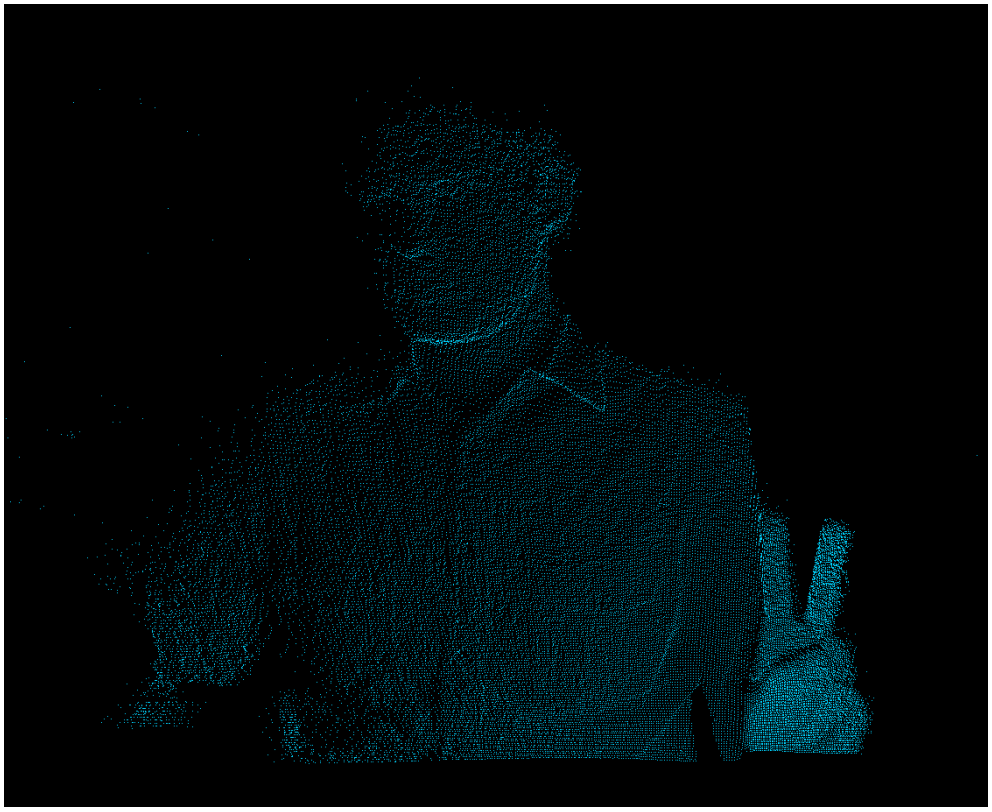
Inter frame compression for 3D dynamic point clouds

Version of October 27, 2017



Shishir Subramanyam

Inter frame compression for 3D dynamic point clouds

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Shishir Subramanyam
born in Bangalore, India

Multimedia Computing Group
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Centrum Wiskunde & Informatica
Distributed and Interactive Systems
Amsterdam, the Netherlands
www.dis.cwi.nl

Inter frame compression for 3D dynamic point clouds

Author: Shishir Subramanyam
Student id: 4512073
Email: s.subramanyam@student.tudelft.nl

Abstract

In recent years Virtual Reality (VR) and Augmented Reality (AR) applications have seen a drastic increase in commercial popularity. Different representations have been used to create 3D reconstructions for AR and VR. Point clouds are one such representation that are characterized by their simplicity and versatility making them suitable for real time applications. However point clouds are unorganized and identifying redundancies to use for compressing them is challenging. For the compression of time varying or dynamic sequences it is critical to identify temporal redundancies that can be used to describe predictors and further compress streams of point clouds.

Most of the previous research into point cloud compression relies on the octree data structure. However this approach was used on relatively sparse datasets. Recently, new dense photorealistic point cloud datasets have become available with the ongoing standardization activities on point cloud compression. To compress them using existing octree based codecs is computationally expensive as the tree depth required to achieve a reasonable level of detail is much higher than what was used previously. We propose a point cloud codec that terminates the octree at a fixed level of detail and encodes additional information in an enhancement layer. We also add inter prediction to the enhancement layer in order to gain further bit rate savings. We validate our codec by evaluating it in the framework set up by standardization organizations such as MPEG. We then demonstrate an improvement over the current MPEG anchor codec.

Thesis Committee:

Chair:	Prof. Dr. Alan Hanjalic, Faculty EEMCS, TU Delft
University supervisor:	Dr. Pablo Cesar, DIS/Faculty EEMCS, CWI/TU Delft
Committee Member:	Prof. Dr. Elmar Eisemann, Faculty EEMCS, TU Delft

Preface

First and foremost, I would like to thank my supervisor Dr. Pablo Cesar. This thesis would not have been possible without his guidance, support and patience. I would also like to thank Rufael and Kees from the Distributed and Interactive Systems group at CWI. Our long and insightful discussions provided me with a better understanding of the thesis and the anchor framework. I am also grateful for the valuable feedback they provided on my writing. These discussions were an immense source of knowledge and provided the background for writing this thesis.

I would like to extend my gratitude to my colleagues at CWI: Marwin, Tom, Britta, Jan Willem and others for their help, our discussions and the much needed coffee breaks.

I would also like to extend my gratitude to my friends at TU Delft. I am especially grateful to Rimit, Vishnu, Chinmay and Srikar for making my stay in Delft feel like home. I would also like to thank my friends from the software architecture project for the fun times.

Lastly, I would like to thank my parents and my brother back in India. Their support was instrumental in keeping me going through challenging times.

The end of this thesis marks the end of my Master's at TU Delft. This thesis was the most challenging and rewarding experience of my journey as a student here. Looking back, I am reminded of a quote by Rabindranath Tagore "You can't cross the sea merely by standing and staring at the water".

Shishir Subramanyam
Delft, the Netherlands
October 27, 2017

Contents

List of Figures

Chapter 1

Introduction

Virtual and augmented reality technologies have been in existence since the 1960s when Ivan Sutherland first proposed the hypothetical ultimate display [37]. He provided an early description of immersive displays that stated: "The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked". In 1968 he created a head mounted display called the "Sword of Damocles" that was suspended from the ceiling and was capable of head tracking and see through optics. In subsequent years VR and AR had limited use and were mostly restricted to military and some medical applications. This was typically done by overlaying virtual information on the physical world or in simulations for aviation and military purposes. It was not until the last decade that VR and AR gained mainstream commercial popularity.

The increased availability of affordable sensors, increased computational power of commodity hardware and the miniaturization of electronics have made AR and VR viable and more usable. In order to deliver immersive AR experiences it is necessary to generate highly photorealistic reconstructions of a natural scene or object. Recent advances in depth sensors have made it affordable to generate such reconstructions in real time [16]. New efficient representations for 3D content have enabled interactive experiences to be shared by multiple users and have enabled meaningful realistic interactions of the virtual and physical world. In addition companies have been able to generate meaningful content for commercial applications that add unique value beyond what other technologies deliver. Examples of such applications are free viewpoint 3D video of live sports events (figure 1.1) and mobile AR games like Pokemon Go and Snapchat's AR camera filters called lenses (figure 1.2).

The rich data captured by new sensors for AR and VR content require efficient representations that can be stored, transmitted and reconstructed at a display device. There are different representations for 3D content that have been developed to facilitate such reconstructions. A convenient theoretical model that can be used as reference against which different representations can be compared is the plenoptic representation. This representation describes every point in the scene including the intensity distribution of light and how each point changes as the viewport is moved around the scene. The different representations

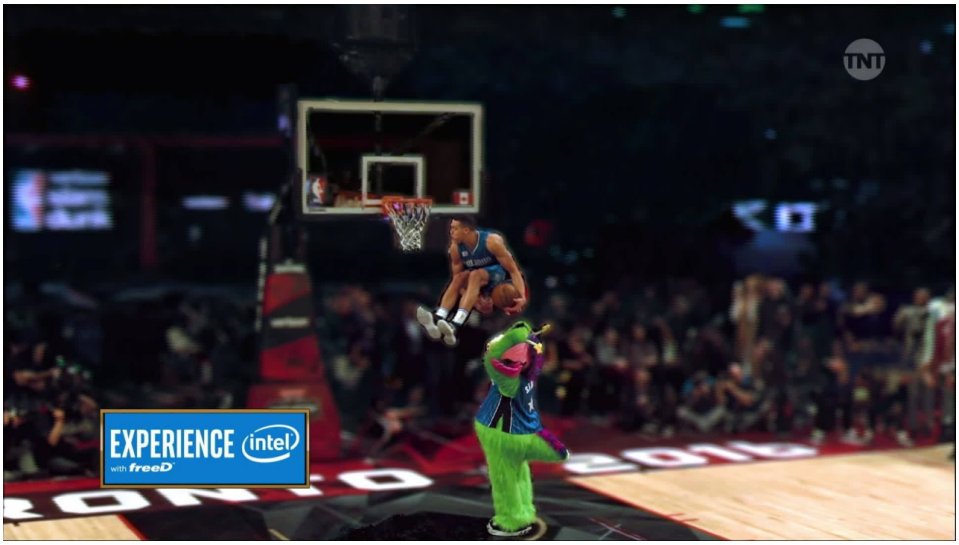


Figure 1.1: Free viewpoint live sports replay technology from Intel [6]

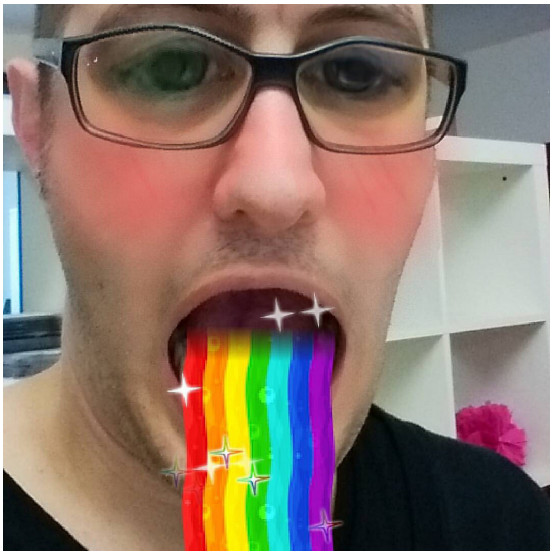


Figure 1.2: Snapchat AR camera filters [11]

that attempt to approximate the plenoptic representation include omnidirectional, depth enhanced, point clouds, light field and holography among others. The general work flow for all these representations has been described by JPEG [20]. Such representations also allow users to interact with the scene in new ways and enable a host of new multimedia applications including VR/AR. These new novel applications also allow users to interact with the representations, from simple controls such as changes of contrast, focus and changing the viewpoint to more powerful interactions such as manipulating objects in the scene. More

powerful interactions include the ability to change the viewpoint freely within the scene. Such interactions and are a prerequisite for 6 degree of freedom AR/VR. This also allows for immersive social AR/VR applications such as tele-immersive video.

Point clouds are one such representation that are useful for applications that are real time and require photorealistic reconstructions. Some of these applications, as identified by the MPEG 3D computer graphics group [13] include real time 3D immersive telepresence, Virtual Reality (VR) content viewing, 3D free viewpoint sports replays, geographic information systems, cultural heritage applications and autonomous navigation using large scale 3D dynamic maps.

Point clouds are a collection of points in 3D space with x,y and z coordinates describing the geometry, with additional attributes at each point such as colors, normals, transparency and material properties. They are characterized by their simplicity and versatility. They do not require any triangulation computation like meshes, do not require heavy pre-processing and are resilient to noise. They are thus suitable for real time applications. Point clouds are versatile as they have no restrictions on the additional attributes that can be stored at each 3D point. For example they can be used as an approximation of light fields by including the intensity distribution of light at each point. These characteristics make point clouds a very powerful format for VR and AR.

However point clouds are also challenging to compress. As they do not have connectivity and topology (or surface information), it is difficult to identify spatial redundancies and occlusions for compression within one frame (intra frame). There are also no explicit point correspondences between frames or even fixed bounding boxes which makes it challenging to identify temporal redundancies for compression across a sequence of frames (inter frame).

Recently, the standardization organizations MPEG and JPEG have issued call for proposals [14, 20] for point cloud compression techniques. The Ad-hoc groups that conduct this activity are comprised of major industry partners who have contributed datasets for the testing and evaluation of submitted proposals. Previous research into point cloud compression especially for compressing 3D reconstructions of humans, consisted of noisy point clouds acquired by consumer depth sensors such as the Microsoft Kinect. These point clouds were relatively sparse and could not capture the finer details of people and their clothing. They were noisy making it harder to fit common bounding boxes and establishing correspondences across frames. Examples of such datasets can be found in [16]. One advantage of such datasets is that they are easy to acquire and do not require any preprocessing making them suitable for real time applications. The new datasets, on the other hand, have been made available as part of the ongoing call for proposals and are highly photorealistic and have low noise, however they are also more dense and consist of 800,000 to 1 million points [14].

The most popular datastructure used to code point clouds is the octree. An octree is used to partition 3D space by recursively subdividing it into 8 octants. For point clouds this recursive subdivision is done for all regions that are occupied. This datastructure is the 3D analog of the 2D quadtree used in video compression. The primary reason octrees have been used for point cloud compression [26, 23, 36] is because they enable a more efficient representation of the point cloud geometry by regularizing the structure and using

Figure 1.3: Skiing dataset acquired using Kinect sensors [16]

occupancy codes for partitioned spaces. This is done during the subdivision process where a code of occupied octants can be generated at each level of the tree. These occupancy codes together describe the entire geometry of the point cloud. The leaf nodes of the tree are voxels and the geometry of points within the voxel can be differentially encoded (for example using coordinates relative to the centroid of each voxel). The attributes such as color are compressed separately for each leaf node either by mapping to an image grid [26, 23, 15] or by graph transform [38, 41, 31]. The graph transform method is not suited for dynamic scenes as the computational overhead becomes prohibitive but they are more suited for static point clouds. For dynamic scenes directly encoding colors by placing them on a 2D image grid has been shown to be more effective [26].

The compression process is repeated for each of the leaf nodes in the octree. The number of leaf nodes in an octree are related to the tree depth exponentially. For a tree of depth n the number of leaf nodes can be up to 8^n depending on how they occupy the 3D space (empty regions are not subdivided further). The computational complexity is therefore exponential with respect to the octree depth or level of detail, as the number of leaf nodes that need to be coded increase by a factor of 8 each time subdivision occurs. This problem is especially relevant to the photorealistic datasets used by MPEG as the density of point clouds is significantly high and requires a higher level of detail to maintain quality. This results in a deeper octree with exponentially more leaf nodes. One approach to address this problem was proposed by Ainala et al [15] where the authors suggest using the octree up to a certain fixed level of detail where this approach is feasible and to encode additional detail in an enhancement layer. They provided a proof of concept through an initial implementation based on plane projection approximation for static point clouds.

In this thesis we propose an octree based compression scheme that extends the codec

Figure 1.4: Dense photorealistic point cloud contributed by 8i technologies [19] to the MPEG Cfp [14]

created by Mekuria et al for dynamic point clouds [26] by adding the enhancement layer approach proposed by Ainala et al [15]. The enhancement layer is based on plane projection approximation where geometry is differentially encoded and colors are scanned to a 2D image, stacked and then compressed using JPEG compression [15]. The second contribution is to extend this mechanism by adding inter prediction to the enhancement layer. This is done by identifying and reusing the redundant leaf nodes of the enhancement layer across frames to exploit temporal redundancies. Results indicate that this leads to an additional bit rate saving of upto 10% on the point cloud sequences we tested at comparable objective quality.

For comparison purposes we use the open source version of the MPEG anchor codec. The approach of Ainala et al. [15] was thus integrated within the MPEG anchor codec. We provide a comparison of results with the initial proof of concept by Ainala et al [15] on the datasets provided by Alexiadis et al [16]. We then demonstrate the bit rate savings and

resulting distortions by adding inter frame coding to the enhancement layer. Lastly we show that our approach can result in upto 15% bitrate savings at comparable objective quality to the anchor codec used by MPEG.

1.1 Research Questions

The research questions investigated in this thesis are:

1.1.1 How can octree based point cloud codecs be improved to scale to dense photorealistic point clouds?

This research question deals with addressing the problem of the exponentially more computational overhead imposed by dense point clouds due to deeper octrees. There is a need to modify the octree based approach by terminating at a preset depth where compression is computationally viable and identifying a different scheme to encode additional geometry and attribute information.

1.1.2 How to identify and exploit temporal redundancies in the enhancement layer of a point cloud codec?

This research question deals with identifying a new scheme to exploit temporal redundancies in streams of time varying dense point clouds after the octree stage is terminated. There is a need to exploit temporal redundancies in the plane projection based enhancement layer to further compress the stream.

1.2 Thesis outline

In the following chapter different representations that are used to reconstruct scenes for VR and AR are described along with their applications. Next The applications of point clouds and previous research done into compressing them is discussed. Finally the datasets and metrics used to evaluate the quality of the decoded point cloud are discussed. The framework and implementation of our approach are discussed in chapter 3. The results of validating our approach against the enhancement layer proof of concept by Ainala et al [15], identifying the additional bitrate savings due to inter prediction and evaluating our approach against the MPEG anchor codec are discussed in chapter 4. Lastly the conclusion and future work that can be done to further improve the codec are discussed in chapter 5.

Chapter 2

Related work

AR and VR technology have now become affordable and viable for the general public. As a consequence, there are many companies engaged and innovating in this sector. The ability to run AR/VR applications on smart phones has also had a significant impact on attracting commercial interest. These companies have been able to create meaningful and unique content for applications in a plethora of industries. In education, this technology has been used to create stronger educational experiences by using a more immersive learning environment such as Google expeditions [5] that uses a head mounted display to show 360 videos and provide guided tours. In health care, VR/AR has been used in training for and in planning surgeries as shown in figure 2.1. Some of the applications and limitations of using AR in surgery have been explored by Khor et al [24]. The authors identify the potential for using AR and VR in anatomical evaluation, broadcasting or recording surgery and training. They also identify the need for smaller and lighter HMDs as well as better data management and protection. Another example are devices like Accuvein [3] that projects an image of the veins on to the patients skin to assist doctors and nurses. In aviation and space travel AR/VR has been used to conduct simulations and to provide training as a cost effective alternative to risking lives and expensive equipment. In travel VR/AR has been used to provide virtual tours and previews such as Marriot Hotel's Teleporter, a service that allows guests to virtually travel to many Marriot properties and VR Postcards a service that allow guests to view travel stories in 3D [8]. In skilled trades especially in the automotive manufacturing, factory workers use smartglasses such as the DAQRI smart helmet that overlays an image of internal structure and additional information such as instructions in the workers field of view [4]. This helps to improve the workers situational awareness and allows them to remotely receive assistance from experts. In entertainment there are many exclusively AR/VR games, some have been published on platforms like Steam in collaboration with Occulus. 8i technologies released a mobile application called holo [2] that allows users to overlay a hologram of celebrities in their smartphone's camera feed and even take selfies with them.

In the rest of this chapter we first discuss the general process flow for 3D content and some of the different 3D representations. We then describe point clouds and their applications. We then review previous research into point cloud compression and finally discuss evaluation metrics and datasets available to test point cloud codecs.

Figure 2.1: A surgeon using Google Glass in the operating theatre [24]

2.1 Process flow

There are a plethora of technologies and representations that can be used to realize immersive AR/VR applications. This includes sensors, display types, reconstruction and representations to capture, store, transmit and display 3D content. In order to ensure interoperability and facilitate integration there is a need for a common overall process flow. The JPEG Pleno initiative has described this as shown in figure 2.2.

The first stage in the process flow is acquisition and creation. A wide range of sensors can be used to capture the scene and generate metadata to facilitate different applications. Some sensors might require pre-processing such as stereo conversion, structure from motion when sensors do not automatically detect depth and voxelization of irregular data to fit it on a regular grid. Synthetic content can also be blended in with naturalistic content from the scene for augmented reality applications.

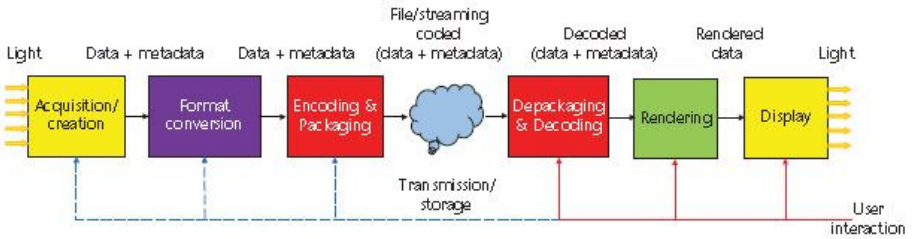


Figure 2.2: End to end plenoptic process flow [20]

The data from the acquisition phase is dependent only on sensor capabilities and constraints. In order to ensure interoperability and to satisfy application specific requirements such as compression, display and user interaction, it is necessary to convert the data to a different format that is more suitable. This format conversion is the next step of the process flow. For example point clouds can be converted to meshes and polygon clouds by calculating the triangulation and adding topology information.

