Dynamic 3D Point Cloud Compression

Miguel Branco Roque Nazaré Ferreira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

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

Jury:

Chairperson: Prof. José Eduardo Charters Ribeiro da Cunha Sanguino

Supervisor: Prof. João Miguel Duarte Ascenso

Member of the Committee: Prof. Nuno Miguel Morais Rodrigues

November, 2017

Acknowledgements

To the most important ones, my Mother, my Father, and my Brother, I want to give them the biggest thank you, as they have accompanied me since the very beginning and they always pushed me further while wishing me the best. They are the reason I am reaching this academic stage.

I would also like to thank Professor João Ascenso, for his exemplar guidance, namely his dedication, availability, and counselling that were decisive in this work. He taught me many things outside the scope of this work, such as methodology and organization skills, and he also provided me his exceptional technical and conceptual knowledge.

I am also equally thankful to Professor Fernando Pereira and Professor Catarina Brites, as they were also equally important with their guidance. Their profound knowledge alongside with their devotion help me uncountable times. This work would probably not be finished without the motivation that each of the three professors continuously gave me. One final thanks, for their patience.

Finally, I would like to thank all my friends, with a special thanks to Hugo Martins, João Miguel Silva, João Baúto, André Ambrósio, and Guilherme Lima, as they endured with me all the good and bad moments, giving me an important support in the final stage of this course.

Resumo

As point clouds são um modelo de representação obtido através da amostragem da função plenóptica através de um conjunto de pontos num sistema de coordenadas 3D, com atributos opcionais associados, e.g. as cores, representando objetos e superfícies. Os avanços recentes na captura e no display de informação visual permitem experiências mais imersivas e realísticas para o utilizador, como a telepresença imersiva 3D e transmissão de desportos 3D. No entanto, uma tarefa desafiante consiste em representar e codificar a informação visual necessária numa maneira eficiente, para facilitar o armazenamento, a transmissão e o processamento dos respectivos dados. Para point clouds dinâmicas, i.e. point clouds que se alteram ao longo do tempo, o desenvolvimento de soluções de codificação está ainda numa fase muito inicial, com bastante espaço para melhoramento. Recentemente, o grupo MPEG lançou um projecto de padronização nesta area.

Considerando as poucas soluções de codificação de point clouds dinâmicas existentes na literatura, esta Dissertação propõe uma solução nova que explora tanto as correlações espaciais como temporais numa point cloud dinâmica. A solução proposta estima movimento entre as tramas actual e passada usando descriptores de ponto, estabelecendo correspondências entre os pontos das duas tramas, modelando o movimento com uma transformada geométrica (Homografia) e produzindo uma trama com compensação de movimento com um passo de “warping”. Depois, as diferenças entre as tramas compensada e actual são codificadas e transmitidas, como no codec da Point Cloud Library (PCL), baseado numa octree com duplo buffer.

A avaliação mostra que a trama compensada com compensação de movimento tem uma melhor predição que a trama anterior, quando existe muito movimento entre duas tramas sucessivas. No entanto, a avaliação do desempenho RD mostra que esta melhoria não é suficiente para que a solução proposta consiga superar o desempenho da solução original do codec da PCL, que utiliza a trama passada codificada.

Palavras-chave: *Point cloud,* Função plenóptica*,* estimação de movimento, homografia, compressão de *point clouds* dinâmicas.

Abstract

Point clouds are a representation model obtained from the sampling of the plenoptic function through a set of points in a 3D coordinate system, with optional associated attributes, e.g. the colours, representing objects or surfaces. Recent advances in capturing and displaying of visual information are enabling more immersive and realistic user experiences, such as 3D immersive telepresence and 3D sports broadcasting. Thus, a challenging task consists in representing and coding the necessary visual information in an efficient way, to facilitate the storage, transmission and processing of such data. For dynamic point clouds, i.e. point clouds changing through time, the development of coding solutions is still at a very initial phase, with large room for improvements. Recently, the MPEG group has launched a standardization project in this field.

Considering the few dynamic point cloud coding solutions available in the literature, this Thesis proposes a novel solution to exploit both temporal and spatial correlations in a dynamic point cloud. The proposed solution estimates motion between the current and previous frames by using point descriptors, establishing correspondences between points in the two frames, modelling the motion with a geometric transform (homography) and then producing a motion compensated frame with some warping step. Then, the differences between the motion compensated and target frames are encoded and transmitted, as in the octree-based double buffering Point Cloud Library (PCL) codec.

The evaluation shows that the motion compensated frame has a better prediction than the previous frame, when there is high motion between two successive frames. However, the RD performance assessment shows that this improvement is not enough for the proposed solution to outperform in rate-distortion the original PCL codec, which uses the past coded frame.

Keywords: Point cloud, plenoptic function, motion estimation, homography, dynamic point cloud compression.

Table of Contents

List of Figures

List of Tables

Acronyms

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
3DSC	3D Shape Context
BPP	Bytes per point
BPV	Bits per vertex
C2C	Cloud to Cloud
C2P	Cloud to Plane
CFP	Call for Proposals
CPC	Constrained Planar Cuts
CVFH	Clustered Viewpoint Feature Histogram
DCT	Discrete Cosine Transform
DPCM	Differential Pulse-code Modulation
ESF	Ensemble of Shape Functions
FPFH	Fast Point Feature Histogram
GFT	Graph Fourier Transform
GIS	Geographic Information Systems
GOP	Group of Pictures
HB-PCC	Homography Based Point Cloud Codec
HMD	Head Mounted Device
ICP	Iterative Closest Point
JPEG	Joint Photographic Experts Group
LCCP	Locally Convex Connected Patches
LED	Light Emitting Diode
LIDAR	Light Detection and Ranging
LoD	Level of Detail

MPEG	Moving Pictures Experts Group
PCE	Principal Curvatures Estimation
PCL	Point Cloud Library
PFH	Point Feature Histogram
PPF	Point Par Feature
PSNR	Peak Signal to Noise Ratio
QoE	Quality of Experience
RADAR	Radio Detection and Ranging
RANSAC	Random Sample Consensus
RD	Rate-distortion
RIFT	Rotation Invariant Feature Transform
RLGR	Run-Length and Golomb-Rice
RMSE	Root Mean Square Error
SAR	Synthetic Aperture Radar
SGW	Spectral Graph Wavelets
SHOT	Signature of Histogram of Orientations
SPFH	Simplified Point Feature Histogram
SQNR	Signal-to-quantization Noise Ratio
SVD	Singular Value Decomposition
USC	Unique Shape Context
VCCS	Voxel Cloud Connectivity Segmentation
VFH	Viewpoint Feature Histogram

Chapter 1

1. Introduction

This first chapter is an introduction to the topic addressed in this Thesis: dynamic point cloud compression. It presents the context, motivation and objectives of the Thesis before describing its structure.

1.1. Context and Motivation

Technology related to visual information has been growing exponentially in the last few decades, namely targeting acquisition, representation, coding, and display of visual information. These technologies allow the creation of new applications and services, and to increase the quality of experience of those already available. Visual information technologies are used in the Human Society to expand the functionalities and performance of areas such as entertainment, communications, education, security, and many others. In some applications, the real world can be visualized with additional elements, not available in the original acquired content, also known as augmented reality. This may be achieved with Head-Mounted Displays (HMD), e.g. Microsoft HoloLens [1]. To make possible the deployment of these applications, powerful 3 Dimensional (3D) representation formats are essential to provide immersive experiences to the users.

One well know approach to mathematically model the visual information in the real world is the so-called plenoptic function which represents the amount of light flowing in every direction through every point in space. As the plenoptic function can perfectly represent the world in terms of visual information, the amount of data that it uses for storage, transmission, and processing is massive, thus becoming rather impractical. More recently, more constrained but yet rich ways to sample the visual information represented with the plenoptic function were proposed: light fields and point clouds. The point cloud is a 3D representation format able to represent the 3D real world as a set of points with some associated attributes; this format provides a simple but efficient way to provide a good user experience for some specific applications, e.g. 3D immersive telepresence, or 3D broadcasting. Point clouds may represent scenes or objects, such as the Colosseum shown in Figure 1 (left) by using sets of points in a 3D coordinate system (x,y,z). Each point can have one or more attributes associated but the colour is the most important and often used attribute; however, other attributes can be used to improve the richness of the visual content represented by the point cloud, e.g. normals or other material properties. In Figure 1 (right) it is shown an example of a high resolution point cloud, which may be perceived with more quality in comparison to the simpler low quality point cloud shown in Figure 1 (left). For the case of

moving objects, the sequence of point clouds representing the moving object in different instants of time is called a time-varying or dynamic point cloud.

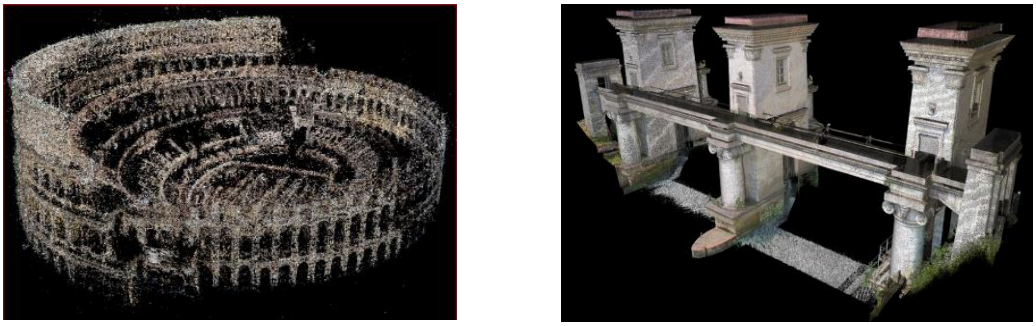

Figure 1: Left: Sparse point cloud of the Colosseum [2]. Right: Coloured point cloud of a bridge [3].

Thus, point clouds can be used to represent volumetric data, with millions of points, namely for applications such as Geographic Information Systems (GIS), as shown in Figure 2 (left), and cultural heritage. They can also be used in streaming applications, which requires the transmission and rendering of the acquired data in real-time. Point cloud streaming may be associated to bi-directional communications, e.g. the tele-presence system shown in Figure 2 (right), or broadcasting, e.g. 3D sports broadcasting. These last applications require the processing of the 3D data at a very fast rate, which fits rather well the characteristics of point clouds, as they may be rendered much faster than other available 3D representation formats, such as meshes and light fields.

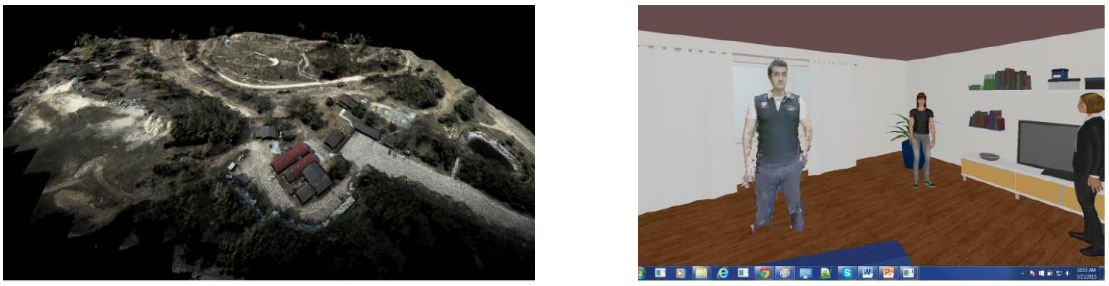

Figure 2: Left: Point cloud for a geographical region [4]. Right: Virtual room containing point cloud avatars, used for real-time communications [5].

The transmission of the point cloud data, made in real-time and containing millions of points, is a serious challenge for bandwidth limited networks, thus demanding the efficient compression of point clouds. To address this problem, notably for the case of dynamic point clouds, a few solutions have been proposed lately in the literature; these solutions are reviewed in this Thesis, and serve as an inspiration for the solution that was developed.

The practical relevance of developing point cloud coding solutions has been also acknowledged by MPEG which has recently launched a Call for Proposals (CFP) for point cloud compression technology [6]; the MPEG Call for Proposals is asking for coding solutions that can fulfil an identified list of requirements, both for static and dynamic point clouds. The main objective is to standardize a point cloud compression solution to assure interoperability between different products and applications; this

effort clearly shows the increasing importance of the point cloud representation format. Finally, also JPEG is addressing point cloud coding in the context of the JPEG Pleno project [7], thus reinforcing the current research and industrial interest in static and dynamic point cloud representations.

1.2. Objectives

In this context, the main objective of this work is to design, implement and assess an efficient dynamic point cloud coding solution that exploits temporal correlation based on the best techniques available at the literature. This objective is also being pursued by the MPEG standardization group in its 3D graphics ad-hoc group. To reach this objective, the work developed under this Thesis' context was organized in the following tasks:

- Review of relevant dynamic point cloud coding solutions available in the literature. However, important concepts about point clouds will be presented before.
- Novel coding solution to compress dynamic point clouds is proposed. The codec designed is able to exploit both the spatial and temporal correlations of the point cloud data over time, considering only the geometry of the point cloud.
- Assessment of the performance of the developed point cloud codec using relevant and appropriate test conditions, benchmarks and methodologies.

1.3. Thesis Structure

This Thesis is structured in five chapters, including this first Introductory chapter, where the context and motivation of the Thesis is presented as well as the objective and structure of this Thesis.

Chapter 2 reviews the state-of-the-art on point cloud representation, notably by introducing some basic concepts, the overall point cloud end-to-end architecture, the main applications and requirements, the most popular acquisition systems, the objective quality metrics and, finally, some key dynamic point cloud coding solutions.

Chapter 3 describes in detail the proposed point cloud compression solution, starting by the codec architecture, walkthrough and a detailed description of each module.

In Chapter 4, the proposed point cloud coding solution is assessed in comparison with the state-of-the-art PCL based codec. First, the test conditions are defined, followed by an evaluation of the accuracy of the motion compensated point cloud which allows to evaluate what is the quality of the predicted point cloud over time, and the RD performance of the proposed point cloud coding solution.

Finally, in Chapter 5, the conclusions and future work are presented.

Chapter 2

2. Point Clouds: Basics and Main Coding Solutions

The main objective of this chapter is to review popular point cloud compression schemes, relevant applications, systems and processing techniques as well as the basic 3D visual concepts. Thus, it is introduced first the plenoptic function and all the representation models associated, giving more emphasis to point clouds. Then it is presented the point cloud system architecture and its applications and requirements, in order to contextualize the importance of point clouds in key applications. Next it will be covered two different acquisition types and the objective quality metrics used to assess a point cloud solution performance. Finally, the last subtopic will present the main coding three coding solutions for point cloud compression.

2.1. Basic Concepts

3D representation formats are much closer to the reality, improving the user Quality of Experience (QoE) by providing a more immersive and powerful 3D experience. Nowadays, there is a promising trend on the development of 3D displays and technologies, which seek to achieved high 3D QoE for immersive visual applications, such as augmented reality. In this section the 3D representation formats are introduced and clarified, which is essential to understand how this type of data can be coded. The following representation models can describe the visual information of our world in 3D:

- **Plenoptic function:** This model attempts to represent all the visual information present in any location without discarding any information. Thus, it is necessary to characterize the behaviour of the electromagnetic field in the visible spectre for every point of the space [8]. The human visual system, which is responsible for our vision, is able to capture the world that surrounds us as consequence of the electromagnetic waves that are emitted from every object in the world in every direction. The human eyes work as sensors of light, just like cameras, and capture the convergence of light coming from every direction in a single point in space. Thus, to record that information, it is necessary to account for every point of space (x,y,z) the sum of all wave fronts, described by a sum of random-phase sinusoids (characterized by λ), being each wave front defined by the azimuth and orientation variables (θ, ϕ). Of course each waves' energy varies in time, so it is also necessary to take into consideration the time (t). With this seven variables information, comes the plenoptic function, $P(x,y,z,\theta,\phi,\lambda,t)$, as shown in Figure 3.

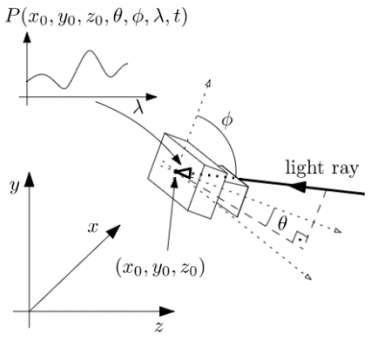

Figure 3: Representation of the plenoptic function [8].

However, this corresponds to a lot of information to store and some basic simplifications can and must be made (check [9] for details). Instead of one function using the wavelength in all the visible spectrum, three plenoptic functions can be used, each representing signals with the Red, Green and Blue wavelengths. This simplification come from the fact that our eyes can separate light into the various colours defined as a combination of red, green and blue. The new three plenoptic functions are represented respectively by P_R, P_G and P_B and have one less dimension, the wavelength λ, than the original function.

- **Multiview video:** The simplest case of plenoptic reduction is multiview video, represented in Figure 4, where several cameras are used to obtain different views of a fixed scene simultaneously. These cameras can be displaced in numerous arrangements, like 1 Dimensional (1D) linear or circular arrays, and can contain either equally or irregularly spaced cameras. Multiview video when reduced to two views, corresponds to stereo, where two cameras acquire a video from two different positions. Often, the plenoptic function is reduced to 3 or 4 dimensions, depending if time is considered, which are, the camera position in the array and the 2 Dimensional (2D) pixel position in each camera, i.e. the position where the light ray hits the sensor.

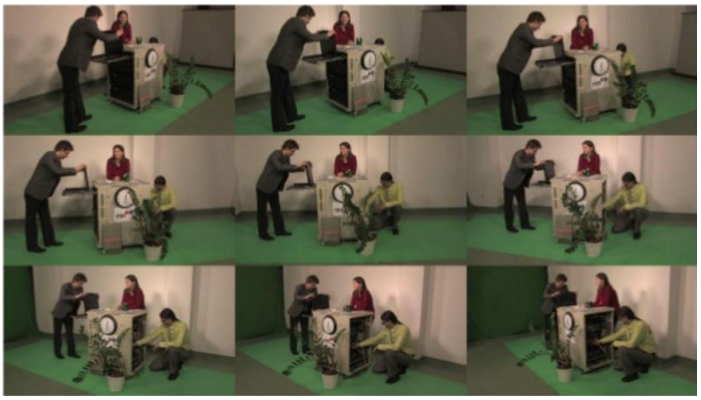

Figure 4: Multiview video example [10].

- **Light Field:** This representation model is a restriction of the plenoptic function to four, or five dimensions if time is considered [11]. In the literature there are several definitions of light field, however, here it is defined as the reduction of plenoptic function to five dimensions, shown in Figure 5 (time not included), corresponding to two pairs of variables: 1) the position inside the plane of the sensor (x,y), 2) the position inside the plane of the lens (u,v), and time (t). So, a

camera or a 2D array of cameras can capture the spatio-temporal information of light fields by these two parallel planes indicating the spatial and angular coordinates of a ray. Naturally, the light field representation can be obtained from a single camera with an array of micro-lenses, obtaining an array of micro-images where each one acquires light obtained from different directions. However, a light field can be also obtained from an array of cameras with several possible arrangements and structures. One of the most attractive applications of light field imaging is the possibility of re-focusing the image after its being acquired, unlike conventional 2D images. The light field acquires a richer representation of the real world in comparison to traditional methods, but that comes at the cost of more complexity in the process of acquisition and rendering.

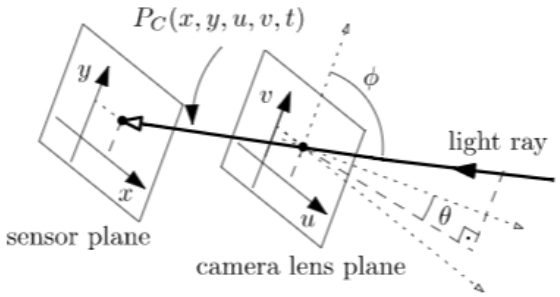

Figure 5: Light field inside a camera [8].

- **Light field plus depth:** Explicit depth information can be added to the light field model and thus adding more information about the geometry of the recorded scene, making the acquired 3D representation model more robust and rich. Usually, depth sensors are used to measure the distance between some sensor and all or some of the objects (and background) of the visual scene. The presence of depth information allows the possibility of synthesizing new views with improved accuracy, enhancing the quality of the point cloud created using the collected data.
- **Point cloud:** A point cloud consists in a set of points in a 3D coordinate system, where each point has one or more attributes associated, for instance colour or normal vectors. Figure 6 (left) shows a coloured point cloud, and Figure 6 (right) shows the normal vectors associated to two points. Although it is a rather complex subject, each point can have its colour associated to the direction where it is being viewed, like in a light field. Point clouds usually represent external surfaces of artificial or real objects. Two types of point clouds are possible: in the first one the data is acquired by sensors and the point cloud is consequently generated; in the second, both data and point cloud are computationally generated. The following concepts are associated with point clouds:
  - Depending if point clouds represent static or moving objects, they can be **static** or **dynamic**. This last case corresponds to a sequence of point clouds acquired in different instants of time.
  - **Organised** or **structured** point clouds can be stored into an organized structure such as a grid, where the data is split into rows and columns [12]. Related to this concept, is the **projectable** point cloud where the index of a point in the organized point cloud and the 3D coordinates of each point are correlated according to a pinhole camera model. In an

unorganized point cloud, its data is in the form of a 1D array, instead of a 2D array like the previous case.

- The point cloud representation model can be further divided into **progressive** and **non-progressive** [13]. In the progressive representation, the point cloud is organised into layers with increasing Level of Detail (LoD). The first layer is coarse, which means that the number of points is usually low. With the remaining refinement layers, it is possible to obtain more information to increase the number of points, and thus the quality of the point cloud representation of a 3D surface. In the non-progressive mode, there is only a unique layer of representation with a certain fixed quality, instead of several layers. In this last case, the point cloud can only be visualized when all the point cloud information is available.
- Point cloud compression can be **lossless** or **lossy**. In the first one, the decoded point cloud is mathematically equal to the original point cloud, meaning the number of points and attributes and their accuracy or precision doesn't change. In the lossy case, the decoded point cloud has some errors when compared to the initial point cloud. Typically, it may have less points or attributes or their precision or accuracy can be lower. The lossy case allows to increase the compression ratio at the expense of lower quality.
- Point clouds have attributes associated to each one of the points, which are critical to have a realistic rendering of the 3D visual scene. The most relevant attributes are the **colour** and **normal vectors**. The first one is associated to each 3D point and defines the corresponding colour. And like previously mentioned, it is possible to associate to a point several colours, dependent to the perspective of view. The normal vector is an attribute associated to each point that defines the perpendicular vector tangent to the surface surrounding that point.

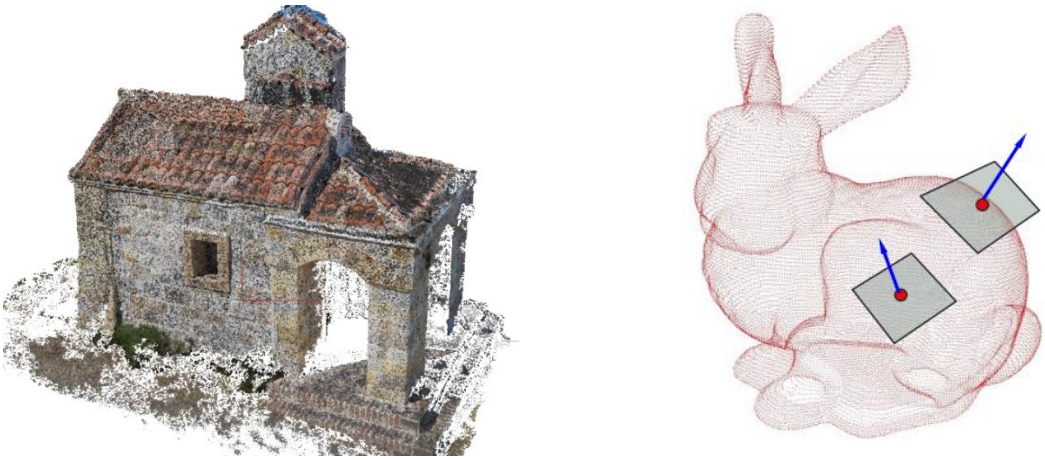

Figure 6: Left: Point cloud with colour [14]; Right: Normal vectors in the bunny point cloud [15].

- **Mesh with texture**: A 3D mesh is a set of vertices which are connected by edges and thus forming a set of 3D surfaces to represent real world objects. The simplest mesh uses triangular surfaces, although surfaces can also be a polygonal with a certain number of edges. They can be obtained from a point cloud by applying triangulation techniques to connect the points of a point cloud usually with the help of the normal attribute of each point. Usually, it maybe also necessary to apply some filtering to remove noise and outliers that are due to the way that point

clouds can be obtained. To render a 3D mesh there are several techniques available most often requiring the knowledge of the sources of illumination and colouring the surfaces using the colour of the vertices.

- **Holography:** Holography is the science or practice of creating a hologram, which is one of the most advanced forms of 3D visual representation [16]. The technique to record a hologram consists in using a laser that is split into two laser beams (object and reference beams) where both reach a record medium. The object beam is diffracted on the object to be recorded and the reference beam is reflected on a mirror. Both beams intercept the record medium causing interference which is recorded. It isn't required glasses or other intermediate optics to see a hologram since a hologram records everything that is perceived by human eyes (depth, texture, shape, etc.), which makes holography a very attractive, although complex, representation model.

2.2. Point Cloud-based Representation System Architecture and Walkthrough

The goal of a point cloud-based representation system, like other similar imaging systems, is to efficiently represent a scene or object acquired from the real world (or even synthetically created) to be visualized in a display in the context of a specific application. Figure 7 shows a simplified architecture of a point cloud-based representation system.

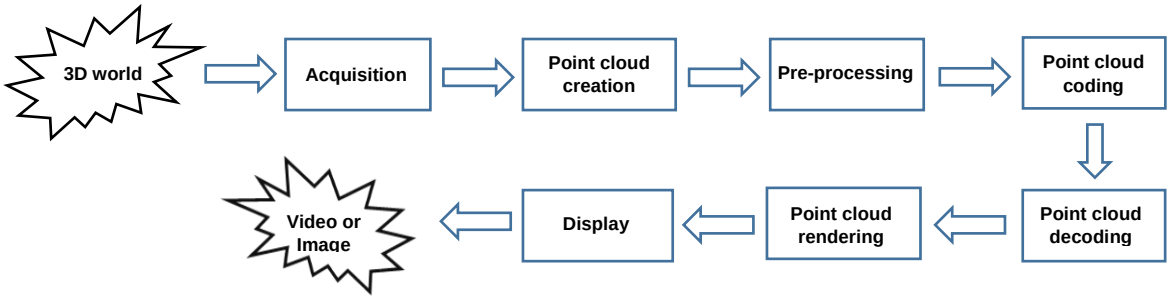

Figure 7: Point cloud-based representation system architecture.

The main modules in a point cloud-based representation system are:

- **Acquisition:** In this first step, 3D world visual data is acquired by appropriately sampling the plenoptic function, typically with reduced sparsity. The acquisition consists in capturing the desired real world scenery visual data by using appropriate devices, notably multiple regular cameras according to some spatial arrangement which measure texture data, depth scanners or a combination of both. If only texture cameras are used, the number of cameras and corresponding positions will take a vital role and determine the final quality as the more cameras are available the more data will be used to create the point cloud [17]; if depth scanners are also used to directly acquire some geometry related data, the number of texture cameras may be reduced without compromising the quality of the point cloud.
- **Point cloud creation:** Point clouds can be either computationally generated or created with acquired texture and/or depth samples as described in the previous step. In the later case, two main cases can be distinguished as already mentioned above: i) only texture samples are

acquired, implying that the point cloud will be created by means of view matching and triangulation, eventually at the expense of less accurate point cloud representations if the number of cameras is not high enough; ii) texture and depth samples are acquired, implying that direct geometry data is available, notably the distance from the scene/object points to the sensor; together with the texture, this geometry data allows the creation of the point cloud.

- **Pre-processing:** This module may include several processing operations, e.g. targeting the creation of a less noisy or denser point cloud. As the samples acquired with the texture/depth sensors include some noise, it may be necessary to filter and then smoothing them. Also the point cloud may include points which are very close or coincident, thus rather redundant, which may justify the reduction the total number of cloud points. The reduction of points can also be done, for instance, if the application scenario for the point cloud requires low resolution, which does not justify the use of a large number of points.
- **Point cloud coding:** If the enormous volume of data that may represent a created point cloud is to be transmitted over a physical channel or stored in some device, it is essential to code it to reduce the representation rate. Data compression is required because most of the channels and storage devices have a limited bitrate. Coding consists in applying some algorithms which by exploiting the redundancy, and eventually perceptual irrelevance, reduce the coding rate, eventually at the expense of quality and complexity, therefore reducing the required transmission bitrate. For instance, point cloud data may be merged and ordered, segmenting the point cloud in several smaller ones, to help the transmission and storage. When interoperability is an issue, the coding algorithms should follow international standards. MPEG is currently developing a standard for lossless and lossy coding of static and dynamic point clouds in order to provide interoperability.
- **Point cloud decoding:** After the transmission of the data over a channel or storage in same device, the second half of the representation chain will target the reconstruction of the visual scene in some display. At the decoding module, the previously coded and transmitted data goes through a decoding process where a perfect or close replica of the originally coded data is obtained depending if a lossless or lossy codec is used. While coding can be done offline, with the exception of real-time applications like telepresence, decoding is usually a real-time process. After decoding, some post-processing operations may be applied to the point cloud resultant from the decoding process, such has adding new points to obtain a smoother point cloud; this process is usually called refinement.
- **Point cloud rendering:** Rendering is the process of creating some data for user visualization using the display at hand based on the decoded point cloud data. If a regular 2D display is available, regular 2D frames have to be rendered from the point cloud model. The rendering process may have to occur in real-time, e.g. for telepresence applications, or may be performed off-line to get highest quality, e.g. for geographic information systems.
- **Display:** Finally, the rendered data is displayed for user consumption. Depending on the application, the display can present either a 2D or 3D video. In the case of dynamic point clouds, the user navigation process may be fixed or free depending if the user can freely choose the

viewpoint or not using some interaction interface. Several types of display can be used, such as 2D displays, stereo and auto-stereoscopic displays and virtual reality head mounted displays.

