encoded point colours/attributes more influential. Thus, an influential weight, $w(P)$, is computed for each point *P*. This weight shall increase with the said influence of the point on the prediction process. For the adaptive quantization, each transform coefficient (meaning residual) is multiplied by a factor $\sqrt{w(P)}$, thus implying that the points with more influence will be less quantized. On the other hand, the update operator leverages the prediction residuals and the influential weights to update the colour/attribute values at each stage of the prediction scheme.

At the decoding side, all the described encoding processes are reversed in order to obtain a reconstructed point cloud as close to the original as possible; naturally, this critically depends on the used quantization steps.

Chapter 3

3. MPEG Video-based Point Cloud Coding

After covering the most relevant coding solutions in the literature, including the MPEG G-PCC Test Model, this chapter will review the Test Model for Category 2 point clouds, now referred as V-PCC [64]. As Category 2 refers to dynamic point clouds, time and temporal redundancy play here a major role. This MPEG coding solution is the framework for the novel coding solution proposed in Chapter 4 and, thus, it will be covered here in more detail; special attention is given to the patch generation process, which is the most important coding module for the core contributions of this Thesis. Hence, this chapter starts by providing an overview of the overall V-PCC architecture and later details the patch generation process.

3.1. Architecture and Walkthrough

The V-PCC architecture is depicted in Figure 35. The main idea behind this Test Model is to first map the point cloud data, both texture and geometry, into video sequences and then use a standard video codec to encode these data. This approach largely relies on the very efficient standard video coding developed during the last two decades, notably the more recent HEVC standard. The point clouds frames are coded using the so-called *Group of Frames* (GoF) where the first frame is Intra coded and the remaining frames are Inter coded, meaning that they are coded exploiting the temporal redundancy.

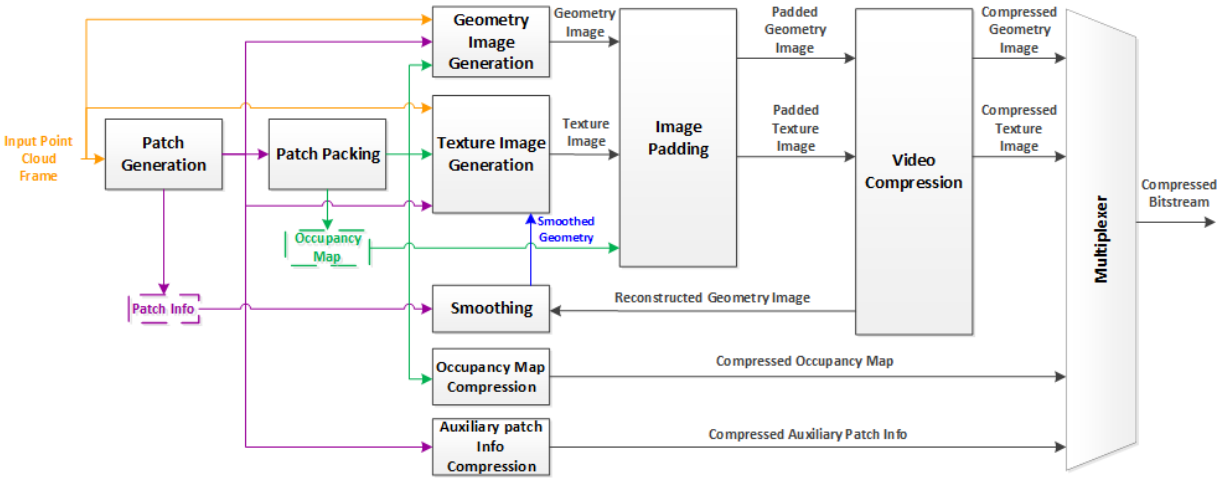


Figure 35 - V-PCC encoding architecture, based on [64].

The V-PCC encoding process works as follows:

- **Patch Generation -** The point cloud is decomposed into the minimum necessary number of patches with smooth boundaries, what is achieved by first calculating the normals at every point. Then, these normals are compared to the normals of six pre-defined planes, each with a different orientation along the 3D axis (*X, -X, Y, -Y, Z* and *-Z* respectively), and basically the planes corresponding to the bounding

box of the point cloud. The points are then associated to the plane with the most similar normal to form clusters, to which is associated an index. The cluster index associated to each point is then iteratively updated taking into account its own normal and the nearest neighbours cluster index in order there are no isolated points with a specific index in the middle of points with a different index. Finally, the patches are created by applying a connected component extraction procedure. A patch is typically defined as a surface or a portion of a surface, used to represent a set of points [65]. In V-PCC, each patch is characterized by the minimum value of the *x,y,z* coordinates of the represented points, by a projection plane index and by a depth-map, where the "depth" represents the third point coordinate value. The clusters associated to the same axis, i.e. *X* and *-X*; *Y* and *-Y; Z* and *-Z*, will be associated to the same projection plane (*YoZ*, *XoZ* and *XoY*, respectively) and the depth "axis" will be aligned with the remaining axis (*X, Y* and *Z*, respectively). The depth maps will later be leveraged to construct the Geometry Image, to be coded by HEVC.

- **Patch Packing –** The extracted patches are mapped into a 2D map, in a process named as *packing*. The packing strategy tries to iteratively insert patches in a *W*×*H* map, while minimizing the unused space, and ensuring that every *T*×*T* map block (e.g. 16x16) is associated with one single patch, where *T*, *W* and *H* are user-defined parameters. As a consequence of this process, an *occupancy map* is produced, indicating for each cell in the packing map if it belongs or not to a point cloud patch. An example of the patches mapped into the 2D map is depicted in Figure 36 - left). To define the patch packing order for the first frame, the patches are sorted in a descending order based on their vertical size, then on the horizontal size and, finally, on the patch index. For the remaining frames in the Group of Frames, the patch packing order is initialized as for the first frame and then for each current frame patch, the best matching patch in the previous frame is searched. To evaluate if the patches are a good match, both patches must have the same projection plane and get a good score on a selected similarity metric that considers the bounding boxes of both patches in the depth maps. The metric computes the ratio between the intersection area over the union area of the two bounding boxes, thus the name *Intersection over Union (IOU).* For each patch in the current frame, the patch in the previous frame achieving the highest *IOU* is chosen; if this *IOU* is greater than a threshold, the current patch is put in a buffer with the relative position of the matching patch in the previous frame. After iterating all the patches of the current frame, the patches in this buffer are packed first with the relative order of their match in the previous frame, while those not part in the buffer are packed afterwards with the initial relative order. Even though, the temporal correlation exploration is of the utmost importance to improve the solution coding efficiency, this tool has only been added in the version 2.0 of V-PCC. Since the V-PCC version used in this Thesis is 1.2, this specific tool is not included.
- **Occupancy Map Compression –** The occupancy map is encoded with a *B0* precision, meaning that the binary occupancy values are associated with *B0*×*B0* sub-blocks. A value of 1 is associated with a sub-block if it contains at least a filled/non-padded pixel, and 0 otherwise. It is important to note that each *B0*×*B0* sub-block is associated with a single *T*×*T* block. Typically, *B0=2* or *B0=4* achieve good subjective results, while significantly reducing the occupancy map coding rate. Naturally, if lossless quality is a requirement, *B0* must be set to 1 as this is the case where no occupancy expansion (meaning no creation of additional points) is made; naturally, this level of accuracy is paid with additional rate.

- **Auxiliary Patch Information Compression –** Besides encoding the depth-map for each patch, it is also necessary to encode other metadata to be able to reconstruct the patches at the decoder side, notably the index of the projection plane, the minimum value of each *(x,y,z)* coordinate and other information to indicate to which patch belongs each *T*×*T* occupancy map block.

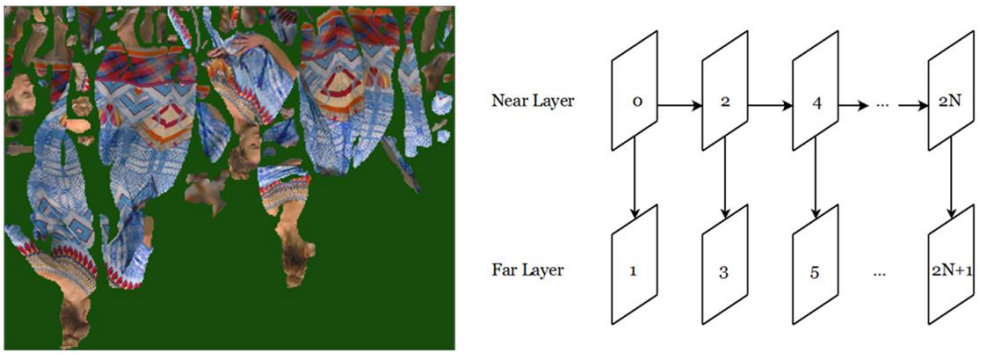


Figure 36 – Left) Packing of the patches into a 2D map; Right) Typical layer and image sequence structure[64]. .

- **Geometry Image Generation –** After the packing, the geometry image generation process takes place, where the 3D to 2D mapping performed during the packing process is exploited to represent the geometry as an image or several images (in practice as video for dynamic point clouds). Since each patch is associated to a certain area of the 2D map, the geometry image pixels in that area will be filled with the depth values corresponding to patch points. In the 3D to 2D mapping, multiple 3D points may correspond to the same mapped pixel, e.g. the outside and inside of an object. To reduce the impact of this effect, each patch is projected onto two maps, so-called *layers*, which means that, in fact, each patch depth-map is composed of two layers. The first layer, named *near-layer*, stores the points with the lowest depth (*D0*), while the second layer, named *far-layer,* stores the points with the largest depth (*D1*) within the interval [*D0*, D0+*ΔD*], where *ΔD* is a user-defined parameter, e.g. *ΔD*=4. The typical layer and sequence of images structure is illustrated in Figure 36 - right).
- **Smoothing –** Because the reconstructed geometry is essential for colour encoding, the encoder performs geometry decoding. The final reconstructed geometry includes a *smoothing* process to avoid discontinuities that may exist at the patch boundaries due to coding artefacts.
- **Texture Image Generation –** As the decoded point clouds may be lossy (regarding the number of points and their positions), the texture image generation process must have access to the decoded/reconstructed geometry to compute the colours associated with the decoded points since there are the points that have to be finally coloured at the decoder. The colours to be coded are determined by a so-called *recolouring* process which transfers the colour information from the original to the reconstructed point clouds. Each non-padded pixel in the geometry image is associated to a reconstructed point, and naturally the same pixels on the texture image to be coded will now be set to the respective reconstructed point texture value.
- **Image Padding -** Between the patches projected in the (texture and geometry) maps, there are empty spaces that are filled using a so-called *padding process*, which will generate piecewise smooth images for video compression. At the decoder, the padded pixels will not be converted to points, as the occupancy map dictates that those pixels do not belong to point cloud patches.

- **Video Compression -** The successive generated texture and geometry maps are coded as video frames using the HEVC video coding standard. These videos are both YUV420 with 8-bit sample depth, where the geometry is monochromatic to represent the third point position component.

Finally, all generated coded streams are multiplexed into a single compressed bit stream.

At the decoder side, the four different coded streams are demultiplexed. After, the geometry is reconstructed by leveraging the patches information, the occupancy map and the geometry images. Lastly, the reconstructed points, which have got their positions from a certain pixel from the decoded geometry image, get their texture values from the same pixel from the decoded texture image.

It is important to mention that, for the V-PCC codec, the number of original and decoded points may be very different, notably the number of decoded points may be much larger which is rather unusual. This is largely due to the *B0* parameter which, e.g. when *B0*=4, adds points to the decoded point when filling the *B0*×*B0* sub-block. This increased number of decoded points creates more dense point clouds, and also more dense surfaces, which was typically more the role of the rendering process.

3.2. Patch Generation Process

Patch generation is one of the most critical processes in the V-PCC encoding architecture, as it has the key task to segment the point cloud into smaller parts, so-called *patches*, which will have its geometry and attribute values projected to a sequence of 2D images, later encoded with a standard video codec. Hence, a poor patch generation result will most likely jeopardise the performance of the following modules in the architecture.

The architecture of the patch generation process is presented in Figure 37. The main idea behind this process is to iteratively generate patches, notably considering the non-represented (e.g. occluded) or poorly represented original points, until the minimum set of patches satisfactorily representing the point cloud geometry is obtained.

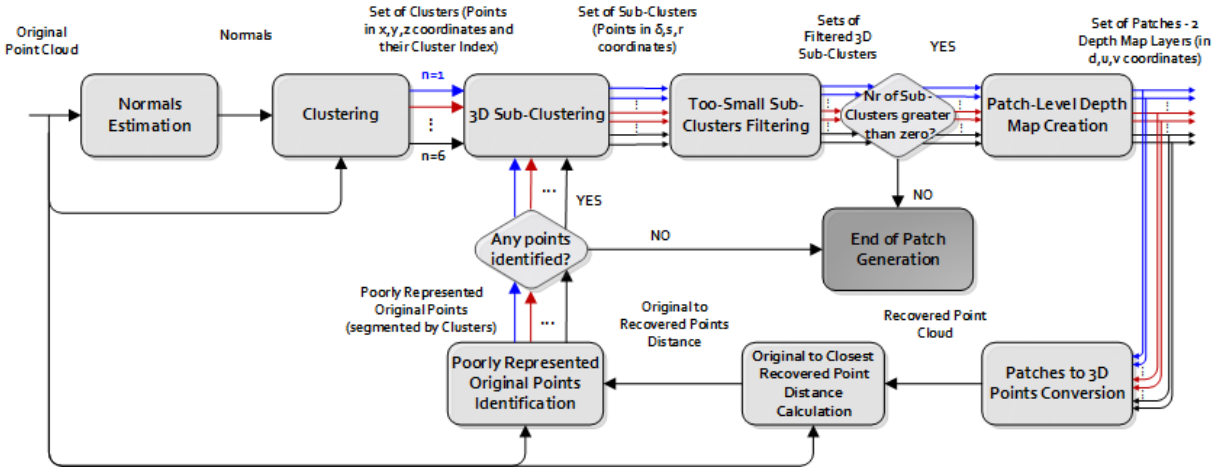


Figure 37 – Patch generation architecture.

The walkthrough for the patch generation proceeds as follows:

- **Normals Estimation –** Taking the original point cloud, normals are estimated for each point by applying a Principal Components Analysis (PCA) to the set of its *N* nearest neighbours (and the point itself), where *N* is a user defined parameter, e.g. *N=16*. PCA is a technique typically used to identify

patterns in data, which in this case starts by computing the set of points coordinates average to compute after the covariance matrix of the difference between the points geometry and the just obtained average. Next, an eigendecomposition is applied to the covariance matrix to obtain, for each point, three pairs of eigenvectors (principal components) and eigenvalues. By definition, each eigenvector is a linear combination of the *x,y,z* coordinates and all the eigenvectors are orthogonal to each other, thus the number of pairs being three. Moreover, the eigenvectors inform the directions of the axes holding most of the data information, i.e. the projection of the data onto the first principal component axis should provide the largest variance possible among all possible axis. Furthermore, the projection of the data onto each succeeding component axis should provide the largest variance possible, given the constraint that each component is orthogonal to the preceding ones [66]. On the other hand, the eigenvalues correspond to the variance of the data projection onto each respective eigenvector axis. Since the set of points typically compose a surface, the two eigenvectors with largest variance are very likely collinear to the surface, while the other eigenvector will consequently be perpendicular to the first two, and in practice to the surface, thus motivating its choice as the normal for the point.

- **Clustering –** The estimated normals for each point are then compared to the normals to six pre-defined planes, notably (*-X, X, -Y, Y, -Z* and *Z*), corresponding to the faces of the object bounding box, by computing the dot product between each normal and each of the six pre-defined planes normals. Each of these planes will have associated a cluster of points based on the similarity of the point normals and their normals themselves. At this stage, each point is associated to the cluster maximizing the dot product between its normal and the plane normal; to each cluster is associated a cluster index, notably 0,1, 2, 3, 4 or 5. This process may generate some isolated points with a specific index in the middle of points with a different index, this means some outliers. This issue is solved by refining the clustering, which in practice updates the cluster index of each point while taking into account its own cluster index and the nearest neighbours cluster indices. Figure 38 illustrates this process for the original frame 1300 of the Longdress sequence [4], notably by showing the result of the initial and final clustering, respectively, where for both clustering results, the points with the same colour share the same cluster index.

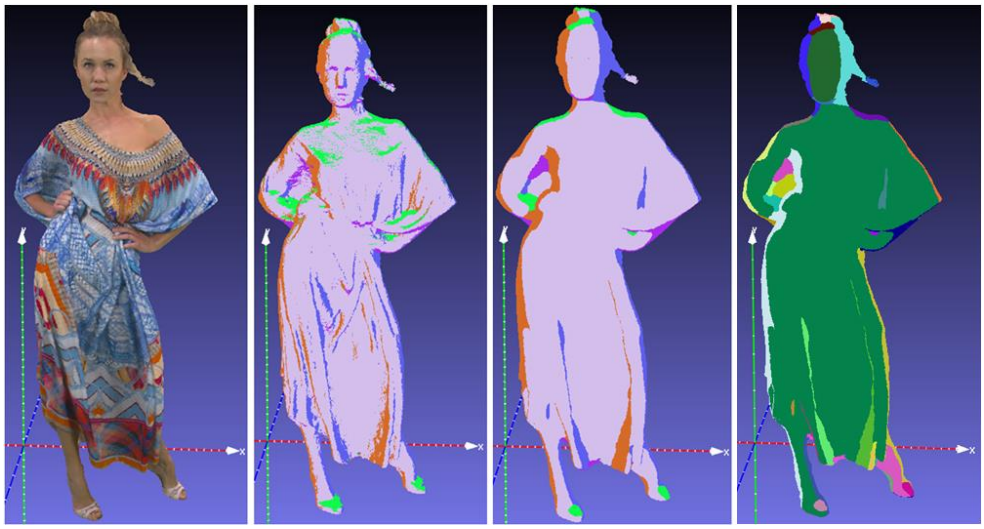


Figure 38 - Clustering results for frame 1300 of Longdress sequence [4]; From left to right: Original point cloud, initial clustering, final clustering after refinement and complete set of sub-clusters .

