UNIVERSIDADE
BEIRA INTERIOR

Static Point Clouds Compression efficiency of MPEG point clouds coding standards

A practical comparison between two powerful PCC codec's

João Pedro Casanova Prazeres

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e de Computadores
(2º ciclo de estudos)

Orientador: Prof. Doutor António Manuel Gonçalves Pinheiro

Junho 2020

Acknowledgements

First and foremost, to my parents, for without them none of my achievements would have ever been possible. To the IT lab and its personnel, for making the material available for testing, and guidance throughout the development of this thesis. To my girlfriend Catarina, for all the love and support throughout this journey, and for putting up with me when sometimes even i would not put up with myself. To my band Divine Ruin, to the people of Skyline, and all my closest friends, for all the stress relieving moments with the most honest and trustworthy persons i know, and for being a living proof that you can still find real friends in such a shallow world. And to my supervisor Prof. Doutor antónio Pinheiro, for all the patience, guidance, comprehension, motivation, and overall friendship.

Contents

List of Figures

List of Tables

Resumo

Os avanços recentes no consumo de conteúdo 3D vêm criar a necessidade de maneiras eficientes de visualizar e transmitir conteúdo 3D. Consequentemente, os métodos de obtenção desse mesmo conteúdo têm vindo a evoluir, levando ao desenvolvimento de novas maneiras de representação, nomeadamente point clouds e lightfields. Um point cloud (núvem de pontos) representa um conjunto de pontos com coordenadas cartesianas associadas a cada ponto (x, y, z), além de poder conter mais informação dentro do mesmo (cor, material, textura, etc). Este tipo de representação abre uma nova janela na maneira como se consome conteúdo 3D, tendo um elevado leque de aplicações, desde videojogos e realidade virtual a aplicações médicas. No entanto, este tipo de dados, ao carregarem com eles tanta informação, tornam-se incrivelmente pesados, tornando o seu armazenamento e transmissão uma tarefa hercúleana. Tendo isto em mente, a MPEG criou um projecto de normalização de codificação de point clouds, dando origem ao V-PCC (Video-based Point Cloud Coding) e G-PCC (Geometry-based Point Cloud Coding) para conteúdo estático. Esta dissertação tem como objectivo uma análise geral sobre os point clouds, indo desde as suas possívei utilizações à sua aquisição. Seguidamente, é efectuado um estudo dos codificadores de point clouds, nomeadamente o V-PCC e o G-PCC da MPEG, o estado da arte da avaliação de qualidade, objectiva e subjectiva, e finalmente, são reportadas as actividades da JPEG Pleno Point Cloud, na qual se teve uma colaboração activa.

Palavras chaves: Point clouds;Compressão;testes subjectivos;point clouds estáticos;métricas

Abstract

Recent advances in the consumption of 3D content creates the necessity of efficient ways to visualize and transmit 3D content. As a result, methods to obtain that same content have been evolving, leading to the development of new methods of representations, namely point clouds and light fields. A point cloud represents a set of points with associated Cartesian coordinates associated with each point (x, y, z), as well as being able to contain even more information inside that point (color, material, texture, etc). This kind of representation changes the way on how 3D content in consumed, having a wide range of applications, from video-gaming to medical ones. However, since this type of data carries so much information within itself, they are data-heavy, making the storage and transmission of content a daunting task. To resolve this issue, MPEG created a point cloud coding normalization project, giving birth to V-PCC (Video-based Point Cloud Coding) and G-PCC (Geometry-based Point Cloud Coding) for static content. Firstly, a general analysis of point clouds is made, spanning from their possible solutions, to their acquisition. Secondly, point cloud codecs are studied, namely V-PCC and G-PCC from MPEG. Then, a state of art study of quality evaluation is performed, namely subjective and objective evaluation. Finally, a report on the JPEG Pleno Point Cloud, in which an active colaboration took place, is made, with the comparative results of the two codecs and used metrics.

Keywords: Point clouds;Compres;Subjective Testing;Static Point Cloud;Metrics

Acronyms

2D	Two Dimensional
3D	Three Dimensional
BRDF	Bidirectional Reflectance Distribution Function
CAD	Computer-Aided Design
DCM	Direct Coding Mode
EDD	Enhanced-Delta-Pack
EOM	Enhanced Occupancy Map
G-PCC	Geometry-based Point Cloud Coding
GBR	Green Blue Red
GOF	Group of Frames
IoU	Intersection over Union
IR	Infrared
HW	Hardware
K-D	K-Dimensional
LoD	Level of Detail
LiDAR	Light Detection and Ranging
MOS	Mean Opinion Score
MPEG	Movie Picture experts Group
MSE	Mean Squared Error
MV	Motion Vector
NC	Neighbour Configuration
PCM	Pulse Code Modulation
po2point	Point-to-Point

po2plane	Point-to-Plane
PSF	Packing strategy Flag
PSNR	Peak-to-Signal Noise Ratio
RAHT	Region-Adaptive Hierarchical Transform
RGB	Red Green Blue
RMS	Root Mean Squared
SIFT	Scale-Invarant Feature Transform
TH	Threshold
TMC2	Test model Category 2
Tof	Time of Flight
TriSoup	Triangle Soup
V-PCC	Video-based Point Cloud Coding
WxH	Width x Height
YUV	Luminance and Chrominances

1 Introduction

This thesis will show a practical analysis between two powerful PCC softwares, as well as a comparison between popular quality metrics. In 2, various ways of acquiring point clouds will be described. In 3, the applications of point clouds are described, as well as requirements for their coding. In 4, a description of representation models is described, followed by rendering and displaying. Sections 6 and 7, both codecs are explained in detail. Sections 8 and 9 explain the methods to undertake subjective testings and how the most popular quality metrics work, and finally section 10 describes the experiments taken to compare both codecs, and a description of how to use them to code point cloud content.

2 Point cloud data

There is a variety of ways in which point cloud data can be obtained. As such, there are multiple possibilities for point cloud acquisition technologies categorisation. Here, they will be categorised in two domains, direct and indirect acquisition.

2.1 Direct acquisition

Point cloud data direct acquisition can be defined as imaging systems designed only to collect 3D information. That acquisition can come in the form of wither sparse points in a 3D space, or of a dense cloud where each pixel on the sensor has an associated depth value, allowing a conversion into a dense point cloud.

2.1.1 Photogrammetry with structured light

The objects are illuminated with structured light patterns, and the camera captures the resulting scene. The point cloud can be created using the deformation of the light patterns caused by the objects 3D shape, achieved using the deformation of a portion of the projected pattern on the surface of the imaged object, to determine the 3D position of the point on the object [1]. The determination of the 3D structure of surfaces without features, such as a walls and floors is the main advantage over photogrammetry under ambient illumination. Photogrammetry without structured light tries to match the image point across different views, and often fails when there is no sufficient visible texture on the image.

2.1.2 LiDAR

Light Detection and Ranging is an active 3D scanning technology using a beam of laser light in the ultraviolet, visible or Infra Red (IR) part of the light spectrum. LiDAR systems differ from laser structured light systems because the distance is not measured by the time a single laser pulse takes to return. Instead in a a laser structured light system, the distance is measured by the deformation of the appearance of a uninterrupted laser stripe or pattern caused by the 3D structure of objects. With LiDAR, a single pulse of light is sent from the laser source, and the 3D data from the scene is extracted by the structure and time profile of the receiving signals. The pulse of laser light in LiDAR systems allows the use of much larger energies and the cover of larger distances. A single laser beam can achieve higher resolutions and accuracy[2, 3]. The properties of the laser beam can be modified to target various objects, clouds and even rain, to suit the desired application. Some systems use an arrangement of rotating mirrors to sweep a pulsing laser beam across the scene, and the detection is done by measuring the time of flight [4]. The disadvantage of LiDAR compared to other system lies in not collecting texture or colour information of the objects being scanned, but works in situations where photogrammetry fails, such as through thick vegetation [5].

2.1.3 Time-of-flight

The Time-of-flight (ToF) camera is a range imaging camera system which measures distance based on the known speed of light, calculating the time of flight of the signal between the camera and the subject for each point of the image. Though the resolution for these kind of cameras is usually low (sensor size ranging from 100 to 500 pixels), these systems are able to capture images in the order of tens to hundreds of images per second [6]. The two most common types, able to output a point cloud representation of the imaged scenes are Photonic Mixed Devices and Range-Gated Imagers.

2.2 Indirect Acquisition

Indirect acquisition can be defined as algorithms applied to create point cloud data from sources which do not directly measure 3D information. Algorithms for extraction of point cloud data from sets of 2D images, or light field data. Sometimes, the algorithms extract dense depth data.

2.2.1 Photogrammetry

In this method, 3D point clouds are created from RGB imagery, by matching points across multiple images of the object or scene, all taken with different view points [7]. Generally, a differentiating point in an image is identified, and matched afterwards to another view of the same scene. The features suitable for matching can be identified by SIFT [8], or other algorithms for local feature detection [7]. Knowing the relative position of the cameras, the search for a point in the second image is restricted to a line, by the epipolar constraint, and having multiple cameras can constrain the search zone even further [7].

2.2.2 Light field data

The popularity of the light field cameras has been increasing in the last decade [9]. They are not only able to capture the light intensity in RGB spectral bands, but also the intensity of the light leaving the scene, according to the angle in a particular point. This can be used to re-focus, changing the viewpoint, depth of field, extraction of 3D information from the image, all of which can be used for point cloud representation [10].

3 Point clouds use cases

In this section possible cases where point cloud representations might play an important role are described, followed by the respective quality requirements.

3.1 Virtual, augmented and mixed reality

Point clouds play an important role in supporting the display of 3D content in virtual, augmented and mixed reality environments [11]. That content can be generated by CAD, or created with 3D scanning equipment. The resolution and number of point will vary, depending mainly on the type of object, or 3D scanner. Point clouds that were created using CAD software will most likely be arranged on regular grids and patterns, contrary to data collected with 3D scanning, which will be arranged in an irregular geometric pattern, unless re-meshed. Important attributes of data used in this applications are colour, gloss, bump or texture maps, as well as a bidirectional reflectance distribution function (BRDF). This type of point clouds needs to be rendered at frames ranging from 5 to 30 fps. Due to this kind of application introducing stringent requirements on the end-to-end media chain, real-time and low latency are of the utmost importance to this case. Progressive coding, view selectivity, region of interest and resolution granularity/scalable bit stream capabilities may be also of importance, as well as the ability to allow the tuning of the quality of the decoded point cloud, making it suitable for displaying in different devices. Some attributes may be encoded in a lossy way (intensity, bump and texture maps, BRDF information), but the need of support for attributes with lossless encoding may be necessary if such attributes support look up or reference a database (e.g database of material properties)

3.2 3D content creation

3.2.1 Design, manufacturing and 3D printing

This type of application mainly involves point clouds specifically created to support the content for 3D printing and traditional manufacture. The data will most likely be arranged on regular grid patterns associated with CAD software; the resolution of the point will be highly dependent on the type of object and the industry in which the object will be used. The point cloud attributes should include colour, material appearance information, most likely linked to databases containing information about used materials. Since this case requires accurate representation of point clouds, lossless representation will be important. Additionally, support for attributes with lossless coding will likely be needed, in the cases where the attribute look to an external database. To safeguard intellectual property, data privacy and security should be associated with the content, with some form of encryption.

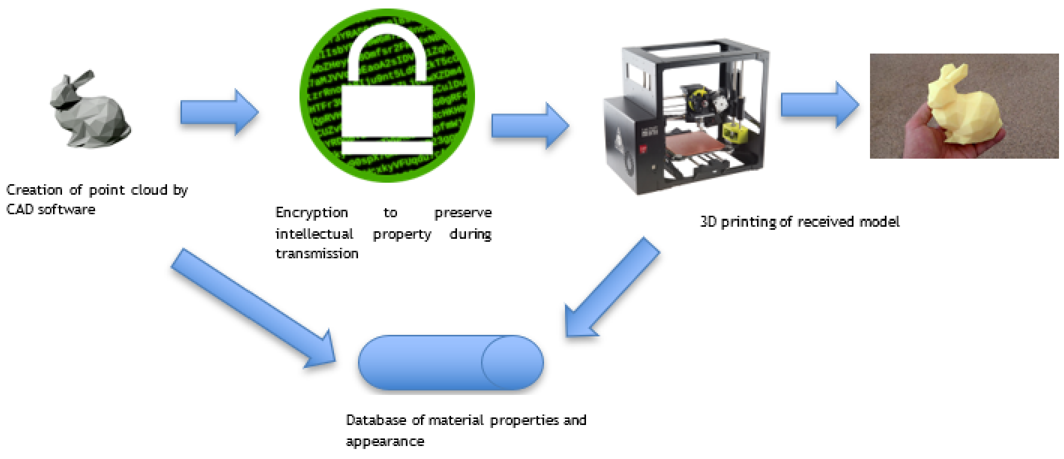

Creation of point cloud by CAD software
Encryption to preserve intellectual property during transmission
3D printing of received model
Database of material properties and appearance

Figure 1: Usage of CAD created point cloud for 3D printing

3.2.2 Motion capture

Motion capture is the process of recording the movement of objects or people, and can be done with or without markers. When done with markers, it often results in point clouds where points correspond to each marker. These type of data capture are important for movement analysis and used in a variety of applications such as sports analysis and movie production. Point clouds also are a useful representation for data resulting from multiple sensors. The priority in this type of cases is and efficient and inter-operable coding and storage methods, sharing some similarities with the previously described in 3.2.1

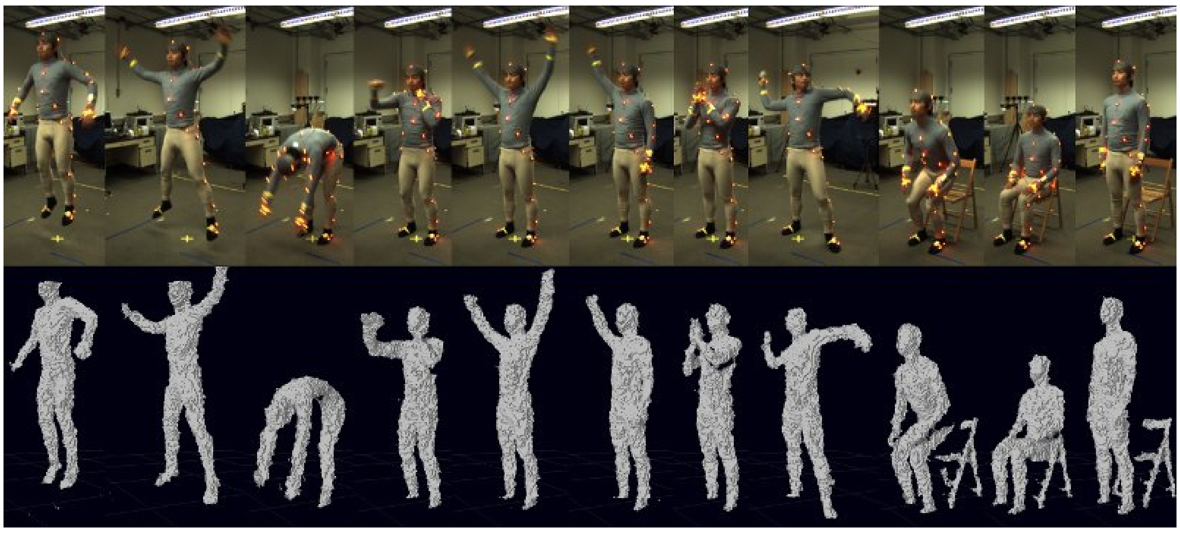


Figure 2: Motion capture example

3.3 Medical applications

3.3.1 3D medical image

Refers to the point clouds obtained by 3D scanning of internal or external human or animal anatomy, with the purpose of research, medical record keeping, diagnosis and preparation of treatment. This type of point clouds need to be encoded in a way that preserves the ability for subsequent analysis, thus requiring an accurate representation of point clouds, meaning that lossless or very low coding will be prioritized. It also may require multiple scans of anatomical structures and internal detail, potentially needing to keep point clouds from individual scans distinct in the coded format. To support examination by medical professionals, view selectivity, region of interest and resolution granularity/scalable bit streams capabilities will most likely be important. Although some point attributes may be encoded by a lossy method, support for attributes with lossless coding will be needed where such attributes are crucial for diagnosis and analysis. There should also be a requirement to provide means to guarantee the privacy and security needs associated with content.

3.3.2 Prosthesis and body parts design and manufacture

This refers to artificially created point clouds to support manufacturing 3D prosthesis. The data will most likely be arranged on regular grids and its patterns should be associated with the CAD software. The data may also be arranged in an irregular geometric pattern associated with point clouds captured from scanned body parts. This type of point clouds should contain attributes such as colour, material appearance information, potentially linked to an external database. Also, highly accurate colour and material appearance representations should be present, due to the need of matching the prosthetic against skin. The requirements for this case are similar to the ones in 3.2.1, although the security should be focused in protecting patient confidentiality, instead of intellectual property.

3.4 Construction and manufacturing

This use case can be divided in 4 sections:

1. Analysis for defect detection: Scanning parts and structures to find possible existing anomaly's.
2. Scanning to support project management: Displaying of 3D content for architecture and infrastructure visualisation.
3. Repair and analysis: Scanning of confined or dangerous places, in order to find flaws in objects.
4. Urban planning and analysis: Record of 3D shapes and details in a large urban area, usually with objects spanning more than 20 meters in extent.

Most cases share the same attributes, such as color and material, with the exception of 4 (usually color is not needed for urban planning), and 2 has the added attributes of gloss, bump, texture maps and BRDF, and the requirements for point cloud coding revolve around lossless coding or low loss coding, in order to preserve sharp edges and fine detail.

3.5 Consumer and retail

In this use case, point clouds record 3D information of:

1. Small objects, such as jewelry, decorations, shoes, etc.
2. Mid-size objects, such as cars or furniture, and large objects, such as houses.

In both situations, the point clouds will have complex structure and material appearance, and high resolution. Mid-size and large objects, may require multiple scans to handle object internal structure and occlusions. It is also possible to merge multiple scans into a whole object. Point clouds in this section share the requirements of Virtual, augmented and mixed reality section, as they are often rendered on websites and mobile devices, with the additions of perceptually lossless rendering, and means to guarantee the protection of intellectual property. Mid-sized and large objects also need support for separate point clouds, for interactivity purposes.

3.6 Cultural heritage

In this use case, point cloud scans are used to visualise and archive cultural heritage in objects and collections. They can be divided into:

1. Small artefacts, such as pottery, bones or fossils.
2. Paintings or murals
3. Mid-sized artefacts, such as statues
4. Large artefacts such as facades or monuments

The point clouds in all cases have complex material appearance and structure, and high resolution, and with the exception of small artefacts, there is a need to scan the object multiple times. This type of point clouds requires progressive and lossless coding, in order to increase quality, and enable the best possible representation and means to protect the cultural property.

(a)

(b)

Figure 3: Examples of point cloud scan of cultural heritage. (a) A bust of Zeus (b) A roman oil lamp

3.7 Remote sensing and geographical information systems

3.7.1 Wide area scanning

This point clouds store information of objects greater than 20 meters. During the capture multiple scans should be required for a reconstruction of the scene, and as such,it is unlikely to manage an appropriate control of the lighting conditions. Colour may be present and usually there is no presence of detailed material appearance such as gloss. It shares requirements with 3.6, but usually it does not require colour accuracy or material appearance.

3.8 Autonomous vehicles, drones

The main purpose here is to spot objects such as obstructions or pedestrians. Resolution may vary with the direction in which the vehicle is moving, and it may or may not require accurate colour and material appearance. The point clouds require lossless or very high quality lossy coding. The representation should preserve sharp edges and fine details, and requirements for real time and low latency are necessary, as there is a need to collect and analyse incoming data continuously.

3.9 Surveillance

Here the aim is to analyze point clouds containing information of objects up to 10 meters in extent, to find objects of interest, such as a trapped person. Here, the point cloud may not require accurate colour or material appearance information, due to some potentially chaotic environments, although it should be considered, as it may help in identification.

4 Representation models

Point clouds are sets of points in a 3D space, with each point associated attributes (colour, material). The scene or object being scanned can be very large, and since a high spatial sampling density is required, each point cloud can have millions of points. Some point cloud capturing devices also export their data in text format, resulting in point cloud with hundreds of megabytes to be stored. Due to the reasons above described, an efficient compression method is important, allowing efficient storage/transmission of point cloud data.

4.1 Octree

One of the most popular representations for point cloud data are octree structures [12, 13]. They can be described as a tree data structure, with each internal node having eight children. A box-shaped region containing at least one point belonging to the point cloud is divided into eight smaller regions, without any gaps or overlaps. This process will be repeated until the point cloud is decomposed to the desired level or precision. Quick search in the neighbour points, or representation of an octant's points by a centroid or medoid, are some of the advantages of octrees.

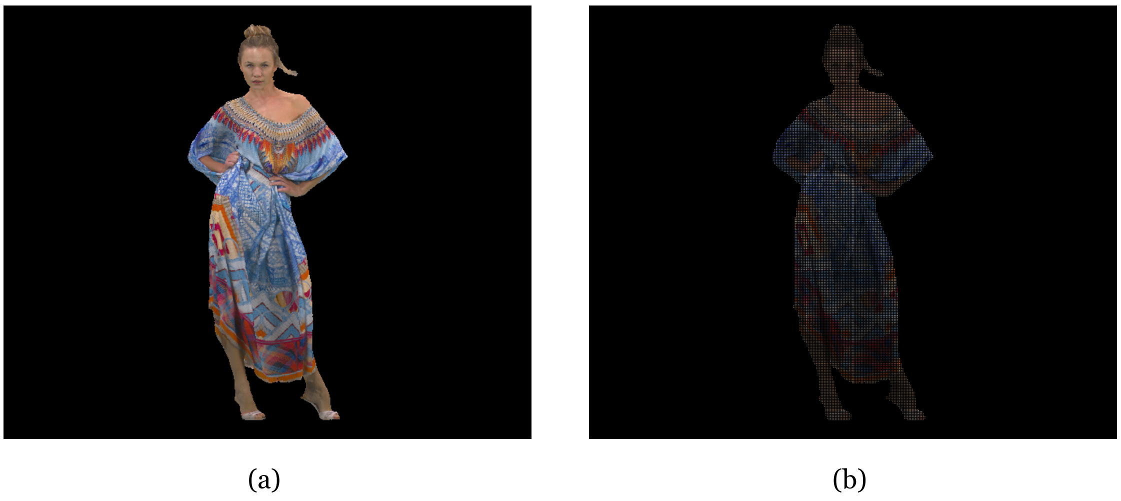


Figure 4: (a) - Original Content, (b) - Content compressed with octree

4.2 Meshes

Point clouds are converted into polygonal meshes using a vast range of methods [14, 15, 16]. Triangles can be used as as polygons, requiring information on point connectivity. Normal information and material reflectance are commonly used attributes. Compression of polygonal meshes can be done by reducing the points through simplification methods. After the mesh construction, it can be simplified through merging nodes, without altering the local structure.

4.3 Voxelized representation

Similar to octree, voxel grid is a spatial data representation structure based on a sub-sampling technique applied on point clouds, resulting in a set of vertices lying on a regularly spaced grid of fixed depth, in which the vertices are represented by voxels that can be occupied or unoccupied, and have a colour value associated. If more than one point falls in the same voxel, its colour value is an average of the point colour values, and its resolution can be set either manually or automatically.

5 Point cloud rendering and display

In section 3 are described the main use cases of point clouds. In many of those applications, the most relevant aspect is the geometry, because point clouds are mostly often used as input for processing, rather than direct observation. Nonetheless, for applications where the final user is a human observer, there is the necessity of converting point cloud data into a displayable representation, allowing distinction between shape and texture of the object. A common way to solve this issue is to render a surface approximating the shell of the originally scanned objects, during the point cloud acquisition.

5.1 Implicit representations

Methods in which the surface represented by the point cloud on acquisition is approximated by a level set of a scalar function f_s with domain in R^3. That level is usually the joining of R^3 points in which the function has its zeros and we can define the surface approximation as [17] :

$$S = p \in R^3 \, s.t \, f_s(p) = 0 \quad (1)$$

The function f_s is obtained by the weighted sum of radial basis functions, which is then fitted to the cloud of points by solving a set of equations that represent as many constraints as the number of points. For the application of this method, normal information at each point is required. If not present, it must be estimated, making a method with already complex calculations, even more cumbrous and likely to fail in regions with low point density. This class of methods is obviously not suitable for point clouds with large numbers of point.

5.2 Explicit representations

This class of methods is used mainly for point clouds dense enough so that the proximity of neighbour points in the cloud data means that the matching 3D scene points are also neighbours, thus the approximation of the local surface is based on simple polygons [18]. Those approximations can be obtained using Delaunay triangulation or Voronoi regions decomposition, resulting in a piecewise planar full surface approximation, which can be smoothed during the final projection, if desired.

6 V-PCC

MPEG V-PCC [19] encodes point clouds using an approach based on projecting each point cloud onto a set of planes, followed by 2D encoding of those projections. At the encoding stage input point, the cloud frame is processed in the following manner. First, the volumetric 3D data has to be represented as a set of 3D projections in different components. At the separation stage, the image is decomposed into far and near components for geometry, and corresponding attributes components. Also, an occupancy map 2D image is created to indicate parts of an image that shall be used. This projection is composed of independent patches based on geometry characteristics of the input point cloud frame. After patch generation, and creation of the 2D frames for video encoding, geometry, attribute and auxiliary information may be compressed. The geometry information may additionally be smoothed outside the encoding loop, and additional smoothing parameters that were used in the point cloud smoothing process, may be transferred as supplemental information for the decoding process. At the end of the process, the separate bitstreams are multiplexed into the output compressed binary file. The next image shows the encoding structure used in VPCC.

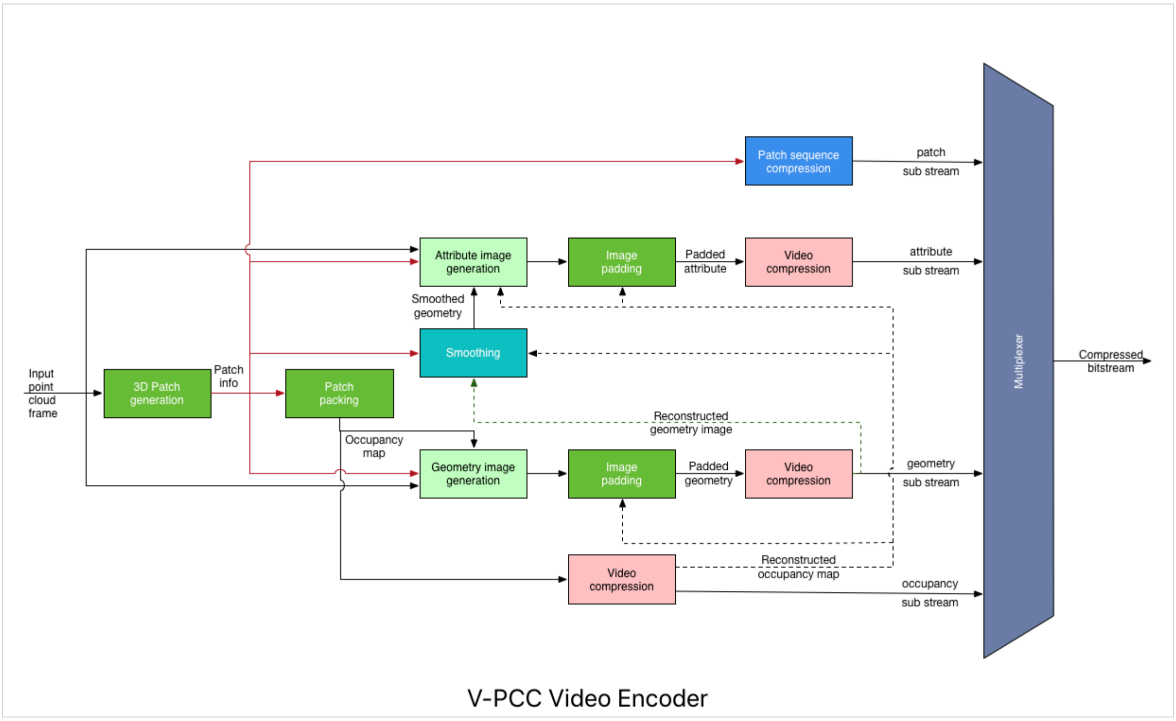


Figure 5: VPCC Encoding Structure

The decoding process begins by demultiplexing the input of the compressed binary file into geometry, attribute, occupancy map and auxiliary information stream Based on the decoded attribute video stream, and reconstructed information from smoothed geometry, if present, occupancy map, auxiliary information, and attributes of the point cloud can be reconstructed. After reconstruction, an additional attribute smoothing method is used for refinement

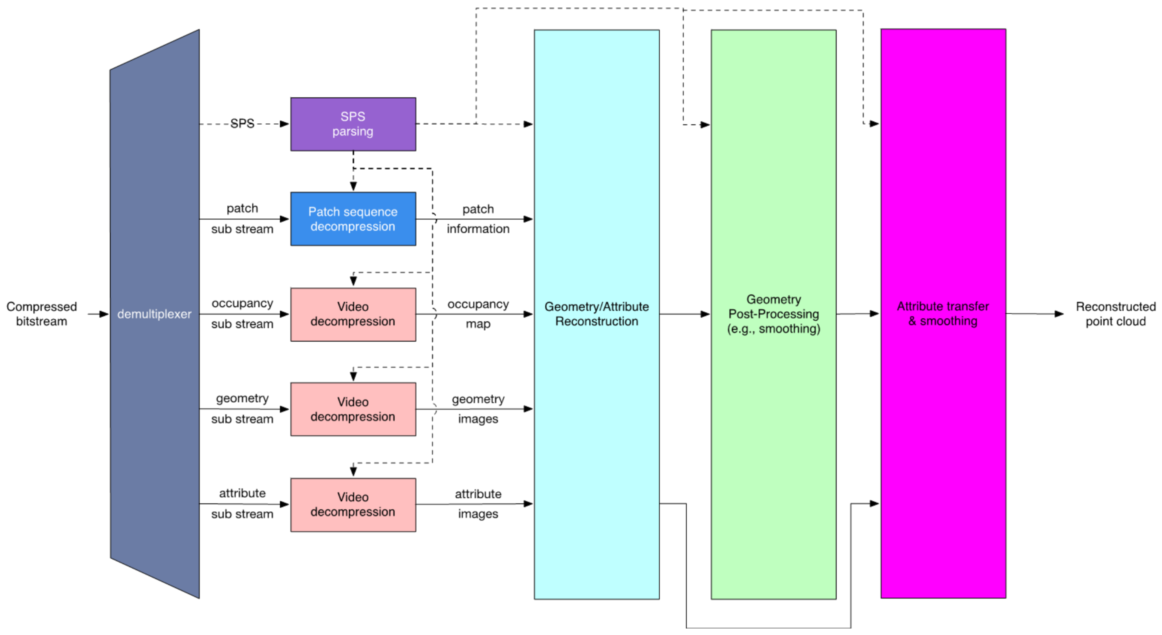


Figure 6: VPCC Encoding Structure

6.1 Representing point clouds with V-PCC

Each V-PCC frame represents a dataset of points inside a 3D volumetric space, with its unique coordinates and attributes. The reconstruction process starts with the atlas information, and then adding the occupancy map, geometry, and attributes information to reconstruct the point cloud.

6.1.1 Patch description

In V-PCC notation, the patch is a collection of information which represents a 3D bounding box of the content, and its associated geometry and attribute information, along with the atlas information required for the reconstruction of the 3D point positions and corresponding attributes from the 2D projections. Axis orientation depends on the projection plane index, and projection mode, and any side of the bounding box, as well as additional 45 degree diagonal projections can be a projection plane. The patch bounding box origin is the point cloud vertex which is the nearest to the point cloud coordinates origin point. The projection image is divided into tile groups, having its origin in the nearest point to the origin of the patch tile group.