Before the data can be transmitted an efficient method to store it is required. In this step an encoder is used to generate a more efficient representation of the data while satisfying requirements like scalability and error resilience. Different objects in the scene can be encoded separately and view dependence can be used to optimize the coding. The compression can be lossless for objects where the best possible representation is required by the application and lossy in applications that require more efficient representations (for real time applications like tele-immersion). Lossy encoding is done based on a rate-distortion optimization and different attributes/meta data can be distorted to different extents based on human perception. In real time distributed applications, progressive decoding might be required to adapt to varying network conditions. For this, the encoding will have to be done for multiple levels of detail. After encoding, the metadata is added and packaged together for transmission or storage.

The data has to be unpacked before it can be used. Decoding is required before rendering and display to recover the description of the scene. If lossy compression was used there will be distortions in the decoded dataset.

The last step is to display the content for the end user. Rendering is required in order to provide the best possible experience to the user and is display dependent. Post processing can also be done to improve aesthetics. Locally stored content can be added to the scene at this stage. The dataset will have to be filtered and transformed to extract relevant information based on the display such as a specific viewport. A range of display devices like head mounted, light field and autostereoscopic displays can be used to view rich 3D content, but compatibility with conventional 2D displays might also be required. Some applications also allow the user to interact with the representation to manipulate objects, as well as make changes to the focal plane, lighting conditions and move the point of view within the scene.

2.2 Representations of 3D content

There are many representations that try to reconstruct a scene by approximating the plenoptic representation. They are defined based on the content that different sensors and camera systems are able to generate by themselves or by processing the raw data and fusing them across sensors (for example photogrammetry can be used to create a 3D point cloud reconstruction from multiple 2D images of the same object). These sensors and camera systems generate visual content that provide a richer description of the scene. Some of the representations are omnidirectional, depth enhanced, point clouds, light field and holography.

Figure 2.3: Facebook x24 omnidirectional camera

2.2.1 Omnidirectional

Stereo information about a scene can be generated by using multiple images of the same scene from different viewpoints. Such panoramic content can be captured by stitching images from an array of cameras or rotating a single camera about it's optical center or by capturing images with wide angle lens that offer a wider field of view. There are two ways to encode omnidirectional content: in the spherical domain or by mapping to a rectangular plane. Most research on encoding omnidirectional content use the latter in order to leverage modern advances in video coding like the H.264/AVC. A comparison of different mapping schemes is shown in [40]. Omnidirectional content does not capture information about the intensity distribution of light and does not contain explicit geometry information so the view can only be rotated around a single point and can capture dynamic scenes. An example of an omnidirectional camera is the Facebook x24 shown in figure 2.3.

2.2.2 Depth enhanced

Depth enhanced content can be captured by extracting depth from multiple images or by using depth sensing cameras like the time of flight camera. Time of flight cameras measure the phase delay of reflected infrared light as described in [21]. This representation also does not capture the intensity distribution of light but can be used for 3D reconstruction as shown in [22].

2.2.3 Point Cloud

Point clouds are captured as a set of points in 3D space with three coordinates and some attached attributes like color and normals. They are typically captured using 3D scanners and LiDAR scanners. They are subsequently used to render 3D images for example by mesh generation. They typically do not capture the intensity distribution of light.

Figure 2.4: Lytro light field camera

2.2.4 Light field

Light fields can be created by capturing a scene using an array of cameras or a single sensor combined with microlenses to sample rays of light. They capture the intensity distribution of light but do not allow the user to move within the scene. The microlenses enable the capturing of ray information, as a consequence changes can be made to an image after capture such as focusing on different objects and making small changes to the view point. Consumer light field cameras such as the Lytro [7], shown in figure 2.4, have this feature which enable it to have a higher depth of field as compared to traditional cameras. Industrial light field cameras such as the cameras created by Raytrix [10] have a dedicated GPU that help create a stereo conversion to 3D representations that allow high precision measurements. These cameras have multiple applications including automated visual inspection of circuit components in a production line.

2.2.5 Holography

This technique portrays the full plenoptic function except time to a 2D map. This representation can be synthetic (computer generated) or can capture naturalistic content using interferometric capturing. Holography is based on capturing the phase information of incident light. This is achieved by splitting a coherent light source (constant phase difference) like a laser into a reference beam and an object beam (that is scattered by the object). The interference pattern of the two beams is measured to capture phase information, thus enabling a 3D representation with realistic parallax. This technology is currently developed more for micro-scale applications as compared to macro applications.

2.3 Point Clouds and their applications

Point clouds are a collection of points in 3D space stored as a combination of geometry and attributes. Each point in the cloud represents a point on the surface of an object and is described by x,y,z coordinates and additional attributes. Some of the possible attributes are color, material properties, transparency and normals at that point in the surface. Point clouds can be static comprising a single frame or dynamic, a sequence of frames or a stream of time varying point clouds.

They can be captured by many different sensors and camera systems. For example, point clouds can be reconstructed from consumer depth sensors such as the Microsoft Kinect as shown by Alexiadis et al [16]. They can be captured by Synthetic Aperture Radar (SAR) and Light Detection and Ranging (LiDAR) sensors as done by Umbra 3D [12] to map geographical areas. Point clouds have also been constructed using photogrammetry from multiple 2D images as done by 8i Technologies [2] to generate reconstructions of people in their Holo application.

Point clouds are characterized by their simplicity and versatility. Some of the benefits include the ability to add new attributes (either directly acquired by the sensor or computed in a processing step), being resilient to noise and not requiring preprocessing. Point clouds are resilient to noise as there are no assumptions on the structure such as the smooth manifold assumption needed for meshes. They also do not require preprocessing such as triangulation and connectivity computations and are hence suited to real time applications.

However point clouds are unorganized and lack any correspondence across frames. This makes it challenging to exploit temporal redundancies for compression.

The new representations presented in the previous section like point clouds have also enabled a host of new applications that allow the user to experience and interact with the reconstruction in new ways. Some of the applications of point clouds are real time 3D immersive telepresence, Virtual Reality (VR) content viewing, 3D free viewpoint sports replays, geographic information systems, cultural heritage applications and autonomous navigation using large scale 3D dynamic maps [13] among others. The requirements for point clouds used in these applications are described in table 2.1.

Application	Point cloud type	Coder Type	Size	Encoder/ decoder complexity	Additional Requirements	Additional Attributes
3D Tele-immersive video	Dynamic	Lossy	100,000 to 10 million	Low encoder and decoder complexity	Bit-rate control, Error resilience to transmission errors	Color attributes with 8-10 bits per component, normals and material properties
VR content viewing with interactive parallax	Dynamic	Lossy	100,000 to 10 million	Low decoder complexity	Bit-rate control, Error resilience to transmission errors, global parameters to define spatial constraints of viewport	Color attributes with 8-10 bits per component, normals and material properties
3D free viewpoint sports broadcasting	Dynamic	Lossy	100,000 to 100 million. Can contain multiple clusters	Low encoder and decoder complexity	Low delay encoding and decoding for multiple clusters or objects (different players)	Color attributes with 8-12 bit per component
Geographic information systems	Static	Lossless	Billions of points	Progressive coding	Region selectivity to obtain subsets representing a smaller geographic area, lossless coding to enable the best representation for inspection of features like vegetation	Additional attributes related to geographic properties
Cultural Heritage	Static	Lossless	1 million to billions of points	Progressive coding	Can contain multiple clusters, lossless coding to enable best representation when possible	Color attributes with 8-12 bits per component, generic attributes like material properties,
Large scale 3D dynamic maps for autonomous navigation	Dynamic	Lossy	Millions to billions of points with up to 1 cm precision	Low encoder and decoder complexity	Low delay encoding and decoding, high precision, region selectivity,	Color attributes with with 8-12 bits per component, normals and reflectance properties

Table 2.1: Application of point clouds [13]

2.3.1 3D Immersive telepresence

3D immersive telepresence is an augmented reality (AR) application for real time communication where each participant is reconstructed in 3D in a virtual world where they can interact with each other. An example is the system created by Mekuria et al [26] shown in figure 2.5. In order to run in real time with a sufficient frame rate, both the encoder and the decoder must have low complexity, additionally the decoding process should be progressive in order to adapt to varying network conditions where a partial bit-stream can be used to reconstruct at a lower level of detail. This application uses dynamic point clouds and requires a lossy codec to work in real time.

Figure 2.5: Screenshot a point cloud rendered compositely with synthetic content for immersive telepresence [26]

2.3.2 VR Content viewing

VR content viewing involves delivering an interactive experience to the user on a head mounted display such as the device in figure 2.6 with a large field of view. As this is not a real time process, the content can have a longer production workflow that includes adding lighting, rendering and other visual effects. The view dependence of such an implementation can be exploited to optimize the encoding process as the user is more likely to view a scene through viewports along the equator as compared to the poles. This application uses dynamic point clouds and requires lossy compression.

Figure 2.6: Head mounted display that can be used to view VR content

2.3.3 Free viewpoint sports replays

Free viewpoint 3D sports replays have been implemented by Replay Technologies [1], this application allows users to playback live sports events with a free viewpoint. This requires low encoding and decoding complexity as the events are live. This application uses dynamic point clouds and requires lossy compression so content can be viewed live.

2.3.4 Geographic information systems

Geographic information systems capture 3D data using scanners like LiDAR and SAR (see figure 2.7), but these datasets are too big to be used directly, these applications require varying levels of detail and region selectivity. They also require lossless compression so that the highest quality representation can be provided to the user that allows accurate inspection of features like vegetation and altitude. This application uses static point clouds.

2.3.5 Cultural heritage

Cultural heritage applications involve archiving and visualizing cultural heritage objects, this requires lossless compression so that the best possible representation can be displayed when required. An example is shown in figure 2.8 where point cloud compression can facilitate streaming and make these objects available to a wider audience. This application uses static point clouds.

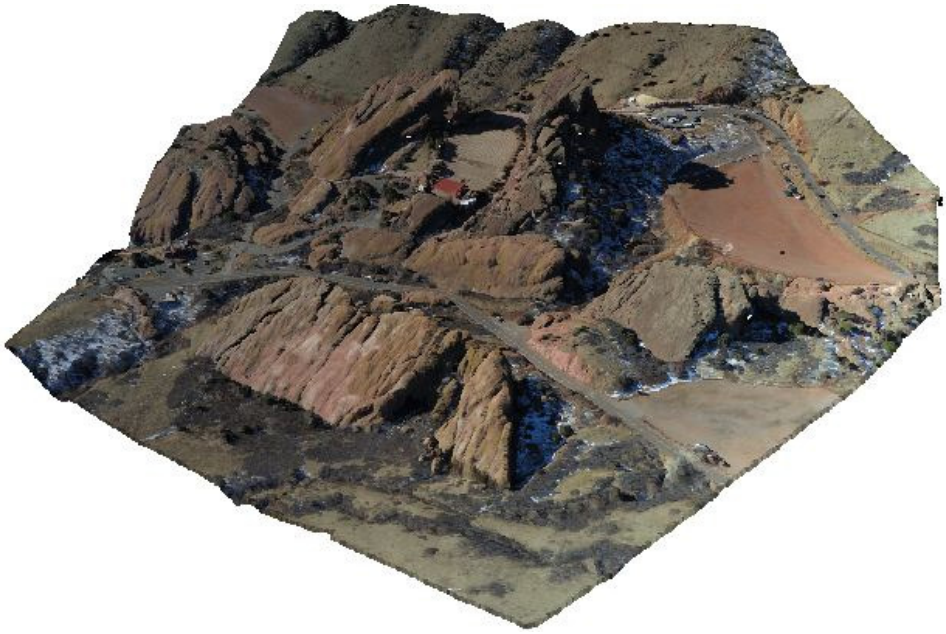


Figure 2.7: Point cloud of a geographical area scanned by LiDAR

Figure 2.8: Building facade [14]

2.3.6 Autonomous Navigation

Autonomous navigation using large scale 3D maps is another application that requires real time implementation, this application additionally requires high precision with depth and colors in order to perform accurate localization and detect surfaces and obstacles. Maps can be created by overlaying point clouds captured using depth sensors over images from the camera as shown in figure 2.9. This application uses dynamic point clouds and requires lossy compression.

Figure 2.9: Map creation where point clouds captured by laser scans are overlayed over camera images [13]

2.4 Point cloud compression

3D point clouds used across different applications can be broadly classified as static or time varying. Compression of static point clouds aims to exploit spatial/geometric redundancies only, while the compression of time varying point clouds additionally aims to exploit temporal redundancies. In addition to this, 3D point clouds also require efficient schemes to code attributes like color. Point clouds were initially used in robotics such as the work of Radu B Rusu et al [35, 34, 33], but they have since been applied to many other areas such as augmented/mixed reality systems [26, 27] and topographical data collection [12].

A popular data structure that has been used to store point cloud information is the octree, shown in Figure 2.10 (an extension of the quadtree used in 2D). In this data structure, the entire frame is divided into 8 cubes, and this process is repeated as long as there are non empty blocks (blocks that are occupied and contain points) as shown in figure 2.11. Individual points are scanned with a depth first approach. This data structure helps to easily identify different levels of detail at different levels of the tree and the entire point cloud can be encoded in terms of occupied tree cells as shown in figure 2.10. This process has been described by Ruwen Schnabel et al. [36], where they also use surface approximations to predict the occupancy coding of the octree. Another approach is described by Yan Huang et

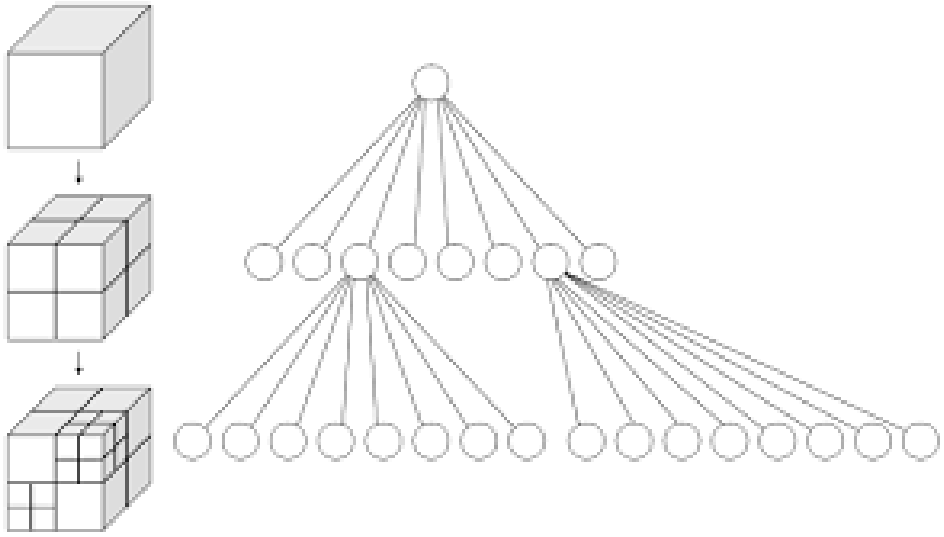


Figure 2.10: Occupancy coding after subdivision [9]

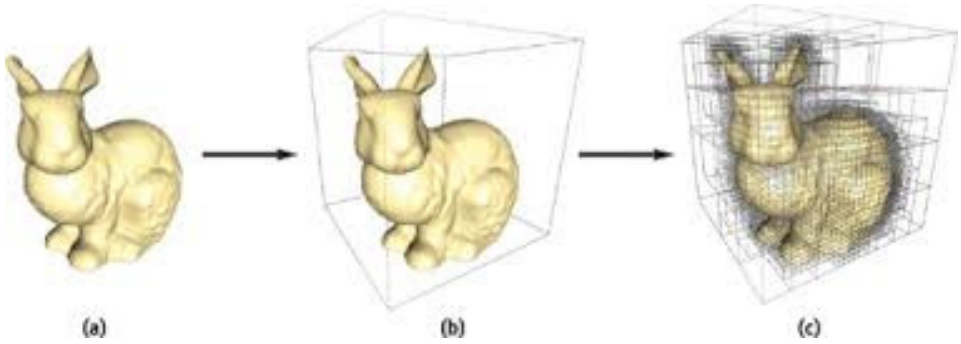


Figure 2.11: An example of recursive subdivision in octrees

al. [39] where the octree is first constructed and then the entropy of the occupancy codes is reduced by bit-reordering. The color and normals are approximated by their statistical average within each of the tree front cells. The color coding is achieved by applying PCA and Generalized Lloyd's algorithm to adaptively quantize the color attribute in a rate distortion optimized manner while reducing the number of quantization bins effectively reducing the number of bits required to store color information.

A popular framework used for storing, processing and transmitting point clouds is the point cloud library, an open source library created initially for indoor robot navigation by Radu B Rusu and Julius Kammerl and integrated with the Robot Operating System (ROS). The framework contains implementations of segmentation, registration, feature extraction, surface reconstruction, visualization, model fitting and compression for point clouds, It also contains the first point cloud codec capable of inter prediction for point cloud streams. The codec uses a double buffered octree data structure to detect spatial changes and coded the

occupancy codes at different levels of the octree using an XOR from the previous frame in the stream. They also use differential coding for point coordinates within a voxel or leaf node of the octree. This is done because the computational cost of processing an octree increases exponentially with an increase in the number of subdivisions.

The compression of point clouds can be split into inter frame and intra frame coding. The intra frame coder looks at one frame in isolation and compresses it by exploiting spatial redundancies. The inter frame coder predicts the current frame based on a previous frame by exploiting temporal redundancies, this is usually done with motion compensation similar to video coding. Point cloud codecs can be single rate (they decompress every frame to a fixed size and quality after transmission) or progressive. In some applications such as tele-immersive video the additional functionality of progressive decoding is a requirement. This means that it is possible to decode a lower quality point cloud from a partial bit stream in order to cope with varying network conditions. In addition to compressing the geometry, point cloud attributes also need to be compressed, different lossy and lossless coding schemes have also been suggested for this. Sparse voxel octrees (SVO) are a popular data structure that have been used to store and process point clouds [23].

Intra Frame coding in octrees can be achieved by entropy coding the occupancy codes as shown in [26] to compress the geometry. Zhang et al. [41] proposed a methodology to compress point cloud attributes using a graph fourier transform. They assume that an octree has been created and separately coded for geometry prior to coding attributes, they construct the graph similar to [38] and calculate a precision matrix (graph laplacian multipled by a scalar that reflects signal variance) for each macro-block and perform an eigen decomposition. They use this to quantize each colour component and entropy code the results. Ricardo Queroz et al. [31] used a region adaptive hierarchical transform in order to use the colors of nodes in lower levels of the octree to predict the colors of nodes in the next level. Each voxel is assigned a weight based on the number of voxels combined to generate them and a transform is used based on these weights. The results are quantized and entropy coded using arithmetic coding. Robert Cohen et al. [18] created a methodology that applies 3D intra prediction between macroblocks similar to video coders like H.264 AVC and HEVC. They used a nearest neighbor interpolation/extrapolation to overcome the problem of sparse macroblocks in point cloud octrees and to identify the value of an attribute at the boundary between macroblocks. These values are used to predict attributes in each macroblock and the residuals are coded using a shape adaptive DCT.

Inter frame coding was demonstrated in [23] where an XOR operation was performed between consecutive frames and the results were stored in an octree and coded using an entropy coder. This approach only predicted geometry and did not address the attributes. This method only worked in cases where objects in the scene did not move significantly across frames as there was no motion compensation and the geometry compression was lossless so the decoding process was not progressive. It was not possible to do a rate-distortion trade off. Thanou et al. [38] found matches between points in consecutive frames and warping the previous frame to the current frame and coding the color residuals using a graph fourier transform [41]. Mekuria et al. used a block based approach to calculate a rigid transform between compatible macro blocks in consecutive frames using the iterative closest points algorithm, this approach involves encoding at multiple levels of detail, thus allowing pro-

gressive decoding and also predicts both geometry and attribute (color) values. Mekuria et al. [15] also presented an improvement to this technique for individual point cloud frames using an enhancement layer. After a fixed octree depth they switch to a new layer that uses plane projection approximantion During octree decomposition in the enhancement layer the flatness of the voxels in a cell is calculated and if this value is below a threshold then the cell is a leaf node in the enhancement layer, otherwise the cell is subdivided. These leaf nodes are then coded separately in the enhancement layer, the remaining octree above is coded using the intra frame coder from the previous approach. The enhancement layer is coded by projecting the voxels in the leaf nodes onto a plane using PCA and selecting a raster scan order using the values of the principal components. These images are then coded using JPEG compression. The residuals after performing PCA are coded using a learning based self adaptive entropy coder (PAQ learning).