- **3D Sub-Clustering –** The following step consists in segmenting the just created clusters into sub-clusters of points (or *connected components* as designated in the V-PCC description [64]) that are spatially connected. This segmentation is done by picking a point, the initial *seed*, and checking if its closest cluster neighbouring points already belong to another sub-cluster. If they do, they are skipped; if they don't, the cluster neighbouring points are included in this seed's sub-cluster and, afterwards, one at a time, they recursively become themselves the *seed*. Each time a new seed is iterated, more points may be included in the sub-cluster, and thus more seeds may be created. This recursive process continues until there are no seeds left to iterate. At this point, the sub-cluster is complete and another point, which does not belong to any sub-cluster, is chosen to be the new seed and the same process takes place. When all cloud points have been iterated, the set of sub-clusters is complete. An illustration of the complete set of sub-clusters for frame 1300 of the *Longdress* point cloud sequence is depicted in Figure 38.The sub-clusters with cluster indices associated to the same axis (*-X* and *X*; *-Y* and *Y,- Z* and *Z*) will be associated to the same projection plane (*YoZ, XoZ and XoY*, respectively), and thus will have the same projection index (0, 1 and 2, respectively). To ease the process of projecting the geometry information into the coming depth maps, each sub-cluster is attributed its own 3D local coordinate system defined by δ (depth), *s* (tangent) and *r* (bi-tangent), where the depth axis is aligned with the normal of the sub-cluster projection plane. Naturally, the sub-clusters sharing the same projection index will also share the same local coordinate system. In V-PCC, the transformation from the global coordinate system (*x*,*y*,*z*) to the local coordinate system (δ,*s*,*r*) is performed as follows: **i)** projection index 0: δ=*x, s*=*z* and *r*=*y*; **ii)** projection index 1: δ=*y*, *s*=*z* and *r*=*x*; and **iii)** projection index 2: δ=*z*, *s*=*x* and *r* = *y*.
- **Too Small Sub-Clusters Filtering –** As some of the sub-clusters may have too few points, and since the coded size of the patch information and other metadata is independent from the sub-cluster/patch size, the rate cost (bits per point) is much greater for these sub-clusters with few points than for sub-clusters with a large number of points. For this reason, a filter is applied at the end of the sub-clustering process where the sub-clusters with less points than a threshold, the so-called *minPointsCountPerCCPatchSegmentation* (e.g. 16), are discarded; this implies that these points will not be part of any sub-cluster, and thus will not be part of the decoded point cloud. If after the filter, no sub-clusters are left, the patch generation process ends, as no patches will be coded.
- **Patch-Level Depth Map Creation –** After the filtering, each sub-cluster will be converted to a patch which has associated a depth map. The architecture of the patch-level depth map creation is illustrated in Figure 39.

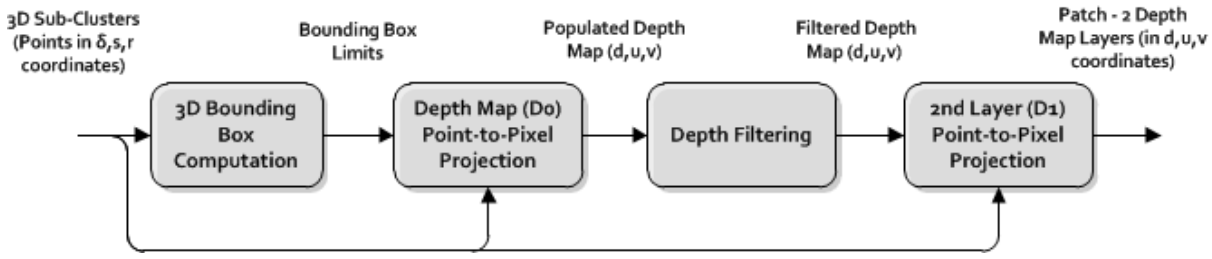


Figure 39 - Patch-level depth map creation architecture.

The patch-level depth map creation process works as follows:

- ***3D Bounding Box Definition* -** First, the 3D sub-clusters bounding box is defined, thus indicating the maximum and minimum value for each coordinate in the local coordinate system (δ,s,r).
- ***Depth Map D0 Point-to-Pixel Projection* -** With the information of the bonding box limits, the 2D depth map is created, where the horizontal (u) and vertical (v) dimensions will correspond to the difference between the maximum and minimum values of the 3D tangent (S) and bitangent (R) axis, respectively. The minimum values for each coordinate are then stored as metadata. On the projection process, each point belonging to each sub-cluster is projected to the depth map near-layer, *D0* (introduced in the previous section), where the (δ,s,r) values will be coded differentially to the minimum value of each coordinate, generating d (pixel coded value), u (depth map columns), v (depth map rows) coordinates, respectively. If multiple points fall into the same pixel, the point with the lowest depth will be chosen (also for the texture), whereas the other points may be included in *D1* or simply discarded from the patch.
- **Depth Filtering -** After, the populated depth map pixels are grouped into $T \times T$ blocks (these blocks can only be associated to a single patch) and, for each block, a pixel will be discarded if: i) its depth value is greater than 255 (8-bit maximum value) or ii) the difference between the depth value and the block lowest depth value is greater than 32.
- **2nd Layer (D1) Point-to-Pixel Projection -** Finally, *D1* is filled with the points that: **i)** were not included in *D0*, due to multiple points falling into the same pixel; and **ii)** have the maximum depth value for a given depth map pixel, while respecting the *surface thickness* (ΔD) constraint, i.e. the difference between *D1* and *D0* depth values for the same depth map pixel must be up to the *surface thickness* threshold (ΔD), e.g. $\Delta D=4$. The idea behind the surface thickness threshold is to have a second depth layer that is able to represent more of the points falling into the same pixels at *D0* layer and have a rather close value (depth) between them (thus the name surface thickness). *D1* (depth map far layer) is firstly copied from *D0* (depth map near layer). Then, the pixels corresponding to new projected points (in *D1*) are changed to the new respective depth value, and finally the obtained *D1* depth maps are differentially coded from *D0*. Including *D1* layer should not spend much rate as *D1* will have a rather low variance, and thus the video codec can efficiently encode this data. In a more recent V-PCC version [64], notably version 3, there is another patch projection mode, which stores the maximum depth value in *D0* and stores the minimum value respecting the surface thickness constraint in *D1*. In V-PCC version 3, the first patch projection mode is known as *mode 0*, while the latter is known as *mode 1*. At the frame-level, there is also a frame projection mode, which evaluates if it is better to simply use the patch projection mode 0 for all patches, or if it is worth to consider both projection modes and choose the one representing more sub-cluster points in the two depth map layers. As previously mentioned, the V-PCC version used in this Thesis is 1.2, which means that this projection mode is not included.

- **Patches to 3D Point Conversion –** After all filtered sub-clusters are converted into patches, a recovered point cloud geometry, which is supposed to be as close as possible to the original geometry, is obtained by converting back all patches depth maps to 3D points.

- **Original to Closest Recovered Point Distance –** Having the recovered point cloud geometry, the distance between each original point and the closest point on the recovered point cloud is computed.
- **Poorly Represented Points Identification –** If for a given point, the just calculated distance is greater than a pre-defined threshold, *dist2MissedPointsDetection* (typically 1), the original point is considered to be poorly represented by the patch. These original points are also known as *missed points*. On the one hand, if no poorly represented original points are found, meaning that the patches are representing well enough the geometry of the original point cloud, the patch generation process ends, and all the patches are given to the packing module. On the other hand, if poorly represented original points are identified, these go back to the 3D sub-clustering module, where new sub-clusters with only poorly represented points will be created, converted to patches and the identification of poorly represented points will be performed again. These iterations will continue until one of the stopping criteria is fulfilled, i.e. no poorly represented points are identified or no more sub-clusters to be coded (after filtering small sub-clusters). For instance, to be properly represented by the patches depth maps, frame 1300 of the *Longdress* sequence iterates three times over the patch generation loop, each time increasing the number of represented points. The result of converting back all generated patches to points after each iteration is shown in Figure 40, where the colour has been attributed to each point using the V-PCC recolouring process.

Figure 40 – Point clouds resulting from converting the depth maps back to 3D points after the 1st, 2nd and 3rd patch generation iterations, respectively, and performing V-PCC recolouring from the original point cloud (frame 1300 frame of the Longdress sequence [4]).

In the patch-level depth map creation module, multiple (3D) points may fall into the same pixel and thus not included either in the near (*D0*) nor far-layer (*D1*) depth maps, or they may be filtered, thus implying that some points from the original point cloud are not coded with the patch depth maps, leading to the so-called discarded points. It is important to note that if *dist2MissedPointsDetection* ≥ *1*, some of these original points that have been discarded will not be identified as poorly represented (they are

considered to be sufficiently well represented), and thus will never be coded. Moreover, even if the original points are identified as poorly represented, they will probably not belong to any sub-cluster again (and consequently not coded), if their distance to the closest recovered point is lower than the threshold *dist2MissedPointsSetection* (typically the distance is 3), since these missed points cannot be the initial *seed*. This means that the original points with a distance to the closest recovered point cloud greater than *dist2MissedPointsDetection* and smaller than *dist2MissedPointsSetection* are considered poorly represented but are often not coded in the depth maps, as these points would most likely cost much rate to be represented by the depth maps without any error, which in turn would not be worth it in an RD sense.

Chapter 4

4. Proposing Adaptive Plane Projections for V-PCC

Following the V-PCC codec reviewing, with special focus on the patch generation process, this chapter proposes a novel coding technique adopting a flexible plane projection approach, the so-called *adaptive plane projection* (AP2V-PCC). The objective of this technique is to increase the coding flexibility and adaptability in terms of projection planes and thus obtain better compression performance when this technique is integrated in the V-PCC framework, in the context of static coding. Some preliminary tests were performed to analyse the colour and geometry RD performance of V-PCC and G-PCC considering static coding, where V-PCC showed superior performance and thus was chosen as the starting point of this Thesis. It is important to note that since in the V-PCC software version 1.2, it is not possible to encode point clouds with a geometry bit precision higher than 10 bits, hence the comparison was performed for point clouds with a bit precision of 10, namely the *People* dataset.

The new plane projection approach includes 3 main tools, which have been added or have replaced some existing V-PCC tools, notably:

- **K-Means Clustering -** To create the initial clusters (set of points) according to the estimated normals thus allowing to group points with similar normals and thus obtaining surfaces of points with a clear orientation.
- **Adaptive Planes Selection and Coordinate System Transformation -** The most suitable plane for the projection of each cluster data is selected and the corresponding coordinate system transformation obtained and applied.
- **Adaptive Map Resolution Control -** A map resolution expansion is applied to address problems related to the adaptive plane coordinates transformation, since the adaptive local coordinate system is naturally not aligned with the global *x,y,z* coordinate system. This may also create more duplicate points due to the used occupancy map precision, typically with *B0=4.*

Besides the main tools, which are directly related to the adaptive plane projection, 2 additional tools were also added, namely:

- **Recolouring Pre and Post-Processing -** As the amount of duplicate points has an impact on the recolouring module, notably different colours may be attributed to different points in the same geometry position, recolouring pre and post-processing stages are included to attribute the same colour to all duplicate points.
- **Decoder Duplicates Removal -** Duplicate points are removed, leaving only one point per position, also guaranteeing that the number of output points is close to the number of input points.

This chapter will first describe the rationale behind the proposed adaptive plane projection solution, followed by its architecture and walkthrough and will finally present in detail each of the main tools, and

each one of the additional tools. In the following, some concepts regarding the point representation errors will be often used and thus, for the sake of clarity, it is important to precisely define them:

- **Duplicate points –** Different points that share the same precise geometry position.
- **Missed points (aka poorly represented points) –** Original points with a distance to the closest point in a reconstructed point cloud superior to 1. This reconstructed point cloud is obtained at the end of the patch segmentation, by converting each patch back to 3D points in global coordinate system, as described in section 3.2.
- **Non-represented points –** Same definition as missed points, however the distance is equal or superior to 1.
- **2D Overlapping points –** Two or more points that are in different 3D positions but after the projection fall into the same depth map pixel; the number of points overlapping in the same map pixel depends, naturally, on the chosen projection plane.
- **3D Overlapping points –** Two or more points that when represented by floating-point values have different geometry positions; however, when its geometry coordinates are rounded, these points become duplicate points. In this case, the overlapping of points depends on the selected local coordinate system.

4.1. Objectives, Benefits and Potential Complications

As mentioned in Chapter 3, V-PCC uses a set of pre-defined planes for the initial points clustering and later a set of three pre-defined planes (intimately related to the respective clustering planes) to project the geometry and texture into 2D depth and texture maps. These pre-defined planes correspond to the faces of a cube and are used for all point clouds, independently of their set of estimated normals and, thus, the shape of the point clouds. If the point cloud set of normals is not well aligned with the global coordinate system *(x,y,z)*, which defines the fixed planes, and as a consequence with the clustering and projection planes, the angular difference between the V-PCC projection planes and the respective clusters set of normals may be large, meaning that the projection planes are not well adapted to the input data. This has two main consequences: **i)** the projected geometry map values will have higher variance, which potentially increases the geometry rate; and **ii)** some points may be occluded during the projection, which has a negative impact both in objective and subjective qualities.

Bearing these issues in mind, the proposed projection approach has the key objectives of improving the RD performance and subjective quality of the decoded point clouds by adopting an adaptive plane projection paradigm, which should provide more ‘friendly’ projection planes for each specific point cloud cluster/sub-cluster. This technique replaces the three V-PCC pre-defined projection planes by varying projection planes adapted to the clusters’ normals (or in alternative to the sub-clusters within each cluster). The new adaptive projection planes are expected to:

- Reduce the depth maps variance as the point cloud main surfaces should be more aligned/parallel to the adaptive projection planes and, therefore, potentially reduce the geometry rate.
- Reduce the number of points overlapping into the same map pixel after the projection (2D overlapping points) as there are less points with an angular difference to the projection plane higher

than 45 degrees, which, in practice, reduces the number of poorly represented original points (or missed points).

Since the adaptive projection planes may no longer be aligned with the global coordinate system (*x,y,z*) but they rather make some angles with these axes, the resulting local/transformed coordinate system values (*δ,s,r*) have to be initially expressed with floating-point precision, before being resampled/rounded to the regular grid used for the depth and texture map projection and later be coded.

Consider the local coordinate system transformation in 2D as illustrated in Figure 41, showing on the left the global coordinate system with the corresponding V-PCC projection plane and, on the right, the selected adaptive plane with the respective local coordinate system transformation. If the same resolution is used for both cases, the geometry distortion can increase for the case of the adaptive plane since more points will overlap in the same 3D bin (3D overlapping points), i.e. two points falling into the same 3D *bin* of the local coordinate system. During the projection, they are both rounded to the centre of the depth map pixel and their rounded depth value is the same. In practice, these two points will be wrongly represented as a single point; one of them is discarded/excluded during patch generation process, i.e. there are points that are not represented by the depth maps.

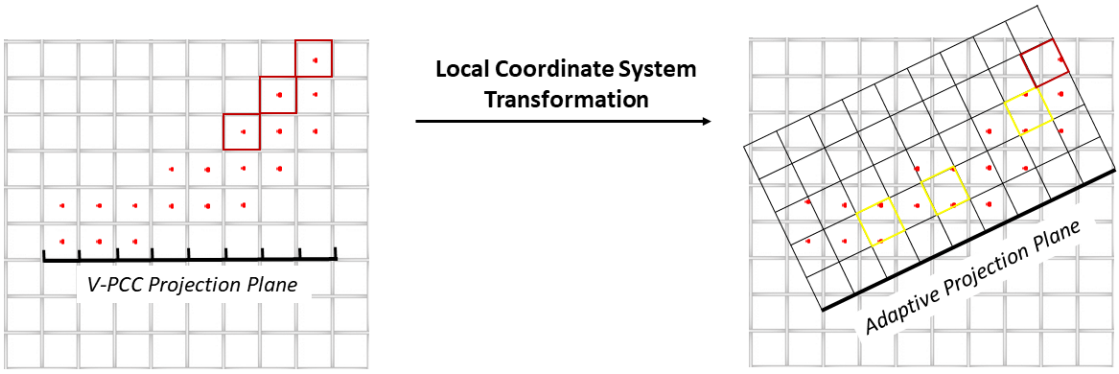


Figure 41- Left) Set of points in global coordinate system. The red-marked 2D bins contain points that will be discarded during the projection, because they are the 3rd point falling into the same pixel after the projection to 1D. Right) Same points, but on local coordinate system. After picking the adaptive projection plane, multiple points would overlap on the same 2D bin (marked as yellow), before the projection to 1D and be counted as the same point.

In the example in Figure 41, the choice of the adaptive projection plane and the local coordinate transformation results in 3 different 2D *bins* with more than one point. If two points fall into the same 2D *bin* (in AP2V-PCC it would be a 3D *bin*), they are only accounted as 1 point, since during the projection both points will be represented as a single pixel with just one depth value. This means that only one of those points will be considered; the other will be excluded. Also, there are points that cannot be represented by the *D0/D1* layer, since they will be the 3rd overlapping point (in AP2V-PCC these would be 2D overlapping points) in the same pixel during the projection. In Figure 41, the total number of discarded points is 3 and 4 for the V-PCC projection and the adaptive projection plane, respectively, mainly because of the overlapping of points into the same bin (3D overlapping points in AP2V-PCC).

Naturally, the problem mentioned above may reduce the impact of the benefits associated to adopting adaptive planes, e.g. reduction on the number of missed points, and thus a solution must be found. A natural way to address this problem is to increase the resolution of the projection maps and this will reduce the number of overlapping/missed points. As a consequence, the size and very likely the

rate of the occupancy map, used for both the geometry and texture, are increased. While using a precision (*B0*) of 4 to code the occupancy map will drastically reduce the rate associated to the occupancy map when compared with *B0* precision equal to 1 or 2, as the spatial resolution increases, more points will be created, notably duplicate points. This increase of duplicate points has an impact on the recolouring process and thus, at the encoder, some pre and post-recolouring processing is performed to ensure that all decoded points with the same geometry are encoded with the same colour. At the decoder, the duplicates are also removed, leaving a single point for each position.

4.2. Architecture and Walkthrough

The proposed Adaptive Plane Projection Patch Generation architecture is illustrated in Figure 42. This architecture is basically the V-PCC Patch Generation architecture, while changing the projection planes to better adapt to the point cloud characteristics and, therefore, achieve a better RD performance and subjective quality. As most modules are inherited from V-PCC, more attention will be dedicated in the following to the new modules to avoid unnecessary redundancy regarding the V-PCC description as these modules are the core contribution of this Thesis.

