VQBA: Visual-Quality-Driven Bit Allocation for Low-Latency Point Cloud Streaming

Shuoqian Wang
SUNY Binghamton
Binghamton, NY, USA
swang130@binghamton.edu

Mufeng Zhu
Rutgers University
Piscataway, NJ, USA
mz526@rutgers.edu

Na Li
Rutgers University
Piscataway, NJ, USA
na.li@rutgers.edu

Mengbai Xiao
Shandong University
Qingdao, Shandong, China
xiaomb@sdu.edu.cn

Yao Liu
Rutgers University
Piscataway, NJ, USA
yao.liu@rutgers.edu

ABSTRACT

Video-based Point Cloud Compression (V-PCC) is an emerging standard for encoding dynamic point cloud data. With V-PCC, point cloud data is segmented, projected, and packed on to 2D video frames, which can be compressed using existing video coding standards such as H.264, H.265 and AV1. This makes it possible to support point cloud streaming via reliable video transmission systems. On the other hand, despite recent advances, many issues still remain and prevent V-PCC from being used in low-latency point cloud streaming. For instance, point cloud registration and patch generation can take a long time.

In this paper, we focus on one unique problem in V-PCC: bit allocation among different sub-streams – the geometry sub-stream and the attribute (color) sub-stream – with the goal of improving the visual quality of point clouds under the target bitrate. Existing approaches either do not fully utilize the available bandwidth or can take a long time to run, which cannot be used in scenarios that require low-latency. To this end, we propose a lightweight, frequency-domain-based profiling method for transforming the dynamic point cloud data into a one-dimension vector. By using two single-layer linear regression models, we can estimate the compressed bitrate for geometry data and color information. This allows us to perform bit allocation between the geometry map and the attribute map with simple calculations. Evaluation results show that compared to the baseline approach, our method can achieve better visual qualities with smaller encoded segment sizes under the target bitrate.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → **Point-based models**.

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3612486

KEYWORDS

Point cloud, video-based point cloud compression, bit allocation, visual quality

ACM Reference Format:
Shuoqian Wang, Mufeng Zhu, Na Li, Mengbai Xiao, and Yao Liu. 2023. VQBA: Visual-Quality-Driven Bit Allocation for Low-Latency Point Cloud Streaming. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3612486

1 INTRODUCTION

As devices such as LiDARs and depth cameras are becoming smaller and more affordable, media formats that can support 6 degrees of freedom (6-DoF) are gaining increased popularity among users in emerging applications. Today, a number of new technologies have been developed to create a world of digital twins. For example, in real-estate, companies such as MatterPort [5] provide services that capture representations of houses and apartments that can be explored in 6-DoF for virtual open house. Groups such as CyArk [2] aim to create a cyber-archive for cultural heritages. One data type commonly used for representing 6-DoF media is point cloud.

To efficiently store and transmit the point cloud data, the Moving Picture Experts Group (MPEG) developed the point cloud compression (PCC) standards [6]. Today, the standard includes two solutions: video-based PCC (V-PCC) is mainly used for uniformly distributed points data, while geometry-based PCC (G-PCC) focuses on sparse point cloud which is usually used for big scenes such as cultural heritages. With recent developments of V-PCC, it becomes possible to encode and stream dynamic point cloud data. To do so, the original point cloud data is segmented and projected to 2D frames. This allows us to use any existing video codecs for encoding, and the encoded data can be transmitted through existing content delivery networks (CDNs) infrastructure.

Today, point cloud processing and encoding incurs very high computation overheads, which prevent wide adoption of this representation. Many factors contribute to the high computation overhead, including point cloud registration for aligning point clouds captured in different coordinate systems [19], segmentation for generating 3D patches which are projected to 2D patches [9], and packing 2D patches onto video frames for encoding [16]. In addition, within the V-PCC pipeline, the *bit allocation* step also requires

a large amount of computation and can take a long time. The most significant difference between video-based point cloud streams and traditional video streams is that a point cloud stream contains multiple sub-streams, including the geometry map, the attribute map (e.g., the color information), the occupancy map, and patch information [9]. Among these sub-streams, the occupancy map and the patch information are typically encoded in a lossless manner, while the geometry map and the attribute map are encoded with lossy encoding. Bit allocation decides how to distribute the available bitrates between these two sub-streams.

Bit allocation usually start with compressed rate estimation, which uses pre-processing for profiling the frame content. This step is required because compression efficiency mainly depends on the content itself. Rate estimation profiles the video content to check if it is encoding-friendly and decides how much details should be discarded (and thus lossy encoding) to fit under the available bitrates. With this result, bit allocation can then decide how to balance the available bitrates between geometry data (stored in the geometry map) and color information (stored in the attribute map).