6.1.2 Patch segmentation

The point cloud frame is decomposed by converting 3D samples to 2D samples on a given projection plane, using the strategy which would lead to the best compression. In TMC2v0, the patch generations process goal, is decomposing the point cloud into a minimum number of patches with smooth boundaries, with the additional objective of minimizing the reconstruction error.

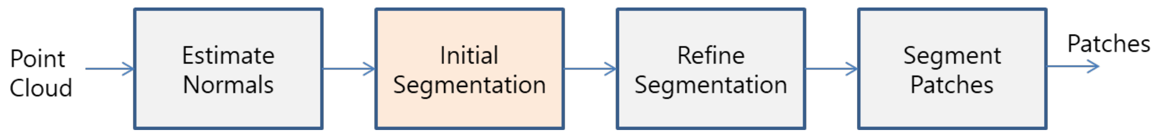


Figure 7: Segmentation process

Initially, the normals are defined for each point of the point cloud. The tangent plane and its corresponding normal are defined for each point, base upon its nearest neighbours m, withing a search distance previously determined. A K-D tree is used for separating the data and discovering neighbours in the proximities of a point p_i and a barycenter $c = \overline{p}$ of that point set is used for defining the normal. Then the baricenter is computed using the following equation:

$$c = \overline{p} = \frac{1}{m}\sum_{i=1}^{m} p_i \tag{2}$$

The estimation of the normal from the eigen-decomposition is defined as:

$$\sum_{i=1}^{m}(p_i - \overline{p})(p_i - \overline{p})^T \tag{3}$$

Bearing this, each point is associated with a corresponding plane from a point cloud bounding box, each defined by a corresponding normal $\overrightarrow{n}_{pidx}$ with values:

- (1.0 ,0.0, 0.0)
- (0.0 ,1.0, 0.0)
- (0.0 ,0.0, 1.0)
- (-1.0 ,0.0, 0.0)
- (0.0 ,-1.0, 0.0)
- (0.0 ,0.0, -1.0)

The plane with the closest normal to the point, is the one associated with it. This can be done by choosing the normal with the max dot product between them:

$$max(\overrightarrow{n}_{pi} \cdot \overrightarrow{n}_{pidx}) \tag{4}$$

The normal sign is defined depending on the point's position, relating to the center. Figure 8 illustrates the projection estimation.

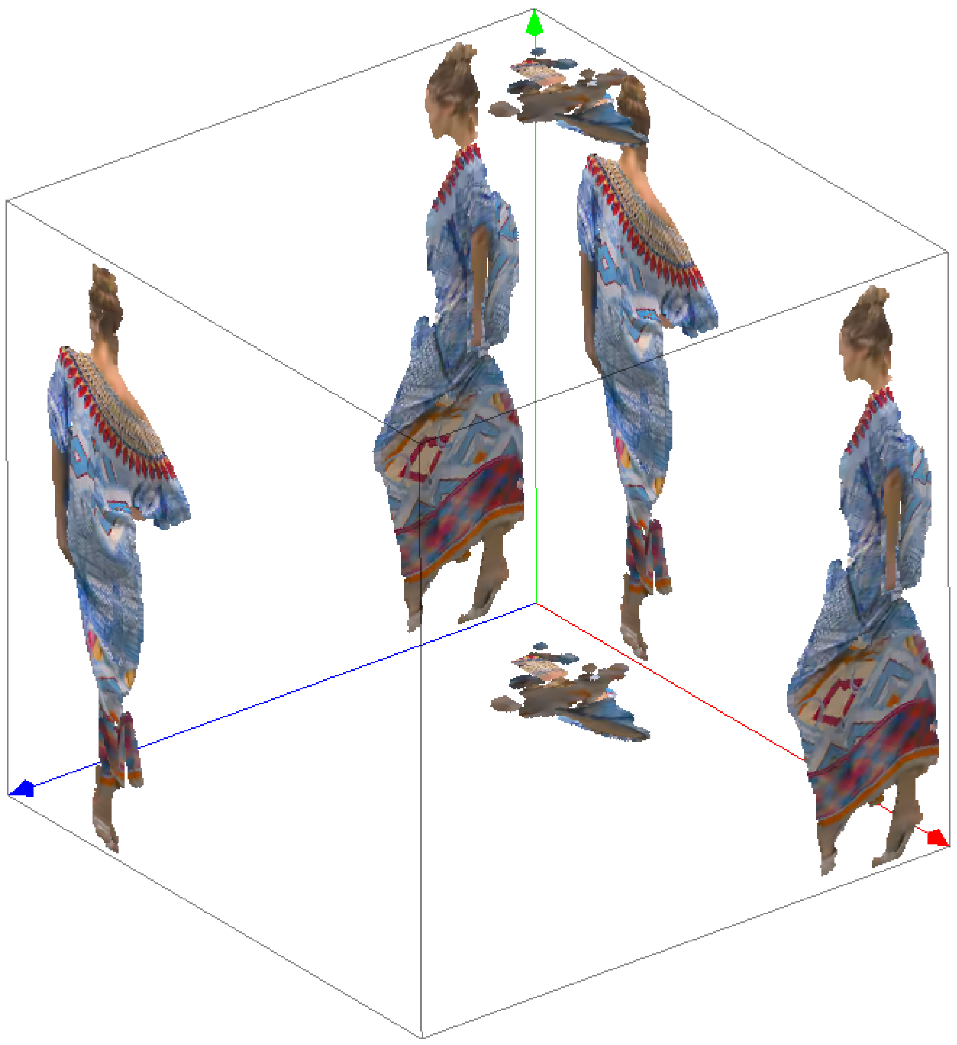


Figure 8: Projection into bounding box

The initial clustering is refined by updating the clustered index associated with each point, based in its normal and cluster indices of its nearest neighbours. Afterwards, the points are clustered based on how close they are to the normals, and the distance between them in the Euclidian space. Final patches are then created from those clusters by grouping similar clusters, and by adding the weight to each plane, the patches are refinements when the Initial Segmentation process decides the projection plane, as it increases the size of the patch in the front or back. Those weights are calculated in the first frame. Segmentation refinement process can be simplified, if a grid based constraint to the neighbouring points search is added, allowing the reduction of complexity and memory bandwidth. It can be implemented in the following way:

1. The (x, y, z) coordinates are partitioned into voxels e.g if a 10-bit point cloud uses a

voxel size of 8, the total voxel numbers along each coordinate would be $\frac{1024}{8} = 128$, resulting in a total number of voxels in the coordinate space to be 128^3

2. Find the filled voxels belonging to the grid, which have at least one point inside them.

3. Calculate a *voxScoreSmoothing* for each filled voxel that is related to each projection plane. This can be done by counting the number of points inside the voxel that are clustered to that projection plane.

4. Find the nearest-neighbouring filled voxels of each filled voxel, recurring to KD-Tree partitioning (*nnFilledVoxels*)

5. A *scoreSmooth* score is computed for each filled voxel, using the following equation:

$$scoreSmooth[v][p] = \sum_{j=1}^{size(nnFilledVoxels[v]} voxScoreSmooth[v][p] \tag{5}$$

   where p is the index of the projection plane and v is the voxel containing the i-th point, making the *scoreSmooth* score the same for all the points inside a voxel.

6. A *scoreNormal* score is computed for each point that is related to each projected plane

$$scoreNormal[i][p] = normal[i] * orientation[p] \tag{6}$$

   where i is the normal vector of the i-th point, and p is the vector of the $p-i$th projection plane.

7. The final score is then calculated for each point related to each projection plane, following the next equation:

$$score[i][p] = scoreNormal[i][p] + \frac{\lambda}{size(nnFilledVoxels[v])} \times scoreSmooth[v][p] \tag{7}$$

   where v is the i-th filled voxel

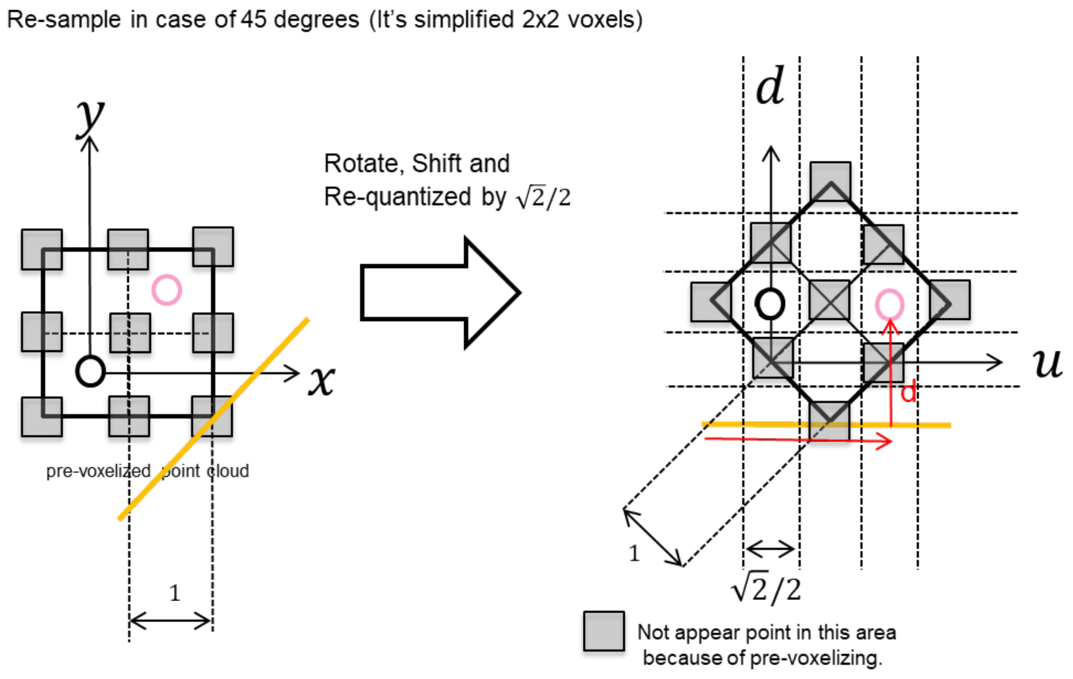
8. Each point is clustered to the projection with the highest final score. The process is repeated for a few iterations.

Other projection planes can be introduced if desired, for visual and quality improvement of the coded point clouds. If this is the case, each plane's corresponding normal is defined as follows:

- (1.0, 0.0, 0.0)

- (0.0, 1.0, 0.0)
- (0.0, 0.0, 1.0)
- (-1.0 ,0.0, 0.0)
- (0.0 ,-1.0, 0.0)
- (0.0 ,0.0, -1.0)
- $(\frac{\sqrt{2}}{2}, 0.0, \frac{\sqrt{2}}{2})$
- $(\frac{-\sqrt{2}}{2}, 0.0, \frac{\sqrt{2}}{2})$
- $(\frac{\sqrt{2}}{2}, 0.0, -\frac{\sqrt{2}}{2})$
- $(-\frac{\sqrt{2}}{2}, 0.0, -\frac{\sqrt{2}}{2})$

While transforming the coordinates, the point is rotated, shifted and quantized. u, v and d are calculated on a patch, and the yellow line represents an additional plane, as show in the next figure:



Re-sample in case of 45 degrees (It's simplified 2x2 voxels)
y
x
pre-voxelized point cloud
1
Rotate, Shift and
Re-quantized by $\sqrt{2}/2$
d
u
d
1
$\sqrt{2}/2$
Not appear point in this area
because of pre-voxelizing.

Figure 9: 3D-2D conversion of a 45 degree projection plane

In the decoding process, the inverse transform of the coordinates is applied to the patch belonging to one of the additional projection planes, being lossless due to the previous voxelization of the point cloud for TMC2.

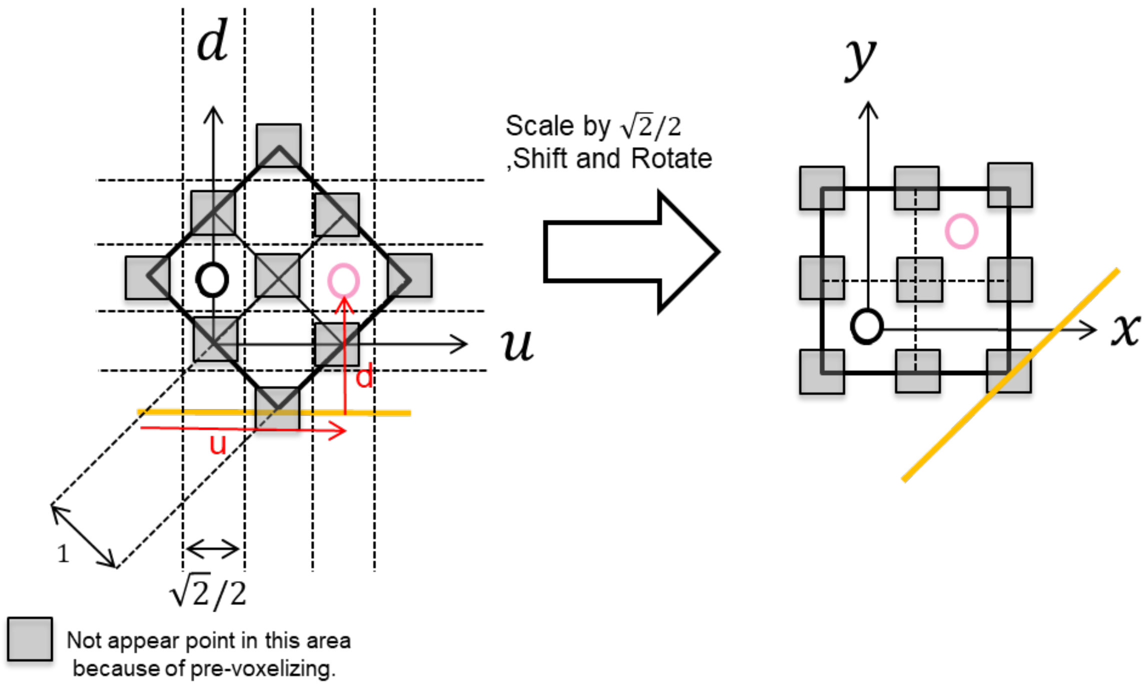


Figure 10: 2D-3D conversion of a 45 degree projection plane

In a low complexity environment, firstly the *scoreSmooth* values are computed for all points without any explicit iterations over the projection planes, and apparent comparisons between the partition of each points, and those of its neighbours. The partition values are instead used to address the *scoreSmooth* value of a place, since those indices are integers between 0 and the number of planes minus one. Afterwards, *scoreSmoothVec* is used to compute the overall score for all points. The following equation describes the number of iterations for the used approach:

$$Number\_of\_iterations = num\_Iters * num\_Points - (num\_Planes + num\_Neighbours) \quad (8)$$

The color value of the points in the vicinity of the patch boundaries in the point cloud, are smoothed before mapping the point cloud onto 2D attribute video frames.

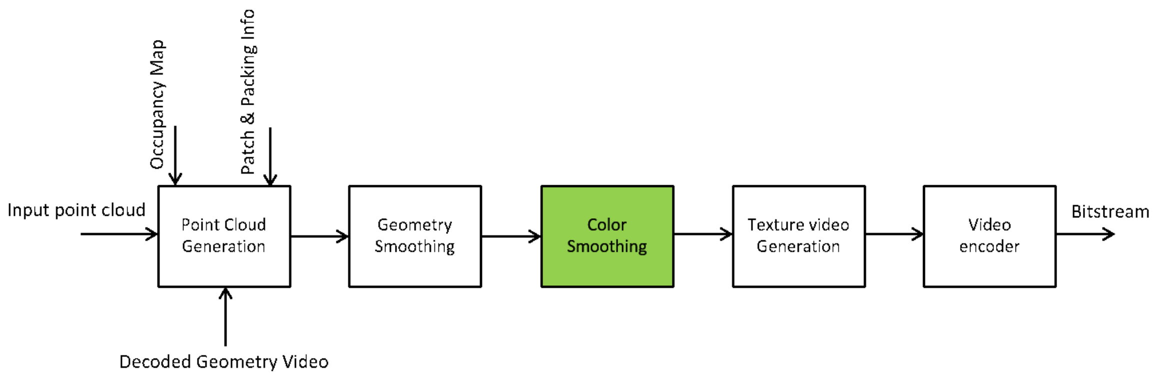


Figure 11: Encoder with color smoothing block

A low complexity method based on 3D super cells can be used as an alternative to KD-Tree for colour smoothing, where all points are grouped into 3D cells, and the color centroid for each cell is computed, which can be done through the following steps:

1. Splitting the decoded points into a 3D grid
2. Computing the color centroids in each cell
3. Identifying the cell containing the query point, for each boundary point.
4. Depending on the query point position inside the cell, select seven neighbouring cells.
5. The cell will be excluded from smoothing, for large color variations.
6. The neighbouring cell will also be excluded if the difference between the color centroid of the current cell and that of the neighbouring cell is greater than a predefined threshold.
7. A tri-linear filter is applied to the current cell centroid, and the remaining neighbouring cells.
8. If the difference between the color of the query point and the smoothed color surpasses a threshold, the query point color will be replaced by the smooth color.

Additionally, if the variation in luma values within a cell surpasses a threshold, there will not be any color smoothing in the cell. Given that the variation cell has been previously calculated, there is no need to calculate local entropy. By removing local entropy, the threshold value of luma variation in cells will be adjusted to compensate the local entropy removal. In the case lossy additional points of the patch being stored in the same geometry video frame with the regular patches, the geometry video frame will be encoded as 10-bits, even though the maximum depth value of the regular patches is 255, due to the geometry coordinate values of the points in the additional points patch being 10 bits, and in order to preserve the fidelity of the original 8-bit values in regular patches due to quantization losses, the depth values in them will be left-shifted by 2, before being stored in the geometry video frame. In the decoder section, the retrieved values from regular patches in the geometry frames shall be scaled back by right-shifting by 2, prior to the addition of the points to the point cloud. When lossy additional points in the patch is stored in another video frame, the regular patches in the geometry frames are encoded in 8-bits, while the separate video frame is encoded in 10-bits. Due to this difference, the decoder needs to be aware of the appropriate conditions on how the values from regular patches should be retrieved and processed in the presence of lossy additional points. They can be signaled to the decoder, or inferred at the decoder from other associated signals

6.1.3 Patch orientation in 3D space

The patches may be oriented by computing the rotated points on local coordinate system planes. After decoding the video planes for each patch, a set of 3D points is determined, being then transformed by a local scale, rotation and translation, resulting in a set of 3D points in the global coordinate system. After all patches are decoded and created, it is possible to apply a global 3D transform, usefull for object which translate and/or rotate in the scene. The two transform's effects are accumulated: the first one at patch level, and the second one at object level. The effect of the first transformation is described as: if a point P_{patch} belongs to the i-th (Ph_i) patch, its coordinates are obtained by decoding the video planes, and then the result is multiplied with the 3D transform associated with (Ph_i).

$$P_{patch} = T_i * P_{decoded} \tag{9}$$

The effect of the second transformation is described as:

$$P_{final} = GT * P_{patch} \tag{10}$$

with GT being the global transform. The cumulative effect is described as:: $P_{final} = (GT * T_i) * P_{decoded}$ Using the following patch information, the center of rotation of the patches in 3d space will be computed as the center of each patch's bounding box:

- 3D coordinates of the patch bounding box
- 3D size of the bounding box

The patch rotation in 3D space can be defined by using quaternions:

$$q = \begin{bmatrix} q \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} w \\ v \end{bmatrix} \tag{11}$$

on axis angles, the expression takes the next form:

$$q = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2) * r \end{bmatrix} \tag{12}$$

The quartenions have the following characteristics:

- Unit quartenion:

$$|q| = 1 \tag{13}$$

$$x^2 + y^2 + z^2 + w^2 = 1 \tag{14}$$

- Multiplication:

$$\begin{bmatrix} w_1 \\ v_1 \end{bmatrix} \begin{bmatrix} w_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} w_1 w_2 - v_1 \cdot v_2 \\ w_1 v_2 + w_2 v_1 + v_1 \cdot v_2 \end{bmatrix} \tag{15}$$

$$q_1 q_2 \neq q_2 q_1 \tag{16}$$

$$q_1(q_2 q_3) = (q_1 q_2) q_3 \tag{17}$$

- Rotation

$$q_p = \begin{bmatrix} 0 \\ p \end{bmatrix} q = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2) * r \end{bmatrix} \tag{18}$$

if q is a unit quartenion and then qq_pq^{-1} results in p rotating about r by θ

$$qq_pq^{-1} = \begin{bmatrix} w \\ v \end{bmatrix} \begin{bmatrix} 0 \\ p \end{bmatrix} \begin{bmatrix} w \\ -v \end{bmatrix} =$$

$$\begin{bmatrix} w \\ v \end{bmatrix} \begin{bmatrix} p \cdot v \\ wp - pv \end{bmatrix} = \begin{bmatrix} wp \cdot v - v \cdot wp + v \cdot p \times v = 0 \\ w(wp - pv) + (p \cdot v)v \times v(wp - pv) \end{bmatrix} \tag{19}$$

In the case of q_1 and q_2 being unit quartenion, the combined rotation starting by q_1 an then by q_2, can be described as:

$$q_3 = q_2 \cdot q_1 \tag{20}$$

- Rotation matrix

$$q = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} \tag{21}$$

$$R(q) = \begin{bmatrix} 1-2y^2-2z^2 & 2xy+2wz & 2xz-2wy & 0 \\ 2xz+2wy & 1-2x^2-2z^2 & 2yz+2wx & 0 \\ 2xz+2wy & 2yz-2wx & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{22}$$

6.1.4 Patch Packing

Patch packing strategy

This process attempts to create the geometry and texture maps, by considering the previously generated patches, and trying to place the geometry and texture data corresponding to each point on a WxH 2D grid. This placement accounts for a user-defined minimum TxT size block, specifying the minimum distance between different patches placed on the 2D grid. The value T is encoded and sent to the decoder. The packing method in TMC 2 uses the following algorithm.

1. The patches are placed on a 2D grid, in a way which guarantees a non-overlapping insertion. Any samples belonging to a patch, are considered occupied blocks.
2. The patches are processed in an orderly manner, based on the patch index list. Each patch is iteratively placed on the grid, and the grid's resolution depends on the original point cloud. The width and height (of the grid) are transmitted to the decoder.
3. If no empty space is available for the next patch, the grid's height value is doubled, and the patch is evaluated again. If the insertion is successful, the height value is trimmed to the minimum value, although it can never be lower than the originally specified in the encoder.
4. The final values for W and H correspond to the frame resolution used to encode texture and geometry video signals, with the appropriate video codec.

The performance when reorganizing the patches for chosen precedence shares similarities with the original method, with the difference of the the decoder not having to reorder the patches, while it also allows the decoding of larger patches instantly.

Patch flexible orientation

To achieve a compact patch packing on 2D surface, flexible patch orientation is needed, allowing for 8 different orientation models: patch rotation by 0, 90, 180, 270 and the respective mirror image. The orientation of the patch is necessary for the decoder to read out the values in the correct order.

Decoupled geometry/attribute packing

A method for signalling the decouple packing strategy can be used if necessary, with the extra metadata used for independent packing. PSF (Packing Strategy Flag) divides the 8-bit codeword into two 4-bit codeword. The 4 most significant bits are related to geometry, while the 4 least significant bits are related to texture. Should the packing strategy for texture be different from the packing strategy for geometry, the extra metadata needs to be sent for each patch, indicating the position of the patch in the texture canvas.

Push-Pull background filling

The push-pull algorithm, which is suited for mip-map texture in conventional graphics, creates a multi resolution representation of the canvas images, filling in the background pixels with lower level pixels

Smoothed PushPull

Based on the method above, it operates as follows: at the i-th LoD, its inter-patch area is additionally smoothed out, reducing the entropy between adjacent pixels, thus resulting in better padding in images of lower detail levels.

Sparse linear model based padding

This method bases itself on the sparse linear model, aiding the creation of a smoother transition between patches, with the following steps:

- Let $C(i,j)$ be the attribute in pixel $P(i,j)$, with the full resolution occupancy $O(i,j)$, which is equal to 1 if the pixel is inside a point cloud patch
- Let $E = \left\{(i_1, j_1), \ldots, (i_k, j_k)\right\}$ be a set of empty pixels, F, a set of full pixels and $S = (F \cup E)$
- $N(i,j)$ the neighbouring pixels of pixel $P(i,j)$

- The padding is formed as a minimisation problem, trying to find the values of empty pixels of E, so that the final padded image is as smooth as possible. Mathematically speaking, it tries to fin the colours $C(i,j)\ldots,C(i_k,j_k)$ such a the following function Θ is minimized:

$$\Theta(C(i_1,j_1),\ldots,C(i_k,j_k)) \quad = \quad \sum_{(i,j)\in S}\left\| \left|N(i,j\right| C(i,j) - \sum_{(p,q)\in N(i,j)} C(p,q)\right\|^2 \qquad (23)$$

where $|N(i,j)|$ are the available neighbours of $P(i,j)$. For interior pixels, $|N(i,j)| = 4$, as for exteriors pixels, $|N(i,j)| < 4$.

Harmonic Background Filling

This method uses the multi-resolution representation combined with the sparse linear optimization. The system is defined by using a linear model, and its solution, optimized on a multi-resolution framework. It is defined as a 7 point Laplacian with Neumann constraints, but solved as a linear system, recurring to Gauss-Seidel relaxation, while initializing the solution from lower dimension representation. The occupancy map determines which positions need to be updated, while also guiding the multi-resolution creation. If a dyadic subsampling of the higher resolution image occurs, the occupied valued is transferred to the lower resolution, if any on its corresponding positions on high resolution are occupied.

Patch Expansion for improving visual quality

As the points for a current patch are connected, the cluster index of each point is verified in order to decide if a point belongs to the patch. If a point is associated with a cluster index which differs from the current patch, the point is considered as belonging to another patch, usually being boundary points which could have been missed after patch generation. To reduce the missed points, the boundary points of a patch are forced to be added in this patch. As the points from a patch are being connected, if a point belonging to a different cluster index is found, it is decided at the patch boundary, and then added in this patch, making so that the patches are expanded around their boundary. In addition, points are added in different patches, thus reducing the possibility of data loss.

Temporary Consistent Patch Order Adjustment

To generate video compression friendly packing result, a consistent, but yet temporary, patch order adjustment is used to reorder the patches in the frame created after the projection stage. In a Group of Frames (GOF), all the patches are sorted in descending order, based on their size, for the first frame, beginning by comparing vertical size, and if they are the same, then the horizontal size is considered. If both have the same value, the patches are sorted by index. for the remaining frames, the temporary consistent patch order adjustment method is used. It initializes the patch list much like the one used in the first frame, but then, based on the patch order in the previous frame, searches matched patches in the current frame. Each patch is iterated in the previous frame. Let $refPatch[i]$ be the i-th patch in the previous frame. For $refPatch[i]$, iterate each patch in the current frame, with $patch[j]$ being the j-th unmatched patch in the current frame, the the coefficient of correlation Q_{ij} between $refPatch[i]$ and $patch[j]$ is computed. To speed the search matched procedure, the patch in 3D space will be projected in 2D space, and the intersection over union between two patches can be regarded as Q_{ij}:

$$Q_{ij} = \frac{preRect[i] \cap Rect[j]}{preRect[i] \cup Rect[j]} \tag{24}$$

where $preRect[i]$ is the bounding box of the region in which $refPatch[i]$ is projected in 2D space. $Rect[j]$ is the bounding box of the region in which $patch[j]$ is projected in 2D space.

$$Q_{ik} = max\left\{Q_{ij}, j = 0, \ldots, N\right\} \tag{25}$$

Q_{ik} is defined as the maximum intersection over union between $refPatch[i]$ and $patch[j]$. k is the index of a maximum IOU (Intersection over Union). If Q_{ik} is larger than a threshold, $patch[k]$ is regarded as the best matched patch in the current frame, setting the patch in the $bestMatchIndex = i$, and storing it in the $reorderPatchBuffer$. Otherwise, it can be considered that no matched patch was found in the current frame for $refPatch[i]$. After all patches are iterated, $reorderPatchBuffer$ will store all matched patches, and follow its order in the previous frame, and the unmatched patches are stored at the end of the buffer

Global Patch Allocation

This method is used to improve the temporal consistency packing within a GOF. By placing each global matched patch in the same location in a corresponding map frame. The global matched patches are obtained using IoU matching strategy, resulting in all the global matched patches having the same location in the occupancy map, in a GOF. For this to be possible, the method starts by computing the union patch of global matched patches in each frame. Let *GlobalPathCount* be the global matched patches in a GOF, and all global matched patches are allocated at the beginning of the patch buffers in every frame. *unionPatch* buffer saves the union of the global matched patch occupancy map in each frame. It then uses a test model packing method to pack all the patches in *unionPatch* to create a global occupancy map. $u0$ and $v0$ can be obtained for each patch in *unionPatch*, and the global occupancy map will restrain the location of all global matched patches in each frame. Then, by using the occupancy map to allocate all global matched patches first, the non-global matched patches are allocated in each frame. If the object in the sequence has slow motion, the method processes a group of frames, otherwise, the *unionPatch* in global occupancy would be much larger than the real occupancy map of the global matched patches in each frame. By dividing the GOF in several subcontexts, a more compact packing result is obtained for each subcontext.

Global Tetris Packing

The method identifies temporal correspondence between patches, creating a *patchVolume*, which is generated by the superposition of aligned occupacy maps, and the size and length of those volumes are considered during the patch placement, maintaining the patch order of matched frames. The algorithm is described as:

1. Perform regular patching
2. Create *patchVolume* with the matched patches
3. Compute the accumulated occupancy map, and the weight of each patch
   (a) From back to front, the last frame is initialized with weight 1 if a backward matched patch is present. If not, it is initialized with 0.
   (b) For the rest, the same initialization is performed, adding $+1$ to the weight of the forward matched patch.
4. For the first frame, the list is sorted according to the weights and sizes.
5. For subsequent frames, place the matched patches using the same order as the previous frame, and then the patches are sorted according to their weights and sizes.

$$w = \{^{1, if backwardmatched}_{0} \tag{26}$$

$$weight(N) = \{^{w+1+eight(N+1), ifforwardmatched}_{w} \tag{27}$$

6.2 Image Generation

This process makes use of the 3D to 2D mapping computed during the packing process, in order to store the geometry and texture of point cloud as images. For better handling multiple points projected onto the same plane, the patches are projected into two images layers, with $H(u, v)$ being the set of points of the current path, which is projected to the same sample (u, v). The near layer stores the point with the lowest depth $D0$, belonging to $H(u, v)$, whereas the far layer, captures the point with the highest depth within $[D0, D0+\Delta]$ belonging to $H(,u,v)$, where Δ is a user-defined parameter describing surface thickness. A surface separation method is necessary, in order to prevent the junction of different surfaces when there is a stack of various different surfaces in the connected component. One solution is the separation of surfaces using the difference of MSE values of points in the RGB domain. Let R_1, G_1, B_1 be attribute values belonging to T0, R_2, G_2, B_2 be attribute values belonging to T1, and $Threshold$ having the value of 20. The patch is separated if:

$$MSE(R_1 - R_2, G_1 - G_2, B_1 - B_2) > Threshold \tag{28}$$

6.2.1 Geometry image generation

Represented by a frame of WxH in the YUV420-8bit format, it can be coded in the following methods:

Differential coding method

The geometry reconstruction process exploits the occupancy map to detect non-empty pixels in the geometry/texture and images/layers. The 3D positions of points of those pixels are calculated by levering the auxiliary patch information, and geometry images. Let P be the point associated with pixel (u, v), $\delta 0, s0, r0)$ be the 3D location of the patch containing that pixel,$g(u, v)$ the luma component of the geometry image, and $(u0, v0, u1, v1)$ its 2D bounding box. P can be expressed in terms of:

- Depth: $\delta(u, v) = \delta 0 + g(u, v)$
- Tangential shift: $s(u, v) = s0 - u0 + u$
- Bi-tangential shift: $r(u, v) = r0 - v0 + v$

The sample values in a patch are calculated using the following expression:

$$I(u, v) = \{^{d-min\_Depth(d-Quantized\_Min\_Depth<max\_Depth)}_{invalid} \tag{29}$$

In which d is the depth of the point corresponding to the location, *maxDepth* is the maximum allowed depth per sequence and *QuantizedMinDepth* is the minimum depth in the patch, quantized by 2^N as:

$$floor(\frac{minDepth}{2^N}) \times 2^N \tag{30}$$

Given the fact that *minDepth* is quantized by 2^N, it reduces the bits required to signal the *minDepth* to $M - N$, where M indicates the bit depth of the geometry data.