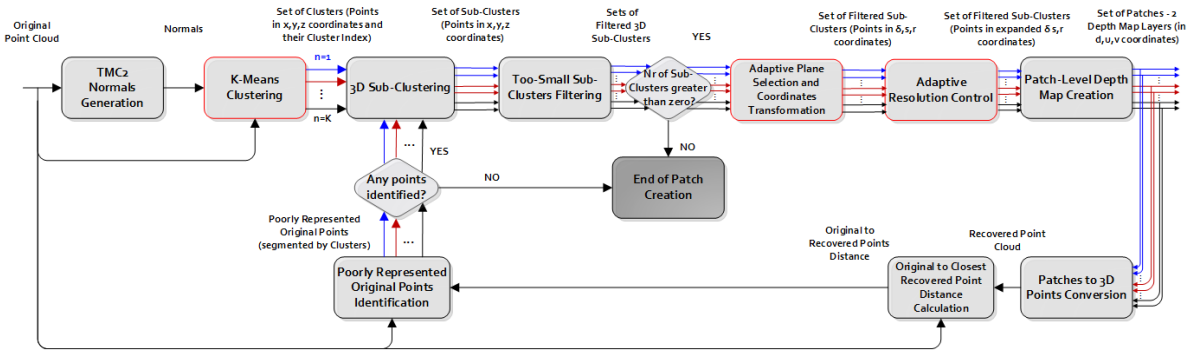


Figure 42 – Adaptive Plane Projection Patch Generation architecture, with the new modules highlighted in red.

The walkthrough for the proposed architecture proceeds as follows:

- **TMC2 Normals Estimation** – The starting point of this solution is the same as for V-PCC, as both need to estimate the normals for each point in the cloud. Since V-PCC already has a module to efficiently perform this task, the normals estimation process is performed exactly as for V-PCC.
- **K-Means Clustering -** In V-PCC, each point is associated to the cluster maximizing the dot product between its normal and the set of pre-defined plane normals and, thus, clustering will result in sets of points (clusters) with a normal centroid (corresponding to the normalized average of the cluster set of normals) closer to the pre-defined plane normal. This is most likely not the best option in terms of adaptation to the point clouds and, thus, to foster the planes adaptability, the first change in the patch generation architecture regards the replacement of the V-PCC normals internal product-based clustering by K-Means clustering, where *K* is a user defined parameter expressing the number of clusters used to adapt to the point cloud set of normals. The result of this process are *K* clusters with a normal centroid, which corresponds to the plane normal, with the assignment of a cluster index (0, …, K-1) for each point in the cloud. If a projection plane is associated to each cluster, the local coordinate transformation is made on a cluster basis.

- **3D Sub-Clustering –** The next step regards the creation of the sub-clusters or patches and it is similar to the V-PCC 3D sub-clustering process with the only change that now *K* may be different from 6; naturally, if more adaptive projection planes are to be used, notably at sub-cluster/patch level, a coordinate system transformation will have to be defined for each patch.
- **Too Small Sub-Clusters Filtering –** As in V-PCC, this module has the purpose to filter the sub-clusters including less points than the threshold *minPointsCountPerCCPatchSegmentation* (e.g. 16).
- **Adaptive Planes Selection and Coordinate System Transformation –** Having the filtered 3D sub-clusters, the adaptive planes have to be selected and the corresponding local coordinate system (*δ,s,r*) transformations defined in order the new coordinates are obtained. There are two main options in terms of adaptive planes selection:
    - **Cluster-based Adaptive Planes –** In this case, the plane adaptation is performed at cluster level implying that an adaptive plane is selected for each cluster. In this case, there is a single coordinate system transformation (*δ,s,r*) for all points in each cluster, meaning that all sub-clusters in a cluster share not only the same cluster index but also the same coordinate system transformation. The cluster normal centroid (determined in the K-Means clustering), which corresponds to the projection plane normal, will be assigned as the depth axis (*Δ*) for all sub-clusters sharing its cluster index. Since all coordinates must be orthogonal to each other, the tangent (*S*) and bi-tangent (*R*) coordinates axis must be orthogonal to each other and collinear with the projection plane.
    - **Sub-Clusters Adaptive Planes –** In this case, the plane adaptation is performed at sub-cluster level, which means that an adaptive plane is computed for each sub-cluster. Here the coordinate system transformation is computed independently for each sub-cluster implying that a finer adaptation is performed in terms of projection planes. Naturally, instead of using the previously selected cluster projection planes, the best adaptive projection plane is computed for each sub-cluster, in this case by calculating the sub-cluster set of normals average and setting its norm to 1. This will lead to a projection plane normal with an average lower angular difference to the normals of the sub-cluster.

The specifics of the two adaptive plane selection approaches will be detailed in the section dedicated to this main tool.

- **Adaptive Map Resolution Control –** The main complication of the global to local coordinate system transformation is that the coordinates are computed with floating-point precision and thus should be resampled into a regular grid, in this case by rounding, to be after projected. This is different from V-PCC where the point coordinates are already expressed with integer values. As previously described, the main consequences of this resampling process are the increase on the number of duplicate points (originated from 3D overlapping points), which after the projection are rounded to the same 2D depth map pixel and have an equal depth value (in practice they only count as one projected point). The overlapping of points into the same 3D bin will, thus, lead to an increase on the number of non-represented points, and thus to an increase of the geometry distortion. These issues may be mitigated by increasing the resolution of the (geometry and thus also texture) projection maps for the three

coordinates. This process corresponds to multiplying each point local system/transformed coordinate system coordinates by an expansion factor corresponding to the resolution increase. This resolution expansion process is performed by multiplying all sub-clusters coordinates by the expansion factor. A study of the RD performance impact of the expansion factor is presented in Chapter 5. Increasing the expansion factor implies increasing the geometry map resolution, and thus potentially the geometry rate. Since the depth and texture maps share the same occupancy map, increasing the resolution for the geometry also implies increasing the texture map resolution.

- **Patch-Level Depth Map Creation -** The patch depth maps are created as in V-PCC. The depth map values to code should have a lower variance due to the more adapted projection planes. However, to address the problem of many points overlapping into the same 3D *bin*, not only the map resolution has increased, but also the geometry range of values, which increases the variance. Thus, a trade-off between these effects must be found.
- **Patches to 3D Points Conversion –** At this step, the generated patches must be converted back to 3D points in the global coordinate system to be able to evaluate if the original points are well represented by the already obtained patches. It is important to stress that the assessment of the geometry quality must be performed in the global coordinate system, since this is the original coordinate system (integer geometry values).
- **Original to Recovered Points Distance Calculation –** This module proceeds in the same way as V-PCC: for each original point, the closest point in the recovered point cloud is searched in order to measure the distance between the two and, thus, the current representation error for each original point.
- **Poorly Represented Points Identification –** This module was also kept unchanged regarding V-PCC and identifies the poorly represented original points: if the distance from an original point to the closest recovered point is larger than a *dist2MissedPointsDetection* (typically 1), the point is considered as poorly represented and is given back to the 3D sub-clustering module for further consideration. At this stage, additional 3D sub-clusters only with poorly represented points are created with their global coordinates and again transformed to a local coordinate system corresponding to the relevant adaptive plane. Next, the additional projected patches are created, and the full patch set converted back again to 3D point clouds until no missed points exist. Finally, the set of created patches is given to the packing module, ending the patch generation module before proceeding with video-like coding.

The new tools require some additional metadata to be sent on the bitstream to reconstruct the point cloud at the decoder side (with some quality loss). This additional information includes the projection planes definition and the adopted expansion factors. Therefore, changing the projection planes to be more flexible and adaptive also incurs on an overhead cost. The performance assessment in Chapter 5 will finally show if the benefits compensate the added costs.

4.3. Main Tools Description

To make the adaptive plane projection approach possible, several new tools were introduced in the V-PCC framework. In this section, the algorithms behind each new tool will be detailed, and the design choices motivated.

4.3.1. K-Means Clustering

To select projection planes as adapted as possible to the respective cluster set of points (with normals), the initial clustering must be performed *differently* from V-PCC where the projection planes are pre-defined and correspond to a sub-set of the clustering planes. Thus, to avoid selecting clusters which are biased by the V-PCC six pre-defined planes, the V-PCC clustering module is replaced by a K-Means based clustering process, which segments the cloud points into a user-specified number of clusters (*K*) considering the normals previously obtained (as for V-PCC). Each resulting cluster is characterized by its number of points, and its normal centroid, which corresponds to the average of the normals associated to the cluster points.

The adopted K-means clustering algorithm implementation was the Rosetta Code [67], which has the objective of minimizing the sum of distances of each point to its cluster centroid. However, considering our target, the clustering follows a different criterion notably maximizing the average internal product of the cluster normals with their cluster normal centroid, the so-called *point cloud average dot product (PCADP)*:

$$PCADP = \frac{1}{\mathrm{N}} \cdot \sum_{p \in P} \vec{n}_p \cdot \vec{n}_{k(p)}, \qquad (14)$$

where *N* is the number of points of a (static) point cloud, *P* is the complete set of points (*p*), n_p is the normal for point *p* and $n_{k(p)}$ is the normal centroid for the cluster of point *p*. *PCADP* measures how suitable a set of planes is for the projection of the corresponding sets of points, since it measures how close to zero is, on average, the angular difference between the cluster normal centroid and each one of the points normals.

Even though the K-means clustering Rosetta Code implementation [67] has been used, some adaptations were required, notably to change the main objective of minimizing the distances to the cluster spatial centroid to maximizing the dot product between each point normal and the respective cluster normal centroid. In summary, the proposed clustering algorithm proceeds as follows:

1. Clusters normal centroids initialization - The first step is to initialize the clusters normal centroids. This is achieved by picking as many random points as desired clusters (*K*) and assigning the normal of each selected point as the initial cluster normal centroid.

2. Points-to-clusters assignment – Next, each point is assigned to the cluster with the 'closest' normal centroid, meaning that the dot product between the point normal and the cluster normal centroid is the largest. The dot product between two vectors (normals in this case) is the same as multiplying the Euclidian magnitudes (or norms) by the cosine of their angular difference, i.e. $A \cdot B = \|A\| \, \|B\| \cos\theta$, where *A* and *B* are two different vectors and *θ* is the angular amplitude between them. Since all (normalized) normals have a magnitude of 1, the dot product will result in the cosine of the normals angular difference. As the cosine is a decreasing trigonometric function, when the argument (angular difference) is within the interval from 0 to 180 degrees, the closer is the angular difference to zero, the greater the dot product is, reaching 1 when the normals are co-linear.

3. Clusters normal centroids update - After all points are allocated to one of the *K* clusters, the cluster centroid is updated to the average of its associated points normals and normalized. If the normals are not normalized, one may incur in wrong assignment of points (and indirectly normals) to clusters,

since the result of the dot product will not only express how close to zero is the angular difference (as desired) but will also depend on the magnitude of the normals (which is not desired).

4. Clustering refinement and stabilization - Steps 2 and 3 are repeated until the clustering stabilizes. The stabilization is considered to be achieved when 99.9% of the points stop being assigned to a cluster different from the current assignment. The number of points that have changed their cluster index is computed at the end of each iteration.

At the end of this clustering process, the desired *K* clusters are defined, in this case clusters which are adapted to the normals associated to each point. Since, as for V-PCC, this proposed clustering process still suffers from a number of isolated points, this means with a cluster index in the middle of a 3D region with most of the points associated to another cluster index, the same V-PCC final clustering refinement still must take place.

The results, to be analysed in Chapter 5, show that PCADP increases, when the V-PCC clustering is replaced by K-Means clustering implying that the proposed clustering provides a better alignment between the final clusters and the projection planes in terms of normals. Two examples of V-PCC and K-Means clustering are shown in Figure 43 for the same number of clusters (*K=6*). It is possible to see that the original frame 1300 of the *Longdress* sequence is more aligned with the *xyz* axis than the original frame 1550 of *Redandblack*, i.e. the bigger surfaces are more parallel to the *XoY* and *YoZ* planes. Consequently, the majority of the estimated normals for the frame 1300 of the *Longdress* sequence are already aligned with the normals of the V-PCC fixed planes. Thus, the resulting set of K-Means clusters is more different from V-PCC for the frame 1550 of *Redandblack*. Naturally, the PCADP metric also increases more for frame 1550 of *Redandlack* sequence, when V-PCC clustering is switched to K-Means Clustering, than for frame 1300 of *Longdress* sequence, notably from 0.805124 to 0.83011 (difference of 0.025) in opposition from 0.81575 to 0.834581 (difference 0.018), respectively.

Figure 43 – Left) Frame 1300 of Longdress sequence. From left to right: Original frame, V-PCC clustering and K-Means clustering results (K=6) [4]. Right) Frame 1550 of Redandblack sequence. From left to right: Original frame, V-PCC clustering and K-Means clustering results (K=6) [4].

From this point on, each cluster will be processed independently, until the patches are converted back to 3D points.

4.3.2. Adaptive Plane Selection and Local Coordinate System Transformation

The objective of this module is to select the adaptive projection planes to be used by each cluster or eventually sub-cluster, thus defining the local coordinate system (*δ,s,r*), and perform the corresponding local coordinate system transformation.

A. Adaptive Plane Selection

There are two alternative approaches to select the planes: *cluster-based adaptive planes* or *sub-cluster adaptive planes*. The two approaches will be described next.

A.1. Cluster-based Adaptive Planes Selection

In this approach, a projection plane is computed for each cluster using the output of the K-Means Clustering module, i.e. from data obtained at cluster level. Therefore, all sub-clusters in the same cluster index will share the same projection plane and local coordinate system (*δ,s,r*). In this case, the rate overhead is rather small since usually the number of clusters is much lower than the number of sub-clusters.

A 3D coordinate system is defined by a triplet of lines orthogonal to one another (the axes), going through a common point (the origin). In turn, the location of each point *p* in a 3D coordinate system is characterised by three values (coordinate values), one for each axis. Each coordinate value corresponds to the distance of *p* to the plane defined by the other two axes. Thus, the global coordinate values are written as three numbers *(x,y,z),* corresponding to the axes (*X,Y,Z*), where the unit vectors of each one of the axes are (1;0;0), (0;1;0) and (0;0;1), respectively. Each of the local coordinate system unit vectors corresponding to the *Δ*, *S* and *R* axes is a linear combination of the global coordinate system unit vectors. To define a local coordinate system, it is only necessary to define each one of the axes unit vectors. In the context of this Thesis, after the adaptive projection plane is computed, the local coordinate system axes unit vectors are assigned as follows:

1. Depth axis (*Δ*) assignment – The cluster normal centroid (or projection plane normal vector) is assigned as the depth axis. The motivation for this choice is that the depth should be perpendicular to the surface, as in the amount of 2D overlapping points will be lower. Moreover, it is desirable to obtain a depth with the lowest variance as this contributes to perform more efficient depth coding when using a standard 2D video codec after the sub-clusters are projected to 2D geometry and texture maps containing the point cloud patches.

2. Tangent axis (*S*) assignment – To obtain the second unit vector defining the local coordinate system, a solution inspired from [68] is proposed, since it is rather simple and targets aligning the local coordinate system with the global coordinate system. The objective of this approach is to have a similar orientation of the sub-clusters when they are projected into the projection plane when compared to their orientation in the usual point cloud rendering which leads to an efficient packing of patches but also better compression with standard 2D video codecs (as the DCT transform basis functions are aligned with the patch direction). An example is shown in Figure 44, where two different sub-clusters (two middle frames) are packed (right-most frame) with the same orientation as they are rendered (left-most frame). Since, after the projection, the tangent axis (*S*) will become the horizontal axis (*u*) of the 2D packed

image, it is desirable that this axis is aligned as much as possible with the horizontal dimension of the point clouds. As the Y-axis of the global coordinate system typically corresponds to the vertical axis of the point cloud (see left-most frame in Figure 44), the tangent axis is aligned with the point cloud horizontal plane (perpendicular to the vertical axis) by computing the external product between the depth axis (Δ) unit vector and the Y-axis unit vector (0;1;0) or its symmetric (0;-1;0). Both unit vectors were tested and the unit vector (0;-1;0) achieved superior geometry and colour RD results for all test material, and thus the S axis is assigned as $S = \Delta \times (0;-1;0)$.This will ensure that the tangent axis will be perpendicular to the depth and to the Y-axis of the global coordinate system, and thus collinear to the typical horizontal plane of the point cloud (XoZ).

3. **Bi-tangent axis (R) assignment –** Since the Y-axis typically corresponds to the vertical axis of the point clouds, and the bi-tangent (R) axis will correspond to the vertical axis (v) of the 2D packed images, the R axis should be as aligned as possible with the global coordinate system Y-axis. As the S axis is collinear with the horizontal plane, assigning the R axis as the external product between the depth and the tangent axis, i.e. $R = \Delta \times S$, aligns it as much as possible with the Y-axis by being orthogonal to an axis collinear to the horizontal plane.

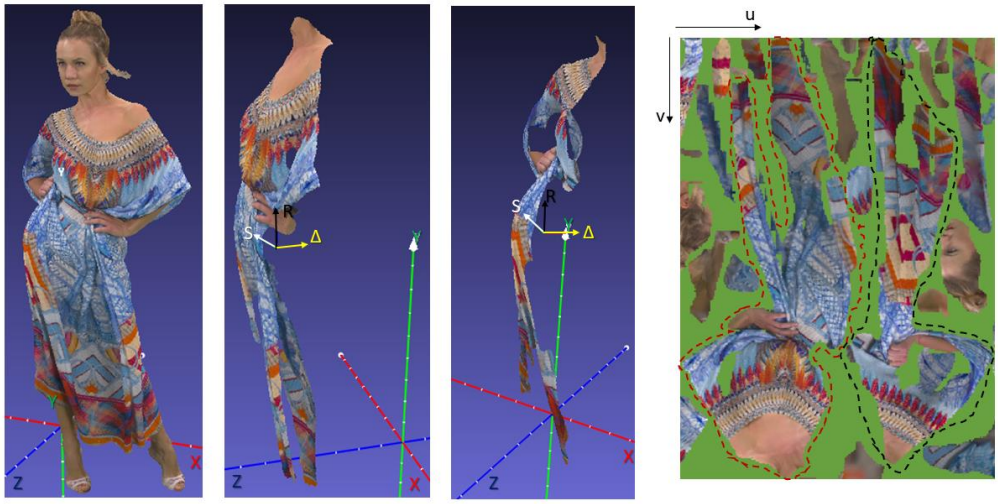


Figure 44 – From left to right: Original frame 1550 of Longdress [4]; two sub-clusters formed with K=6, where the local coordinate system is represented with the yellow (depth), white (tangent) and black (bi-tangent) axis and the global coordinate system is represented as red (X), green(Y) and blue (X) axes; partial packing result where the two sub-clusters projections to a 2D texture image(highlighted in red and black, respectively) are aligned as the rendered point cloud, which is presented on the left.