In the following, the coding module will deserve more attention as this technology is the main topic of this report.

2.3. Applications and Requirements

As for all 3D representation models, point clouds are better targeted to some specific applications and conditions, for instance applications working with a lot of volumetric data and involving user interaction. They are also commonly used for inspection purposes, due to their simple and fast rendering, and real-time applications like telepresence. To make these applications effective, appropriate point cloud coding solutions have to be developed while fulfilling the needs of relevant applications, leading to several requirements that will be identified and discussed by the end of this section. Based on the MPEG relevant documents [18], several groups of applications can be identified:

1. 3D immersive telepresence

This type of applications usually involves real-time communication between two geographically distant sites which should be as realistic and transparent as possible. Often these applications involve the usage of virtual reality technology, where virtual objects represented by means of point clouds are placed in the real world and displayed on a TV or HMD. Figure 8 shows an example where a little girl is being recorded in real-time by a set of cameras, as shown in the top left of the figure, and the corresponding coded point cloud data is being transmitted, and after rendered and finally displayed at the user headset, in this case a Microsoft HoloLens HMD [1]. At the bottom left of the figure, the actual footage on the HMD is shown. As this type of application normally includes (bidirectional) conversation, it has some specific demands, for instance real-time encoding and decoding, lossy compression and error resilience.

Figure 8: Telepresence environment [19] using a HoloLens HMD [1].

2. 3D broadcasting

3D broadcasting of sports like basketball and baseball may use free viewpoint video, as shown in Figure 9. This is another type of application for point clouds where users can interact with the rendered video to change the angle of view. Replay Technology [20] is an example of such applications. In this case, interoperability is usually important, this means the coding and file format must follow standard

solutions in order the devices from multiple manufacturers may still interoperate. The requirements must consider low delay encoding and decoding as well as colour attributes coding, with 8-12 bits per component. Unlike telepresence, 3D broadcasting of sports usually contains clusters of point clouds, as they are usually used to represent more than one player at a time.

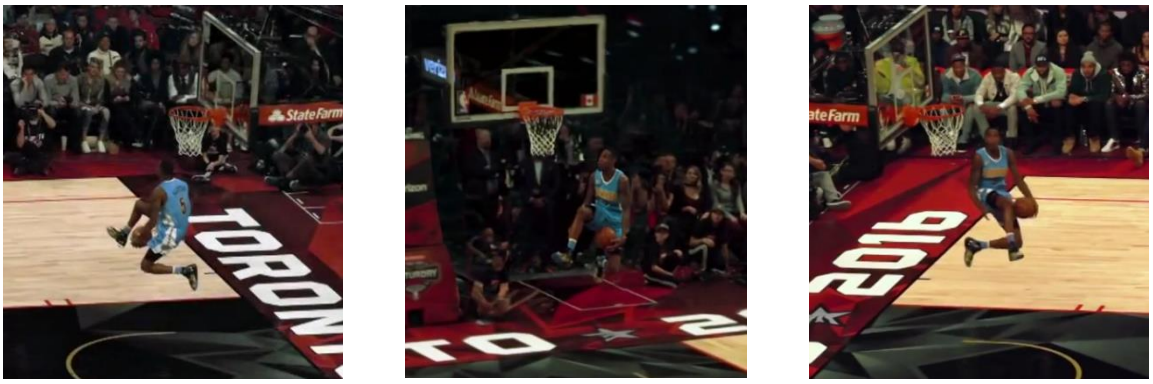

Figure 9: Different perspectives of a basketball player at the same time instant [20].

3. Geographic information systems

Another important application area for point clouds is geographic information systems, where some software application shall render, analyse and control a geographic area, represented using a typically large point cloud, see Figure 10. To obtain high precision and resolution, some geographic information systems use Light Detection and Ranging (LIDAR), and Synthetic Aperture Radar (SAR) sensor technologies. Their measurements may provide a very dense set of points, this means a dense point cloud, which often requires a remote server to store the data. To limit the servers' capacity, there is a need to efficiently represent the large point clouds by using appropriate, ideally standard, coding solutions. These applications typically consider requirements such as region selectivity, lossless compression and progressive decoding.

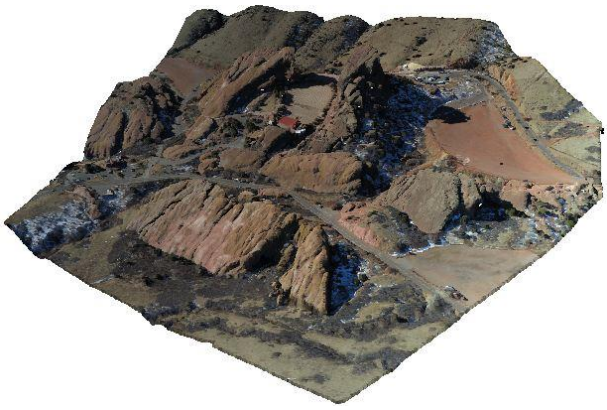

Figure 10: Point cloud of Red Rocks, Colorado [21].

4. Cultural heritage

Another application for point clouds is the representation and reconstruction of objects from cultural heritage exhibitions and collections to be interactively visualized by the users. Figure 11 shows an example of a display for this type of applications, where the user can freely rotate the object to get new perspectives. These objects are scanned with 3D laser scanning technology, like LIDAR or structured light [22]. Relevant requirements are progressive coding, lossless coding and colour and material attribute coding, e.g. to provide streaming to a wide public audience.

Figure 11: Left: Façade of a cathedral. Right: Column (based on [23]).

5. The requirements

The applications listed above allow identifying a set of requirements, which MPEG has been listing [18]:

- **Lossless compression:** As this requirement is relevant for some geographic information systems, the point cloud codec shall allow performing lossless coding of the point cloud, including both geometry and attributes. This means that the decoded point cloud is mathematically equal, notably with the same number of points and attributes, to the original point cloud (before coding).
- **Lossy compression:** As some transmission channels cannot afford the high bitrate that is needed to losslessly represent a point cloud data, the point cloud codec shall be able to perform lossy coding to reduce the resulting channel bitrate, eventually at the cost of some quality degradation in the decoded point cloud.
- **Progressive compression:** The point cloud codec shall allow the progressive coding of the point cloud, which means a coding/decoding process based on several layers where each layer may be associated to a growing number of points, quality, etc. naturally at the cost of growing rate.
- **Error resilience:** To avoid propagation of errors and the corresponding negative subjective impact, the point cloud codec should be as error resilient as possible, e.g. limiting the propagation of errors in time.
- **Low complexity:** Some applications are to be made available in computational and battery scarse conditions, and thus low encoding and decoding complexity are critical requirements.
- **Low delay coding:** Some applications, such as telepresence, have to work in real-time and thus the delay corresponding to coding operations, this means the delay independent of the available computational resources, should be limited.
- **Colour attribute coding:** The point cloud codec shall include the capability to code colour attributes.
- **Material related attributes coding:** The point cloud codec shall include the capability to code additional attributes, notably material related attributes, e.g. to support efficient rendering.
- **View dependence:** The point cloud codec shall allow to code point cloud attributes that vary with the visualization angle, this means being view dependent. As example, although not common, the colour attributes for each point may change with the incident direction of the viewpoint.

- **Selectivity:** The point cloud codec shall allow easy (random) access to the cloud points associated to a specific region (subset of the point cloud).

As the full set of requirements is quite demanding, some trade-offs come naturally, e.g. a point cloud lossless codec may not have a great performance in rate reduction while a lossy codec may increase its compression if the complexity and delay requirements are less demanding. The actual point cloud coding solutions should be flexible enough to accommodate various requirements trade-offs, as there are no codecs that are able to perform the best in all requirements simultaneously. Naturally, some specific coding solutions may achieve very good performances for some specific requirements, making them ideal for the applications where these requirements are critical.

2.4. Point Cloud Acquisition Systems

This section will review in some detail the various ways to acquire the necessary data for point cloud creation. As mentioned before, this acquisition may happen in two main different ways: using only texture cameras, like when acquiring a light field or using texture cameras plus depth sensors, which is a more direct acquisition as geometry information is explicitly acquired. As shown in Figure 7, the acquisition is the first step of the point cloud-based system architecture and probably one of the most important as it has a big impact on the remaining architectural modules. The cameras setting, linear or circular, in a 1D or 2D array, along with the density of cameras, will definitely determine the QoE provided by the finally rendered views [17]. Naturally, this involves a quality-cost trade-off, where the cost is not only related to the acquisition equipment but also to processing itself. If the rendering requirements ask for a high quality, like in free navigation and broadcasting, the equipment and processing cost may increase as more precise data and more sophisticated processing may be required.

2.4.1. Texture

This type of acquisition paradigm only involves texture cameras, this means cameras measuring colour information as RGB or YUV signals, luminance and chrominance data. For a point cloud-based acquisition system, the acquired textures are used to obtain the point clouds with triangulation techniques and view matching, creating the 3D point coordinates. These techniques require visual information about the acquired object from more than one perspective, so a multi-camera setup is required. However, the acquisition of a static point cloud can be done with a single camera acquiring different perspectives at different times. For the dynamic point cloud two systems can be used:

- **1D array of cameras:** This system corresponds to a linear or circular horizontal arrangement of traditional texture cameras, e.g. the circular array in Figure 12 (left). Because the visual information is acquired from only 1 dimension, horizontal, the acquired point cloud has limited applications, with probably few or no interactivity for the user. For increasing quality, a denser array can be used.
- **2D array of cameras:** This system complements the first one. By adding another dimension to the arrengement, the point cloud of the scene or object will be more detailed, making possible interaction for the user. A denser setup, e.g. the 2D array of cameras in Figure 12 (right), will

result in point clouds more accurate, leading to a rendering closer to the real world. Examples of this systems can be found in [17] and [24].

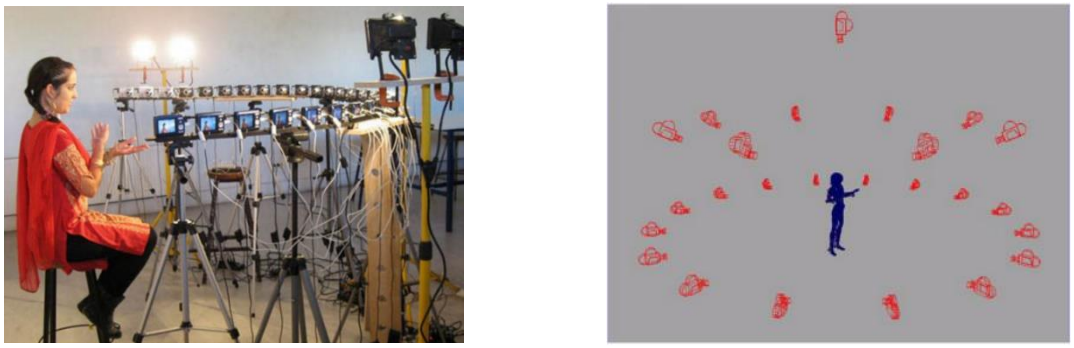

Figure 12: Left: 1D array of cameras [25]. Right: 2D array of cameras [17].

2.4.2. Texture Plus Depth

This type of acquisition paradigm involves traditional texture cameras like the previous system, to provide attributes like colour for the points in the point cloud, and depth sensors that measure the distance between the sensor and the scene objects being acquired, thus providing more accurate and direct data about the scene geometry. The objective of this acquisition systems is to obtain the positioning data, 3D coordinates, performing as little triangulation of images as possible. The most used depth acquisition systems are:

- **Structured light:** This technology uses one or more light projectors to project light patterns, e.g. grids, horizontal bars or random dots, in the visible or infrared spectrum, on the acquisition target; at the same time, one or more cameras capture images of the target with the projected light [26] where the projected patterns have been distorted depending on the object depths. For each dot or bar projected on the target, a corresponding matching pixel is obtained in the captured images, thus defining the 3D coordinates. The principle is that projecting some light patterns onto a 3D shaped surface produces a captured image with the projected patterns that appear distorted from other perspectives than that of the projector, and can be used for an exact geometric reconstruction of the surface shape. A well-known device using this technology is Xbox Kinect [27], shown in Figure 13 (left).
- **Time of flight:** This technology is associated to a sensor that emits a continuous light signal and detect properties, like phase shift, from the signal reflected by the target object /scene. The light source can be either a laser or a Light Emitting Diode, LED, working in infra-red or in the visible spectrum [28]. The measured properties are used to determine the time between the emission and reception, which is used with the speed of light to calculate the distance between the emitter and the target. As the target distance is usually short, this type of systems are usually cheaper comparing to some types of depth sensors. Figure 13 (middle), shows a typical time of flight device.
- **LIDAR:** This technology is associated to a sensor using a spatial narrow laser beam as a probe to measure the distances between a target and an optical sensor. The distances are calculated by measuring the time between the emission of the laser pulse and the detection of the reflected

signal, which is backscattered by the object [29]. Comparing to a similar type of time measuring system, known as Radio Detection and Ranging (RADAR), LIDAR uses shorter wavelengths which represents an advantage for working with materials with weaker reflections, e.g. rocks; moreover, its narrow laser beam allows reaching a higher resolution. The LIDAR specifications determine the precision/quality of the obtained point cloud data, notably the working wavelength the number of pulses per second, which allow obtaining more points per second and to differentiate "multiple returns", e.g. the backscatter from the top of tree canopy and the ground below. The LIDAR system can be used airborne, in a plane as shown in Figure 13 (right), or terrestrial, e.g. in a car. This type of solution can measure rather large distances.

Figure 13: Left: Kinect in Xbox 360 [27]. Middle: Heptagon's SwissRanger SR4500 [30]. Right: Airborne system using LIDAR [29].

2.5. Point Cloud Objective Quality Metrics

After being lossy encoded, the decoded point cloud data is prone to have some distortion, this means geometry and/or texture errors, and thus it is important to evaluate the quality of the decoded data, evaluating in this way the coding performance of the used point cloud codec. To evaluate the decoded point cloud quality, it is possible to use subjective assessment methodologies, i.e. assessments involving human subjects, as well as objective quality metrics, i.e. assessments involving some mathematical models. The objective quality evaluation metrics can be classified as full reference, reduced reference and no reference depending on how much they exploit knowledge on the original content in the quality assessment process. In subjective quality evaluation, the decoded content is assessed based on the opinion of several subjects that visualize the rendered content, which is very important since this assessment should represent the average quality of the user experience. Subjective assessment may happen as single or double stimulus process, meaning that the subjects will evaluate the rendered content without visualizing at the same time some reference content, e.g. the original content rendered from the original, non-coded, point cloud. Because subjective assessment is cumbersome, time consuming and expensive, objective quality metrics become very important. Naturally, the scores of a good objective quality metric must correlate well with the subjective assessment results.

This section reviews the most relevant objective quality metrics for point clouds available in the literature. In this context, the metrics adopted by MPEG assume particular relevance for its standardization project on point cloud coding. MPEG has been identifying the best ways to evaluate point cloud codecs, notably a framework for objective evaluation and benchmarking of point cloud codecs can be found in [31]; moreover, evaluation criteria and datasets can also be found in [32] and

[33]. The point cloud objective quality evaluation is often based on metrics already used for 3D mesh and video coding, such as the Peak Signal to Noise Ratio (PSNR), which is computed by comparing the original and decoded point cloud geometry or attributes. This type of quality evaluation is full reference, as it compares directly the original and decoded contents. The most relevant objective quality metrics for point clouds can be divided in the following two main types:

- **Cloud to Cloud (C2C):** This first set of metrics compares the two point clouds, the original point cloud and the decoded point cloud, using point by point metrics, even if the two points clouds don't have the same number of points.

  Each point in a point cloud can be represented as in equation (1), including its position (x,y,z), an optional colour attribute [c] and other optional attributes $[a_0..a_A]$ [32].

$$v = \Big(\big((x,y,z),[c],[a_0..a_A]\big): x,y,z \in \mathbb{R}\,,[c\ \in (r,g,b)|r,g,b \in \mathbb{N}]\,,\big[a_i \in [0,1]\big]\Big) \quad (1)$$

  The original point cloud, with K points without a strict order, is defined in equation (2) and the decoded or degraded one, with N points, where N may be lower or equal than K, is defined in equation (3).

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

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

  For C2C two distinct quality metrics can be assessed, notably metrics for geometry and for the attributes of the point cloud. For geometry, it can be evaluated by computing the **geometric PSNR**, see equation (4)*,* which expresses the ratio between the original data energy and the error energy introduced by the coding process. This coding error is itself a distortion metric, corresponding to the root mean square, *rms*, of the distance between the two sets of points. The *rms* distance is an average of the distance between a point in the original point cloud, and the corresponding closest neighbour in the degraded point cloud, taken for all the points in the point cloud, as represented in equation (5). In the geometric PSNR, the used *rms* is the worst case of the *rms* calculated between the original point cloud and the degraded one, or the opposite, equation (6). These metrics are computed for all points in V_{or} regarding the corresponding closest point in V_{deg}, and do not require the two point clouds to have the same number of points.

$$psnr_{geom} = 10log_{10}\left(\frac{\left\|max_{x,y,z}(V_{deg})\right\|_2^2}{\left(d_{sym\_rms}(V_{or},V_{deg})\right)^2}\right) \quad (4)$$

$$d_{rms}(V_{or},V_{deg}) = \sqrt{\frac{1}{K}\sum_{vl\in Vor}\left[v_l - v_{nn\_deg}\right]^2}\,, \quad (5)$$

v_l is a point in V_{or} and v_{nn_deg} is the corresponding nearest point in V_{deg}

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

To overcome the limitations associated to an average error, it may be more expressive from the quality assessment point of view, to consider the worst case distance. This means choosing the biggest distance between a point in one point cloud, and the closest found neighbour in the other point cloud. The so-called **Hausdorff distance** follows this type of approach as may be seen in equation (7):

$$d_{hauss}(V_{or}, V_{deg}) = max_{vl \in vor}\left(\left\|v_l - v_{nn\_deg}\right\|_2\right) \quad (7)$$

$$d_{sym\_hauss}(V_{or}, V_{deg}) = \max\left(d_{hauss}(V_{or}, V_{deg}), d_{hauss}(V_{deg}, V_{or})\right) \quad (8)$$

As for the attributes of the point cloud, their quality can also be assessed using the *rms* or *PSNR* metrics. The colour attributes PSNR can be computed per individual RGB or YUV component, by comparing the corresponding colour components (luminance or chrominance) of the two point clouds, the original and the decoded point clouds. As an example, in equation (9), shows how to compute the luminance (Y) PSNR and in equation (10) the corresponding *rms*:

$$psnr_y = 10log_{10}\left(\frac{\left\|max_y(V_{deg})\right\|_2^2}{\left(d_y(V_{or}, V_{deg})\right)^2}\right) \quad (9)$$

$$d_y(V_{or}, V_{deg}) = \sqrt{\frac{1}{K}\sum_{vl \in Vor}\left[y(v_l) - y\left(v_{nn\_deg}\right)\right]^2} \quad (10)$$

- **Cloud to plane (C2P):** To obtain quality assessments closer to the subjective assessments performed by a human, this type of metric proposes to assess the quality of the decoded point cloud by measuring distances not between two points but rather between one point and a surface, in this case a plane associated to a point in the original cloud. In practice, instead of using a point in the original point cloud, like in C2C metrics, this type of metrics uses a plane to represent the original cloud under the assumption that what is important to measure is the distance between the decoded points and the surface that may be rendered from the original point clouds. The target is to obtain an objective quality assessment that is perceptually more representative, as the decoded points close to the surface defined by the original points are subjectively good independently of the distance to the closest original point.

  One example of this type of metric is defined in [34] and currently under study by JPEG. In this method, illustrated in Figure 14, for each point in the reference/original point cloud, a normal vector is needed to define a plane tangent to the corresponding point; if the normals are unavailable, they have to be estimated using the nearest points in its neighbourhood. Then, an error vector is computed between each point in the decoded point cloud and the nearest point in the original point cloud, represented as an **oblique arrow** in Figure 14. This point to point

error vector is then projected in the plane orthogonal to the normal vector direction, represented as a **dashed and vertical arrow** in Figure 14, to compute a new projected error vector. The amplitude of this projected error vector should better represent the error distance between the decoded points and the surface rendered from the original point cloud. In fact, a decoded point may have a very high point to point error but a rather low point to plane error; if the point to plane error is, in fact, subjectively more expressive, then the C2P metrics may be more subjectively meaningful although there are still no solid studies validating this assumption.

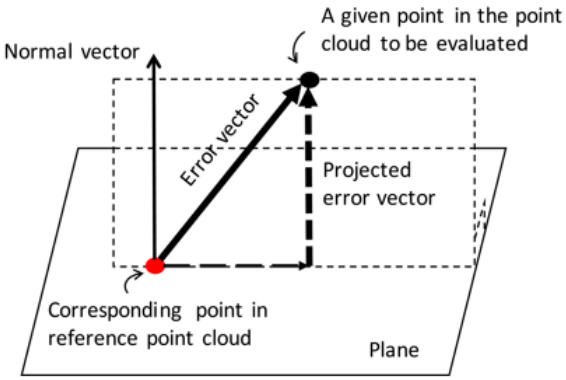

Figure 14: A cloud to plane metric example [34].

In C2P metrics, the obtained projected error vectors can be designated as type 1 if the decoded points are far from the plane, or surface, and take random directions, like the error in Figure 14, or as type 2 if the decoded point is close to the plane, even if the error vector is very long. In practice, the type 2 errors tend to keep the point cloud shape as the decoded points are close to the plane; this fact leads to think that C2P metrics may have a better subjective quality correlation comparing to C2C metrics.

A software, available in [32], can assess a pair of original and decoded point clouds and compute the presented C2C and C2P metrics in only a few seconds.

2.6. Point Cloud Coding Solutions

In this chapter, the most relevant dynamic point clouds coding solutions in the literature are described. Due to the objectives of the final M.Sc. Thesis, the solutions to be reviewed mainly target the coding of dynamic, this means evolving in time (and thus not static), point clouds. However, the coding of static clouds will be briefly reviewed since in some cases it is essential to understand how the correlation in space is exploited before exploiting the correlation along time. For each one of the coding solutions, each following subsection will present the objectives of the coding scheme, its coding architecture and walkthrough, the most relevant tools and, finally, its performance.

2.6.1. Real-time Compression of Point Cloud Streams

Objectives

In [36], a real-time point cloud compression scheme able to handle unorganized point clouds is proposed. The performance of this coding solution does not depend on the point cloud density, which

may be dense or sparse. This is a lossy coding solution as some quantization is performed while compressing the point cloud. The proposed solution is able to code both the geometry and the attributes, e.g. colours, of the point cloud and also exploits both the spatial and temporal redundancies in dynamic point clouds. The point clouds are represented as an octree data structure, which is a hierarchal structure in form of a tree, commonly used to divide a 3D space. The implementation of this coding solution [36] is available in the open-source Point Cloud Library (PCL) [12] which is a rather popular library for the point cloud research community. PCL includes several types of tools to process point clouds, notably to deal with generic and unorganized point clouds, and includes filtering, segmentation, recognition and visualization tools among others.

Architecture and walkthrough

Figure 15 (left) shows the encoder architecture for the point cloud coding system. The input consists of a dynamic point cloud and the output is a bit stream containing the compressed data. To represent a point cloud using the octree data structure, successive partitions of the occupied 3D space in 8 octants are performed as shown in Figure 15 (right). Each octant corresponds to a node in the octree, and its occupancy can be represented in binary (with '0' meaning empty and '1' meaning full), implying that each node with 8 children can be represented with a single byte. As the octants are always ordered in the same way (tree traversal), the bytes can be serialized in a consistent way as shown in Figure 15 (right), before being entropy coded and transmitted. The smallest octant contains one or more cloud points and correspond to the minimum voxel resolution, thus imposing the so-called *octree depth*.

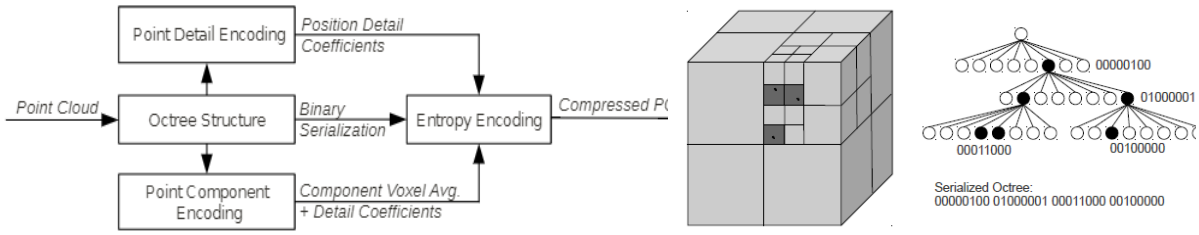

Figure 15: Left: Encoding architecture Right: 3D space division and corresponding octree with serialization [36].

As shown in Figure 15 (left), the encoder is composed by four modules, where the octree structure, point detail encoding and point component encoding are responsible for the creation of three streams of symbols, which are then compressed by the entropy encoding module, and finally transmitted as a single (multiplexed) bit stream. The encoding process proceeds as follows:

- **Octree structure (creation):** For each point cloud frame (time instant), an octree is created within a specific bounding box, fully containing the cloud points. The output is a byte stream, more specifically a serialization of the octree bytes as shown in Figure 15 (right); this process changes, depending if the octree is static or dynamic (details will be provided later).
- **Point detail encoding:** When the leaf nodes of the octree contain more than one point of the point cloud, this module encodes the specific position of each point inside a voxel, thus, improving the point cloud resolution and also the number of points transmitted as otherwise all points in the leaf node would be represented by its central point (thus with a higher error).
- **Point component encoding:** Similar to the point detail encoding module, this module encodes the attributes of each point in a leaf node, notably colour. This coding process involves

computing the distance of each colour attribute to the average colour of all points inside that specific leaf node/voxel. Other attributes can also be coded, although this codec does not propose any specific method with that purpose.

- **Entropy encoding:** This last module entropy encodes all the incoming symbol streams using some knowledge on the symbols statistics. For example, when Huffman coding is used, the most probable symbols are represented with a shorter coding word. A range encoder is used in this case for entropy encoding, where bytes and not bits are used in the coding process, which makes the process faster. Range coding is similar to arithmetic encoding, except that encoding is done with digits in any base, instead of with bits, and so it is faster when using larger bases (e.g. a byte) at small cost in compression efficiency.

Main tools

In this codec, there are two main tools, corresponding to two main modules in the architecture, with very high impact in the final compression performance. Those tools will be now further detailed.

A. Octree structure (creation)

For every point cloud frame, an octree data structure is created which is then used to exploit the spatial, and optionally temporal, redundancies. Thus, there are two distinct working cases for this module, defined next:

- **Static octree encoding (Intra coding mode):** This case corresponds to coding the point cloud frame or frames independently from each other, generating the so-called *Intra frames*, where only spatial redundancy is exploited (even if temporal redundancy exists for dynamic point clouds). To create the octree, each point of the point cloud is traversed, creating the branches of the tree and the corresponding nodes. In the point cloud traversal process, each voxel is decomposed in smaller voxels, the so-called octants; as the name indicates, there are always eight new octants for each decomposed voxel. As some octants do not contain any point, the corresponding octree node is unoccupied; as these octants are empty, they do not require further division and correspond to a '0' in the octree representation. After the octree is fully created, the bytes representing the occupancy of all nodes are compressed by the entropy encoder and later transmitted. To limit the size of the octree, the octree depth may be limited, thus limiting the minimum size of a voxel. This will also determine the precision (or resolution) of each cloud point, i.e. the maximum error that the decoded 3D point positions will have when compared to the original 3D point positions. If the point cloud has multiple frames and no temporal redundancy is exploited, all point cloud frames are Intra coded.
- **Differential octree encoding (Inter coding mode):** In this case, the temporal correlation is exploited using the so-called *Inter frames*. This implies that each point cloud frame with the exception of the first may be coded using as reference the previous (decoded) point cloud frame. The process proceeds as follows:
    1. An octree with 16 nodes for each branch is created which simultaneously stores two point cloud frames. The first eight children of each branch constitute the so-called *buffer A*, while the remaining 8 children nodes constitute *buffer B*.