Thanou et al. [38] represent a sequence of point cloud frames as weighted undirected graphs and treat the underlying point coordinates and attributes as signals on the vertices of the graph, they assume that the point cloud has voxels on a regular 3D axis aligned grid with a given step size. Voxels are connected to each other by an edge if the distance between them is not greater than the unit distance in any direction, the weight of each edge reflects the connectivity pattern of nearby voxels. They then analyze the signals (coordinates and attributes) using spectral graph wavelets to identify features and match them across frames using the minimum Mahalanobis distance in order to create a set of sparse motion vectors that are used to warp the graph of the reference frame to the current frame, and then the current frame is coded based on the set difference between the current frame and the motion compensated reference frame. The residuals are quantized and entropy coded based on the approach described in [41]. Aamir Anis et al. [17] created an intermediate representation based on subdivisional triangular meshes where the authors re-sample the object or scene to obtain a set of consistently evolving point clouds represented by the vertices of triangular meshes. They then use the connectivity of the resampled points to apply a graph based wavelet transform to code geometry, color attributes and motion vectors. They encode using inter-frame coding when the mesh topology of consecutive frames are identical. They then apply the Graph Bior [28] filter banks to transform the vertex attributes for intra coded frames and the motion vectors for inter coded frames, these results are then quantized and entropy coded using the RLGR (Run Length Golomb-Rice) coder[25].

A limitation of octree based point cloud compression is that as the depth d of the octree increases the computational cost increase exponentially (upto 8^d). This problem has become more acute with more photorealistic datasets such as the datasets contributed by members of the Ad-hoc group for MPEG point cloud compression. These point clouds are far more dense as compared to the kinect datasets used to test the compression algorithms proposed in the past [23, 26] and require deeper octrees to have the same level of detail.

In order to overcome this issue our approach is to terminate the octree at a fixed level of detail that is computationally viable. Additional information is stored in an enhancement layer and will only be used if network conditions and computational resources allow it. The enhancement layer is based on a plane projection approximation where the geometry coordinates are transformed, sorted and differentially coded while the color attributes are placed in a 2D image grid using the same sort order for raster scanning. The resulting

images are stacked and compressed using JPEG compression. In this way the bond between geometry and attributes are maintained. The transformation is determined using principal component analysis. This process has been described in detail in the next chapter.

2.5 Evaluation metrics

In general quality assessment is a difficult and complicated task even for media formats where metrics are well developed such as video and meshes. Evaluation should be performed both objectively and subjectively in order to obtain reliable results. Objective evaluation is based on mathematical models that compare specific features of the original and decoded files. Subjective evaluation is performed by groups of users who view the content in controlled conditions. Subjective testing is usually more reliable but more expensive and time consuming. Subjective and objective evaluation of video codecs has been surveyed by Papadakis et al [29].

Point clouds can be split into:

- Static: A single frame of a static object
- Dynamic: A stream of point clouds that vary over time usually of a moving object

Codecs in general can be split into:

- Lossless: The decoded point cloud is identical in every respect to the original point cloud
- Lossy: The number of points in the decoded cloud are reduced in an attempt to remove redundant information. As a result, point correspondences are lost and we need to identify corresponding points in the original cloud. This is usually done by searching for the nearest neighbor in the original point cloud.

In previous research, point cloud codecs have mostly been evaluated using existing quality metrics from mesh and video compression such as the Peak Signal to Noise Ratio (PSNR).

The extent of compression is measured using the bit rate. For static point clouds this is measured as the number of bits per point. In case of lossy compression we divide the size of the entire decoded cloud by the number of points in the original cloud. For dynamic or time varying streams of point clouds the bit rate is measured in Mbits/sec. The same frame rate is maintained for the original and the decoded point cloud streams.

In principal, to evaluate lossless codecs the bit rate and run time are sufficient. However due to differences in how various platforms handle floating point numbers, the decoded geometry need not be numerically identical to the original cloud. For example if the point coordinate value 0.24 from a cloud stored as a ASCII PLY file is read, the binary32 representation stored in memory is 0.23999999463558197021484375E-1 and the resulting residual needs additional compression. The direct solution is to either use only fixed precision numbers or to develop a floating point precision compression scheme but this can affect the overall compression performance. To address this issue, without affecting compression,

MPEG has defined a set of rules for evaluating point cloud compression [14] to check if a decoded point cloud is lossless. The first criteria is that the number of points in the original point cloud A and the decoded point cloud B are the same. The next criteria is that for every point in A there exists a point in B (and vice versa) such that the difference in coordinate values are zero (or within a small tolerance level). The last criteria is to ensure that no two points in A share the same corresponding point in B (and vice versa).

To evaluate lossy codecs we require additional metrics to measure the extent of the distortions and artifacts introduced during compression. These artifacts can be spatial and temporal. In most of the previous research, the distortions in point clouds are measured separately for geometry and attributes. In this thesis we use only full reference metrics where the entire original cloud is compared to the decoded cloud. Full reference metrics are the most accurate but require the most computational effort and are usually used to evaluate codecs. Reduced and no reference metrics can be used for real time rate distortion optimization.

Other evaluation criteria that are defined for compression efficiency are based on the ability to operate at different ranges of bitrates or the ability of the codec to scale to varying network conditions, complexity of encoding and decoding as well as the level of detail scalability of different codecs.

In order to evaluate lossy point cloud codecs using full reference metrics it is necessary to identify corresponding points in the original and decoded clouds. However the decoded cloud contains fewer points than the original cloud so we use nearest neighbor search to identify corresponding points. In the implementation by Mekuria et al [26] this is done using the K-D tree based nearest neighbor search provided in PCL [32]. The next subsections detail the evaluation of geometry, attributes distortions and the rate distortion curve used to evaluate point cloud codecs.

2.5.1 Geometry distortion

In previous research, the point to point distance (D1) has been used between corresponding points from the decoded and original cloud as an error vector. Another more recent metric is the point to plane distance (D2). The idea is to measure the distance between the decoded point and the original object surface as a better indicator of geometric distortion. In 3D meshes, identifying the underlying surface is relatively easy but in point clouds this is harder to estimate. One solution is to convert the point cloud to a mesh but this process requires preprocessing (such as outlier removal) to meet the manifold assumption of meshes and is computationally expensive. The D2 metric uses the surface normal at the point in the original cloud as model of the surface under the assumption that this is a reasonable representation of the surface.

Given a geometric surface it is trivial to estimate the direction of the normal at any point on the surface. However since point clouds only sample points on the real surface it is necessary to calculate the surface information. The direct approach to obtain the underlying surface is to employ surface meshing techniques and then compute normals from the mesh. However this approach is computationally expensive. Another approach is to use an approximation of the surface by considering the neighborhood of the target point. This is

done by considering all points in the vicinity at a fixed radius around the point and computing the covariance matrix. The surface normal at the target point is then computed as the eigen vector corresponding to the smallest eigenvalue of the covariance matrix.

Using the point to point error vector and the surface normal at the point on the original cloud, the point to plane distance can be calculated. The orthogonal projection of the error vector on to a line parallel to the surface normal is point to plane error vector. This process is illustrated in figure 2.5.1. The D2 metric results in higher errors for points (in the decoded cloud) that are further away from the local surface plane in the original cloud as compared to the D1 metric.

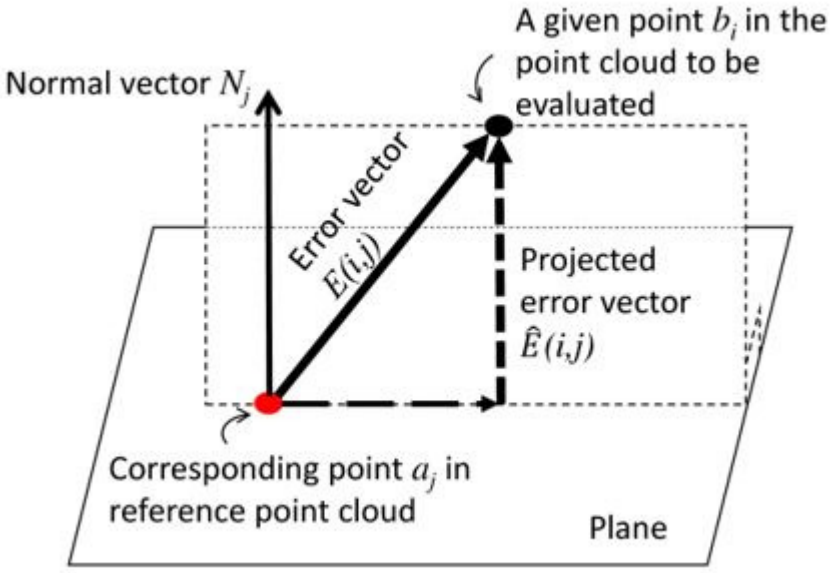


Figure 2.12: Objective metrics to evaluate distortion [14]

In order to have a full reference metric we need a distance metric to aggregate the two error vectors $d^{B,A}(i)$ defined for point cloud geometry across all points i in the decoded cloud B and their nearest neighbors in the original cloud A the two distance metrics. There are two ways to calculate the distance metrics: RMSD and Haus.

Root mean square distance (RMSD)

RMSD is obtained by calculating the square root of average across the squared length of error vectors for all points N_B in the decoded cloud as shown in equation 2.1.

$$d_{rmsd} = \frac{1}{N_B} \sum_{i=1}^{N_B} d^{(B,A)}(i) \tag{2.1}$$

Hausdorff (Haus) metric

The Haus metric is calculated as the maximum length of the error vectors for every point in the decoded point cloud.

$$d_{Haus} = max_{\forall i \in B} d^{B,A}(i) \tag{2.2}$$

Both these metrics can be used symmetrically or in both directions from the original to the decoded cloud or vice versa by identifying nearest neighbor matches twice. The maximum value from both directions can be used to obtain the symmetric RMSD or Haus metric. In this thesis we use only the error vectors for each point in the decoded cloud using RMSD in order to compare our implementation with the MPEG anchor in the objective evaluation procedure as specified in the recent call for proposals [14].

2.5.2 Attribute distortion

In this thesis we consider only color as the point cloud attribute to be compressed. This is because most point cloud datasets contain only colors when using sensor data directly. Datasets captured by laser scan can also contain reflectance, but these are usually used for scanning static images of large geographic areas. Attributes like normals are added in a separate processing step that can be run after decoding, so we exclude them from the codec. If the original cloud uses a color space other than YUV we convert it to YUV. To evaluate objective quality we only use colors in the YUV space as it is closer to human perception. Using the nearest neighbor matches between the decoded cloud and the original cloud we calculate the mean square error for each color channel separately. These can be weighted based on color perception with a higher weight for the Y channel. The MPEG CfP recommends using a weight of 6:1:1 in favor of the Y (luma) channel. The distortion is calculated using PSNR.

In video coding other objective metrics like Structural Similarity (SSIM) and Video Quality Metric (VQM) have also been used. SSIM is calculated on various windows of an image, this technique is ill suited to 3D point clouds as they are sparse and the computational overhead would be significant. VQM is based on subjective observations of factors like block distortion, blurring, color distortion, global noise and unnatural motion.

2.5.3 Rate distortion curve

In order to compare the objective quality performance of two point cloud codecs, we use objective metrics that have already been used for 3D meshes and video such as the Peak Signal to Noise Ratio (PSNR). The noise value is the aggregated error vectors for geometry and attributes. In order to calculate the peak signal value for geometry (in the original cloud) we identify the nearest neighbor of every point in the original point cloud using a kd tree to search and use the maximum value of this distance for static point clouds. For dynamic point clouds we use a constant value for all frames in order to reduce computational cost. Using these values the PSNR for geometry distortion can be calculated using both the D1 and the D2 metric. For the color attribute the peak value is set to 255 for each channel as they are

each assigned 8 bits. Using this we calculate the PSNR for each color channel. Each codec is then run with the same target bit rate multiple times at different values and the objective quality is recorded so a rate-distortion curve can be plotted. The overall improvement from one point cloud codec to the other can be expressed using the Bjontegaard Delta for bit rate similar to evaluating video codecs. This value is the average bit rate savings offered by one codec for comparable objective quality as compared to a second anchor metric. In our evaluation we follow this procedure for the two geometry distortion metrics and each color channel separately.

In addition to these objective metrics, subjective testing is also done to evaluate the quality of lossy point cloud codecs. This is done because the objective quality metrics are not as well developed as video codecs and are unable to capture the presence of artifacts in motion compensated coding.

2.5.4 Subjective evaluation

Subjective evaluation methods are based on human observation and the perceived quality. The most common evaluation methods for video compression [29] are Double Stimulus Continuous Quality Scale(DSCQS) and Double Stimulus Impairment Scale(DSIS). Other subjective methods include Single Stimulus Continuous Quality Evaluation (SSCQE), Absolute Category Rating (ACR), Degradation Category Rating (DCR) and Pair Comparison(PC). Double stimulus methods repeatedly present the original and testing material in sequence with small intervals.

2.6 Datasets

In order to evaluate our implementation we have used two sets of point cloud datasets. Previous research into point cloud compression was done using noisy point clouds captured using consumer depth sensors such as the Microsoft Kinect. In order to compare our implementation to the enhancement layer implementation of Ainala et al [15] we use the full 3D reconstructions of moving objects captured by Alexiadis et al [16] using multiple consumer depth sensors shown in table 2.2. These datasets are relatively sparse and noisy.

In order to identify the additional bitrate savings due the inclusion of inter prediction in the enhancement layer and to evaluate the performance of our codec against the MPEG anchor we use the datasets listed in the MPEG CfP [14]. These datasets were contributed by companies that are a part of the ad-hoc group on point cloud compression. In order to have photorealistic reconstructions they are significantly more dense and relatively noise free compared to the previous datasets. They are used to create the rate distortion curves shown in Chapter 4. These datasets are summarized in table 2.3.

The other datasets available to test point cloud codecs include the static datasets used by MPEG, these are shown in table 2.4. There is also a repository of point cloud datasets hosted by JPEG and contributed by their industry partners, these have been listed in table 2.5.

Name	Number of frames	Approximate mean number of points	Approximate size
Skiing3-Zippering	190	340K	2.0 GB
Jogging1-Zippering	231	290K	2.0 GB
Dimitris-Zippering	231	310K	4.0 GB
Dimitris2-Zippering	121	310K	1.2 GB
Alex-Zippering	901	170K	7.5 GB
Christos-Zippering	762	270K	6.4 GB
S04-5KW-DimitriosSession4-Zippering	2351	230K	15 GB
S17-5KW-Xenia-Zippering	6531	190K	1.2 GB

Table 2.2: Point cloud datasets captured using multiple consumer depth cameras [16] `http://vcl.iti.gr/reconstruction/`

Name	Number of frames	Approximate mean number of points	Approximate size
Queen	250	1 million	5.0 GB
8i VFB-Loot*	300	780K	4.9 GB
8i VFB-Red and Black*	300	700K	4.5 GB
8i VFB-Soldier*	300	1.5 million	6.7 GB
8i VFB-Long dress*	300	800K	5.3 GB

Table 2.3: Point cloud datasets of dynamic objects used for evaluation against the MPEG anchor [14]. Datasets marked with a * are also available in the JPEG PLENO database

Name	Geometry Precision	Number of points	Approximate size
Egyptian Mask	32	1 272,689	7.4 MB
Statue_Klimt	32	499,886	13 MB
Arco Valentino Dense*	32	1,530,552	4.5 GB
Facade 9	32	1,602,990	43 MB
Frog67	32	3,630,907	96.9 MB
Facade15	32	8,929,532	230 MB
Facade64	32	19,714,629	526 MB
Queen-frame-0200	10	1,000,993	20 MB
Loot_vox10_1200	10	805,285	160 MB
Redandblack_vox10_1050	10	787,237	170 MB
Soldier_vox10_0690	10	108,9091	189 MB
Shiva35	32	1,010,591	30 MB
House57	32	5,001,077	133 MB
Palazzo Carignano Dense*	32	4,203,962	4.2 GB
Head39	32	14,025,710	190 MB
Longdress_vox10_1300	10	857,966	170 MB
Landscape14	32	72,145,549	9.3 GB
Stanford Area2	32	54,989,822	1.4 GB
Stanford Area4	32	47,485,046	1.3 GB

Table 2.4: Point cloud datasets of static objects used by MPEG[14]. Datasets market with a * are also available in the JPEG PLENO database

Name	Type	Image Set	Number of frames	Voxel/Point Count	Approximate size
Andrew10	Dynamic	Microsoft Voxelized Upper Bodies	300	1024 X 1024 X 1024 voxels	8.7 GB
David10	Dynamic	Microsoft Voxelized Upper Bodies	300	1024 X 1024 X 1024 voxels	6.79 GB
Phil10	Dynamic	Microsoft Voxelized Upper Bodies	300	1024 X 1024 X 1024 voxels	7.63 GB
Ricardo10	Dynamic	Microsoft Voxelized Upper Bodies	300	1024 X 1024 X 1024 voxels	4.23 GB
Sarah10	Dynamic	Microsoft Voxelized Upper Bodies	300	1024 X 1024 X 1024 voxels	4.58 GB
Science museum shipping galleries	Dynamic	ScanLAB Projects	41 clusters of 4-5 point clouds	approx 500,000 points per cloud	400 GB
Bi-plane	Static	ScanLAB Projects	1	1024 X 1024 X 1024	5.7 GB
Villa La Tesoriera	Static	GTI-UPM	1	741966 points	7.32 GB

Table 2.5: Point cloud datasets used by the JPEG Pleno initiative [20]

Chapter 3

Framework and implementation

The work presented in this thesis has been implemented using the framework of the MPEG anchor codec. This anchor has been approved by academic and industrial partners of the MPEG adhoc group on point cloud compression as a reference codec for the call for proposals [14]. Our solution is built on top of the implementation done by Mekuria et al as described in [26] that is also available at `https://github.com/cwi-dis/cwi-pcl-codec`. Mekuria et al added their implementation to the compression module of the open source project PCL (Point Cloud Library) `https://github.com/PointCloudLibrary/pcl`.

3.1 Point Cloud Library

The codec in PCL created by Julius Kammerl et al [23] features an inter frame XOR based codec as described in the chapter 2. PCL contains functions to perform the octree subdivision, traversal and search. They use a double buffered octree to store successive frames in order to perform inter prediction. The buffers are switched every time a new frame is loaded, as a result the current frame and the previous frame are always available in memory.

The geometry or point coordinates and the attributes like color are encoded separately while maintaining the bond between them.

The codec in PCL features an inter prediction algorithm for point cloud geometry. This algorithm is based on the double buffered octree and the occupancy codes of octree nodes as shown in figure 3.1. An XOR operation is performed across the double buffer in order to reduce the entropy of the final bitstream and make statistical compression more efficient. In addition, at the leaf nodes an origin location is calculated at the lowest (smallest x, y and z) coordinates in the voxel, then the distances between each voxel local point and the origin is calculated. These distances are then discretized and coded using positive integers based on the specified precision. This quantization of point coordinates is the last step in lossy compression and a bitstream is generated. Next, a lossless statistical compression algorithm can be applied to the resulting bitstream to further compress it. As the voxel local point coordinates are stored as integers a range coder can be used on the bitstream. This algorithm is the integer arithmetic version of arithmetic coding.

This approach only compresses point cloud geometry and does not cover attributes like

color. In our implementation we use selected modules from this codec such as octree subdivision, traversal, serialization and composing a frame header for the output bitstream.

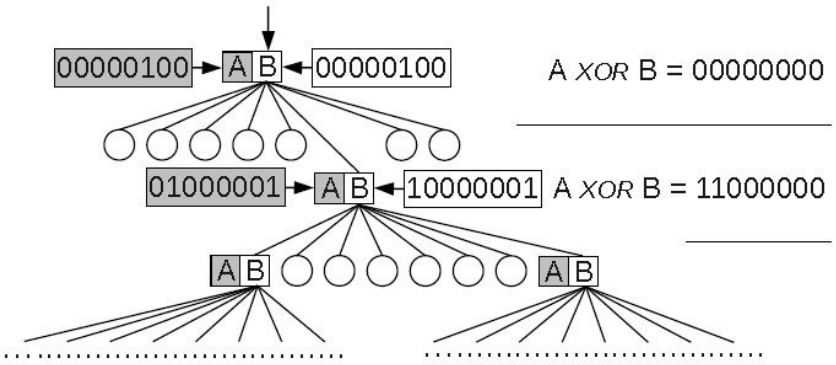


Figure 3.1: Differential occupancy coding of the double buffered octree [23]

3.2 MPEG Anchor codec

The MPEG anchor codec improves the PCL codec primarily by adding a color coder and an inter prediction scheme that also includes predicting attributes (color). The approach is shown in figure 3.4. The intra frame coder encodes geometry at fixed levels of detail by differentially encoding voxel local points from the PCL codec. This process is repeated at different pre-specified levels of detail so that the decoding process can be progressive and a final level of detail can be chosen by the decoder based on prevailing network conditions. A carry less byte range coder is then used on all supported LoDs for lossless entropy coding of the final bitstream. The range coder has been preferred over arithmetic coding as it operates at a byte level making it much faster on general microprocessors. The intra frame coder also encodes colors by performing a depth first traversal of the octree and scanning the colors to a structured JPEG image grid using a zig zag pattern to further exploit the correlations among co-located points.