In Figure 44, a partial packing result is shown for two projected sub-clusters. Note that the bi-tangent (R) axis vector points upwards for both sub-clusters, but as the 2D vertical axis (v) direction points downwards, the orientation of all sub-clusters is upside down. After generating the texture maps, the green spaces appearing in the partial packing result will be padded (horizontally); thus, by having both patches (vertically) aligned, there is more spatial correlation, which can be exploited by a 2D standard video codec to increase the compression efficiency.

A.2. Sub-Cluster Adaptive Planes Selection

In this approach, a projection plane is computed for each sub-cluster, thus there are many different projection planes within a single cluster. As a consequence, the local coordinate system transformation is also applied independently for each sub-cluster. The objective of performing these tasks on a sub-

cluster-basis is to obtain an even better adaptation to the point cloud data characteristics. For this case, two different techniques have been tested, notably:

- **Principal Components Analysis (PCA) –** This technique has already been used for the points normal estimation and, thus, it is applied in the same way, but now for the entire set of sub-cluster points, instead of the *N* nearest neighbour of a point (for normal estimation). The result will be three pairs of eigenvectors, the so-called *principal components,* and *eigenvalues*, where the eigenvalues represent the variance resulting from projecting the points onto each of the principal components axis. The projection of points onto the first principal component axis should provide the largest variance among all possible axis, and the projection of data onto each next principal component axis should still provide the largest variance possible, given that each component is orthogonal to the preceding ones. Thus, having the PCA result, the depth (*Δ*), tangent (*S*) and bi-tangent (*R*) axes are assigned to the principal component axis with the lowest to largest variance (expressed by the eigenvalues), respectively. The depth is attributed to the eigenvector with lowest variance since one of the adaptive plane objectives is to reduce the depth variance. Moreover, typically the projection plane associated to the lowest variance component is also very well adapted to the set of normals; in the case of planar surfaces, the lowest variance vector is usually the one orthogonal to the surface. The bi-tangent axis (*R*) will be later projected to 2D as the vertical axis and, since the 2D padding process is predominantly done horizontally, it is easier to explore the correlation if the longest side of the projected sub-cluster is oriented vertically, as it is happening in Figure 44 for the two sub-clusters. Thus, the *R* axis is assigned the eigenvector axis with the greatest variance, and consequently the S axis is assigned the remaining eigenvector axis.
- **Average Sub-Cluster Normal Selection** - Instead of selecting the projection planes obtained from K-Means Clustering, the average of the sub-cluster set of normals is computed and selected as the depth axis (*Δ*). As previously mentioned, the K-Means Clustering is based on the maximization of the dot product between each point normal and its cluster normal centroid, the so-called *point cloud average dot product* (PCADP). The projection plane computation at sub-cluster level follows the same objective: the goal is to maximize the average of the dot product (lower angular values between vectors) between the sub-cluster normals and the respective sub-cluster projection plane normal. The tangent and bi-tangent axis are determined in the same way as for the cluster-based approach, naturally using the sub-cluster projection plane and the already assigned depth axis for the respective sub-cluster.

Note that these two solutions select the projection planes with different criteria: the PCA sub-cluster-based approach minimizes the geometry values variance, while the average sub-cluster normal selection maximizes the sub-cluster-based PCADP. Naturally, they are related since the estimated normals on the second solution are also computed with PCA. As an example, Figure 45 shows the local coordinate systems obtained with the PCA and average of the sub-cluster set of normals solutions; in both cases, K-Means Clustering was performed with *K=6*.

Since it is not clear which criterion is the best from the compression point of view, an evaluation was performed to understand the impact of each solution in terms of geometry RD performance.

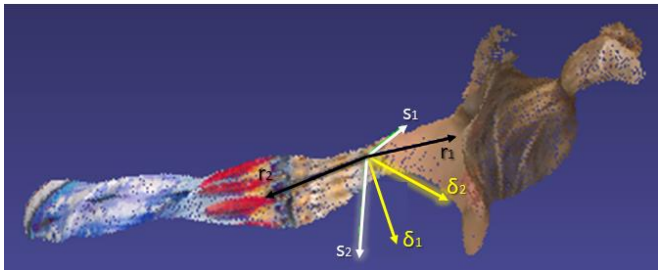


Figure 45 – PCA computed local coordinate axis identified by index 1, and average of the sub-cluster normal local coordinate system identified with index 2, for a sub-cluster of frame 1300 of Longdress sequence [4]. selection (in orange).

PCA vs Average Sub-Cluster Normal Selection

As shown by the charts in Figure 46, the geometry RD results are rather similar but with a slight gain for the average sub-cluster normal selection solution. An expansion factor adaptive resolution control parameter (see Section 4.3.3) of 1.6 and *K=12* have been used, as these parameter values are within the best performance range for both approaches. The BD-PSNR, which provides the average value for how much the PSNR has increased or decreased for the same rate, also shows that the PCA technique is outperformed (although not much) by the average normal selection technique. The BD-PSNR for the average normal selection solution taking the PCA solution as reference for frame 1550 of *Redandblack* sequence is 0.088 dB, while for frame 1300 of Longdress sequence is 0.023 dB. If a lower K was used, the BD-PSNR values would have a larger difference, since the sub-clusters typically become smaller with the increase of *K*. Smaller sub-clusters tend to be more planar, which is the situation where the PCA lowest variance eigenvector has the largest associated PCADP. This effect does not occur for the average normal selection, since every normal is computed from a neighbouring set of points and then all these normals are averaged.

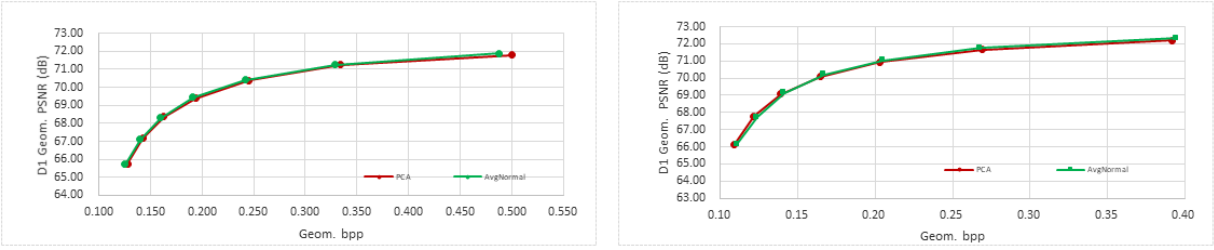


Figure 46 – D1 metric PSNR RD chart versus the corresponding geometry bpp (including the occupancy and patch info) for frames 1550 and 1300 of Redandblack (a) and Longdress (b) sequences, respectively. The expansion factor is 1.6, while coding on sub-cluster-based mode with PCA (in red) and the average sub-cluster normal selection (in green).

Cluster vs Sub-Cluster-based Plane Selection

After describing the two alternative adaptive plane projection approaches, the geometry D1 PSNR RD performance for sub-cluster and cluster-based plane selection is illustrated in Figure 47 for frame 1550 of *Redandblack* and frame 1300 of *Longdress*, while using *K=12* and *EF=1.6*, as these parameter values are within the best performance range, see Chapter 5. The cluster-based approach is outperformed for both point clouds. In the cluster-based plane selection solution, 12 planes are sent, and each sub-cluster has to signal to which of the clusters it belongs. This allows to achieve a lower rate

overhead compared to the sub-cluster plane selection solution, which has to transmit specific plane information for each of the sub-clusters. However, having more adapted planes should also bring improvements in terms of the depth maps variance with a corresponding geometry rate reduction after encoding with a standard 2D video codec. The selected solution is thus the sub-cluster projection plane selection.

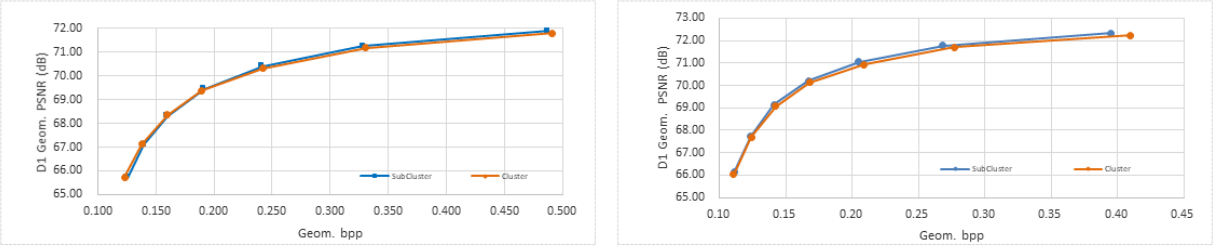


Figure 47 - Geometry D1 RD performance of frame 1550 and 1300 of Redandblack (a) and Longdress (b) sequences, respectively, for AP2V-PCC with Sub-Cluster (in blue) and Cluster-based (orange) plane selection modes.

B. Local Coordinate System Transformation

Independently of the approach used to obtain the local coordinate system, since (δ,s,r) is a linear combination of the global coordinate system (*x,y,z*), a *3×3* transformation matrix is computed, on a cluster or sub-cluster basis, determining how to obtain (*δ,s,r*) coordinates from (*x,y,z*) coordinates. Each row of the matrix corresponds to the transformation (linear combination) of (*x,y,z*) as follows:

$$\begin{bmatrix} \delta \\ s \\ r \end{bmatrix} = \begin{bmatrix} t_{\delta x} & t_{\delta y} & t_{\delta z} \\ t_{sx} & t_{sy} & t_{sz} \\ t_{rx} & t_{ry} & t_{rz} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (15)$$

where each coefficient t_{ij} is the transformation coefficient from the global coordinate *j* to the local coordinate *i*. The three coefficients used to transform the global coordinates (*x,y,z*) to a local coordinate *i*, i.e. each one of the rows (t_{ix}, t_{iy}, t_{iz}), also known as *transformation vectors*, are the unit vectors of the respective local coordinate axis.

Since all the transformation vectors (rows) are normalized, the transformation coefficients can only have values within the interval of -1 to 1. Note that it is not necessary to send all nine coefficients, since only the first transformation vector must be selected and the other two are computed from the first. Thus, the transmission of more than the first transformation vector would incur in an unnecessary rate overhead. Each coefficient is encoded with 8 bits by first multiplying each coefficient by 127, rounding and then adding an offset of 127, which produces values within the discrete interval of 0 to 254. The coefficients could be coded with a different precision, e.g. 16 bits, but the rate overhead caused by more precision does not bring any improvement from the RD sense, and 8 bits was found to be a good choice. At the decoder side, the decoded value is subtracted by the offset 127 and divided by this same value.

To transmit the necessary information for the decoder to compute the planes that are now adapted to the content, it is necessary to send three coefficients of 8 bits (total of 24 bits) for each cluster if the cluster-based approach is adopted. Since each sub-cluster corresponds to a patch, it is also necessary to indicate for each patch to which of the *K* clusters it belongs, as auxiliary information, so that it is possible to unambiguously associate a projection plane to each patch. On the other hand, for the sub-

cluster-based approach, three coefficients (24 bits) are sent for each sub-cluster, thus defining one plane for each sub-cluster.

At the decoder side, the same exact matrix used to transform the global coordinates to the local coordinate system is obtained from the information received. Note that any mismatch between the encoder and decoder transformation matrices will lead to a performance penalty. If the resolution of the local coordinate system was not expanded (see next section), it would be possible to obtain the coded points in the original (global) coordinate system by building the inverse transformation matrix and applying it to the points in the local coordinate system as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} t_{\delta x} & t_{\delta y} & t_{\delta z} \\ t_{sx} & t_{sy} & t_{sz} \\ t_{rx} & t_{ry} & t_{rz} \end{bmatrix}^{-1} \begin{bmatrix} \delta \\ s \\ r \end{bmatrix} \tag{16}$$

After applying the inverse transformation, the resulting (x,y,z) coordinate values may be expressed initially at the decoder with floating-point precision. Since the original point cloud geometry is represented with integer values, these decoded values are rounded to integers.

4.3.3. Adaptive Resolution Control

The adaptive projection plane and the definition of a local coordinate system are performed with the objective of lowering the variance of the depth values to be coded in the geometry maps but also to reduce the number of points that cannot be represented by the *D0* and *D1* depth map layers, the so-called *discarded points*. However, the global to local coordinate system transformation has some negative consequences since the local coordinates are no longer aligned with the map grid as the global coordinates; notably, the local coordinate system values have to be initially expressed with floating-point precision and later resampled (rounded) into the regular grid, to be coded by a standard video codec. This rounding process will increase the geometry distortion and may cause overlapping of points into the same 3D *bin* (causing 3D overlapping points) of the local coordinate system, as previously illustrated in Figure 41. The solution to this negative effect is to increase the resolution of the local coordinates.

By increasing the resolution, the geometry distortion caused by the rounding process will be reduced, as the 3D *bins* will be now smaller leading to a lower number of 3D overlapping points. For the same reason, with the increase of the local coordinates resolution (or the decrease of the 3D *bins* size), the points will be better represented, notably with a lower number of non-represented points. The resolution increase is controlled by a user-defined parameter, the so-called *expansion factor*. The expansion factor is equal for all sub-clusters and, thus, each point local coordinate value is multiplied by this parameter after the respective local coordinate system transformation.

To show the effect of increasing the resolution, a 2D example is provided in Figure 48, which takes the local coordinates transformation result previously shown in Figure 41 and applies the resolution expansion technique with an expansion factor of 2.

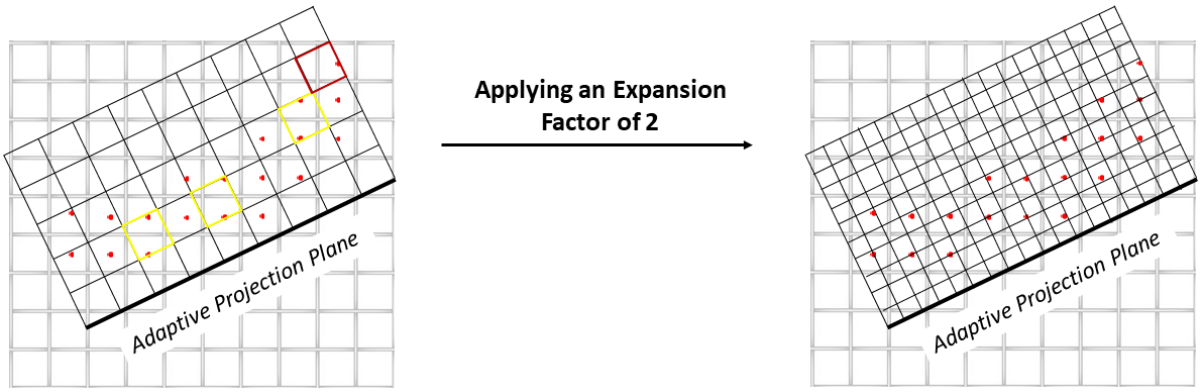


Figure 48- 2D example of resolution control after the local coordinate system transformation with an expansion factor of 2. The yellow-marked bins contain 2 points, and the red-marked bins have points that will not be projected since there is a 3rd overlapping point in the same depth map pixel. No yellow and red marked bins exist after applying the resolution expansion.

In Figure 48, the number of 2D *bins* (in AP2V-PCC it would be a 3D *bin*) containing 2 points after the resolution expansion became zero, and no point will be discarded. Points can be discarded after the projection when more than 2 points fall into the same pixel of the respective depth map; when 2 points have their depth difference lower than the surface thickness threshold, the point with the lowest depth value will be represented in the *D0* layer, while the other point will be represented in the same pixel but in the *D1* layer. Note that, with the V-PCC projection planes, the points would still overlap in the same pixel after the projection even if an increase of resolution was applied.

Naturally, the main drawback of the local coordinates resolution increase is that the sub-clusters will need more pixels to be represented and thus larger resolution will be used for the geometry and texture maps (after packing), as well as for the occupancy map. The geometry, texture and occupancy maps resolutions must be the same to guarantee a correct re-projection (creation of the decoded point cloud) and otherwise different occupancy maps would have to be used for the geometry and texture. With this resolution increase, the quality will increase but also the bitrate and thus a trade-off must be found in terms of expansion factor.

As the expansion factor is the same for all sub-clusters, only one expansion factor needs to be encoded for the entire point cloud. The expansion factor is encoded using an 8 bits binary representation. At the decoder side, this expansion factor is retrieved from the bitstream and all recovered transformation matrices are multiplied by this value before being transform inverted. Naturally, the final step to get back the global coordinate system values is to invert each sub-cluster transformation matrix (multiplied by the expansion factor) and apply it to the respective points.

4.4. Additional Tools

Besides the just presented main tools, which are part in the Adaptive Plane Projection Patch Generation architecture, additional tools were implemented, with the objective of adapting the recolouring to this new paradigm, and to eliminate the duplicate points at the decoder. These tools are the so-called *recolouring pre- and post-processing* and the *duplicates removal*. While the former is an extension of V-PCC recolouring module (inside the Texture Image Generation module of V-PCC architecture - Figure 35), the latter is included as the final step of the decoder.

4.4.1. Recolouring Pre and Post-Processing

After the depth map of each patch is obtained, the packing as well as the padding procedure take place and the geometry maps are produced; these maps are encoded with HEVC and reconstructed at the encoder. From the decoded geometry and occupancy maps, the decoded point cloud geometry is reconstructed and using also the decoded texture maps and pixel-to-point correspondences (created with the decoded geometry and occupancy maps), the decoded colours are assigned to the respective point. Therefore, the decoder will receive and decode colour data for the decoded points (degraded geometry) and not for the original points, which are never available at the decoder. Since the occupancy map precision is set to 4 (*B0=4*) as this drastically lowers the rate when compared to a map precision of 1 or 2, and the adaptive grid resolution is increased, the number of reconstructed duplicate points increases significantly. The number of duplicate points has an impact on the recolouring process since this module considers the closest point relationships from *A* (original) to *B* (reconstructed) point cloud, and vice-versa, to allocate the colour data, notably:

- ***B*-to-*A* closest relationship:** The closest *A* point to each *B* point is searched, and consequently, each *B* point is associated to the closest *A* point. Thus, all points in the same position in *B* are associated to the same closest point in *A*.
- ***A*-to-*B* closest relationship:** The closest *B* point to each *A* point is searched. Since there are multiple points in the same *B* position (duplicates), there are multiple *B* points that can be selected as the closest point to *A*, but only one of them is randomly chosen.