2. The nodes of each *buffer* store the points of a single point cloud frame and these *buffers* are alternatively filled with the successive input point cloud frames. This means that every time a new point cloud frame needs to be coded, the octree is traversed, as in the static case, to insert the points in each voxel, thus re-filling the respective *buffer*.
3. When a voxel does not exist in the previously processed octree structure, a new child node is created. It may be necessary to expand the bounding box if points in a new frame are outside the previously defined bounding box.
4. Every time a *buffer* is overwritten with a new point cloud frame, a direct comparison between the nodes of the two octrees in the double buffer is made, and a stream of bytes is generated to be entropy coded. Figure 16 (left) shows this comparison, or XOR operation, between a new point cloud frame that was just inserted in *buffer* B, with the last point cloud frame stored in *buffer* A. This comparison, produces a byte stream with the changes associated to the occupancy of the nodes. In the XOR operation, '0' corresponds to no changes and '1' corresponds to changes in the occupancy of the nodes.
5. Using this information, the decoder only needs to update the octree nodes where changes have occurred. When less motion exists between successive point cloud frames, the XOR operation will result in no changes detected between the octrees in the buffers and more zeros will be sent, thus reducing the entropy of the output stream and thus the bitrate to represent the point cloud after entropy coding.

B. Point Detail encoding

Since this codec targets real-time systems that have strict memory and computational requirements, the size of the octree structure must be limited in depth, which may result in the same voxels containing more than a single point of the point cloud. Without this module, all points in a voxel would be represented at the center of the voxel, thus coding the point cloud geometry eventually with some significant noise. To increase the geometry accuracy, this module allows to code the position of each point in a more precise way by computing for each of the points within a leaf node, the distance to a reference position, in this case the smallest *(x,y,z)* in the same voxel, as shown in Figure 16 (right). This produces a set of *(x,y,z)* distances for each point, which are coded and transmitted. It is possible to adjust the precision associated to the position of each point by quantizing this distance to obtain different resolutions for the point cloud, i.e. different accuracies for each point (error between the decoded and the original point).

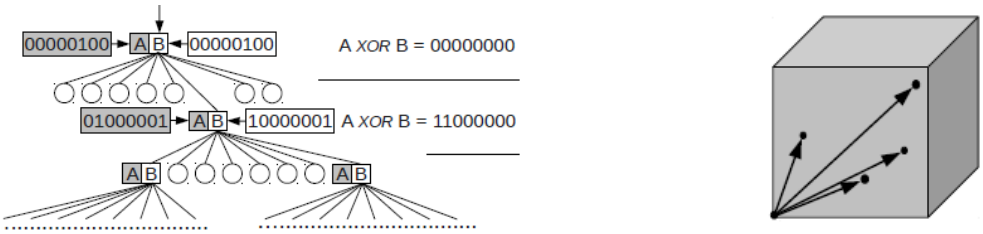

Figure 16: Left: XOR between two consecutive octrees associated to two point cloud frames. Right: Point detail reference distances [36].

Performance assessment

The performance of this point cloud coding solution was studied using a 15 seconds sequence, acquired at a 30 Hz rate, and containing a man waving [36]. The uncompressed point cloud has, on average, 12 bytes per point, *bpp*. The used coding modes were the static octree compression (Intra mode as defined above), and the differential octree compression (Inter mode as defined above). Figure 17 (left) shows the average *bpp* for the compressed data versus the point precision when using static octree compression and differential octree compression, with no point detail encoding. It is clear from the results that the Inter coding mode is beneficial as it increases the compression efficiency regarding the Intra coding mode; however, for lower point resolutions, the Inter mode compression gains are lower. Figure 17 (right) shows the bitrate along time (frames) when static and differential octree compression are used. The visible peaks for differential coding are associated to motion effects in the point cloud sequence. However, the main observation is that the differential octree coding clearly outperforms the static coding approach.

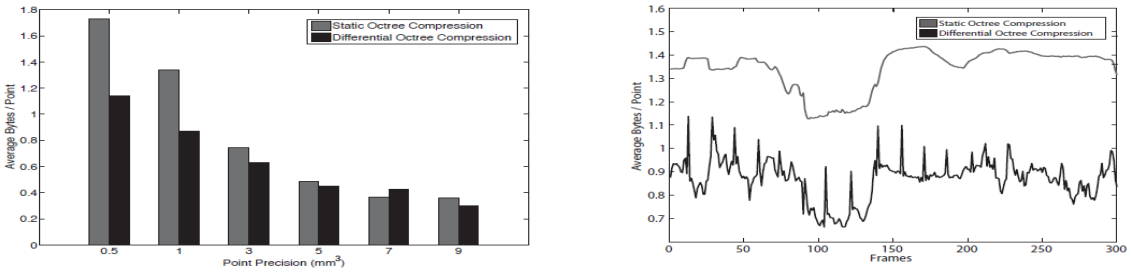

Figure 17: Experimental results for static vs. differential octree compression Left: Average bytes/point versus point precision. Right: Average bytes /point along time, notably first 300 frames of the test sequence [36].

Table 1 shows the compression factor and bitrate (bytes per point) as a function of the precision, for static and differential octree coding. For a point detail precision of 0.5 mm, the static octrees can reduce the average *bpp* from 12 to 1.73, which corresponds to a compression factor of almost 7. When differential octree compression is used, the *bpp* is reduced to 1.14, resulting in a compression factor around 11. The compression efficiency gain obtained by adopting differential octree coding is 34% in this case. Table 1 also shows that the compression efficiency gain decreases as the point precision decreases, due to a saturation effect caused by the transmission of a fixed frequency table necessary for entropy decoding, i.e. an overhead that must be sent for every frame.

Table 1: Experimental results for each precision level [36].

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

The proposed solution achieves a better compression but has some disadvantages, notably the temporal redundancy is only exploited between adjacent frames, a limitation resulting from the real-time

encoding requirement. Another disadvantage is the decrease of compression efficiency gain for differential coding, regarding the static coding, for lower point resolutions. A problem that is found in the paper is the lack of a rate-distortion curve, that can compare the objective quality of the point clouds for different bitrates, in order to assess the quality of the decoded point cloud for different compressions.

2.6.2. Design, Implementation and Evaluation of a Point Cloud Codec for Tele-immersive Video

Objectives

The codec proposed in [5] is part of a real-time and generic point cloud system using an octree structure to represent the point cloud; this system is suitable for augmented and immersive 3D video. This coding solution offers lossy compression for the geometry and attributes, notably the colour; moreover, progressive coding is performed, where the coding bit stream can be partially decoded to obtain a point cloud with less quality using less bandwidth, and better quality may be obtained by decoding more data. For the geometry, spatial redundancy is exploited by decomposing the 3D space into an octree and entropy encoding the occupancy nodes; moreover, temporal redundancy is exploited by using an Inter-prediction mechanism based on the Iterative Closest Point (ICP) algorithm. The ICP algorithm is applied to macroblocks, which are groups of voxels with a fixed size, to obtain a rigid transform. For the colour attributes coding, only spatial redundancy is exploited by mapping the octree colour data into a 2D image grid, and then applying JPEG coding. This solution is currently being used as reference for the future development of a MPEG point cloud coding standard.

Architecture and walkthrough

The architecture of the proposed point cloud codec is shown in Figure 18. In this codec, the incoming point cloud frames are represented as octrees, where two operations are performed: Inter coding to exploit temporal redundancy, and Intra coding to exploit spatial redundancy.

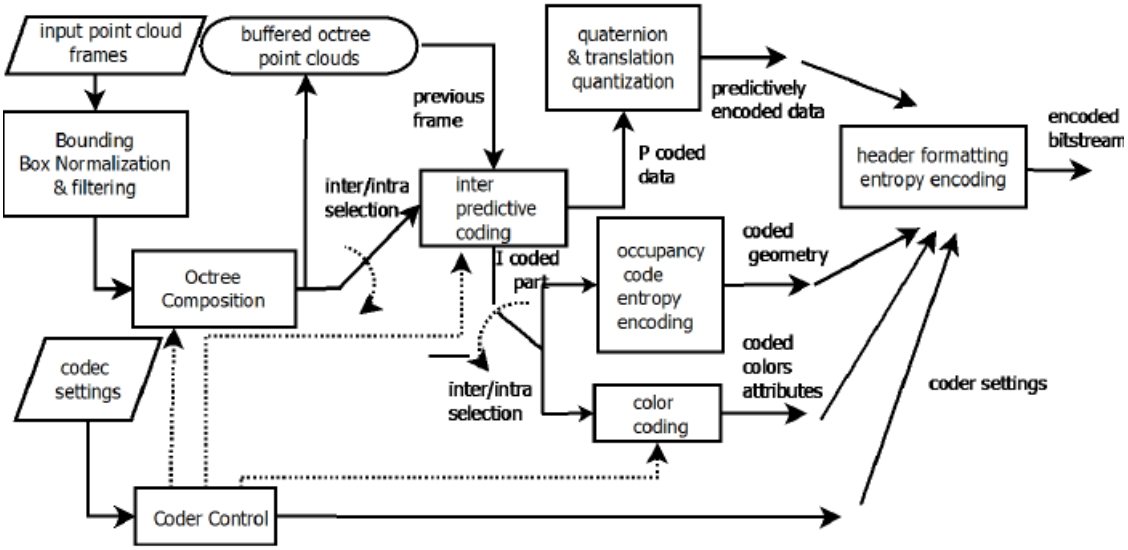

Figure 18: Encoder architecture of the point cloud codec [5].

The function of each module in the encoder architecture is described next:

- **Bounding box normalization & filtering:** In this first module, a bounding box defined by two opposite corners with minimum (x_min, y_min, z_min) and maximum (x_max, y_max, z_max) coordinates is first obtained. If the point cloud frame is an Intra frame, the computed bounding box is expanded in the two opposite corners with a defined constant. For Inter frames, if the computed bounding box fits the previous frame's bounding box, then the previous expanded bounding box is re-used. If the bounding box exceeds the previous expanded bounding box, the frame needs to be Intra coded and a new expanded bounding box is computed. However, before the bounding box creation, the point cloud gets filtered, removing points that do not include a minimum number of neighbours within a given radius, this means rather isolated points.
- **Octree composition:** Each incoming point cloud is represented as an octree by recursively sub-dividing each occupied voxel into eight smaller voxels, appropriately dividing the 3D space corresponding to the full computed bounding box. Each level of the octree corresponds to a level of detail, referred as LoD, and is associated to octree occupancy codes that are transmitted to the decoder. An occupancy code is just a bit indicating which octree nodes are occupied, and may thus be further divided into lower levels.
- **Inter predictive coding:** For every Inter coded point cloud frame, the ICP algorithm is applied to parts of the full point cloud, which iteratively transforms (with rotations and translations) the previous point cloud until minimizing the difference between the actual, original point cloud frame and its prediction. In practice, the ICP is applied to pre-defined, so-called macroblocks, which fulfil certain conditions, as explained in main tools, thus producing a rigid transform. Otherwise, Intra coding is applied for the macroblocks failing to fulfil these conditions.
- **Quaternion & translation quantization:** The rigid transforms previously obtained, in practise the predictors, are decomposed into a rotation matrix, which is converted into a quaternion, and a translation vector. The quaternion corresponds to a system of 4 variables, where one is a real number and the 3 others are the so-called quaternion units (*i, j* and *k*); the quaternion is used due to the rotational nature of the transform. The quaternion is then quantized, while the translation vector is quantized in the three components defining the three axis, all using 16 bits per value or component.
- **Occupancy code entropy encoding:** For both Intra and Inter frames, this module codes the point cloud geometry by applying entropy coding to the serialized byte stream that represents the octree with occupancy codes.
- **Colour coding:** For both Intra and Inter frames, this module is responsible to compress the colour attributes of the points, by mapping the octree to a 2D image grid.
- **Coder control:** This module controls some of the other modules' operations; this is simply performed by loading a configuration file with appropriate pre-specified settings.
- **Header formatting entropy encoding:** A range encoder is used for entropy encoding of the various symbol/bit streams.

Main tools

Although there are several tools in this codec to compress the point cloud, this section will focus on the three most relevant tools considering the scope of this report, this means dynamic point clouds coding:

- **Inter predictive coding:** This tool is responsible for the prediction of the point cloud geometry between successive frames. As more or less temporal redundancy may exist, within each point cloud frame only the successfully predicted parts are coded with the Inter coding mode, while the unsuccessfully predicted parts are coded with the Intra mode. This module works as follows:
  1. The current frame (P) and the previous Intra (I) frame are divided into macroblocks, which include several voxels of the octree; in this case, a macroblock encompasses 4 levels above the final octree depth level, which means that each macroblock includes 16x16x16 octree voxels.
  2. For each occupied macroblock in the Inter frame, or P macroblocks, the following steps are performed:
     - The first step consists in checking if the corresponding macroblocks in the previous frame are also occupied. The occupied P macroblocks that correspond to unoccupied I macroblocks will be stored in a buffer to be Intra coded, as they cannot be predicted.
     - In the second step, a verification is made for the previously positively corresponded macroblocks, by checking if the number of points in the P macroblock is within a ± 50% range of the number of points in the corresponded I macroblock.
     - The last step is only executed for the positively corresponded macroblocks that have fulfilled the condition in the previous step: it consists in computing the colour variance for those macroblocks and discarding those ones whose variance is below a pre-defined threshold. This step assured that the geometry is well predicted, while discarding high-variance colour attributes between corresponded macroblocks. The macroblocks that fail any of these conditions are Intra coded.
  3. The corresponded macroblocks that pass the previous two conditions are eligible for Inter coding and a prediction is made using a rigid transform between the macroblocks. In this case, the ICP algorithm is applied using only the geometric positions of both corresponded macroblocks. In case the ICP prediction from the I macroblock fits well the P macroblock, according to a pre-defined threshold, the P macroblock is Inter coded using a rigid transform which allows to map the points in the I macroblock to the P macroblock. This rigid transformation is converted to a quaternion matrix and a translation vector and all values are quantized with 16 bit. In case the prediction fails the threshold, the macroblock is Intra encoded.
  4. The colour of the Inter predicted macroblocks can be optionally coded by sending an offset with three components that corresponds to the colour difference to the matched macroblocks from the previous frame. This may be applied to compensate colour variances in the macroblocks due to changes in lighting, thus originating different brightness in the two macroblocks.

5. In order for the decoder to be able to reconstruct the predicted macroblocks, each Inter coded macroblock is described by a key containing the quantized quaternion matrix, the translation vector, and optionally the three colour offset components. The decoder can decode these Inter coded macroblocks in any order, which allows the parallelization of the encoding and decoding processes, useful for real-time compression.

- **Occupancy code entropy encoding**: This tool is responsible for the efficient compression of the point cloud geometry in the spatial domain, by processing the octrees computed for both the Intra or Inter coded point cloud frames, and coding the occupancy codes. For Inter coded frames, this is done only for macroblocks that could not be predicted, and that are stored in a buffer, constructing an octree to which Intra coding is performed. Entropy coding is applied and two LoDs layers are constructed to reduce the overhead; this means that the 2-layers scalable bit stream allows decoding two point clouds, one coarser and another with higher, predefined detail. This will also allow to keep the encoder simple and fast, thus resulting into two possible resolutions for the decoded point cloud frames.
- **Colour point attributes coding:** This operation needs to be efficiently performed since a large amount of data rate is associated to colour point attributes. This module starts by traversing the octree in a depth-first search, which iterates over all nodes of the octree by going to each branch of the tree as far as possible in depth before backtracking. While the octree is traversed, the colour attributes of each occupied leaf node are mapped to a standard 2D image grid. Then the JPEG standard is used to code the resulting image. In Figure 19, the sequence of increasing numbers corresponds actually to the order by which the 2D image is filled from the octree depth-first traversal. Each one of the grid entries will be filled with one colour attributes, following the octree traversal. The JPEG standard, using the popular discrete cosine transform (DCT), quantization and Huffman coding, is applied to the created 8x8 blocks/grids. Thus, only spatial redundancy is exploited in this module as not temporal predictions are made.

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 19: Scan pattern used to fill the image grid [5].

Performance assessment

To evaluate the compression efficiency of this codec, the datasets in [35] were used; these point clouds contain realistic tele-immersive data. For the objective assessment, the datasets consisted in dynamic point clouds named as *Alex, Christos* and *Dimitrios*, and the performance of the proposed codec was compared against the performance of the point clouds codec presented in Section 2.6.1 [5], which uses Differential Pulse-Code Modulation (DPCM) for colour coding. As for the subjective assessment, the test consisted in 20 users experiencing a realistic 3D tele-immersive system in a virtual

3D room scenario, acquired with multiple sensors (in this case, Microsoft Kinect), and allowing interaction with 3D avatars. Only the *Alex* and *Christos* datasets were used and they were coded with the following 3 codecs: i) DPCM from [5]; ii) JPEG; and iii) Inter coding mode. The last two codecs correspond to the colour and geometry coding used in the codec proposed in this section. The rendering was done at 23 Hz, and the Inter predictive codec used a prediction structure of alternating Intra and Inter frames, this means IPIPI….

- **Objective assessment:** The objective quality assessment has been made using the MPEG objective metrics already presented in Section 2.5. In the objective assessment, the performance of the two types of coding modes was evaluated:
  - **Intra coding:** For the evaluation of the geometry component, the used metric was the geometric PSNR presented in Section 2.5. The rate-distortion (RD) plots for several LoDs (which correspond to the octree level of the decoded point cloud) are shown in Figure 20 (left). As expected, the highest PSNR requires a high bit rate and can be obtained for LoD 11, which provided a realistic representation for close range point clouds in a 3D immersive system. Naturally, a lower PSNR is obtained for a low bit rate, notably using LoDs of 8 and 9, which only provides realistic representations for point clouds that are far away. The performance of the colour coding solution was also evaluated, in this case, with the component colour PSNR metric defined in Section 2.5. The results are plotted in Figure 20 (right), which shows the number of bytes per point for different LoDs. The chart includes 7 curves, where four correspond to DPCM coding, used to code the colour attributes as in the solution reviewed in Section 2.6.1 [5], in this case using 4, 5, 6 and 7 bits per colour component. The other three curves correspond to JPEG coding using 75 and 80 as quality parameters. The difference between the two solutions is evident: The JPEG based solution requires less *bpp* for coding the same LoD in comparison to the DPCM solution. For both solutions, the *bpp* decreases for the higher LoDs, due to the increased number of represented points, which results in an increase in spatial correlation.

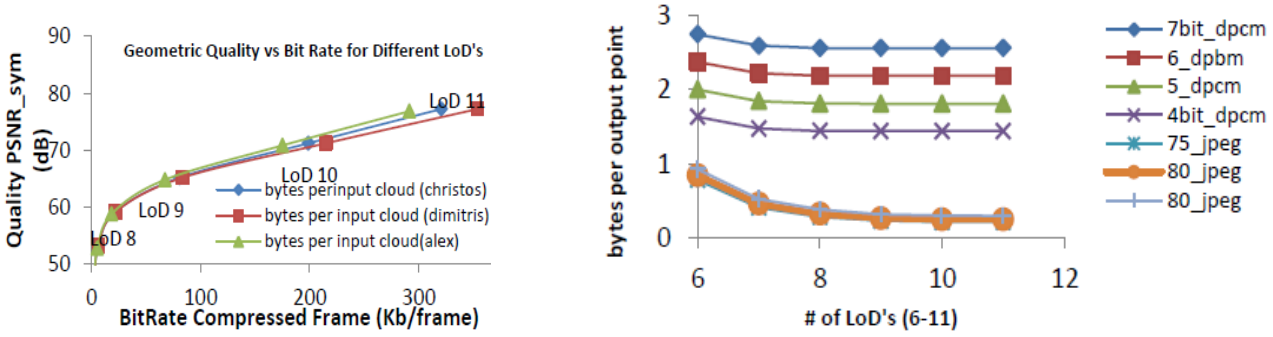

Figure 20: Left: Geometric PSNR vs. bitrate for different LoDs. Right: Bytes per point for different LoDs [5].

  - **Inter predictive coding**: The geometry compression performance for the Inter coding mode was also evaluated by comparison with the pure Intra coding mode, using the geometric PSNR metric presented in Section 2.5. In Figure 21 (left), the total bitrate for the compressed Intra vs. Inter coded frames is shown for three different point clouds. For each point cloud, the blue

bar corresponds to the Intra coded frames while the red bar corresponds to the Inter coded frames. Figure 21 (right) shows the resulting geometric PSNR for the same datasets using a LoD of 11. Observing both charts allows concluding that the use of Inter coding is beneficial as it reduces the coding sizes between 20.5% and 34.8%, although at a significant cost of a 10 dB loss in the geometric PSNR (quality).

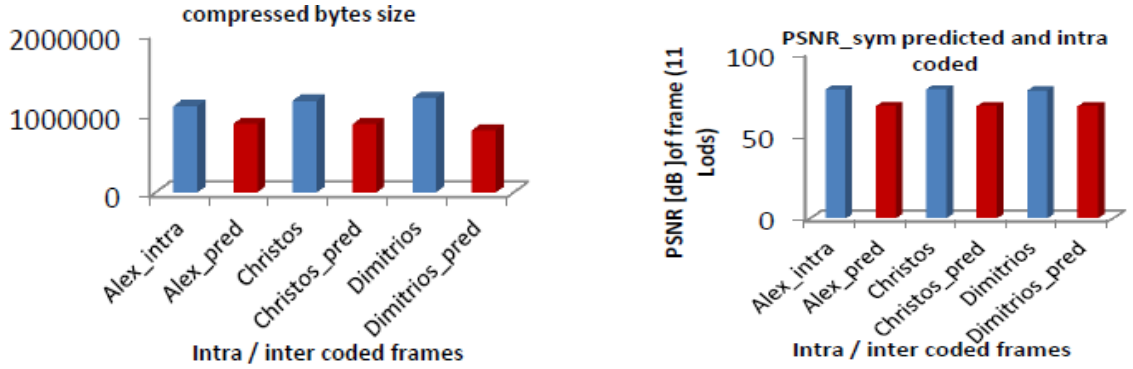

Figure 21: Left: Compressed bytes' size for Intra and Inter coded frames. Right: Geometric PSNR for Intra and Inter coded frames [5].

- **Subjective assessment:** At the end of the assessment experience, the users filled a questionnaire focusing on 8 quality aspects, choosing from a progressive scale ranging from 1-bad to 5-excellent. The subjective quality results are presented in Figure 22 for the 3 used codecs: DPCM, JPEG, and Inter coding, named Or., JPEG and pred. in the charts' axis, and the 2 datasets used were Alex and Christos. While the bars represent the mean opinion scores, the lines represent the standard deviation. In all charts, the use of Inter coding for the geometry does not significantly reduce the subjective quality which somehow contradicts the objective results above and validates the proposed system. However, these results may have been influenced by the point cloud acquisition method, which is a low cost consumer setup that resulted in defective point clouds.

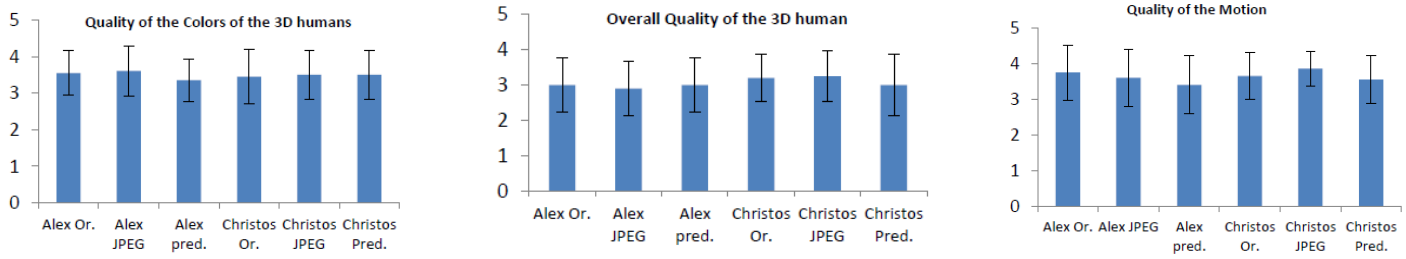

Figure 22: Left: Subjective quality for the decompressed colour. Middle: Subjective quality for the decompressed 3D human. Right: Subjective quality of motion [5].

The main conclusion is that the codec described in this section proposes an Inter coding mode that is able to successfully exploit the temporal redundancy in the point cloud geometry and colour, thus improving the compression performance with respect to pure Intra coding. However, the assessment has not enough information, e.g. the colour assessment does not include the colour PSNR. Another limitation is that the geometric PSNR was not assessed for other codecs.

2.6.3. Graph-based Compression of Dynamic 3D Point Cloud Sequences

Objectives

This solution [37] targets the compression of dynamic point clouds in terms of geometry and colour attributes exploiting temporal correlation. The main idea is to exploit a graph based representation, with the corresponding 3D point positions and colour attributes being signals of the vertices of the graph. In [37] the motion between adjacent point cloud frames is estimated obtaining motion vectors which are used to predict and encode both geometry and colour attributes of the graph.

Architecture and walkthrough

In Figure 23 it is shown an overview of the Inter coding performed in this solution. Only the first frame of each sequence is coded in Intra mode, with the coding solution of [38] for geometry and [39] for colour attributes. All subsequent frames are Inter coded. To exploit the temporal redundancy of point cloud sequences, this codec performs motion estimation and transmits the coded motion vectors to the decoder. Then, the geometry and the colour attributes are predicted using the motion vectors to create a motion compensated point cloud frame. The geometry residual is coded using a double buffered octree approach similar to the one described in Section 2.6.1. The residual coding of the colour attributes also requires the creation of an octree. However, the irregular structure of the octree occupied leaf nodes is organized into a graph based representation.

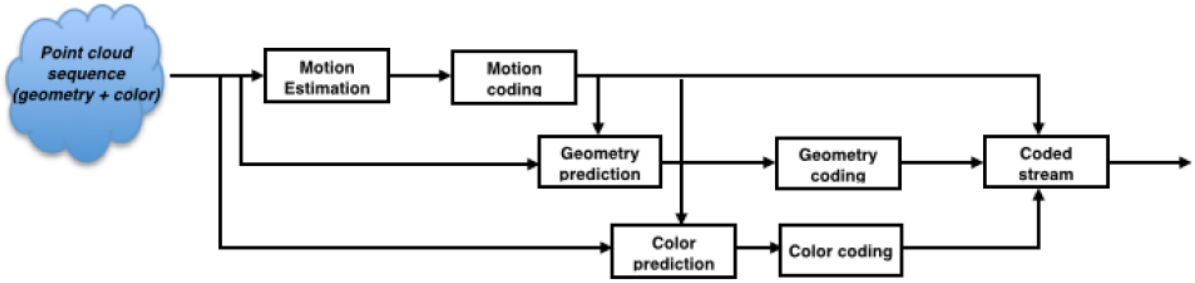

Figure 23: Encoding architecture of the codec [37].

In this context, a graph is a simple type of data structure where nodes (vertices), which are defined with 3D locations, can be arbitrary connected with edges. In this case, a weighted graph is computed from the point cloud data where a matrix with weights is assigned to each edge and each weight is inversely proportional to the distances between vertices. It is possible to create a graph from an octree, where each of the occupied voxels is connected to a set of neighbouring occupied voxels using the K – nearest neighbours, a method widely used in the literature. The architecture of the proposed codec as shown in Figure 23 is described next:

- **Motion estimation:** In this module a motion field is obtained, by computing features for each point of the graphs of the previous frame and target frames and obtaining correspondences between the two frames.
- **Motion coding:** The motion vectors previously generated are compressed using the Graph Fourier Transform (GFT), a transform that allows a spectral representation of the 3D coordinates and colour components signals, exploiting the smoothness of the motion vectors estimated.

- **Geometry prediction:** For each point cloud Inter frames, the estimated motion field is used to predict the 3D positions by using a reference point cloud frame already available (known by the decoder).
- **Geometry coding:** The structural difference between the motion compensated 3D positions and target frame 3D positions are encoded and transmitted in a bit stream using an octree double-buffering approach.
- **Colour prediction:** For each point cloud Inter frame, temporal redundancy of the colour attributes is exploited by encoding the residual between the target frame and the motion compensated prediction.
- **Colour coding:** The residual between the motion compensated colour predicted attributes and the target point cloud colour attributes is encoded using graph transforms and applying uniform quantization.
- **Coded stream:** This module is responsible to apply entropy encoding to the three symbol streams and thus generating a bit stream that can be transmitted to the decoder side.

Main tools

In the Inter coding mode of this solution, motion estimation is performed between two consecutive point cloud frames using a feature matching approach, resulting in a dense motion field. The obtained motion field is used to predict both 3D geometry and colour attributes of the target, and to be-coded, point cloud frame. The residual obtained between the predicted and target frames is then encoded. The following tools are used:

- **Motion estimation:** This module uses features, which is a compact piece of information used to represent the 3D point cloud data. Typically, features are designed to be invariant (or robust) to several transformations or perturbations. In this solution, features are used to estimate the motion between consecutive frames. This module works as described next:
    1. The features are computed for each node, or vertex, of the graph taking into account the eight closest neighbouring vertices around it, giving this way a sense of orientation to the features descriptors. The graph spectral features are obtained, for each orientation and for both geometry and colour attributes, by computing Spectral Graph Wavelet (SGW) coefficients that result in feature descriptors. This is performed for the graphs of the current frame and target frame.
    2. Then, it is necessary to compute the correspondences between two adjacent frames, the reference and target point cloud frames which are represented as graphs with features in each vertex. To perform this matching, the features computed in the previous step and assigned to the vertices of the two graphs are compared to measure their similarity. This is done by computing the Mahalanobis distance, which measures the distance between features, by taking into account the contribution of geometry and colour parts, obtained using a different scale SGW. For each node, or vertex, in the target frame, the node in the reference frame with the minimum Mahalanobis distance (i.e. the best score) is selected as the best match.