The inter frame coder in their codec reuses the intra frame coder in combination with a lossy prediction scheme based on the iterative closest points (ICP) algorithm from PCL. This is shown in figure 3.3. Macroblocks are defined at a predefined number of levels above the final level of detail or voxel size. The inter frame coder identifies shared macroblocks in the previous frame and the current frame that are comparable and eligible for prediction. This eligibility is based on the macroblock point count and color contrast changes. The ICP algorithm then identifies a rigid transform between shared macroblocks in the both frames and if the algorithm converges the transform is used as a predictor. The transform is stored as a rotation quaternion and three quantized translation components that occupy 16 bits each. The framework can optionally be used to calculate a color offset between corresponding macroblocks across frames. Macroblocks can be uniquely identified and randomly accessed using a key k(x,y,z) containing three 16 bit integer values. Macroblocks

that are coded using inter prediction are stored and transmitted as a structure with 18 bytes as shown in figure 3.2. An additional 3 bytes can be used to store the optional color offset.

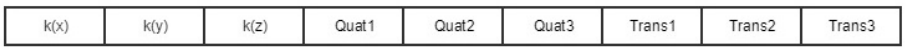


Figure 3.2: Memory Structure of a predicted macroblock in blocks of 2 bytes each

3.3 Enhancement layer

Ainala et al proposed adding an enhancement layer to the MPEG anchor [15]. They only implemented a proof of concept. We first implemented and integrated their proposal with the MPEG anchor in order to evaluate its actual suitability and effectiveness in compressing the dense point clouds used in the MPEG CfP [14].

This was done by specifying the depth at which the octree based codec is terminated and the enhancement layer takes over. The goal is to use the octree based codec as long as it is computationally viable and encode additional details in an enhancement layer. Both the intra frame coding and inter prediction modules of the anchor are used. During the subdivision process, once the octree target level of detail is reached, we move to the enhancement layer. Internally the enhancement layer is treated as an additional level of detail with the octree subdivision function overwritten to terminate when the flatness criteria is met. This is done by first calculating the covariance matrix of points within a tree node using PCL functions from the Eigen library. The centroid of points is used as the mean.

The flatness criteria, shown in equation 3.1, is then calculated as the ratio of the eigenvalues of the covariance matrix within a node. A comparison between the smallest eigenvalue (direction of minimum variation) and the sum of all eigenvalues is used an an indicator of flatness. If the node is flat enough we then apply principal component analysis (PCA) on all voxels within the node. This is done by using the eigenvectors corresponding to the eigenvalues calculated in the previous step to create a 3X3 rotation matrix. We then rotate the point coordinates to remove linear correlations. This orthogonal transformation converts the x,y,z coordinates to u,v,w in decreasing order of variance. The point coordinates are then sorted based on formula 3.2, differentially coded and quantized to integer values. The orthogonal transformation is stored as a quaternion using 6 bytes and added to the bit stream. The resulting bit stream is further compressed using lossless statistical compression by using the range coder implemented in the MPEG anchor.

$$\theta = \frac{min(\lambda 1, \lambda 2, \lambda 3)}{(\lambda 1 + \lambda 2 + \lambda 3)} \tag{3.1}$$

$$index = sort(\alpha * u + v) \tag{3.2}$$

Color is then coded for each flat node by using the index from equation 3.2 onto a 2D grid of fixed width. The last row is padded with zeros to complete the image. All flat nodes

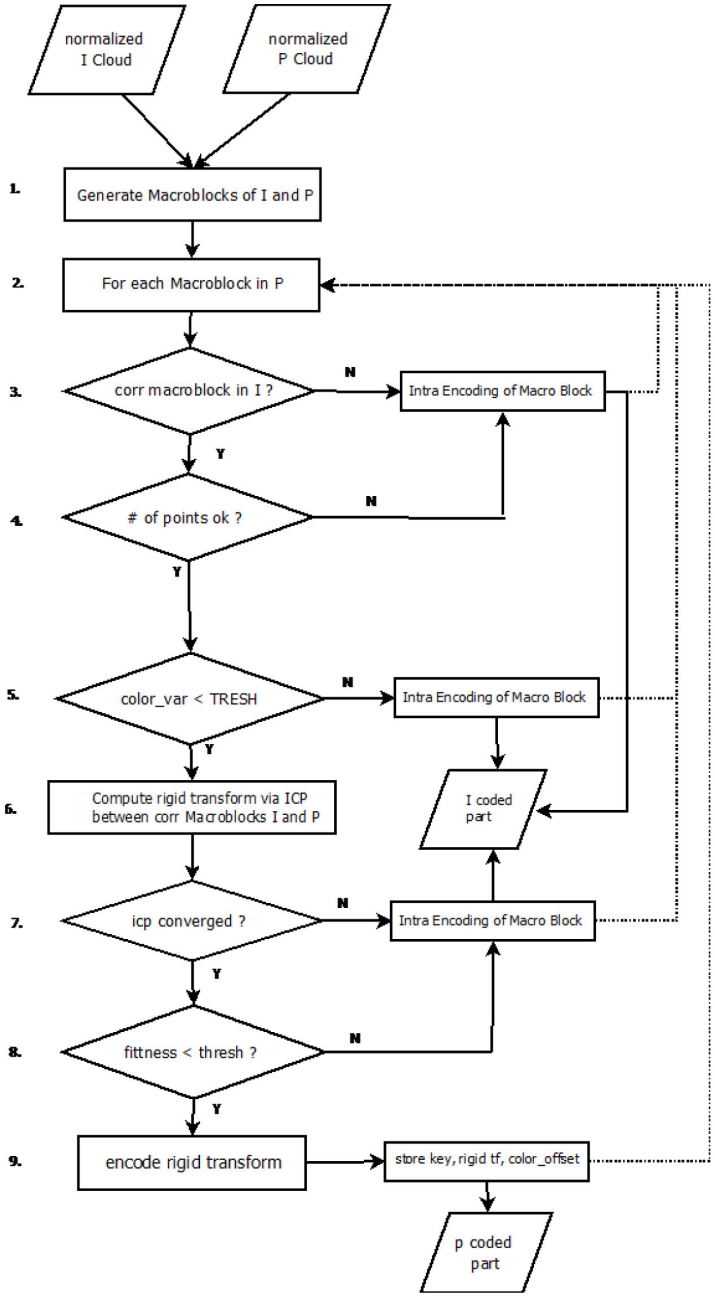


Figure 3.3: Inter frame prediction in the MPEG Anchor [26]

are then scanned in depth first order, the images corresponding to the nodes are stacked together and compressed using JPEG compression. The bond between geometry and color is maintained as we use the same sort order to encode both given by formula 3.2.

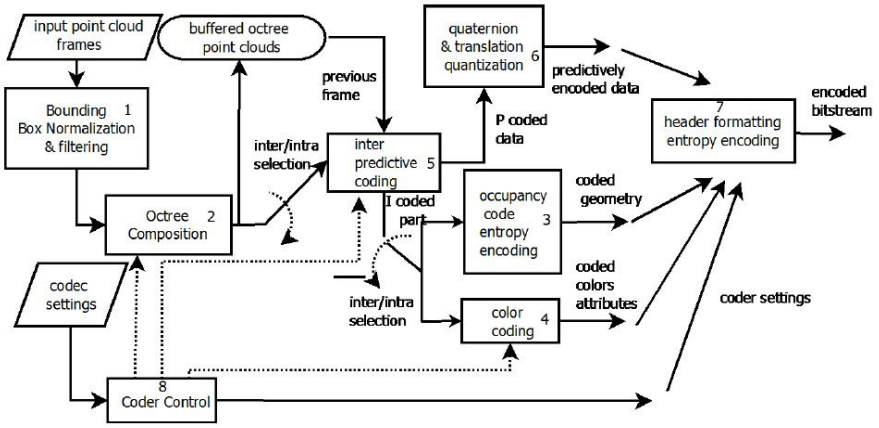


Figure 3.4: Schematic of the time varying point cloud compression codec [26]

3.4 Enhancement layer inter prediction

Inter-prediction in the enhancement layer is performed by using the double buffered octree structure from PCL. As mentioned in the previous section the octree subdivision criteria is changed from voxel occupancy to flatness in the enhancement layer. Octree nodes are terminated when they are sufficiently flat. The images generated while compressing the color attribute are associated to their corresponding nodes in the double buffer. If a node in the enhancement layer of the current frame has the same flatness profile as an enhancement layer node in the previous frame we mark the node for inter prediction. We compare the flatness θ as well as the eigen values of the covariance matrix.

The node is identified by the k(x,y,z) macroblock key associated with it. Each component of the key is stored using 16 bit integer values. For enhancement layer nodes that are marked for prediction, the key of the corresponding node from the previous frame is added to the bitstream. While decoding the 2D image of colors, transformed geometry coordinates and the quaternion associated with the macroblock from the previous frame is reused for the current frame.

The decoder then skips processing these nodes and places them in the depth first scan order. They are then recombined with the intra coded blocks and the decoded point cloud is constructed. The algorithm we implemented is shown in figure 3.5. Some of the modules are explained in the following subsections.

3.4.1 Frame header

At the start of the bit stream associated with every frame we add the frame header. This is used to apply pre-specified codec settings. The settings are read from a configuration file that is required as one of the inputs to the codec. The frame header contains all the fields required by the MPEG anchor such as the bit allocation for macroblock size, prediction method. color bit allocation and mapping mode. We extend the header to include settings

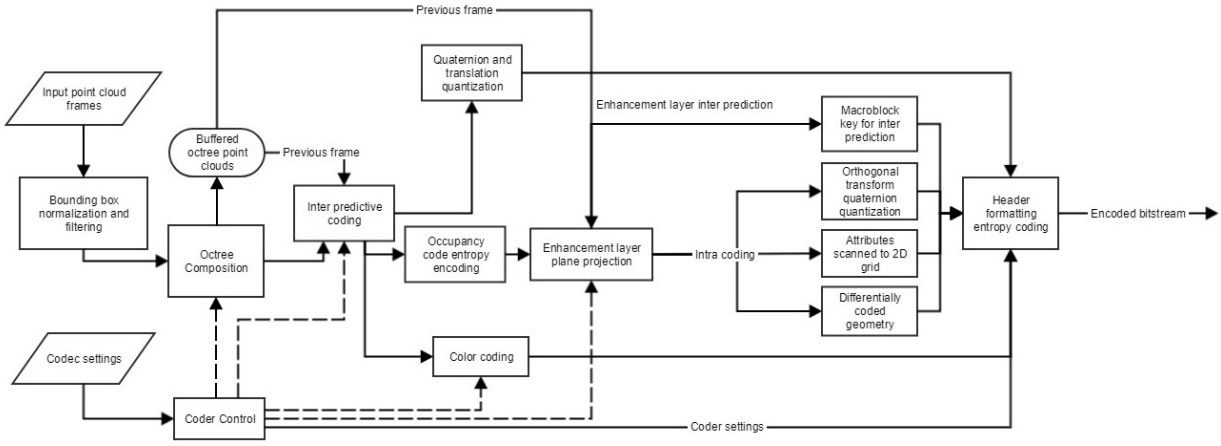


Figure 3.5: Enhancement layer inter prediction added to the MPEG anchor

for the enhancement layer. This includes the level of detail where the enhancement layer begins, the flatness threshold θ used during subdivision and lastly the mapping mode used during raster scan of color attributes to 2D images in the enhancement layer.To generate the results presented in the next chapter we used the JPEG lines mapping scheme implemented in the anchor. We also include the threshold values used while comparing the flatness of corresponding enhancement layer nodes in successive frames for inter prediction.

3.4.2 Principal Component Analysis (PCA)

The plane projection done in the enhancement layer is done using PCA. Principal component analysis uses a set of orthogonal transformations to convert a set of linearly correlated input to linearly uncorrelated variables. We implement PCA by using the Eigen library from PCL. The covariance matrix is calculated and eigen decomposition is performed for all points within an enhancement layer node. Each node can be processed independently and therefore can be run in parallel. We use the Open Multi-Processing API to implement parallelization. We sort the eigenvalues of a flat enhancement layer node in ascending order, we then use the corresponding eigenvectors to form the rotation quaternion needed for plane projection. The geometry of points after rotation are stored by differentially coding the coordinates and applying a quantization scheme. The amount of information retained per coordinate, after quantization is proportional to the eigenvalues corresponding to them. The same sort order is maintained for geometry and attributes.

3.4.3 Quaternion coding

Quaternions are a number system that extend complex numbers. They can be represented in the form a + bi + cj + dk. Where a,b,c and d are real numbers and i, j, k are quaternion units. The properties of quaternions are shown in equation 3.3. A crucial property of quaternions

is that quaternion multiplication is noncommutative. The relationship between i, j and k are thus similar to the cross product rules for unit cartesian vectors. As a result quaternions can be used to efficiently express a rotation in three dimensional space. In our implementation we store the orthogonal transformation used for plane projection in the form of a quaternion. This is then quantized and stored as three numbers occupying 16 bits each. Each quaternion is associated with the macroblock key of the corresponding enhancement layer node. During the decoding process we use PCL functions and the quaternion to rotate the coordinates back to their original values.

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.3}$$

Chapter 4

Results

In this chapter we present the results from evaluating our codec using the framework described in the previous chapter. The research questions we hoped to answer are first, to identify modifications that can be made to existing octree based point cloud codecs to compress new dense and photorealistic datasets. To address this we implemented the enhancement layer proposed by Ainala et al. [15]. We first validate our results against the findings reported by Ainala et al to ensure we get the same result.

The second research question we hoped to answer was to detect and exploit temporal redundancies in the enhancement layer. We detect these redundancies across successive frames by examining the local flatness profile of point cloud geometry and re-use the decoded stream for similar patches across frames. We first demonstrate the benefit of adding inter prediction to the enhancement layer by examining the incremental compression results with and without inter prediction. We then evaluate the overall effectiveness of our approach by rigorously evaluating the compression performance in the same manner as a proponent to the MPEG call for proposals. This procedure involves running our codec at four different bit rates specified by MPEG and calculating the average benefit to objective quality over the anchor at comparable bitrates.

4.1 Validation of the Implementation

We first validate our implementation of the proof of concept provided by Ainala et al [15] after the integration with the MPEG anchor. The validation is done using the skiing dataset scanned by Alexiadis et al in [16]. We set the parameters to the same level as Ainala at al used in their implementation [15]. The octree level of detail was set to 6 and the flatness θ was measured for each of the 22 occupied octree leaf nodes.

As seen from the figure 4.1 and figure 4.2 we can validate that the process of switching to the enhancement layer and flatness calculation is the same in our implementation.

Next we set the flatness criteria for the enhancement layer to 0.05 and the octree depth at which we switch to the enhancement layer to 6. For the same skiing dataset we record the size of the encoded stream and the geometric distortion for the stream.

The results from table 4.1 indicate that that our implementation is similar, the difference

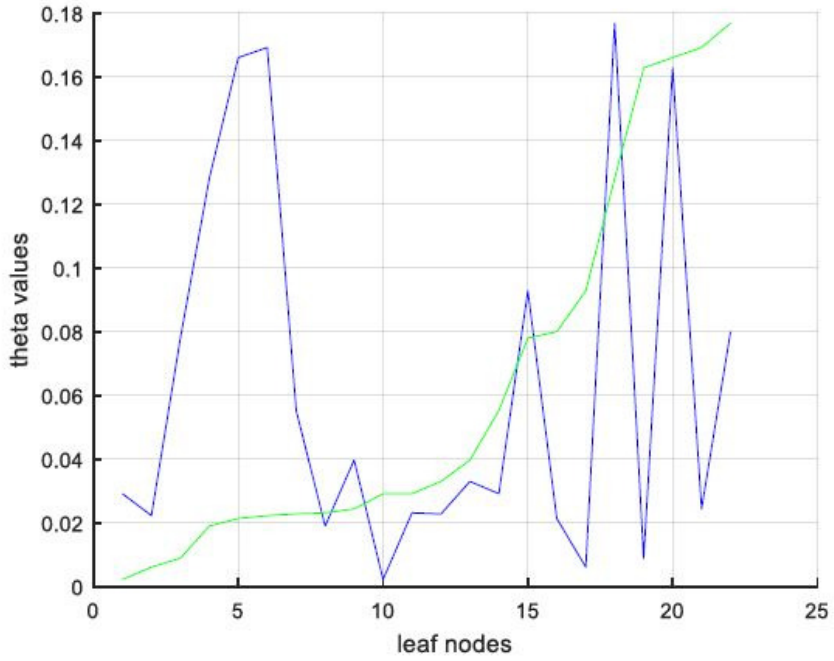


Figure 4.1: Distribution of flatness θ across the nodes of the octree for the skiing dataset from Ainala et al implementation [15]

can be attributed to changes in the MPEG anchor and the slightly different configurations for the octree layer.

Codec used	Size of compressed output (bytes)	Geometry D1 distortion PSNR (dB)
MPEG Anchor	86893	77
Ainala et al [15] enhancement layer	49305	73
Our implementation	48756	78

Table 4.1: Compression results for the skiing dataset [16] using the MPEG anchor, Ainala et al's implementation [15] and out implementation

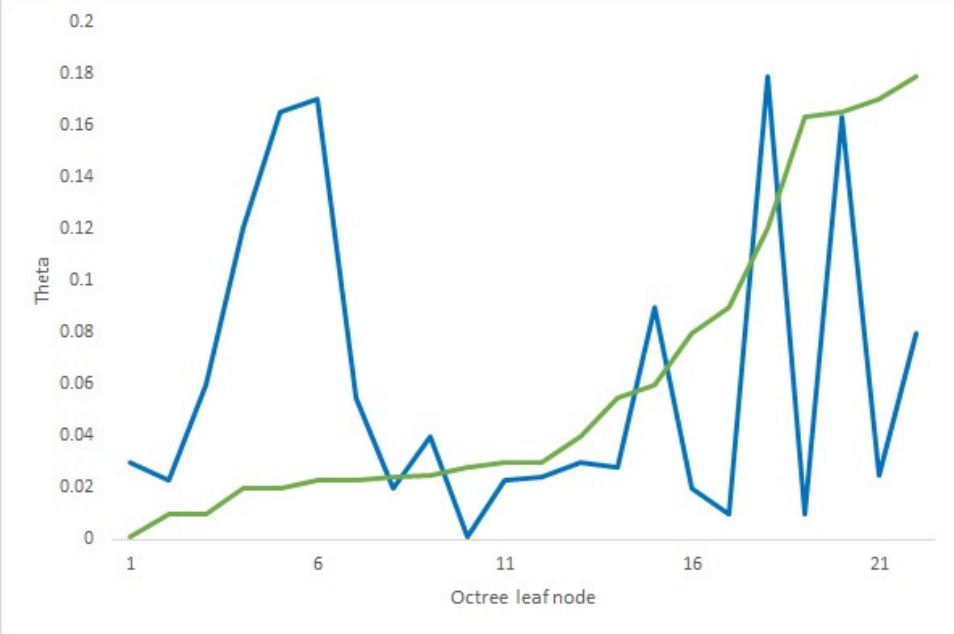


Figure 4.2: Distribution of flatness θ across the nodes of the octree for the skiing dataset from our implementation

4.2 Improvement of the Enhancement Layer

The second contribution of this thesis is the implementation of inter prediction to the enhancement layer. To measure the impact we use the top four bitrates of the longdress dataset [19]. We calculate the BD PSNR at the same bit rates to calculate the average improvement in PSNR offered by the enhancement layer with inter prediction as compared to the intra coded enhancement layer. The rate distortion curves are shown in figures 4.3 to 4.7. We observe a BD PSNR of 4.31% and 9.52% for geometry distortions calculated using the D1 and D2 metrics respectively. We also observe a BD PSNR of 9.92%, 12.84% and 15.90% in the Y, U And V color channels respectively. This demonstrates an incremental benefit to both geometry and attributes by employing inter prediction. We are able to retain more information across the board while maintaining similar bit rates.