Interleaved absolute depth coding method

Instead of 2 layers encoding, a single layer is encoded, reducing the requirement of the codec. If $D0$ represents the $depth0$ geometry frame associated to the depth values of the near layer, and $D1$ represents the $depth1$ geometry frame associated with the depth values of the far layer, the interleaved frame is formed by taking one pixel from $D0$, then from $D1$, then again from $D0$ and so forth. Only the interleaved frame is available for the reconstruction process, although both the $D0$ and $D1$ frames are needed to reconstruct the original point cloud. As such, the missing depth values are predicted by interpolation of the neighbours If a certain pixel belonging to an interleaved depth frame contains a $D0$ value, but the $D1$ value is missing, the prediction of the missing value is made on the neighbours of the current pixel in the interleaved frame that contains a $D1$ value, and vice-versa. The predicted $D0$ and $D1$ values are denoted as $\widehat{D0}$ and $\widehat{D1}$. If a pixel location (x, y) in the interleaved frame contains the value of $D0$, $\widehat{D1}$ is computed as:

$$\widehat{D1}(x,y) = func(D1(x,y-1), D1(x-1,y), D1(x,y+1), D1(x+1,y)) \tag{31}$$

The opposite is:

$$\widehat{D0}(x,y) = func(D0(x,y-1), D0(x-1,y), D0(x,y+1), D0(x+1,y)) \tag{32}$$

The prediction of missing $D1$ values is accomplished by finding the mean value of $D1$ neighbours, whereas the prediction of $D0$ values is accomplished by finding the minimum values of the $D0$ neighbours. To assure that the differences between $D1$ and $D0$ depth values does not exceed the fixed surface thickness of the point cloud at a certain pixel, additional conditions are used during the prediction process.

- For $\widehat{D1}$, if $\widehat{D1}$ at a pixel (x, y) exceeds the existing $D0$ value of $(D0(x, y)$ plus the surface thickness, $\widehat{D1}(x, y)$ is replaced by $(D0(x, y)$ + Surface Thickness.
- For $\widehat{D0}$, if $\widehat{D0}$ at a pixel (x, y) is lower than the existing $D1$ value of $(D1(x, y)$ minus the surface thickness, $\widehat{D0}(x, y)$ is replaced by $(D1(x, y)$ - Surface Thickness.

The neighbours pixels with occupancy value of 1 and those belonging to the same patch as the point in which the missing value is being predicted are also considered for prediction. It should be taken in consideration that while predicting the missing value, the $D0$ value at a certain pixel cannot exceed the $D1$ value.

3D geometry padding

This technique consists in the selection of a value for positions which generate a reconstructed point which will be as close as possible to the original point cloud, by performing a search in a range of possible depth values.

$$\overline{d} = \frac{1}{\sum_{i=1}^{4} I(i)} \sum_{i=1}^{4} I(i) \times d(i) \tag{33}$$

$$d_{optimal} = min_{d \in [\overline{d} - \Delta_{min} + \Delta_{max}]} dist(X(d), 3Dpointcloud) \tag{34}$$

$$\Delta_{min} = \{^{\overline{d}, \overline{d} < 8}_{8} \tag{35}$$

$$\Delta_{max} = \{^{max - \overline{d}, \overline{d} > max - 8}_{8} \tag{36}$$

Spatially adaptive geometry interpolation

An optimization is performed using a RDO-like approach in the encoder, adding the required signalling in the bitstream so that the decoded is aware of the interpolation coding mode to use, thus adapting the interpolation coding mode to the characteristics of point cloud. The optimization is performed at each occupancy map resolution block, using the reconstructed depth image by:

- Computing the cost of each candidate mode.
- Selecting the interpolation coding mode with the lowest cost.
- Signalling in the bitstream the selected interpolation coding mode.

Pre-patch projection

The optimization process selects if a single projection mode is used for the entire frame, choosing from two different modes. If a single projection mode is chosen, mode 0 is used, otherwise, the projection mode is selected on a per-patch basis.

- **Mode 0**: The minimum depth value stored in depth 0 image, maximum depth value satisfying the surface thickness constraint in depth 1 image
- **Mode 1**: The maximum depth value stored in depth 0 image, minimum depth value satisfying the surface thickness constraint in depth 1 image.

The surface thickness constraint is defined as $|D1 - D0| \leq surfaceThickness$ The frame-level decision is indicated by the frame projection mode parameter, and two modes can be defined:

- **Frame projection mode 0**: Corresponding to fixed projection mode, with all the patches using projection mode 0.
- **Frame projection mode 1**: Corresponding to variable projection mode, with the possibility of each patch using a different projection mode

The frame projection mode can be decided by using the following algorithm:

- Generating the minimum depth images for the frame, projecting to the XY, XZ and YZ planes.
- Computing the ratio for the three depth patches combined, between the number of occupied positions with the same value in $depth0$ and $depth1$, and the number of total occupied positions.
- If the ratio is smaller than *ProjectionModeSelectionThreshold*, Frame projection mode is selected to 0, otherwise its set to 1.

When the frame projection mode is a variable, a per-patch projection mode is sent, indidicating the mode to be used to de-project each patch. The patch projection is decided using the following algorithm:

- Regarding the point cloud
  - Minimum depth images for the frame are generated, projecting to the XY, XZ and YZ planes
  - Maximum depth images are generated for the frame, projecting XY, XZ and YZ planes
- Regarding each connected component
  - Minimum depth patches are generated are generated, *minimumDepthPatch*, by projecting the minimum distance along the normal axis, i, of the connected component
  - The number of positions is in *minimumDepthPatch* is counted (*counterMinimum*, by projecting the same depth as the co-located position in minimumDepthFrame[i]
  - Repeat using maximum distance, to obtain counterMaximum.
  - If $counterMinimum \geq counterMaximum$, set the projection mode for the current patch to 0, otherwise its set to 1

Enhanced delta depth code for lossless coding

This coding method changes the depth image so that multiple points can be coded, by using the following algorithm:

1. For each position between D0 and D1 included, use one bit to indicate whether of not this position is occupied, along one projection line
2. Concatenate all bits used in the previous step to form a codeword, nominated Enhanced-Delta-Pack (EDD)
3. Pack EDD coded of one frame in DepthImg1, to be further coded by video codecs

While creating the patches, a D0 depth patch and an EDD-code patch are made for each connected component. If D1-D0=N, there are (N-1) positions between D0 and D1. Let (u,v) be the column and row index in the image, and ON(u,v) be the occupancy map. For lossless coding, the occupancy precision is 1, making the size of the occupancy map the same as the size of the depth images D0(u,v) and D1(u,v):

- If OM(u,v) is zero, it is not modified.
- If OM(u,v) is 1, and (D1(u,v) - D0(u,v)) is less than or equal to 1, means that there are no depth positions between D0 and D1, and thus OM(u, v) is not modified.
- If OM(u, v) is 1, and (D1(u,v) - D0(u,v)) is greater than 1, OM(u,v) is modified as follows
  - A PCM code is used to specify the occupied depth positions between D0(u,v) and D1(u,v). Since its most likely that the depth position between D0 and D1 to be occupied, the occupancy map is modified as follow:

$$OM^{new}(u,v) = OM^{old}(u,v) + 2^{(N-1)-1-PCMcode)} \tag{37}$$

After that, a missedPointsPatch is created to serve as container for those missing points, and The EDD codes (D1+EDDcode) are stored in DepthImg1. To signal the color of the in-between points in the stream, the position in the packed texture images for keeping the colour of these points is determined during the point cloud geometry reconstruction. The un-occupied blocks are sorted in a list is used to store the color of this new point, and since the decoder has the same information and can repeat the operation, the color in those between points can be retrieved.

Generalized Enhanced Occupancy Map for Depth

The PCM code requires (D1(u,v) - D08(u,v)) to be larger than 1, so that it is able to use EOM mode at location (u,v). So, instead of using fixed layer numbers as a reference for the bounds of PCM code, this method computed the bounds of PCM code procedurally. Let *layer_count* be the number of layers in the stream, and D(I) denote the depth of layer 1 at (u,v). A variable Dmin is set to MAXDEPTH, and Dmax is set to 0. Then, for all layers, if Dmin > D(I), Dmin is set to D(I, and if Dmax < D(I), Dmax is set to D(I). Once Dmin and Dmax are computed, the EOM code is generated, replacing 0 and D1 by Dmin and Dmax, respectively, with its nominal length being:

$$N = Dmax - Dmin \tag{38}$$

Since the boundaries of the EOM code are flexible, a direction flag can be used to signal the order of the occupancy bits (Dmin to Dmax = 0 and Dmax to Dmin = 1). With this, EOM can applied in any of the occupied location of the patch canvas. Additionally, a min bit count and upper bound count are also used to limit the value of N to a certain range, making the final value of N take the following form:

$$N = max(minbitcount, min(maxbitcount, Dmax - Dmin))$$

Given this changes, the occupancy bits of the PCM codeword spans over an already coded point in a depth layer. To avoid this, the occupancy bits of the PCM codeword skip over the layer-coded locations, allowing for the PCM coverage to be extended without increasing the number of bits needed for its coding.

6.2.2 Occupancy map generation

The occupancy map is a binary map which indicates for all grid cells, whether it belongs to the empty space of the point cloud. When a block of the occupancy map is occupied, even if only partially, all the points belonging to that block will be reconstructed in the decoding process (with occupancy precision of 4, 16 points per block are reconstructed.

6.2.3 Occupancy map refinement

This method reduces the occupancy map and patch to index data when few points are present in a block on encoder side, thus avoiding noise during the reconstruction stage.

1. In a first phase:

    - The occupancy map is updated to block (4x4):

   (a) Count the number of point $NP_{smallBlock}$ in the original point cloud by using the full occupancy map precision
   (b) If $NP_{smallBlock} \leq 1$, the occupancy map is unoccupied, setting it to one otherwise

2. In a second phase:

   - Let OM_{reduce} be the updated occupancy map of the original point cloud at full resolution. From OM_{reduce}, update the block to patch index at a block resolution of 16x16, and for each block do as follows:
     (a) count the number of occupied blocks: NB_{block}
     (b) If $NB_{Block} < 4$, the block is marked as unoccupied, otherwise, it is marked as occupied.

6.2.4 Attribute image generation

The texture generations procedures exploits the reconstructed/smoothed geometry to compute the colours associated with the re-sampled points

Interleaved attribute image generation

This method represents attribute video, consisting of 2 layers (e.g C0 and C1) as an interleaved image is preformed as follows. C0 is the colour0 of the attribute frame containing the colour values of the near layer, and C1 represents the colour1 of the attribute frame containing the colour values of the far layer. A single attribute frame based on the interleaving of C0 and C1 is then formed, encoded and transmitted to the decoder. The interleaved attribute frame C2 is created by taking alternate attribute from both C0 and C1 frames across all rows and columns. The prediction of the re-sampled points is done using the interleaved attribute frame, and involves the prediction of missing attribute values belonging to the interleaved frame. For this, The D0 and D1 values are checked for equality, and then the attribute value is stored at that pixel. The prediction of the remaining attributes is done by finding the mean of the neighbours in the same attribute layer. If a point $(x + y)$ value is even, and it belongs to the far layer, the attribute values of the neighbouring far layers are averaged to assign the value to this point. For odd $(x + y)$, to assign the value to the points of the near layer, the values of the neighbour points in the near layer are averaged:

$$\widehat{C1}(x, y) = mean(C1(x, y-1), C1(x-1, y), C1(x, y+1), C1(x+1, y)) \tag{39}$$

$$\widehat{C0}(x, y) = func(C0(x, y-1), C0(x-1, y), C0(x, y+1), C0(x+1, y)) \tag{40}$$

Knowing that the neighbours should belong to the same patch, as the point being processed, if they don't belong to the same patch, they are not considered for the prediction of the missing point

6.2.5 Duplicate points pruning

The created geometric images are de-projected in order to generate the geometry for the decoded point clouds. For each existing point in a patch, two points are created based on the stored coordinates in geometry layer 0 and 1 an in the patch auxiliary information. Due to this, the de-projection process need to be modified in order to only create one point cloud per (u,v) coordinates of the patch, if coordinates stored in the geometry layers are equal. The process has the following steps:

- For each point belonging to a patch, a 3D point is created with the coordinates stored in layer 0
- The coordinate in layer 0 is compared with the coordinate in layer 1
- If they are equal, no additional point is created, otherwise a second 3D point is created, with the coordinates in layer 1.

In the encoder, the creation of texture images T0 and T1, which store the reconstructed points colour, proceeds as:

- If no 3D point is generated from D1, a dummy colour is stored in that position i T1.
- On the contrary case, T1's colour is generated as in the current TMC2

6.2.6 Geometry smoothing

Geometry grid smoothing

The geometry refinement process's goal is to filter the patch boundaries in order to improve the visual quality of the reconstructed point cloud, as the smoothed geometry is used for attribute patch generation. The smoothing process is applied to the patch edges and the decoded points centroid are calculated previously in a small grid. After the derivation of the centroid and the number points in the 2x2x2 grid, a trilinear filter is applied with the centroid. The grid size is 8 due the occupancy precision being set to 4, and in the geometry grid smoothing the bounding box is divided into the grids by grid size, with the number of grids being calculated as follows.

$$numOfGrid = \frac{boundingBox.Xrange}{gridSize} \times \frac{boundingBox.Yrange}{gridSize} \times \frac{boundingBox.Zrange}{gridSize} \quad (41)$$

The test model cashes the needed information for the filter of the entire point cloud frame in advance, but in another implementation, the memory size can be reduced. It is at least needed for the trilienear filter to store the neighbout 2x2x2 grid information, meaning that, regardless of the grid size, the minimum memory size is obtained with numOfGrid=8.

$$minMemorySize = 8 \times (sizeOf(int) \times 4+) = 256Byte \quad (42)$$

Patch border filtering

This aims to deform the 2D contour of the current patch, reducing the distance between this contour, and the ones from the adjacent patches. For all the contour's points, the distance to other patches is calculated, taking the neighbours map in account. Two kinds of deformations can be made for each point of the contour:

- Erosion, removing the current point from the contour
- Dilatation that increases the contour

This can be performed on time for all points in the contour, if some are removed or added, they need to be processed again. The final stage is the filtering process is the filling process, guaranteeing that there are no holes in between two patches in the 3D domain, and can be described in two steps:

1. The patch's contour points are locally changed to smooth the transition between two patches
2. 3D points are added between two patches in order to fill the hole which could appear in the depth direction

Colour Smoothing

The patches are packed in 2D video frames for encoding and transmission to the receiver. As non-neighbouring patches in 3D space are usually packed next to each other in 2D videos, the pixel value from non-neighbouring patches an be mixed up by the block based video codec. Some artifacts may appear at path boundaries in the reconstructed point cloud, mainly at lower bit rats. The neat patch boundaries are smoothed as follows:

- Identification of points near patch boundaries in the reconstructed point cloud
- Smooth out the colour of the points in a small neighbourhood

The points near the boundary can be identified by the occupancy map, available to the decoder. Single-pass boundary points identification is implemented with the iteration over all points, to identify the boundary layers. For all query points, two groups of neighbouring points are defined:

- The first included the intermediate neighbouring points
- The second includes the points-2-pixel apart from the query point.

For a query, if any neighbouring pixel is empty, the query point is identified as a boundary point, and if the query point is located at an edge of the image or is 1-pixel apart from any edge, that point will be identified as a boundary point as well. For the boundary points detection for points near the occupancy map edges, a point is identified as a boundary point, if one of its neighbours is empty. In the second pass, all the neighbouring points of the first pass boundary points are identified as boundary points as well.

6.2.7 Image padding

This process fills the empty space between patches, thus generating a piece wise smooth image suited for video compression, using the following method:

- Each block of TxT pixels is processed independently
- If empty, the pixels of the block are filled by copying the last row or column of the previous TxT block in raster order
- If the block is full, nothing is done
- If empty and filled pixels are present, the empty pixels are iteratively filled with the average values of their non-empty neighbours.

6.2.8 Texture Padding Improvement

This method is built on a fact that far T0 and near T1 projection layers have the same occupancy map, causing the dilated region in T0 and T1 to have the same shape.

$$T2_{ij} = \frac{T0_{ij} + T1_{ij}}{2} \tag{43}$$

The first dilation of T0 and the second for T1 are replaced with T2:

$$T0_{ij} := T1_{ij} := T2_{ij} \tag{44}$$

6.3 Video Compression

The images/layers which where generated are stored as video frames and compressed using the HM video codec using the HM configurations provided as parameters.

6.3.1 Lossless video compression for Geometry and texture

The input cloud, which miss projection onto the 2D frames during the 3D-2D transformation in a separate patch and stores the 10 bit X, Y, Z geometry values in three colour planes of 10 bit 4x4x4 format video frames. This group of points is called missed-points-match. An occupancy map corresponding to that group is generated and the patch information is set to the decoder using the existing mechanisms of TMC2.HM-16.18+SCM-8.7 are used for lossless coding. Due to HM s/w in 4x4x4 giving priority to encode the first component, data is converted to GBR format for the lossless encoding mode to achieve a performance boost. It also provides some limited functionality to perform colour conversions at the input and output of the HM encoder and decoder, while also including support to convert (R, G, B) samples to (G, R, B) in the encoder, and (G, B, R) to (R, G, B) in the decoder .

6.3.2 Lossy video compression for geometry and texture

HM-16.18+SCM-8.7 video codec is used for lossy compression for texture and geometry video and for occupancy map lossy coding.

Motion vector prediction improvement for point cloud coding

For a block in the current frame, the corresponding 3-D position is found using the patch and depth information. Then, a motion estimation is performed in the nearest 3-D position in the reference frame, which is then used as a motion vector block of the attributes block. The vector is then inserted in the first position of the advanced motion vector prediction list, with the duplication detection is performed between the first original MVP and derived MVP. After the determination of the corresponding patch in the reference frame, the MV (motion vector), can be firstly determined by the 3D difference between the current patch and the reference patch, given that the 3D positions of the start pixel of the patch are different.

$$MV_{X_{3D}} = patch3DShiftu_{cur} - patch3DShiftu_{ref}$$

$$MV_{Y_{3D}} = patch3DShiftv_{cur} - patch3DShiftv_{ref}$$

Secondly, the 2d difference between the 3D positions of the current patch and the reference patch is also used, considering the occupancy block resolution.

$$MV_{X_{2D}} = patch3DShiftu_{ref} - patch3DShiftu_{cur} \times obr$$

$$MV_{Y_{2D}} = patch3DShiftv_{ref} - patch3DShiftv_{cur} \times obr$$

The final motion vector is a combination of these two parts

$$MV_X = MV_{X_{3D}} + MV_{X_{2D}}$$

$$MV_Y = MV_{Y_{3D}} + MV_{Y_{2D}}$$

After the addition of the derived motion vector as the candidate, the encoder chooses from the motion vector predictor, the zero motion vector. the proposed 3D motion vector, and the motion vector from the partition 2Nx2N by using rate distortion optimization in order to determine the center of te search range, if the 3D motion vector is within the smallest range distortion cost and is also chosen as the start point, it is more likely that the encoder finds the corresponding block in the reference frame, providing a significant performance improvement.

6.4 Atlas (Auxiliary patch information) Compression

For each patch, the following metadata is encoded/decoded:

- The index of the projection plane
  - Index 0: plane (1.0, 0.0, 0.0)
  - Index 1: plane (0.0, 1.0, 0.0)
  - Index 2: plane (0.0, 0.0, 1.0)
  - Index 3: plane (-1.0, 0.0, 0.0)
  - Index 4: plane (0.0, -1.0, 0.0)
  - Index 5: plane (0.0, 0.0, -1.0)
- 2D bounding box (u0, v0, u1, v1)
- 3D location (x0, y0, z0) of the patch, with its representation in thermos of depth $\delta 0$, the tangential shift s0 and bi-tangential shift r0. The 3D location $(\delta 0), s0, r0)$ is computed the following way:
  - Index 0: (x0, z0, y0)
  - Index 1: (y0, z0, x0)

– Index 2: (z0, x0, y0)

– Index 3: (x0, z0, y0)

– Index 4: (y0, z0, x0)

– Index 5: (z0, x0, y0)

Additionally, a list defining the normal axis for the 45-degree projection planes is added:

– Index 6: $(\frac{\sqrt{2}}{2}), (0.0), (\frac{\sqrt{2}}{2})$

– Index 7: $(\frac{-\sqrt{2}}{2}), (0.0), (\frac{\sqrt{2}}{2})$

– Index 8: $(\frac{\sqrt{2}}{2}), (0.0), (\frac{-\sqrt{2}}{2})$

– Index 9: $(\frac{-\sqrt{2}}{2}), (0.0), (\frac{-\sqrt{2}}{2})$

Mapping information for each TxT block and its associated patch index, is encoded in the following way:

- Let L be the ordered list of the patch index such that their 3D bounding box contains the TxT block associated to that index, and with the same order as the one used to encoded the 3D bounding boxes.
- The empty space between pacthes is considered a patch, being assigned the special index 0 and added the list L.
- Let I be the index of the patch to which belongs the current TxT block and J be the position of I in L. J is arithmetically encoded, leading to better compression efficiency.

The temporally consistent packing method processes all the matched patches of the current frame by populating the occupancy map, and then all the unmatched patches are positioned in an unoccupied space of that same occupancy map, thus increasing the possibility to pack the patches in a similar order from the previous frame. The correlation between consecutive frames of the patches can be used to reduce the entropy of the auxiliary patch information, using the patch in a previous frame as reference, and the following information may be encoded using differential method. To signal the matched patch index, the index of the patch in the frame is used as the predictor, and defined as 0 for the zero-th patch, and as *predicted index = previous predicted index + 1 + previous delta index.* The following method can be used

```
\\ For the first Y matched patches
 for(i=0; i<Y; i++)
 {
encode the differential value of
idx ,u0,v0,v1,u1,d1,sizeU0,sizeV0 with Golomb–Rice coding
 }

\\ For the remaining X–Y matched patches

    if (F==0)
    {
    encode F with 1 bits
     for(i=0, i<X–Y; i++)
        {
        encode u0,v0,u1,v1 and d1 with max_N bits
        encode differential value of sizeU0 and sizeV0 in
        consecutive patches with Golomb–Rice coding
        }
    }else{
    encode F with 1 bits ,
    encode A with 5 bits ,
    encode max_N if the bits in A = 1
    encode u0,v0,u1,v1 and d1 with max_N bits
    encode differential value of sizeU0 and
    sizeV0 in consecutive patches with Golomb–Rice coding
    }
```

6.5 Recolouring

Give the input cloud's positions/attributes and the reconstructed positions $(\tilde{X}_i)_{i=0,\dots,N_{rec}-1}$, the aim of the attributes transfer procedure is the determination of the values which minimize the attribute distortions.

6.5.1 Direct Colour Transfer

- Let $(X_i)_{i=0\dots N-1}$ and $(\tilde{X}_i)_{i=0,\dots,N_{rec}-1}$ be the input and reconstructed positions.
- Let N and N_{rec} be the number of points in the original point cloud and the reconstructed point cloud

- For each $\tilde{X}_i$ in the reconstructed point cloud, let X_i^* be the nearest neighbour in the original cloud, and a_i^* the attribute associated with X_i^*.
- For each $\tilde{X}_i$, let $\mathbb{Q}^+(i) = (X_i^+(h))_{h \in (1,\ldots,H(i))}$ be the set of points in original cloud sharing $\tilde{X}_i$ as the nearest neighbour in the reconstructed cloud, with H(i) being the number of elements in $\mathbb{Q}^+(i)$ and $X_i^+(h)$ one of the elements of $\mathbb{Q}^+(i)$
- If $\mathbb{Q}^+(i)$ is empty, the value of a_i^* attribute will be associated with $\tilde{X}_i$.
- Otherwise, the attribute $\tilde{a_i}$ associated with the point $\tilde{X}_i$ is obtained by:

$$\tilde{a_i} = \frac{1}{H(i)} \sum_{h=1}^{H(i)} a_i^+(h) \tag{45}$$

6.5.2 Distance-weighted colour transfer

For each point for target p_r:

1. Find the nearest neighbours in source cloud to p_r, creating a set of points Ψ_1
2. Find the set of source points that p_t belongs to in their set of nearest neighbours, creating a set of points Ψ_2.
3. The distance weighted average is computed as:

$$\overline{\Psi_k} = \frac{\sum_{q \in \Psi_k} \frac{c(q)}{\Delta(q,p_r)}}{\sum_{q \in \Psi_k} \frac{1}{\Delta(q,p_r)}} \tag{46}$$

   which $\Delta(a,b)$ is the Euclidean distance between a and b, and c(q) is colour of point q.
4. The average of Ψ_1 and Ψ_2 are computed and it is used as centroid colour
5. A backwards search from the centroid point is performed, in order to exclude the points if their absolute differences to the centroids colour is larger than the threshold th_c
6. The weighted average of both sets of points is updated, and transferred top_r

6.6 Patch information coding

6.6.1 Patch type coding methods

The data units could be decoded in several different modes, with the selected mode being depended on the image type to decode, and it is signaled using the 0-th order of the exp-Golomb binarization, according to the following:

For intra frames: I Intra = 0 I PCM = 1 I END = 14

for inter frames: P SKIP = 0 P INTER = 1 P INTRA = 2 P PCM = 3 P END = 14

6.6.2 Patch 2D to 3D coordinates conversion coding

PatchNormalAxis is the projection plane which will be used to generate the patch and can range from 0 to 5, with each patch having a dedicated normal direction, instructing the patches to be placed in a corresponding plane, associated with a viewId value for a given patch. PatchProjectionMode is the mode that defines the direction of the patch normal axis. The depth values should be either added of subtracted, depending on the value of the mode, from the indicated patch origin point.

6.6.3 Point Local Reconstruction Mode

For lossless coding its restricted to point clouds with a maximum size of 1024, due to the coordinated being represented in absolute value, and compressed in 10-bit video. This methods splits the entire $(2^N \times 2^N \times 2^N)$ cube, with N being the bit depth of the geometry, into several $2^M \times 2^M \times 2^M$ where M is the depth of the video. Afterwards, for each sub-cube which contains at least one missing point, create a missing point patch and the geometry coordinates of those points are represented inside the cube, relative to one of the corners of the sub-cube. The encoder uses octree decomposition to split the space into 1024^3 cubes, and proceeding as follows for each cube:

- Create a missingPointsPatch objects, which stores the sub-cube position in the input bounding box.
  - Update the origin of the sub-cube(u1, v1, d1)
  - Update the number of missing points inside
- Create a 2D patch in the luma video frame
  - Update the location in packed image (u0, v0)
  - Add the differential coordinated of each missing point.

$$dX = X - u1; dY = Y - v1; dZ = Z - d1 \tag{47}$$

The point local reconstruction mode is used simultaneously with the one-layer coding mode, adapting locally the reconstruction. For each 16x16 block, a reconstruction mode is determined on the encoder side, and signaled to the stream. To maximize compression efficiency, multiple prediction contexts are used, and the optimal one is selected automatically to encode the mode of the current block, based on the neighborhood

6.7 Occupancy map coding

Binary values are associated with B0xB0 sub-blocks in the same TxT block. If a sub-block contains at least a non-padded pixel, its value is associated with 1 (full-block), and 0 otherwise (non-full block), and the binary information is encoded for each TxT block, indicating if it is full or not.

6.8 Shape Reconstruction

The geometry gradient of a patch is used to determine the points to be separated. This method fills holes caused by projection points to the same location of a 2D plane, allowing the reduction of the number of missed points, and improves the visual quality of the reconstructed point cloud. For this determinations, a Sobel filter is applied on the D0 layer for each patch to calculate its gradient. If that gradient is larger than a predefined threshold t_1, points projected in that 2D location will be regarded as high gradient points, and those with similar orientation are grouped.. The number of high gradient points is counted and another threshold is used to filter the groups with fewer points.

7 G-PCC

The G-PCC (Geometry-Point Cloud Compression) [20] has two encoding modules to compress the geometry information: Octree and Triangle Soup (TriSoup), in which the first is based on an octree, while the second approach is based on surface reconstruction, using triangular primitives, after the enclosure of the model in an octree structure. In both the encoder and the decoder, the positions of the point cloud are decoded first, with the attribute coding dependent on the decoded geometry.

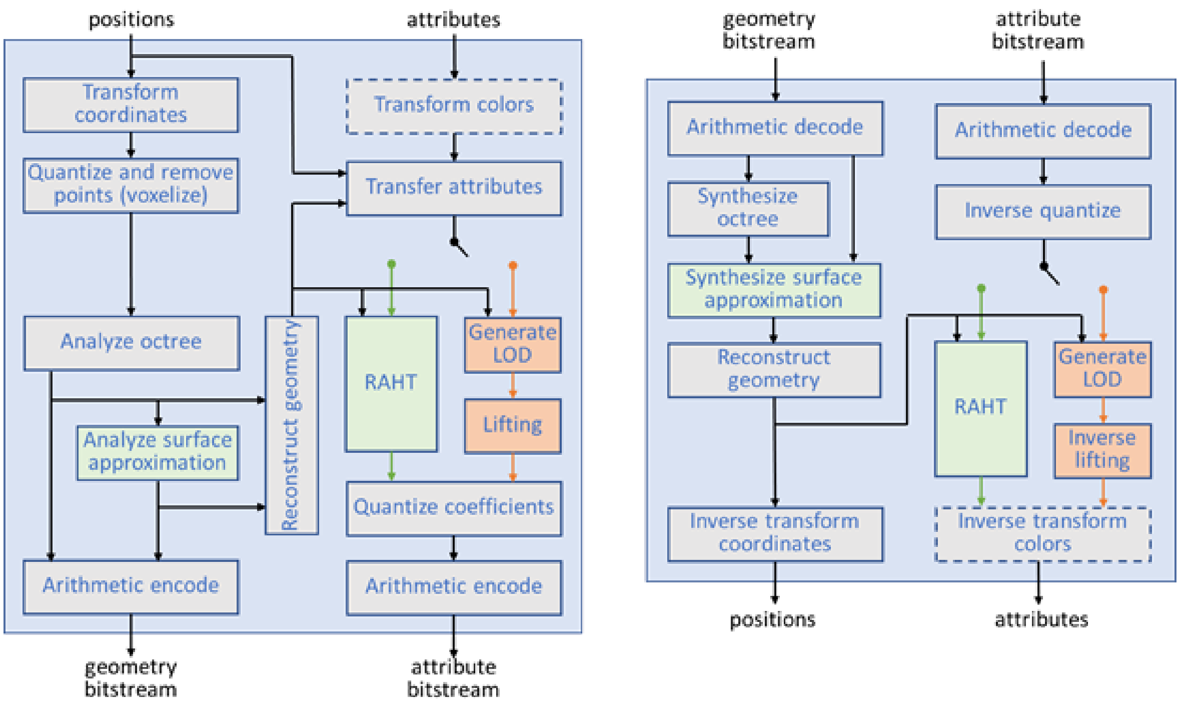


Figure 12: G-PCC encoder and decoder structure

7.1 Pre and post Processing

Let $X_n = (x_n, y_n, z_n)$ be the group of points positions belonging to the point clouds, $n = 1, \ldots, N$, be the number of the number of points in the point cloud, $A_n = A_{1n}, A_{2n}, \ldots, A_{Dn}$ the optional attributes of the point cloud with D being the number of attributes for each point. The point cloud geometry consists of only the points positions, and the attributes comprise only the attributes, and often, both the geometry and attributes are expressed in application-specific spaces. G-PCC provides pre and pos processing, to change between the application-specific spaces, and finite-resolution internal spaces, where the compression occurs.