In "common test condition" (CTC) [22], MPEG uses five fixed combinations of quantization parameter (qp) settings for geometry map and attribute map encoding. For all these five combinations, the qp for geometry map is lower than the qp for the attribute map. This means more geometry details are retained compared to color details. However, using fixed assignments would under-use or over-use the target bitrate, and thus may result in inferior visual quality under the given target bitrate. Existing works have proposed both profiling-based [15, 17] and non-profiling-based approaches [27]. However, these approaches either require a long pre-encoding time, which may not be appropriate for low-latency streaming or use heavyweight, computationally-intensive algorithms.

In this paper, we propose VQBA – a viewport-quality-driven bit allocation strategy using lightweight profiling and rate estimation. Unlike pre-encoding based approaches, we use a frequency-domain-based (DCT-based) video content profiling method, which can describe the potential compression efficiency of the video content by a one-dimensional vector. We call this vector the "segment profiling vector". With this vector, we use a single-layer linear regression model to predict the compressed/encoded bitrate of the sub-stream, which can be computed using only one vector dot product. Benefited by the non-encoding design and the block-operation-based implementation, we can calculate a good bit allocation solution for multiple sub-streams of a point cloud with less than one second of time. This is significantly shorter compared to existing pre-encoding-based approaches that take tens of minutes. Overall, this paper makes the following contributions:

- We propose a frequency-domain-based method for lightweight profiling of the dynamic point cloud content with no need for time-consuming pre-encoding.
- We design a greedy bit allocation strategy based on the profiling and rate estimation results.
- Evaluation results show that our proposed bit allocation scheme can calculate the encoding parameters for multiple sub-streams of a point cloud in less than one second of time, and the resulting visual quality is on par with heavy-weight pre-encoding based approaches.

2 BACKGROUND AND RELATED WORKS

2.1 Point Cloud

Point cloud is a data type that can be explored with 6-DoF: from any position in 3D space and in any orientation $(yaw, pitch, roll)$. For each data point in the point cloud, it typically contains six parameters, the Cartesian coordinates of (x, y, z) and color information, e.g., (R, G, B). Point cloud data capturing and recording usually requires setting up multiple devices "looking at" the scene in an "outside-in" manner. To obtain data about the coordinates of points, 3D devices such as RGB-Depth cameras and/or LiDARs are needed to record the geometry information of the scene. If multiple point cloud frames are captured consecutively, we refer to the recorded stream a *dynamic point cloud stream*.

2.1.1 Point Cloud Communications. With the recent boom of virtual reality (VR) and augmented reality (AR) applications, point cloud communications emerge as a mechanism for transmitting representations of real-world that supports 6-DoF navigation. For example, Gunkel et al. presented a VR communication system that transmits the spatial (i.e., depth) information in addition to the color information [11]. However, a limitation of this system is that it cannot support full 6-DoF. To achieve more immersive experience, it is important to address the high computation and latency of point cloud capture and compression. For capture, recent works have proposed fast registration methods [12, 21]. For compression, both video-based approach (V-PCC) and geometry-based approach (G-PCC) are still slow today. Overall, point cloud communication is still at an early-stage, with many challenges remaining to be solved.

2.2 Video-Based Point Cloud Compression

V-PCC is developed by MPEG for dynamic point cloud content compression [9, 23]. We depict the overall workflow of V-PCC in Figure 1. V-PCC aims to use existing video coding standards, e.g., HEVC, for encoding the dynamic content.

2.2.1 Patch Generation. The first step of V-PCC is segmentation [14, 18, 25, 26]. It segments the point cloud frame twice [8, 9, 23]. The first time is coarse segmentation: all the data points are distributed to six faces of a cube based on their normal vectors. Then, refine segmentation clusters neighboring points who have the closest normal orientations. Adjacent points with similar normal vectors will be grouped together, forming a "flat" surface. Only one normal is calculated to indicate the normal of the surface, while the remaining details are dropped. After refine segmentation, the complicated geometry will be divided into a number of mini-planes. Finally, the patches will be rasterized into blocks with the size of 16×16 and packed into a 2D frame [8, 9, 23]. For points that are projected to the same patch, information about their depth from the patch is stored in the geometry map and later encoded as the geometry sub-stream in V-PCC (Figure 1).