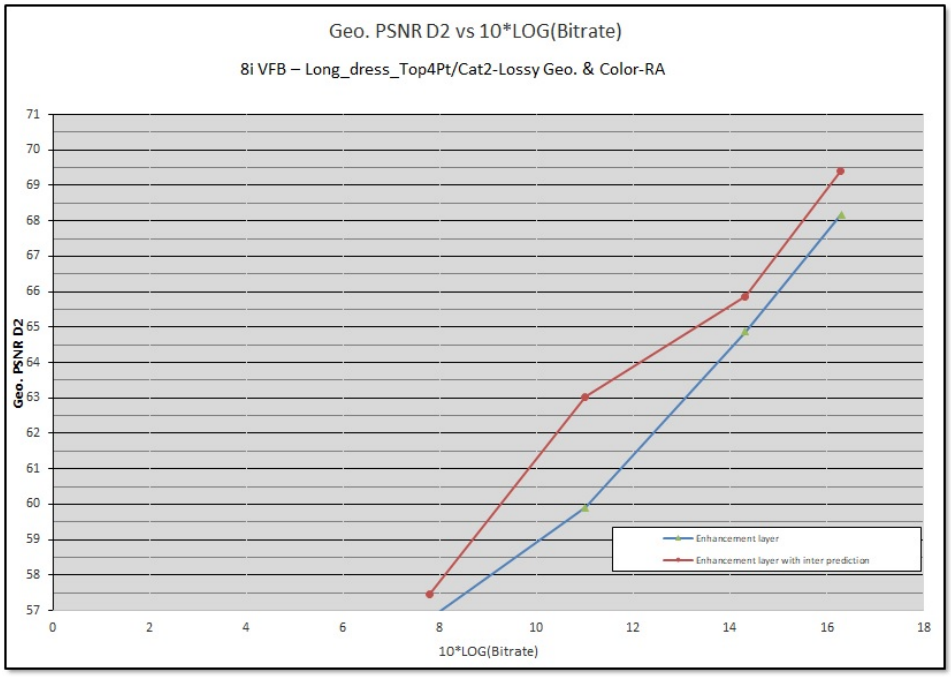


Figure 4.3: Geometry distortion PSNR using the D2 metric for the longdress dataset with inter prediction

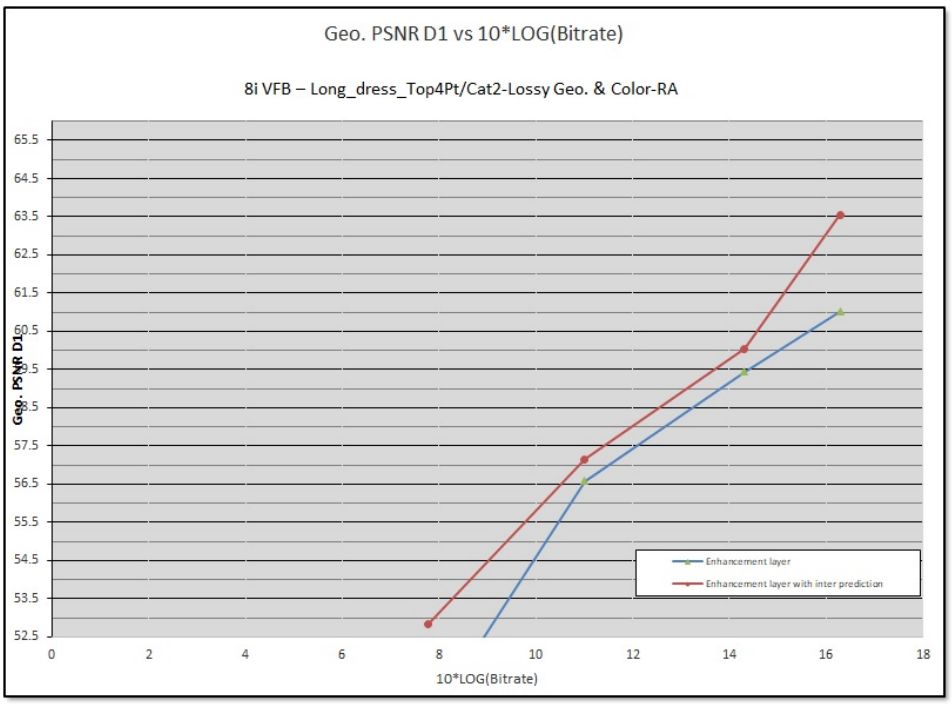


Figure 4.4: Geometry distortion PSNR using the D1 metric for the longdress dataset with inter prediction

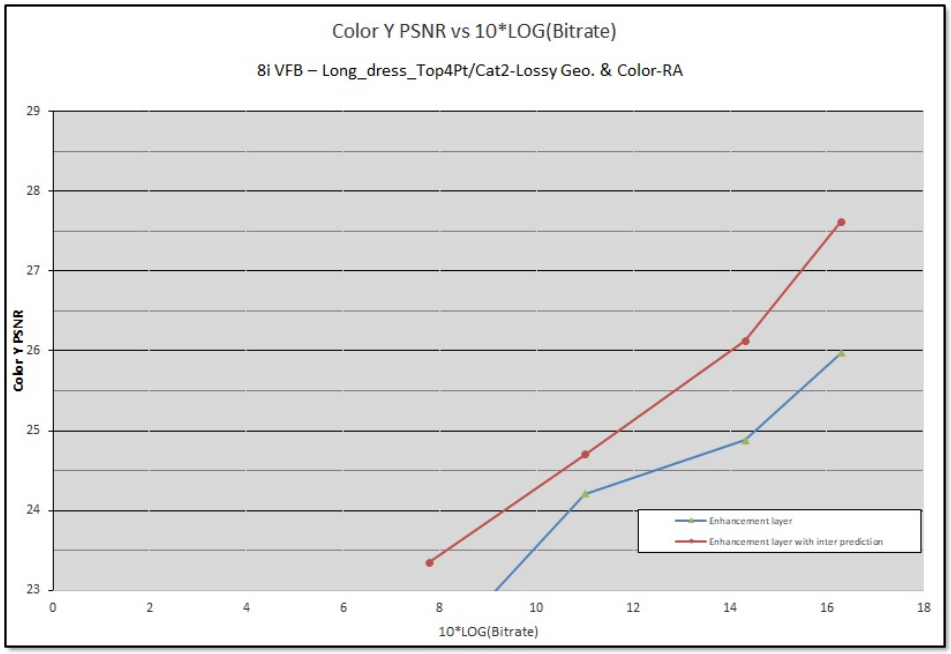


Figure 4.5: Attribute distortion PSNR for the Y channel in the longdress dataset with inter prediction

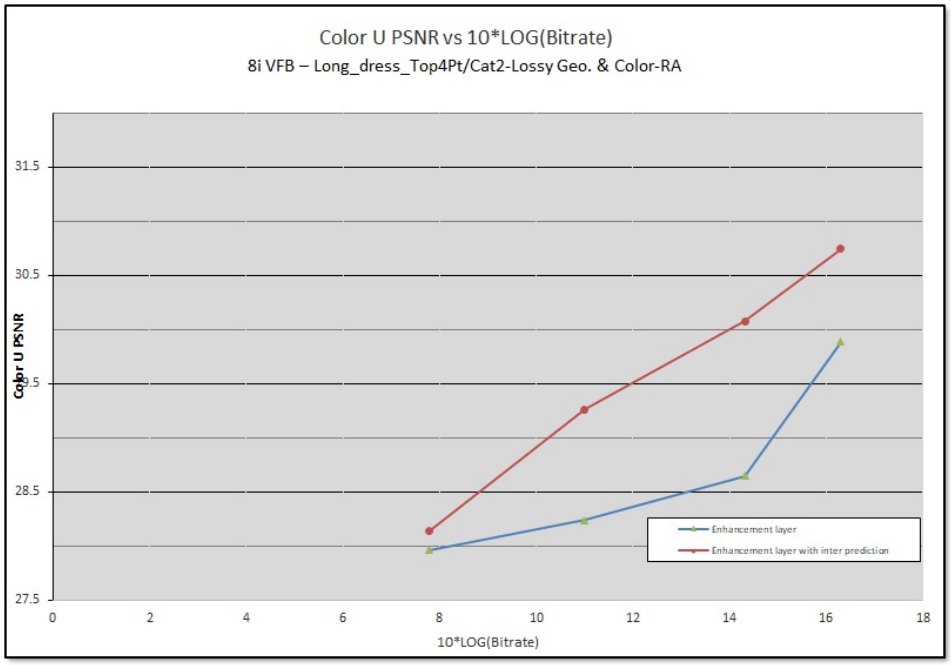


Figure 4.6: Attribute distortion PSNR for the U channel in the longdress dataset with inter prediction

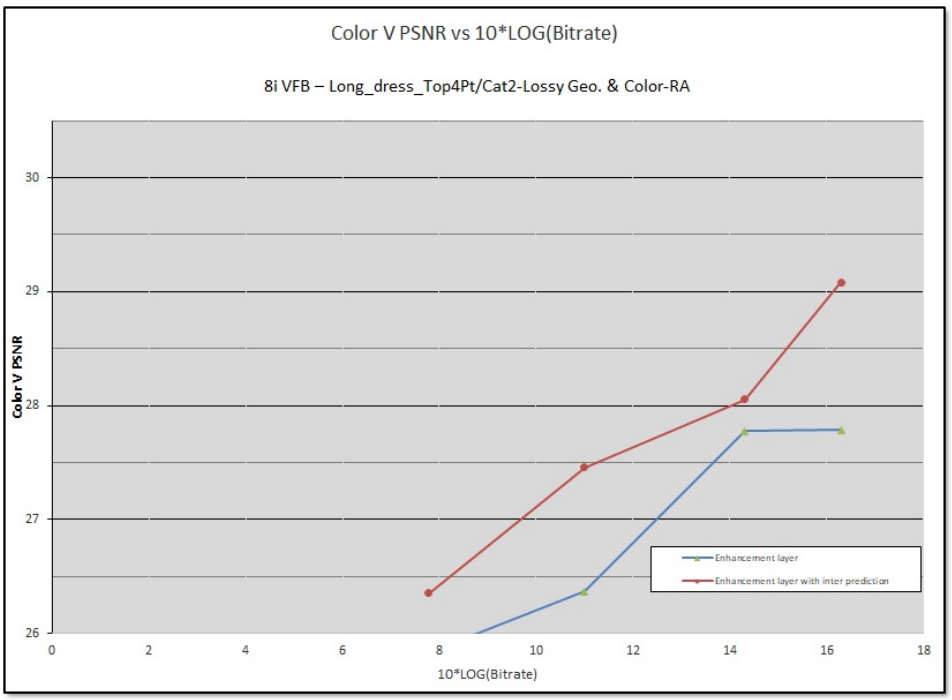


Figure 4.7: Attribute distortion PSNR for the V channel in the longdress dataset with inter prediction

4.3 Results with inter prediction

In order to evaluate the enhancement layer inter prediction codec we use four datasets from the MPEG CfP: Loot, Red and black, soldier and Queen [14].

We then use the evaluation spread sheet provided as part of the CfP. The octree and enhancement layer configuration is modified so that we generate a decoded point cloud at all the bit rates specified by the CfP with a 10% tolerance. We measure the D1 and D2 geometry distortion (RMSD) and the attribute/color distortion using the evaluation tool provided by MPEG. The results for our implementation are provided in table 4.3 and the results for the MPEG anchor are mentioned in table 4.2 for comparison.

The rate distortion curves are then plotted for geometry and each color channel separately. We use only the top 4 bitrates for each dataset as per the CfP. These have been shown in figures 4.8 to 4.32.

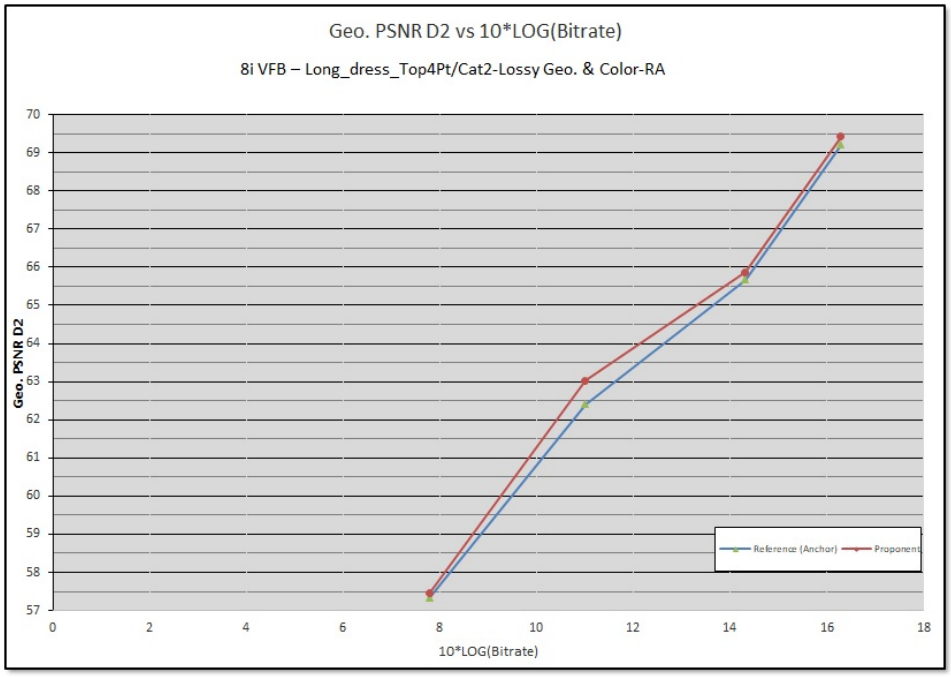


Figure 4.8: Geometry distortion PSNR using the D2 metric for the longdress dataset

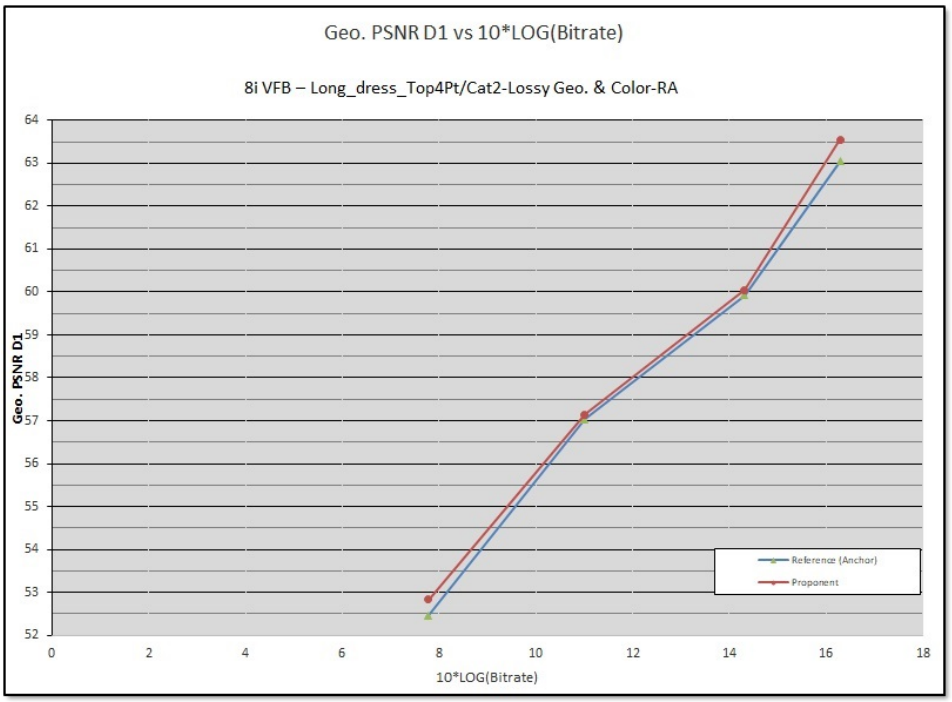


Figure 4.9: Geometry distortion PSNR using the D1 metric for the longdress dataset

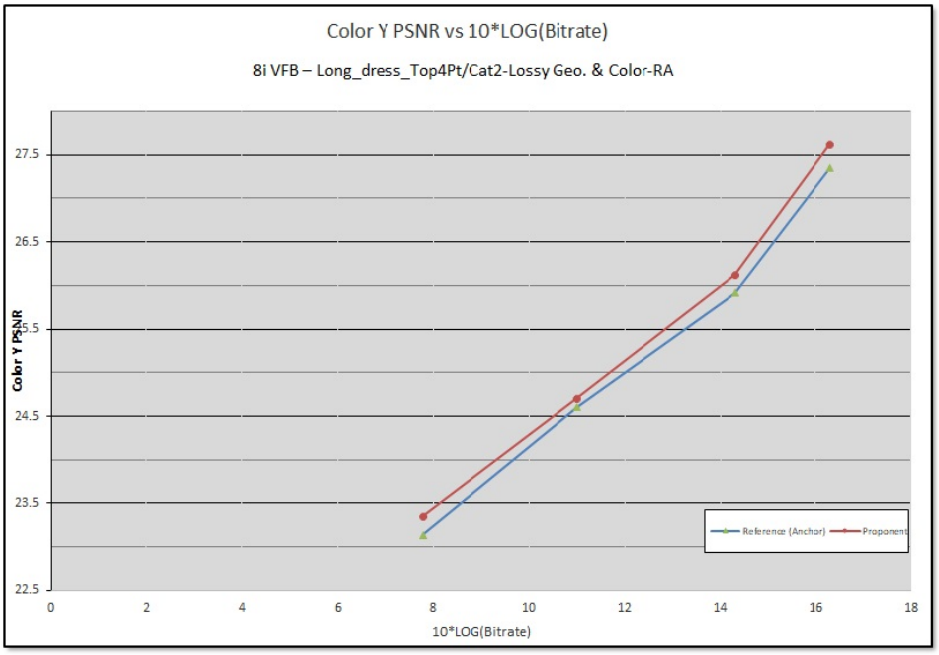


Figure 4.10: Attribute distortion PSNR for the Y channel in the longdress dataset

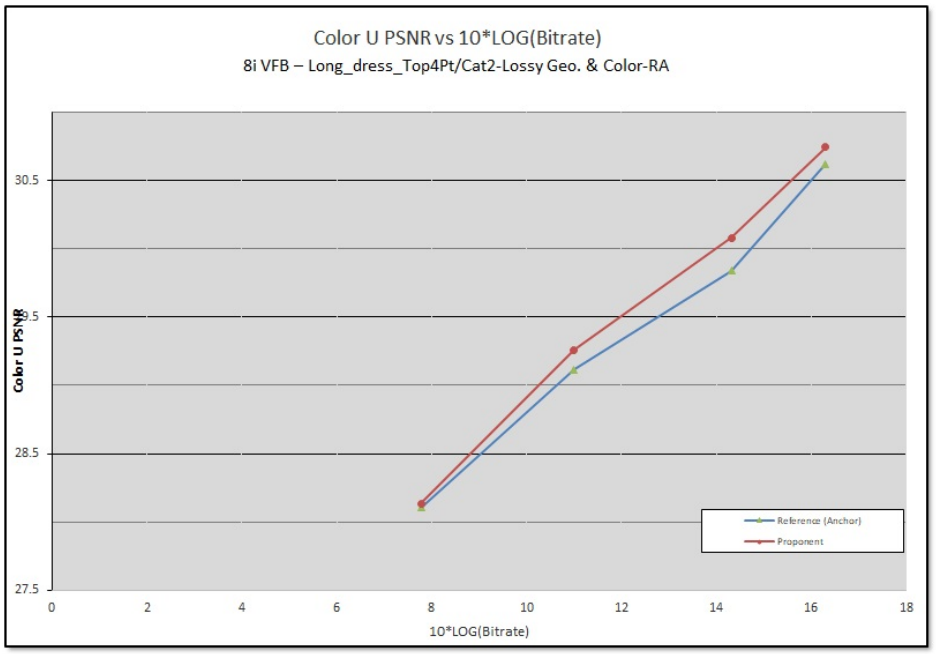


Figure 4.11: Attribute distortion PSNR for the U channel in the longdress dataset