7.1.1 Coordinate transform and inverse

The original application specific points positions are represented by floating point numbers, and don't possess structure, lying instead in a coordinate system denoted X_n^{orig}, $n = 1, \ldots, N$. Internal coordinates X_n^{int} are obtained from original coordinates by the following equation:

$$X_n = \frac{X_n^{orig} - T}{s} \tag{48}$$

with $T = (t_x, t_y, t_z)$. T and s are such that the point positions lie in a bounding cube $[0, 2^d]^3$, for some non-negative integer parameter d. Positions in the internal coordinate system which have been compressed and decompressed are denoted as $\hat{X}_n$, n = 1, ..., N_out, with N_out being the number of points in the decoded point cloud. The position for the decoded point in the original coordinate system are obtained from the decoded point positions in the internal coordinate system by the following transformation

$$\hat{X_n^{orig}} = s\hat{X}_n + T \tag{49}$$

and can be also expressed as

$$\begin{bmatrix} \hat{X_n^{orig}} \\ \hat{Y_n^{orig}} \\ \hat{Z_n^{orig}} \\ 1 \end{bmatrix} = \begin{bmatrix} s & 0 & 0 & t_x \\ 0 & s & 0 & t_y \\ 0 & 0 & s & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{X}_n \\ \hat{Y}_n \\ \hat{Z}_n \\ 1 \end{bmatrix} \tag{50}$$

In the case of Trisoup Geometry, s is specified in the configuration files by the triSoupIntToOrigScale parameter, with $T = [0, 0, 0]$ and d the triSoupDepth parameter. The component points (x_n, y_n, z_n) which are outside $[0, 2^d]^3$ are clipped to $[0, 2^d - 1]$ if needed. If the Octree case is selected, $\frac{1}{s}$ is the positionQuantizationScale parameter, $T = (minx_n^{orig}$, min y_n^{orig},min $z_n^{orig})$ and d determined by:

$$d = Ceil(\log_2(Max(x_n^{orig}, y_n^{orig}, z_n^{orig}) + 1)) \tag{51}$$

making sure that $[0, 2^d]^3$ is the smallest bounding cube containing the point positions in internal coordinates.

7.1.2 Point quantization and duplicate point removal

Positions are internally represented as non-negative d-bit integers before compression, thus the point positions in the internal coordinate system must be rounded, in the following way:

$$X_n = Round(X_n^{int}) \tag{52}$$

with X_n^{int} being the position in the internal coordinate system, and $Round$ is the function that rounds the components of a vector to the nearest integer. After quantization, multiple points may have the same position (duplicate points), and the removal process is optional. If enabled, it removes the points with the same quantized coordinates. Multiple points will be merged, if they have the same position and different values. Quantization, removal of duplicate points and assigning attributes to the remaining points is called voxelization. The voxels are unit cubes $[i - 0.5, i + 0.5] \times [j - 0.5, j + 0.5] \times [k - 0.5, k - 0.5]$, for integers i, j, k between $[0, 2^d - 1]$. Voxels are said to be occupied if they contain any point belonging to the point cloud.

7.2 Octree geometry encoding/decoding

For the octree geometry codec, a cubical axis-aligned bounding box B is defined by $(0, 0, 0)$ and $(2^d, 2^d, 2^d)$. Then an octree structure is created by subdividing B, with the cube being subdivided into 8 sub-cubes. An 8-bit code (occupancy code) is generated by associating a 1-bit value with each sub-code, to indicate if it contains points or not, which may be duplicated. As such, multiple points may be mapped to the same sub-cube size of 1. Due to this, the number of points for each sub-cube of dimension 1 is also encoded. The decoding process reads the dimensions of B from the bit stream, building the octree structure by subdividing B in accordance with the occupancy codes. Every time a sub-cube of dimension 1 is reached, the number of points c is decoded and c points located at the origin of the sub-cube are generated.

7.2.1 Direct coding

Octree representation shows efficiency at representing points with a spatial correlation, as it tends to factorize the higher order bits of the point coordinates. For an octree, each level of depth refines the coordinates of points within sub-volume by one bit for each component, and following compression is obtained by entropy coding the split information. On the other hand, isolated points P cannot be better coded than directly coding the coordinates with no compression, as there are no other points within the volume to correlate with. The act of directly coding point coordinates in a volume or sub-volume is called Direct Coding Mode. Isolated points pollute the distribution of patters, changing the balance of the distribution and penalizing the coding of other patterns, and it is extremely beneficial to remove the isolated points in octree coding, for better compression performance in volumes where point correlation exists. Octree coding and Direct Coding Mode can be combined.

A new eligibility condition is introduced, and it depends on information coming from the parent node itself or the neighbours of the parent node. If the node is not eligible, octree coding is applied, otherwise:

- A binary flag is coded to the signal if the DCM is applied or not to the node,
- If the flag value is 1 (DCM applied), points belonging to the associated volume are directly coded using DCM, otherwise, the octree coding process continue for the current mode.

If the node is eligible for DCM, a flag is coded to signal if the DCM is applied or not, and it may be determined based upon the number of points the volume attached to the node possesses. If the number is less than or equal to a threshold th, DCM is activated. In GPCC, $th = 2$ (up to two points can be decoded in a volume). When DCM coding occurs, it follows then next steps:

1. The number of points is coded using a truncated unary binarizer, followed by a binary entropy coder. One flag signals the number of points being coded, and then is entropy coded.
2. (X, Y, Z) are coded independently for each point, and relatively to the volume associated with the node, and then direct pushed into the bitstream.

7.3 Neighbour-Dependent Entropy context

Here the configuration is selected depending on the six neighbours N of the parent mode, and from those 6 neighbours, a neighbour configuration number (NC) is deduced to code occupancy pattern, with NC being an integer ranging from $[0, 63]$. The value 0 means no occupied neighbour, whereas the value 63 means that all neighbours are occupied, and among this 64 distributions, one is chosen directly. If NC=0:

- If the parent node only has one occupied child node, then the position is directly coded using 3-bits to code the occupied child node position in XYZ, inside the associated volume.
- Otherwise, the 0-th distribution corresponding to NC=0 is used.

By construction of the octree, a current cube associated with a current node is surrounded by six cubes of the same depth sharing a face with it. Using a breadth-first scanning order makes it so that the occupancy map of the six cubes neighbouring the current cube is known before decoding the 8-bit occupancy pattern of the current node. The set $D_j = [b_0, \ldots, b_{j-1}, NC]$ of states can be used with Optimal Binarization with Update on-the-fly, to code the occupancy bit b_j. The size of, for example, D_7 is $128x64 = 8192$ states, making this impractical for HW implementation, and also leads to dilution of occupancy statistics into too many states to obtain optimal compression performance. To solve this problems, the 64 neighbouring configurations NC are reduced to 10 invariant configurations, by geometry in-variance. If the local geometry correlation of the point cloud is invariant under 3D isometries, the neighbouring configuration can be transformed using $90\deg$ rotations and symmetry, to match only one of the ten configurations.

7.3.1 Intra Prediction

Using the six neighbours which share the same depth and also sharing a face with the current mode will not provide all possible information about local geometry (at least 26 neighbours should share a face, edge or vertex with the current node), but the number of possible patterns of occupancy for the 26 neighbours is far to high to be used directly in the sets D_j of states, even with complex and direct state reduction, thus a way of reducing the 26-neighbour pattern, to a ternary information which predicts the value of occupancy bits b_J is needed. Before applying any geometrical transform in order to reduce the neighbour configuration from NC to NC_{10}, an occupancy score $score_m$ is computed for each sub-node SN_m of a current node:

$$score_m = \frac{1}{26}\sum_{k=1}^{26} w_{k,m}(\delta_k) \tag{53}$$

where m is the sub-node index, k is the neighbour index, $w_{k,m}$ is the weight from the neighbour k to sub-mode m, and δ_k is the occupancy status f of the neighbour k. Using the anisotropy argument, the weights are considered as a function W of the euclidean distance $d_{k,m}$ between the neighbour k and sub-node m, and the occupancy status δ_k.

$$w_{k,m}(\delta_k) = W(d_{k,m}, \delta_k) \tag{54}$$

This function W is found by:

$$W(d_{k,m}, \delta_k) = \begin{cases} {\scriptstyle W0(d_{k,m}), \delta_k=0} \\ {\scriptstyle W1(d_{k,m}), \delta_k=1} \end{cases} \tag{55}$$

Where $W1$ and $W2$ are two look-up-tables with eight entries each:

$$W0 = [-1, -6, 12, 20, 14, 28, 22, 12] \tag{56}$$

$$W1 = [27, 39, 20, 8, 18, 4, 11, 18] \tag{57}$$

They have been constructed in a way such that the higher score $score_m$ indicates a higher probability of occupancy of the sub-node SN_m and the transition between a low and high probability of occupancy is the sharpest possible. The score can take many different values to be usable as in the set of states, and the probability of a sub-node to be occupied also depends on the number No of occupied neighbours in the 26. It is transformed into a ternary information $Pred_m$ belonging to the set, of three prediction states, by using $th_0(No)$ and $th_1(No)$. If $score_m$ is lower than $th_0(No)$, $Pred_m$ is set to "predicted non-occupied", if higher than $th_1(No)$, it is set to "Predicted occupied", and set to "not predicted" if the score value is in between the two thresholds After the geometrical transform reduces the neighbouring configuration from NC to NC_10, each child node CC_j inherits a prediction value $Pred[j]$ and the set of states takes the following form:

$$\mathcal{D}_j = [b_0, \ldots, b_{j-1}, NC_{10}, C[j], Pred[j]] \tag{58}$$

This causes the size of the eight sets D_j of states to be multiplied by a factor three, and the set of states becomes three copies of the sub-set without prediction

$$\mathcal{D}_j = [b_0, \ldots, b_{j-1}, NC_{10}, C[j]] \times Pred[j] \tag{59}$$

The two thresholds are determined empirically for the fie cases of occupied neighbours, and obtained from the following look-up-tables:

$$TH0 = [62, 60, 61, 59, 59]$$
$$TH1 = [67, 66, 65, 66, 64]$$

7.4 Trisoup geometry encoding/decoding

This is an option which represents the object surface as a series of triangle mesh, applicable for a dense surface point cloud. The decoder creates a point cloud from the mesh surface in a specified voxel granularity, assuring the density of the reconstructed point cloud. *Trisoup node size* defines the size of the triangle node. The octree encoding and decoding stop at leaf level l, and the leaf nodes of the octree represent cubes of width $W = 2^{maxNodeSizeLog_2,-l}$ and the octree is pruned.

7.4.1 Vertice Determination

When *trisoup node size*>0, the blocks are $2 \times 2 \times 2$, and there is the necessity of representing the voxels within the block. Within each block, geometry is represented as a surface intersecting each edge of the block, and since there are 12 edges in a block, there can be 12 intersections within a block, denominated as vertex. A vertex along an edge is detected exclusively when there is at least one occupied voxel adjacent to the edge among all blocks that share the edge, with its position being the average position along the edge of all such voxels adjacent to the edge, among all blocks sharing that edge. Vertices are shared across neighbouring blocks, guaranteeing both continuity across blocks of the surface, and reducing the number of bits needed to code the collection of vertices, which is coded in two steps:

1. All unique edges of occupied blocks are computed, and a bit vector determines which segment contain a vertex and which do not.
2. For each segment with a vertex, the position of the vertex along the segment is uniformly quantized to a small number of levels.

7.4.2 Triangle reconstruction

The vertices on the edges of a block determine a non-planar polygon surface through the block, triangulated as follows. With (x_i, y_i, z_i) be the coordinates of the vertex situated on the edges of the block. The centroid is computed

$$\begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{bmatrix} = \frac{1}{n} \sum_{i=1}^{n} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{60}$$

The mean-removed coordinates

$$\begin{bmatrix} \overline{x}_i \\ \overline{y}_i \\ \overline{z}_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{bmatrix} \tag{61}$$

and finally the variances

$$\begin{bmatrix} \sigma_x^2 \\ \sigma_y^2 \\ \sigma_z^2 \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} \overline{x}_i^{"} \\ \overline{y}_i^2 \\ \overline{z}_i^2 \end{bmatrix} \tag{62}$$

Then the minimum $(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ project each vertex onto the axis.

7.5 Recolouring

Given $(\tilde{X}_i)_{i=0,\ldots,N_{rec}-1}$, the attribute transfer procedure aims to determine the attribute values that minimize distortion of the attributes. If duplicated points are merged: For all $\tilde{X}_i$, X_i^* is its nearest neighbour in the original could and a_i^* the attribute associated with X_i^*, $\mathbb{Q}^+(i) = (X_i^+(h))_{h\in(1,\ldots,H(i))}$ the set of points that share $\tilde{X}_i$ as their neighbour in the reconstructed cloud. In the case of $\mathbb{Q}^+(i)$ being empty, the attribute a_i^* is associated with $\tilde{X}_i$, otherwise, the attribute associated with $\tilde{X}_i$ is obtained by:

$$\tilde{a}_i = \frac{1}{H(i)} \sum_{h=1}^{H(i)} a_i^+(h) \tag{63}$$

7.6 Attribute coding

7.6.1 Predicting transform

Level of detail generation

This process organizes the points in a set of refinement levels R_l, according to a set of Euclidean distances previously specified d_l. This is a deterministic process, operating on the quantized positions according to the octree decomposition process. All points are marked as not-checked, then the process iterates over all points, ignoring those who have already been checked. When the process reaches a not checked point, the minimum distance D of that point to the checked points V is computed, if $D < d_l$, the point is ignored, otherwise is added to R_l and V, repeating until all points have been checked. The level of detail 1 is obtained by the union of refinement levels $R_0, \ldots, R_l$. G-PCC uses a bottom-up approach instead of the one previously described, and an approximate nearest neighbour, accelerating LOD and predictor creation. $(P_i)_{i=1,\ldots,N}$ is the set of positions associated with the point cloud points and $M(i)_i = 1, ;N$ are the Morton codes associated with $P(i)$. D_0 is the initial sampling distance and ρ is the distance ratio between LODs The points are ordered according to their Morton codes in ascending order, with *I* being the array created by this sorting. Each iteration k extracts the points belonging to the k-th LOD and the predictors are build from $k = 0$, until all points are assigned to a LOD.

Simplified prediction structure in case of LOD equals one

With $(P_i)_{i=1,\ldots,N}$ being the set of positions associated with the points of the point cloud and $(M_i)_{i=1,\ldots,N}$ the Morton codes associated with P_i. The points are once again sorted in ascending order by their Morton codes, and let *I* be the created array. The encoder/decoder compresses/decompresses respectively the points according *I*. At each iteration *i*, a P_i is selected, and the distances of that points to the previous points are analyzed, and the *k* nearest neighbours are selected for prediction.

7.6.2 Lifting transform

This process is an improvement of the predicting transform, with the two main differences between the methods are the introduction of an update operator, and the use of an adaptive quantization strategy

Update Operator

The LoD predicition strategy previously described makes points in the lower LoD's more influential as they are used more often for prediction. An update operator is introduced to avoid that issue.

- Assume $w(P)$ as the influence weight associated with a point P. this value is set to 1 for all points
- All points are traversed according to the inverse order defined by the LOD structure.
- For each $Q(i,j)$, the weights of the neighbours $P \in \nabla(Q(i,j))$ are updated as:

$$w(P) \longleftarrow w(P) + w(Q(i,j),j)\alpha(P,Q(i,j)) \tag{64}$$

Now let $\delta(P)$ be the set of points $Q(i,j)$ such that $P \in \nabla(Q(i,j))$. The update operation for P is defined as:

$$Update(P) = \frac{\sum_{Q\in\Delta(P)}[\alpha(P,Q) \times w(Q) \times D(Q)]}{\sum_{Q\in\Delta(P)}[\alpha(P,Q) \times w(Q)]} \tag{65}$$

Adaptive quantization

The weights previously computed are leveraged to guide the quantization process (coefficients associated with P are multiplied by a factor of $\sqrt{w(P)}$. An inverse scaling process by the same factor is applied after the inverse quantization on the decoder side.

7.6.3 RAHT

Transform coding, Spatial transform and Quantization

Voxel colours $(\tilde{Y}_n, \tilde{U}_n, \tilde{V}_n), n = 1, \dots, N_{vox}$ are transform coded, by spatial transform, quantizer and entropy coder. The colours are spatially transformed by Region-Adaptive Hierarchical Transform, in order to obtain transformed colouts $(TY_n.TU_n, TV_n), n = 1, \dots, N_{vox}$ The transformed coordinates are then quantized by a uniform scalar quantizer to obtain the quantized transform coordinates $(\hat{TY}_n, \hat{TU}_n, \hat{TV}_n), n = 1, \dots, N_{vox})$

7.6.4 Attribute Entropy Encoding

The quantized transformed coefficients are entropy encoded using arithmetic coder. GPCC has an efficient binarization scheme for transform coefficients, requiring only a binary arithmetic encoder. The binarization approach has two different behaviours, for mono-dimensional attributes, and the three dimensional attributes.

8 Quality evaluation models

8.1 Subjective quality assessment

This is the most accurate way of estimating visual quality of a content. Test subjects are shown a number of original and degraded stimuli, and grade their quality based on a numeric scale, usually from 1 (very annoying) to 5 (no noticeable differences). The results of the evaluations are then averaged and expressed as the Mean Opinion Score (MOS), representing the quality grade attributed by the subjects to a given stimulus. These grades are collected following established methods and procedures, defined by experts, aiming to guarantee that identical experimental settings and conditions are implemented in different assessment campaigns. The most popular and widely adopted methods for subjective quality assessment in 2D image are described in the ITU-R Recommendation BT.500-13 [21], while the ITU-R Recommendation BT.2021-1 [22] contains the best methods for stereoscopic 3DTV systems. The test methods can be divided into single or double stimuli methods, based on the number of sitmuli a subject visualized to assess the quality of distorted content, but different grading scales are used to address different assessment problems. There is no standardization for point cloud subjective quality assessment. Amongst the many reasons for that, one of the most prominent difficulties is the fact that the type of rendering for this type of content is of the utmost importance. The simplest way of visualization is as a raw collection of points. Although this approach ensures an unique representations, it is an unnatural way of 3D model consumption, making the assessment of its visual quality a daunting task, mainly for naive observers. There are a number of ways to overcome this problem:

- A point is replaced by a surfel primitive, with its center being the point's coordinates and its attributes. The visual outcome of this method is highly dependable on size and quality of normals, which define the orientation of the surfel.
- Surface reconstruction algorithms used as a rendering methodology. The issue is, different algorithms may lead to different results, affecting the perceived quality of reconstructed content.

The display device must also be considered when conducting subjective experiments, as the various ways to consume point cloud content, offer different ways of interactivity, from conventional monitors, to head-mounted displays.

9 Objective quality assessment

These are mostly focused on full-reference metrics, which implies that distorted content must be compared to the original, computing a representative degradation value. The metrics will be categorized based on the type of degradation they can capture, which will be structure, texture or both.

9.1 Point cloud structure

In this category, metrics focus on geometry degradations of point cloud, and can be grouped in four classes:

9.1.1 Point-to-point

Based on the geometric distance of points between the reference and content. For every point b_k in the distorted point clouds, a point a_i in the original content. Then, individual error is computed using Euclidean distance $E = (a_i, b_k)$, and then associated to every point b_k, thus indicating the displacement between the original point cloud, and the distorted content [23].

$$E(a_i, b_k) = |(\overrightarrow{v}_{b_k}^{a_i})|_2 \tag{66}$$

9.1.2 Point-to-plane

For this metric, the reference normals are required, as this metric is based on the projected error along the normal surface of a reference point. If the original content is set as reference, the computation is straightforward. On the contrary case, the normal of a point is estimated by computing an average of the normals of a nearest neighborhood, belonging to the original content, implying every point of the distorted point cloud is a representative of its associated neighborhood. Finally, the projected error is estimated across this average normal. For every point b_k of the distorted content a point a_i in the original content is identified, then the projected error $E(a_i, b_k)$ across the normal N_{a_i} of the corresponding reference point is computed [23]:

$$E(a_i, b_k) = \left|(\overrightarrow{v}_{b_k}^{a_i} \cdot (N_{a_i})\right| \tag{67}$$

9.1.3 Plane-to-plane

This metric is based on the angular similarity of tangent planes corresponding to associated points between the original content and the distorted content. For each point b_k in the distorted content, a point a_i is identified in the original content, usually using the nearest neighbour algorithm, and then, using the normals from both points, the angular similarity of tangent planes is computed, using angle θ (angle between the normal vectors) as a reference [24].

$$AngularSimilarity = 1 - \frac{\theta}{\pi} \tag{68}$$

The error is associated to each and every single point in the distorted content, providing an approximation of the difference between the underlying local surfaces. This metric requires both contents to have normals present, and in their absence, they should be estimated.

9.1.4 Plane-to-mesh

For this specific case, there is no singular way to reconstruct an object using only a set of points. The plane-to-mesh metric is based on distances projected from the original content to the distorted content, a polygon mesh. The objective score heavily relies on the used surface reconstruction algorithm to convert point cloud content to a polygonal mesh, making this a sub-optimal solution for quality assessment of point cloud structure. In all but the plane-to-mesh metric, the metrics have an individual error, which is associated to every point belonging to the content under evaluation. For a total degradation value, Root Mean Square (RMS), Mean Squared Error (MSE), the Hausdorf distance, or an average of the individual values is computed. Both the distorted content and the original content should be used as reference in the metrics, as different pairs of points are obtained, leading to different objective scores. This is referred as symmetric error. PSNR (Peak-to-Signal Noise Ratio) is used in point-to-point and point-to-plane, to account for differently scaled contents. PSNR is defined as the ration of the squared maximum distance of the nearest neighbours of the original content.

9.2 Point Cloud Texture

Methodologies aiming to assess the color of a distorted model, have their pillar on conventional formulas associated with 2D content representation. For every b_k associated with the distorted content, there is a point a_i identified by a nearest neighbour algorithm. Following the formation of every pair of points, standard formulas, such as PSNR, use the color attributes, in order to compute a total color degradation value. Although conversions between them are common, the computations can be made either in RGB or YUV color space. Metrics for structure and color degradation often use the nearest neighbor algorithm for point associations belonging to both the original and distorted contents, and as such, they are executed in parallel.

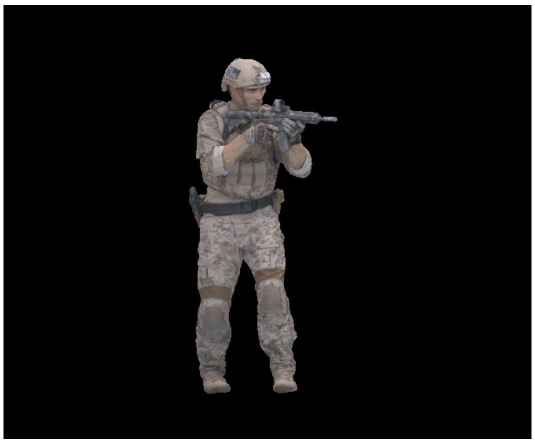
10 Experimental work

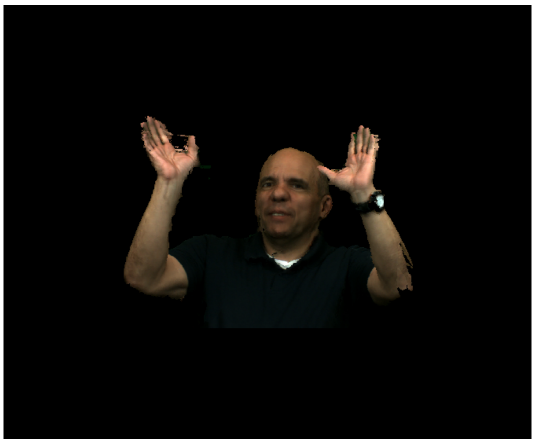
For the studies, a dataset of 6 point clouds were used,*Long Dress, Loot, Soldier, Red And Black, Ricardo10, Sarah9*. The point clouds in the set were compressed by using both V-PCC and G-PCC encoders. In the MPEG G-PCC encoder, only the Lifting transform was considered, and the content was encoded in 5 quality levels (R01-R06) spanning from very low to very high quality. For V-PCC codec, *Lossy Geometry-Lossy Attributes*, was the selected encoding condition and the coding mode *All Intra*, due to the usage of static point clouds. Six rates were used (R01-R05) spanning from very low to high quality, and one additional rate, which had a lower quality than R01 (RB01), comparable to the lowest quality rate point used in G-PCC.

(a) Longdress

(b) Loot

(c) Red And Black


(d) Soldier


(e) Ricardo10

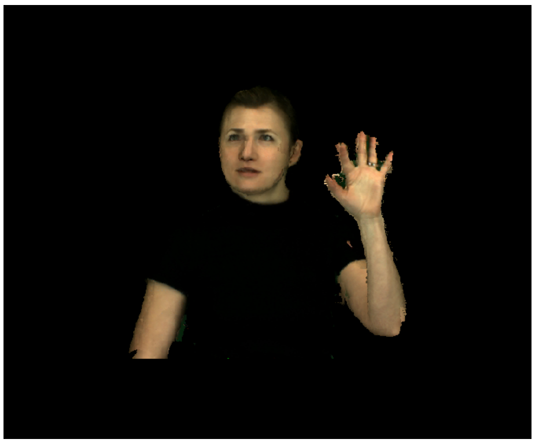

(f) Sarah9

Figure 13: Original Content

10.1 Coding with V-PCC

As described above, the para parameters for V-PCC codec, the *All-Intra* coding mode and the *Lossy-Geometry-Lossy-Consition* was the encoding condition were selected. Furthermore, a configuration mode needs to be selected, as well as the rate of the point cloud and the sequence to encode. For all point clouds, the coding mode and the encoding condition will not change, as the only parameters which require adjustments are the rate and sequence. The following sections shall describe the coding parameters, and the final results of each codec.

10.1.1 Coding parameters

Table 1: Ctc-common parameters

colorTransform	0
segmentation parameters	
nnNormalEstimation	16
maxNNCountRefineSegmentation	1024
iterationCountRefineSegmentation	10
voxelDimensionRefineSegmentation	4
searchRadiusRefineSegmentation	192
occupancyResolution	16
minPointCountPerCCPatchSegmentation	16
maxNNCountPatchSegmentation	16
surfaceThickness	4
maxAllowedDist2MissedPointsDetection	9
maxAllowedDist2MissedPointsSelection	1
lambdaRefineSegmentation	3
packing parameters	
minimumImageWidth	1280
minimumImageHeight	1280
colouring parameters	
bestColorSearchRange	0
numNeighborsColorTransferFwd	8
numNeighborsColorTransferBwd	1
useDistWeightedAverageFwd	1
useDistWeightedAverageBwd	1

skipAvgIfIdenticalSourcePointPresentFwd	1
skipAvgIfIdenticalSourcePointPresentBwd	1
distOffsetFwd	4
distOffsetBwd	4
maxGeometryDist2Fwd	1000
maxGeometryDist2Bwd	1000
maxColorDist2Fwd	1000
maxColorDist2Bwd	1000
occupancy map parameters	
maxCandidateCount	4
smoothing parameters	
flagGeometrySmoothing	1
gridSmoothing	1
gridSize	8
thresholdSmoothing	64
neighborCountSmoothing	64
radius2Smoothing	64
radius2BoundaryDetection	64
colour pre-smoothing parameters	
thresholdColorPreSmoothing	10.0
thresholdColorPreSmoothingLocalEntropy	4.5
radius2ColorPreSmoothing	64
neighborCountColorPreSmoothing	64
flagColorPreSmoothing	1
partitioning	
enablePointCloudPartitioning	0

Table 2: Ctc-all-intra parameters

constrainedPack	0
deltaCoding	0
globalPatchAllocation	0

Table 3: Rate configuration

Rate	GeometryQP	TextureQP	Occupancy Precision
R1	32	42	4
R2	28	37	4
R3	24	32	4
R4	20	27	4
R5	16	22	2
RB1	36	47	4

Table 4: Sequence configuratrion

Parameter	Longdress	Loot	Red and Black	Soldier	Ricardo10	Sarah
geometry3dCoordinatesBitdepth	10	10	10	10	10	9
geometryNominal2dBitdepth	8	8	8	8	8	8
frameCount	1					
startFrameNumber	1	1200	1550	0690	1200	0139
groupOfFramesSize	1	1	32	1	1	1
iterationCountRefineSegmentation	50	-	-	-	-	-
minNormSumOfInvDist4MPSelection	0.33	0.46	0.36	0.36	0.46	0.46
partialAdditionalProjectionPlane	0.17	0.17	0.15	0.15	0.17	0.17
voxelDimensionRefineSegmentation	-	2	2	2	2	2
partitions/ROIs and tiles						
roiBoundingBoxMinX	0,0,0,0					
roiBoundingBoxMaxX	1023,1023,1023,1023					
roiBoundingBoxMinY	0,256,512,768					
roiBoundingBoxMaxY	255,511,767,1023					
roiBoundingBoxMinZ	0,0,0,0					
roiBoundingBoxMaxZ	1023,1023,1023					
numTilesHor	2					
tileHeightToWidthRatio	1					
numCutsAlong1stLongestAxis	2					
numCutsAlong2ndLongestAxis	1					
numCutsAlong3rdLongestAxis	1					

10.1.2 Bitrate V-PCC

Table 5: Bitrate V-PCC

Longdress		
Rate	Points In	Points Out
Rb01	857966	692658
R01	857966	786404
R02	857966	844331
R03	857966	847585
R04	857966	869132
R05	857966	834446

Loot		
Rate	Points In	Points Out
Rb01	805285	641948
R01	805285	709137
R02	805285	778311
R03	805285	804164
R04	805285	825997
R05	805285	797654

Red and Black		
Rate	Points In	Points Out
Rb01	757691	604109
R01	757691	703382
R02	757691	749495
R03	757691	758003
R04	757691	791645
R05	757691	744512

Soldier		
Rate	Points In	Points Out
Rb01	1089091	863467
R01	1089091	990677
R02	1089091	106771
R03	1089091	1078987
R04	1089091	1109362
R05	1089091	1062926

Ricardo10		
Rate	Points In	Points Out
Rb01	1414040	1176347
R01	1414040	1207753
R02	1414040	1245891
R03	1414040	1257592
R04	1414040	1264473
R05	1414040	1192778

Sarah9		
Rate	Points In	Points Out
Rb01	299363	255101
R01	299363	274172
R02	299363	281527
R03	299363	283604
R04	299363	285574
R05	299363	268389

The commands for coding content on V-PCC are as follows

```
PccAppEncoder
  −−config=common/ctc−common.cfg
  −−config=sequence/my_longdress_vox10_1300.cfg
  −−config=condition/ctc−all
```

```
    --intra.cfg
    --config=rate/ctc-r1.cfg
    --configurationFolder=mycfg/
    --uncompressedDataFolder=pcdata/
    --frameCount=1
    --colorSpaceConversionPath=HDRConvert
    --videoEncoderPath=TAppEncoderStatic
    --nbThread=1
    --keepIntermediateFiles=1
    --reconstructedDataPath=longdress_vox10_1300_ai_r01.ply
    --compressedStreamPath=longdress_vox10_1300_ai_r01.bin
```

This example codes the R1 rate of the Longdress point cloud. To code other rates, it is only necessary to change the config=rate line, to the desired rate (r1,r2,r3,r4,r5,rb1) and to change the name of the final two lines to the desired rate as well. To code other content the config=sequence line must be changed to the desired sequence configuration.