3. A sparse set of matches are selected from the entire set of matches computed for all nodes of the target point cloud frame. The objective is to eliminate inaccurate correspondences, especially the ones that have large erroneous displacements. The following criterion was applied: for the target point cloud frame, which can be represented in clusters containing groups of 26 closest nodes, only the node matched with the best score, and with a score below a certain threshold, is chosen to represent each cluster. The remaining node matches are discarded.
4. A dense motion field is computed from the motion vectors retained in the previous step. Thus, motion vectors are recomputed for the nodes of target frame for which a motion vector was previously discarded. It is performed an interpolation which assumes that all nodes in target frame that are in close to each other (in the same cluster) follow a similar motion. This interpolation consists in a regularization problem that estimates the motion vectors for all unmatched nodes based on the principle that the motion signal crossing every graph edge has a smooth transition. For that purpose, it is even allowed the smoothing of the motion vectors of already matched nodes. The output of this module, a dense motion field, is then transmitted to the next module.

- **Motion coding:** The input dense motion field is transformed with the Graph Fourier Transform, which can be rather efficient since the motion vectors were interpolated with smooth constraints and thus, are well correlated spatially. The transform coefficients are then uniformly quantized independently in the three dimensions, x, y, and z, using the same quantization constant. The resulting coefficients are then entropy encoded with an adaptive Run-Length and Golomb-Rice (RLGR) entropy encoder, starting with the Run-Length encoding and followed by the Golomb-Rice encoding. The first encoding technique achieves compression by encoding sequences containing the same repeated symbols with a single and shorter word. The second technique encodes the data based on its probability, by dividing the probabilities values with a power of two, and then constructing the codeword based on the quotient and remainder. Figure 24 shows the corresponding architecture for the motion coding, where $v_t{}^*$ is the motion vector that is encoded, and $\hat{v}_t{}^*$ is the corresponding decoded motion vector obtained doing the reverse operations, in both encoder and decoder sides.

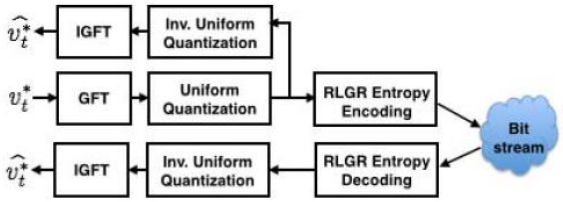

Figure 24: Architecture of the encoder and decoder of the motion coding solution [37].

- **Geometry prediction and coding:** The stored reference frame, which is an Intra frame for the first frame and Inter for every subsequent frame, plus the decoded motion vectors are used in the prediction of the target frame geometry. The reference point cloud frame geometry (3D locations) is warped to obtain a motion compensated point cloud frame, thus creating a prediction of the target frame. Then, the dynamic geometry compression algorithm used in

Section 2.6.1 is applied but in this case, the two buffers don't correspond to the reference point cloud frame and the target current frame, but to the motion compensated (warped) frame and target frame. By performing the XOR operation, the changes between the nodes of this two frames can be serialized and entropy encoded, before being sent to the decoder side. Finally, the decoder side uses the serialized data with the decoded motion vectors to obtain the target frame, which is then stored to compute the subsequent frame, in the same way. Thus, the proposed solution of Section 2.6.1 is much enhanced, it does not correspond to a simple differential encoding but to a motion compensated prediction plus residual enhancement as usual in standard 2D pixel video codecs.

- **Colour prediction and coding:** The first frame of the sequence is coded with the Intra coding mode, which uses the coding solution [39]. For each subsequent frame, the obtained warped frame in the geometry prediction is used to predict the colour attributes of the points in the target frame. For each node of the target frame, a colour value prediction is computed by averaging the colours of the nearest points in the warped frame, where usually the 3 nearest points are used. Then the difference of the colours of the nodes in the warped and target frames, or residual, is encoded in order to exploit the temporal redundancy, by applying a GFT as in [39], for each colour component separately. The resulting Fourier coefficients are then uniformly quantized and entropy coded with an arithmetic encoder.

Performance assessment

The used data sets consisted in two dynamic sequences with human bodies moving: a sequence of a woman (full body) in a dress with 64 frames, a sequence of man (full body) with 30 frames, and a sequence of a man (upper body) with 63 frames. The first two were captured with the system found in [24], while the last one was captured with the system found in [40]. The experiments were performed by *voxelizing* the point cloud frames to resemble the data collected with the system found in [40].

The first experiment is used to evaluate the performance of the motion estimation algorithm, and consists in selecting two consecutive frames of the same sequence and applying the motion estimation and coding algorithm. The number of vertices that are selected as best match between the two graphs is 500, corresponding to less than 10% of the occupied voxels in the reference frame. The motion of the remaining nodes is then estimated based on the algorithm previously described. In Figure 25 (left) it is shown the superposition of the two adjacent frames, where it is visible the motion, e.g. in the woman's head and left arm. In Figure 25 (middle), 5 correspondences of the 3D positions of the 500 matched vertices are explicitly shown, represented in red. Every green point in the target frame has a correspondence to a red point the reference frame, and thus the matched vertices cover uniformly the entire surface of the point cloud object (woman). In Figure 25 (right) it is shown the result of the superposition of both the motion compensated frame and the target frame; it is possible to conclude that the motion compensated frame is much closer to the target frame, in comparison to the reference frame, and thus, less residual information is necessary to be transmitted.

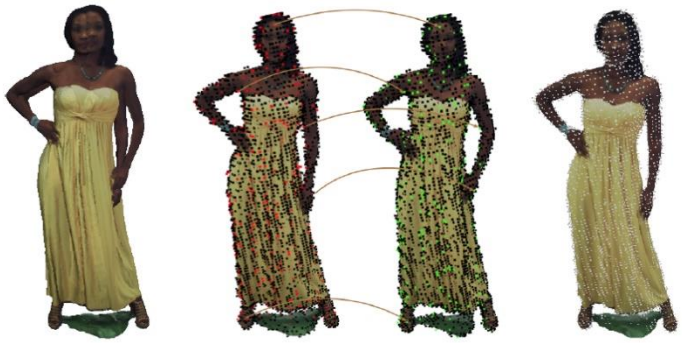

Figure 25: Left: Superposition of the reference and target frames. Middle: Correspondences between the target and reference frames. Right: Superposition of motion compensated frame and target frames [37].

The overall performance of the proposed codec for 3D geometry and colour attributes was also assessed, where the first frame is coded as Intra, and all the subsequent frames are coded as Inter frames, using as reference the previous frame. Figure 26 (left), computes the average Signal-to-Quantization Noise Ratio, or SQNR, for different bits per vertex, *bpv*, for the motion vectors coded in the signal domain (blue function) and in the Graph Fourier domain (red function). The crosses or circles in the two functions correspond to different quantization step-sizes for all 64 frames of the woman in a dress sequence. The results show that the SQNR computed for the motion vectors, increases (for the same bitrate) when the GFT is used to encode them with respect to the signal domain approach. Figure 26 (middle) plots the total coding rate for the geometry of the man sequence, when different coding rates are used to encode the motion vectors. This is performed using three solutions, the simple octree solution [38] (Intra coding mode), the dual (buffer) octree based algorithm [5] described in Section 2.6.1, and the proposed motion compensated dual octree algorithm [37] described in this section. Naturally for the first two solutions the bitrate to code the motion vectors is zero since motion vectors are not even estimated. For the last solution in red, different quantization step-sizes were employed, obtaining different coding rates for motion vectors. The lowest coding rate for the geometry part can be achieved for the proposed motion compensated dual (buffer) algorithm, by coding motion vectors at a rate of 0.1 *bpv*, outperforming all other solutions.

The colour attribute compression solution based on motion compensated for colour prediction was also assessed. The input data was organized into blocks of 16x16x16 voxels. Figure 26 (right) presents a graph which shows the average colour component PSNR for the R, G, and B components, with respect to the bitrate (*bpv*). This is computed for the first 10 frames of both man and woman sequences, and for two solutions, the independent coding [39], which is a static octree algorithm for colour coding, and the differential coding, which corresponds to the colour coding algorithm of the current solution. For every experiment (one for each sequence and coding solution), the quantization is performed with 5 different step-sizes and the results are shown in Figure 26 (right). From the results obtained, it is possible to conclude that the differential coding achieves a better performance especially for low bitrates. For higher bitrates, the differential coding performance does not increase significantly, while the other solutions' performance increase significantly and eventually approximates to the differential coding performance.

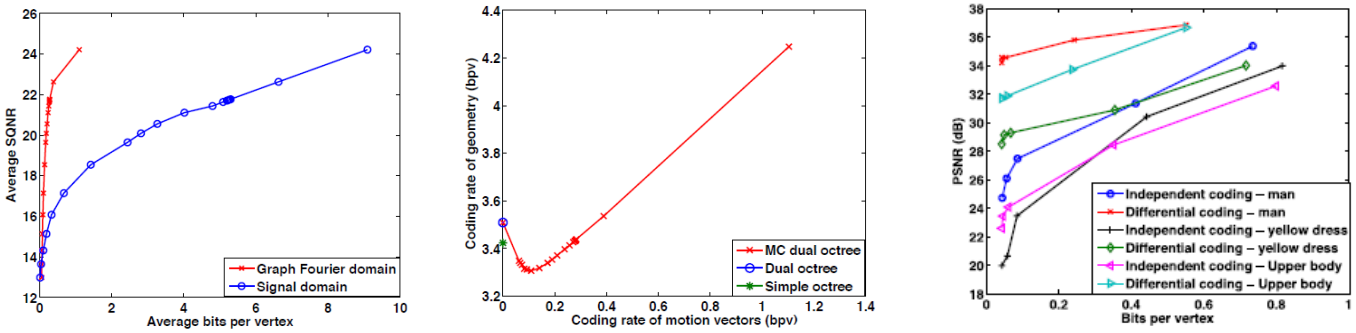

Figure 26: Left: Average SQNR obtained for motion vectors before and after GFT (man sequence). Middle: Coding rate used in 3D geometry (man sequence) for the simple octree [38], Dual octree [36], and MC octree [37] algorithms. Right: Colour attribute PSNR obtained for independent [39] and differential coding (man, woman, and upper body sequences) [37].

This solution has several advantages to compress dynamic point clouds by exploiting temporal correlations with efficient motion estimation and compensation tools. As shown in the experimental results, the proposed solution leads to efficient predictions and is able to efficiently code the motion vectors, which brings a significant increase of the overall performance of the point cloud coding system.

Chapter 3

3. Homography Based Point Cloud Codec: Architecture and Tools

This third chapter introduces the proposed point cloud coding solution, from here called Homography Based Point Cloud Codec, or simply HB-PCC, starting with a walkthrough of the full architecture, and describing the most important modules, i.e. modules that are rather relevant in the HB-PCC. One important reminder is the HB-PCC only codes the geometry of dynamic point clouds, leaving the remaining attributes, e.g. the colours, to be addressed in other related works.

3.1. Architecture and Walkthrough

The objective of this section is to introduce the HB-PCC, notably its architecture and walkthrough where the objective of each module is briefly presented. Figure 27 shows the encoding architecture for the HB-PCC. This architecture shall not only exploit the point cloud spatial and statistical redundancies but also the temporal redundancy in consecutive point cloud frames. Naturally, an Intra coding mode not exploiting temporal correlation is necessary to be used for the first frame and some periodic point cloud frames, notably to provide random access capacities.

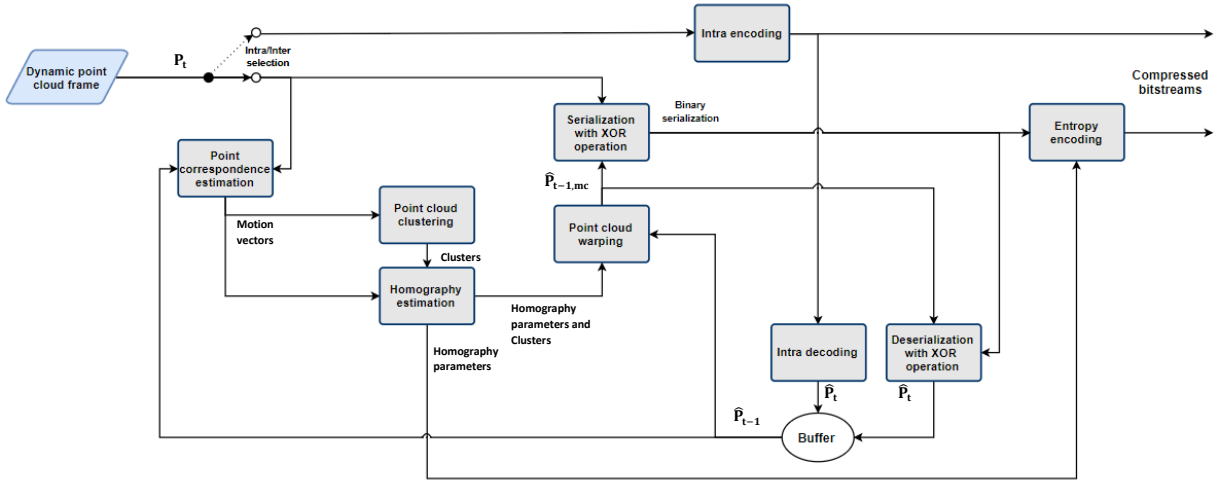

Figure 27: Encoder architecture of the HB-PCC.

The main objectives of the encoding architecture modules are:

- **Intra encoding:** The first frame, as well as some periodic frames from the point cloud sequence, are encoded in the so-called *Intra coding mode*, i.e. exploiting only the spatial and statistical redundancies. Here, the Intra coding mode consists in decomposing the point cloud into an

octree, where the occupancy of each node is described with one byte. After a consistent ordering of all bytes, the serialized octree is then entropy encoded and sent to the decoder. This Intra coding solution is the one proposed in [36] and available in PCL.

- **Point correspondence estimation:** This module is the first step of the Inter coding mode and has the key objective to estimate the motion between the target point cloud to be coded, P_t, and a reference, or previous, point cloud already coded and decoded, $\hat{P}_{t-1}$. The motion is estimated at single point level by matching a selected descriptor in the two point clouds. For each point in P_t, a correspondence, equivalent to a **motion vector,** is estimated to describe its trajectory from a point in $\hat{P}_{t-1}$, thus defining point correspondences for all points in P_t. A filter is used to remove some 'poor' correspondences, as it will be explained in the next section.
- **Point cloud clustering:** The previous point cloud, $\hat{P}_{t-1}$ is segmented into subsets (or parts) with a fewer amount of points, known as **clusters**. Thus, each cluster shall contain a set of points which represent a part of the point cloud, as this is important to obtain good predictions using the homographies estimated in the next step.
- **Homography estimation:** Since the correspondences (motion vectors) are costly to send directly from a rate point of view, for each cluster obtained above, a compact and more concise representation of the motion is computed using the correspondences. In this case, a geometric transformation, also called *homography*, is estimated for each cluster and its parameters are transmitted to the decoder.
- **Point cloud warping:** In this module, each cluster of the previous point cloud frame is warped to obtain a new *motion compensated* point cloud frame**,** $\hat{P}_{t-1,mc}$**,** which should be as close as possible to the target, to be coded, frame. This is performed by applying to each cluster the geometric transformation with the parameters estimated in the previous step.
- **Serialization with XOR operation:** This module uses the serialization with XOR scheme proposed in [36] to determine the residual after the point cloud warping. The residual between the target point cloud and the motion compensated point cloud represents the information that could not be predicted and is computed considering an octree dual buffer structure. This residual is serialized, coded and finally transmitted to the entropy encoder as in the PCL.
- **Entropy encoding:** The incoming residual symbol streams and warping parameters are entropy encoded to exploit the symbol statistics. The output bitstream is then sent to the decoder.

The decoder architecture is presented in Figure 28 and has the objective of receiving the compressed point cloud data from the encoder and reconstructing the corresponding decoded point cloud frame, essentially by performing the inverse of some encoder operations.

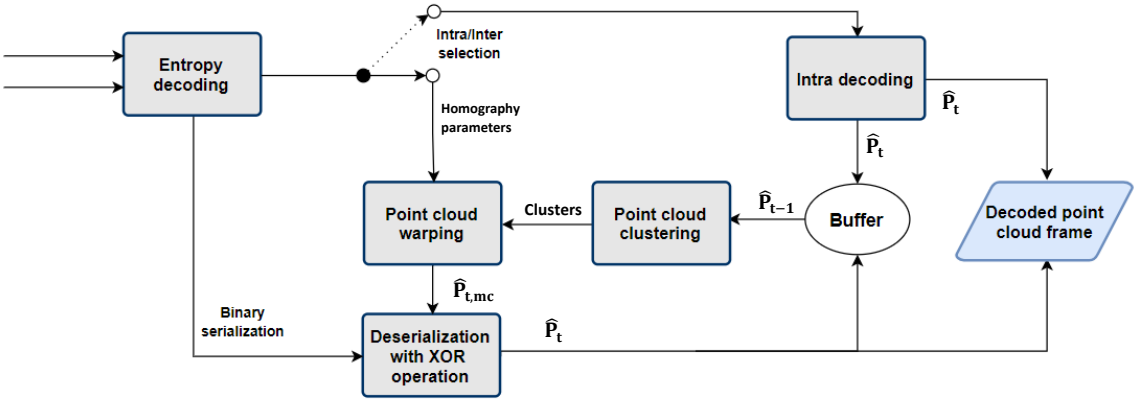

Figure 28: Decoding architecture of the proposed dynamic 3D point cloud coding solution.

In brief, the decoder modules perform as follows:

- **Entropy decoding:** The bitstream containing the point cloud data, generated in the encoder and transmitted, after exploiting the temporal redundancy, is now entropy decoded in the first module of the proposed decoder. The received sequence of bits is thus transformed into a sequence of symbols, notably the residual between the motion compensated and target point clouds.
- **Intra decoding:** For the first point cloud frame, and some periodic frames, the decoder uses the static point cloud decoding solution proposed in [36], thus obtaining the reconstructed point cloud.
- **Point cloud clustering:** The decoded reference frame gets decomposed into the same clusters as previously done at the encoder, when the Inter coding mode is to be applied. Each of these clusters has associated a set of homography parameters produced at the encoder that were transmitted to the decoder.
- **Point cloud warping:** Using the various sets of transmitted geometric transform parameters, (one set for each cluster) this module produces a motion compensated point cloud frame**,** $\hat{P}_{t-1,mc}$, by warping the positions of the previous point cloud, i.e. the last decoded point cloud, $\hat{P}_{t-1}$, to new positions; naturally, this process is performed for each cluster.
- **Deserialization with XOR operation:** This module applies the deserialization with XOR scheme proposed in [36] to recover the decoded point cloud. The inputs are the created temporal/warped prediction, this means the motion compensated point cloud frame, $\hat{P}_{t-1,mc}$, and the decoded residues which is basically a dual-buffer octree. The output is the decoded point cloud with some selected precision in terms of point geometry. This point cloud is stored in a buffer, as it will be used to estimate the point cloud prediction for the following point cloud frame based on the received warping parameters, should the Inter coding mode be chosen for the next frame.

In the next sections, only the most important modules of the proposed encoder will be detailed, namely the first three modules of the Inter Coding mode: Point correspondence estimation, Point cloud clustering, and Homography estimation. The remaining modules mostly consist in a direct use of operations available in the PCL and do not need a detailed description as they have been already presented in Chapter 2.

3.2 Point Correspondence Estimation

In this section, the Point correspondence estimation module is described, notably its objectives and overall approach first and, after, the various tools involved in more detail.

3.2.1 Objectives and Technical Approach

The Point correspondence estimation is one of the key modules in the Inter coding mode defined for the HB-PCC. Its main objective is to estimate the motion associated to the target point cloud frame in relation to the previous point cloud frame. This module plays a similar role to motion estimation in standard 2D video codecs. The architecture for this module is presented in Figure 29 and includes the following steps:

1. **Point cloud subsampling:** This first step is applied to the two point cloud frames adjacent in time, one being the target frame, i.e. the frame that is going to be encoded and transmitted, and the other the reference frame, i.e. the previous decoded frame, $\hat{P}_{t-1}$, which is available in a buffer at both the encoder and decoder. The subsampling has the objective of reducing the total number of points of each point cloud, resulting in less processing time for the next steps. Because the subsampling of a point cloud is performed by representing sets of close points by their centroid, this step may have influence on the results of the Point correspondence estimation, notably if too many points are removed from the original point cloud, thus modifying its geometry.
2. **Point descriptor extraction:** This step regards the characterization of a point cloud frame using a selected descriptor. In this case, a point-based descriptor, the Point Feature Histogram (PFH) or Fast Point Feature Histogram (FPFH), [41][42], is computed for all the points in both the subsampled previous and target point cloud frames. Because these point clouds typically represent objects, the extracted descriptors should be able to compactly describe the relevant object, or parts of it, in the two successive point cloud frames.
3. **Point descriptor matching:** The descriptor for each point in the target subsampled point cloud is compared with the descriptors of all the points in the previous subsampled point cloud or, to reduce the involved complexity, some selected points closer to the target point. The “best” match according to the adopted matching criterion, the Euclidean distance, between the descriptors, is selected. Since the two subsampled point clouds correspond to different instants in time of the same visual scene, the matching or correspondence defines a motion vector for each point of the target subsampled point cloud using as reference the previous subsampled point cloud.
4. **Correspondence filtering:** Prior to this last step, each point of the target subsampled point cloud has been matched with the point in the previous subsampled point cloud with the highest similarity as determined by the matching criterion. However, in the previous subsampled point cloud, some

of the points may have no matches, only one match, or several matches. Therefore, in this step, for all the points in the previous subsampled point cloud with several matches, the worse matches are removed leaving only one best match; in this case, the best match is the one corresponding to the lowest matching criterion value. This allows to have a more reliable set of correspondences which is essential for the homography estimation module of the encoding architecture.

5. **Full cloud correspondences computation:** Finally, to close the Point correspondence estimation module, for all the points that passed the filtering process and belong to a match between the two subsampled point clouds, the nearest point in the corresponding full point cloud is found. This means that for each match between the subsampled point clouds, a match in the original, full point cloud is obtained as the last step of this module to give more precision to the homography estimation process (although the number of correspondences is still the one determined using the subsampled clouds).

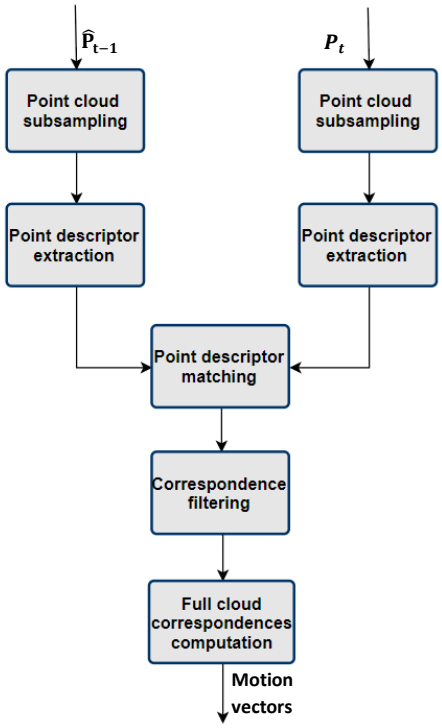

Figure 29: Point correspondence estimation module architecture.

3.2.2 Point Cloud Subsampling

In this first sub-module of the Point correspondence estimation module, a simple subsampling, or downsampling, of the target and previous decoded point clouds is performed. The main advantage of performing this subsampling is the reduction of the complexity and computational time of the subsequent steps in the Point correspondence estimation module. However, there is a drawback: the geometry of the subsampled point cloud will not be exactly the same as the original or full point cloud; in fact, the more points are removed in the subsampling process, the more details are lost. An example of subsampling can be found in Figure 30: the left image shows the full (original) point cloud of the so-called *Bunny*, available in [43]; the middle image shows a subsampled point cloud of *Bunny* with 50% of the original points, and the right image shows a subsampled point cloud of *Bunny* with 10% of the

original points. In the last image, the *Bunny* object is still visible but its quality is much worse comparing to the first image, due to the reduction of the number of points.

To subsample a point cloud an octree is needed, requiring a certain voxel resolution and a point cloud as inputs. The octree, as already detailed, divides the 3D space that contains the input point cloud into a regular grid of voxels with the given voxel resolution. Then, for each voxel, all the points from the input point cloud falling inside are approximated by a single point with the coordinates of the corresponding centroid, i.e. the average of the coordinates of all points in the voxel. The subsampling process has, therefore, a key control parameter, which is the size of the 3D voxel grid or the voxel resolution. When increasing the size of the voxels, more points will fall in each voxel, meaning that less points will be present in the subsampled point cloud.

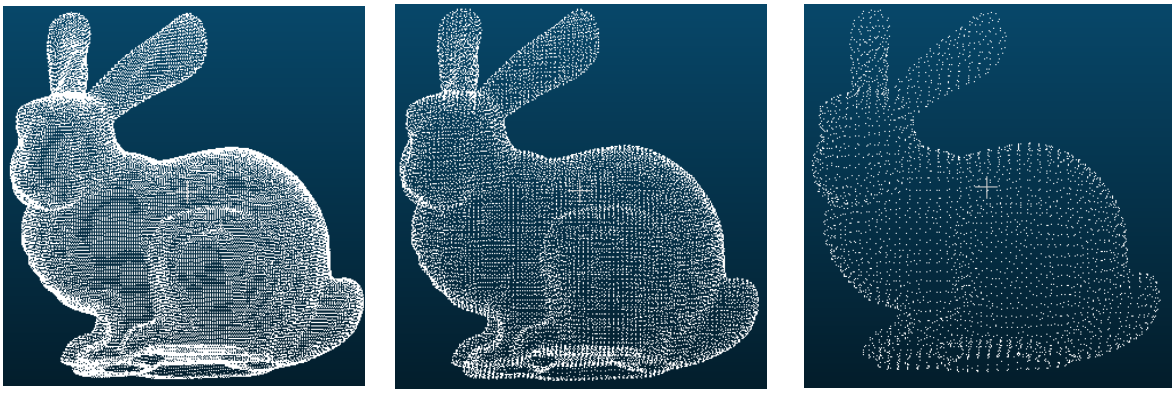

Figure 30: Left: Original Bunny point cloud [43]. Middle: Subsampled Bunny point cloud with 50% points. Right: Subsampled Bunny point cloud with 10% points.

3.2.3 Point Descriptor Extraction

The second sub-module of the Point correspondence estimation module has the objective of computing a descriptor for each point of the target and previous subsampled point cloud frames.

3.2.3.1 Overview of Point Cloud Descriptors

In the field of image processing, it is important to identify parts of the image containing key pieces of information, e.g. an edge, to facilitate some computational tasks, e.g. object recognition or registration. These are called *features*, and they can be represented through *descriptors*, which describe elementary content characteristics, e.g. associated to shape, normals and texture. 3D descriptors can be used to find correspondences in a pair of similar point cloud frames, e.g. created by some motion, and therefore be used in a dynamic point cloud coding solution. However, 3D descriptors still need a lot of improvement as the current options are based on critical trade-offs between computational effort and accuracy and descriptiveness; this area of research has been very active in the last few years.

The point cloud descriptors available in the literature can be classified in two main types: the so-called *global descriptors*, which characterize the whole point cloud with a single descriptor; and the so-called *local descriptors*, which characterize a small region or even a single point, in a point cloud. Table 15 included in Appendix A provides an extensive summary of available point cloud descriptors. Among

these descriptors, the PFH, FPFH and PCE (Principal Curvatures Estimation) [44], describe each single point of the point cloud, rather than the whole point cloud or regions of the point cloud.

3.2.3.2 Point Feature Histogram (PFH) Extraction

A Point Feature Histogram descriptor for a specific point is obtained by accounting all the relationships between the points in its k-neighborhood and their estimated surface normals. The objective is to capture the sampled surface variations by accounting all the interactions between the directions of the estimated normals; in this context, the quality of the descriptor should be rather dependent on the quality of the surface normal estimations at each point. More specifically, the PFH descriptor extraction involves the calculation of three angles between three computed vectors, for every pair of points in a k-neighbourhood around a query point (this means the point being described); these angles are structured in a histogram which defines the descriptor. The sequence of steps involved in obtaining the PFH descriptor are shown in the left image of Figure 31, and a visual representation of the angles computed in the descriptor is shown in the right image, to be detailed later in this section.

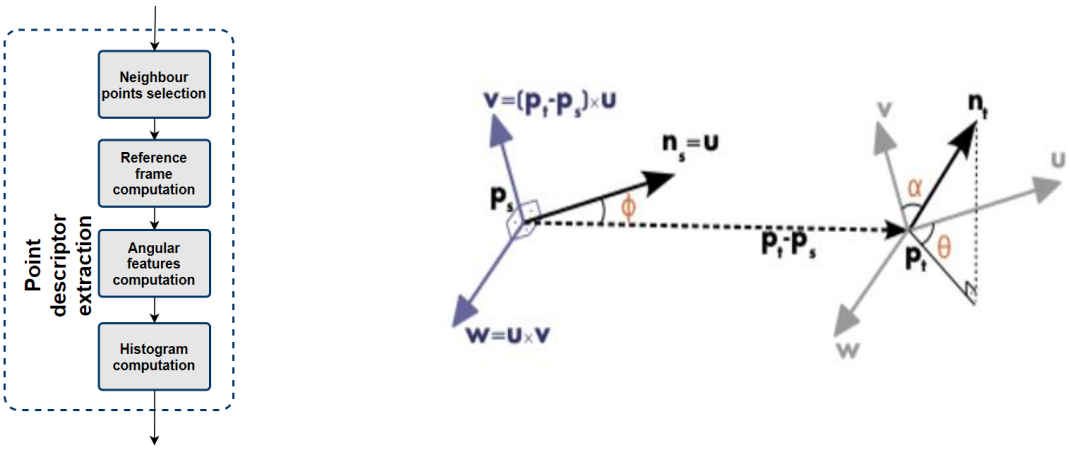

Figure 31: Left: PFH point descriptor extraction sub-module architecture. Right: Example of vectors and angles computed for a pair of points, according to the PFH descriptor [41].