After, defining these relationships, a list is created for each *B* point. This list contains the set of *A* points that share the respective *B* point as their nearest neighbour (*A*-to-*B* closest relationship). If for a certain *B* point, the list is empty, the *B*-to-*A* closest relationship defines the colour attribution, i.e. the colour of the closest *A* point is attributed to the *B* point. If the list is not empty, the colour of the *B* point is computed as the average of the *A* points colours that are in the respective list. Since that to define *A*-to-*B* relationship, only one *B* point can be selected as the closest point to *A*, and multiple *B* points in the same position may exist, it is very likely that the list created for each *B* point within the same position will be different. Because of this recolouring procedure, it is possible to have multiple points in the same position of the reconstructed point cloud with different colours, and thus the recolouring process should not take as input a reconstructed point cloud with duplicates (referred as R). This is achieved by including a pre- and post-recolouring stages at the encoder, as illustrated in Figure 49, where the recolouring modules are highlighted in red.

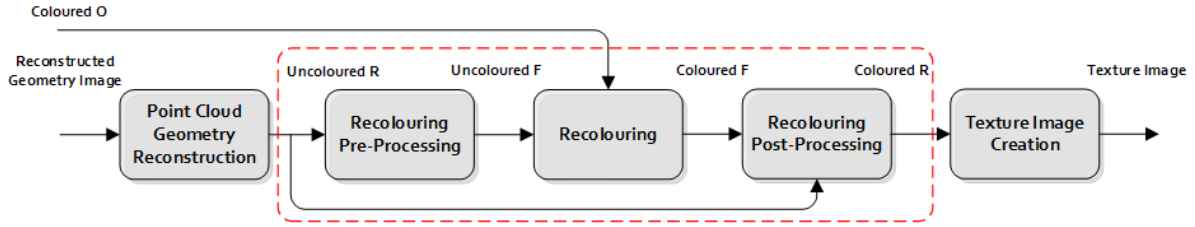


Figure 49 – Recolouring pre- and post-processing stages, where O stands for original point cloud, R stands for reconstructed point cloud and F stands for filtered point cloud (obtained from R).

The modules are now described:

- **Point Cloud Geometry Reconstruction –** The point cloud geometry of each sub-cluster is firstly obtained from the reconstructed geometry image at the local coordinate system (each sub-cluster is in its own local coordinate system). However, *R* must be in the global coordinate system for the recolouring procedure, and thus the respective inverse transformation matrices are applied to all reconstructed sub-clusters. Consequently, the geometry will be represented by floating-point values, which are not rounded before the recolouring, since applying the recolouring procedure to the reconstructed point cloud with its geometry represented by floating-point values is more efficient.
- **Recolouring Pre-Processing –** A pre-processing step is proposed, where a filtered version of the reconstructed point cloud (referred as *F*) is created from *R* by merging the duplicates into a single point.
- **Recolouring -** F is recoloured from the original point cloud (referred as *O*) with the procedure described above, i.e. the recolouring process inherited from V-PCC.
- **Recolouring Post-Processing -** After the recolouring, *R* takes the colour from the point cloud *F* now with colour. This means all points in the same position in *R* (duplicates) get the same colour from the corresponding position in *F*. This way, all duplicates have the same colour, while for V-PCC, different points within the same position may have different colours.
- **Texture Image Creation –** Finally, the texture image is created, as in V-PCC. It is important to note that each point in *R* was created from a pixel in the geometry and occupancy maps, and thus the same pixel in the texture map is set with the respective point colour.

Since the duplicates are encoded through the occupancy map, they have an associated pixel in the texture map and will be decoded with the respective texture of this pixel. If only the texture of the points contemplated in the point cloud *F* (with no duplicates) were set in the respective texture map pixels, the remaining points (duplicates), which are contemplated in point cloud *R* (and not in *F*), would have its texture map pixel value set by the padding process. This would not ensure that all pixels associated to points in the same 3D position have the same colour, and thus the necessity of the post-processing.

It is important to stress that the duplicates are in close (but different) 2D pixel positions in the geometry, texture and occupancy maps, and by setting the colour equal to all of them, the colour correlation between these pixels in the texture maps increases which should be positive from the compression efficiency point of view.

4.4.2. Duplicates Removal

Since the occupancy map has been coded with a precision (*B0*) of 4 and the resolution has increased, more reconstructed points are created than in V-PCC. Since the increase of points occurs mostly due to the resolution increase, most of these points are duplicates. Notably, it may happen that, for a large expansion factor, the number of decoded points becomes 5 or 6 times larger than the number of original points. From the decoding point of view, this is rather ineffective since although many points may exist in the same position, a renderer can only represent one.

With the recolouring pre and post-processing modules described above, the recolouring procedure has set to the same colour all the points in the same position (still represented with floating-point values). Thus, at the decoder, the same uncoloured R point cloud (representing the geometry with floating-point values) is obtained from the decoded geometry image, and each point in R gets the colour from the

respective pixel in the texture map. However, at the decoder, the geometry of these points must be represented with integer values, since the original point cloud geometry is represented this way. For this reason, the geometry of these points is rounded originating again the effect where different points get the same position with different colours (becoming duplicates). The average colour of the points falling in the same position is computed, and then all duplicate points are merged into a single point, with the respective average colour. By merging the duplicates and setting the colour of the position to the average as a final decoding step, the number of decoded points gets very close to the number of original points, as opposed to V-PCC, where the duplicate points are kept in the decoded point cloud.

Chapter 5

5. AP2V-PCC Performance Assessment

The goal of this chapter is to assess the performance of the novel point cloud coding solution proposed in Chapter 4, the so-called *Adaptive Plane Projection* for V-PCC (AP2V-PCC). This chapter starts by presenting the adopted test material, test conditions and objective quality metrics. Then, appropriate values for some key parameters are identified based on experiments, notably the number of clusters (*K*) for the K-Means Clustering and the expansion factor to increase the maps resolution. Using the appropriate parameter values, the RD performance assessment for both colour and geometry data is presented and discussed. Finally, a brief informal subjective assessment is presented. It is important to note that as the sub-cluster adaptive projection solution outperformed the cluster-based, the former will be used in this chapter.

5.1. Test Material

This section describes the used test material, namely, the main properties of each selected point cloud. The test material was obtained from the MPEG repository, which was created for the MPEG Point Cloud Compression Call for Proposals [6]. As previously explained, the proposed AP2V-PCC solution was developed for the coding of static content since none of the techniques proposed in the context of this Thesis targets a better exploitation of temporal correlation. Thus, static point clouds from the *People* dataset were selected as test material, namely: **i)** *Loot;* **ii)** *Redandblack;* **iii)** *Soldier; and* ***iv)*** *Longdress*, which are shown in Figure 50. These point clouds have 10-bit precision and can be encoded with the V-PCC codec; other point clouds from the static dataset (so-called category 1) have higher bit precision (more than 10 bits) and cannot be currently encoded with V-PCC, although in the future this situation may change. The properties of each selected point cloud are now detailed:

1. ***Loot*** –This point cloud has 805,285 points and represents a man standing up, pretending to be playing the piano. The texture does not have many high frequency components, only bluish tones and many flat textured areas with a few edges. In terms of geometry, the point cloud is rather simple to encode, excluding the hands.

2. ***Redandblack*** – Even though this point cloud has less points than *Loot*, as it contains 757,691 points, it is considerably more complex to encode, as the lady's hair has a lot of geometry detail. In this point cloud, there are body parts with rather different depths, e.g. between the legs and the dress. Moreover, the dress (which is red and black) has a pattern which is not very easy to code in terms of texture, mainly because it has many high frequency components.

3. ***Soldier*** – This is probably the most complex point cloud to encode in terms of geometry with 1,089,091 points; it represents a soldier holding a weapon, thus showing significant changes in terms

of depth, and has a rather large amount of geometry detail. On the other hand, the texture is not as complex, since the pattern of the soldier clothes are rather simple, with greyish tones.

4. ***Longdress*** – This point cloud, which is composed by 857,966 points, represents a woman making a pose. In terms of texture, this point cloud is rather hard to encode (many textured edges after the 2D projection) since the dress has a rather complex texture pattern. The hair and the bottom part of the dress also have a high amount of geometry detail.

Figure 50 – Selected test point clouds: from left to right Loot; Redandblack; Soldier; Longdress [4].

5.2. Test Conditions

This section presents the test conditions for the RD performance assessment, which are also used for the final informal subjective assessment. Using appropriate quantization parameter values for both geometry and texture maps, it is possible to control the trade-off between bitrate and decoded quality in the MPEG V-PCC framework. Note that the geometry map is a composition of depth maps, while the texture map is closer to a typical 2D image/video, notably after padding. Naturally, geometry quantization and texture quantization have very different objective and subjective impacts, and thus, there are different sets of quantization values for geometry and texture, both user-defined parameters. These and other parameters used for the RD performance assessment are defined by the MPEG Common Test Conditions document [69]; this is important in order results for different coding solutions may be immediately compared.

MPEG Common Test Conditions

The MPEG common test conditions (CTC) are typically used to offer a well-defined environment to evaluate enhancements or alternative solutions to MPEG V-PCC along its design and specification process. There are three sets of test conditions, one for each content category, notably static, dynamic and dynamic acquisition content. For V-PCC, the configuration parameters, namely common configurations, quantization points, etc. are set in configuration files. Since the objective of this Thesis

is to code for lossy geometry and lossy colour, the configuration for this case was used as specified in [69]. As the test material is static, the chosen coding mode was All-Intra.

Naturally, the selected coding benchmarks are V-PCC and G-PCC, since they are the state-of-the-art coding solutions. CTC defines the quantization points (QPs) and other test conditions for both solutions. To have a wider range of qualities and rates and reach similar bitrates for the MPEG V-PCC and AP2V-PCC solutions, which is desirable to assess the quality improvements, the CTC quantization values have been extended. The geometry and colour QPs adopted are presented in Table 1, where R1 to R5 were inherited from the MPEG CTC [69].

Table 1 – Quantization Points (QPs) for RD performance assessment [69], where the test points inherited from MPEG are in italic.

V-PCC			***AP2V-PCC***		
Test Point	**Geom. QP**	**Col. QP**	**Test Point**	**Geom. QP**	**Col. QP**
R1	*32*	*42*	**R01**	40	50
R2	*28*	*37*	**R02**	36	47
R3	*24*	*32*	***R1***	*32*	*42*
R4	*20*	*27*	***R2***	*28*	*37*
R5	*16*	*22*	***R3***	*24*	*32*
R6	13	18	***R4***	*20*	*27*
R7	10	14	***R5***	*16*	*22*

Note that R6 and R7 (for V-PCC) do not follow the decrement step as for the other test points for both geometry and colour, since they would achieve rates that are much larger than those obtained by AP2V-PCC. On the other hand, R01 does not follow the quantization point increment, since the maximum possible QP value is 50 which introduces a limitation.

Regarding G-PCC, the RD results presented in this Thesis are the anchor results obtained from [70]. The results were taken while using the *simple octree coding* and *Pred/Lifting Transform* for geometry and texture encoding, respectively (see Section 2.6). The CTC defines G-PCC geometry and colour QPs as presented in Table 2:

Table 2 - Quantization Points (QPs) for G-PCC RD performance assessment [69].

Test Point	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**
Geom. QP	8	4	2	4/3	8/7	16/15
Col. QP	256	128	64	32	16	8

It is important to note that the quantization function quantizes different data for V-PCC and G-PCC, and thus, the QP values are naturally very different among both solutions.

5.3. Objective Evaluation Metrics

Having detailed the test material and the test conditions, this section will outline the adopted objective evaluation metrics, which were already defined in detail in Section 2.4. Besides defining the test

conditions, the CTC document [69] also defines the objective evaluation metrics. Following the MPEG framework, the objective evaluation is performed based on RD performance, where the RD curves are plotted using PSNR as the quality metric. Thus, the quality will be presented for geometry in the form of D1 and D2 PSNR, as in Eq. 9, corresponding to the point-to-point and point-to-plane metrics, respectively. Regarding texture, there is a colour PSNR for each one of the three colour channels of the YUV colour space, as in Eq. 13. Since static content is being encoded, the rate is reported as bits per input point (bpp) and the bpp is calculated as the ratio between the compressed size (in bits) and the number of input points.

Since all point clouds in the test material are voxelized to 10 bits, the peak value for the geometry PSNR is 1023 ($2^{10} - 1$). For the colour PSNR, the peak is still 255 since it comes from the fact that the maximum colour value component-wise is 255 (2^8-1 bits). Besides the RD charts, the Bjontegaard-Delta bit-rate (BD-rate) and Bjontegaard-Delta PSNR (BD-PSNR) are also presented. The BD-rate represents the average value of how much the bit-rate is reduced (negative values) or increased (positive values) for the same PSNR. Similarly, the BD-PSNR represents the average value of how much the PSNR has increased (positive values) or decreased (negative values) for the same rate.

5.4. AP2V-PCC Parameters Optimization

The objective of this section is to study the impact of the AP2V-PCC parameters on the RD performance and, based on this assessment study, select appropriate parameter values. This optimization study was performed as follows: for each coding parameter, a range of relevant values is adopted and the best parameter value in terms of RD performance is selected and used to find the best value for the next studied parameter. The final objective assessment (see Section 5.5) is performed with the best values found for all parameters. Since the adaptive planes are expected to have more impact on the geometry, the decision on the best parameters considers mostly the results for the geometry component, taking as anchor the V-PCC results. To select the best values for the parameters, it is necessary to have a fully-working pipeline and, thus, reasonable initial values were attributed to each parameter. The parameters optimized are now described:

- **Number of Clusters (*K*) –** This parameter corresponds to the number of clusters in the K-Means Clustering module. The tested values for *K* were 6, 8, 10, 12 and 14. The lowest tested value was 6, since V-PCC uses 6 fixed clusters, and using a lower *K* value would lead to projection planes very poorly adapted to the point cloud data. The largest used value was 14, a value much greater than V-PCC, and using a larger *K* value would lead to a rather large number of sub-clusters, which would add a significant metadata overhead increase.
- **Expansion Factors –** This parameter defined the resolution expansion for the 3D local coordinates, and thus also the resolution for the occupancy, geometry and texture maps to be coded. The used expansion factors were within the interval [1.0; 1.8], with an interval of 0.2. The lowest value is defined as if there was no expansion (1.0), while the largest value is defined as 1.8, since larger values would lead to a much larger rate comparing to the one obtained for V-PCC.

To start by selecting the best *K*, an expansion factor of 1.6 was selected, which should not be very far from its best value, while using a sub-cluster projection approach.

5.4.1. Number of Clusters (*K*) Optimization

Figure 51 shows the geometry PSNR D1 RD performance, while varying the number of clusters (K) for frame 690 and 1300 of *Soldier* and *Longdress* sequences, respectively. The geometry bitrate considers the geometry map, occupancy map and patch information (no colour bitrate).

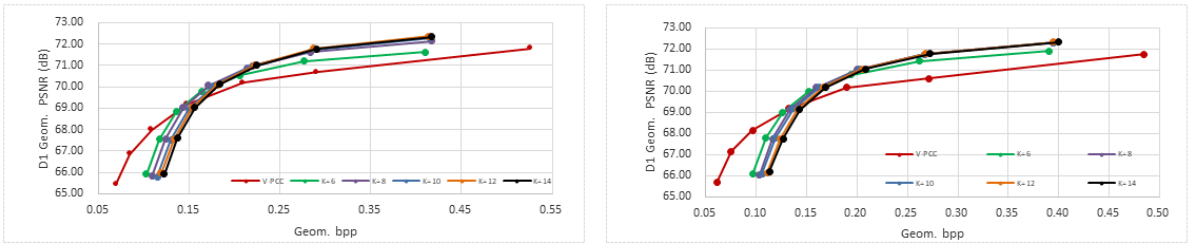


Figure 51 – D1 Geometry RD performance for frame 690 and 1300 of Soldier (a) and Longdress (b) sequences, respectively, for AP2V-PCC with K= {6, 8, 10, 12, 14}.

For the lower rates, the lowest *K* values achieve the best geometry PSNR D1 RD performance, since they have the lowest number of sub-clusters (or patches), and thus a lower overhead of metadata. As the quantization parameter decreases (along the dots over each curve from left to right), the bitrate associated to the depth maps (not counting occupancy map nor patch info) starts increasing but the patch information and occupancy map rates remain the same (as they are independent of the depth quantization). This means that, for larger rates, the rate associated to the occupancy map and patch information becomes less significant when compared with the depth map rate. Consequently, for larger *K*, the PCADP and variance get lower, implying a lower depth map rate and a higher PSNR, and thus they start outperforming the lower *K*. Since AP2V-PCC only outperforms V-PCC for larger bpp, the best *K* value for the higher bitrates was selected. The BD-PSNR over the 4 highest quantization points for all *K* was computed using as reference the V-PCC 4 highest quantization points. The largest BD-PSNR value was achieved for *K=12* for both point clouds, and thus this value was selected as the best *K*.

5.4.2. Expansion Factors Optimization

From now on, *K* is set to 12, as this was the best value obtained from the study described in the previous section. The variation of geometry PSNR D1 RD performance with the Expansion Factors (*EFs*) is illustrated in Figure 52 for the same point clouds used in the *K* value optimization.

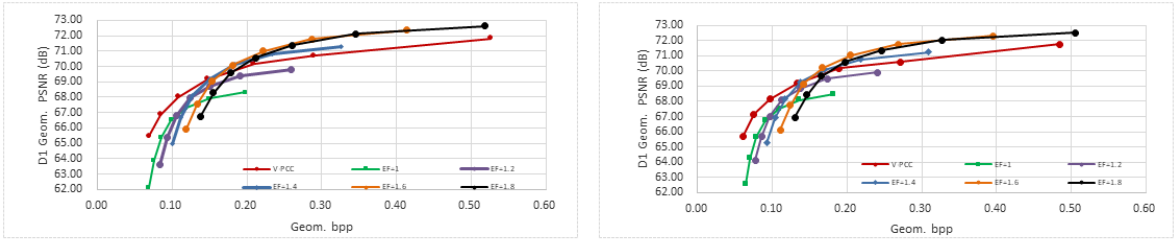


Figure 52 - D1 Geometry RD performance for frame 690 and 1300 of Soldier (a) and Longdress (b) sequences, respectively, for AP2V-PCC with EF={6, 8, 10, 12, 14}.