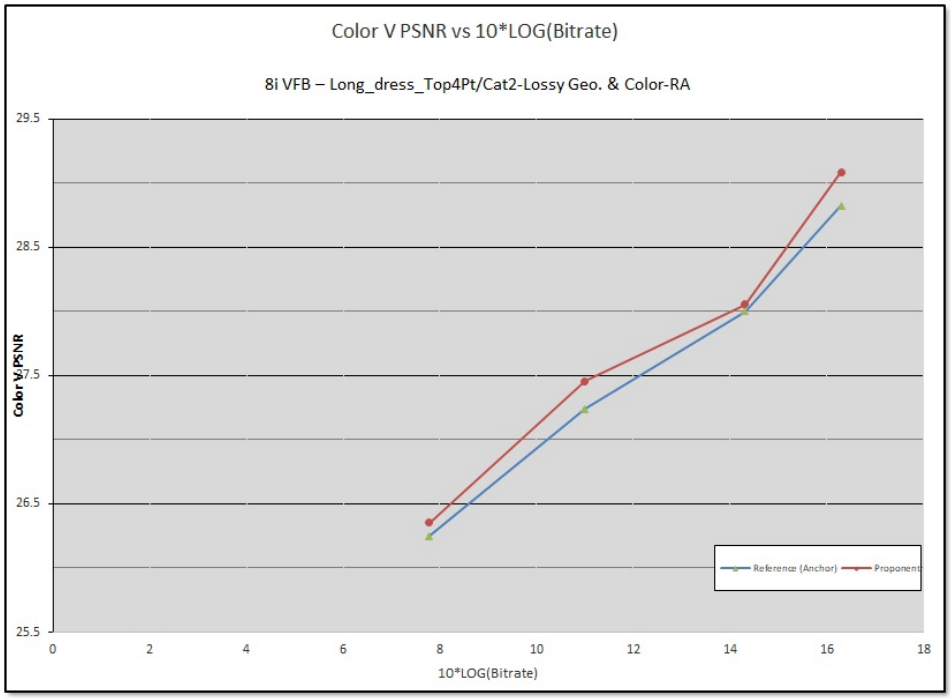


Figure 4.12: Attribute distortion PSNR for the V channel in the longdress dataset

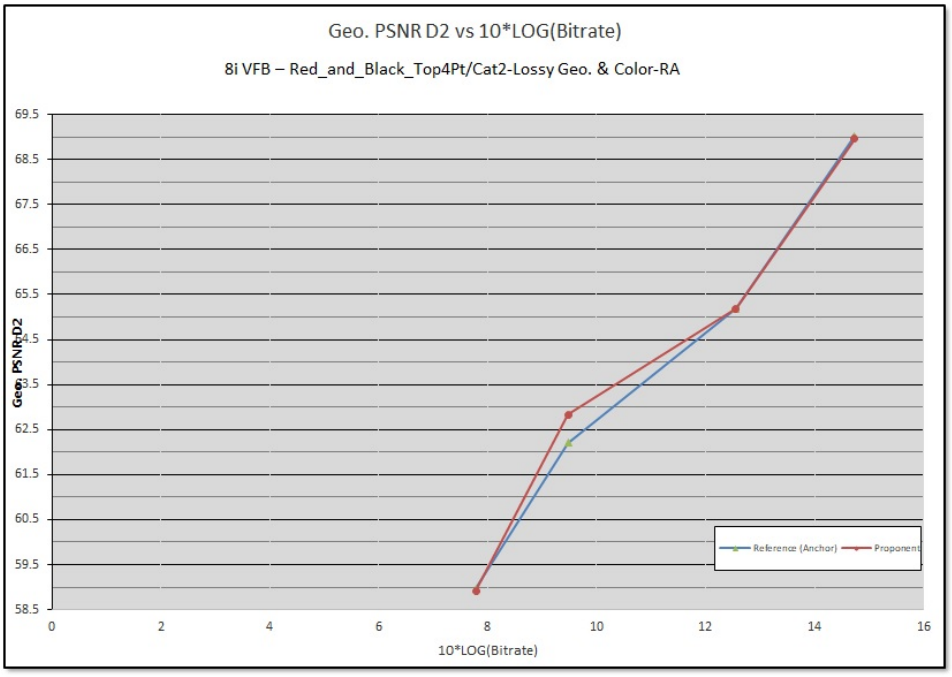


Figure 4.13: Geometry distortion PSNR using the D2 metric for the redandblack dataset

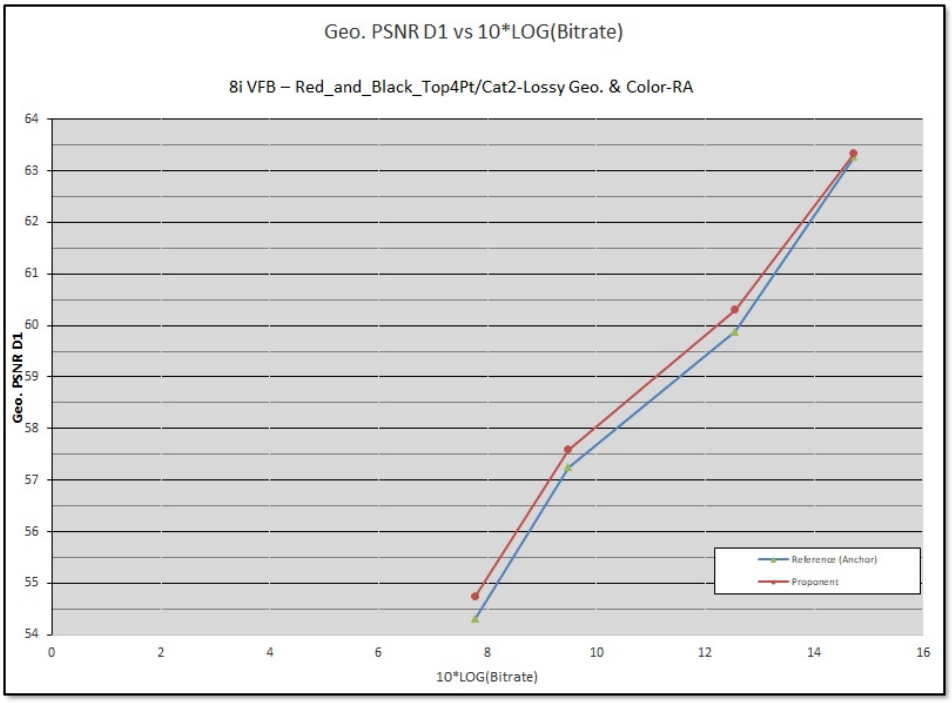


Figure 4.14: Geometry distortion PSNR using the D1 metric for the redandblack dataset

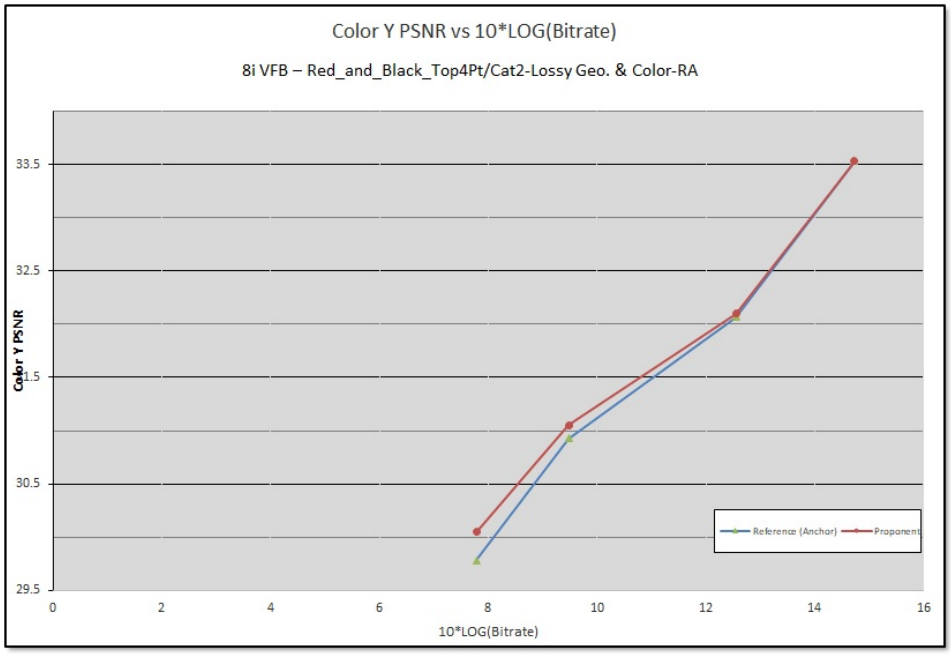


Figure 4.15: Attribute distortion PSNR for the Y channel in the redandblack dataset

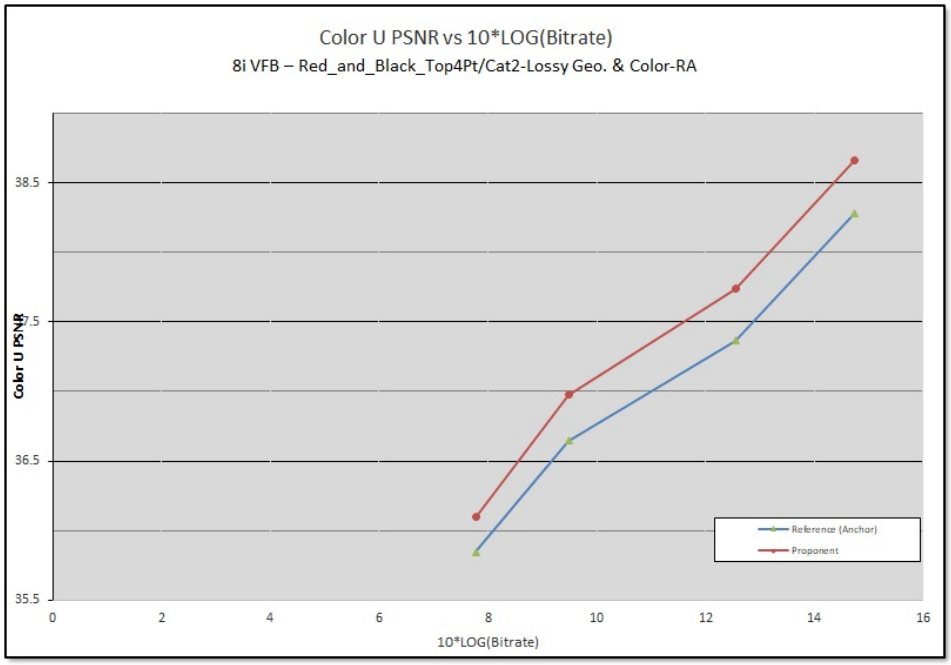


Figure 4.16: Attribute distortion PSNR for the U channel in the redandblack dataset

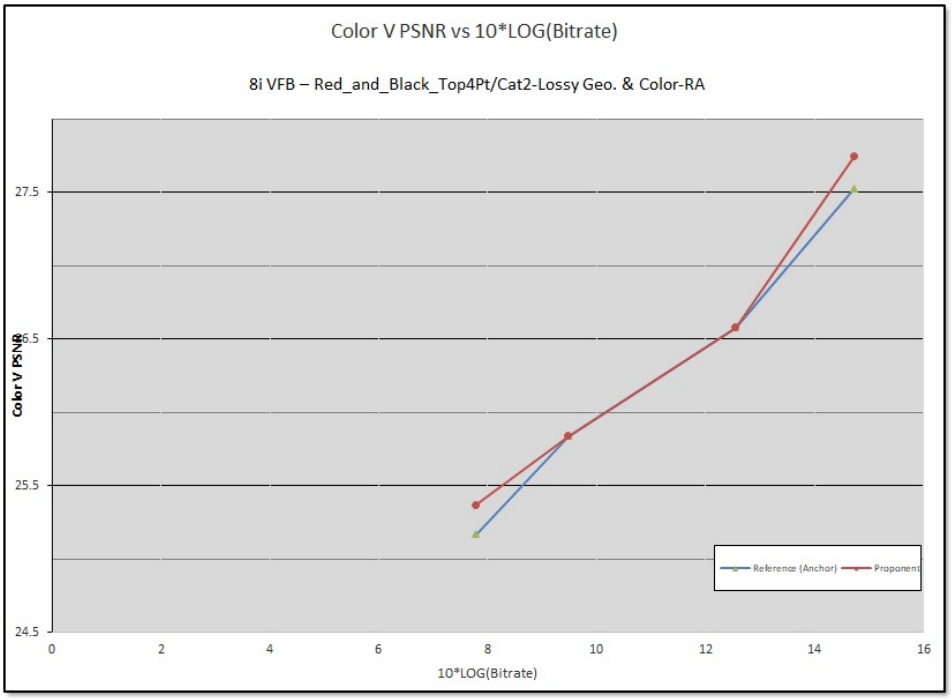


Figure 4.17: Attribute distortion PSNR for the V channel in the redandblack dataset

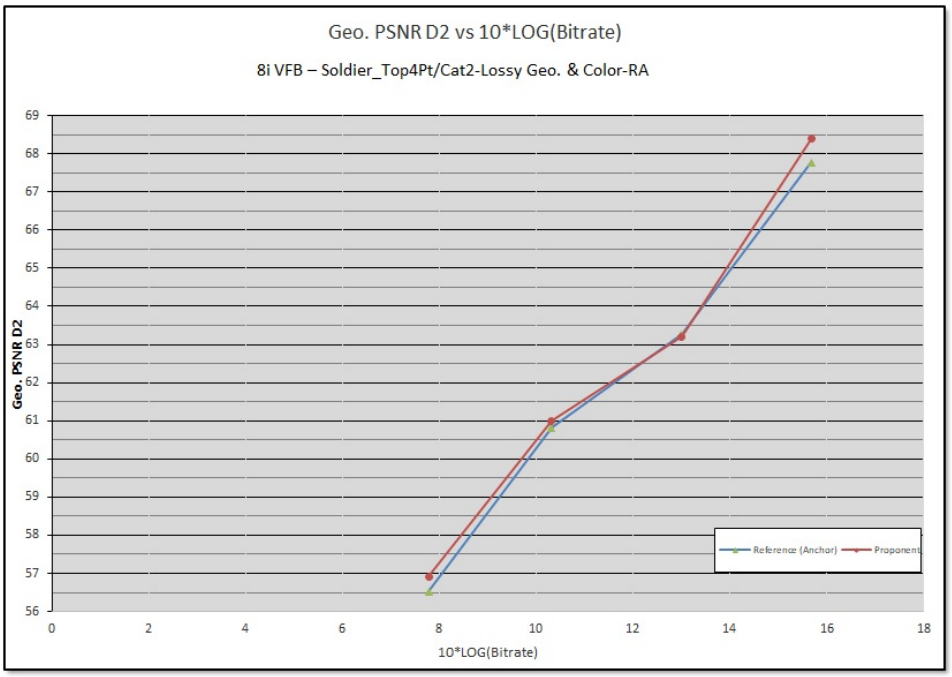


Figure 4.18: Geometry distortion PSNR using the D2 metric for the soldier dataset

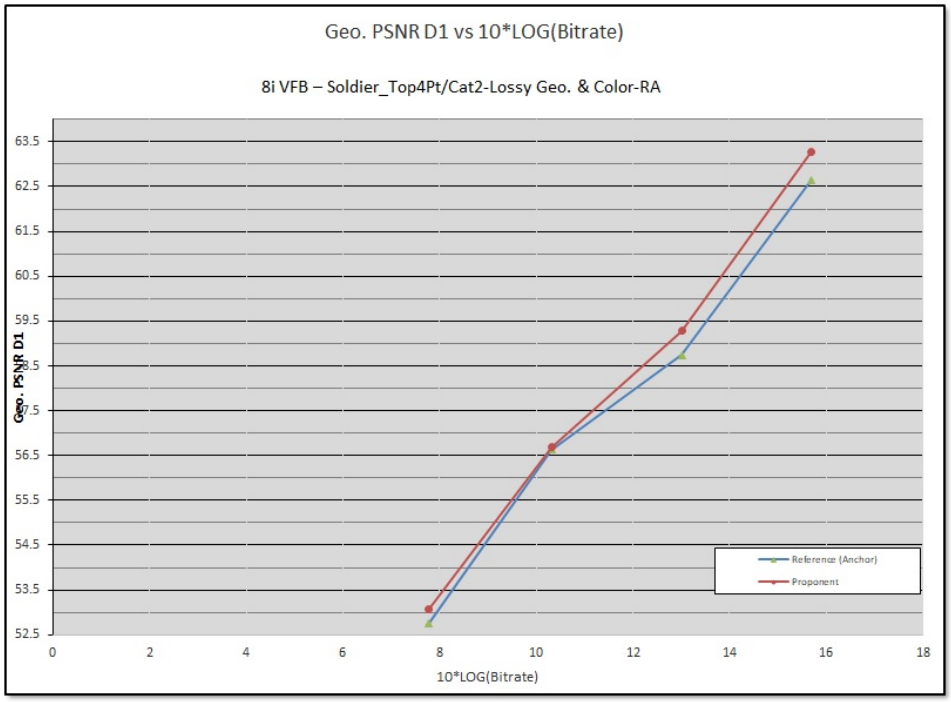


Figure 4.19: Geometry distortion PSNR using the D1 metric for the soldier dataset