For each point in the target and previous subsampled point clouds, the corresponding PFH descriptor is obtained as detailed in the following sequence of steps:

- **Neighbour points selection:** The first step consists in selecting the neighbouring points within a certain radius, or the **nearest k-neighbours**, for the query point in the point cloud, by performing a k-dimensional, k-d tree search. The number of nearest neighbours, or the so-called *radius size*, will be important for the quality of the descriptor as a small neighbourhood is not enough to contrast the query point, but a big neighbourhood may also make the descriptors very similar.
    - The k-d tree algorithm is a useful space-partitioning tool that allows to speed up some operations, e.g. finding the closest neighbours in the space around a certain point [46]. For point clouds, a k-d tree is a 3D (k=3) binary tree, including all the 3D points organized in a tree. At each level of the k-d tree, the space is divided by splitting each node into two children nodes (tree branches) in a specific dimension, i.e. x, y and z planes. Each children node corresponds to the median point in the splitting plane, thus forcing half the points to be located after the

split point, and the other half before the split point. Figure 32, left, shows how a k-d tree can organize the 3D points in the point cloud, where the root level of the k-d tree splits the space along the y dimension (red cut), the next level of the k-d tree splits each voxel (space subdivision obtained) along the x dimension (green cuts), and the following level of the k-d tree cuts all obtained four voxels along the z dimension (blue cuts).

  - The nearest search operation is performed starting at the root tree node and comparing the current node's point coordinates with the query point. Each node is divided into two branches, one containing all the nodes with greater value than the "father" node for the current cut plane, and the other branch having all nodes with smaller value; this enables a fast search for neighbours in the k-d tree.

- **Reference frame computation:** For each pair of points inside the k nearest neighbourhood of the query point, three vectors ($\vec{u}$, $\vec{v}$ and $\vec{w}$), corresponding to the so-called *reference frame*, are computed as follows:
  - First, a normal is obtained for each query point (here "normal" refers to the perpendicular vector of a point, and not the three normals computed for this descriptor), by using the k-nearest neighbours of the query point. This normal is an important property of a surface and may be estimated by simply computing the perpendicular vector to the underlying surface of a point cloud (based on the neighbouring points) for each selected query point.
  - For each pair of two points in the k-neighbourhood of the query point, three vectors are computed using (11), (12) and (13). First, it is established a source point, p_s,with a normal $\vec{n}_s$, by finding the point in the pair of points which has its normal closer to a line that connects the two points. The other point, the target point, p_t, has a normal $\vec{n}_t$. The first vector, $\vec{u}$, is simply the normal of the source point. The second vector, $\vec{v}$, is obtained by computing the difference between the coordinates of the source and target points, and multiplying by the vector $\vec{u}$. Finally, the third vector, $\vec{w}$, is the multiplication of the vectors $\vec{u}$ and $\vec{v}$. In Figure 31, right, the three vectors ($\vec{u}$, $\vec{v}$ and $\vec{w}$) computed in (11), (12) and (13) are shown for the source and target points. These vectors will be used as a reference frame.

$$\vec{u} = \vec{n}_s \tag{11}$$

$$\vec{v} = (p_t - p_s) \times \vec{u} \tag{12}$$

$$\vec{w} = \vec{u} \times \vec{v} \tag{13}$$

- **Angular features computation:** Using the three vectors previously computed ($\vec{u}$, $\vec{v}$ and $\vec{w}$) as a reference coordinate frame (axis), for each pair of points, three angular features (α, ϕ, θ) are now computed using the equations (14), (15) and (16). In Figure 31 right, it is possible to visualize the three angles, where the reference coordinate frame is represented in grey.

$$\alpha = \vec{u} \cdot \vec{n}_t \tag{14}$$

$$\phi = \frac{\vec{u} \cdot (p_t - p_s)}{d} \tag{15}$$

$$\theta = \text{atan}(\vec{w} \cdot \vec{n}_t, \vec{u} \cdot \vec{n}_t) \quad (16)$$

- **Histogram computation:** For every pair of points, a triplet (α, ϕ, θ) is constructed and used to calculate a histogram of triplet values. Therefore, a histogram with 125 bins is created by accumulating the occurrences for every combination of the three angular features, each one characterized with 5 bins, thus $5^3 = 125$ bins. A bin is simply an interval of values, implying that 5 bins correspond to dividing the whole range of every angular feature into 5 equally-sized intervals. Therefore, each feature value range is divided into 5 bins, and the number of occurrences in each bin counted. An example of a PFH histogram is shown in Figure 32, right, where the percentage of occurrences is plotted for each bin.

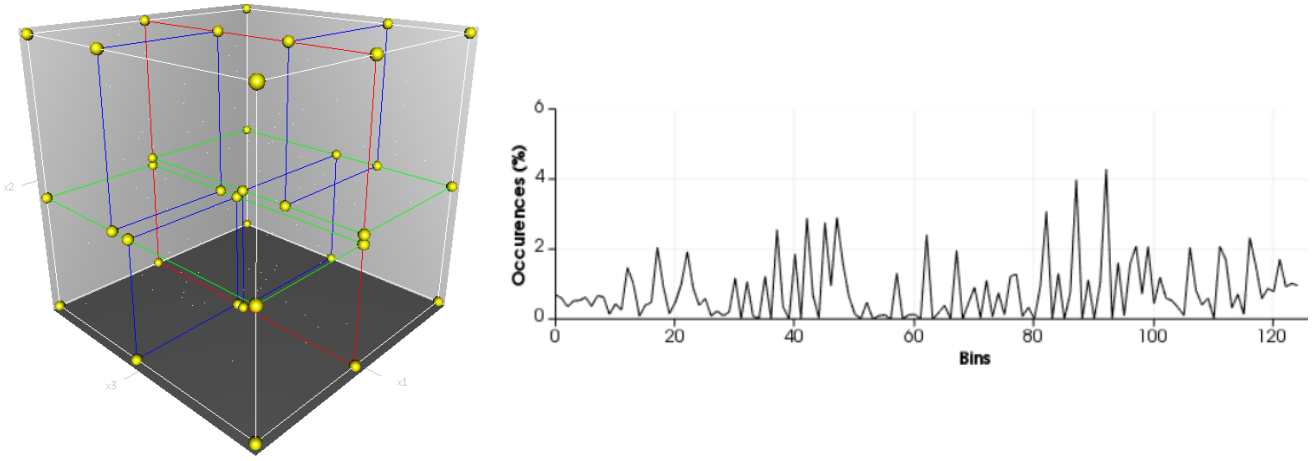

Figure 32: Left: K-d tree space partitioning [46]. Right: PFH descriptor histogram example.

3.2.3.3 Fast Point Feature Histogram (FPFH) Extraction

The FPFH descriptor, which is a less complex version of the PFH descriptor, has the objective to capture the geometrical properties of each cloud point while reducing the PFH descriptor complexity. This is achieved by simply reducing the number of pairs of points chosen for each k-neighbourhood and the number of bins in the final histogram. As the FPFH descriptor is based on the PFH descriptor, only the differences will be detailed in the following. For each point, the descriptor is extracted with the following sequence of steps:

- **Neighbour point selection and normals estimation:** The nearest k-neighbours and the corresponding points normals are obtained in the same way as for the PFH descriptor.
- **Simplified Point Feature Histogram (SPFH) computation:** For each query point, the Simplified Point Feature Histogram (SPFH) is computed as in the PFH descriptor by obtaining the same three angular features, but using now only the pairs of points in the k-neighbourhood containing the query point, and only using $3 \times 11 = 33$ bins, to reduce the computational time for matching and increase the descriptor compactness.
- **Histogram computation:** After computing the SPFH for all points in the point cloud, this last step computes the histogram for each point in the point cloud, by weighting the SPFH values of each neighbouring point of the query point *p* according to (17), where p_k is the

neighbour *k*, and w_k is the Euclidean distance between the query point and the neighbouring point *k*.

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_k}SPFH(p_k) \quad (17)$$

Comparing to FPH, the number of histogram bins is reduced from 125 to 33, thus reducing the size of the new descriptor. An example of a FPFH histogram is shown in Figure 33, where the percentage of occurrences is plotted for each bin.

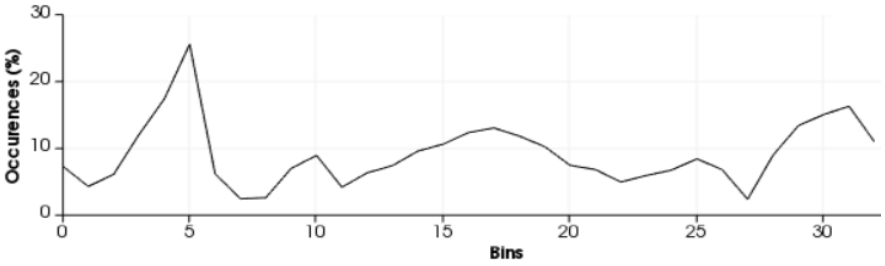

Figure 33: FPFH descriptor histogram example.

In [45], the author compares the descriptors listed in Table 15 in terms of accuracy for an object recognition task, to conclude that the PFH descriptor provides the overall best results in several recall-precision curves, if colour data is not available. The FPFH descriptor has worse performance compared to the PFH descriptor, however it has a lower computational complexity. These two descriptors characterize each point by capturing the geometrical properties of the point through a multi-dimensional histogram of values. Both are robust to noise and relatively low complex [41][42]. They have good descriptive power, as they characterize the geometry information around each point and can be used to establish if the point resides on an edge, cylinder, plane, etc. These descriptors are also robust to scaling and pose variations in the point cloud. While the PFH descriptor is more time-consuming, the FPFH descriptor has less descriptive power. These two descriptors are eligible, however only one, FPFH, will be applied in this Thesis, due to the relatively low computational time.

3.2.4 Point Descriptor Matching

Using the PFH or FPFH descriptors computed as mentioned in the previous step, the correspondence for each point in the target subsampled point cloud regarding the previous subsampled point cloud has to be determined. With this purpose, the descriptor of each point in the target subsampled point cloud and the descriptor of each point within a certain window, in the previous subsampled point cloud are compared, and a correspondence is obtained by choosing the most similar point according to a distance metric. The size of this *search window* corresponds to a new parameter. Each step is detailed next:

- **Nearest neighbours search:** For each point in the target subsampled point cloud, the nearest neighbours in the previous subsampled point cloud are obtained with a k-d tree search. This step reduces the overall complexity because the correspondence search will be performed using a tree structure. Because the search is only performed in the neighbouring points, and not all points of the

previous subsampled point cloud, it is guaranteed that there are no correspondences that are far away from the original point. This “window search” in the previous subsampled point cloud makes sense, because it is unlikely that a point travels a big distance between two consecutive frames.

- **Euclidean distance computation and matching:** The PFH or FPFH descriptors for each point in the target subsampled point cloud are then compared to the corresponding descriptors of the neighbouring points in the previous subsampled point cloud, using the Euclidean distance between the descriptors. The neighbouring point with the lowest Euclidean distance is taken as the correct correspondence, i.e. a correspondence is obtained between the target point cloud and the previous subsampled point cloud, for each point in the target subsampled point cloud.

3.2.5 Correspondences Filtering

The correspondences previously obtained that create trajectories between the points of the previous and target subsampled point clouds are now filtered. Following the point descriptor matching process, it is possible that several points in the target subsampled point cloud corresponded to a same, single point in the previous subsampled point cloud, which is not good for homography estimation. Thus, at this stage, all the correspondences that have the same point in the previous subsampled point cloud and a different point in the target subsampled point cloud are inspected and only one correspondence, the one with lowest Euclidean distance is kept. This guarantees that each point in the previous subsampled point cloud does not have multiple correspondences to different points in the target subsampled point cloud.

3.2.6 Full Cloud Correspondences Computation

After filtering the correspondences, the nearest neighbour of the points in the subsampled point clouds are searched in the corresponding previous and target full point clouds. This means that every selected match in the subsampled space is used to infer the corresponding match in the full cloud space. Thus, the homography estimation can be performed with full cloud precision points. Naturally, the number of correspondences in the full cloud match the number of correspondences detected in the subsampled clouds, i.e. not every point the full point cloud will have a correspondence.

3.3. Point Cloud Clustering

This section starts by describing the objectives of the Point cloud clustering module and briefly reviewing the types of clustering techniques currently available in the literature to conclude with a description of the specific clustering tool used in the HB-PCC.

3.3.1 Overview of Clustering Methods

After the computation of the final correspondences between two successive point cloud frames is performed, it is important to define in the previous point cloud smaller 3D regions. This is essential as it lowers the computational time of the next module, the homography estimation, since it is faster to estimate an homography for a smaller set of points. Also, if the clusters are small enough, most of the points in each cluster should undergo a similar motion, meaning more accurate homographies can be

obtained and therefore a more accurate motion compensated point cloud (prediction) is achieved than if large regions of the point cloud were instead used. Therefore, the residual energy that must be transmitted to the decoder will be lower and better RD performance can be obtained. Naturally, assuming one homography transformation is needed per cluster, there is a quality-rate trade-off to pay, since the homography parameters need to be transmitted to the decoder and this requires some rate. The cost of using a high number of clusters, and sending the corresponding homography parameters, might offset the gains that are achieved with the increased accuracy of the prediction. Thus, at some stage, it may be beneficial to send a higher residual at the cost of less homography parameters by using a lower number of clusters for the overall point cloud.

This module is very important and has a high impact on the overall compression performance of the dynamic point cloud. According [47], there are several approaches for point cloud clustering, which are detailed next:

- **Edge based methods:** This approach focus on detecting the boundaries of each cluster, i.e. the points that form the contour of each cluster. This is achieved by detecting fast variations of some point features, e.g. the directions of the normals. Because these methods are based on comparing neighbouring points, they will have a fast performance; however, they underperform when used in noisy and uneven density point clouds.
- **Region based methods:** By grouping neighbouring points with similar properties, these methods create regions, or clusters, where a small region is grown by successively merging similar points (bottom-up). Also, a large region can be divided into smaller regions (top-down), by finding dissimilarities between them. These two main ways of performing region based clustering are described below:
    - **Seeded-region methods (bottom-up):** Several initial seed points are chosen using some criterion, e.g. the curvature, and the remaining points are merged with the defined seeds or previously clustered data based on another criterion, e.g. proximity, forming the several regions. These methods are susceptible to noise, time consuming, and can cause under-segmentation, e.g. if a single region includes several objects.
    - **Unseeded-region methods (top-down):** This method starts by grouping all points in a single region, and then recursively subdivides it into smaller regions, based on some criterion and a division threshold. Sometimes over-segmentation may occur, meaning that an object gets subdivided into several regions. Because of this, these methods do not always perform well, and usually require prior object knowledge, to help in the subdivision process.
- **Attributes based methods:** The first step of this type of methods is to compute or exploit available attributes for the point cloud to achieve a high quality segmentation. The second step is the clustering of the point cloud into clusters with similar attributes. These methods usually have a great accuracy, meaning that the clusters will represent objects without under/over segmentation, at the cost of an increased complexity.
- **Model based methods:** Geometric shapes are used to cluster the point cloud in regions, where all the points in the same region have some relation between them (e.g. every point lies on a

plane). In other words, the points associated to a specific cluster are correlated and with a same mathematical representation; this can be achieved using algorithms like Random Sample Consensus (RANSAC). These methods are robust to outliers and but for complex point clouds they do not produce a good clustering result.

- **Graph based methods:** In this class, a graph is created from the point cloud. Each point is a vertex and is related to the neighbouring points with edges (connections). These methods allow the clustering of complex scenes, i.e. scenes with many different objects, with great accuracy, with the trade-off of being time consuming, or needing offline training steps.

3.3.2 Supervoxel Clustering

Table 16 in Appendix B lists the clustering techniques that are available in the PCL. These techniques are associated to the clustering types presented above and have different strengths and weaknesses, thus typically targeting different applications; for example, the conditional Euclidean clustering technique is an attribute based clustering method, and the region growing is a region based clustering method.

In the HB-PCC, the previous point cloud is divided in clusters, where each point cloud cluster should contain a set of neighbouring points following the same motion to obtain good predictions. Unfortunately, none of the PCL available clustering techniques can directly perform clustering based on correspondences and so the best technique was selected. After detailed analysis, it was decided to select as clustering tool the so-called *Supervoxels* clustering technique available in PCL [48]. This clustering technique groups neighbouring points into clusters where each cluster, called *supervoxel,* has neighbouring points with similar normals and colours.

The architecture of the proposed Point cloud clustering module is shown in Figure 34; it includes the selected Supervoxel clustering technique, divided in a few submodules, and a final clustering refinement tool. The adopted Supervoxels clustering technique is well suited for the HB-PCC as it can easily generate any number of clusters, that are evenly distributed in the point cloud, and where each cluster contains a similar amount of neighbour points with similar normals between them. The last submodule, cluster refinement, is responsible for dealing with some points that disappear when using the Supervoxel clustering technique, i.e. some points are removed by this technique as they are considered noise. Therefore the objective of this last submodule is to use the information about the clusters obtained in the Supervoxel clustering technique to form clusters without removing any points of the previous point cloud.

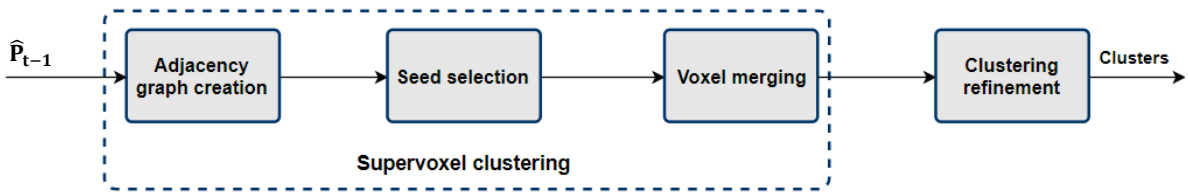

Figure 34: Point cloud clustering module architecture.

The Point cloud clustering technique uses as input, the previous decoded point cloud that is available at both encoder and decoder and therefore avoids the transmission of any additional side information that would be necessary to define clusters in the target point cloud. The four modules in the architecture presented in Figure 34 work as follows:

- **Adjacency graph creation:** A 26-adjacency graph is obtained from the previous decoded point cloud, using a k-d tree search, after dividing the 3D space in a grid of voxels using a certain selected resolution, the so-called *voxel resolution*, R_{voxel}. Each node in the adjacency graph, associated to an occupied voxel in the point cloud, has at most 26 connections, 6 possible connections on the faces, 12 possible connections on the edges, and 8 possible connections on the vertices.
- **Seed selection:** The seeds are the initial points picked to define each cluster (or supervoxel). The clusters are simply obtained by grouping the remaining ungrouped voxels with a specific seed, with this association made based on a selected distance metric. In this Supervoxel clustering technique, the seeds are obtained by dividing the 3D space into *seed voxels* using a certain resolution, the so-called *seed resolution*, R_{seed}, which is a parameter of the HB-PCC. At this stage, one seed is picked for each *seed voxel* in the grid, by finding the voxel closer to its center. Naturally, this implies that the *voxel resolution* must be higher than the *seed resolution*, $R_{voxel} > R_{seed}$, meaning that each *seed voxel* should include several voxels. An example of changing the *seed resolution*, for a specific *voxel resolution*, is presented in Figure 35, where the *seed resolution* is increased from left to right. A filter is then used to remove the seeds that are considered noisy points, i.e. that are almost isolated in the 3D space. In Figure 36 (left), two seeds are represented as small cubes, the *voxel* and *seed resolutions* are represented with different radii, and the green plane is used to search for neighbouring voxels to filter isolated seeds.
- **Voxel merging:** At this stage, occupied voxels that are not seeds are merged into the clusters using the following process:
  - First, for every seed, a distance metric, equation (17), is computed to its neighbouring voxels, and each of those neighbouring voxels is grouped with the seed whose computed distance is the shortest.
  - After, for every seed, the distance to the neighbours of the voxels that were previously merged is computed, and those voxels are then grouped with the seeds whose computed distance is the smallest. This step is successively repeated for each seed and stops when the neighbouring voxels have a distance larger than $\sqrt{3}R_{seed}$, or when all neighbouring voxels are already merged in clusters. Voxels with a distance greater than $\sqrt{3}R_{seed}$ from all seeds are removed and will not be added to any cluster.

  This process corresponds to a breadth-first search, shown in Figure 36 (right), where each level of the adjacency graph is checked for every seed, before going to the next level. The distance metric to be used is defined in (18)

$$D = \sqrt{w_c D_c{}^2 + \frac{w_s D_s{}^2}{3 R_{seed}{}^2} + w_n D_n{}^2} \quad (18)$$

where w is a configurable weight for each of the 3 components, D_i is a distance for each component and R_{seed} is the *seed resolution*. The indices *c*, *s* and *n* refer to the 3 separate components, colour, spatial (based on Euclidean distances) and normals, respectively. The colour distance is computed in a normalized RGB space, the spatial distance computed with the x,y,z coordinates, and the normal distance is computed as an angle between surface normal vectors. Considering the objective proposed in this Thesis which targets only the coding of the geometry of dynamic point cloud, the colour weight is set to 0.

- **Cluster refinement:** In the Supervoxel clustering technique, the input point cloud is divided into clusters; however, some voxels are considered noise and therefore some points of the point cloud are removed. In this context, this submodule is responsible to refine the clusters in a way that every point, from the input point cloud is clustered. The process occurs as detailed next:
  - Using the Supervoxel clustering technique, the clusters, which all together have less points than the input point cloud, are obtained.
  - For every point in the previous point cloud, a nearest neighbour is searched in all the clusters obtained with the Supervoxel clustering technique. The point in the previous point cloud is then merged to the cluster including its nearest neighbour.

  With this Cluster refinement submodule, the clusters obtained in the previous submodule stay the same, and the points that were removed in the process are included in the corresponding closest cluster.

Figure 35: From left to right, the seed resolution is increased resulting into a higher number of clusters.

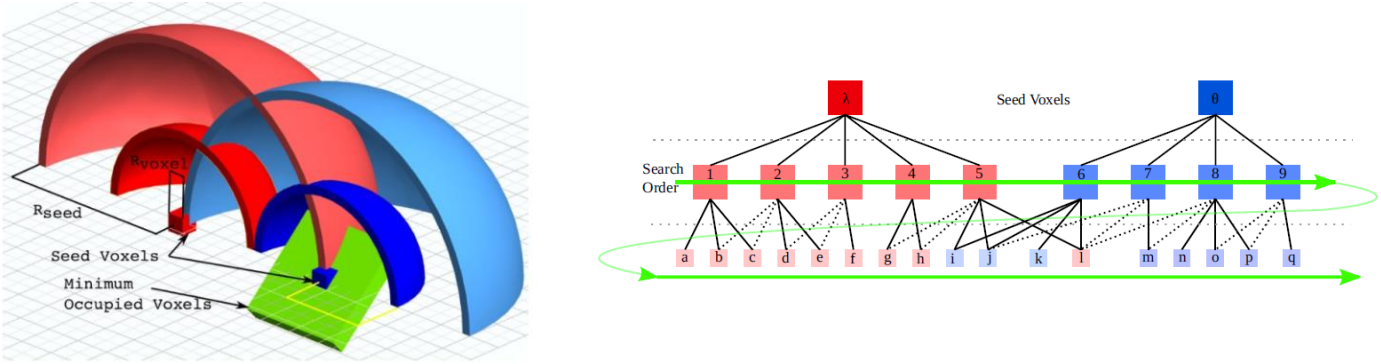

Figure 36: Left: Example of a 3D space with a seed resolution, a voxel resolution, and two seeds [48]. Right: Example of a breadth-first search in an adjacency graph [48].

3.4. Homography Estimation

An homography is a geometry transformation that can be used to describe complex motions such as zooms and perspective deformations, as well as more basic motions such as rotations and translations. The objective of this module is to compute the homography parameters of the point cloud warping process for each obtained cluster; in practice, an homography matrix is determined, which will be used to obtain the points of the estimated target point cloud given the points of its corresponding previous, decoded point cloud. Using a single homography matrix it is possible to represent the correspondences (motion vectors) for each cluster. The homography matrix is also a compact representation of the motion and requires much less information to describe the motion between consecutive clusters in the point cloud frames when compared to the motion vectors (for each point); since this data must be coded and transmitted to the decoder it is a rather important feature.

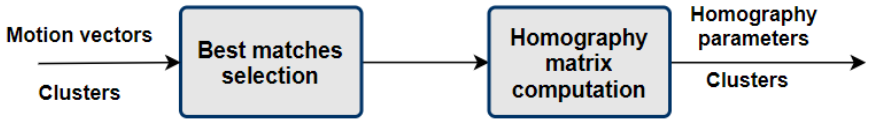

Figure 37: Homography estimation module architecture.

The architecture of this module, presented in Figure 37, is detailed in the following steps:

- **Best matches selection:** The inputs of this module are the clusters defined for the previous decoded point cloud and their correspondences to the target point cloud, and a parameter: the number of best matches. Following the process presented before, only some points in the clusters have a correspondence, and some of these correspondences might be more accurate than others. The idea of this step is to find the best matches, or correspondences, per cluster, and eliminate the remaining ones. To achieve this, the following steps are executed:
  - For each correspondence, a Euclidean distance is computed between the two descriptors that were matched; In fact, this distance was already computed in Section 3.2.4. This

distance considers the differences between all the bins of the two descriptors associated to each point of the correspondence.

- The correspondences are sorted based on the distance computed in the previous step, and the ones with lowest distance are simply chosen. The remaining correspondences are thus eliminated.
- By using only the best correspondences for each cluster, the homography estimation can achieve better prediction results. Should the cluster have less correspondences than the parameter containing the number of best matches, all available correspondences in the same cluster are used. The reason why the best matches selection is applied within each cluster, instead of the whole previous point cloud, is that every cluster needs a minimal number of correspondences to estimate the homography parameters, which cannot be guaranteed if the best matches were decided at the whole point cloud level.

- **Homography matrix computation:** In this second step, a homography transformation, or matrix, is computed for each cluster. An example of such matrix can be found in (19). The matrix in (19) can be decomposed in a rotation matrix, represented from r_1 to r_9 and a translation vector, represented from t_1 to t_3. The homography estimation algorithm available in the PCL uses a singular value decomposition (SVD), which is a widely-used method to find a rigid transformation between two sets of points, using known correspondences. This algorithm uses as inputs the points in the previous point cloud, the points in the target point cloud, and the correspondences between the two sets of points and works at the cluster level. The output is a set of homography matrices, each one corresponding to a cluster of the previous point cloud.

$$\boldsymbol{H} = \begin{vmatrix} \boldsymbol{r_1} & \boldsymbol{r_2} & \boldsymbol{r_3} & \boldsymbol{t_1} \\ \boldsymbol{r_4} & \boldsymbol{r_5} & \boldsymbol{r_6} & \boldsymbol{t_2} \\ \boldsymbol{r_7} & \boldsymbol{r_8} & \boldsymbol{r_9} & \boldsymbol{t_3} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{vmatrix} \tag{19}$$

As a final remark, only the homography parameters, $r_1 \ldots r_9$ and $t_1 \ldots t_3$ are sent to the decoder uncompressed, since the last line is constant. The impact of not compressing these parameters is discussed in the next chapter. The remaining modules in the architecture of the HB-PCC were implemented by simply using operations available in the PCL and, therefore, do not need a detailed description. The walkthrough in Section 3.1 should be enough to understand what is done and what is achieved in those modules.

Chapter 4

4. Performance Assessment

In this chapter, the Homography Based Point Cloud Codec (HB-PCC) is assessed and compared with the most relevant point cloud coding solution available, the PCL codec [36]. As the HB-PCC is an *upgrade* of the PCL codec, notably including a temporal prediction stage where a motion compensated frame replaces the previous frame used as (simple) temporal prediction in the PCL Inter codec, a first assessment is performed in terms of the temporal prediction error; that is, the motion compensated frame and the previous frame are used as predictions with the objective of evaluating how good are these predictions for the target frame. Finally, the RD performance of the HB-PCC solution and the PCL Intra and Inter codecs will be compared with the objective of assessing which one is the most efficient coding solution. However, before the assessment itself, this chapter starts by presenting the test conditions, notably the test datasets, the benchmarks, the coding conditions and the metrics used to perform the assessment, followed by the description of the HB-PCC main parameters optimization.