Figure 52 shows a clear trend, where the largest *EF* values outperform the lowest *EF* values for larger bitrates and the opposite happens for the lower bitrates. This observation is valid from *EF=1.0* until *EF=1.6*, but *EF=1.8* does not outperform *EF=1.6* for any rate, since there is a saturation effect of the geometry quality for *EF=1.8* despite the increase in terms of bitrate. This means that when the expansion factor goes from 1.6 to 1.8, the increase on the geometry quality does not compensate the investment in the geometry bitrate. Since this occurs between *EF=1.6* and *EF=1.8*, the selected expansion factor is 1.6, as this value allows achieving the highest PSNR and is the last *EF* that outperforms all other lower *EF* values for larger rates.

At the next objective and informal subjective assessment, the coding parameters will be *K=12*, *EF=1.6* and the sub-cluster-based approach will be used.

5.5. RD Performance Assessment

The goal of this section is to present the RD performance for AP2V-PCC and to benchmark it with two different alternative solutions: Static V-PCC (this means only the Intra coding component) and also G-PCC which corresponds to the MPEG PCC solution for static content. This section starts by analysing the geometry RD performance and after the colour RD performance.

5.5.1. Geometry RD Performance Assessment

As previously mentioned, the geometry RD performance will be assessed based on the PSNR computed for the Point-to-Point (D1) and Point-to-Plane (D2) metrics. Note that V-PCC decoded point clouds have some duplicate points, and thus duplicated points are dropped before applying the quality metric (it will keep one point per 3D occupied position with the average colour); duplicate points do not change the perceived quality of the point cloud, but would influence the D1 and D2 metric values as the same position would be accounted multiple times.

A. RD Performance for Geometry and Total Rates

The RD performance for the four selected test point clouds selected is presented in Figure 53 to Figure 60 for geometry bits per point (depth map, occupancy map and patch information rate), and for total bits per point (also including colour rate). For the geometry bpp charts, two of the G-PCC test points (the ones with highest bitrates) have been omitted, so that similar geometry rates for all solutions are used; the main reason for the differences is the very different bitrate distribution between colour and geometry for G-PCC and V-PCC. The D1 RD results are presented in Figure 53 to Figure 56 and the D2 RD results are presented in Figure 57 to Figure 60.

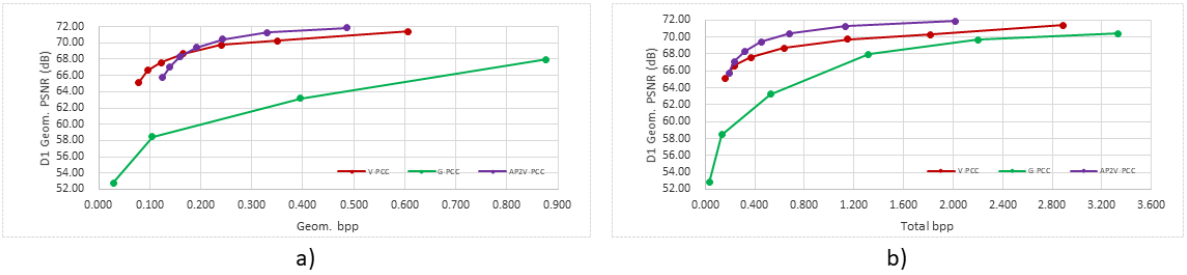


Figure 53 – D1 Geometry RD chart for (a) geometry bpp and (b) total bpp for frame 1550 of Redandblack sequence.

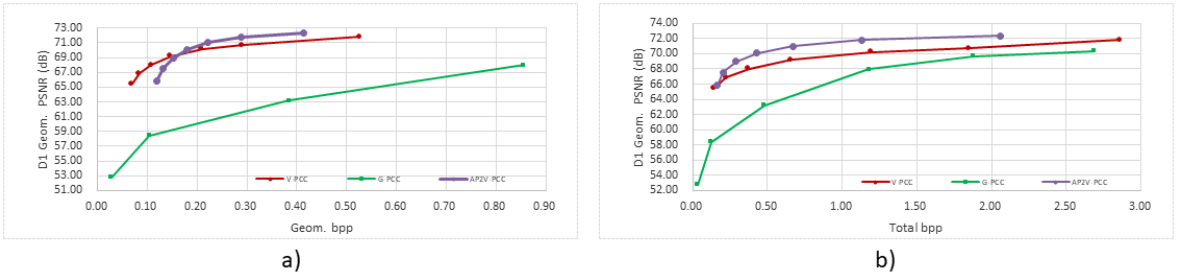


Figure 54 – D1 Geometry RD chart for (a) geometry bpp and (b) total bpp for frame 690 of Soldier sequence.

Figure 55 - D1 Geometry RD chart for (a) geometry bpp and (b) total bpp for frame 1300 of Longdress sequence.

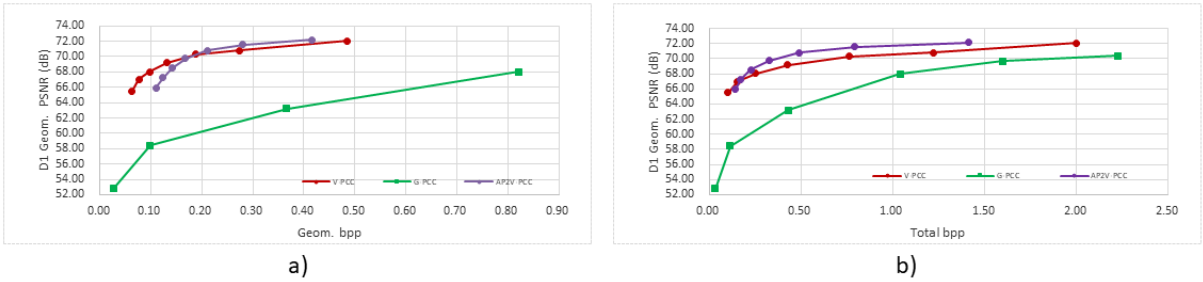


Figure 56 - D1 Geometry RD chart for (a) geometry bpp and (b) total bpp for frame 1200 of Loot.

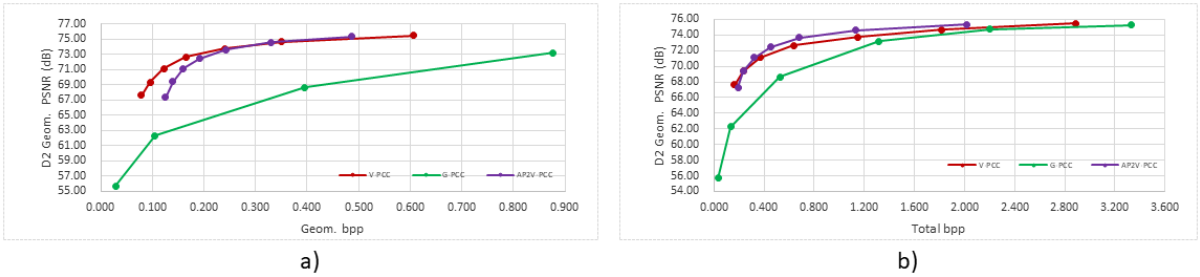


Figure 57 - D2 Geometry RD chart for (a) geometry bpp and (b) total bpp for frame 1550 of Redandblack sequence

Figure 58 - D2 Geometry RD chart for (a) geometry bpp and (b) total bpp for frame 690 of Soldier sequence.

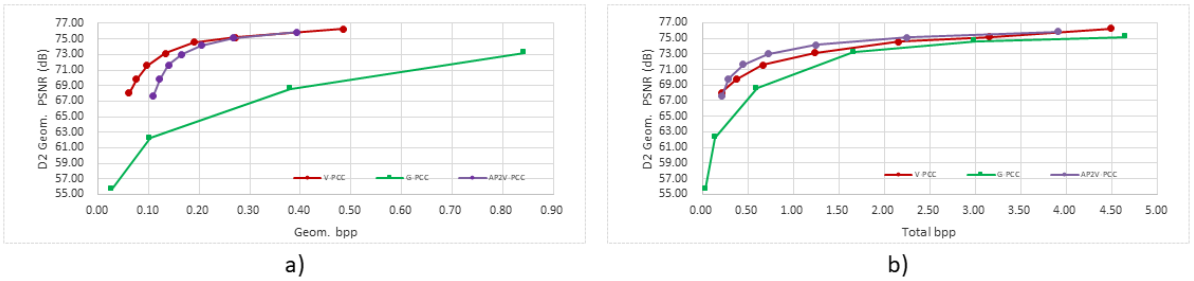


Figure 59 – D2 Geometry RD chart for (a) geometry bpp and (b) total bpp for frame 1300 of Longdress sequence.

Figure 60 - D2 Geometry RD chart for (a) geometry bpp and (b) total bpp for frame 1200 of Loot sequence.

From the available results for the test content, it can be concluded that both AP2V-PCC and V-PCC outperform G-PCC, in terms of geometry RD performance for both geometry and total rate. The G-PCC quality is rather low, since the ratio between the number of output points and the number of input points is very low for lower rates, e.g. around 0.02 for the lowest RD point. On the other hand, for AP2V-PCC and V-PCC, this ratio is very similar for different test points, namely between 1.0 and 1.5, respectively. This difference is caused by the techniques and data structures employed to code the geometry, notably octree encoding for G-PCC, and video-based coding for AP2V-PCC and V-PCC. Since the video-based coding techniques achieve the best RD performance, it is useful to assess the BD-Rate and BD-PSNR for AP2V-PCC taking as reference V-PCC, thus identifying the gains of the new coding solution; the results are presented in Table 3.

From the RD charts and Table 3, it can be concluded that AP2V-PCC outperforms V-PCC when considering the total rates, for both quality metrics, except for the first one or the first two test points; however, these results have to be considered together with the colour RD performance results because there is a trade-off involved. For geometry rates, AP2V-PCC outperforms V-PCC only for larger rates, when using the D1 metric, and is similar for the D2 metric. On the other hand, for lower rates, AP2V-PCC is outperformed by V-PCC for both quality metrics.

Table 3 - D1 and D2 Geometry BD-rate and BD-PSNR for AP2V-PCC using V-PCC as reference considering geometry and overall rates.

Sequence	Geometry rate				Overall rate			
	D1		D2		D1		D2	
	BD-rate (%)	BD-PSNR (dB)	BD-rate (%)	BD-PSNR (dB)	BD-rate (%)	BD-PSNR (dB)	BD-rate (%)	BD-PSNR (dB)
Redandblack	**-2%**	**0.423**	25%	-0.607	**-42%**	**1.286**	**-17%**	**0.604**
Soldier	5%	**0.361**	31%	-0.820	**-48%**	**1.549**	**-24%**	**0.870**
Longdress	4%	**0.461**	38%	-0.915	**-55%**	**1.583**	**-27%**	**0.837**
Loot	14%	**-0.064**	47%	-1.252	**-32%**	**0.977**	**-2%**	**0.173**
Average	5%	0.295	35%	-0.899	**-44%**	**1.349**	**-17%**	**0.621**

B. Rate Distribution

The fact that AP2V-PCC is outperformed by V-PCC for the lower rates, while the opposite happens for larger rates, can be explained with the help of Figure 61, where the rate distribution for the different test points is depicted for the two coding solutions using as example frame 1550 of *Redandblack* sequence.

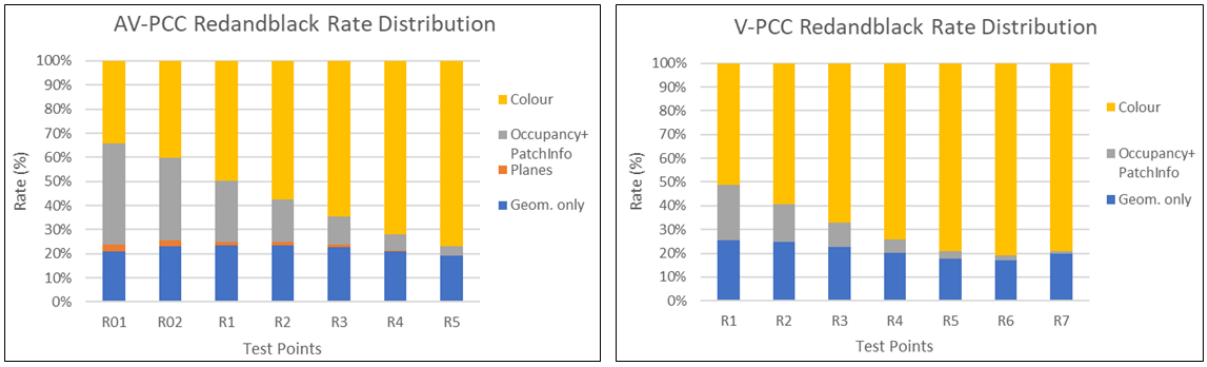


Figure 61 – Rate distribution when encoding frame 1550 of Redandblack with AP2V-PCC (left) and V-PCC (right).

For the *Redandblack* sequence, the occupancy map and patch information cost 0.037 bpp for V-PCC and 0.08 bpp for AP2V-PCC, or 0.085 bpp if the rate to transmit the planes is also included. The occupancy map rate increase is due to the increase of its resolution (from 1,638,400 to 2,355,200 'occupancy pixels' with a 1.6 expansion factor) while the increase on the patch information bitrate is due to the need to transmit the information to define the adaptive planes; when *K* increases, the number of sub-clusters also increases, and thus the number of patches (with the associated metadata) will also increase (from 125 in V-PCC to 173 in AP2V-PCC). Since the occupancy and patch information rate overhead is fixed, it starts becoming less and less significant as the overall rate increases since the geometry and texture rates will be much larger. Since the colour rate is much larger than all other rates, when the RD performance assessment is performed with total rate, the occupancy map and patch information bitrate lose their influence for the lower test point, and thus, AP2V-PCC starts outperforming V-PCC for the lower test point, than when the comparison is performed with only the geometry rate.

C. Missed Points Evaluation

To achieve a better understanding of the AP2V-PCC to V-PCC gains, it is important to analyse the number of missed points obtained at the end of the patch generation process, since this number also indicates the geometry quality of the point cloud representation associated to the planar depth maps. Moreover, the reduction of missed points (and non-represented points) is one of the main motivations to apply the adaptive plane projection technique which is costly in rate. The information on the number of missed points, and number of non-represented points, which have been both defined on Section 4, is presented on Table 4.

Table 4 – The number of missed points, points with a distance to the reconstructed point cloud equal to 1 and non-represented points, for the test material.

Sequence	Missed Points			Non-Represented Points		
	V-PCC	AP2V-PCC	Reduction (%)	V-PCC	AP2V-PCC	Reduction (%)
Redandblack	2,091	1,015	51	25,209	6,944	72
Soldier	1,637	344	79	28,759	3,654	87
Longdress	1,706	534	69	24,257	3,249	87
Loot	733	624	15	15,651	5,499	65
Average	1,542	629	59	23,469	4,832	79

As shown in Table 4, the AP2V-PCC encoded point clouds achieve a lower number of non-represented points when compared to V-PCC; *Soldier* and *Longdress* are the point clouds with the lowest number of non-represented points, and also with the best RD performance results, as confirmed by the BD-rate and BD-PSNR in Table 3. For the *Loot* point cloud, the reduction on the number of missed points for AP2V-PCC with respect to V-PCC is smaller and, as such, achieves the lowest RD gains, when comparing both solutions.

D. Projection Planes Suitability Assessment

Finally, an assessment of how well the points (which represent a surface) are adapted to the respective projection plane is presented in Table 5 using the sub-cluster-based PCADP. The weighted standard deviation of the patch depth values (SDPD) for the D0 layer is also presented in the same table computed as:

$$SDPD = \frac{1}{\mathrm{N}_{Patches}} \cdot \sum_{patch \in Patches} \mathrm{N}_{patch} \cdot StdDev(DM_{patch}), \quad (17)$$

where $N_{Patches}$ is the number of filled pixels in the D0 depth map, *Patches* is the complete set of patches, N_{patch} is the number of D0 filled pixels in each patch and $StdDev(DM_{patch})$ is the standard deviation of the D0 filled depth map values for each patch. It is worth stressing that the standard deviation above is only computed for D0 layer, since D1 layer depth values are always between 0 and 4, which is a much lower range of values, when compared to D0.

Table 5 – Sub-cluster-based PCADP and patches standard deviation (SDPD) for V-PCC and AP2V-PCC

Sequence	Sub-cluster PCADP		Patches Standard Deviation (SDPD)	
	V-PCC	AP2V-PCC	V-PCC	AP2V-PCC
Redandblack	0.805	0.880	21.923	13.391
Soldier	0.810	0.890	22.359	11.745
Longdress	0.816	0.904	23.905	10.496
Loot	0.828	0.907	22.474	15.154
Average	0.815	0.895	22.665	12.697

Since PCADP indicates how adapted are the projection planes to the data, its increase corresponds to a lower number of overlapping points in the same 2D depth map pixel (after projection). Table 5 shows that the PCADP is always higher for AP2V-PCC, which means that the corresponding planes are better adapted for this coding solution. On the other hand, the SDPD informs about the variance of the depth map values. Note that, for AP2V-PCC, the depth coordinate values have been multiplied by the expansion factor, which greatly increases the patches depth maps standard deviation. Despite this depth expansion, the standard deviation is still considerably lower than V-PCC for all test material. This means that more points respect the surface thickness threshold (maximum difference for a pixel depth value between D1 and D0 layers), and thus less points are filtered during the patch-level depth map creation. Hence, the number of non-represented points is lower for all the point clouds, since the PCADP is greater and the variance is lower for all test material.

Once more, frame 1200 of *Loot* sequence is the point cloud with the greatest V-PCC PCADP and also the point cloud for which the standard deviation difference between AP2V-PCC and V-PCC is the lowest. This explains why *Loot* is the sequence with the lowest difference on the number of non-represented points and consequently worst RD results (largest BD-rate and lowest BD-PSNR) among the 4 test point clouds as shown in Table 4 and Table 3, respectively. Frame 1550 of *Redandblack* sequence is the point cloud with the largest number of non-represented points in AP2V-PCC, and this is explained by the fact that it has the lowest PCADP and the second standard deviation. This poor adaptation is due to the hair and high angular amplitudes between some adjacent parts of the body. The two remaining point clouds (*Soldier* and *Longdress*) have large PCADPs and low standard deviations, and thus a low number of non-represented points are obtained as well as the best geometry RD performance.

5.5.2. Colour RD Performance Assessment

The colour RD performance results for the YUV colour space are presented in Figure 62 to Figure 67. Since the colour coding process is made for the decoded geometry, only the total rate has to be considered. There is naturally a trade-off between the colour and geometry qualities when a certain total rate is considered.