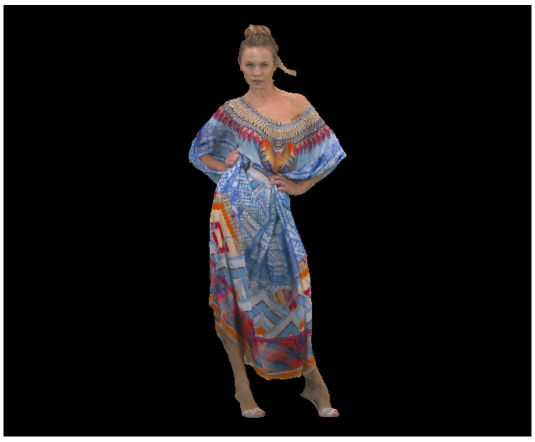
10.1.3 Resulting Images

(a) Longdress R01

(b) Longdress R02

(c) Longdress R03

(d) Longdress R04


(e) Longdress R05

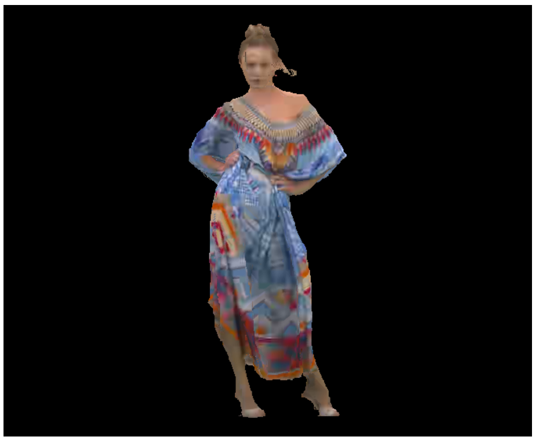
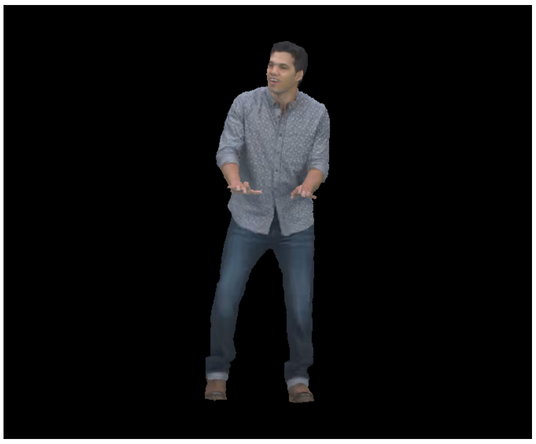
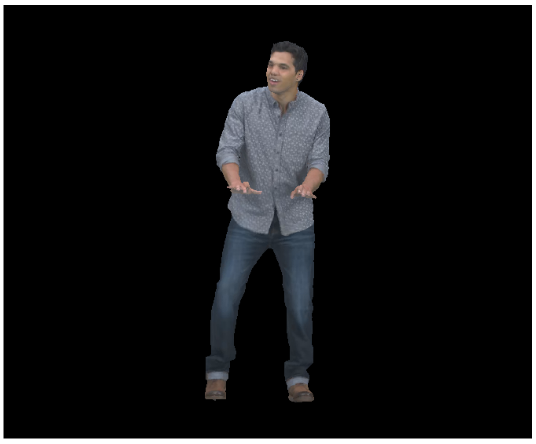

(f) Longdress RB01

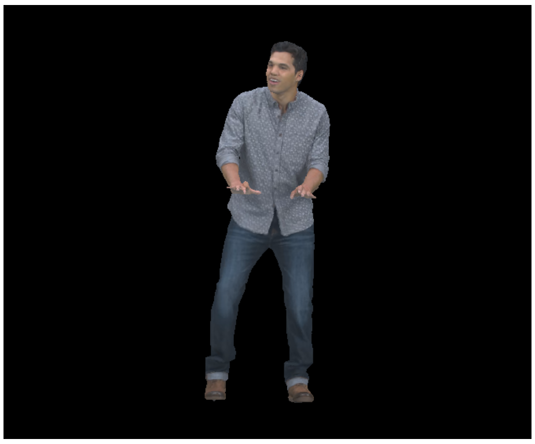
Figure 14: Final coding of Longdress with V-PCC

(a) Loot R01


(b) Loot R02

(c) Loot R03


(d) Loot R04


(e) Loot R05

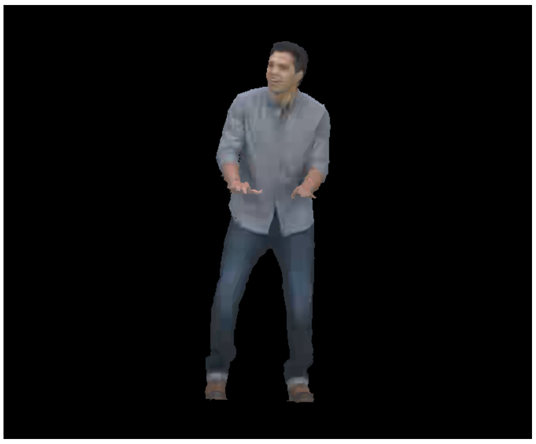

(f) Loot RB01

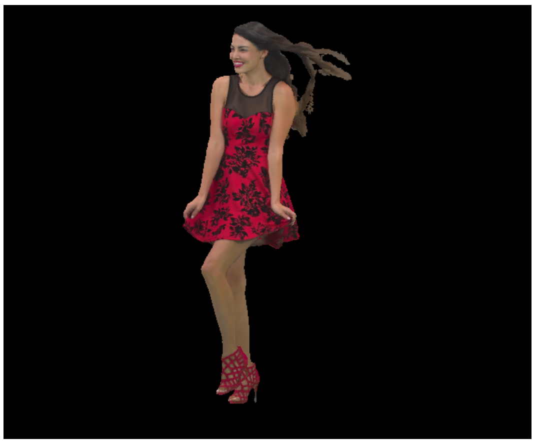
Figure 15: Final coding of Loot with V-PCC

(a) Red and Black R01

(b) Red and Black R02

(c) Red and Black R03

(d) Red and Black R04


(e) Red and Black R05

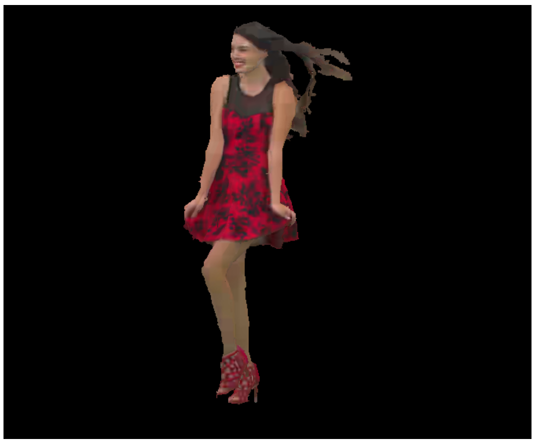
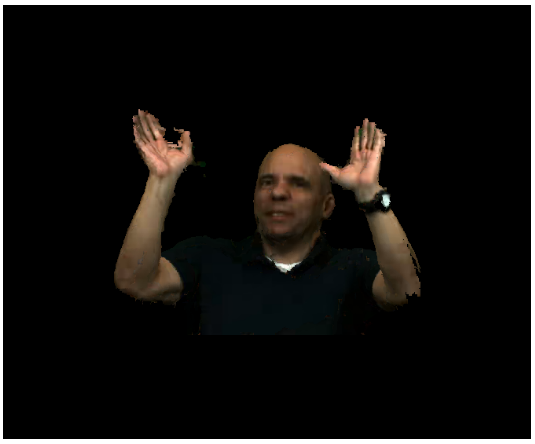
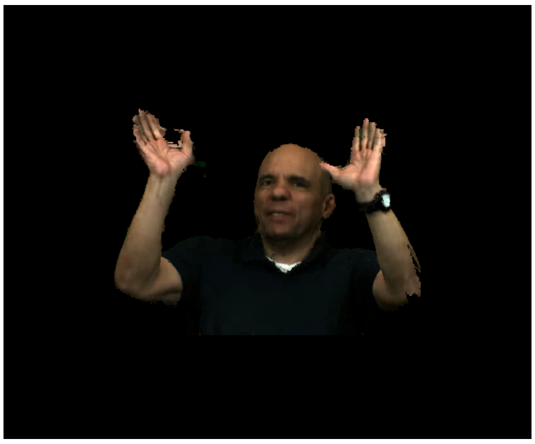

(f) Red and Black RB01

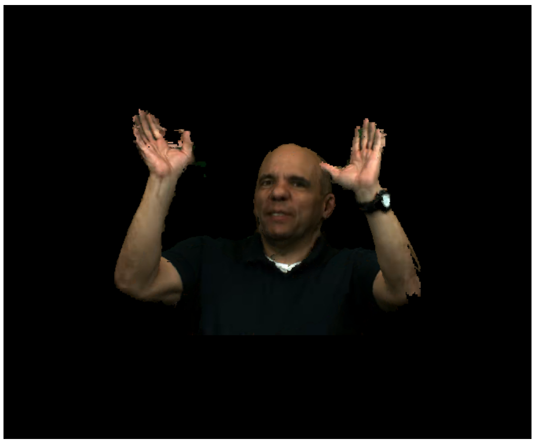
Figure 16: Final coding of Red and Black with V-PCC



(a) Ricardo10R01

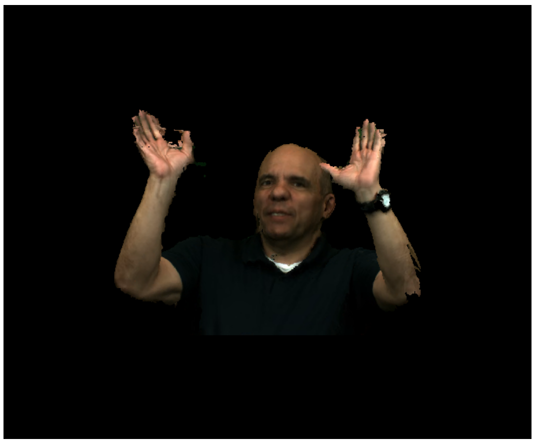


(b) Ricardo10R02

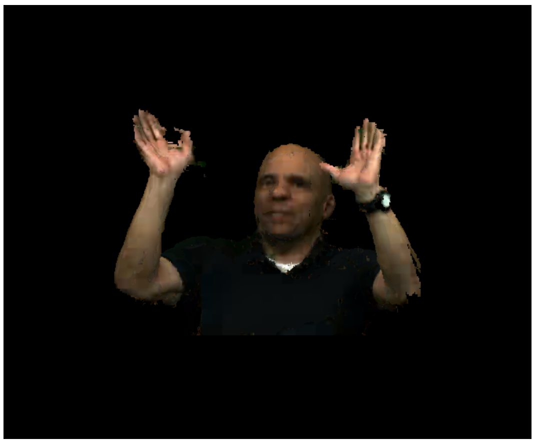


(c) Ricardo10R03

(d) Ricardo10R04

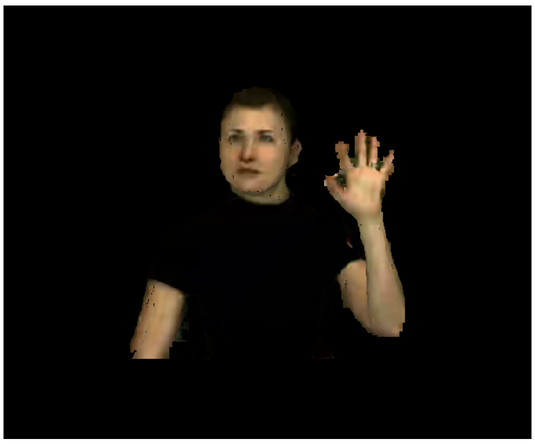


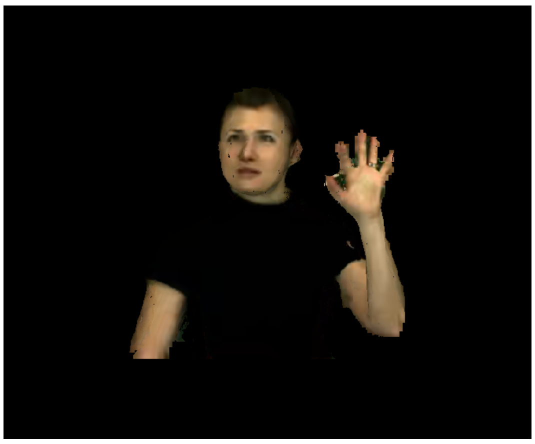
(e) Ricardo10R05

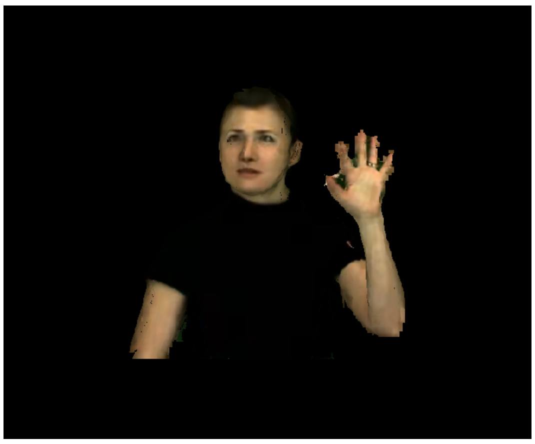


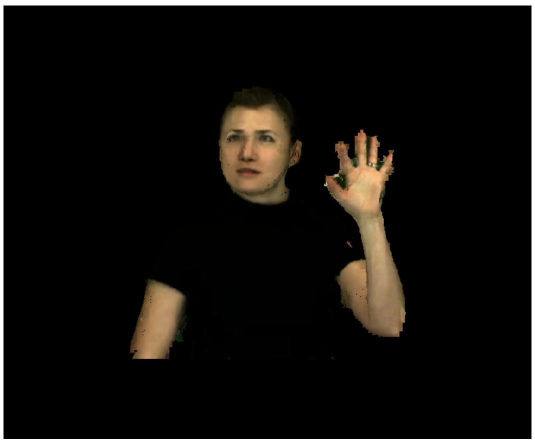
(f) Ricardo10RB01

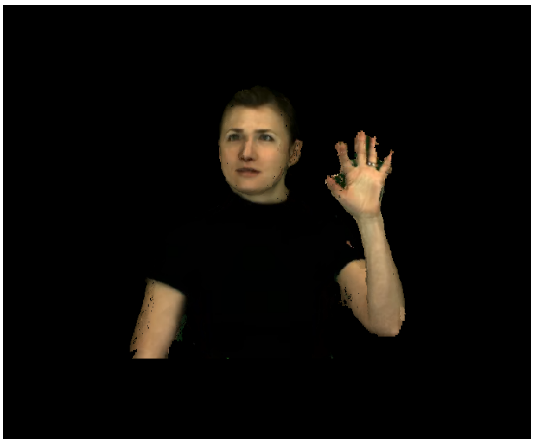
Figure 17: Final coding of Ricardo10 with V-PCC


(a) Sarah9R01


(b) Sarah9R02


(c) Sarah9R03


(d) Sarah9R04


(e) Sarah9R05

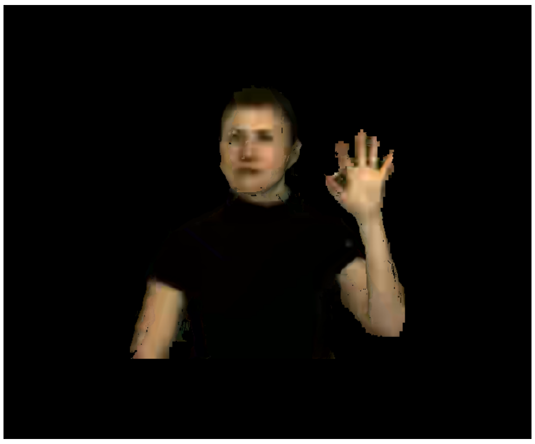
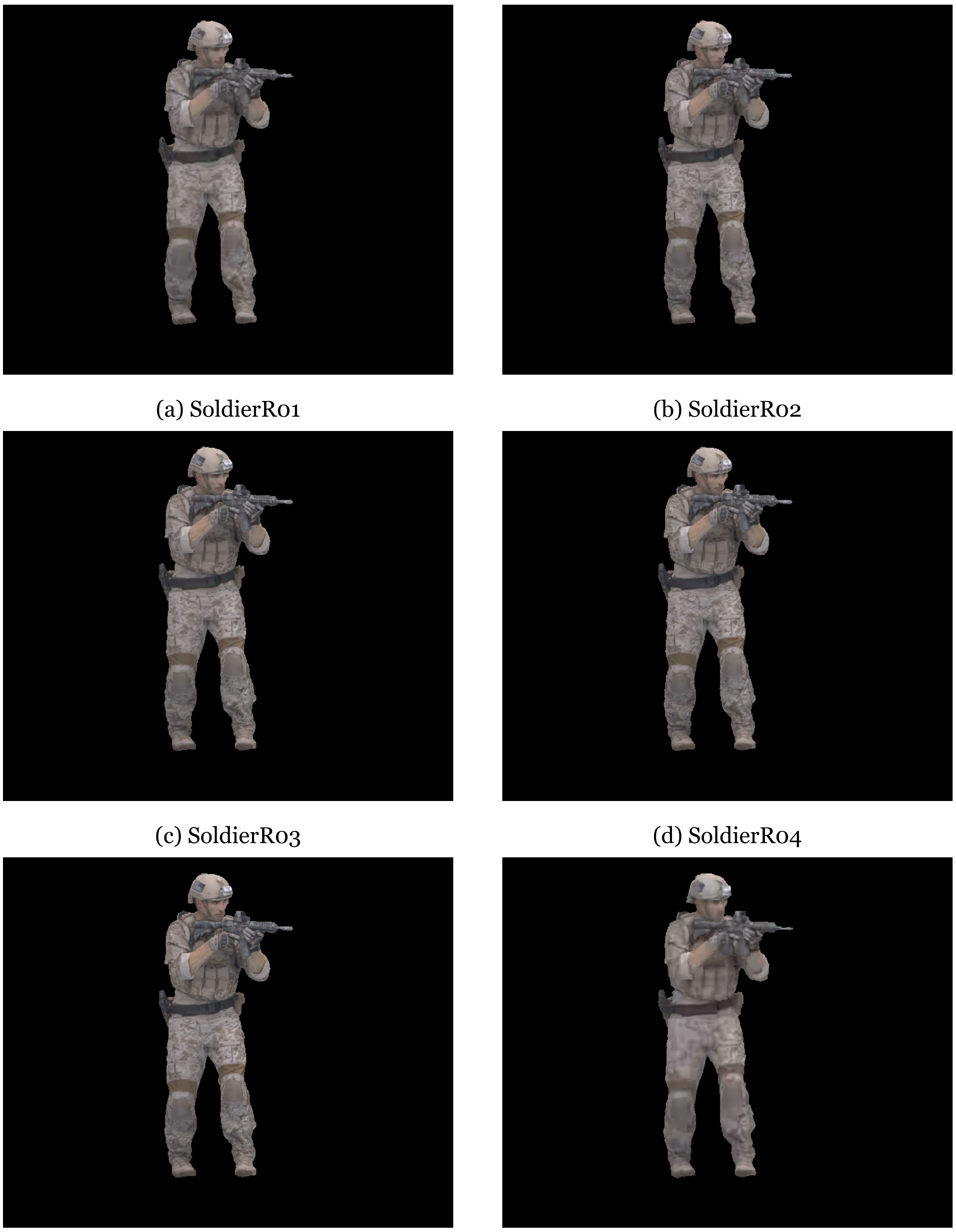

(f) Sarah9RB01

Figure 18: Final coding of Sarah9 with V-PCC



(a) SoldierR01 (b) SoldierR02

(c) SoldierR03 (d) SoldierR04

(e) SoldierR05 (f) SoldierRB01

Figure 19: Final coding of Soldier with V-PCC

10.2 Coding with G-PCC

As it was previously referred, only the lifting transform was considered for G-PCC, with 5 quality levels. The encoding condition was *Lossy-Geometry-Lossy-Attributes* for both octree and TriSoup. For the decoder, the values are always the same

10.2.1 Octree Coding parameters

Table 6: Decoder parameters in the G-PCC codec

mode	1
colorTransform	1

The values for the decoder are described in the following table:

Table 7: Octree parameters for the encoder in G-PCC

Parameters	R1	R2	R3	R4	R5	R6
mode	0					
trisoup_node_size_log2	0					
mergeDuplicatedPoints	1					
ctxOccupancyReductionFactor	3					
neighbourAvailBoundaryLog2	8					
intra_pred_max_node_size_log2	6					
positionQuantizationScale	0.125	0.25	0.5	0.75	0.875	0.9375
colorTransform	1					
transformType	2					
numberOfNearestNeighborsInPrediction	3					
levelOfDetailCount	12					
positionQuantizationScaleAdjustsDist2	1					
dist2	3					
lodDecimation	0					
adaptivePredictionThreshold	64					
qp	52	46	40	34	28	22
qpChromaOffset	0					
bitdepth	8					
attribute	color					

10.2.2 G-PCC Octree Bitrate

Table 8: Bitrate for G-PCC Octree

Longdress		
Rate	Points In	Points Out
R01	857966	15685
R02	857966	62123
R03	857966	238492
R04	857966	509139
R05	857966	674930
R06	857966	764041

Loot		
Rate	Points In	Points Out
R01	805285	14828
R02	805285	58133
R03	805285	222706
R04	805285	476836
R05	805285	632549
R06	805285	717396

Red and Black		
Rate	Points In	Points Out
R01	757691	13832
R02	757691	55024
R03	757691	211142
R04	757691	450075
R05	757691	596688
R06	757691	675202

Ricardo10		
Rate	Points In	Points Out
R01	1414040	20609
R02	1414040	83399
R03	1414040	342627
R04	1414040	781837
R05	1414040	1073763
R06	1414040	1237647

Sarah9		
Rate	Points In	Points Out
R01	299363	18355
R02	299363	74065
R03	299363	167054
R04	299363	227530
R05	299363	262681
R06	299363	280743

Soldier		
Rate	Points In	Points Out
R01	1089091	20065
R02	1089091	79099
R03	1089091	302674
R04	1089091	647359
R05	1089091	856781
R06	1089091	970797

The commands to encode content with G-PCC are as follows:

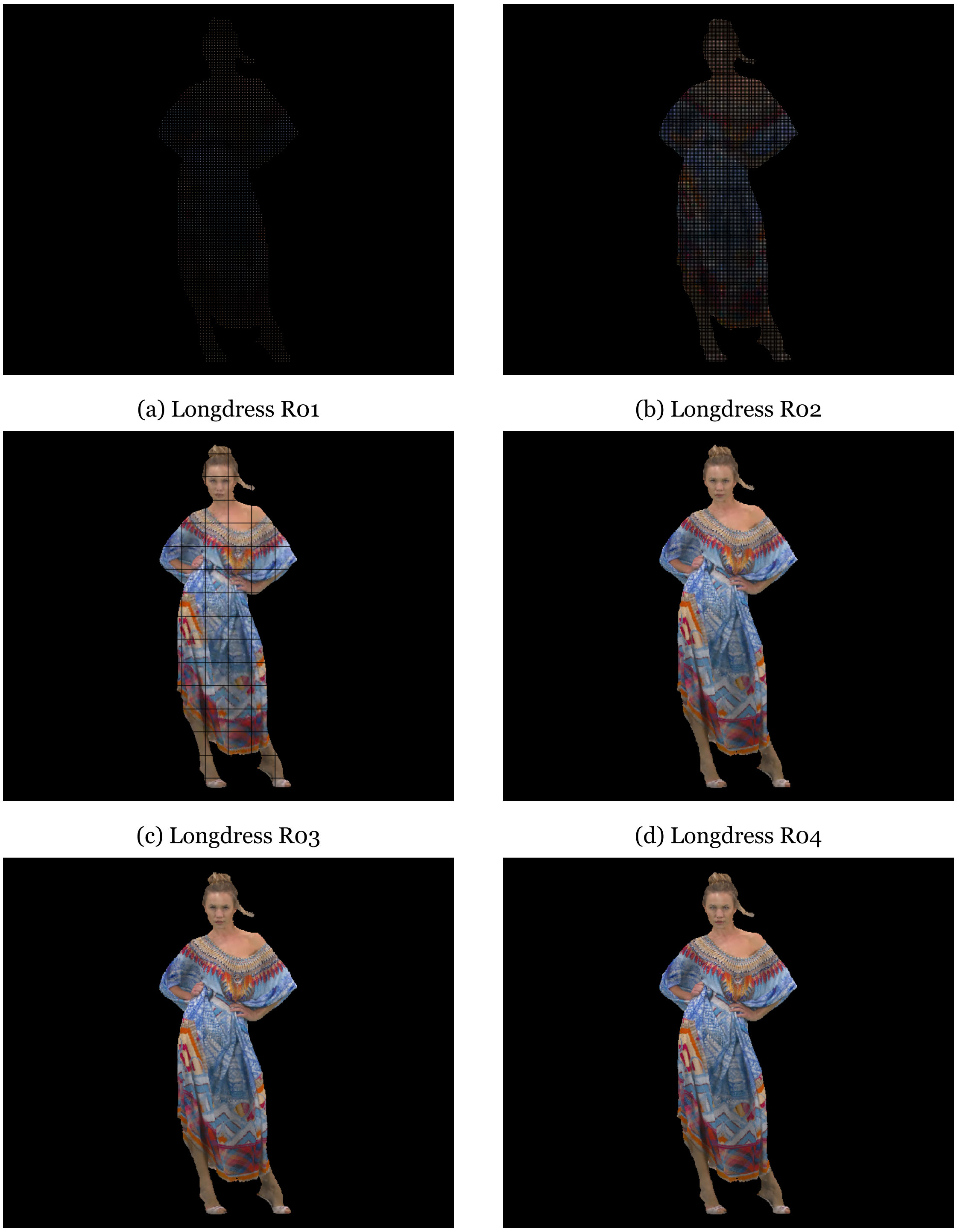
```
make –f $PWD/scripts/Makefile.tmc13–step –C
–$PWD/vox/soldier/trisoup/r01
–VPATH=$PWD/cfg_ctc/trisoup–predlift/...
    lossy–geom–lossy–attrs/soldier_vox10_0690/r01
–ENCODER=$PWD/build/tmc3/tmc3
```

```
–DECODER=$PWD/ build/tmc3/tmc3
–PCERROR=/home/user/mpeg–pcc–dmetric–master/test/pc_error
–SRCSEQ=$PWD/vox/soldier/trisoup/r01/soldier_vox10_0690.ply
–NORMSEQ=$PWD/vox/soldier/trisoup/r01/soldier_vox10_0690.ply
```

The example codes the r01 rate of the Soldier point cloud, using TriSoup coding. For octree coding, the code would be:

```
make –f $PWD/scripts/Makefile.tmc13–step –C
–$PWD/vox/soldier/trisoup/r01
–VPATH=$PWD/cfg_ctc/octree–predlift/...
    lossy–geom–lossy–attrs/soldier_vox10_0690/r01
–ENCODER=$PWD/ build/tmc3/tmc3
–DECODER=$PWD/ build/tmc3/tmc3
–PCERROR=/home/user/mpeg–pcc–dmetric–master/test/pc_error
–SRCSEQ=$PWD/vox/soldier/trisoup/r01/soldier_vox10_0690.ply
–NORMSEQ=$PWD/vox/soldier/trisoup/r01/soldier_vox10_0690.ply
```

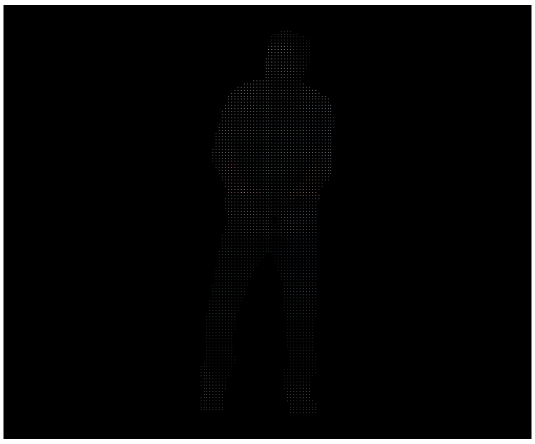
10.2.3 Resulting Images

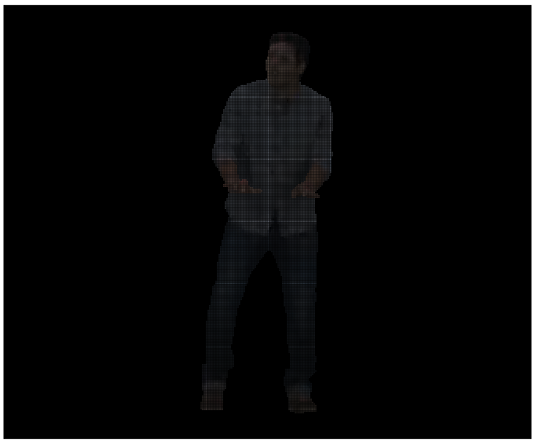

(a) Longdress R01 (b) Longdress R02

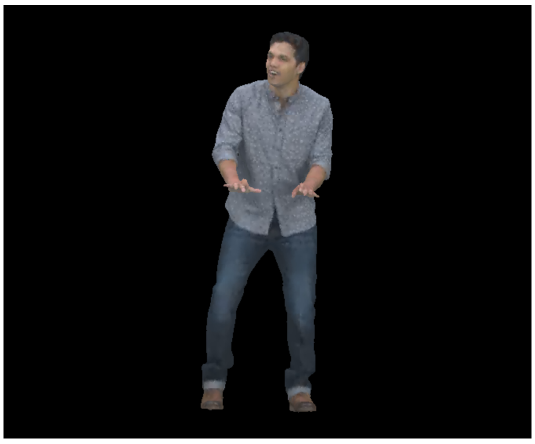
(c) Longdress R03 (d) Longdress R04

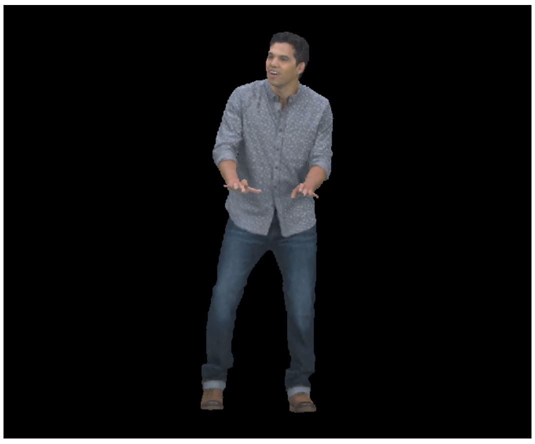
(e) Longdress R05 (f) Longdress R06

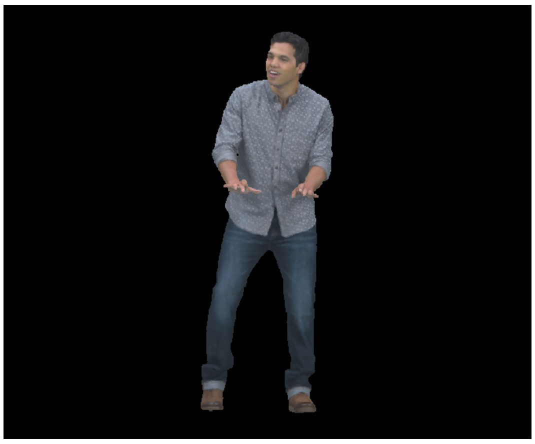
Figure 20: Final coding of Longdress with G-PCC (OCTREE)