4.1. Test Material, Conditions and Benchmarks

As mentioned in Chapter 1, MPEG has recently launched a Call for Proposals on Point Cloud Compression [49], targeting to define an efficient codec for point clouds. This call for proposals clearly defines the test material and coding conditions that will be used to compare the submitted proposals, and so it makes sense to use some of the recommended datasets and coding conditions to assess the HB-PCC solution and the PCL benchmarks. The datasets are sequences of high-quality point clouds, publicly available, and the coding conditions are set to guarantee that the codecs are compared for the same levels/ranges of compression and quality. Finally, the metrics used to evaluate the codecs consist in the root mean square (*rms)* error between the original and decoded point clouds and the PSNR to assess the quality of the decoded point cloud against the corresponding original point cloud.

4.1.1 Point Cloud Datasets

To assess the HB-PCC and the PCL codec, three datasets were selected out of five datasets used in the MPEG Call for Proposals; these datasets are available in [50]. All the selected datasets, in this case dynamic point clouds corresponding to a sequence of point cloud frames, show captured or computationally generated point clouds of people with some amount of motion. While this is a rather

restrictive content selection, as only one specific situation, the natural motion of the human being, is assessed, all the five MPEG datasets are rather similar. The three datasets selected for performance assessment in this Thesis are, see Figure 38:

- ***Redandblack*:** This dataset is a dynamic point cloud, in this case a voxelized point cloud, i.e. the points in the point cloud are constrained to a regular 3D grid, where each division corresponds to a voxel [51]. The spatial resolution of the point cloud is $1024\times1024\times1024$ voxels, leading to an octree depth of 10 ($2^{10} = 1024$). The largest point cloud dimension is the height, which is approximately 1.8 *meters* tall, so each voxel has an approximated size of 1.75 *mm,* ($1.8m \div 1024$). Table 2 shows the main characteristics of this dataset, namely the type of motion, the number of frames and frame rate, and the average, maximum and minimum number of points in the point cloud frames. The total duration of this sequence is 10 *seconds* ($300 \div 30 = 10$). This dataset corresponds to a woman with a red dress, and the motion in the sequence contains two rather different periods: first, the woman rotates the body and lifts up the dress, and then she moves up her arms. The dress and the hair have a wavy nature, which is likely to create unpredictable motion in those regions. Two *Redandblack* frames are shown in Figure 38, left.
- ***Soldier*:** This dynamic point cloud is very similar to the *Redandblack* dataset, notably in terms of spatial resolution, voxel resolution, frame rate, number of point cloud frames, and duration [51]; those characteristics can be found in Table 2. This dataset has less motion than the *Redandblack* dataset and is characterized by a long period where the soldier talks on the radio with little motion, and a shorter period where the soldier rotates the body and points the gun, with medium motion. Two different *Soldier* frames are shown in Figure 38, middle.
- ***Queen*:** Differently from the two previous datasets, which were captured from real humans, this dataset was computationally generated, and has some different characteristics in terms of motion behaviour; again, Table 2 shows the most relevant properties. This dataset contains a woman moving her arms, with an object on one hand. A main difference to the two previous datasets is in the motion, synthetically generated, with a first part where the woman moves her arms slowly and a second part with a constant horizontal rotation of her body and again a slow movement of her arms. Figure 38, right, shows two *Queen* point cloud frames.

Table 2: Characteristics of the selected point cloud datasets.

Dataset	Type of motion	Number of frames	Frame rate (Hz)	Average number of points	Largest number of points	Lowest number of points
Redandblack	High	300	30	727 289	816 879	640 070
Soldier	Medium	300	30	1 075 517	1 103 974	1 044 513
Queen	Low	250	25	1 002 411	1 014 959	996 668

Figure 38: Left: Redandblack frames numbers 20 and 130. Middle: Soldier frames numbers 20 and 207. Right: Queen frames numbers 120 and 230.

4.1.2 Coding Benchmarks

To evaluate the HB-PCC performance, some reference performance must be used, in this case the PCL codec's performance is considered, as it is the most relevant coding solution available in the literature. Thus, the PCL codec is going to be used as a benchmark, both in terms of *rms* error (*rmse)* and RD performance. In practice, three different codecs will be considered:

- **PCL Intra codec:** This is the original PCL codec [36], developed for static point cloud compression, and also used as the Intra coding mode in HB-PCC, this means for the point cloud frames which are not coded exploiting the temporal correlation. Although it only exploits spatial redundancies, by obtaining the octree of a point cloud and entropy encoding its binary serialization, some spatial redundancy is exploited, thus achieving some relevant compression regarding the non-compressed format.
- **PCL Inter codec:** The PCL Inter codec [36] adds an Inter coding mode to exploit the temporal redundancy, which consists in detecting the changes in terms of voxel occupation of two consecutive octrees (with a XOR operation), and entropy encoding the XOR output stream. Typically, the first frame and some periodic frames are encoded with the Intra coding mode, this means using the PCL Intra codec.
- **HB-PCC:** The solution proposed in Chapter 3, which is basically an upgrade of the PCL Inter codec, tries to better exploit the temporal redundancy by using a homographic transformation, thus leading to a motion compensated point cloud more similar to the target point cloud. Then, the same XOR operation as used for the PCL Inter codec is applied between the target and the motion compensated point clouds. Because the motion compensated point cloud is more similar to the target point cloud than the previous point cloud (used in PCL Inter), less changes should be detected between the two octrees, and thus a more efficient compression is expected.

4.1.3 Coding Conditions

As for the coding conditions, they are the same for the three codecs evaluated, except for the specific HB-PCC parameters, which are naturally specific of that codec. The coding conditions applied to the set of selected point clouds are:

- **Geometry only:** In the scope of the proposed HB-PCC solution, only the point cloud geometry is coded, leaving the remaining attributes, e.g. the colour, to be addressed in future work.
- **Group of pictures (GOP) size:** In a sequence of point clouds, the group of pictures (GOP) defines the number of consecutive point cloud frames that are encoded in Inter coding mode, and thus cannot be individually decoded to grant random access, like in 2D video. The chosen GOP sizes were 2 and 8; a GOP size of 2 implies that the incoming point cloud frames are alternately coded with the Intra and Inter coding modes.
- **Target point cloud voxel resolution:** To define the RD performance curves shown in Section 4.4, the target resolution for the coded point cloud can be specified, and consists in changing the size of the octree voxels for the coded point cloud. Six voxel resolutions were selected to obtain the six RD points as defined in Table 3; while Q_1 corresponds to the lowest quality (and bitrate), Q_6 corresponds to the highest quality (and bitrate).
- **HB-PCC parameters:** The proposed HB-PCC solution is controlled by several parameters, such as the FPFH descriptor radius and normal radius, the search window size, the subsampling voxel resolution, the supervoxel seed resolution and the number of best matches. Thus, exhaustive tests were performed with the objective of finding the optimal HB-PCC coding parameters values in the sense of minimizing the *rmse* between the motion compensated point cloud and the target point cloud, as it will be described in Section 4.2. The tests were performed for a small set of *Redandblack* frames, changing only one parameter in each test. The parameters values selected are reported in Section 4.2.

Table 3: Voxel resolutions for the six selected RD points.

Quality points	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Voxel resolution (millimetres)	10.8	7.2	4.5	3.4	2.7	1.7

4.1.4 Objective Quality Metrics

From the point cloud objective quality metrics presented in Section 2.5, only the cloud to cloud metrics were used for the performance assessment of the codecs, as the cloud to plane metrics are still at a rather experimental stage. In summary, two objective quality metrics were used, notably:

- **Symmetric *rms* distance or error, equation (5) -** The symmetric *rmse* is a good indicator to analyse how similar or different two point clouds are. Considering two point clouds adjacent in time, the *rmse* between them can help to understand the amount of motion occurring between those two point clouds; typically, the higher the amount of motion, the higher is the *rmse*. This is the reason

why this metric is used in the coding parameters optimization presented in Section 4.2, and in the prediction quality assessment presented in Section 4.3.

- **Geometric PSNR, equation (4) -** The geometric PSNR is used to evaluate the quality of one point cloud in comparison to another point cloud. If the two point clouds correspond to the original and decoded point clouds for a given frame, the geometric PSNR is an important metric that can quantify their resemblance, making it the best tool to evaluate the quality of the decoded point clouds in Section 4.4.

4.2. Parameters Optimization

The three main modules of the HB-PCC architecture described in detail in Chapter 3, this means Point correspondence estimation, Point cloud clustering, and Homography estimation, are controlled by several parameters that can lead to rather different performance results; some of them can have a rather major impact on the quality of the motion compensated point clouds. Thus, a set of experiments was required to identify the parameters values leading to the highest quality motion compensated point clouds. As in these experiments the computational time is an unavoidable barrier, it was decided to perform the set of experiments in a shorter point cloud, in this case ten frames of the dataset *Redandblack.* Depending on the parameter to optimize, up to ten different candidate values were tested. This optimization procedure is, thus, based on the ten frames' average *rmse*, computed between the motion compensated and target point clouds, since the critical objective is to obtain the best possible motion compensated point clouds; in practice, this means that the lowest average *rmse* determines the selection of the optimal parameter value.

This set of experiments tried to optimize one parameter at a time, meaning that it started by optimizing one parameter while the others were taking recommended, default values. After, each consecutive experiment changed the previously tested parameter from the default value to the corresponding optimized value. The order of the parameters optimization was defined so that the first parameters optimized were the ones with the greater impact on the motion compensated point clouds, e.g. the quality of the descriptor, and on the computational time, e.g. the search window size. The parameters optimization is detailed for each parameter in the following sections.

FPFH descriptor radius and normal radius

The radii of the FPFH descriptor and of the normal is probably the most important HB-PCC parameter as it is responsible for the quality, i.e. distinctiveness, of the descriptors for each point in the point cloud. If the descriptor radius is too small, there might be not enough information to distinguish the descriptors; on the contrary, if it is too large, the local properties of the points are lost, thus resulting into similar descriptors. Also, the normal radius needs to be lower than the descriptor radius, according to the PCL [52]. With these hints in mind, the two radii were tested in pairs, using 10 values for the descriptor radius and 3 values for the normal radius (totalling 30 tests for these parameters). The descriptor radius was tested from 0.018 *m* to 0.18 *m*, with ten regular steps. As for the radius of the normal, the tested values were one sixth, one third, and one half of the descriptor radius that is being tested in the pair. Table 4 summarizes the average *rmse* obtained with these tests. The optimized

parameters, a descriptor radius of 0.144 *m* and a normal radius of 0.048 *m*, were close to the default values recommended by PCL, 0.09 *m* and 0.054 *m* respectively; however, as it can be observed from Table 4, some other tested parameter values achieved similar performance scores, thus suggesting that there is not a single value, but rather a small range of values, that can produce the best motion compensated point clouds.

Table 4: Average rmse for the descriptor and normal radius parameters.

Descriptor Radius (*m*)	**Normal Radius**		
	$\frac{1}{6}\times Descriptor\ radius$ **(*m*)**	$\frac{1}{3}\times Descriptor\ radius$ **(*m*)**	$\frac{1}{2}\times Descriptor\ radius$ **(*m*)**
0.018	8.574	4.132	3.633
0.036	3.963	3.074	3.059
0.054	2.641	2.753	2.830
0.072	2.511	2.552	2.565
0.090	2.391	2.433	2.494
0.108	2.322	2.318	2.318
0.126	2.338	2.366	2.407
0.144	2.204	2.122	2.242
0.162	2.300	2.154	2.206
0.180	2.249	2.303	2.257

Search window size

The second parameter optimized was the size of the search window used in the point descriptor matching process, see Section 3.2.4, due to its significant impact on the total computational time when determining the correspondences. In the search window size optimization process, two conditions were established: First, the previous point cloud is searched in a radius that should be sufficiently large to always find points even if there is a large motion; second, the search is only performed for a given number of nearest neighbours, and the search is stopped after reaching that number, to reduce the computational time. The number of neighbours was used here as the search window size, and it was expressed as a percentage of the total number of points in the previous subsampled point cloud, ranging from 0.5% to 5% with ten equal steps. Table 5 shows the results obtained for this parameter optimization test. There is no default value here recommended by PCL since the search window concept was added when designing the HB-PCC solution. A search window size of 2.5% was the value selected as it led to the lowest average *rmse*. As it can be observed from Table 5, increasing this search window size may have the effect of adding noise, as more correspondences are compared, and thus more “bad” matches are potentially created.

Table 5: Average rmse for the search window size parameter.

Search window size (%)	Average *rmse*
0.5	2.019
1.0	1.973
1.5	2.020
2.0	2.034
2.5	1.962
3.0	2.076
3.5	2.127
4.0	2.120
4.5	2.175
5.0	2.209

Subsampling voxel resolution

The third parameter to optimize was the subsampling voxel resolution; this parameter is associated to first action performed in the Point correspondence estimation module, the subsampling of the input point cloud. As for the search window size, the subsampling voxel resolution has a big impact on the computational time; however, as the number of points in the incoming point cloud is reduced, this subsampling process needs to be carefully performed to avoid compromising the quality of the motion compensated point clouds. The tested values ranged from 0.0018 to 0.0063 with equal steps. Table 6 presents the results for this parameter optimization test. The value selected for the subsampling voxel resolution, 0.0063 *m,* achieves almost the same *rmse* performance as the other values; this parameter is very likely the less impactful on the motion compensated point clouds. However, the big advantage of selecting this value is the resulting lower number of points in the subsampled point clouds, which is approximately 10% of the original points, thus allowing a great reduction on the computational time of subsequent steps.

Table 6: Average rmse for the subsampling voxel resolution parameter.

Subsampling voxel resolution (m)	Average *rmse*	Average number of points in the subsampled point clouds
0.0063	1.932	65 347
0.0054	1.935	88 365
0.0045	1.944	125 952
0.0036	1.972	193 597
0.0027	1.943	332 980
0.0018	1.962	699 844

Supervoxel seed resolution

While the previous three parameters are associated to the first HB-PCC module, the Point correspondence estimation, the supervoxel seed resolution is associated to the second HB-PCC module, the Point cloud clustering. This parameter is responsible for dividing the previous point cloud

into clusters, thus controlling the number, and thus the size, of the clusters. A careful choice is required for this parameter as more clusters result in more overhead to be coded per point cloud frame; however, few clusters may result into a bad quality motion compensated point cloud and thus a larger rate coded per point cloud frame. The tested values ranged from 0.216 *m* to 0.324 *m* with equal steps. Table 7 shows the results for this parameter optimization test. The optimal value, 0.342 *m,* was selected not as the one corresponding to the lowest *rms* score but as the lowest number of clusters, because the improvement on the motion compensated point clouds would not very likely compensate the increased overhead that would have to be sent. The PCL default value here, 0.18 *m,* was not included in the test as it would result in a too large number of clusters.

Table 7: Average rmse for the supervoxel seed resolution parameter.

Supervoxel seed resolution (m)	Average *rmse*	Average number of clusters
0.216	1.846	50
0.234	1.804	46
0.252	1.801	37
0.270	1.806	34
0.288	1.807	30
0.306	1.805	27
0.324	1.808	23
0.342	1.809	20

Number of best matches

The last parameter to optimize is the number of best matches, which are selected in the HB-PCC Homography estimation module. This module is responsible for selecting several best matches per cluster, according to the lowest Euclidean distance of the descriptors in a correspondence. If the number of selected best matches exceeds the number of available correspondences in a cluster, all the correspondences are used to compute the homography matrix. The number of best matches has a major impact on the quality of the motion compensated point clouds as some clusters have only a few correspondences and some have many correspondences, with some correspondences being a rather "bad" match. The number of best matches values selected for this test are presented in Table 8, were the last row corresponds to all matches being used, i.e. no selection is made. The selected optimal value, 80, is far from the average number of correspondences per cluster, and surprisingly a low number when compared to the average number of points in each cluster, 3 267 (65 347 ÷ 20) for the used material. However, using less matches makes sense, as it brings a good reduction of the *rmse*.

Table 8: Average rmse for the number of best matches parameter.

Number of best matches	Average *rmse*
5	4.487
20	2.933
50	2.298
80	1.706
100	1.747
150	1.764
250	1.767
500	1.940
1000	2.001
-	1.809

The optimal values for the several parameters are marked in green in the tables above. Some final remarks regarding the parameters optimization:

- In the supervoxel seed resolution, the lowest *rmse* was not the only factor taken into account since the number of clusters will contribute with overhead to the overall point cloud rate. Because the *rmse* does not change significantly as the number of clusters increases, the lowest number of clusters was chosen instead of the lowest *rmse*, to reduce the overhead in the bitstream.
- As the number of correspondences per cluster fluctuates a lot and, if a cluster contains a low number of points, the number of correspondences can be very low, deteriorating the prediction of that cluster, and thus the motion compensated point cloud, picking a low number of clusters to mitigate this effect seems to be appropriate.
- The number of tested parameter values was wide enough to show that for almost every parameter there was an optimal value. The exceptions were the subsampling voxel resolution and the supervoxel seed resolution, because the average *rmse* value was very similar for the tested values. This might demonstrate that the subsampling performed does not significantly influence the quality of the descriptor as it does not change significantly the point cloud geometry.
- Finally, it is important to note that the number of point cloud frames used for the tests may not be enough to perform a good parameter optimization as only ten frames from a single dataset were used; however, the processing time was a big constraint that had to be considered.

4.3. Temporal Prediction Rmse Assessment

After optimizing several parameters, the HB-PCC solution is ready to create good motion compensated point clouds, and thus it is appropriate to assess the quality of these motion compensated point clouds. As the motion compensated point clouds replace the previous point clouds as the prediction in the Inter coding mode, and the next operation in both the PCL Inter and HB-PCC codecs

is the XOR encoding, the motion compensated point cloud and also the previous point cloud are both compared to the target point cloud to assess the expected reduction of the prediction error. This assessment was made for all frames of the three selected datasets in order to obtain the best insight possible. The several tests made were organized into two main classes:

1. ***rmse* assessment with original point clouds:**
    A. **Without motion compensation:** This case measured the *rmse* between the original target point cloud and the original previous point cloud and it is labelled as "Original without MC*".*
    B. **With motion compensation:** This case measured the rmse between the target point cloud and the motion compensated point cloud (this means using the previous original point cloud and motion compensation), and it is labelled "Original with MC*".*
2. ***rmse* assessment with coded point clouds:**
    A. **Low quality without motion compensation:** The *rmse* is now measured using the (original) target point cloud and the previous point cloud after coded and decoded with low quality, i.e. with a big voxel resolution, namely Q_1 shown in Table 3; this case is labelled as "Low quality without MC".
    B. **Low quality with motion compensation:** For this case, the motion compensated point cloud is created using a decoded, low quality, previous point cloud with motion compensation; this case is labelled as "Low quality with MC".
    C. **High quality without motion compensation:** This case is similar to case 2.A but now using a decoded, high quality, previous point cloud, with a small voxel resolution, namely Q_6 shown in Table 3; this case is labelled as "High quality without MC".
    D. **High quality with motion compensation:** This case is similar to case 2.C but now using a prediction with motion compensation; this case is labelled as "High quality with MC".

Appendix C shows the results obtained for the *rmse* assessments, namely Figure 45 corresponds to the *Redandblack* dataset, Figure 46 corresponds to *Queen* dataset, and Figure 47 corresponds to *Soldier* dataset, each containing the *rmse* results for the six cases defined above. Other relevant statistical data, i.e. minima, maxima and averages, can be found in Tables 9, 10, and 11, followed by the analysis and conclusions of all *rmse* results collected for the selected datasets.

Table 9: Maximum, minimum and average rmse values for the six rmse assessment cases for the Redandblack dataset.

		Original	Compressed – High quality	Compressed – Low quality
Maximum	Previous point cloud prediction	4.894	4.922	5.212
	Motion compensated point cloud prediction	3.104	3.180	5.372
Average	Previous point cloud prediction	2.446	2.537	3.687
	Motion compensated point cloud prediction	2.088	2.121	3.732
Minimum	Previous point cloud prediction	1.370	1.516	3.131
	Motion compensated point cloud prediction	1.440	1.429	3.143

Table 10: Maximum, minimum, and average rmse values for the six rmse assessment cases for the Queen dataset.

		Original	Compressed – High quality	Compressed – Low quality
Maximum	Previous point cloud prediction	1.234	1.536	3.421
	Motion compensated point cloud prediction	0.759	0.939	3.433
Average	Previous point cloud prediction	0.298	0.660	2.982
	Motion compensated point cloud prediction	0.364	0.634	3.085
Minimum	Previous point cloud prediction	0.000	0.458	2.941
	Motion compensated point cloud prediction	0.000	0.542	2.970

Table 11: Maximum, minimum, and average rmse values for the six rmse assessment cases for the Soldier dataset.

		Original	Compressed – High quality	Compressed – Low quality
Maximum	Previous point cloud prediction	2.577	2.644	3.779
	Motion compensated point cloud prediction	1.333	1.436	3.316
Average	Previous point cloud prediction	0.540	0.830	3.034
	Motion compensated point cloud prediction	0.545	0.721	3.022
Minimum	Previous point cloud prediction	0.139	0.521	2.946
	Motion compensated point cloud prediction	0.289	0.589	2.928

By considering the *rmse* results shown in Appendix C and the additional information in the tables above, for the three datasets, the following main conclusions may be derived:

- The three datasets have different motion behaviours, as shown by the *rmse* assessment, with the *Redandblack* scoring the most motion, followed by the *Soldier* and, finally, the *Queen* datasets. The amount of motion can be associated to the *rmse* values, with a higher *rmse* corresponding to more motion between two subsequent point cloud frames. By looking to the dark blue curves in the Appendix C charts, "Original without MC" data in the *rmse* assessments, one can observe that *Redandblack* contains irregular, but always high motion, while the remaining two datasets contain irregular motion, with some spikes corresponding to parts of the point clouds with more motion.
- Comparing the "Original without MC" and "Original with MC" results, the dark blue and yellow curves in the Appendix C charts, three situations occur:
    - *The motion compensated point cloud has a lower rmse than the previous point cloud,* implying that a better prediction of the target point cloud was obtained. This happens for the parts of the datasets with medium, and high motion, e.g. the *rmse* spikes in frames 19-50 of *Redandblack*, the *rmse* spikes in frames 185-209 of *Queen*, and the *rmse* spikes in frames 169-187 of *Soldier*.
    - *The motion compensated point cloud has similar rmse comparing to the previous point cloud,* which means that both point clouds correspond to equivalent predictions of the target point cloud. While this situation is not desired, it may have a small impact as the motion compensated point cloud is similar to the previous point cloud as prediction; however, the rate overhead corresponding to the transmission of the computed homographies will probably make the prediction with motion compensation worse as the prediction just using the previous point cloud, which does not have this overhead. Examples of this situation are common for *Redandblack* and happen often between frames 124 and 240.
    - *The motion compensated point cloud has worse rmse than the previous point cloud,* which means that the motion compensated point cloud is a poorer prediction than the previous point cloud; this is highly undesirable. This type of situation must be avoided, and a workaround will be presented in the next section. This situation is very frequent for low *rmse* areas, e.g. for the *Queen* frames 0 to 181, and the *Soldier* frames 0 to 96.
- Analysing the "Original without MC" and "High quality without MC" results, coloured in dark blue and orange in the Appendix C charts, respectively, two observations can be made:
    - The two curves, for every dataset, are very similar, due to the evident resemblance between the original previous point cloud and a high quality decoded previous point cloud. The *rmse* temporal evolution, i.e. ups and downs along time, is very similar between these two curves.
    - A curious effect is, however, observable in the *rmse* curves of *Queen* and *Soldier*. For the lower *rmse* zones, the distance between the two curves, "Original without MC" and "High quality without MC", is larger than for the higher *rmse* zones. For example, for the *Queen* frames 181 to 209, the *rmse* difference of the two curves, "Original without MC" and "High

quality without MC", reduces from approximately 0.5 to approximately 0.3. This happens because the lower *rmse* zones are more sensible to degradation introduced by the quantization than higher *rmse* zones like the *rmse* spikes.

- The "Low quality without MC" and "Low quality with MC" results, coloured in gray and green, respectively, in the Appendix C charts, show the largest differences in terms of *rmse*. Two main observations that can be derived from the comparison of the "Low quality without MC" and "Low quality with MC" results:
    - Due to the large voxel size, the *rmse* increase is very significant comparing to the "Original without MC" and "Original with MC" cases. The visual impact on the decoded point cloud is evident: it contains about 3% of the original points, and some geometry details are lost, e.g. sharp surfaces are smoothened.
    - The same effect that happened between the "Original without MC" and "High quality without MC" still persists and is more evident. The big *rmse* spikes that appeared in all datasets, and principally in *Queen* and *Soldier*, corresponding to higher motion zones, are much less affected by the degradation introduced by the quantization, because those zones had already a big error with respect to the original data, and thus, the degradation associated to the "low quality" coding has a lower impact.
- Analysing tables 8 to 10 allows to further derive the following conclusions:
    - The maximum *rmse* for the "Original" and "High Quality" cases, and for all datasets, always decreases when the motion compensated point cloud are used as prediction instead of the previous point cloud, meaning that higher *rmse* zones have better predictions.
    - The maximum *rmse* for the "Low Quality" case has a lower value when the motion compensated point cloud is used as prediction only for the *Soldier* dataset. The prediction using a low quality decoded previous frame is more difficult, e.g. the homographies are less representative of the real motion, which justifies the rising of the maximum *rmse* for the *Redandblack* and *Queen* datasets.
    - The average *rmse* has a great reduction when using motion compensated point clouds for the *Redandblack* dataset, except for the "Low Quality" case. However, for the remaining two datasets, there is a small *rmse* reduction for the "High Quality" case, and even a small increase for the "Original" case. The reason for this to happen is the lower amount of motion for the *Queen* and *Soldier* datasets, which is responsible for the average *rmse* increase when using a motion compensated point cloud. As it has been seen previously in this section, the motion compensated point cloud typically has a higher *rmse* value than the previous point cloud in a low motion scenario.
    - The minimum *rmse* when using the motion compensated point cloud usually increases, except for *Redandblack* for the "High Quality" case. Although there is no apparent reason to justify this effect, a possible explanation is that since the homography is applied to rather large clusters, the homography matrix may rather poorly represent the transformation of many points for some clusters, notably if they contain little motion (it is a minimum). Another possible explanation is the existence of "bad matches", which introduce noise in the

homographies, and even the smallest noise can have a negative impact, e.g. points that do not move between two point clouds may have a small movement unduly applied by the homography.

4.4. RD Performance Assessment

Finally, after assessing the *rmse* with and without motion compensation applied to the previous point cloud, for the three datasets, this section will assess the HB-PCC RD performance.

The previous section has shown that an undesirable situation in terms of *rmse* may happen for several frames, this means having a motion compensated point cloud with worse *rmse* than the simple use of the previous point cloud. To avoid the negative impact of this situation, a rather simple prediction mode switch has been added to the proposed HB-PCC solution. This prediction mode switch is added after the point cloud motion compensation and simply decides whether to perform prediction with or without motion compensation, depending on which of these predictions has the lowest *rmse*. This would guarantee that the selected prediction mode is always the one with the smallest *rmse*. Naturally, this requires a signalling bit to be sent to the decoder, which informs if the prediction is made using the previous decoded point cloud with or without motion compensation. Thus, the RD performance assessment made in this section corresponds to the HB-PCC solution with prediction mode switch.

It is important to stress that this switch is applied at frame level while a much better solution would be to have the same switch at the cluster level to more selectively and locally exploit the benefits of motion compensation. Unfortunately, it is not possible to implement a cluster-based switch in due time.

As mentioned in the coding conditions presented in Section 4.1.3, two different distortion/quality metrics are adopted for the RD performance evaluation, i.e. the PNSR and *rmse.* This assessment will be made for GOP sizes of 2 and 8, this means 1 Intra frame out of 2 and out of 8, respectively. Figures 39 and 40 show the RD performance assessment for the *Redandblack* dataset for GOPs size 2 and 8, respectively, Figures 41 and 42 show the same RD performance assessments for the *Queen* dataset*,* and, finally, Figures 43 and 44 show the same RD performance assessments for the *Soldier* dataset. Tables 12 to 14 also show relevant statistical data about the HB-PCC coding of the three datasets, namely: 1) the average percentage of initial points that the decoded point clouds contain, for each quality point; and 2) the percentage of frames that were coded using as prediction the motion compensated point cloud, using as prediction the previous point cloud or even using Intra coding, for the two GOP sizes. By analysing all the results obtained, the following conclusions can be drawn:

- **Content dependence** - As expected, for the three tested codecs, HB-PCC, PCL Inter and PCL Intra, the RD performance is extremely dependent of the content of the datasets, notably in terms of motion. While PCL Intra is the worst performing codec for *Queen* and *Soldier*, as expected, for *Redandblack* the situation is rather different as the best codec is PCL Intra. This behaviour may be explained by the high motion characterizing this dataset and by the not good enough motion compensation. When in the presence of high motion between two successive frames, many changes at octree level have to be coded and transmitted, in Inter coding mode, if the motion compensation does not perform well, eventually overcoming PCL Intra in terms of rate.