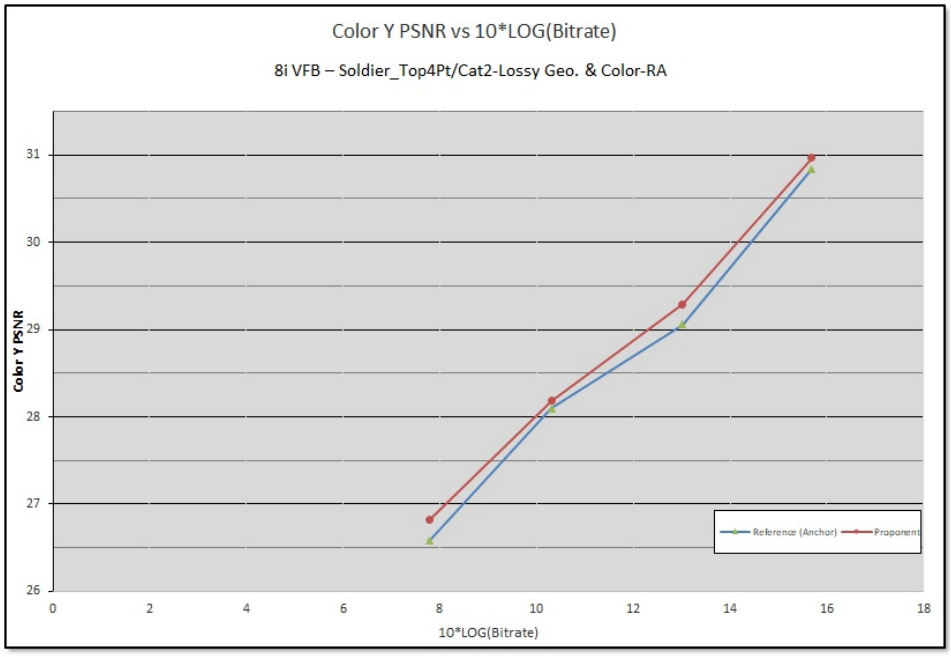


Figure 4.20: Attribute distortion PSNR for the Y channel in the soldier dataset

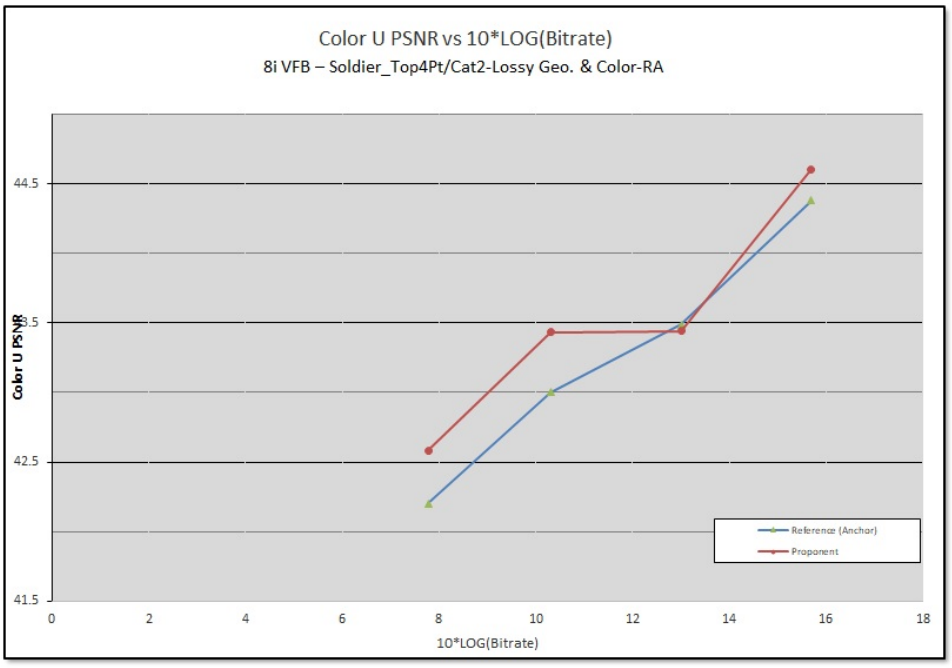


Figure 4.21: Attribute distortion PSNR for the U channel in the soldier dataset

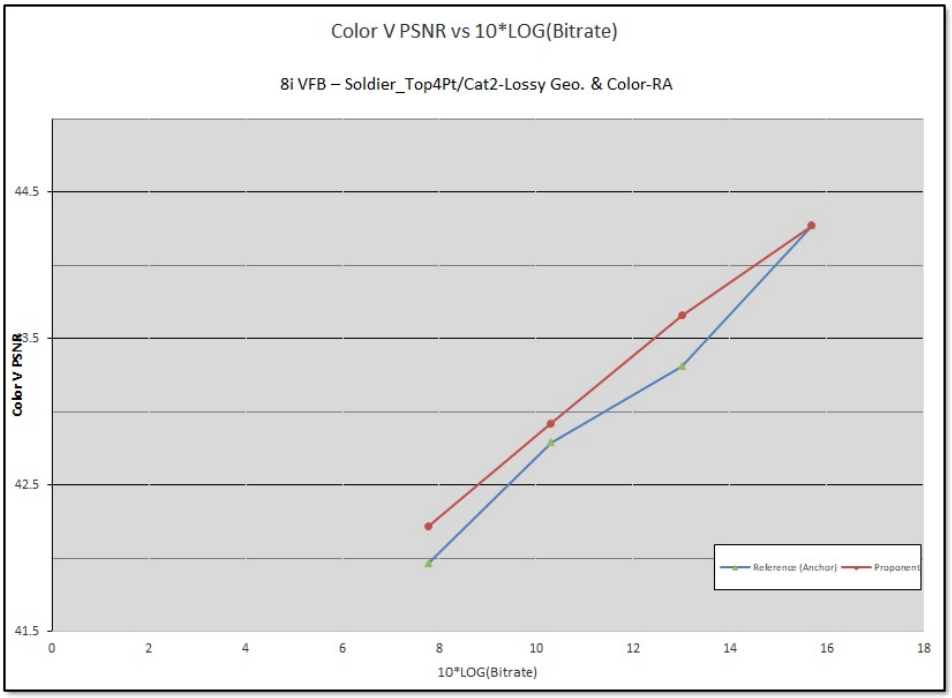


Figure 4.22: Attribute distortion PSNR for the V channel in the soldier dataset

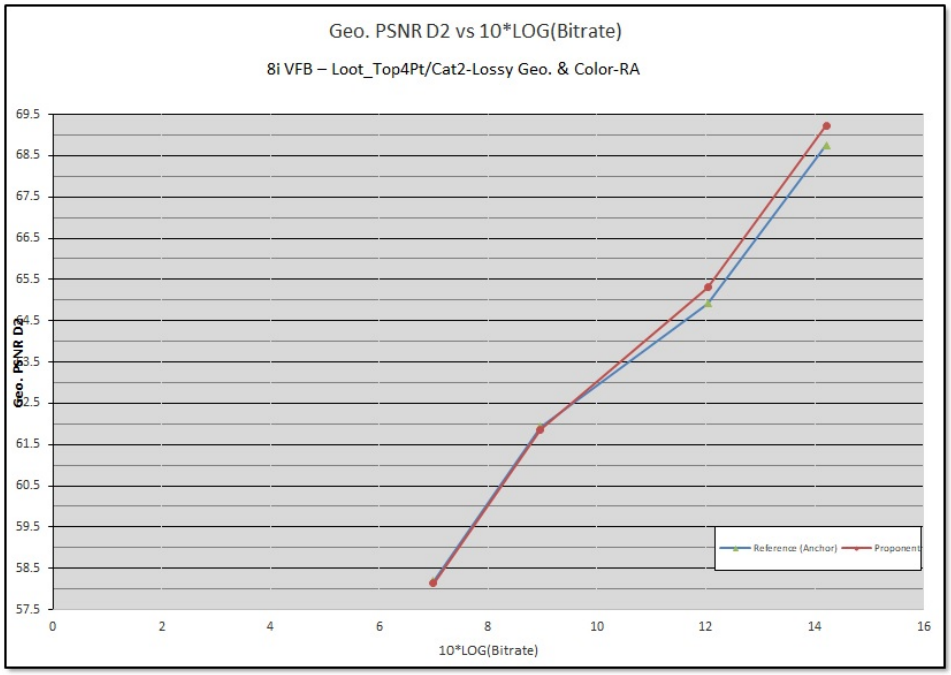


Figure 4.23: Geometry distortion PSNR using the D2 metric for the loot dataset

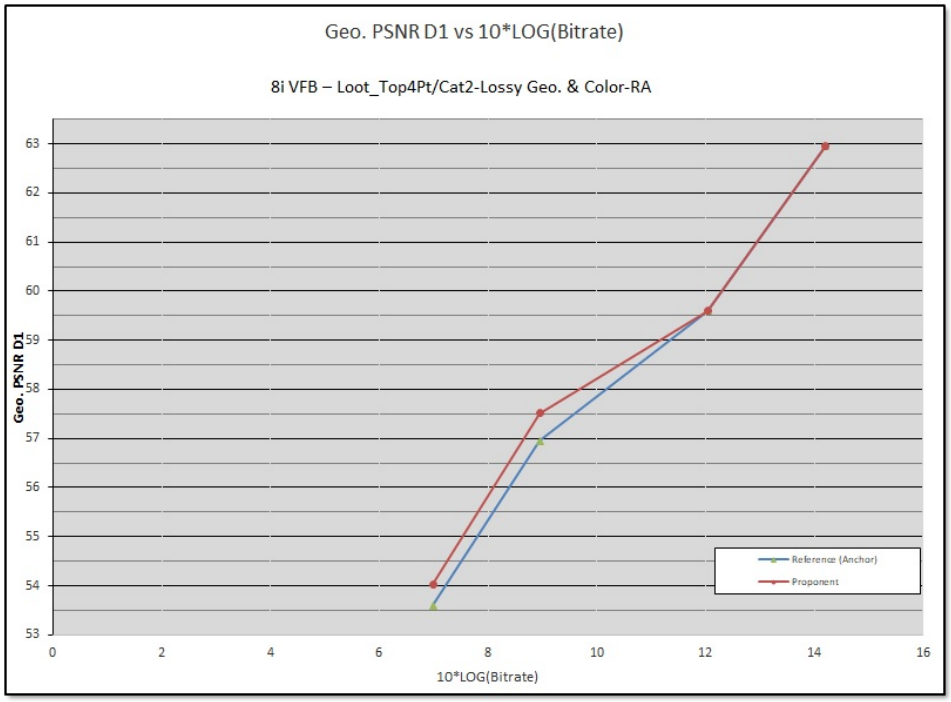


Figure 4.24: Geometry distortion PSNR using the D1 metric for the loot dataset

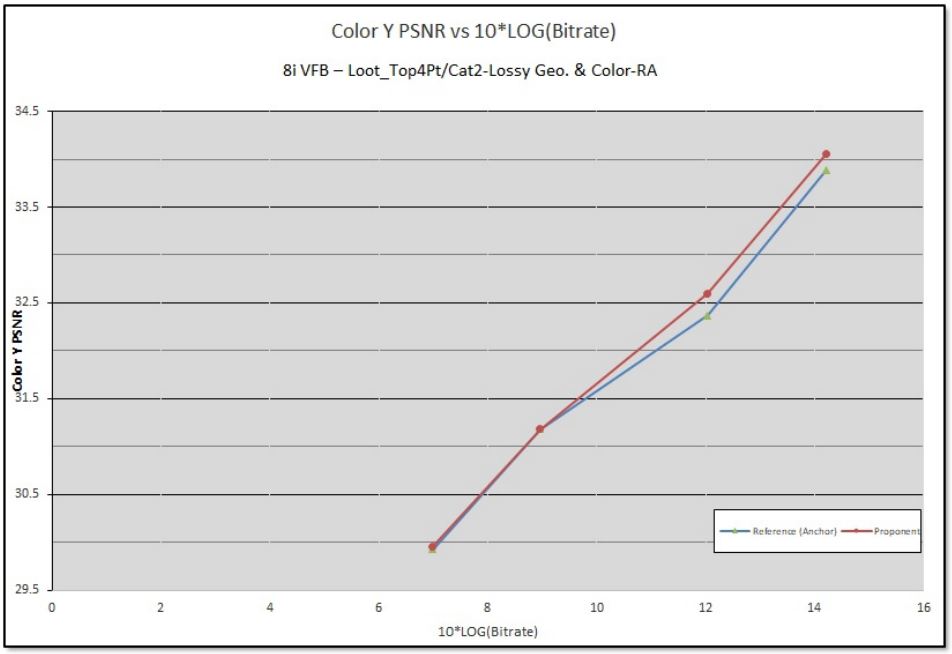


Figure 4.25: Attribute distortion PSNR for the Y channel in the loot dataset

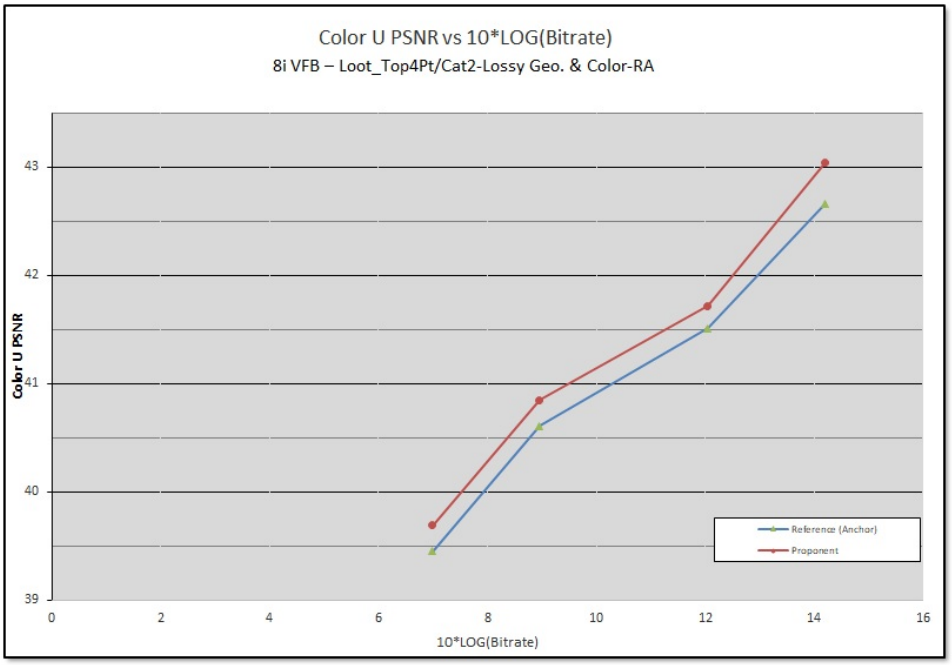


Figure 4.26: Attribute distortion PSNR for the U channel in the loot dataset

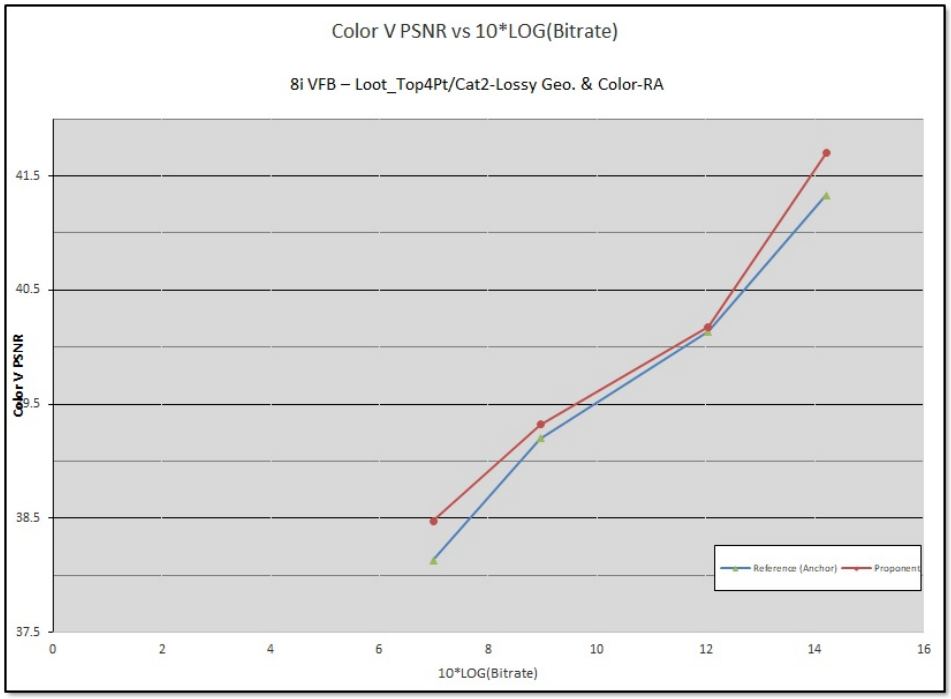


Figure 4.27: Attribute distortion PSNR for the V channel in the loot dataset

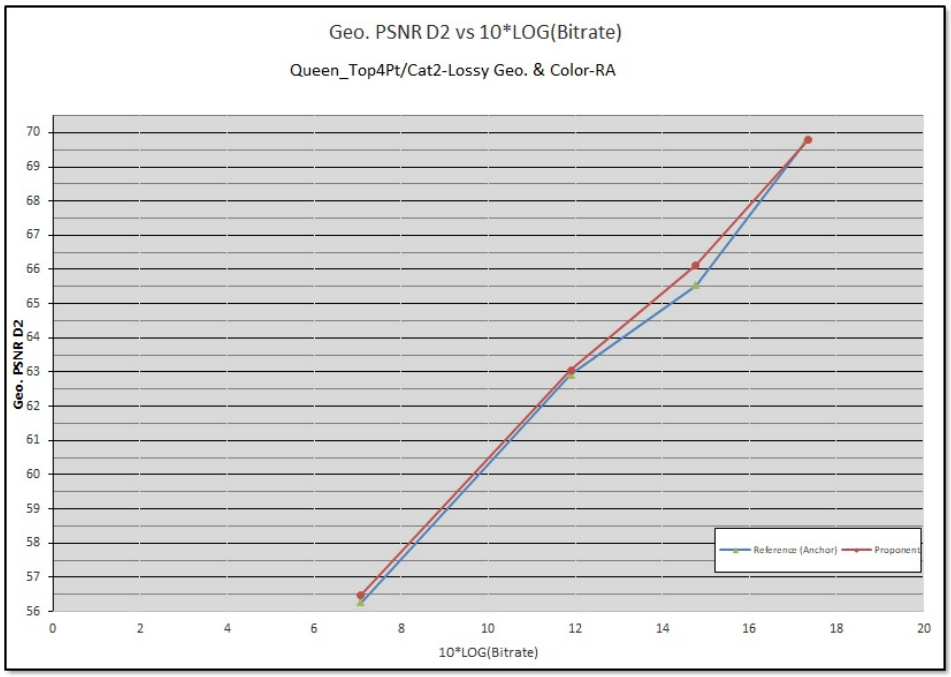


Figure 4.28: Geometry distortion PSNR using the D2 metric for the queen dataset

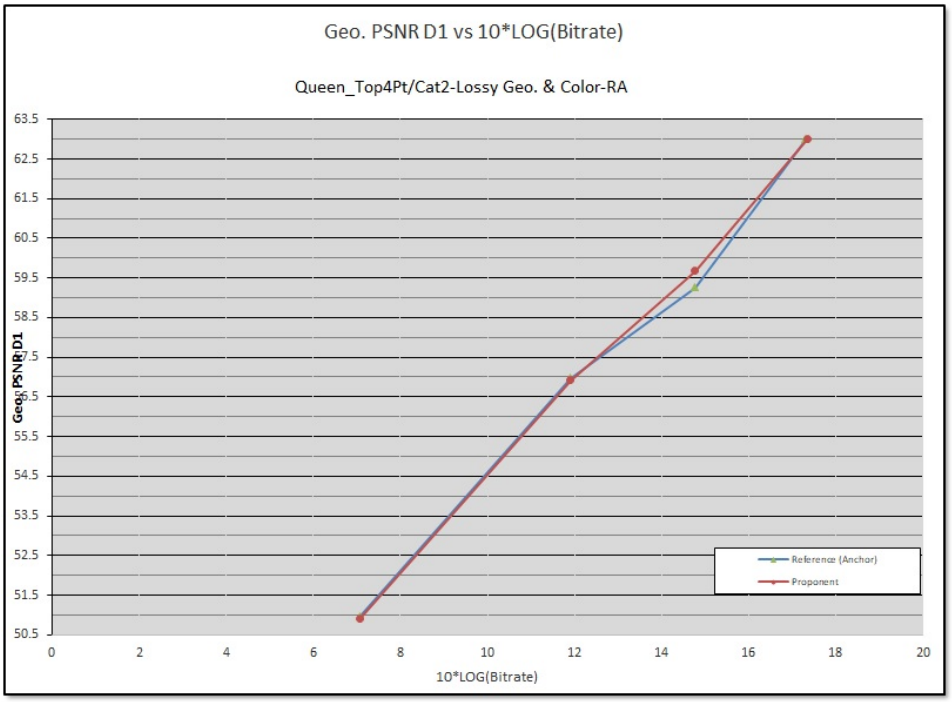


Figure 4.29: Geometry distortion PSNR using the D1 metric for the queen dataset

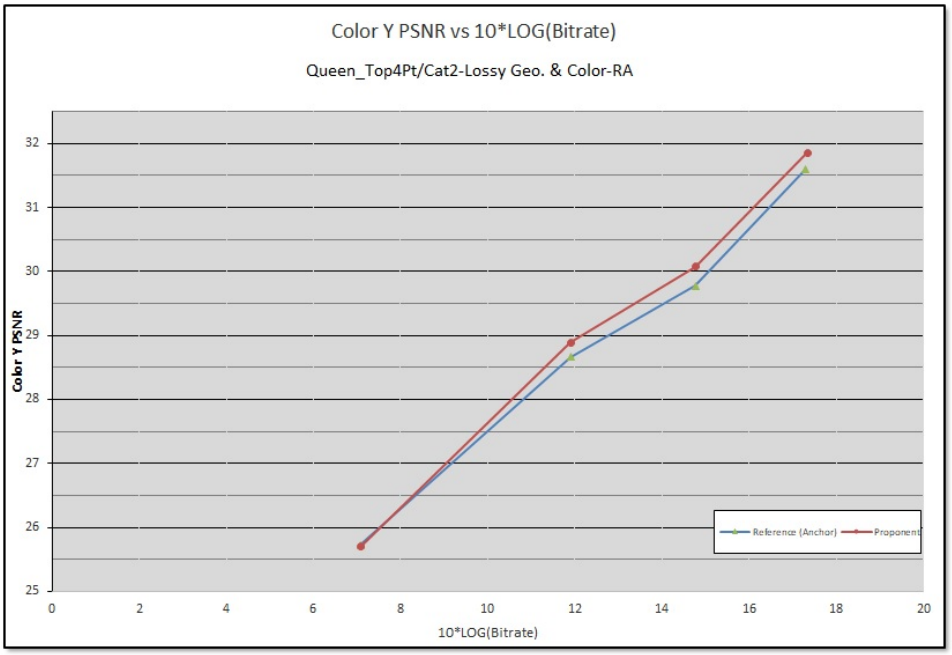


Figure 4.30: Attribute distortion PSNR for the Y channel in the queen dataset

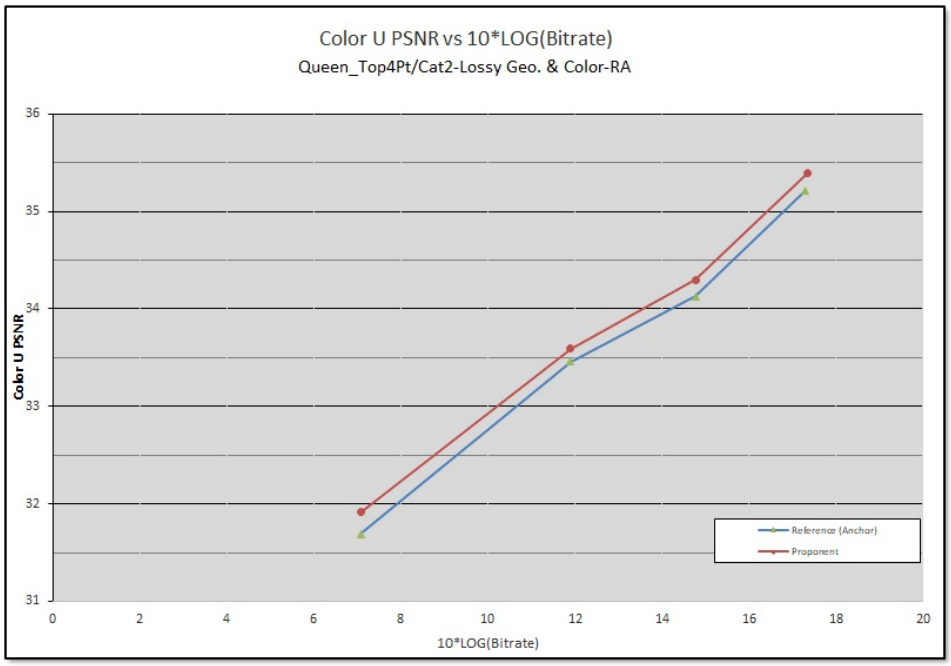


Figure 4.31: Attribute distortion PSNR for the U channel in the queen dataset

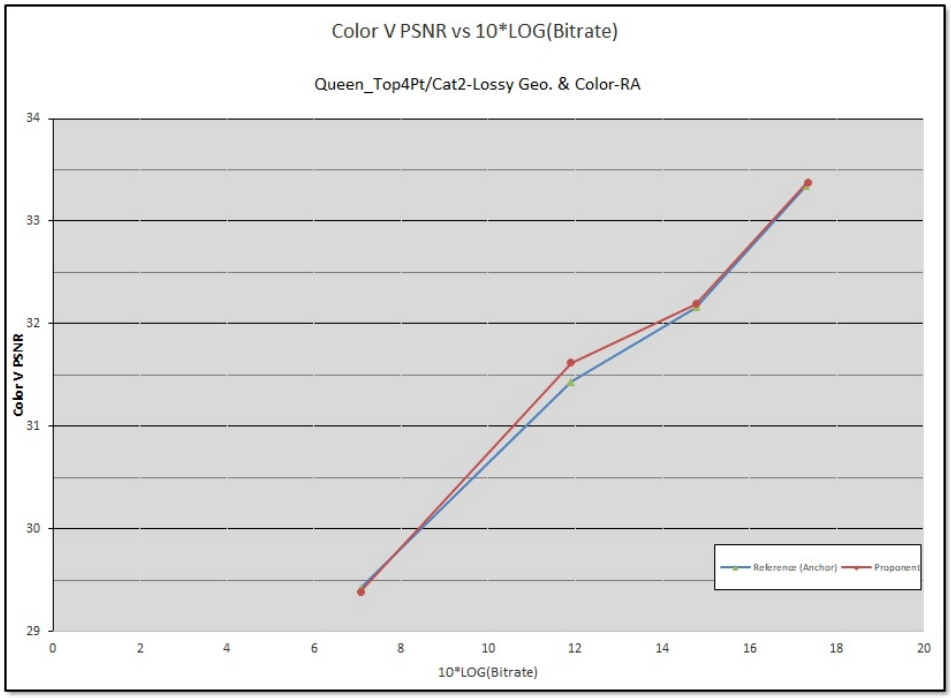


Figure 4.32: Attribute distortion PSNR for the V channel in the queen dataset

Stream Name	Rate (Mbit/s)	Geo. MSE D1	Geo. PSNR D1 (dB)	Geo. MSE D2	Geo. PSNR D2 (dB)	Color Y PSNR (dB)	Color U PSNR (dB)	Color V PSNR (dB)
P00S20C04R05	53.74	1.58	63.00	0.33	69.79	31.61	35.22	33.35
P00S20C04R04	30.00		59.26		65.53	29.78	34.13	32.16
P00S20C04R03	15.50	6.30	56.98	1.61	62.93	28.66	33.46	31.43
P00S20C04R02	5.10	25.13	50.97	7.46	56.26	25.72	31.69	29.42
P00S20C04R01	1.50	100.40	44.96	32.38	49.88	23.67	30.53	28.15
P00S21C04R05	26.36	1.59	62.96	0.42	68.75	33.88	42.66	41.33
P00S21C04R04	16.00		59.59		64.92	32.37	41.51	40.14
P00S21C04R03	7.87	6.34	56.95	2.02	61.92	31.19	40.61	39.20
P00S21C04R02	5.00		53.60		58.19	29.93	39.45	38.13
P00S21C04R01	2.71	25.40	50.92	9.44	55.22	28.92	38.53	37.28
P00S22C04R05	29.73	1.48	63.27	0.40	69.02	33.53	38.28	27.52
P00S22C04R04	18.00		59.88		65.18	32.07	37.36	26.58
P00S22C04R03	8.87	5.92	57.25	1.91	62.20	30.93	36.65	25.84
P00S22C04R02	6.00		54.32		58.97	29.78	35.85	25.17
P00S22C04R01	2.97	23.67	51.23	8.79	55.57	28.57	35.00	24.46
P00S23C04R05	37.03	1.71	62.64	0.53	67.77	30.84	44.38	44.27
P00S23C04R04	20.00		58.75		63.26	29.06	43.49	43.31
P00S23C04R03	10.74	6.83	56.63	2.61	60.81	28.09	43.00	42.79
P00S23C04R02	6.00		52.76		56.52	26.58	42.20	41.97
P00S23C04R01	3.36	27.35	50.60	12.18	54.13	25.74	41.76	41.51
P00S24C04R05	42.60	1.56	63.05	0.38	69.21	27.34	30.62	28.82
P00S24C04R04	27.00		59.92		65.67	25.92	29.84	28.00
P00S24C04R03	12.60	6.23	57.03	1.81	62.40	24.60	29.12	27.24
P00S24C04R02	6.00		52.47		57.33	23.14	28.11	26.25
P00S24C04R01	3.87	24.93	51.00	8.48	55.69	22.67	27.78	25.93

Table 4.2: Objective quality results for the MPEG anchor

Stream Name	Rate (Mbit/s)	Geo. MSE D1	Geo. PSNR D1 (dB)	Geo. MSE D2	Geo. PSNR D2 (dB)	Color Y PSNR (dB)	Color U PSNR (dB)	Color V PSNR (dB)
PnnS20C04R05	54.23	1.59	63.00	0.33	69.79	31.86	35.39	33.38
PnnS20C04R04	30.00	0.00	59.68	0.00	66.12	30.08	34.30	32.19
PnnS20C04R03	15.50	6.33	56.92	1.61	63.05	28.89	33.59	31.62
PnnS20C04R02	5.10	25.23	50.92	7.49	56.48	25.70	31.91	29.39
PnnS20C04R01	1.50	101.00	45.41	32.44	49.93	23.71	30.68	28.15
PnnS21C04R05	26.36	1.60	62.96	0.42	69.24	34.05	43.05	41.70
PnnS21C04R04	16.00	0.00	59.59	0.00	65.31	32.60	41.72	40.18
PnnS21C04R03	7.87	6.36	57.52	2.04	61.86	31.19	40.85	39.32
PnnS21C04R02	5.00	0.00	54.02	0.00	58.14	29.96	39.69	38.48
PnnS21C04R01	2.71	25.50	50.97	9.49	55.72	29.09	38.49	37.32
PnnS22C04R05	29.73	1.49	63.33	0.40	68.95	33.53	38.66	27.74
PnnS22C04R04	18.00	0.00	60.30	0.00	65.18	32.10	37.74	26.58
PnnS22C04R03	8.87	5.93	57.60	1.92	62.83	31.05	36.98	25.84
PnnS22C04R02	6.00	0.00	54.76	0.00	58.92	30.05	36.10	25.37
PnnS22C04R01	2.97	23.72	51.49	8.83	55.85	28.77	35.03	24.55
PnnS23C04R05	37.03	1.73	63.27	0.53	68.38	30.97	44.60	44.27
PnnS23C04R04	20.00	0.00	59.28	0.00	63.20	29.29	43.44	43.66
PnnS23C04R03	10.74	6.87	56.69	2.62	61.00	28.18	43.43	42.92
PnnS23C04R02	6.00	0.00	53.07	0.00	56.92	26.82	42.58	42.22
PnnS23C04R01	3.36	27.38	50.55	12.19	54.07	25.80	41.72	41.92
PnnS24C04R05	42.60	1.57	63.55	0.38	69.42	27.62	30.74	29.08
PnnS24C04R04	27.00	0.00	60.04	0.00	65.87	26.12	30.08	28.05
PnnS24C04R03	12.60	6.24	57.14	1.81	63.03	24.70	29.26	27.46
PnnS24C04R02	6.00	0.00	52.84	0.00	57.44	23.35	28.14	26.35
PnnS24C04R01	3.87	25.07	51.31	8.55	55.75	22.78	27.84	26.03

Table 4.3: Objective quality results for enhancement layer codec with inter prediction

		Top 4 points BD Rate (Piecewise Cubic)				
Category	Dataset	Geo. PSNR D1 (dB)	Geo. PSNR D2 (dB)	Color Y PSNR (dB)	Color U PSNR (dB)	Color V PSNR (dB)
A	Queen	-0.77%	-3.86%	-7.89%	-10.08%	-7.76%
A	8i VFB Loot	-4.78%	-2.70%	-4.74%	-11.81%	-5.50%
A	8i VFB Red_and_Black	-6.47%	-3.62%	-3.30%	-21.32%	-1.53%
A	8i VFB Soldier	-5.89%	-1.74%	-7.14%	-16.79%	-16.74%
B	8i VFB Long_dress	-2.97%	-6.78%	-7.26%	-11.71%	-11.23%
Overall	Avg. for Class A	-4.48%	-2.98%	-5.77%	-15.00%	-7.88%
Overall	Avg. for Class B	-2.97%	-6.78%	-7.26%	-11.71%	-11.23%
Overall	Overall Average	-4.17%	-3.74%	-6.07%	-14.34%	-8.55%

Table 4.4: BD PSNR results summary

Using the rate distortion results shown above we calculate the Bjontegaard Delta PSNR for each dataset considered. These results have been shown in table 4.4. We observe a significant difference in the results while using the D1 and the D2 metric for geometry distortion especially in the Soldier and Longdress datasets. The average improvement to geometry distortion using the D2 metric at similar bitrates is 3.74% (as compared to the anchor). The differences is distortion benefits across the datasets is caused by the varying flatness of local patches. This could be due to differences in the sensors and capture setup used to create these datasets. The queen dataset for example is a synthetic object and is less likely to have flat local surfaces. Other datasets such as the 8i Redandblack are naturalistic and were captured by using an array of cameras and calculating the point locations using photogrammetry. This process could have introduced flat local patches that our codec was able to use to compress the cloud.

We also observe significant improvements in objective quality of the U and V color channels, this can partly be attributed to the quantization scheme used for JPEG compression in the enhancement layer. In our implementation we are able to retain more information in these channels while maintaining the same bitrates as the anchor.

The decoded point clouds for the same frame (frame15) from the longdress dataset have been shown for rate point 3 at 13 Mbits/sec and rate point 5 at 42 Mbit/sec in figures 4.33 and 4.34 respectively. We can observe the loss of finer details in facial features and clothing at the lower bit rate.

Figure 4.33: Decoded point cloud at rate point 3

Figure 4.34: Decoded point cloud at rate point 5

Chapter 5

Conclusion

In this chapter we summarize the work we have done, the contributions, results and provide suggestions for future work.

In this thesis we worked on extending the state of the art MPEG anchor codec. This codec is used as a reference for the MPEG call for proposals on point cloud compression. The first contribution of this project is to implement the enhancement layer proof of concept provided by Ainala et al [15] within the anchor framework. This is done to evaluate the performance and suitability of this approach to the dense photorealistic point clouds contributed by the members of the MPEG ad hoc group on point cloud compression. The second contribution is to improve on this approach by adding inter frame prediction to the enhancement layer so that additional compression can be achieved by exploiting temporal redundancies. Our approach combines octree based point cloud coding with an enhancement layer similar to scalable video coding and lossy color coding based on JPEG image compression. Our approach offers an easy implementation and is parallelized.

The objective quality of point clouds compressed using our codec was rigorously tested using the evaluation criteria applied to proponents of the MPEG call for proposals. We use PSNR based quality metrics that were developed by adapting concepts from mesh and video coding to point clouds.

We validated the implementation of the approach proposed by Ainala et al and verified that the enhancement layer subdivision process provides the same result. We also checked the size of the encoded stream and observed only small differences from their results on the same dataset. This can be attributed to differences in the framework and in the configuration settings used for the octree compression layer.

The results from comparing the all intra enhancement layer approach with our enhancement layer inter prediction approach are promising. There are significant improvements as demonstrated by the BD-PSNR. We observe an improvement of around 9.5% to objective quality of geometry and 9.9% for the Y color channel at comparable bit rates. This comparison was done by plotting a rate distortion curve covering the top 4 bit rates specified by MPEG using the longdress dataset.