(a) Loot R01


(b) Loot R02


(c) Loot R03


(d) Loot R04


(e) Loot R05

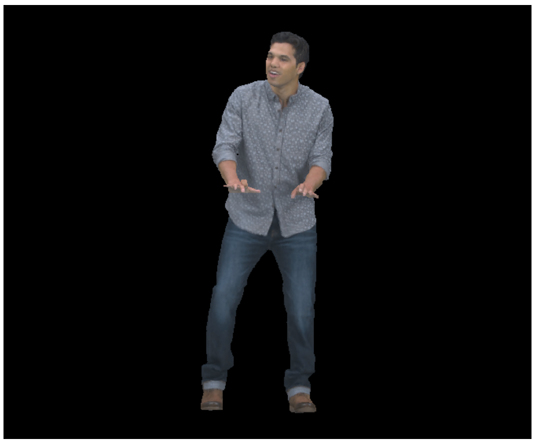
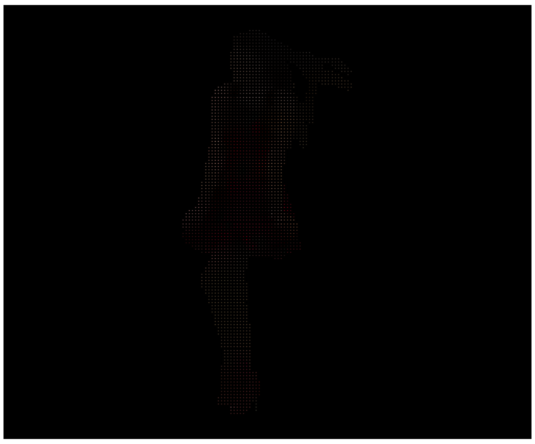
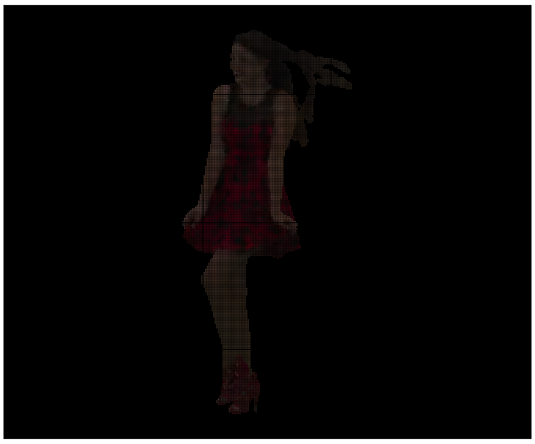
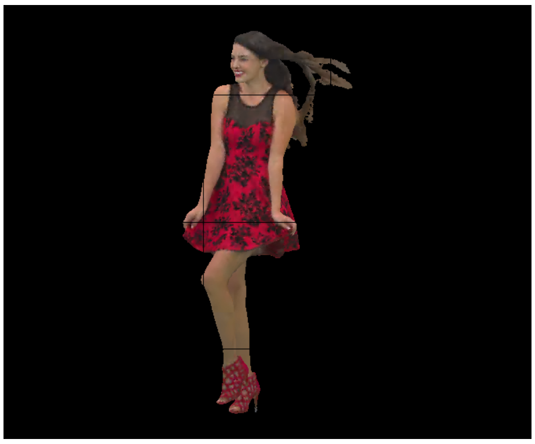
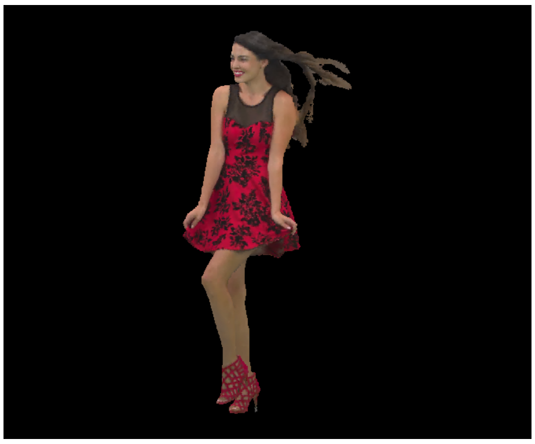

(f) Loot R06

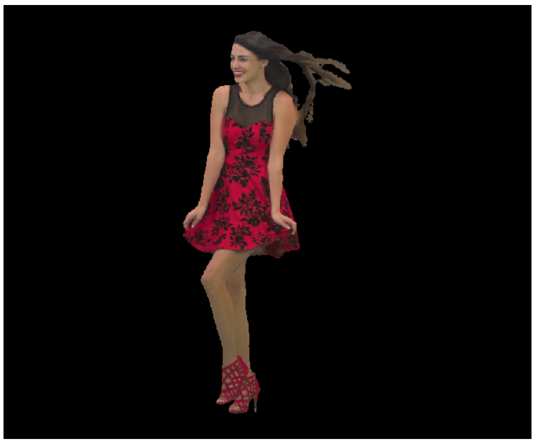
Figure 21: Final coding of Loot with G-PCC (OCTREE)


(a) Red and Black R01


(b) Red and Black R02


(c) Red and Black R03


(d) Red and Black R04


(e) Red and Black R05

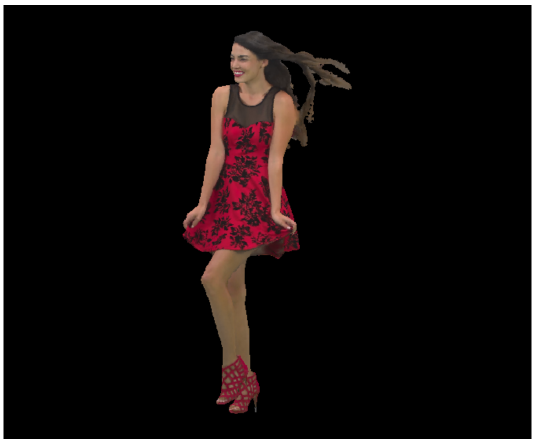
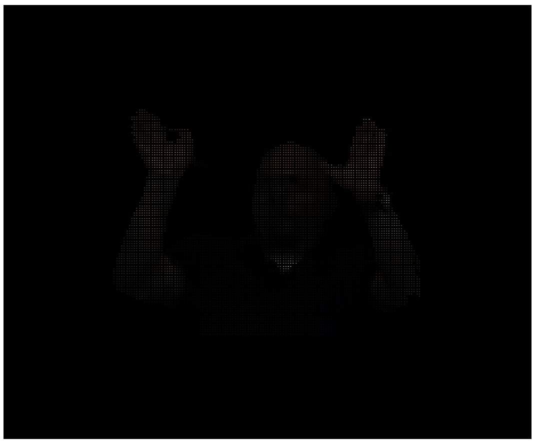
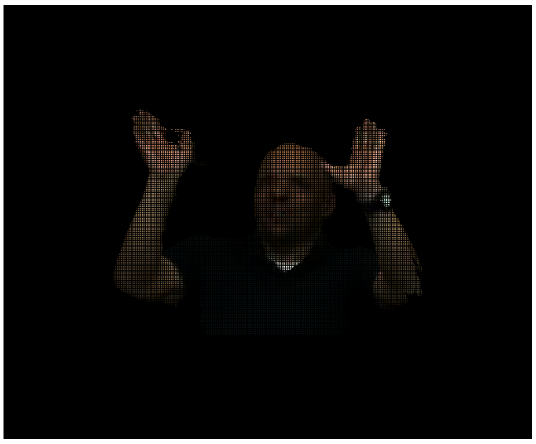
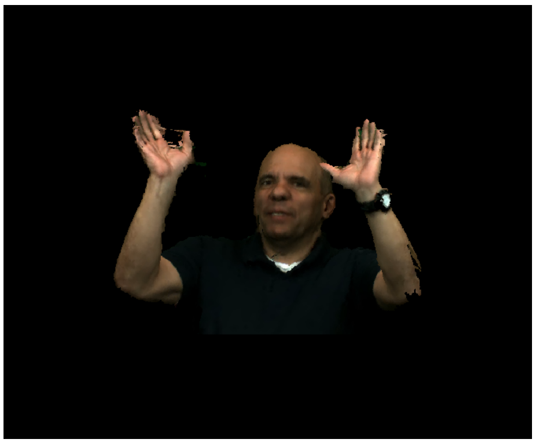

(f) Red and Black R06

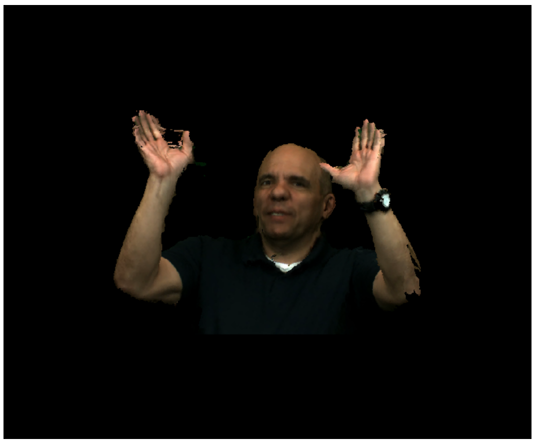
Figure 22: Final coding of Red And Black with G-PCC (OCTREE)


(a) Ricardo10 R01


(b) Ricardo10 R02

(c) Ricardo10 R03


(d) Ricardo10 R04


(e) Ricardo10 R05

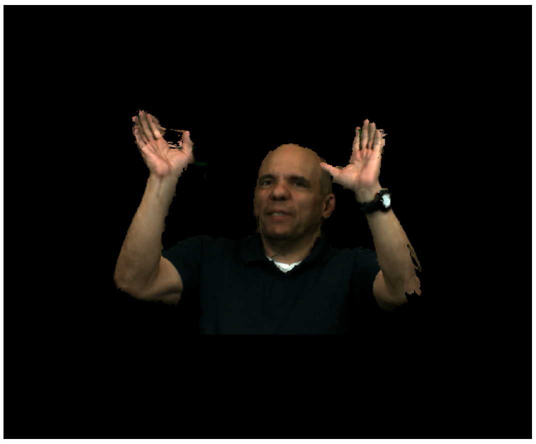
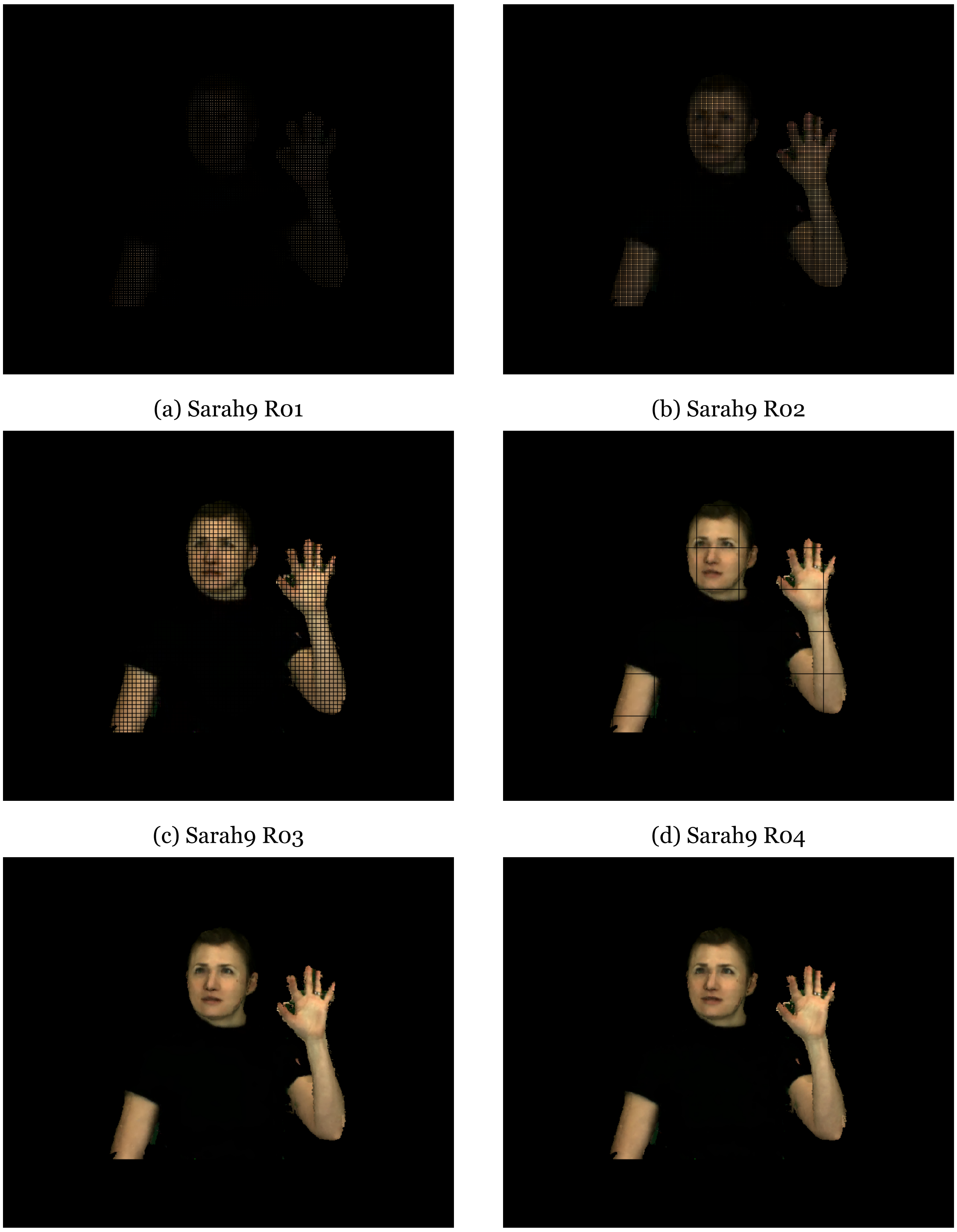

(f) Ricardo10 R06

Figure 23: Final coding of Ricardo10 with G-PCC (OCTREE)



(a) Sarah9 R01

(b) Sarah9 R02

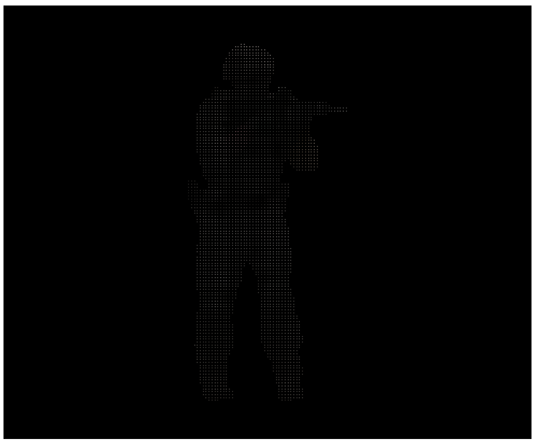
(c) Sarah9 R03

(d) Sarah9 R04

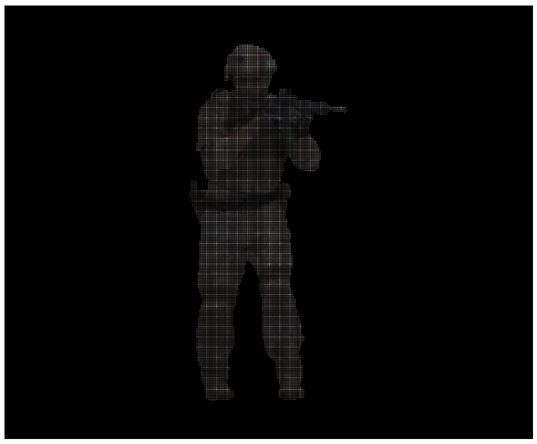
(e) Sarah9 R05

(f) Sarah9 R06

Figure 24: Final coding of Sarah9 with G-PCC (OCTREE)



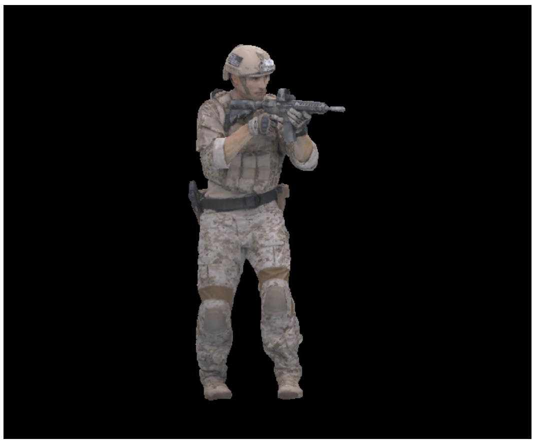
(a) Soldier R01



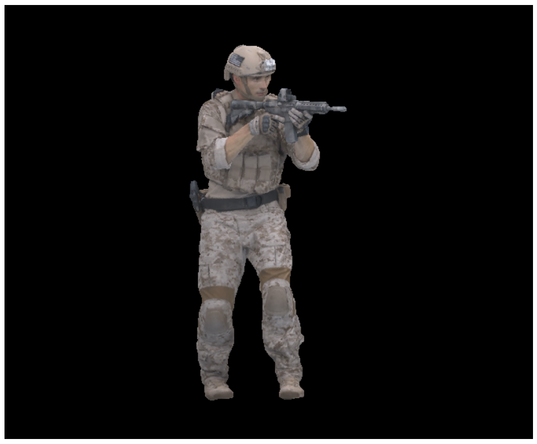
(b) Soldier R02

(c) Soldier R03

(d) Soldier R04



(e) Soldier R05



(f) Soldier R06

Figure 25: Final coding of Soldier with G-PCC (OCTREE)

10.2.4 G-PCC Trisoup Coding Parameters

Table 9: Trisoup parameters for the decoder in G-PCC

Parameters	R1	R2	R3	R4	R5	R6
mode	0					
trisoup_node_size_log2	4	3	3	3	3	2
mergeDuplicatedPoints	1					
ctxOccupancyReductionFactor	3					
neighbourAvailBoundaryLog2	8					
intra_pred_max_node_size_log2	6					
positionQuantizationScale	1					
colorTransform	1					
transformType	2					
numberOfNearestNeighborsInPrediction	3					
levelOfDetailCount	12					
positionQuantizationScaleAdjustsDist2	1					
dist2	3					
lodDecimation	0					
adaptivePredictionThreshold	64					
qp	52	46	40	34	28	22
qpChromaOffset	0					
bitdepth	8					
attribute	color					

10.2.5 G-PCC Trisoup Bitrates

Table 10: Bitrate for G-PCC Trisoup

Longdress		
Rate	Points In	Points Out
R01	857966	607613
R02	857966	642798
R03	857966	642798
R04	857966	642798
R05	857966	719478
R06	857966	857966

Loot		
Rate	Points In	Points Out
R01	805285	565258
R02	805285	599131
R03	805285	599131
R04	805285	599131
R05	805285	673881
R06	805285	805285

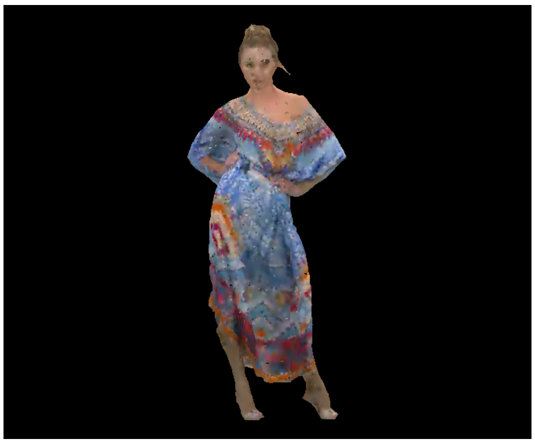
Red and Black		
Rate	Points In	Points Out
R01	757691	13832
R02	757691	55024
R03	757691	211142
R04	757691	450075
R05	757691	596688
R06	757691	675202

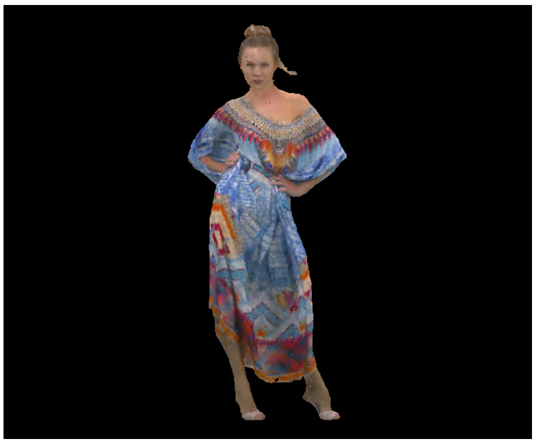
Ricardo10		
Rate	Points In	Points Out
R01	1414040	639760
R02	1414040	762658
R03	1414040	762658
R04	1414040	762658
R05	1414040	1024191
R06	1414040	1414040

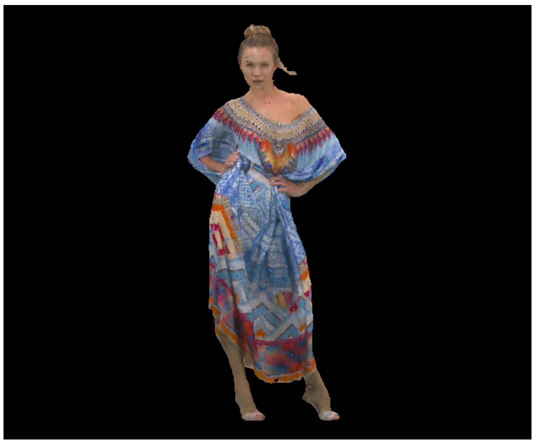
Sarah9		
Rate	Points In	Points Out
R01	299363	144267
R02	299363	169350
R03	299363	169350
R04	299363	169350
R05	299363	216490
R06	299363	299363

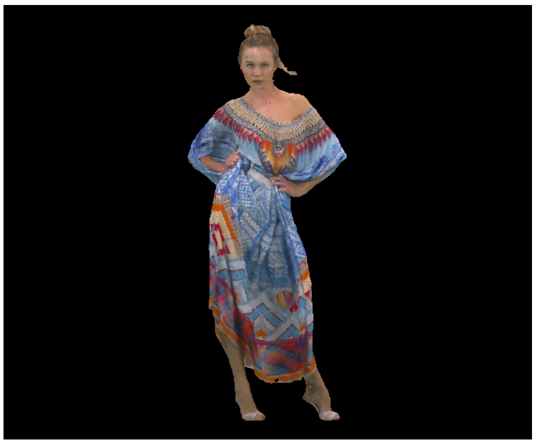
Soldier		
Rate	Points In	Points Out
R01	1089091	754051
R02	1089091	800026
R03	1089091	800026
R04	1089091	800026
R05	1089091	907062
R06	1089091	1089091

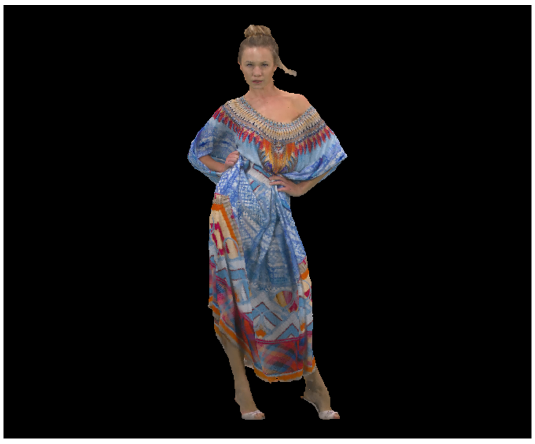
10.2.6 Resulting Images


(a) Longdress R01


(b) Longdress R02


(c) Longdress R03


(d) Longdress R04


(e) Longdress R05

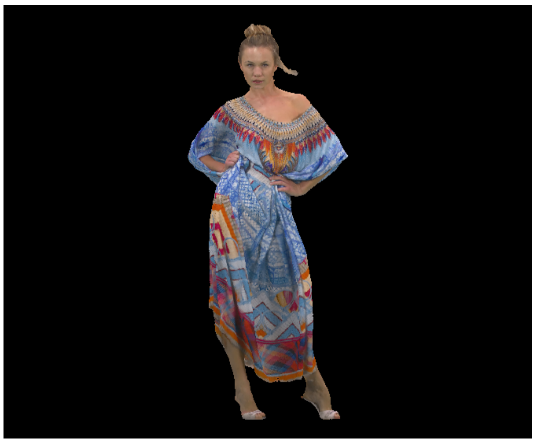
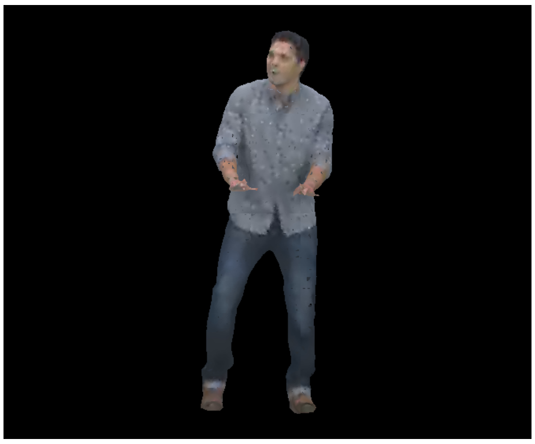
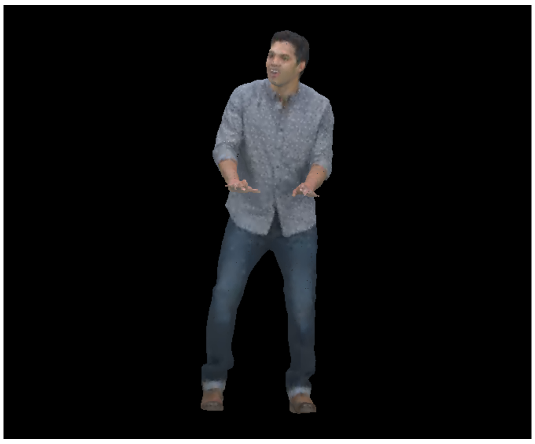
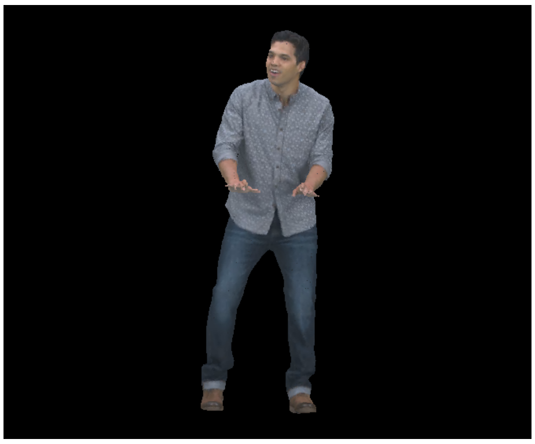

(f) Longdress R06

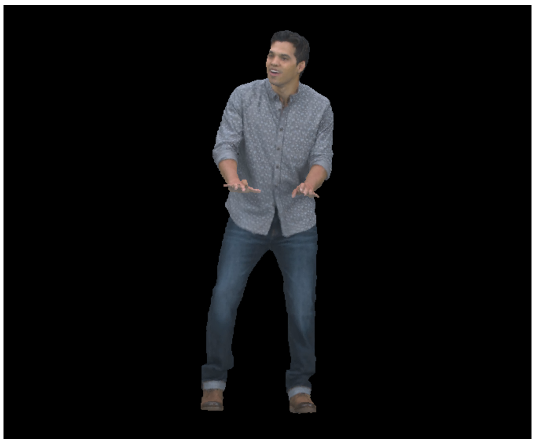
Figure 26: Final coding of Longdress with G-PCC (TRISOUP)


(a) Loot R01


(b) Loot R02

(c) Loot R03


(d) Loot R04


(e) Loot R05

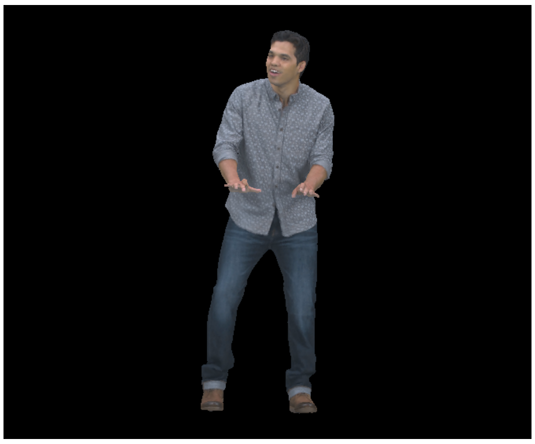
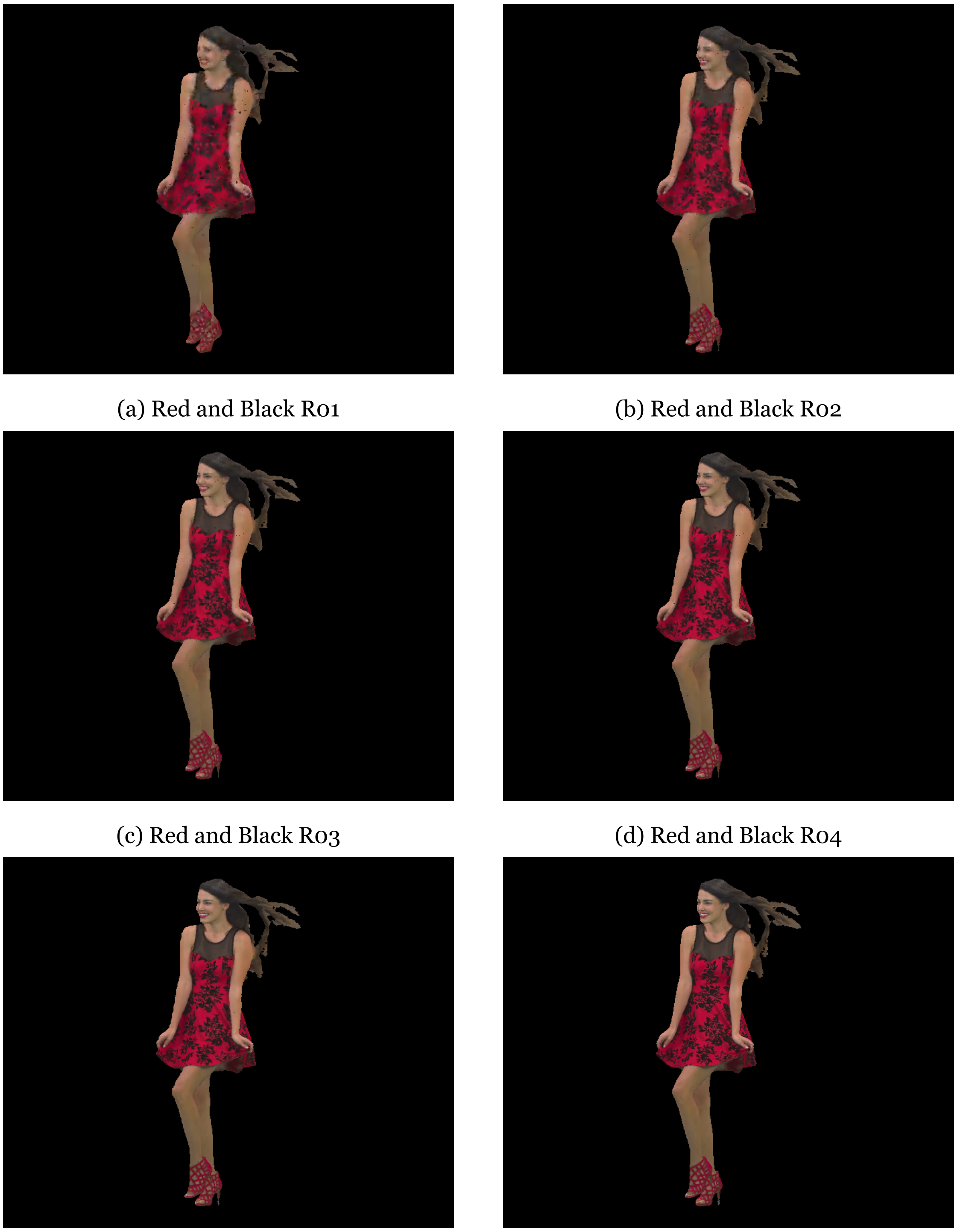

(f) Loot R06

Figure 27: Final coding of Loot with G-PCC (TRISOUP)