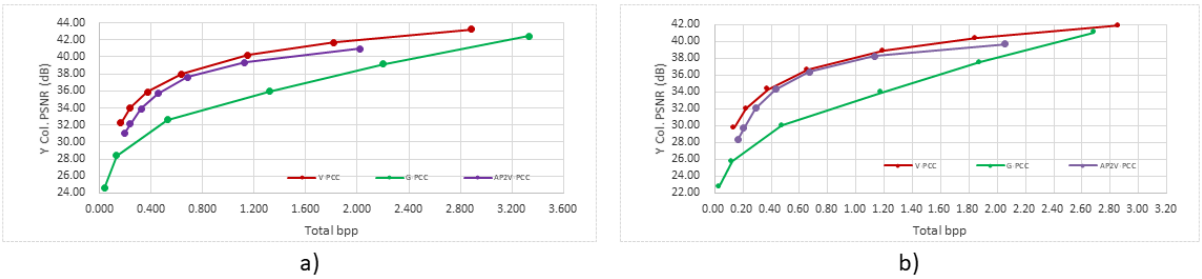


Figure 62 - Y component RD performance for (a) frame 1550 of Redandblack sequence and (b) frame 690 of Soldier sequence.

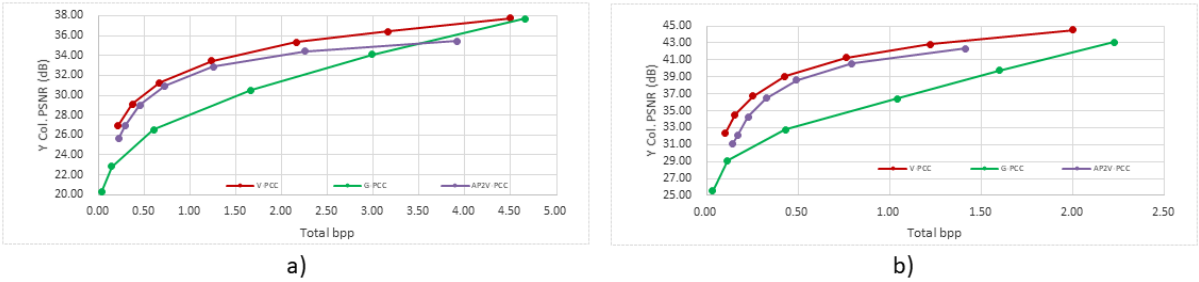


Figure 63 - Y component RD performance for (a) frame 1300 of Longdress sequence and (b) frame 1200 of Loot sequence.

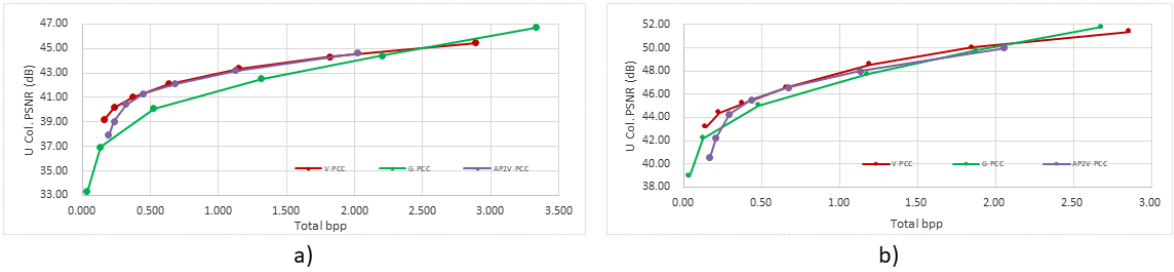


Figure 64 - U component RD performance for (a) frame 1550 of Redandblack sequence and (b) frame 690 of Soldier sequence.

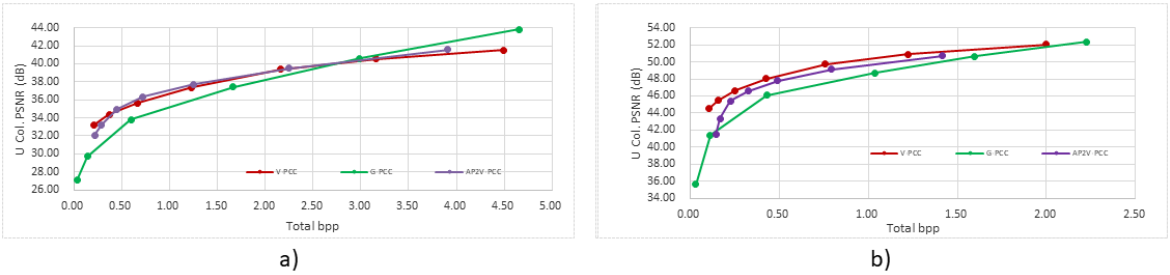


Figure 65 - U component RD performance for (a) frame 1300 of Longdress sequence and (b) frame 1200 of Loot sequence.

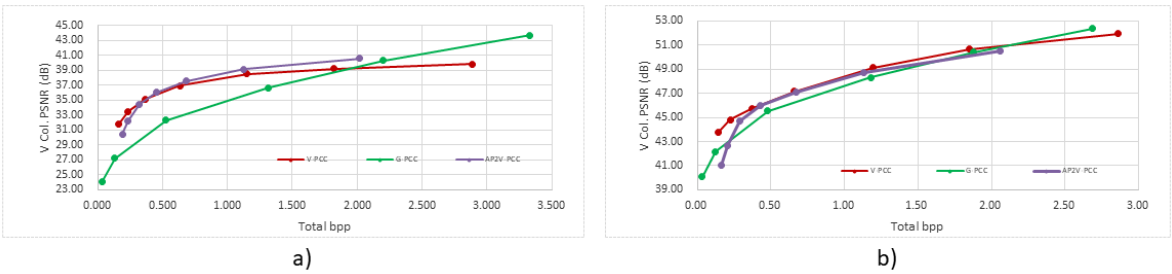


Figure 66 - V component RD performance for (a) frame 1550 of Redandblack sequence and (b) frame 690 of Soldier sequence.

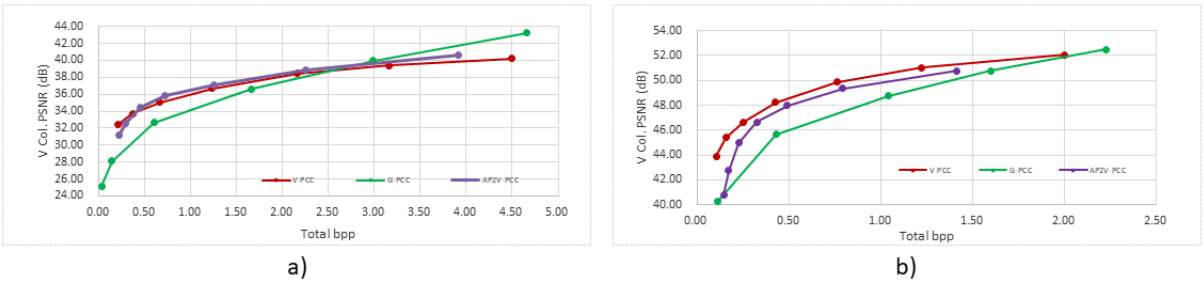


Figure 67 - V component RD performance for (a) frame 1300 of Longdress sequence and (b) frame 1200 of Loot sequence.

The Y (luma) component has the highest perceptual impact, since the Human Visual System has a lower acuity for colour differences than for luminance [71]. Thus, the BD-rate and BD-PSNR are presented for the luma component in Table 6, comparing AP2V-PCC to the reference V-PCC, since they are the two best performing solutions. Regarding the chrominances, both AP2V-PCC and V-PCC encode the texture images with a 4:2:0 sub-sampling, meaning that the luminance has four times the resolution of the chrominances to explore the fact that chrominance values have a lower perceptual impact. On the other hand, G-PCC uses a different transform to encode the colour attribute, and thus the results for G-PCC do not show the typical 4:2:0 sub-sampling saturation for larger rates.

Table 6 - Y component BD-rate and BD-PSNR using as reference V-PCC.

Sequence	BD-rate (%)	BD-PSNR (dB)
Redandblack	30%	-1.091
Soldier	21%	-0.846
Longdress	28%	-0.942
Loot	38%	-1.408
Average	29%	-1.072

From the Y RD charts and Bjontegaard-delta values, it is clear that currently the colour attribute does not benefit as much from the adaptive plane projections as the geometry, since AP2V-PCC is outperformed for all sequences. To understand this effect, the *Loot* Y RD chart is depicted on Figure 68, containing the Y colour quality after recolouring (V-PCC_reconstructed and AP2V-PCC_reconstructed curves), i.e. considering the decoded geometry with original colour (and thus with only geometry errors affecting the colour quality), and the decoded colour quality with the associated rate (after being compressed by the standard video codec and decoded) (V-PCC and AP2V-PCC curves). Since the results for the recoloured point clouds have been obtained before encoding, there is no rate associated to them. However, to facilitate the comparison between the recoloured and decoded point clouds, the rate attributed to each test point of the reconstructed point cloud is the one resulting from the encoding.

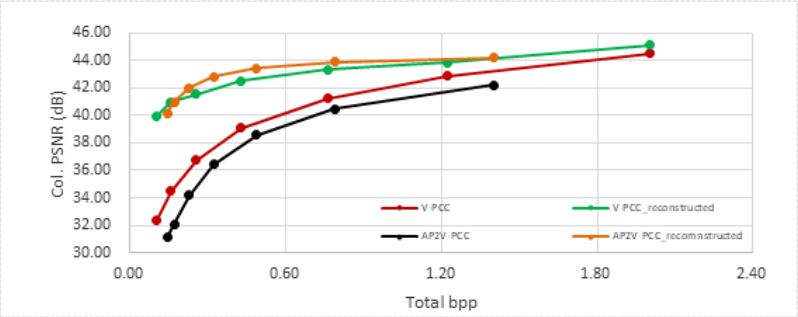


Figure 68 – Y component RD performance for frame 1200 of Loot sequence comparing the final decoded colour quality with quality of recoloured point cloud without colour quantization (suffix '_reconstructed').

As shown in Figure 68, AP2V-PCC has better colour quality than V-PCC (excepting the first quantization point) after recolouring (due to the better geometry), but after colour coding, V-PCC outperforms AP2V-PCC for the same total rate. This can be explained by the fact that geometry gains are achieved at the price of increasing the depth maps and occupancy map resolutions, and this increase in resolution is also propagated to the texture maps, thus increasing the rate. Even though, the colour quality for the recoloured point cloud increases due to the better geometry, if the same total rate is spent, the final AP2V-PCC colour quality decreases when compared to V-PCC also because the texture map resolution has to be increased, following the depth and occupancy maps resolution increase.

5.6. Informal Subjective Assessment

To better understand the AP2V-PCC strengths and weaknesses, notably in comparison with its parent codec, V-PCC, an informal subjective assessment was performed for two distinct cases: **i)** both solutions with low total rate; **ii)** both solutions with the lowest quantization step. The point clouds used for this experiment are frame 1550 of *Redandblack* sequence and frame 1200 of *Loot* sequence. The results are presented in Figure 69 and Figure 70.

The informal subjective assessment was performed using *Meshlab*, an open source software to visualize, edit and process 3D meshes and point clouds, which can be found in [72]. A close inspection of Figure 69 and Figure 70, showing results for the lower rates, indicates that the colour of the AP2V-PCC point clouds is rather blurred when compared to the corresponding V-PCC point clouds, since the quality of the colour is lower. However, for this same range of rates, there are multiple holes in the point clouds coded with V-PCC, which are not present in the AP2V-PCC results, due to the increased quality of the AP2V-PCC geometry. For the best quality case, the AP2V-PCC colour blurring is not perceptible anymore, as the colour quality is already rather large. However, both V-PCC sequences still have artefacts, notably for frame 1550 of *Redandblack* sequence, there are still some missed points (e.g. forehead and nose), resulting from the fact that these points have not been included in the patches during the patch generation process. Moreover, for frame 1200 of Loot sequence, there is a standing out artefact, namely the ear. This part has several holes, which are caused by the high angular difference of the ear estimated normals considering the respective projection plane. Since AP2V-PCC uses adaptive planes, the angular difference is lower and the ear does not have noticeable holes.

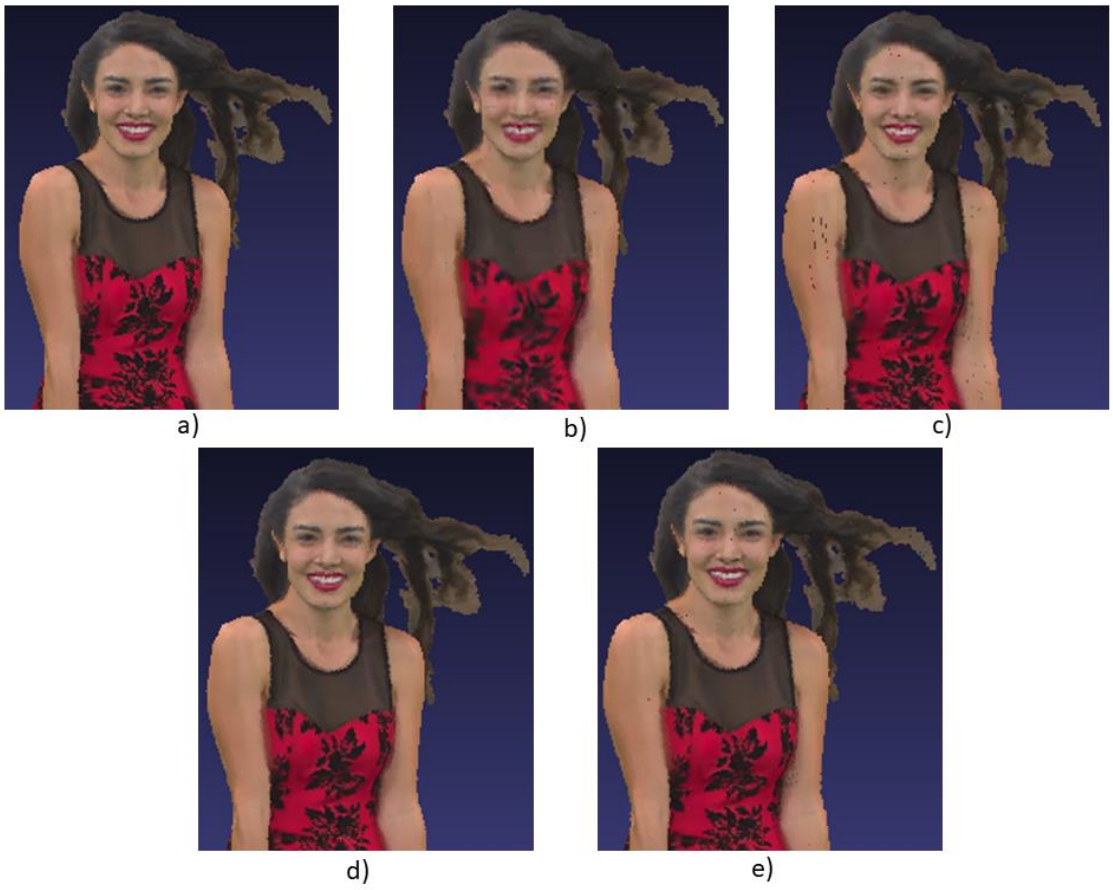


Figure 69 – Frame 1550 of Redandblack [4]: a) Original; b) AP2V-PCC for 0.32 total bpp; c) V-PCC for 0.37 bpp; d) AP2V-PCC for 2.11 total bpp; V-PCC for 2.89 total bpp.

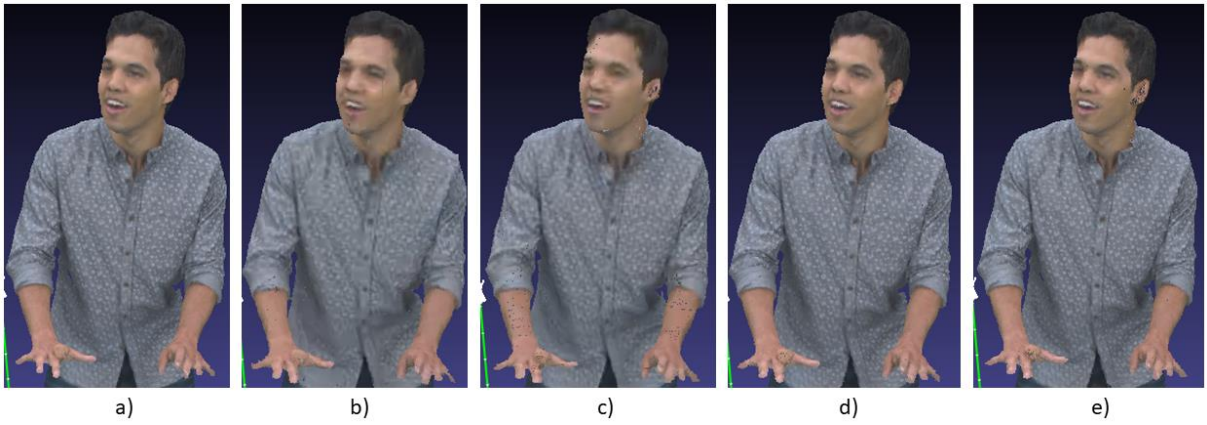


Figure 70 - Frame 1200 of Loot [4]: a) Original; b) AP2V-PCC for 0.25 total bpp; c) V-PCC for 0.24 bpp; d) AP2V-PCC for 1.47 total bpp; V-PCC for 2.00 total bpp.

Following the objective and subjective assessment, it is possible to outline the strengths and weaknesses of the proposed AP2V-PCC coding solution:

- **Strengths –** AP2V-PCC outperforms the other coding solutions in terms of geometry for medium to larger total rates. The projection planes are better adapted to the points, and thus the number of missed points is reduced, which has a high impact on the subjective quality. The number of output points is very close to the number of input points if the duplicate points are removed.

- **Weaknesses –** The main AP2V-PCC weakness is the slightly worse colour RD performance when compared to V-PCC; therefore, the colour of the decoded point clouds at lower rates has a blurred effect, which is not noticeable for V-PCC point clouds at similar rates.

In conclusion, the proposed AP2V-PCC coding solution adapts the projection planes to obtain a better geometry RD performance with a slight penalty on the colour RD performance, and thus further investment on this technique should be done to address the colour performance penalty.

Chapter 6

6. Conclusions and Future Work

This chapter will start by outlining the key contributions of this Thesis and the main conclusions of this work. After some future work is proposed.