- **HB-PCC versus PCL Inter** - The comparison between PCL Inter and the proposed HB-PCC solution, leads to the unexpectable conclusion that the proposed solution always "looses", i.e. always uses more rate to reach the same quality, even if the bitrate increase may be small. This is unexpected, as the previous section has shown a *rmse* drop when using the motion compensated point cloud as prediction for some frames, and the prediction mode switch added to HB-PCC guarantees that the motion compensated prediction is used only when its *rmse* is lower. A possible justification for this behaviour may be related to the quality of the motion compensation, with the homography estimation deteriorating some zones of the point cloud with low/no motion, even if it compensates well some zones with a lot of motion. So, even if the average *rmse* is reduced, the fact that some zones with low/no motion were deteriorated may annul, and eventually reverse, the rate gain resulting from the good motion compensation for the zones with a lot of motion. Keep in mind that the decision of using motion compensation may only be done for the full frame (all clusters) or no cluster at all. Such justification demanded further investigation, but no time was available. Another important topic deserving further investigation is the XOR technique since, in this Thesis, the XOR technique available in PCL [36] was straightforwardly applied without a deeper study.
- **GOP size impact** - The increase of the GOP size from 2 to 8 frames has the impact of increasing the differences between the RD curves for the three coding solutions. Naturally, PCL Intra keeps always the same RD curve, while PCL Inter and HB-PCC will have approximately seven times more frames coded in Inter mode. For *Redandblack*, both PCL Inter and HB-PCC increase the rate compared to PCL Intra due to the very high motion of this dataset, while for *Queen* and *Soldier* those coding solutions reduce the rate regarding PCL Intra, meaning that the exploitation of temporal redundancy brings additional compression, at least when the motion is not too high.
- **Geometry characteristics impact** - For the three codecs, *Soldier* is the dataset with worst RD performance, while *Queen* has the best RD performance. As *Soldier* is the dataset with the largest average number of points in the decoded point cloud, as shown in tables 2 and 11, it becomes more expensive to reach a target quality. On the other hand, *Queen* does not have the smallest average number of points in the decoded point cloud, but still achieves the lowest bitrate for a target quality, probably because, as it is computationally generated, its geometry is simpler, less noisy, and more fitted to an octree, thus reaching a higher compression, than e.g. *Redandblack.*
- **Motion compensation decision level impact** - Tables 13 and 14 show how much the motion compensated prediction and the simple previous cloud prediction are used by HB-PCC, for the two selected GOP sizes, for all the datasets and for Q_6 and Q_1, respectively. As shown in the previous section by the *rmse*, *Redandblack* has a large number of frames for which the motion compensated prediction error is lower than previous point cloud prediction error, and thus HB-PCC uses more frequently the motion compensation prediction mode for this dataset. Keep in mind that the maximum percentage of temporal prediction usage is about 50% for GOP 2 and 89% for GOP 8 as the remaining percentage is for Intra coding. Obviously between GOP size 2 and 8, the number of point cloud frames coded in Intra mode decreases, and the number of point cloud frames coded in Inter mode increases. For the more stable datasets, the percentage of frames using motion compensated prediction is rather low for the simple reason that this codec can only decide to use motion

compensation for the frame or no motion compensation at all. This is a killing limitation as motion compensation should be adaptively decided at the cluster level, depending on the motion characteristics of each cluster. In the HB-PCC solution, it is possible that motion compensation is being peformed for clusters for which it is not worthwhile in terms of prediction error with the additional onus of having to pay the rate for the homography parameters transmission (which, by the way, is not even optimized).

- **Target point cloud voxel resolution impact –** The analysis of Tables 13 and 14 also show that, in general, the decrease of the target point cloud resolution, which occurs when moving from Q_1 to Q_6, has the impact of increasing the percentage of frames coded using motion compensated prediction while decreasing the percentage of frames coded using the previous point cloud as prediction. When increasing the size of the voxels, more points will fall in each voxel, meaning that the total number of points of the point cloud to code, as well as the reference point cloud, will reduce. As a consequence, each point cloud cluster will enclose a smaller amount of point correspondences, possibly poorly related with each, deteriorating the prediction of that cluster, and thus the motion compensated point cloud.

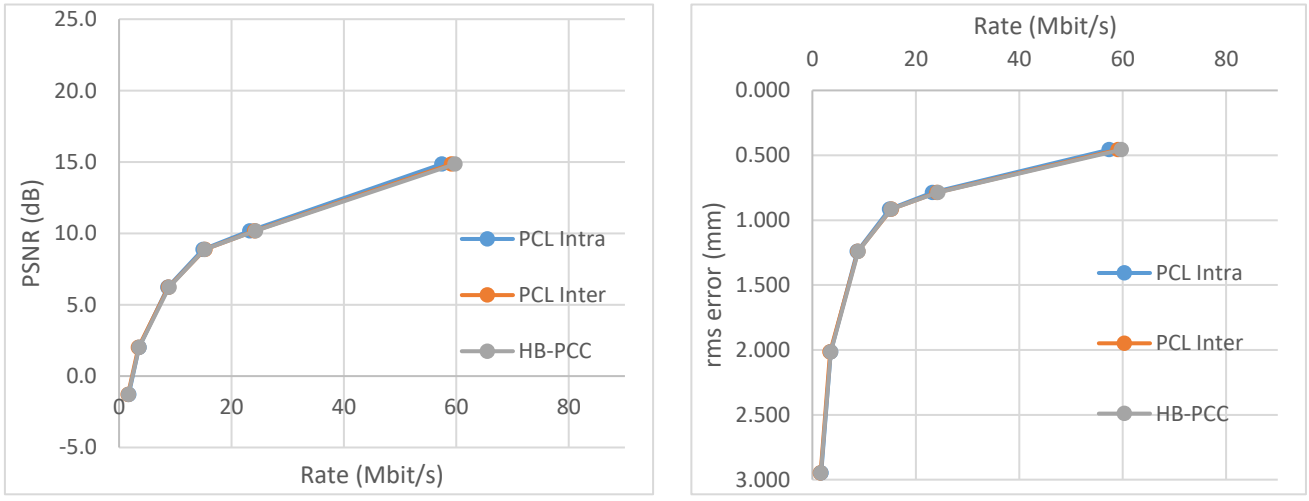

Figure 39: Left: PSNR versus rate for the Redandblack dataset, GOP size 2. Right: rmse versus rate for the Redandblack dataset, GOP size 2.

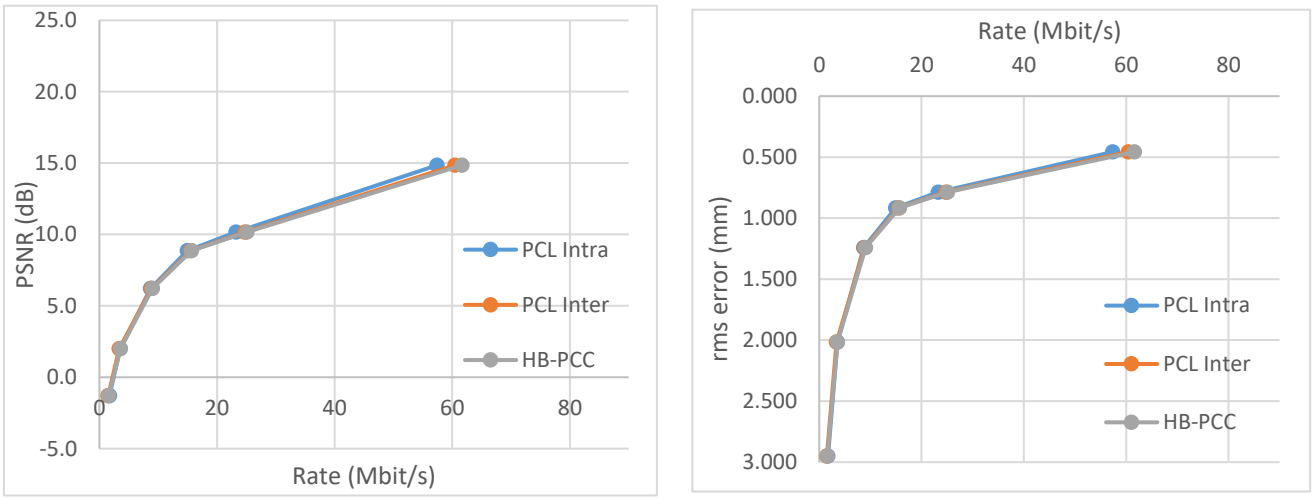

Figure 40: Left: PSNR versus rate for the Redandblack dataset, GOP size 8. Right: rmse versus rate for the Redandblack dataset, GOP size 8.

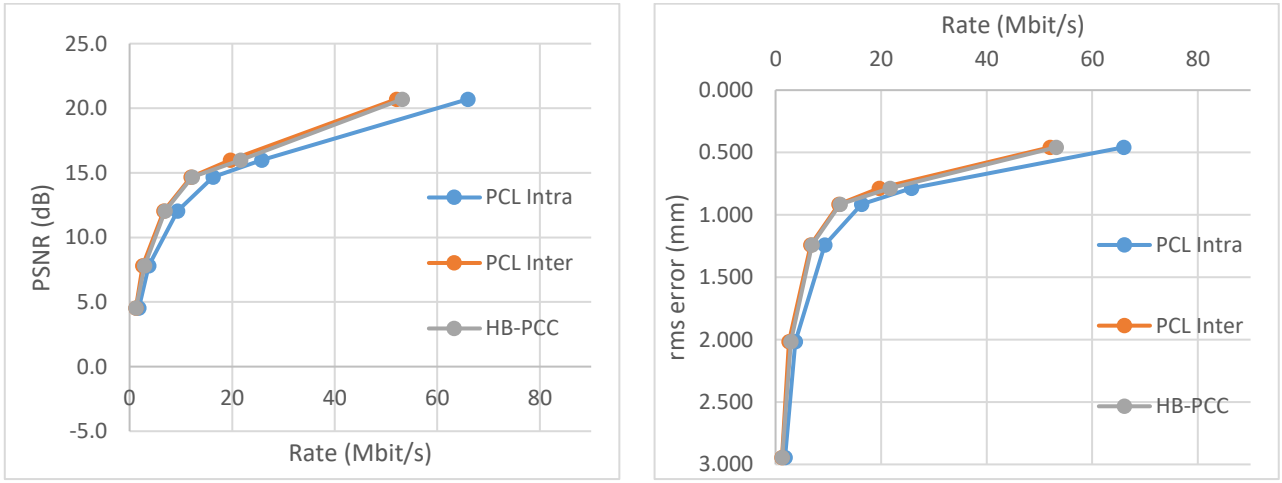

Figure 41: Left: PSNR versus rate for the Queen dataset, GOP size 2. Right: rmse versus rate for the Queen dataset, GOP size 2.

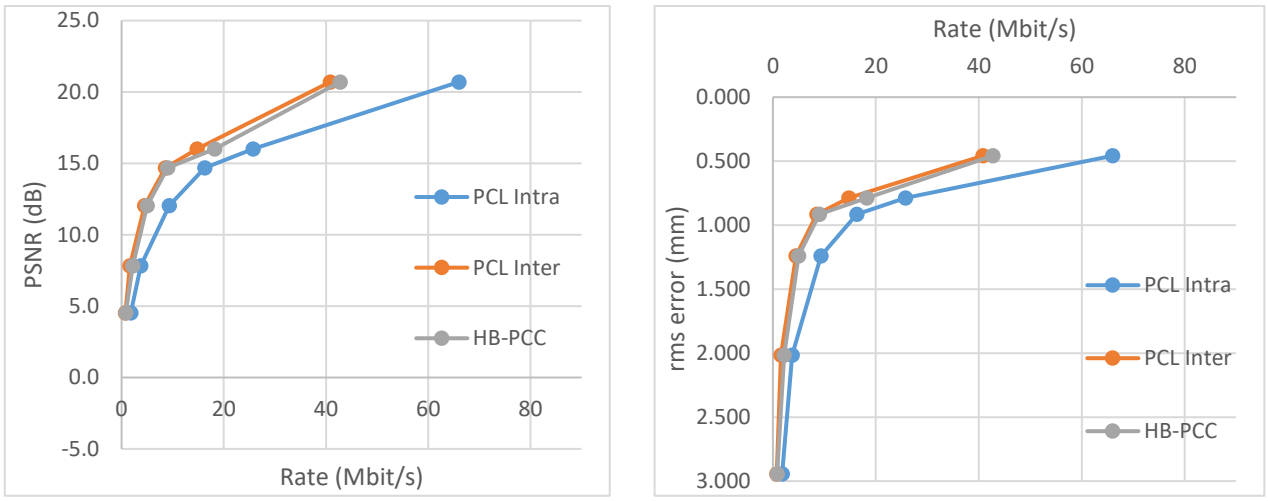

Figure 42: Left: PSNR versus rate for the Queen dataset, GOP size 8. Right: rmse versus rate for the Queen dataset, GOP size 8.

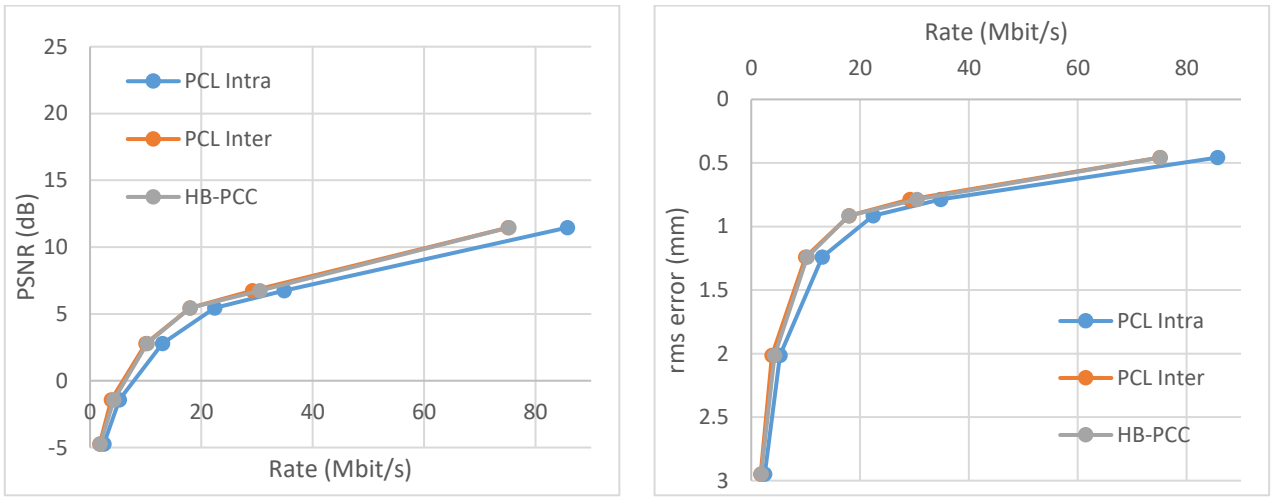

Figure 43: Left: PSNR versus rate for the Soldier dataset, GOP size 2. Right: rmse versus rate for the Soldier dataset, GOP size 2.

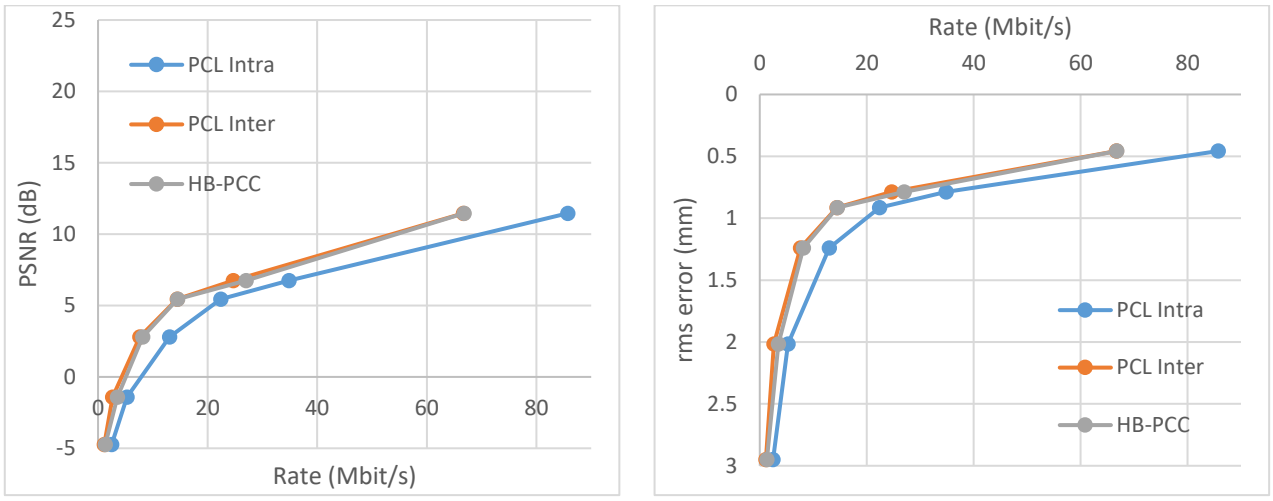

Figure 44: Left: PSNR versus rate for the Soldier dataset, GOP size 8. Right: rmse versus rate for the Soldier dataset, GOP size 8.

Table 12: Average percentage of initial points remaining in the decoded point cloud for the three datasets.

	Average percentage of initial points remaining in the decoded point cloud (%)		
Quality Point	*Redandblack*	*Queen*	*Soldier*
Q_1	3.24	2.65	3.28
Q_2	7.24	6.06	7.29
Q_3	18.11	15.70	18.16
Q_4	30.63	27.36	30.68
Q_5	47.75	44.02	47.81
Q_6	100.00	100.00	100.00

Table 13: Percentage of frames coded using either the motion compensated point cloud, the previous point cloud, and the Intra coding mode, for the three datasets, using the HB-PCC for Q_6.

		Frames coded using as prediction the motion compensated point cloud (%)	**Frames coded using as prediction the previous point cloud (%)**	**Frames coded in Intra coding mode (%)**
Redandblack	**GOP 2**	38.7	11.0	50.3
	GOP 8	69.7	19.0	11.3
Queen	**GOP 2**	14.8	34.8	50.4
	GOP 8	25.6	63.2	11.2
Soldier	**GOP 2**	17.0	32.7	50.3
	GOP 8	28.7	60.0	11.3

Table 14: Percentage of frames coded using either the motion compensated point cloud, the previous point cloud, and the Intra coding mode, for the three datasets, using the HB-PCC for Q_1.

		Frames coded using as prediction the motion compensated point cloud (%)	**Frames coded using as prediction the previous point cloud (%)**	**Frames coded in Intra coding mode (%)**
Redandblack	**GOP 2**	18.7	31.0	50.3
	GOP 8	53.3	35.4	11.3
Queen	**GOP 2**	6.0	43.6	50.4
	GOP 8	10.4	78.4	11.2
Soldier	**GOP 2**	15.3	34.4	50.3
	GOP 8	29.4	59.3	11.3

Chapter 5

5. Conclusions and Future Work

This final Chapter starts by presenting the principal achievements and then the main conclusions, as well as some possible research directions for future work.

5.1. Key Achievements

The first contribution of this Thesis, was the design, implementation and assessment of a dynamic point cloud compression. This contribution included using point feature extraction, for each point of a target and a previous point cloud, finding correspondences between the point features of a target and a previous point cloud, motion estimation with a geometry transformation (homography), clustering of a previous point cloud, and finally, warping of each cluster in a previous point cloud to obtain a motion compensated point cloud of better quality.

Another contribution was the integration of all above techniques in a novel solution, the Homography Based Point Cloud Codec, or HB-PCC, which then uses the PCL double-buffering octree-based approach to obtain and code the residuals between a target point cloud and a motion compensated point cloud.

Finally, the assessment of the *rmse* of the HB-PCC motion compensation point clouds versus the previous point clouds, as predictions of a target point cloud, was performed. To complement the *rmse* assessment, a RD performance assessment was also made to compare the HB-PCC and PCL Intra and PCL Inter codecs.

5.2. Conclusions

The objective of this Thesis was to better exploit the temporal correlation in dynamic point clouds sequences with suitable motion estimation and compensation techniques and to integrate them in a well-known octree based point cloud codec. When the quality of the predicted frame is evaluated through the suitable *rmse* metric, the results were better in some cases, with an evident reduction of the average *rmse* (better quality) for the sequence with high motion – *Redandblack*. However, the average *rmse* of

the remaining sequences, *Queen* and *Soldier*, that have lower amount of motion didn't bring the expected results, since the *rmse* has slightly increased for the *rmse* assessments with original point clouds situation, and did not significantly changed for the "High quality" situation. Although, for these sequences, the parts that contain a higher degree of motion have a lower *rmse*, which allows to conclude that the framework designed to obtain a predicted (motion compensated) frame performs better for sequences of point clouds, or temporal parts of a sequence, with high motion.

The other objective of this Thesis was to achieve higher coding efficiency for dynamic point clouds compared to the currently available solutions, in this case the solution available in the Point Cloud Library which uses the previous past coded frame in a dual-buffer octree-based approach. However, an unexpected result was observed when comparing the PCL Intra and PCL Inter codecs: for sequences of point clouds containing high motion the PCL Intra outperforms PCL Inter, which may difficult the exploration of the motion correlation between consecutive frames. For the proposed solution, HB-PCC, despite the fact that, for some cases, the quality of the motion compensated point cloud frame is better (in terms of *rmse*) when compared to the previous point cloud frame (which represents state-of-the-art), a better RD performance was not obtained, unfortunately. Even when two modes were introduced to select between the previous and the motion compensated point cloud frames (with a *rmse* criteria), the RD performance is still lower between the proposed motion compensated solution and the previous frame. Therefore, more time is needed to further investigate why the proposed solution underperforms and exploit possible solutions to improve the HB-PCC compression efficiency. This lack of performance may result from the homography estimation technique which may not be accurate for static areas of the point cloud, i.e. areas with little or no motion. Thus, for frames (or areas) containing little, or no motion, the previous point cloud is a better prediction compared to the motion compensated point cloud. However, when high motion occurs, a better prediction quality is achieved. For these cases, the RD performance of the proposed solution didn't increase mainly because the dual-buffering XOR based approach requires to XOR the occupancy bytes of corresponding nodes and slight displacements between the motion compensated frame and the target frame, create the need to transmit occupancy bytes for new nodes but also to delete nodes for which there is no prediction. In addition, the HB-PCC solution includes additional headers (bitrate overhead) which contain uncompressed homography parameters that for lower bitrates have a big impact. However, the work performed is an early attempt of this complex coding solution and the results obtained may encourage others to improve the proposed motion compensated framework as well as make modifications in the octree-based PCL codec to achieve better RD results.

5.3. Future Work

Following the conclusions, this final section addresses possible improvements that can increase the proposed point cloud codec RD performance assessment. These are described next:

- **Improving correspondences accuracy:** With the proposed framework there is still some erroneous correspondences between the points of the previous and target point clouds, i.e. correspondences which do not accurately described the motion of the object. Several

techniques can be used to obtain better correspondences and thus to obtain a better motion compensated point cloud, which may improve the final RD performance:

  - **New point descriptors:** Improving the FPFH descriptor or selecting a descriptor that is more representative of a surface around a point may bring improvements in the correspondence matching process. One possibility is to include other attributes such as colours like the PFH-RGB descriptor [53]. Other possibility is to develop a descriptor more suitable to track motion for dense point clouds mitigating the problem of ambiguous descriptors in nearby points.
  - **Removing redundant points:** In the original PFH paper [41], the authors suggest a so-called *persistence analysis* with the objective of removing redundant points, and leaving only the most unique (or distinct) points of a point cloud. This technique should be combined with another technique (e.g. interpolation of the correspondences) since some areas of the point cloud, e.g. flat areas, would not have any point available to perform a correspondence between the two point clouds.

- **Improving spatial based clustering:** The clusters obtained in the supervoxels PCL clustering technique are not the best approach to model motion, as they represent clusters that have points close to each other, with similar normals and with similar colours. A good improvement would be to obtain clusters of points close to each other but that contain similar motion vectors (correspondences). Since this solution requires good correspondences, it is first suggested to improve the quality of the correspondences, as described in the previous two points.
- **Coding the homography parameters:** In the proposed HB-PCC solution, the homography parameters were not compressed and therefore a suitable compression scheme is necessary. For lower bitrates, i.e. for the first two or three RD points, the overhead of these parameters is high and the coding of the homography parameters is especially important. Different techniques could be applied from quantization to differential coding of the parameters of one cluster to the other.
- **Selecting clusters from the previous point cloud:** One of the identified problems was the low number of points, and thus correspondences, belonging to some clusters in the previous point cloud. The clusters obtained in the HB-PCC vary a lot in size and number of correspondences, which indubitably lead to some bad warpings. One greedy and difficult way to solve this problem, is to analyse each cluster individually, and assess the quality of the cluster belonging to the previous point cloud and the motion compensated cluster. Then the final prediction is formed using clusters of points belonging to the previous point cloud or the motion compensated based on some criteria of quality. The difficulty of this solution is that a cluster in the previous point cloud does not have a corresponding cluster in the target point cloud, to assess its quality.

In conclusion, some objectives of this Thesis were achieved, namely obtaining a motion compensated point cloud which is a better prediction than the previous point cloud. However, it was found that better RD performance was not obtained compared with the available point cloud coding solutions, thus leaving room for future improvements, as suggested in the Future Work.

Appendix A

A. An Overview on the Point Cloud Descriptors Available in the Literature

In Table 15 are listed the principal characteristics of the main descriptors used for point cloud processing. Although not mentioned, all these descriptors can only describe the geometry of the point cloud, meaning that their only input are the positions of each point in the point cloud. The table is arranged so that the type of descriptors that only define a single point of the point cloud will appear first, and descriptors that define a group of points in the point cloud appear second. In Table 15, some descriptors are modifications to a previous descriptor, wielding more advantages and eventually some disadvantages, which is the case of the FPFH, VFH and CVFH descriptors, all variations of the PFH descriptor, and the USC descriptor, an upgrade of the 3DSC descriptor.

In terms of performance the author of [45] tested all the listed descriptors, in the same circumstances, where several objects where acquired as point clouds in three different perspectives, where two perspectives are for training, and a last perspective is for testing. Using the Harris 3D keypoint extractor [53], the PFH descriptor outperformed the others, and the ESF descriptor ranked second. Using a point cloud sub-sampling with 1 cm the PFH descriptor ranked second, after the SHOT descriptor, and with a sub-sampling with 2 cm, the PFH descriptor managed to rank first again, with the SHOT descriptor coming second.

Table 15: Characteristics of the point cloud descriptors available in PCL.

Name of the descriptor	Type	Approach	Output	Advantages	Disadvantages	Matching criteria	Available in PCL
Point feature histograms (PFH) [41]	Single point, local descriptor	For each point a neighbourhood is determined, and for all pairs of points inside that neighbourhood three unit vectors are computed, forming the called Darboux *uvn* frame. This frame is then used to obtain 3 angles and a distance, originating the descriptor	Each point has a histogram with 125 bins	Robust to noise. Associates points to geometry. Multi-dimensional	Run-time goes up exponentially. High memory usage Invariant to scale and pose	Kullback-Leibler distance	Yes
Fast point feature histograms (FPFH) [42]	Single point, local descriptor	For each point, the Darboux *uvn* frame is computed only for pairs containing that point and its neighbours, obtaining a simplified (fast) version of the PFH descriptor	Each point has a histogram with 33 bins	Associates points to geometry. Multi-dimensional. Fast	Lack some descriptive power. Invariant to scale and pose	Huber penalty measure	Yes
Principal curvatures estimation (PCE) [54]	Single point, local descriptor	For each feature point, the directions and magnitudes of the principal surface curvature are computed, by obtaining the eigenvectors and eigenvalues respectively	Each point has a descriptor with 5 values	Fast	Invariant to viewpoint. Susceptible to noise	Euclidean distance	Yes
3D shape context (3DSC) [55]	Multi-point, local descriptor	The spherical volume around a point is subdivided. The divisions are based on a logarithm radius, and equally spaced azimuth and elevations. Each bin, or division, has a weighted count, obtained using all the points inside. The north pole of the spherical grid is aligned with the feature point normal, and for each division in azimuth, it is computed a descriptor	For a subdivided spherical volume, several descriptors are computed with the weighted counts of the points that fall in each bin	Robust to noise. Robust to distortions in shape	High memory usage. Does not define a unique local reference frame, or RF	$\ell 2$ distance	Yes
Unique shape context (USC) [56]	Multi-point, local descriptor	For a point, a weighted covariance matrix is computed, according to its neighbours. This matrix is decomposed in eigenvectors, obtaining a local RF. After obtaining the RF, a spherical volume is obtained around the point and the descriptor is computed like the 3DSC	For a subdivided spherical volume, a single descriptor is computed with weighted counts of the points that fall in each bin, plus 3 values to define a local RF	Define a local RF. High accuracy		Euclidean distance	Yes

Rotation invariant feature transform (RIFT) [57]	Multi-point, local descriptor	A circular normalised patch is divided in concentric rings of same width, and a gradient orientation histogram is computed for each ring	The descriptor is defined by 8 bin histograms, one for each of the four rings	Variant to viewpoint. Robust to non-rigid deformations Robust to variable texture patterns	High computational complexity. High memory usage	Earth mover's distance [58]	Yes
Signature of histograms of orientations (SHOT) [59]	Multi-point, local descriptor	This descriptor is a combination of signature and histogram techniques. First, an eigenvalue decomposition is made to obtain a local RF. Then, the neighbourhood of the point is separated in a spherical grid, where each bin has a weighted histogram of normals	The RF is defined by 9 values, and the histogram has 32 bins	Robust to noise. High descriptive power. Represents geometry	Susceptible to point density variations	Euclidean distance	Yes
Point pair feature (PPF) [60]	Multi-point, global descriptor	For each pair of points, the PPF holds information about: the difference of their positions; the angle between their normals; the angles between their normals and a normal obtained from the two points difference. Then, the global descriptor is obtained by grouping together similar PPF	Each of the several PPF local descriptors is defined by 4 values	Robust to noise. Work on sparse point clouds. Fast		Variation of the Generalized Hough Transform	No
Viewpoint feature histogram (VFH) [61]	Multi-point, global descriptor	This descriptor adds an extra vector to the FPFH descriptor, by adding a histogram of the angles between the viewpoint direction and the normals of each point	3 histograms with 45 bin, plus a histogram with 128 bin	Fast. Low computational complexity. Variant to viewpoint. Robust to noise	Uses a priori segmentation. Invariant to scale. Invariant to rotations around the camera's view. Sensitive to occlusions	Euclidean distance	Yes
Clustered viewpoint feature histogram (CVFH) [62]	Multi-point, global descriptor	This descriptor determines a set of stable regions, and for each point of that set, a VFH feature is computed. The resulting descriptor has the histograms of the 4 angles found in the VFH feature, and an extra Shape Distribution Component (SDC) in order to differentiate surfaces	3 histograms with 45 bin and a histogram with 128 bin (same as VFH), plus an extra histogram with 45 bin	Variant to scaling. Robust to occlusions	Applies a smoothing algorithm a priori. Applies post-processing. Invariant to rotations around the camera's view	Own distance metric	Yes

		with very similar normal distributions, but different point distributions					
Ensemble of shape functions (ESF) [63]	Multi-point, global descriptor	This descriptor concatenates 10 histograms of 64 bins. The histograms describe the distance between two random points, the angle of two lines created with three random points, and the area of a triangle formed by three random points. Each of the values is classified in on, off, mixed, depending if that point resides inside, outside, or partially inside a surface	10 Histograms with 64 bins	Fast. High descriptive power		ℓ1 distance	Yes

Appendix B

B. An Overview on the Clustering Techniques Available in the PCL

Table 16 contains a list of clustering techniques that are available in PCL for usage, containing a short description, as well as the most important characteristics. To be mentioned that some of the available techniques existed prior to PCL and were applied in other context, and the PCL was only responsible for adapting them, in terms of working for 3D point clouds. Another important aspect, is that each technique has its own strengths and weaknesses, and some were designed for specific contexts.