(a) Red and Black R01

(b) Red and Black R02

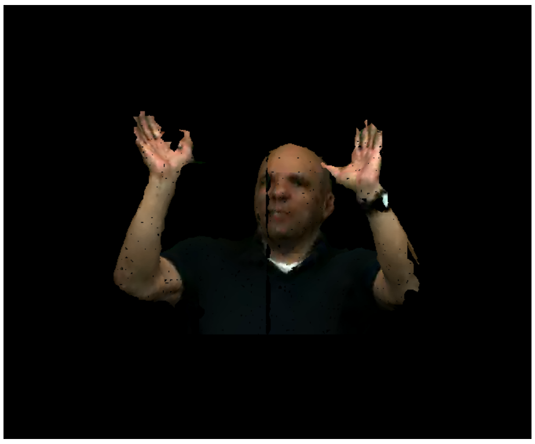
(c) Red and Black R03

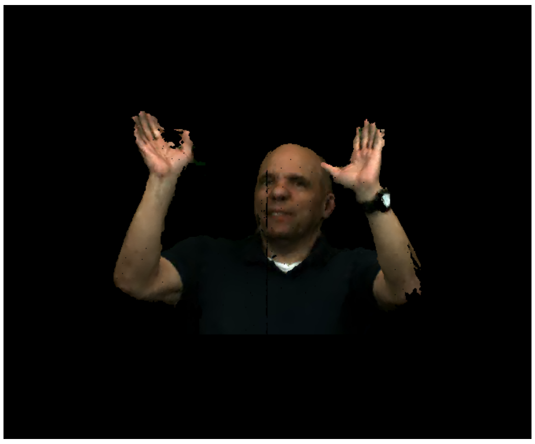
(d) Red and Black R04

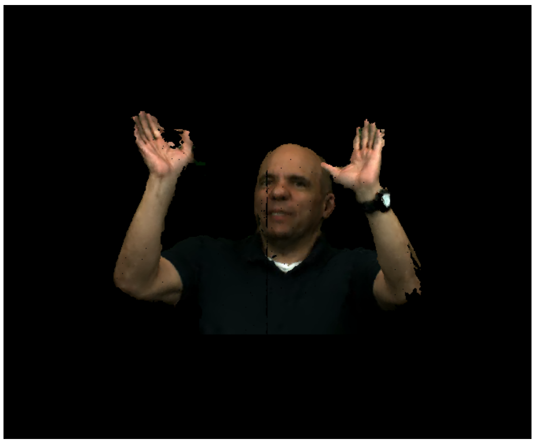
(e) Red and Black R05

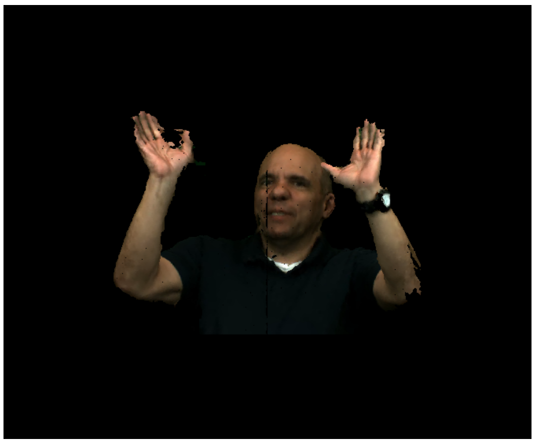
(f) Red and Black R06

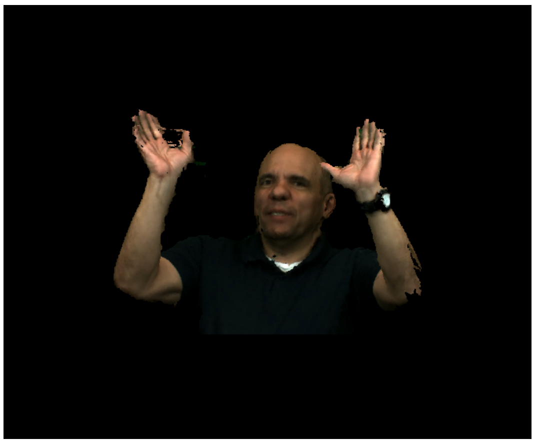
Figure 28: Final coding of Red and Black with G-PCC (TRISOUP)


(a) Ricardo10 R01


(b) Ricardo10 R02


(c) Ricardo10 R03


(d) Ricardo10 R04


(e) Ricardo10 R05

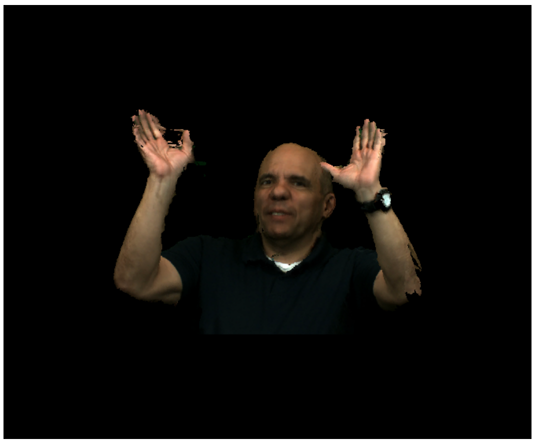
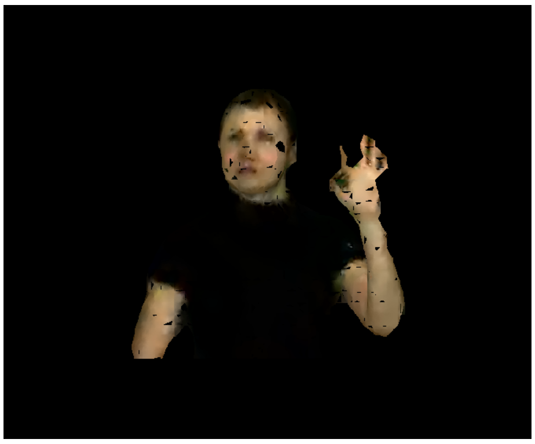
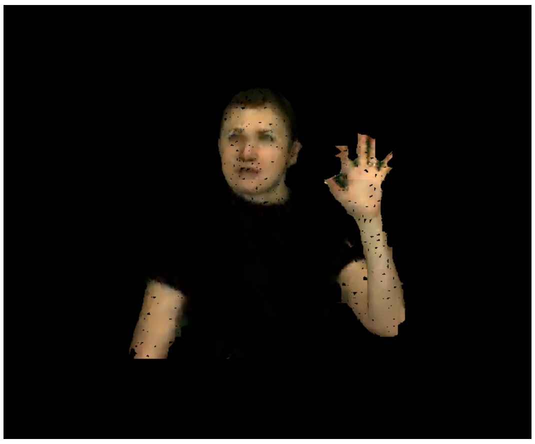

(f) Ricardo10 R06

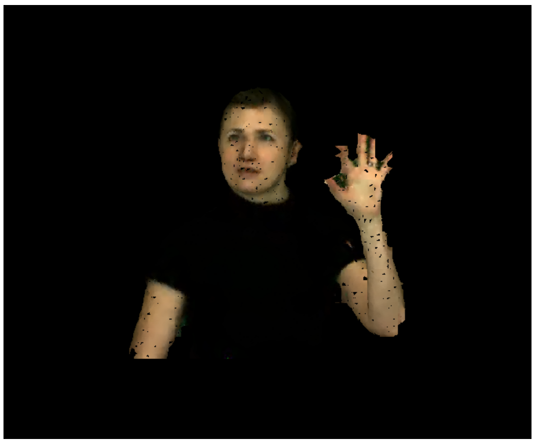
Figure 29: Final coding of Ricardo10 with G-PCC (TRISOUP)



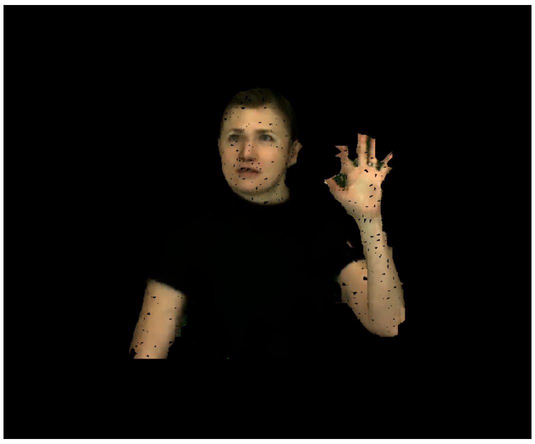
(a) Sarah9 R01



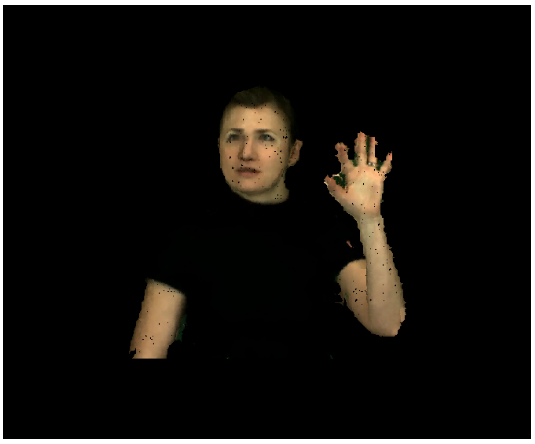
(b) Sarah9 R02



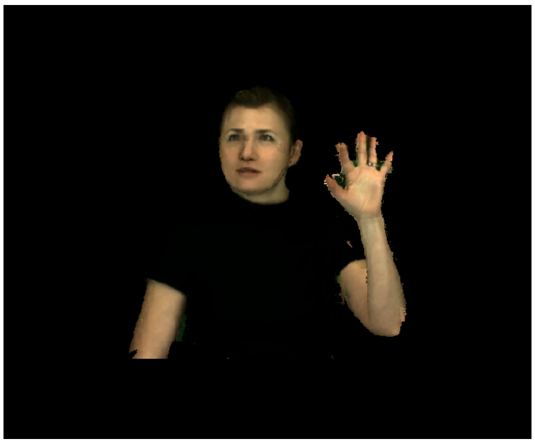
(c) Sarah9 R03



(d) Sarah9 R04



(e) Sarah9 R05



(f) Sarah9 R06

Figure 30: Final coding of Sarah9 with G-PCC (TRISOUP)

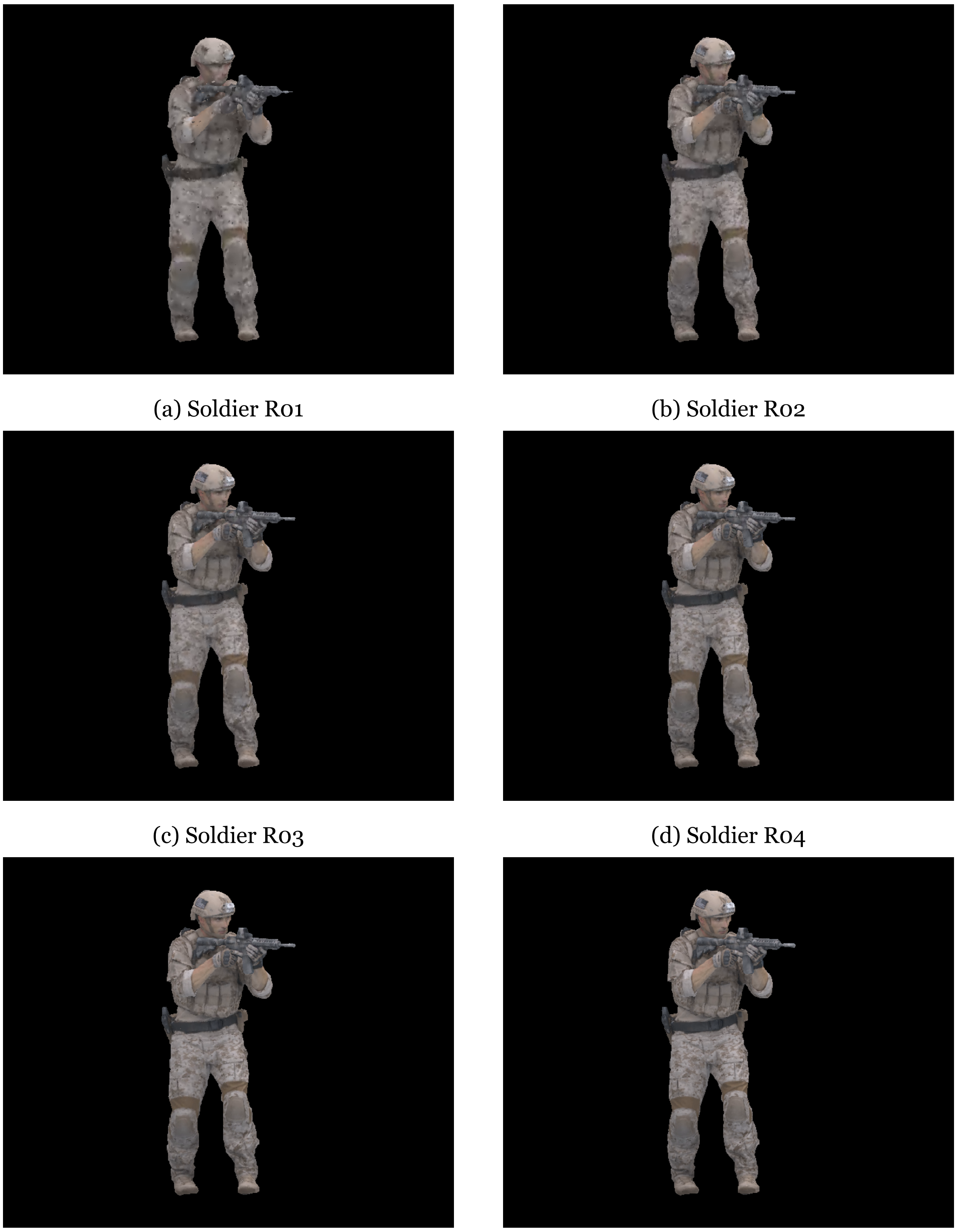

(a) Soldier R01 (b) Soldier R02 (c) Soldier R03 (d) Soldier R04 (e) Soldier R05 (f) Soldier R06

Figure 31: Final coding of Soldier with G-PCC (TRISOUP)

10.3 Preparing the videos

The videos were prepared using the software cloud compare. The purpose of the the evaluation was to test the quality of the codecs, when displayed in 2D and 3D Stereo. Due to the Covid-19 Pandemic, the latter could not be done, but the preparations for the test were made, and thus are described.

2D and 3D

The point cloud is loaded into cloud compare software, and then viewpoints are selected to create an animation, which were dependent on the type of content. The point clouds used in the study can be categorized as:

- Full Bodies
- Upper Bodies

The viewpoints for the full bodies were:

1. X=0, Y=0, Z=0
2. X=0, Y=-90, Z=0
3. X=0. Y=-180=, Z=0
4. X=0, Y=90, Z=0
5. X=0, Y=0, Z=0

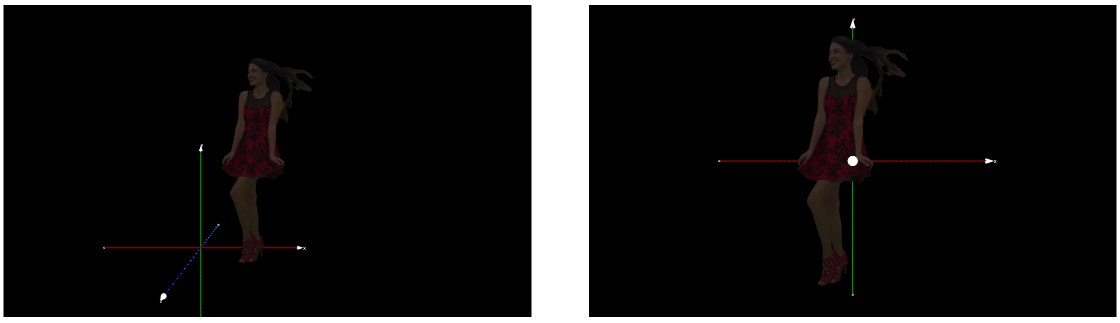
This creates a clockwise rotation around the Y axis. The viewpoints for the full bodies were:

1. X=0, Y=-45, Z=0
2. X=0, Y=-0, Z=0
3. X=0. Y=45=, Z=0
4. X=0, Y=0, Z=0
5. X=0, Y=-45, Z=0

The point cloud rotates 45 degrees around the Y axis The frames are exported at 30fps with a 12 second duration, and a 6200kps bitrate. A video for the original and distorted content is made, and then merged using FFmpeg software of MATLAB. For the creation of the 3D Stereo, four videos needed to be recorded, at $\frac{1}{4}$ of the screen resolution, two for the orginal content and other two for the distorted content. Then two of the videos needed to be translated, in order to create a 3D Stereo image. Then the videos were merged by following the order of (Original centered video,Distorted centered video,Original translated video, Distorted translated video). In an embryonic stage, Unity software was used to show the point clouds, and to create 3D video. The software excelled at representing the content, but in video creation, when the images overlapped, the video quality was dubious, and so the option of using unity to create 3D stereo video was discarded.

10.4 Creating 3D videos with unity

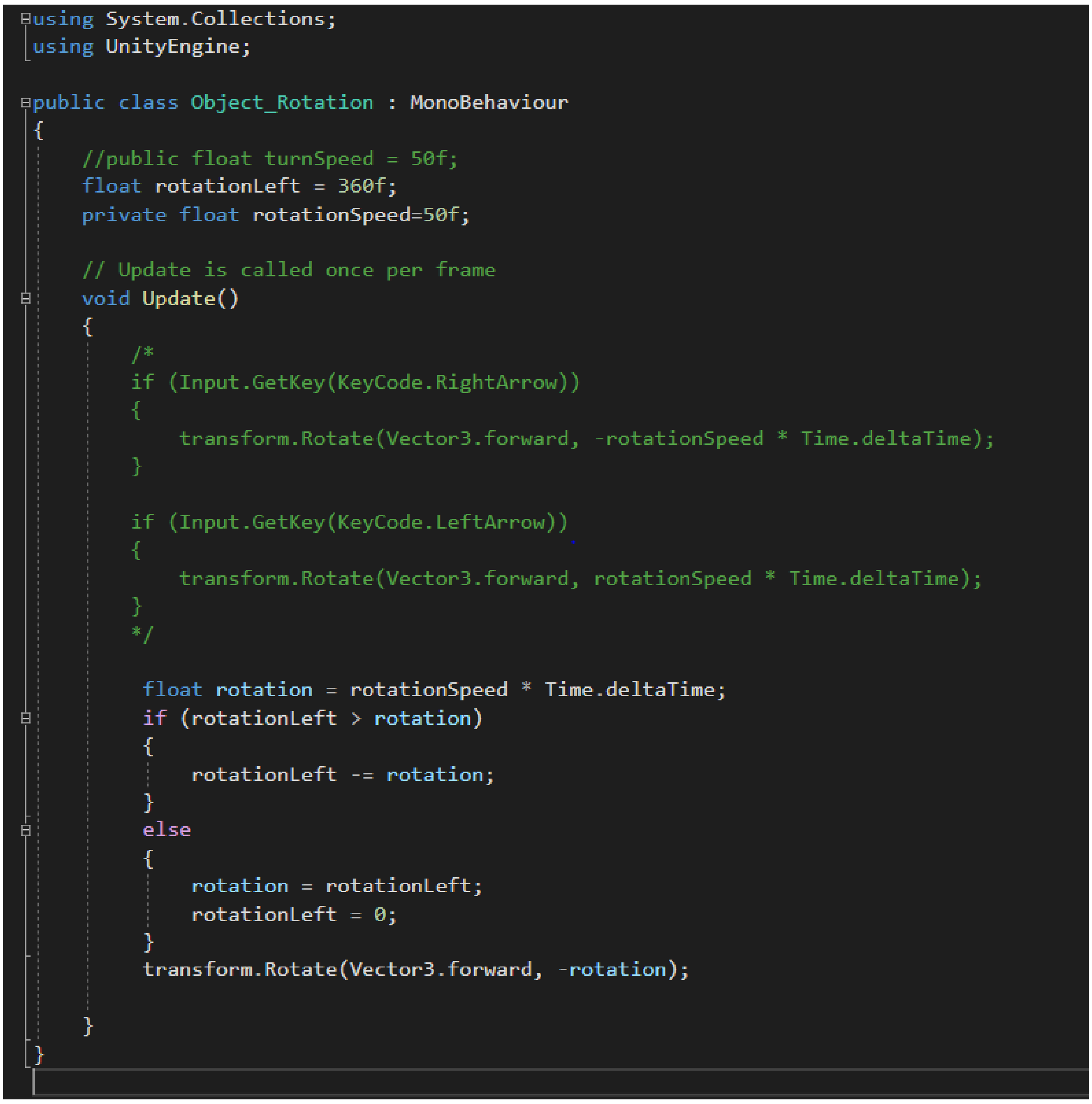
To create videos using unity, first the point cloud needs to be imported to the software. After importing the point cloud, it is necessary to create a code for moving the point cloud as we wish. Since the videos for the subjective testings involve the rotation of content, a small code was created to make the point cloud rotate around its axis. The point cloud should be checked for its axis. Using Meshlab Software, the axis can easily be visualized, and if the point cloud its not centered, by selecting the option *Filters→ Normals, Curvatures and Orientation, → Transform: Translate, Rotate, Set Origin*, the point cloud can be easily centered, and ready to be imported to unity.



(a) no (b) yes

Figure 32: Centering the axis with MeshLab

After centering the point cloud, a code to make the point cloud rotate needs to be created. The commented section of the code allows to control the point cloud rotation direction using the right and left arrows, and was developed mainly for testing. The non commented section of the code makes the point cloud rotate automatically.



```
using System.Collections;
using UnityEngine;

public class Object_Rotation : MonoBehaviour
{
    //public float turnSpeed = 50f;
    float rotationLeft = 360f;
    private float rotationSpeed=50f;

    // Update is called once per frame
    void Update()
    {
        /*
        if (Input.GetKey(KeyCode.RightArrow))
        {
            transform.Rotate(Vector3.forward, -rotationSpeed * Time.deltaTime);
        }

        if (Input.GetKey(KeyCode.LeftArrow))
        {
            transform.Rotate(Vector3.forward, rotationSpeed * Time.deltaTime);
        }
        */

         float rotation = rotationSpeed * Time.deltaTime;
         if (rotationLeft > rotation)
         {
             rotationLeft -= rotation;
         }
         else
         {
             rotation = rotationLeft;
             rotationLeft = 0;
         }
         transform.Rotate(Vector3.forward, -rotation);

    }
}
```

Figure 33: Code for point cloud rotation in unity

After importing the point cloud and creating the script, it needs to be attached to the point cloud, and the point cloud should be positioned in the Unity scene world. After positioning the point cloud, a camera was created to record the point cloud. It was positioned directly in front of the point cloud, in a way such as the point cloud is aligned and centered with the camera. After positioning, four videos were recorded for each content, in the order described bellow.

1. Original content centered
2. Distorted content centered
3. Offset original content

4. Offset distorted content

Due to the nature of 3D recordings, the content needed to be re-scaled in half, in the X and Z axis. This can be made with Cloud Compare, and the final point cloud should look like:

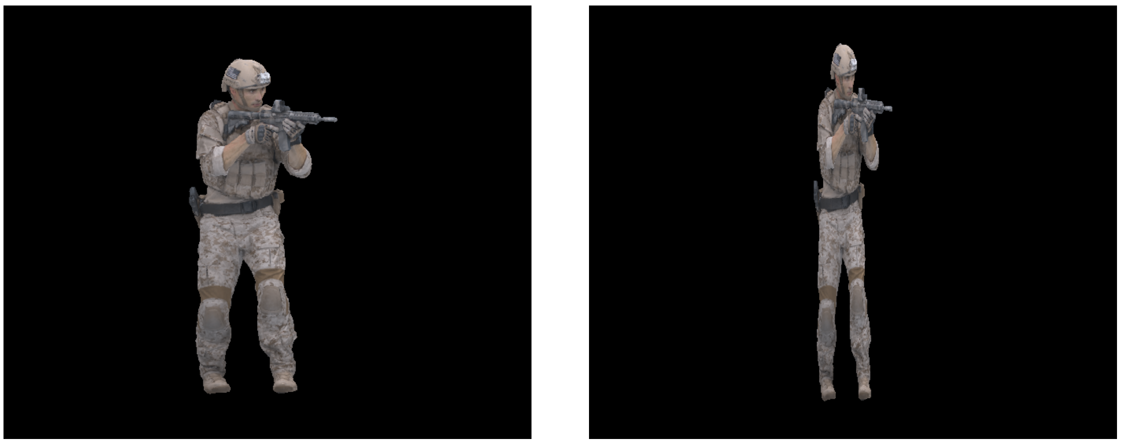

(a) Original Content (b) Re-scaled

Figure 34: Re-scaled content using Cloud Compare

Finally, Unity Recorder is used to record a 12 seconds video, at 30 fps, with a 6200kbps rate. The videos needed to be recorded at a 420×1080 resolution, given that the 3D display present in the university laboratory has a 1920×1080 resolution. After recording four videos in the above order, the final result should look like this:

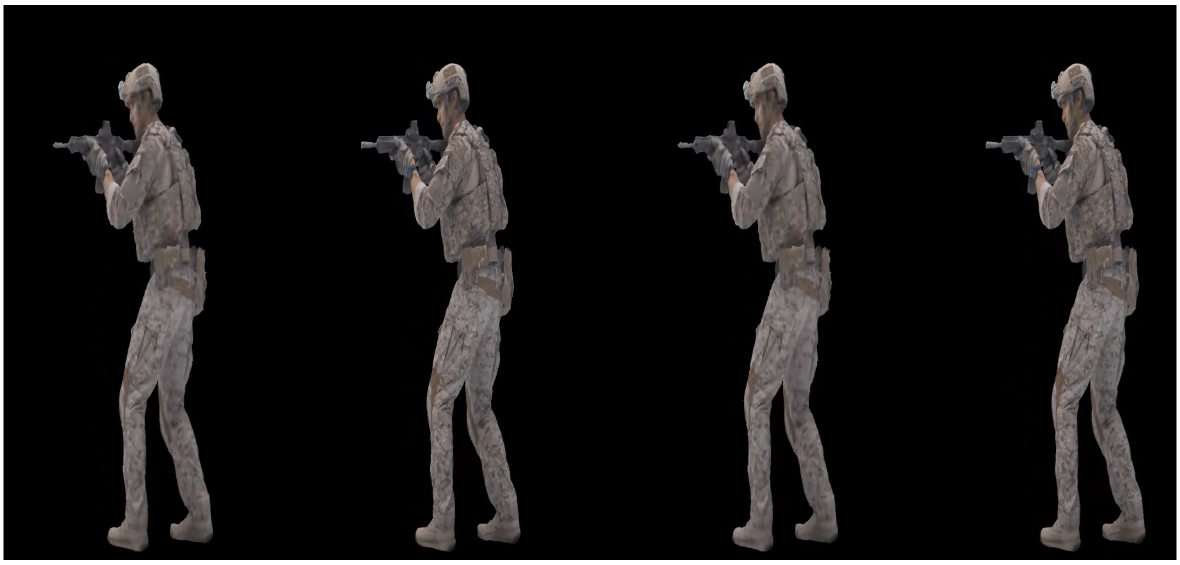


Figure 35: Frame from a 3D video to be used in subjective testings

10.5 Subjective Testing

For the test, and as a part of [25], evaluations were conducted across four universities: Universidade da Beira Interior (UBI), Universidade de Coimbra (UC), University North (UNIN) and University of Technology Sydney (UTS), with ages spanning from 19 years old to 59 years old. The following table shows the data relative to the test subjects.

	Males	Females	Overall	Age span	average Age
UBI	7	9	16	19-32	22
UC	7	8	15	18-54	28
UNIN	10	5	15	19-59	29
UTS	21	6	27	21-47	32

Table 11: Subjects Information

The subjects were shown a stimuli of twelve seconds, where the preparation for those stimuli was described above. After a twelve seconds stimuli, the subjects were asked to evaluate the differences between the original content and the distorted content, rating them from 1-5(1-very annoying, 2-annoying, 3-slightly annoying, 4-perceptible but not annoying, 5-imperceptible). All the tests were made in one session, and the tests were conducted in an Eizo ColorEdge CG318-4K, with 31.1 inches and a 4069×2160 resolution(UBI, UTS), a Monitor Sony KD-49X8005C, with 49 inches and a resoltion of $3840 \times 2160(UC)$, a Sony TV KD-55x8505C with 55 inches and a resolution of 3840×2160 (UNIN). The subjects were 1.2m apart from the screen (UBI,UTS), 1.8m apart (UC) and 1.5m apart (UNIN). Prior to the testing, a training session took place, where the subjects were shown a stimuli which would not be included in the test. To avoid biases, half of the subjects were shown stimuli with the original content on the right side, and the other half, with the original content on the left side. The following images show the correlation between the labs, and a comparison between MOS and content bitrate.