Finally the comparison of our approach to the MPEG anchor demonstrates an improvement over the current state of the art dynamic point cloud codecs by adding an enhancement layer with inter prediction. We observe an overall improvement of 2.7 - 6.7% in geometry

distortions at similar bitrates and an improvement of 6.1% , 14.3% and 8.5% in the Y, U and V color channels respectively. There is a significant improvement to distortion in the color channels, this is because of the quantization scheme used in JPEG compression. We are able to retain more color information while operating at the same bit rates as compared to the anchor. Our results also indicate that many local patches in the point cloud exhibit flat characteristics, this can be a consequence of the nature of the capturing process used.

Figure 5.1: Compression artifact in the decoded stream introduced by inter prediction

Our implementation can be further improved by using the GPU to calculate the covariance matrix and performing the eigen decomposition in the enhancement layer using platforms like Nvidia's CUDA. The statistical compression used can also be improved by training and using a learning based self adaptive entropy coding scheme such as the PAQ coder. PAQ is a shallow neural network that uses context modeling coupled with arithmetic coding.

Future work on compression of static point clouds can investigate improving the efficiency of graph transform based approaches where attributes are modeled as signals on the graph. Research into signal processing techniques for unorganized data can greatly benefit point cloud compression. Efficient calculations and lossy signal representations are needed to use this approach.

Future work on the compression of dynamic point clouds can greatly benefit from intermediate representations between 3D meshes and point clouds. Further research is needed to efficiently create such representations. An important criteria for such representations is the need for them to be resilient to noise as identified by Pavez et al. [30]. This will help in identifying correspondences across frames are further exploit temporal redundancies to compress the point cloud stream.

There is also a need to improve the objective quality metrics used to evaluate point clouds. The metrics currently in use do not correlate well to subjective quality and the

process of calculating normals for the D2 metric is too computationally expensive to be used for rate distortion optimization. Further research into point cloud specific objective metrics is needed. Some compression artifacts such as the artifact shown in figure 5.1 are difficult to detect with the current objective metrics. If new efficient and perceptually relevant metrics are developed we can detect such artifacts in real time and switch to intra coding to avoid them. There is also a need for better post processing filtering for point clouds.

The effectiveness of using point clouds for AR/VR applications can be drastically improved by standardization activities. There is a lot of variation introduced by the wide variety of sensors used in the capture process and the rendering for different display scenarios. As a consequence it is a lot harder to create perceptually relevant distortion metrics, in video coding for example the input and output representations are clear and as a result metrics are relatively well defined and effective. The most effective representation for VR and AR in the future can be a blend of different representations for different types of content rendered together in a single immersive environment. Point clouds codecs developed in future research should be designed and assessed in this context.

Bibliography

[1] 3d free viewpoint sports replay broadcasting. http://replay-technologies.com/. Accessed: 2017-02-12.

[2] 8i holo. https://8i.com/holo.

[3] Accuvein vein illumination. https://www.accuvein.com/about-us/company-info/.

[4] Daqri professional grade ar. https://daqri.com/professional-grade-ar/.

[5] Google expeditions. https://edu.google.com/expeditions/#about.

[6] Intel freed technology. https://www.intel.com/content/www/us/en/sports/technology/intel-freed-360-replay-technology.html.

[7] Lytro light field imaging platform. https://www.lytro.com/. Accessed: 2017-02-12.

[8] Marriot hotels virtual reality travel experience. http://news.marriott.com/2015/09/marriott-hotels-introduces-the-first-ever-in-room-virtual-reality-travel-exper

[9] Octree data structure. https://commons.wikimedia.org/wiki/File:Octree2.svg. Accessed: 2016-10-28.

[10] Raytrix 3d light field cameras. https://www.raytrix.de/. Accessed: 2017-02-12.

[11] Snapchat lenses. https://www.snap.com/en-US/news/page/3/.

[12] Visualization of large point clouds. http://umbra3d.com/visualization-of-large-point-clouds/. Accessed: 2016-10-28.

[13] Use cases for point cloud compression, iso/iec jtc1/sc29 wg11 doc. n16331, geneva, ch. June 2016.

[14] Call for proposals for point cloud compression iso/iec jtc1/sc29 wg11 n16732, geneva ch. January 2017.

[15] Khartik Ainala, Rufael N. Mekuria, Birendra Khathariya, Zhu Li, Ye-Kui Wang, and Rajan Joshi. An improved enhancement layer for octree based point cloud compression with plane projection approximation. *Proc. SPIE 9971, Applications of Digital Image Processing XXXIX*, September 2016.

[16] D. S. Alexiadis, D. Zarpalas, and P. Daras. Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15(2):339–358, Feb 2013.

[17] Aamir Anis, Philip A. Chou, and Antonio Ortega. Compression of dynamic 3d point clouds using subdivisional meshes and graph wavelet transforms. *Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, March 2016.

[18] Robert A. Cohen, Dong Tian, and Anthony Vetro. Point cloud attribute compression using 3-d intra prediction and shape-adaptive transforms. In *DCC*, 2016.

[19] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A. Chou.

[20] Touradj Ebrahimi, Siegfried Foessel, Fernando Pereira, and Peter Schelkens. Jpeg pleno: Toward an efficient representation of visual reality. *IEEE MultiMedia*, October 2016.

[21] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Horaud. *Time-of-Flight Cameras - Principles, Methods and Applications*. Springer, November 2012.

[22] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, April 2012.

[23] J Kammerl, N Blodow, R B Rusu, S Gedikli, E Steinbach, and M Beetz. Real-time compression of point cloud streams. *Robotics and Automation (ICRA), 2012 IEEE International Conference*, pages 778–785, May 2012.

[24] Wee Sim Khor, Benjamin Baker, Kavit Amin, Adrian Chan, Ketan Pate, and Jason Wong. Augmented and virtual reality in surgerythe digital surgical environment: Applications, limitations and legal pitfalls. *Annals of Translational Medicine*, 4, 2016.

[25] Henrique S. Malvar. Adaptive run-length / golomb-rice encoding of quantized generalized gaussian sources with unknown statistics. *Proceedings of the Data Compression Conference*, (6), March 2006.

[26] Rufael Mekuria, Kees Blom, and Pablo Cesar. Design, implementation and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, January 2016.

[27] Rufael Mekuria, Pablo Cesar, Ioannis Doumanis, and Antonella Frisiello. Objective and subjective quality assessment of geometry compression of reconstructed 3d humans in a 3d virtual room. *Proc. SPIE 9599, Applications of Digital Image Processing XXXVIII, 95991M*, September 2015.

[28] Sunil K Narang and Antonio Ortega. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE Transactions on Signal Processing*, 61(19), October 2013.

[29] A. Papadakis and K. Zachos. Subjective and objective video codec evaluation. In *2011 15th Panhellenic Conference on Informatics*, pages 194–198, Sept 2011.

[30] Eduardo Pavez, Philip A Chou, Ricardo L de Queiroz, and Antonio Ortega. Dynamic polygon cloud compression. *Microsoft Research Technical Report*, October 2016.

[31] Ricardo De Queiroz and Philip A. Chou. Compression of 3d point clouds using a region-adaptive hierarchical transform. *IEEE Transactions on Image Processing 25*, June 2016.

[32] Radu Bogdan Rusu. 3d is here: Point cloud library. *Robotics and Automation (ICRA), 2011 IEEE International Conference*, 2011.

[33] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. *2008 IEEE/RSJ on Intelligent Robots and Systems*, pages 3384–3391, September 2008.

[34] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Persistent point feature histograms for 3d point clouds. *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, pages 119–128, July 2008.

[35] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56:927–941, November 2008.

[36] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. *Proceedings of Symposium on Point-Based Graphics 2006, Eurographics*, July 2006.

[37] Ivan E Sutherland. The ultimate display. *Multimedia: From Wagner to virtual reality*, 1965.

[38] Dorina Thanou, Philip A. Chou, and Pascal Frossard. Graph-based motion estimation and compensation for dynamic 3d point cloud compression. *Image Processing (ICIP), 2015 IEEE International Conference on*, September 2015.

[39] Huang Y, Peng J, Kuo C, and Gopi M. A generic scheme for progressive point cloud coding. *IEEE Transactions on Visualization and Computer Graphics*, 14:440–443, 2008.

[40] Matt Yu, Haricharan Lakshman, and Bernd Girod. A framework to evaluate omnidirectional video coding schemes. *Mixed and Augmented Reality (ISMAR), 2015 IEEE International Symposium*, September 2015.

[41] Cha Zhang, Dinei Florencio, and Charles Loop. Point cloud attribute compression with graph transform. *Image Processing (ICIP), 2014 IEEE International Conference on*, October 2014.