The PCL is constantly working on developing new features or improving old ones, such as the point cloud clustering techniques, meaning that some techniques here mentioned are very recent. Also, some of the techniques are customizable, i.e. the user can input conditions to the clustering performed, which is a very useful feature.

Table 16: Main clustering techniques for point cloud clustering.

Clustering technique	Clustering methods used	Requirements	Approach	Advantages	Disadvantages
Conditional Euclidean	Attributes		This algorithm goes through all points and creates regions around these points by adding every neighbour point that is below a defined threshold, based on the Euclidean distance of the two points. Certain conditions like the points having an intensity, curvature, or normal below a defined threshold, can be optionally added	Can be used on unorganized point clouds	
Region growing segmentation	Seeded-region and attributes	Points normals and point curvatures	The points that lay inside a flat plane, and therefore having the lowest curvatures, are considered seeds. Each seed corresponds to the initial point of a cluster, from where the merging algorithm will start to add the points that are not seeds. For each seed, it is computed the angle between its normal and the normal of each neighbour: if below a threshold, the neighbour is added to the region containing the seed; if above the threshold, the neighbour becomes a seed. The process ends after creating regions for every seed		Slow
Color-based region growing segmentation	Seeded-region and attributes	Point curvatures and point colours	Works like the region growing algorithm, bust instead of using the normals, it uses the difference of the colour of the two points. There are also two new criteria used to merge adjacent regions, in order to avoid over and under segmentation		Slow
Voxel Cloud Connectivity Segmentation (VCCS) [48]	Seeded-region, graph, and attributes	FPFH descriptor	The first step is to compute an adjacency graph. Then it selectsseeds for each cluster, or supervoxel, and filter the seeds that are isolated points. After that, it computes a distance based on colours, positions and FPFH descriptors, between the center of each supervoxel, and the center of adjacent unassigned voxels, and if it is the smallest distance seen by that voxel, the voxel is added to the supervoxel. This is done by “levels” of adjacency, where all supervoxels add the adjacent unassigned voxels of the current “level”, before going to the next “level”	Fast	Susceptible to point density variations. Produces over segmentation
Locally Convex Connected Patches (LCCP) [64]	Seeded-region, graph, and attributes	VCCS	Starts by using the VCCS technique and obtaining supervoxels, which are over segmented clusters of the points cloud. Then, based on two criterions, each adjacency between two successive supervoxels is either separated or not. Each cluster will then be constituted by adjacent supervoxels that were chosen to not be separated		Susceptible to noise

Constrained Planar Cuts (CPC) [65]	Seeded-region, graph and attributes	VCCS	Starts by using the VCCS technique and obtaining supervoxels. Then it uses a RANSAC to decide which edges between the supervoxels can actually be separated, and similar to the LCCP technique, the clusters will correspond to groups of supervoxels that were not separated		Susceptible to noise
Random Walker [66]	Graph		First it establishes an adjacency graph. From that graph, several sets of points are obtained, and a label is computed. Based upon mathematical principles, including a system of equations, for each node of the graph is given the corresponding best label. Finally, clusters are obtaining by comparing the labels of neighbour nodes and merging similar labels		
Min-cut [67]	Graph	3D location and a radius	First it is created an adjacency graph with two extra vertices, source and sink, connected to every node, or point of the point cloud. Each node, or point, will have three different weights, one for the connections with the adjacent nodes, and the other two for the connections with the source and sink. By using this different weights, each node is associated to a foreground or background, which are the two clusters obtained in this technique		Slow Requires manual inputs (the position of the center of the object)
Difference of normals [68]	Attributes		First, for each point in the point cloud, it is computed two normals, where one was obtained using a small neighbourhood and the other was obtained using a big neighbourhood. Then, for each point, it is computed the difference between these two computed normals. The points that have similarities in this computed difference, will then be merged into clusters	Can be used on unorganized point clouds	

Appendix C

C. Temporal Prediction Rmse Assessment Charts

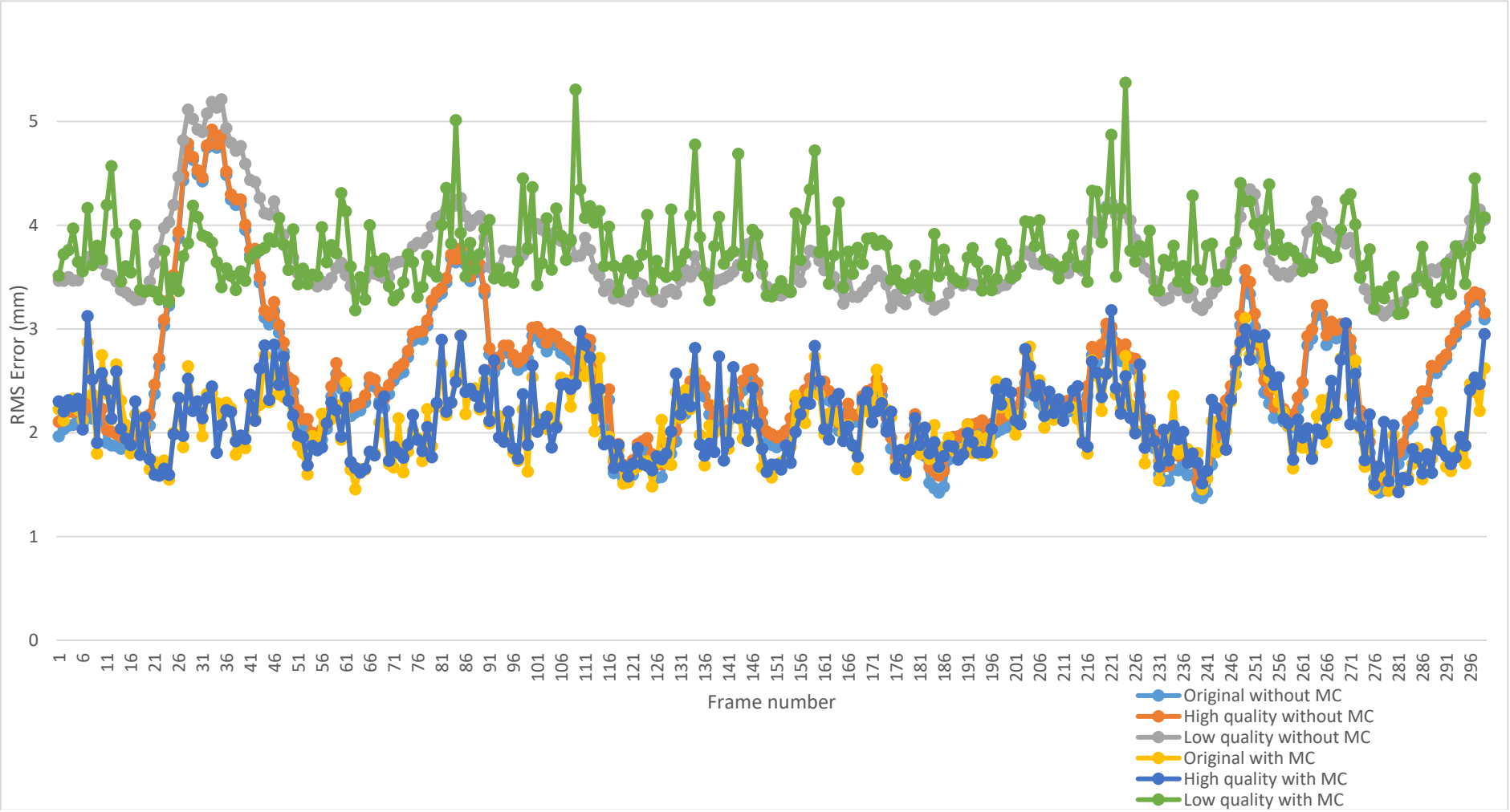

Figure 45: Redandblack rms error temporal evolution.

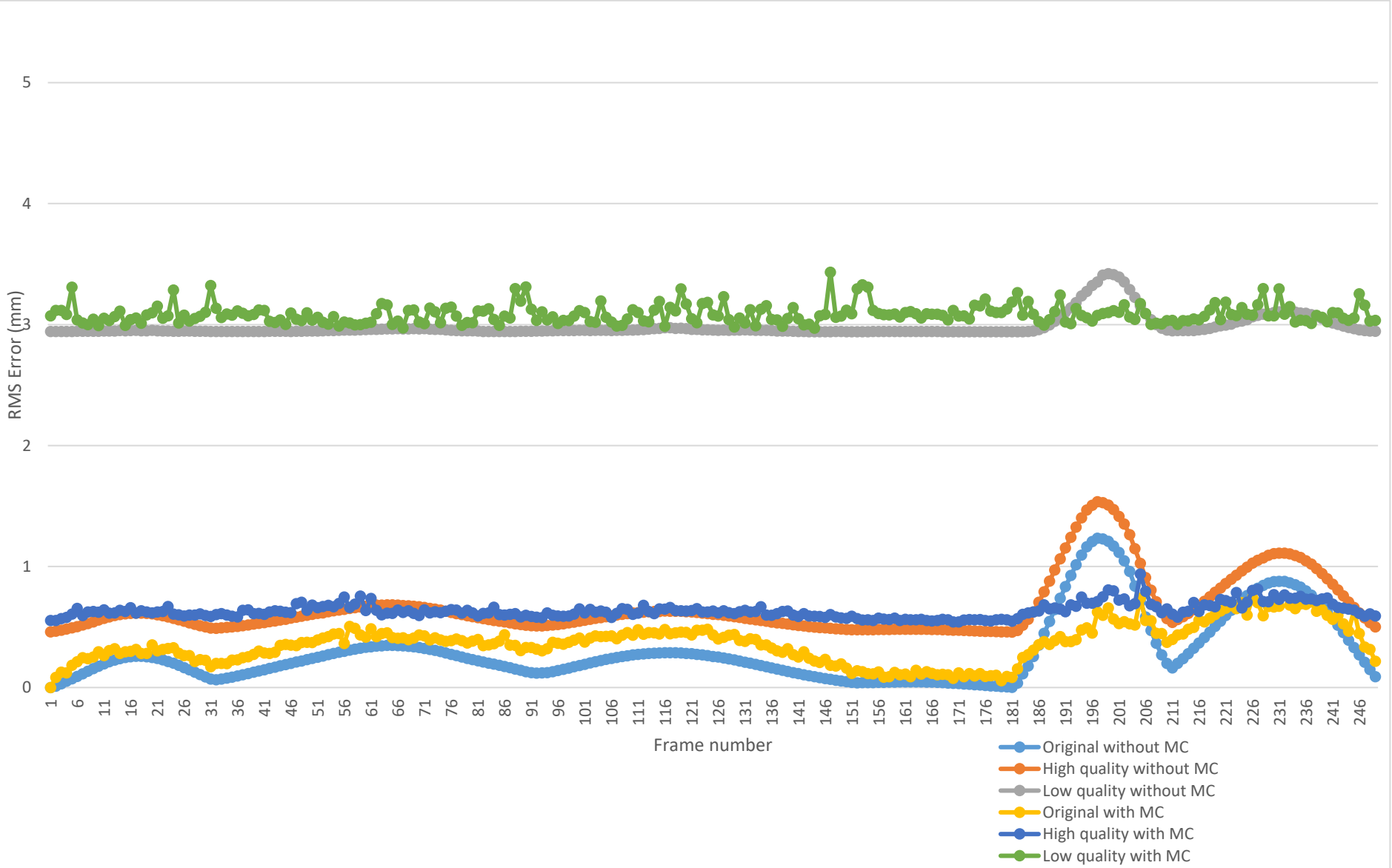

Figure 46: Queen rms error temporal evolution.

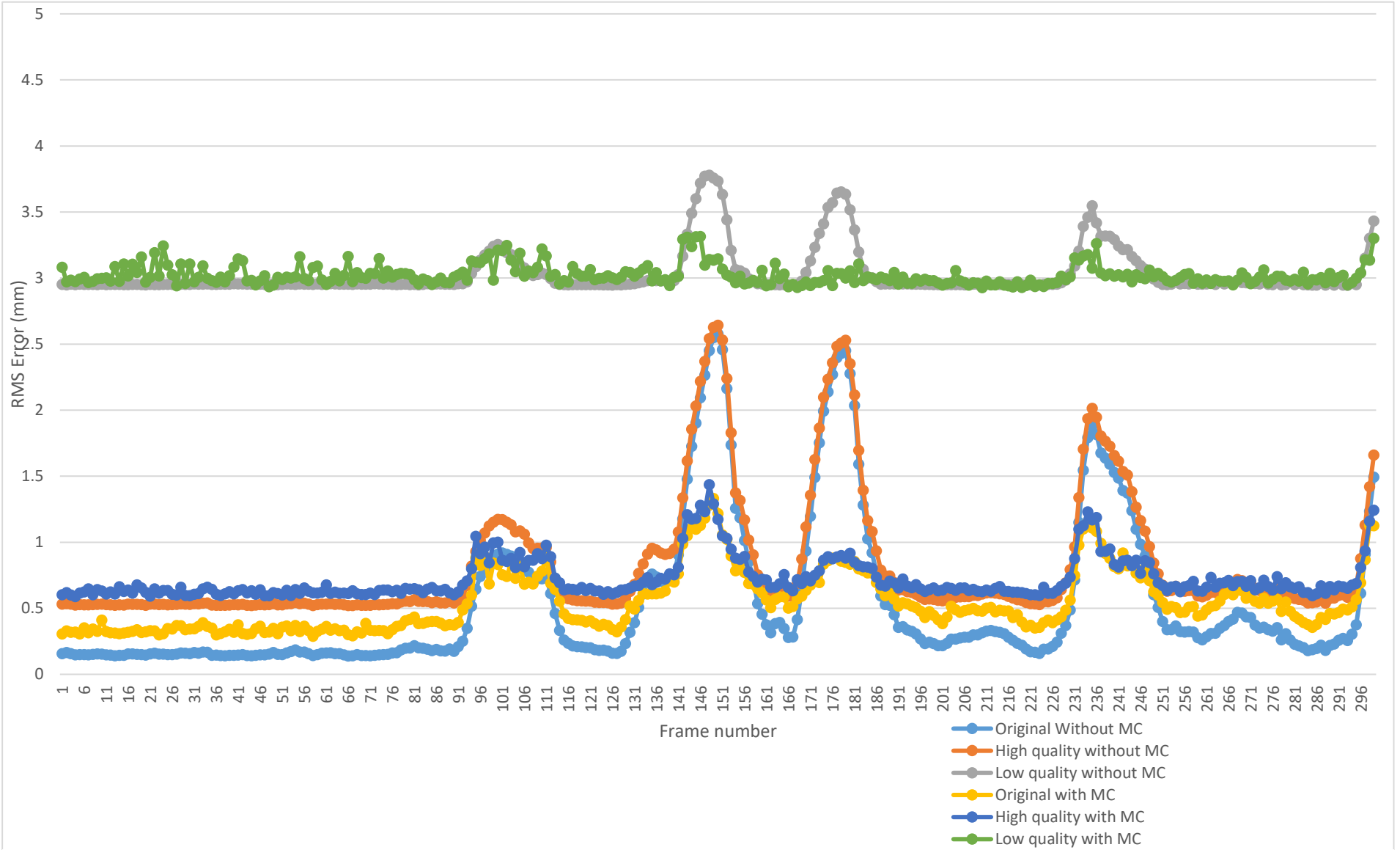

Figure 47: Soldier rms error temporal evolution.

References

[1] Microsoft, “Microsoft HoloLens”, [Online]. Available: https://www.microsoft.com/microsoft-hololens/en-us. [Accessed on 20 10 2016].

[2] “Dense Modelling”, [Online]. Available: http://grail.cs.washington.edu/rome/dense.html. [Accessed on 30 12 16].

[3] “3Data”, [Online]. Available: http://3d-ata.com/en/portfolio-item/tls-applicatons-plecniks-river-barriers. [Accessed on 30 12 16].

[4] Aeryon Labs Inc., “Surveying and mapping”, [Online]. Available: https://www.aeryon.com/commercial/survey-and-mapping. [Accessed on 31 12 2016].

[5] R. Mekuria. K. Blom, P. Cesar, “Design, implementation and evaluation of a point cloud codec for tele-immersive video,” IEEE Transactions on circuits and systems for video technology, (to be published), March 2016.

[6] MPEG Doc. N16334, “Draft call for proposals for point cloud compression,” ISO/IEC JTC1/SC29/WG11 WG11, Geneva, Switzerland, June 2016.

[7] JPEG Doc. N72006, “JPEG PLENO – Scope, use cases and requirements ver.1.5,” ISO/IEC JTC1/SC29/WG11 WG11, Geneva, Switzerland, June 2016.

[8] F. Pereira, E.A.B. Silva, G. Lafruit, “Plenoptic imaging: representation and processing,” (to be published), April 2016.

[9] E. H. Adelson, J. R. Bergen, “The plenoptic function and the elements of early vision,” The MIT Press, Computation models of visual processing, pp. 3-20, Cambridge, Massachusetts, USA, 1991.

[10] K. Wegener, “Multi-view video acquisition system”, [Online]. Available: http://pt.slideshare.net/KrzysztofWegner/poznan-multiview-video-acquisition-system. [Accessed on 07 10 2016].

[11] “Wikipedia on light field”, [Online]. Available: https://en.wikipedia.org/wiki/Light_field. [Accessed on 08 10 2016].

[12] “Point Cloud Library (PCL)”, [Online]. Available: http://pointclouds.org. [Accessed on 08 10 2016].

[13] J. Peng, C. Kim, C. Kuo, “Technologies for 3D mesh compression: a survey,” Journal of visual communication and image representation, vol. 16, no. 6, pp. 688–733, December 2005.

[14] “Autodesk 3DS Max 2014”, [Online]. Available: http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-Help/index.html?url=files/GUID-49CE0ACB-1345-4D50-B6E5-361DBFDB5B33.htm,topicNumber=d30e158270. [Accessed on 15 10 2016].

[15] “Geometric modelling: digital representation and analysis of shapes“, [Online]. Available: http://www.enseignement.polytechnique.fr/informatique/INF555/Slides/lecture7.pdf. [Accessed on 15 10 2016].

[16] “Wikipedia on Holography”, [Online]. Available: https://en.wikipedia.org/wiki/Holography. [Accessed on 15 10 2016].
[17] A. Smolic, “3D video and free viewpoint video - From capture to display,” Pattern recognition, vol.44, nº 9, pp. 1958-1968, New York, USA, September 2011.
[18] C. Tulvan, R. Mekuria, Z. Li, “Use cases for point cloud compression (PCC),” ISO/IEC JTC1/SC29/WG11 MPEG2015/ N16331, Geneva, Switzerland, June 2016.
[19] Microsoft, “Holoportation”, [Online]. Available: https://www.microsoft.com/en-us/research/project/holoportation-3. [Accessed on 20 10 2016].
[20] “freeD™ free dimensional video”, [Online]. Available: http://replay-technologies.com. [Accessed on 20 10 2016].
[21] “Wikipedia on point cloud”, [Online]. Available: https://en.wikipedia.org/wiki/Point_cloud. [Accessed on 20 10 2016].
[22] “ScanLAB projects 3D scanning”, [Online]. Available: http://scanlabprojects.co.uk/3dscanning. [Accessed on 20 10 2016].
[23] “Culture 3D cloud”, [Online]. Available: http://c3dc.fr. [Accessed on 20 10 2016].
[24] A. Collet, et al., “High-quality streamable free-viewpoint video,” ACM transactions on graphics, vol. 34, no. 4, August 2015.
[25] M. Zhang, ”Creating 3D portraits using an array of digital cameras”, [Online]. Available: http://petapixel.com/2013/01/21/creating-3d-portraits-using-an-array-of-digital-cameras, January 2013. [Accessed on 28 11 2016].
[26] R.J. Valkenburg, A.M. McIvor, “Accurate 3D measurement using a structured light system,” Image and vision computing, vol. 16. no. 2, pp. 99-110, February 1998.
[27] Microsoft, “Kinect for Xbox 360”, [Online]. Available: http://www.xbox.com/en-US/xbox-360/accessories/kinect. [Accessed on 18 11 2016].
[28] S. B. Gokturk, H. Yalcin, C. Bamji, “A time-of-flight depth sensor – system description, issues and solutions,” Conference on computer vision and pattern recognition workshop, Washington, D.C., USA, July 2004.
[29] S. Sangam, “Light detection and ranging”, Department of electronics and communication engineering, CMR Institute of Technology, Bangalore, India, August 2012.
[30] Heptagon, “SwissRanger”, [Online]. Available: http://hptg.com/industrial. [Accessed on 18 11 2016].
[31] R. Mekuria, P. Cesar, “MP3DG-PCC, open source software framework for implementation and evaluation of point cloud compression,” Proceedings of the 2016 ACM multimedia conference, pp. 1222-1226, Amsterdam, Netherlands, October 2016.
[32] MPEG Doc. N16332, “Evaluation criteria for PCC (point cloud compression),” ISO/IEC JTC1 SC29 WG11, Geneva, Switzerland, February 2016.
[33] MPEG Doc. N16333, “Draft dataset for point cloud coding (PCC),” ISO/IEC JTC1 SC29 WG11, Geneva, Switzerland, June 2016.
[34] D. Tian, H. Ochimizu, C. Feng, R. Cohen, A. Vetro, “Evaluation metrics for point cloud compression,” ISO/IEC JTC 1/SC29/WG1, (to be reviewed), 73rd meeting, Chengdu, China, October 2016.

[35] Information Technologies Institute “Visual computing lab”, [Online]. Available: http://vcl.iti.gr/reconstruction. [Accessed on 10 12 2016].

[36] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, E. Steinbach, “Real-time compression of point cloud streams,” IEEE International Conference on Robotics and Automation, St. Paul, Minnesota, USA, May 2012.

[37] D. Thanou, P. A. Chou, P. Frossard, “Graph-based compression of dynamic 3D point cloud sequences,” IEEE transactions on image processing, vol. 25, no. 4, April 2016.

[38] J. Katajainen, E. Mäkinen, “Tree compression and optimization with applications,” International journal of foundations of computer science, vol. 1, no. 4, pp. 425–447, 1990.

[39] C. Zhang, D. Florêncio, C. Loop, “Point cloud attribute compression with graph transform,” IEEE international conference on image processing, pp. 2066-2070, October 2014.

[40] C. Loop, C. Zhang, Z. Zhang, “Real-time high-resolution sparse voxelization with application to image-based modeling,” High-performance graphics conference, pp. 73–79, California, USA, July 2013.

[41] R. Rusu, N. Blodow, Z. Marton, and M. Beetz, “Aligning point cloud views using persistent feature histograms,” International Conference on Intelligent Robots and Systems, Nice, France, September 2008.

[42] R. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” International Conference on Robotics and Automation, Kobe, Japan, May 2009.

[43] “The Stanford 3D Scanning Repository”, [Online]. Available: http://graphics.stanford.edu/data/3Dsc anrep. [Accessed on 14 08 17].

[44] R. Rusu, “Semantic 3D object maps for everyday manipulation in human living environments,” PhD Thesis, Technical University of Munich, 2010.

[45] L. A. Alexandre, “3D Descriptors for Object and Category Recognition: a Comparative Evaluation,” IEEE International Conference on Intelligence Robotic Systems, Vilamoura, Portugal, October 2012.

[46] “Wikipedia on k-d tree”, [Online]. Available: https://en.wikipedia.org/wiki/K-d_tree. [Accessed on 10 05 2017].

[47] A. Nguyen, B. Le, “3D Point Cloud Segmentation: A Survey,” IEEE Conference on Robotics, Automation and Mechatronics, Manila, Philippines, November 2013.

[48] J. Papon, A. Abramov, M. Schoeler, F. Worgotter, “Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds,” IEEE Conference on Computer Vision and Pattern Recognition, Portland, USA, October 2013.

[49] MPEG Doc. N16763, “Call for Proposals for Point Cloud Compression V2,” ISO/IEC JTC1/SC29/WG11 WG11 MPEG2017, Hobart, Australia, April 2017.

[50] “MPEG Datasets”, [Online]. Available: http://157.159.160.118/MPEG/PCC/DataSets/pointCloud/. [Accessed on 15 03 2017].

[51] 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) WG11M40059/WG1M74006, Geneva, Switzerland, January 2017.

[52] “Point Feature Histograms (PFH) Descriptors”, [Online]. Available: http://pointclouds.org/documentation/tutorials/pfh_estimation.php. [Accessed on 25 09 2017].

[53] C. Harris, M. Stephens, “A combined corner and edge detector,” Alvey vision conference, pp. 147–152, Manchester, England, 1998.
[54] R. Rusu, N. Blodow, Z.C. Marton, M. Beetz. - “Aligning Point Cloud Views using Persistent Feature Histograms”, In Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, September 22-26 2008.
[55] A. Frome, D. Huber, R. Kolluri, T. Bulow, J. Malik, “Recognizing Objects in Range Data Using Regional Point Descriptors,” European conference on computer vision, vol. 3023, pp. 224-237, 2004.
[56] F. Tombari, S. Salti, L. Di Stefano, “Unique shape context for 3d data description,” Proceedings of the ACM workshop of 3D object retrieval, pp. 57-62, Firenze, Italy, October 2010.
[57] S. Lazebnik, C. Schmid, J. Ponce, “A sparse texture representation using local affine regions,” IEEE transactions on pattern analysis and machine intelligence, vol. 27, no. 8, pp. 1265-78, August 2005.
[58] Y. Rubner, C. Tomasi, L. J. Guibas, “The Earth Mover's Distance as a Metric for Image Retrieval,” International journal of computer vision, vol. 40, no. 2, pp. 99-121, November, 2000.
[59] F. Tombari, S. Salti, and L. Di Stefano, “Unique signatures of histograms for local surface description,” Proceedings of the 11th european conference on computer vision, pp. 356-369, Berlin, 2010.
[60] B. Drost, M. Ulrich, N. Navab, and S. Ilic, “Model globally, match locally: Efficient and robust 3d object recognition,” IEEE Conference on computer vision and pattern recognition, pp. 998–1005,San Francisco, USA, 2010.
[61] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, “Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram,” International conference on intelligent robots and systems, Taipei, Taiwan, October 2010.
[62] A. Aldoma, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, M. Vincze, and G. Bradski, “Cad-model recognition and 6 dof pose estimation,” 3D representation and recognition, November, 2011.
[63] W. Wohlkinger, M. Vincze, “Ensemble of shape functions for 3D object classification,” IEEE international converence on robotics and biomimetics, pp. 2987 –2992, December, 2011.
[64] S. C. Stein, M. Schoeler, J. Papon, F. Worgotter, “Object Partitioning using Local Convexity,” IEEE Conference on Computer Vision and Pattern Recognition, June, 2014.
[65] M. Schoeler, J. Papon, F. Worgotter, “Constrained Planar Cuts - Object Partitioning for Point Clouds,” IEEE Conference on Computer Vision and Pattern Recognition, October, 2015.
[66] L. Gardy, “Random Walks for Image Segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 11, pp. 1768-1783, November, 2006.
[67] A. Golovinskiy, T. Funkhouser, “Min-Cut Based Segmentation of Point Clouds,” IEEE International Conference on Computer Vision Workshops, September, 2009.
[68] Y. Ioannou, B. Taati, R. Harrap, “Difference of Normals as a Multi-Scale Operator in Unorganized Point Clouds,” IEEE International Conference on 3D imaging, Modeling, Processing, Visualization and Transmission, October, 2012.