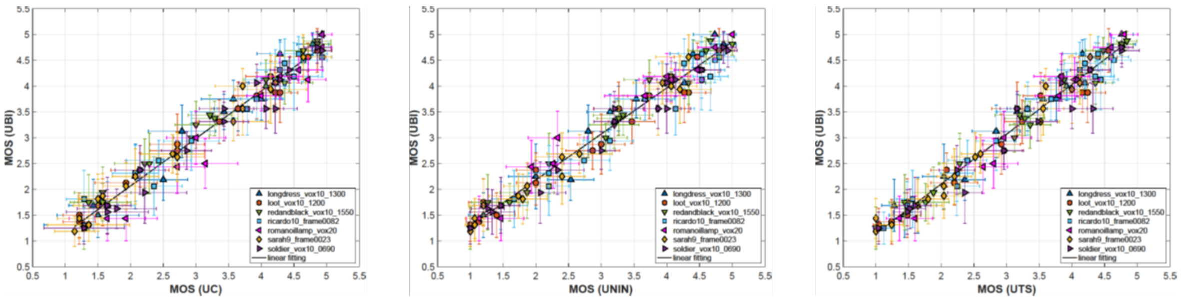


Figure 36: Correlation Between labs 1

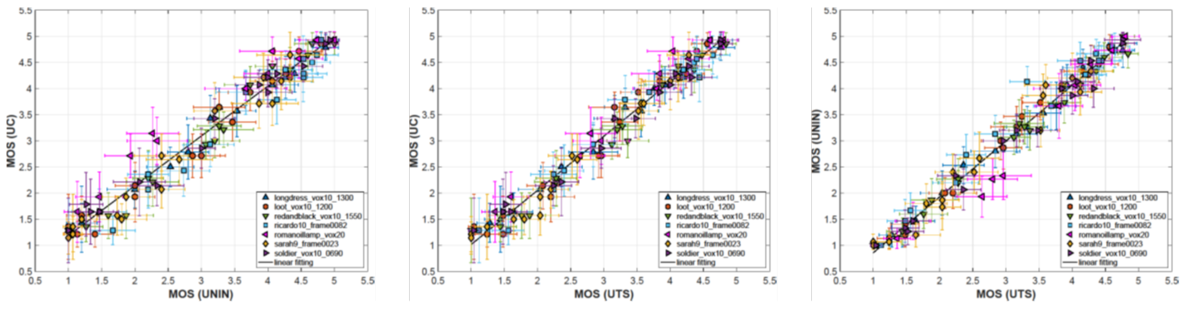
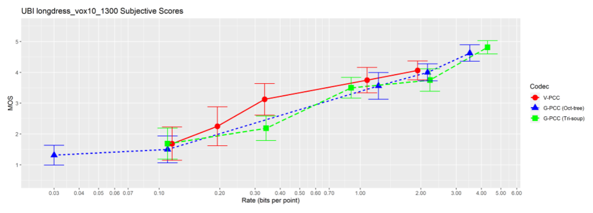
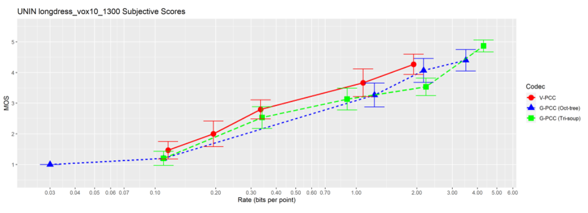


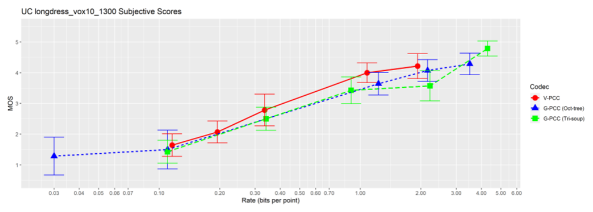
Figure 37: Correlation Between labs 2



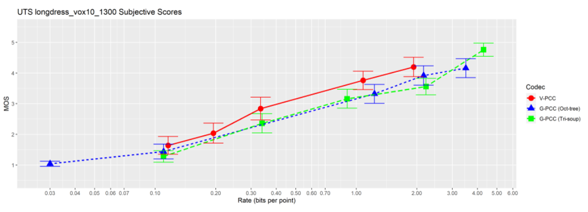
(a) UBI



(b) UN

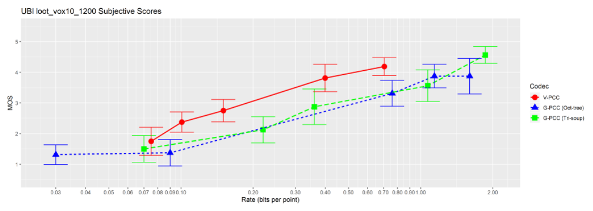


(c) UC

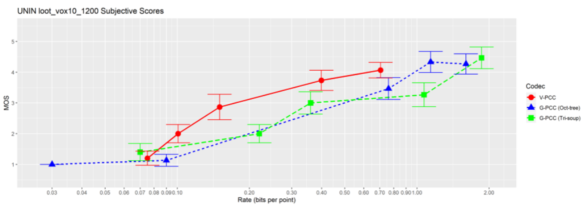


(d) UTS

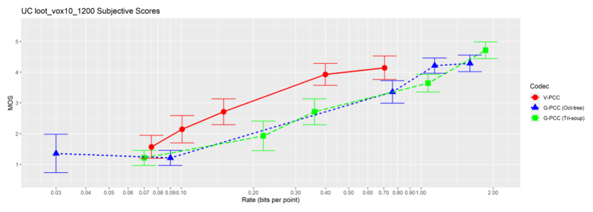
Figure 38: Longdress: All labs



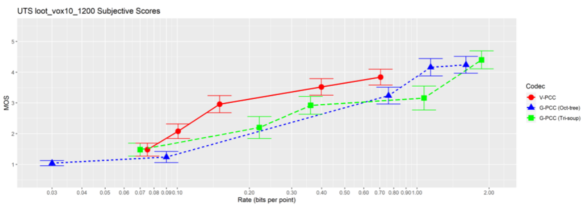
(a) UBI



(b) UN

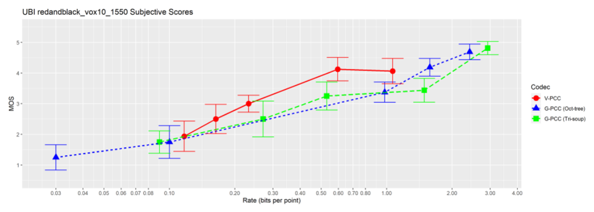


(c) UC

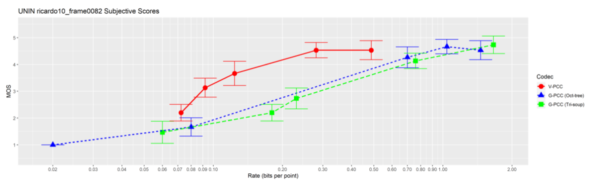


(d) UTS

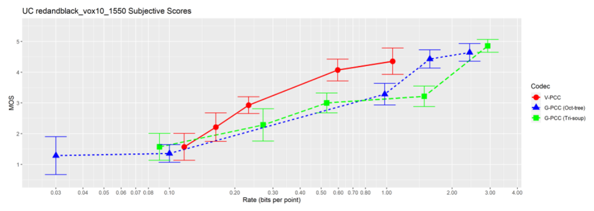
Figure 39: Loot: All labs



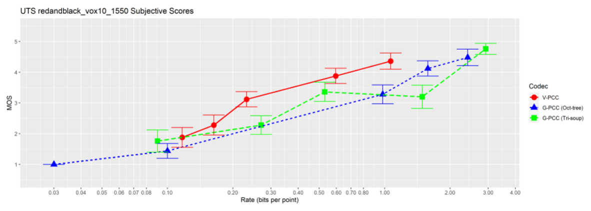
(a) UBI



(b) UN

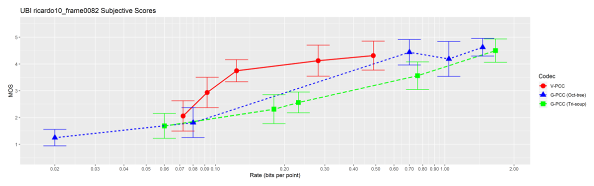


(c) UC

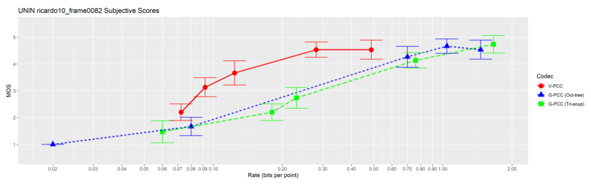


(d) UTS

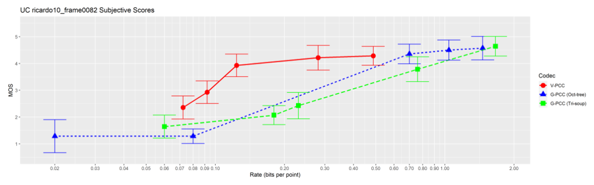
Figure 40: Red And Black: All labs



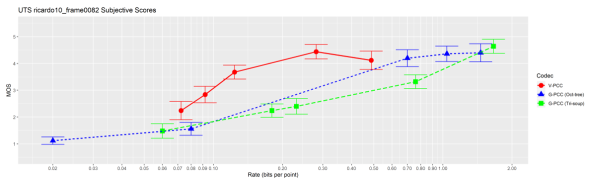
(a) UBI



(b) UN

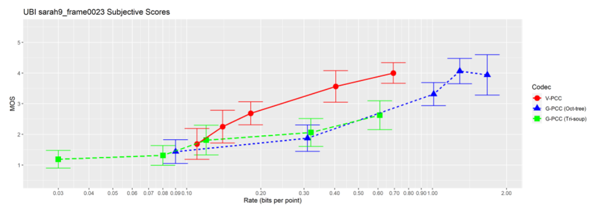


(c) UC

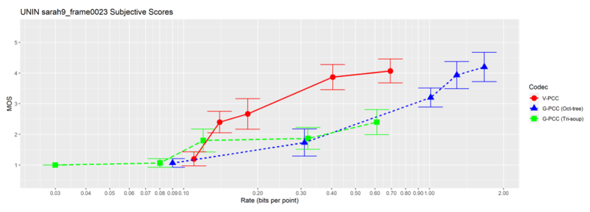


(d) UTS

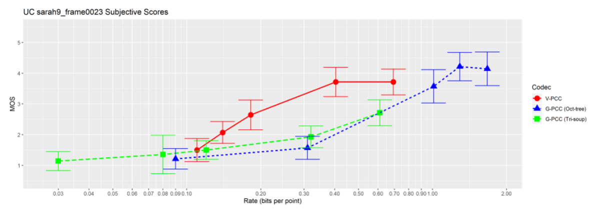
Figure 41: Ricardo10: All labs



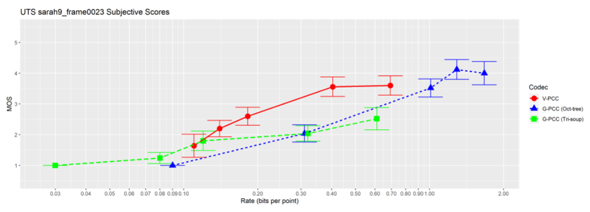
(a) UBI



(b) UN

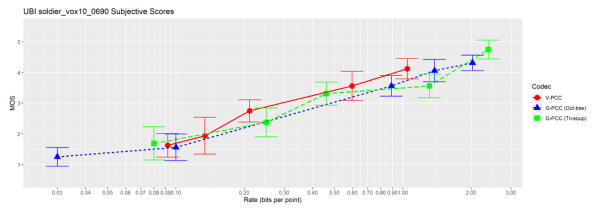


(c) UC

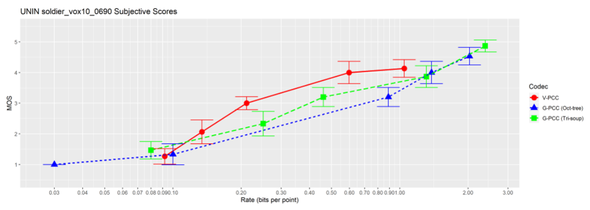


(d) UTS

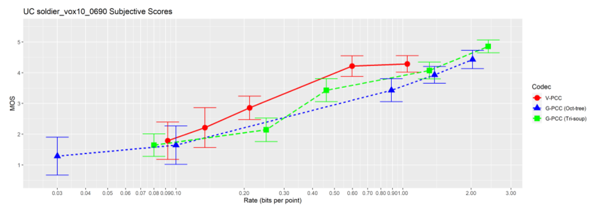
Figure 42: Sarah9: All labs



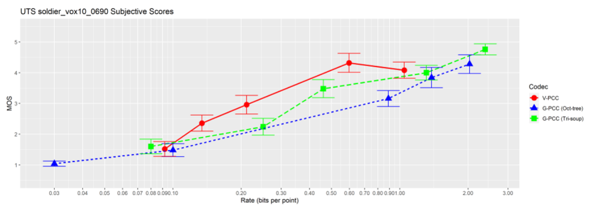
(a) UBI



(b) UN



(c) UC

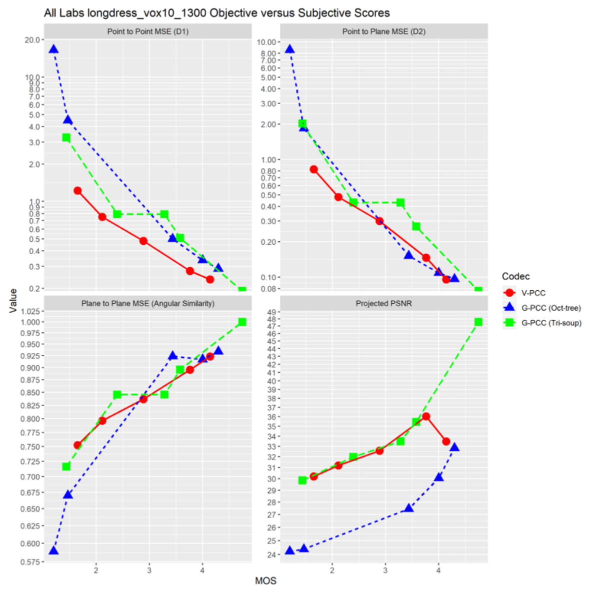


(d) UTS

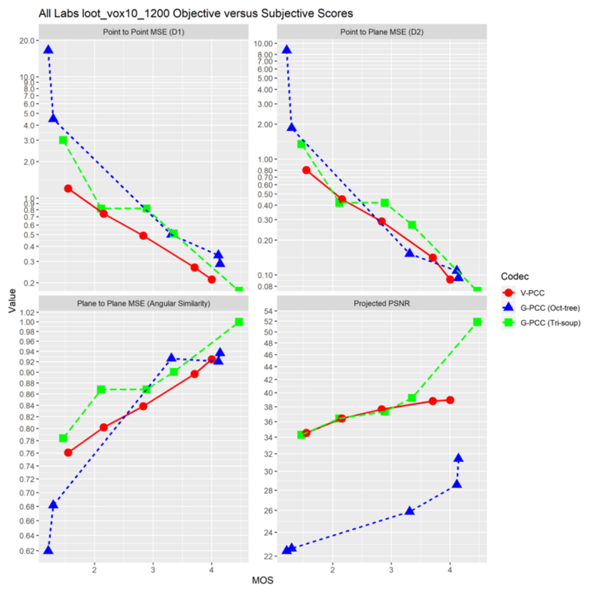
Figure 43: Soldier: All labs

10.6 Objective Testings

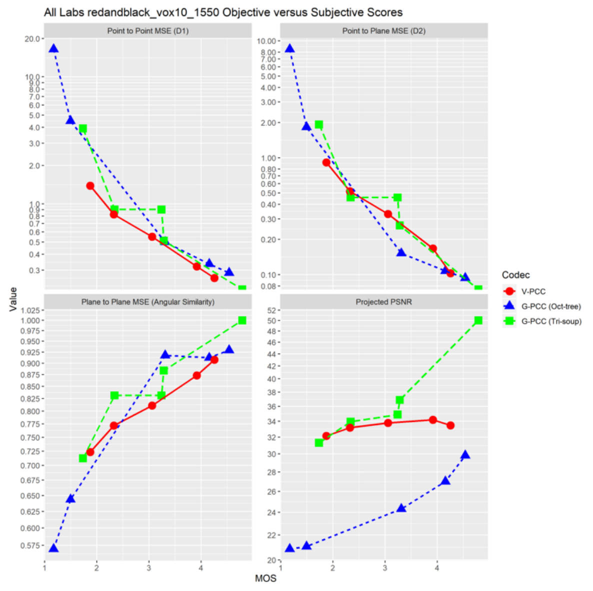
To estimate geometric distortions, point-to-point (po2po) and point-to-plane (po2plane) metrics were used, using Mean Squared Error and Haussdorf distance as a error measure. the geometry PSNR (Peak Signal Noise Ratio) is computed as well for the two considered distances. For each stimuli, the normal vectors of each content was estimated, using the Cloud Compare software. The following images show the obtained metrics of all labs.



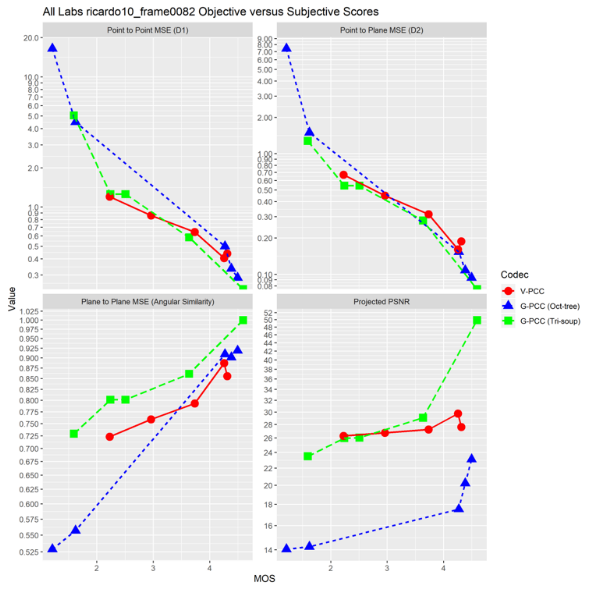
(a) Longdress



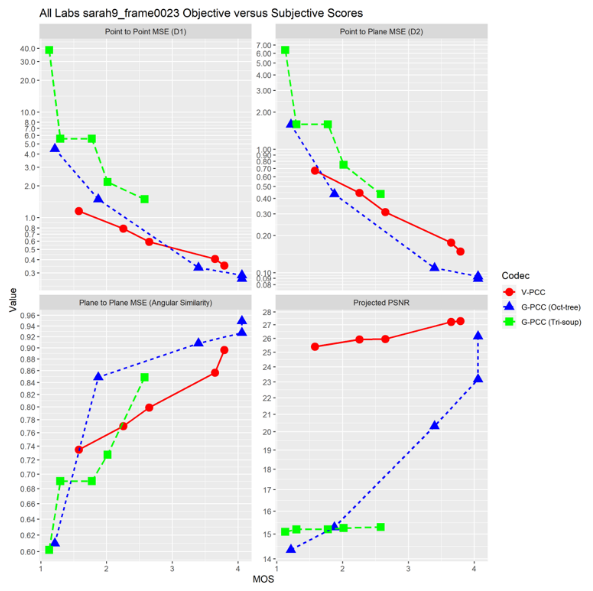
(b) Loot



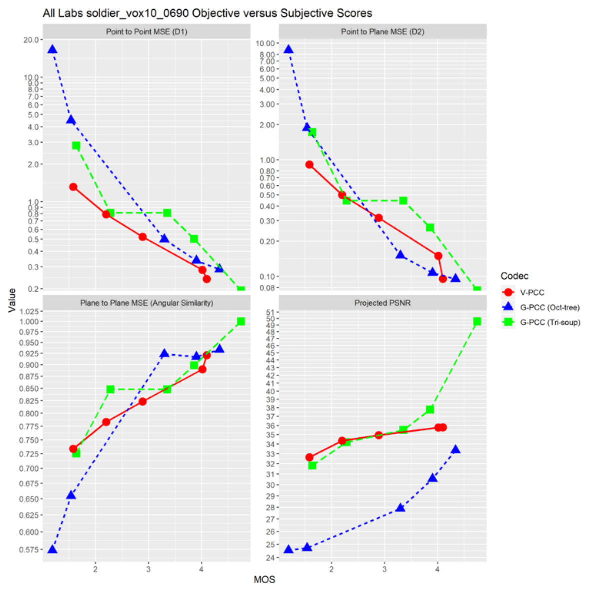
(c) Red And Black



(d) Ricardo10



(e) Sarah9



(f) Soldier

10.7 Result Analysis

10.7.1 Compression efficiency

The subjective testing conducted revealed that V-PCC is more efficient than G-PCC in the compression of the tested static point clouds[25]. This conclusion results from the subjective testing described in figures 38 to 43. Using projections of the initial point cloud and considering the maturity of image compression technology it is possible to overtake the typical octree based methods for point cloud compression. V-PCC is however a high complexity method that requires efficient computation and reduces the application to point clouds that are somehow composed by a reduced number of points.

10.7.2 Objective metrics representation

For an analysis of the metrics performance, the correlation methods proposed in [26], more specifically, Pearson Correlation Coefficient (PCC), Spearman-Ranked-Order-Correlation-Coefficient(SROCC), Absolute Prediction Error (RMSE) and Outlier Ratio (OR) were used on pairs of MOS and predicted MOS computed from the respective metric. Form table 12 it is possible to observe that the metrics that provide the best representation of the subjective results are po2plane_MSE (represented in boldface) followed by po2point_MSE.

Table 12: Metrics Table

Metric	PCC	SROCC	RMSE	OR
po2point_MSE	0.946	0.934	0.368	0.666
po2plane_MSE	**0.959**	**0.951**	**0.321**	**0.544**
point2point_MSE_PSRN	0.868	0.855	0.540	0.752
point2point_MSE_PSRN	0.913	0.910	0.443	0.588
po2point_HAU	0.401	0.531	1.045	0.844
po2plane_HAU	0.534	0.613	0.966	0.877
po2point_HAU_PSRN	0.548	0.456	0.911	0.870
po2plane_HAU_PSNR	0.580	0.547	0.887	0.847
Color_Y_MSE	0.876	0.892	0.551	0.766
Color_Cb_MSE	0.683	0.694	0.834	0.844
Color_CR_MSE	0.594	0.616	0.918	0.844
Color_Y_PSNR	0.887	0.892	0.525	0.688
Color_Cb_PSNR	0.693	0.694	0.822	0.844
Color_CR_PSNR	0.626	0.617	0.890	0.855
pl2plane_AVG	0.922	0.910	0.439	0.600
pl2plane_RMS	0.925	0.912	0.432	0.622
pl2plane_MSE	0.925	0.912	0.432	0.611

11 Conclusions and Future Work

Apart concluding on the V-PCC superiority over G-PCC, this work led to several important facts:

1) Some objective metrics provide a reliable representation of the perceptual quality. Several metrics had correlations above 0.9. In particular po2plane_MSE overpass correlation values of 0.95, revealing a very reliable representation.

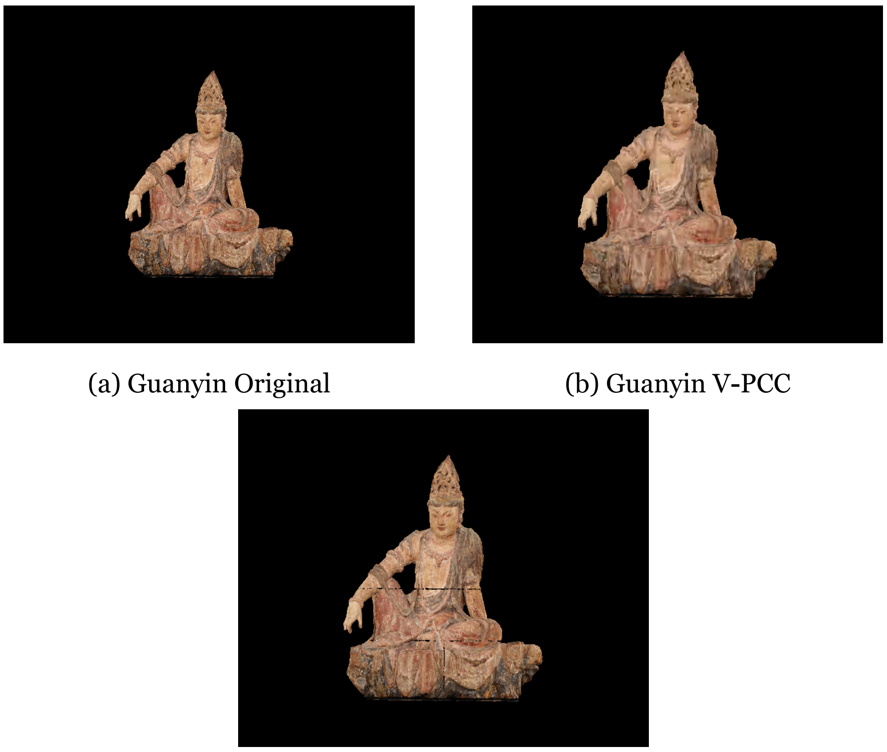
2) This work allowed to improve the knowledge on MPEG Point cloud codecs that probably are the best current codecs for point clouds. Installing them and understanding their parametrization is not straightforward.

3) This work also allowed to get familiarized with the current point cloud technology and applications, mostly based on cloudcompare package and unity. The work reveled that, while Unity is a great software to represent point clouds in a 3D environment, and a good candidate for testing in a virtual reality environment, the recorded videos with the Unity camera showed

an inferior quality than the videos made with Cloud Compare. This is most likely due to some kind of post processing made by Unity software, whereas the Cloud Compare software allows the extraction of each individual frame, and then the video is created using MatLab, by putting together all the frames.

This work will be the basis of the future work on point cloud compression, representation and quality evaluation.

Subjective testings representing the point clouds using a 3D stereo representation and virtual reality equipment were planned, but due to the ongoing COVID-19 pandemic, the conduction of those tests was not possible, as it was impractical to bring test subjects to the laboratory. Additional point clouds were also to be tested. Some examples are shown in the following figures.



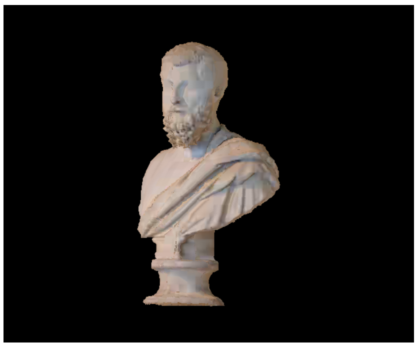
(a) Guanyin Original

(b) Guanyin V-PCC

(c) Guanyin G-PCC

Figure 45: Guanyin Point Cloud

(a) Rhetorician Original


(b) Rhetorician V-PCC

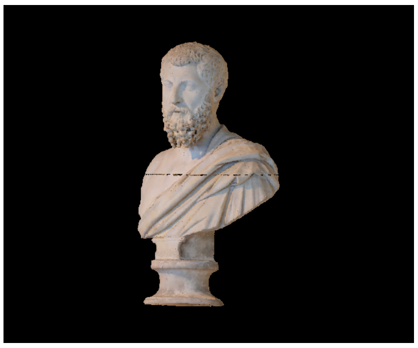
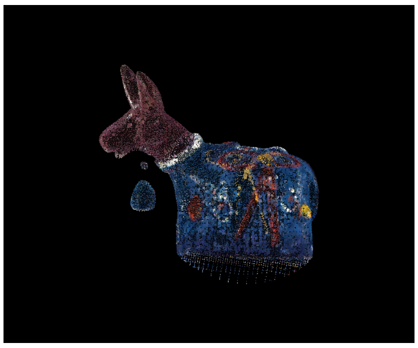
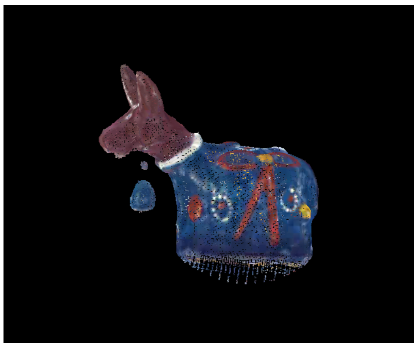

(c) Rhetorician G-PCC

Figure 46: Rhetorician Point Cloud


(a) Bumbameuboi Original


(b) Bumbameuboi V-PCC

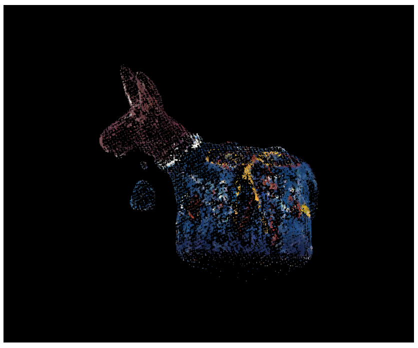
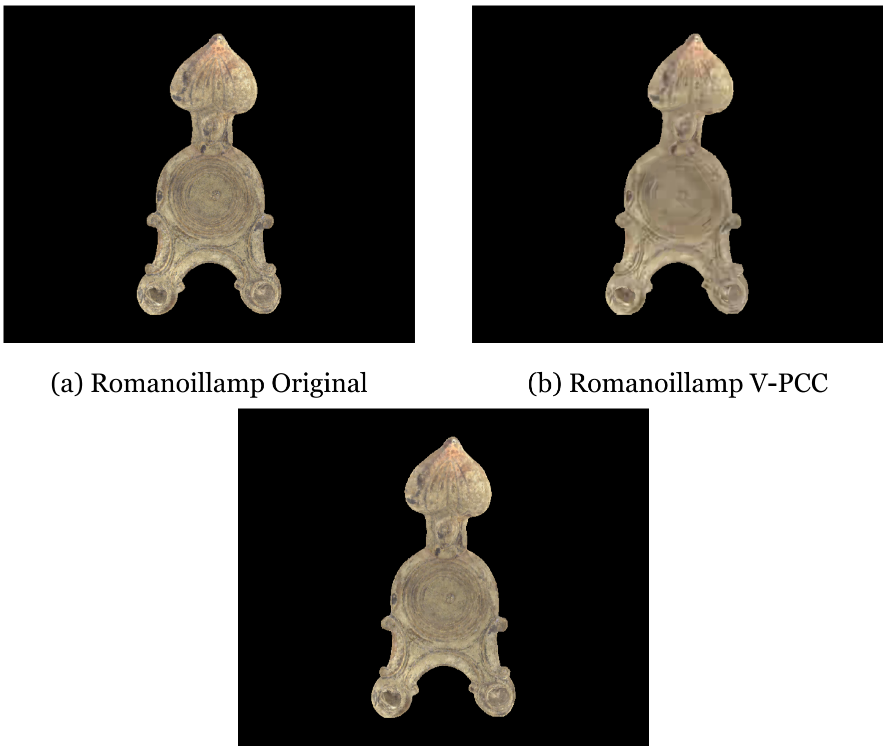

(c) Bumbameuboi G-PCC

Figure 47: Bumbameuboi Point Cloud


(a) Romanoillamp Original (b) Romanoillamp V-PCC

(c) Romanoillamp G-PCC

Figure 48: Romanoillamp Point Cloud

Due to a wide range of point cloud applications in robotics, the integrity of point clouds codec using different codecs also need to be tested, to verify if the decision of systems that use point clouds has an input, take the same decisions with non-coded point clouds or coded point clouds. Moreover, other codecs will be tested in the near future. Recently codecs based on machine learning have been tested for point clouds and a complementary study on the quality domain, is also needed.

Bibliography

[1] J Geng. Structured-light 3d surface imaging: a tutorial. *Adv. Opt. Photon.*, pages 128–160, Jun 2011.

[2] K Waters. Lidar – accuracy versus resolution.

[3] M Tully. Just how accurate is lidar.

[4] Lidar uk. (2017). how does lidar work?

[5] Australian uav. (2017). drone data vs lidar: Clarifying misconceptions.

[6] Helicopter flight test of 3d imaging flash lidar technology for safe, autonomous, and precise planetary landing.

[7] A multiple-baseline stereo. ieee transactions on pattern analysis and machine intelligence. pages 53–363, April 1993.

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, November 2004.

[9] Lytro illum 40 megaray light field camera.

[10] G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu. Light field image processing: An overview. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):926–954, 2017.

[11] S Perry. Jpeg pleno point cloud – use cases and requirements v1.1.

[12] J Peng and Kuo C.-C. J. Geometry-guided progressive lossless 3d mesh codingwith octree (ot) decomposition. pages 609–616, July 2005.

[13] M. Bennewitz C. Stachniss A. Hornung, K. M. Wurm and W. Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 3(6):688–733, December 2005.

[14] Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. 3d mesh compression: Survey, comparisons, and emerging trends. 04 2015.

[15] P Alliez and C Gotsman. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[16] Jingliang Peng, Chang-Su Kim, and C.-C. Jay Kuo. Technologies for 3d mesh compression: A survey. *Journal of Visual Communication and Image Representation*, 16:688–733, 12 2005.

[17] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Gaël Guennebaud, Joshua Levine, Andrei Sharf, and Cláudio Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 03 2016.

[18] R. Wang, J. Peethambaran, and D. Chen. Lidar point clouds to 3-d urban models: a review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(2):606–627, 2018.

[19] V Zakharchenko. “algorithm description of mpeg-pcc-tmc2”. *ISO/IEC JTC1/SC29/WG11 MPEG2018/N17767*, Jul 2018.

[20] D. Flynn K. Mammou, P. A. Chou and M. Krivokuća. G-pcc codec description v2. *ISO/IEC JTC1/SC29/WG11 N18189*, Jan 2019.

[21] ITU-R BT.500-13. ”methodology for the subjective assessment of the quality of television pictures”. *International Telecommunication Union/ITU Radiocommunication Sector*, January 2012.

[22] ITU-R BT.2021. ”subjective methods for the assessment of stereoscopic 3dtv systems”. *International Telecommunication Union/ITU Radiocommunication Sector*, Feb 2015.

[23] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3460–3464, 2017.

[24] E. Alexiou and T. Ebrahimi. Point cloud quality assessment metric based on angular similarity. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2018.

[25] S Perry, H.P Cong, L.A Cruz, J Prazeres, M Pereira, A Pinheiro, E Dumic, E Alexiou, and T Ebrahimi. Quality evaluation of static point clouds encoded using mpeg codecs. In *International Conference on Image Processing (ICIP 2020)*, 2020.

[26] ITU-T P.1401. “methods, metrics and procedures for statistical evaluation, qualification and comparison of objective quality prediction models”. *International Telecommunication Union*, 2012.