6.1. Main Achievements and Conclusions

The objective of this Thesis was to design a point cloud coding solution able to improve the MPEG Video-based Point Cloud Coding (V-PCC) standard performance for static point clouds by adapting the planes on which the points are projected and transformed into maps. Both coding solutions still benefit from the very efficient HEVC standard to code the projected depth and texture maps.

From the results, it can be concluded that the objective was largely achieved, since a new solution bringing significant geometry compression gains and improving the subjective quality for larger rates was designed. The main achievements of this Thesis are:

- **Better geometry RD performance** is achieved for medium to large total rates, at the price of a worse colour RD performance, where the trade-off between the two compensates for large rates; in some sense, the new tool changes the depth-texture trade-off with benefit for some rates/qualities.
- **Less holes and better subjective quality for larger rates** are achieved for the decoded point clouds, due to a better adaptation of the projection planes, which is translated into a lower number of points being discarded and never sub-clustered again during the patch generation.
- **No duplicates and number of output points closer to number of input points**, since the duplicates are discarded at the decoder as having the same colour.

The rationale behind dropping the fixed projection planes and computing the projection planes adaptively was to reduce the angular difference between the points normals and the respective projection planes normals, as well as the variance of the patches' depth maps. The initial assumption was that these two effects would enable a lower number of non-represented points and, as a consequence, to increase the geometry RD performance. To be able to project into new planes, new local coordinates had to be calculated and initially expressed with floating-point values. As the texture, geometry and occupancy maps have regular grids, the floating-point values had to be rounded to fit them, incurring in a large geometry distortion, and overlapping of multiple points in the same 3D *bin*. To solve this issue, the 3D local coordinates resolution was increased, which reduced the rounding error and the number of points overlapping in the same 3D *bin*. This increase of resolution created more duplicate points due to the occupancy map being coded with *B0=4*. Hence, to avoid duplicate points with different colours, recolouring pre and post-processing processes were added, ensuring that all duplicates have the same colour, and thus can be discarded at the decoder.

As highlighted in Chapter 5, the angular difference of the point normal to the respective projection plane normal, which is quantified by the PCADP metric, and the depth standard deviation were both reduced. As foreseen by the initial assumptions, the number of non-represented points were drastically reduced. Since a lower number of non-represented points is correlated to an increase on the geometry quality, the geometry has effectively PSNR increased, but AP2V-PCC only outperforms V-PCC for medium to larger rates, since the overhead of sending the new projection planes, increasing the number of patches and increasing the resolution of the occupancy map, is dominant for lower rates. This increase in geometry quality also increases the colour quality of the recoloured point cloud (at the encoder), which will be after HEVC encoded. However, the effect of increasing the geometry and occupancy maps resolutions to improve the geometry quality also propagates to the texture maps and increases the colour rate if a similar quality is targeted. Thus, to achieve similar rates to V-PCC, the colour quality must be more quantized in AP2V-PCC, which makes the AP2V-PCC colour quality dropping more regarding the mentioned recoloured point cloud than the V-PCC colour quality. This makes the rate investment on improving the geometry (instead of on a lower colour quantization) not to be worth it from a colour RD point of view. The better geometry RD performance and lower number of non-represented points is reflected on the subjective assessment, since the V-PCC holes and related artefacts disappear. However, the lower AP2V-PCC colour quality brings a blurred effect for lower rates.

As a final note, one can conclude that the assumptions initially made were correct and, in fact, the predicted performance gains were largely obtained.

6.2. Future Work

Finally, considering the limitations of the proposed solution, some future work directions may be defined:

- **Sub-cluster-based expansion factors optimization –** This work item aims the optimization of the adaptive resolution control. Currently, all 3D sub-clusters local coordinates are expanded with the same expansion factor; however, since the sub-clusters are different and have different projection planes, each sub-cluster should, in principle, be expanded differently according to a certain criterion. The criterion may be the minimization of some parameter, e.g. the number of missed points, colour distortion, geometry distortion, etc. However, since the resolution increase would not be equal for all sub-clusters, an expansion factor would have to be sent for each of them. Moreover, the resolution increase can also be different for different coordinates, e.g. depending on the angles between the patch and plane normals, thus implying a more complex optimization. By adapting the expansion at the sub-cluster level, the RD performance is expected to increase for geometry and colour, since the rate should be spent more efficiently, namely on the sub-clusters needing a larger resolution increase, instead of having all sub-clusters getting the same resolution increase.
- **Dynamic extension –** This work item targets the extension of AP2V-PCC to dynamic point clouds. While V-PCC was designed for dynamic content, the new tools introduced in AP2V-PCC were not designed with the intention of exploring the temporal redundancy. Consequently, the tools (and their respective implementation) would have to undergo the necessary adaptations to encode a sequence of correlated point cloud frames instead of only one. Moreover, a study of the impact of each coding

parameter on the exploitation of the temporal redundancy would also have to be performed, so that each one is optimized to this new scenario.

In summary, there are still opportunities for further enhancements to improve the proposed AP2V-PCC coding solution; ideally, some of these tools could be integrated in one of the next versions of MPEG V-PCC.

References

[1] F. Pereira, E. A. B. Silva, and G. Lafruit, “Plenoptic imaging: representation and processing,” in *Academic Press Library in Signal Processing*, vol. 6, R. Chellappa and S. Theodoridis, Eds. Academic Press, pp. 75–11, 2018.

[2] D. Thanou, P. A. Chou, and P. Frossard, “Graph-based compression of dynamic 3D point cloud sequences,” *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1765–1778, Apr. 2016.

[3] R. L. de Queiroz and P. A. Chou, “Motion-compensated compression of dynamic voxelized point clouds,” *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3886–3895, Aug. 2017.

[4] E. d’Eon, B. Harrison, T. Myers, and P. A. Chou, “8i Voxelized full bodies - a voxelized point cloud dataset,” ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva, Switzerland, Jan. 2017.

[5] JPEG Convener, “JPEG Pleno – scope, use cases and requirements,” Doc. ISO/IEC JTC1/SC29/WG1/N72006, Geneva, Switzerland, Jun. 2016.

[6] MPEG 3DG and Requirements Subgroups, “Call for proposals for point cloud compression V2,” Doc. ISO/IEC JTC1/SC29/WG11/N16763, Hobart, AU, Apr. 2017.

[7] E. H. Adelson and J. R. Bergen, “The Plenoptic Function and the Elements of Early Vision,” *Computational Models of Visual Processing*, pp. 3–20, 1991.

[8] “Plenoptic camera - The Camera Maker.” [Online]. Available: http://cameramaker.se/plenoptic.htm. [Accessed: 14-Oct-2017].

[9] “The self-reconfigurable camera array - Advanced Multimedia Processing Lab.” [Online]. Available: http://chenlab.ece.cornell.edu/projects/MobileCamArray/. [Accessed: 14-Oct-2017].

[10] L. Shapiro and G. Stockman, “3D models and matching,” in *Computer vision*, 1st ed., NJ: Prentice-Hall, pp. 517–526, 2001.

[11] “Coherent light - The Free Dictionary.” [Online]. Available: http://www.thefreedictionary.com/coherent+light. [Accessed: 11-Oct-2017].

[12] R. Collier, C. Burckhardt, and L. Lin, “Introduction to basic concepts,” in *Optical holography*, 1st ed., New York: Academic Press INC., pp. 3–35, 1971.

[13] “Dense Modelling.” [Online]. Available: http://grail.cs.washington.edu/rome/dense.html. [Accessed: 14-Oct-2017].

[14] “Virtual reality vs. augmented reality - Augment News.” [Online]. Available: http://www.augment.com/blog/virtual-reality-vs-augmented-reality/. [Accessed: 16-Oct-2017].

[15] S. C.-Y. Yuen, G. Yaoyuneyong, and E. Johnson, “Augmented reality: an overview and five directions for AR in education,” *Journal of Educational Technology Development and Exchange*, vol. 4, no. 1, Jun. 2011.

[16] “The 20 best augmented-reality apps - Digital Trends.” [Online]. Available: https://www.digitaltrends.com/mobile/best-augmented-reality-apps/. [Accessed: 16-Oct-2017].

[17] “Six Flags and Samsung partner to launch first virtual reality roller coasters in North America - Business Wire.” [Online]. Available: http://www.businesswire.com/news/home/20160303005730/en/Flags-Samsung-Partner-Launch-Virtual-Reality-Roller. [Accessed: 16-Oct-2017].

[18] “Six degrees of freedom - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Six_degrees_of_freedom. [Accessed: 12-Oct-2017].

[19] M. Weinmann, “Preliminaries of 3D point cloud processing,” in *Reconstruction and analysis of 3D scenes*, 1st ed., Cham: Springer International Publishing, pp. 17–38, 2016.

[20] J. Salvi, J. Pagès, and J. Batlle, “Pattern codification strategies in structured light systems,” *Pattern Recognition*, vol. 37, no. 4, pp. 827–849, Apr. 2004.

[21] R. Horaud, M. Hansard, G. Evangelidis, and C. Ménier, “An Overview of depth cameras and range scanners based on Time-of-Flight technologies,” *Machine Vision and Applications*, vol. 27, no. 7, pp. 1005–1020, Oct. 2016.

[22] “Microsoft Kinect review - CNET.” [Online]. Available: https://www.cnet.com/au/products/microsoft-kinect-motion-sensor-wired-lpf00004/review/. [Accessed: 02-Nov-2017].
[23] “Optical encoders and LiDAR scanning.” [Online]. Available: http://www.renishaw.com/en/optical-encoders-and-lidar-scanning--39244. [Accessed: 01-Nov-2017].
[24] H.-H. Vu, P. Labatut, J.-P. Pons, and R. Keriven, “High accuracy and visibility-consistent dense Multiview Stereo,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 889–901, May 2012.
[25] M. J. Westoby, J. Brasington, N. F. Glasser, M. J. Hambrey, and J. M. Reynolds, “‘Structure-from-Motion’ photogrammetry: a low-cost, effective tool for geoscience applications,” *Geomorphology*, vol. 179, pp. 300–314, Dec. 2012.
[26] N. Snavely, S. M. Seitz, and R. Szeliski, “Modeling the world from Internet photo collections,” *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, Nov. 2008.
[27] “ULB unicorn photogrammetric point cloud data,” MPEG, MPEG m41742, Macau, Oct. 2017.
[28] A. Swart, J. Broere, R. Veltkamp, and R. Tan, “Refined non-rigid registration of a panoramic image sequence to a LiDAR point cloud,” in *Photogrammetric image analysis*, 1st ed., vol. 6952, U. Stilla, F. Rottensteiner, H. Mayer, B. Jutzi, and M. Butenuth, Eds. Berlin: Springer Berlin Heidelberg, pp. 73–84, 2011.
[29] “Point Cloud Library.” [Online]. Available: http://pointclouds.org/. [Accessed: 26-Oct-2017].
[30] M. Berger *et al.*, “State of the art in surface reconstruction from point clouds,” EG’2014 - STAR, Strasbourg, France, Apr. 2014.
[31] L. Linsen, K. Müller, and P. Rosenthal, “Splat-based ray tracing of point clouds,” *Journal of WSCG*, vol. 15, no. 1–6, pp. 51–58, 2007.
[32] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, “Computing and rendering point set surfaces,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, Jan. 2003.
[33] “Poisson surface reconstruction: user manual - CGAL 4.11.” [Online]. Available: https://doc.cgal.org/latest/Poisson_surface_reconstruction_3/index.html. [Accessed: 17-Nov-2017].
[34] T. Weyrich *et al.*, “A hardware architecture for surface splatting,” *ACM Transactions on Graphics*, vol. 26, no. 3, p. 90, Jul. 2007.
[35] A. Adamson and M. Alexa, “Ray tracing point set surfaces,” SMI’ 03, Washington DC, USA, May 2003.
[36] “Ray tracing (graphics) - Wikipedia,” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Ray_tracing_(graphics)&oldid=800949469. [Accessed: 18-Nov-2017].
[37] “Rasterization: a practical implementation - Scratchapixel.” [Online]. Available: https://www.scratchapixel.com//lessons/3d-basic-rendering/rasterization-practical-implementation. [Accessed: 18-Nov-2017].
[38] “Oculus Rift review - TechRadar.” [Online]. Available: http://www.techradar.com/reviews/gaming/gaming-accessories/oculus-rift-1123963/review. [Accessed: 02-Nov-2017].
[39] “Introduction to virtual reality - Technobyte.” [Online]. Available: https://www.technobyte.org/introduction-virtual-reality/. [Accessed: 25-Nov-2017].
[40] C. Tulvan, R. Mekuria, Z. Li, and S. Laserre, “Use cases for point cloud compression (PCC),” Doc. ISO/IEC JTC1/SC29/WG11/N16331, Geneva, Switzerland, Jun. 2016.
[41] S. Orts-Escolano *et al.*, “Holoportation: virtual 3D teleportation in real-time,” UIST ’16, Tokyo, Japan, Oct. 2016.
[42] “Microsoft introduces the world to ‘holoportation’ - TechRadar.” [Online]. Available: http://www.techradar.com/news/wearables/microsoft-introduces-the-world-to-holoportation-1317809. [Accessed: 09-Oct-2017].
[43] “Intel® freeD™ Technology.” [Online]. Available: https://www.intel.com/content/www/us/en/sports/technology/intel-freed-360-replay-technology.html. [Accessed: 09-Oct-2017].
[44] “Shipping Galleries | Science Museum.” [Online]. Available: http://scanlabprojects.co.uk/work/shipping-galleries/. [Accessed: 09-Oct-2017].
[45] “Mobile Mapping System - Mitsubishi Electric.” [Online]. Available: http://www.mitsubishielectric.com/bu/mms/. [Accessed: 09-Oct-2017].
[46] “Mobile Mapping System features - Mitsubishi Electric.” [Online]. Available: http://www.mitsubishielectric.com/bu/mms/features/. [Accessed: 09-Oct-2017].

[47] R. Mekuria, Z. Li, C. Tulvan, and P. Chou, “Evaluation criteria for PCC (point cloud compression),” Doc. ISO/IEC JTC1/SC29/WG11/N16332, Geneva, Switzerland, Feb. 2016.
[48] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, “Geometric distortion metrics for point cloud compression,” ICIP’2017, Beijing, China, Sep. 2017.
[49] A. Javaheri, C. Brites, F. Pereira, and J. Ascenso, “Subjective and objective quality evaluation of compressed point clouds,” MMSP’2017, Luton, United Kingdom, Oct. 2017.
[50] E. Alexiou and T. Ebrahimi, “On subjective and objective quality evaluation of point cloud geometry,” QoMEX’2017, Erfurt, Germany, May 2017.
[51] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, “Real-time compression of point cloud streams,” ICRA’2012, Minnesota, USA, May 2012.
[52] R. Mekuria, K. Blom, and P. Cesar, “Design, implementation, and evaluation of a point cloud codec for tele-immersive video,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, Apr. 2017.
[53] “Tree traversal - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Tree_traversal. [Accessed: 11-Dec-2017].
[54] C. Zhang, D. Florencio, and C. Loop, “Point cloud attribute compression with graph transform,” ICIP’2014, Paris, France, Oct. 2014.
[55] R. A. Cohen, D. Tian, and A. Vetro, “Attribute compression for sparse point clouds using graph transforms,” ICIP’2016, Arizona, USA, Sep. 2016.
[56] S. Veronica, T. Christian, G. Adrian, and P. Marius, “Reconstruction platform and datasets for 3D point cloud compression,” ISO/IEC JTC1/SC29/WG11/M36523, Warsaw, Poland, Jun. 2015.
[57] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, Mar. 2011.
[58] M. Dou, J. Taylor, H. Fuchs, A. Fitzgibbon, and S. Izadi, “3D scanning deformable objects with a single RGBD sensor,” CVPR’2015, Jun. 2015.
[59] R. L. de Queiroz and P. A. Chou, “Compression of 3D point clouds using a region-adaptive hierarchical transform,” *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.
[60] “P.913 : Methods for the subjective assessment of video quality, audio quality and audiovisual quality of Internet video and distribution quality television in any environment.” [Online]. Available: https://www.itu.int/rec/T-REC-P.913/en. [Accessed: 02-Jan-2018].
[61] K. Mammou, P. A. Chou, and E. S. Jang, “PCC Core Experiment 0.1 on convergence between TMC1 and TMC3,” ISO/IEC JTC1/SC29/WG11 N17352, Gwangju, Korea, Jan. 2018.
[62] K. Mammou, P. A. Chou, D. Flynn, and M. Krivokuća, “PCC Test Model Category 13 v3,” ISO/IEC JTC1/SC29/WG11 N17762, Ljubljana, Slovenia, Jul. 2018.
[63] P. A. Chou, O. Nakagami, and E. S. Jang, “Point Cloud Compression Test Model for Category 1 v1,” ISO/IEC JTC1/SC29/WG11 N17340, Gwagju, Korea, Jan. 2018.
[64] V. Zakharchenko, “Algorithm description of mpeg-pcc-tmc2,” ISO/IEC JTC1/SC29/WG11 MPEG2018/N17767, Ljubljana, Slovenia, Jul. 2018.
[65] “Surface patch -The Free Dictionary.” [Online]. Available: https://encyclopedia2.thefreedictionary.com/surface+patch. [Accessed: 22-Sep-2018].
[66] “Principal components analysis - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Principal_component_analysis. [Accessed: 23-Sep-2018].
[67] “Rosetta Code - K-means++ Clustering.” [Online]. Available: http://rosettacode.org/wiki/K-means%2B%2B_clustering. [Accessed: 25-Sep-2018].
[68] T. Golla and R. Klein, “Real-time Point Cloud Compression,” IROS’2015, Hamburg, Germany, Oct. 2015.
[69] S. Schwarz, G. Martin-Cocher, D. Flynn, and M. Budagavi, “Common test conditions for point cloud compression,” ISO/IEC JTC1/SC29/WG11 N17766, Ljubljana, Slovenia, Jul. 2018.
[70] D. Flynn, “PCC TMC13v3 performance evaluation and anchor results,” ISO/IEC JCTC1/SC29/WG11 W17768, Ljubljana, Slovenia, Jul. 2018.
[71] S. Winkler, C. J. van den Branden Lambrecht, and M. Kunt, “Vision and Video: Models and Application,” in *Vision Models and Applications to Image and Video Processing*, 1st ed., Boston, MA: Springer, pp. 201–229, 2001.
[72] “MeshLab.” [Online]. Available: http://www.meshlab.net/. [Accessed: 10-Oct-2018].