2.2.2 Packing. After segmentation, the generated patches need to be placed on a 2D frame. Since the captured object is not static, the geometry structure varies in different frames. Any changes on the geometry structure will affect the segmentation result, which can further affect the packing process. In addition, the changing structure can significantly increase the difficulty of inter-frame

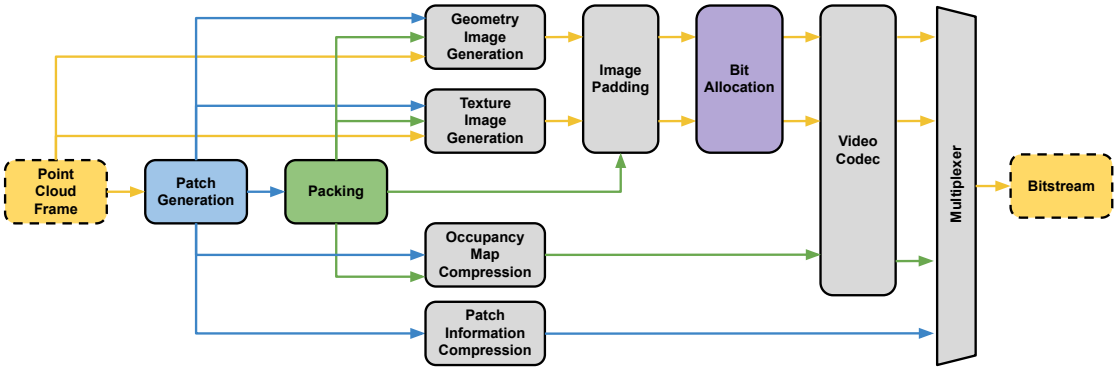


Figure 1: This figure shows the V-PCC encoding pipeline. In this paper, we focus on the "bit allocation" component in the V-PCC pipeline. Our proposed VQBA approach can improve the visual quality of reconstructed point clouds with reduced latency compared to existing approaches.

Table 1: qp combinations in $r1$ to $r5$ as introduced in CTC [22]

Setup	r1	r2	r3	r4	r5
Q_g: qp for geometry map	32	28	24	20	16
Q_c: qp for attribute (color) map	42	37	32	27	22

prediction, which should be avoided. Global packing is proposed to place similar patches over time at similar positions on the 2D frames, which significantly reduces the vibration caused by content motion between consecutive frames [8, 9, 23]. However, the trade-off here is that the frame resolution will be larger. Global tetris packing (GTP) [3] improves global packing by allowing non-uniform patches to be rotated to fill the 2D plane with fewer patch movement.

2.2.3 Bit Allocation/Rate Control. A point cloud stream is encoded as multiple sub-streams, including the geometry map representing the structure information of the points, the attribute map containing the color information, the occupancy map indicating whether a corresponding pixel is a valid one, and patch information. The occupancy map and the patch information are typically encoded using lossless encoding methods, while the geometry and attribute sub-streams are encoded in a lossy manner to achieve higher compression ratio.

In common test condition (CTC) provided by MPEG [22], five qp combinations of quantization parameters are provided for encoding the geometry and attribute sub-streams, regardless of what the input point cloud content is. In Table 1, Q_g is the qp value used for geometry map encoding, and Q_c is for attribute map encoding. There are two main issues with this approach. First, adjusting Q_g and Q_c at the same time may not be a good idea compared to changing Q_g and Q_c separately. Such bundled changes may cause the changing steps of the encoded bitrate being too big, exceeding or wasting the available bandwidth. In addition, for different point cloud contents, the amount of motion and texture complexity can vary substantially. Limiting the qp selections to a limited set of fixed setups may not give the best results for all contents.

Existing works in bit allocation for point cloud compression mainly use two approaches. The first is based on segment profiling, which usually requires pre-encoding for measuring the complexity of the point cloud content. For example, Li et al. [15] proposed a method which needs to encode the first segment of the stream twice for pre-encoding using two combinations of Q_g (the qp for geometry map) and Q_c (the qp for attribute (color) map) values: $(28, 37)$ and $(20, 27)$. Liu et al.[17] proposed to find the optimal qp combination of the given point cloud segments through three pre-encodings. The three (Q_g, Q_c) combinations used in their approaches are $(30, 40)$, $(36, 30)$, and $(38, 28)$. Pre-encoding results are used to calculate the parameters of their optimization model, which is used for calculating the best qp combinations. However, the pre-encoding process can take a long time and should be avoided if possible. To reduce and limit the per-encoding time, Li et al. [15] only pre-encodes the first segment of the point cloud stream. The same idea can be incorporated into the approach by Liu et al.[17] as well. However, this approach assumes that the full content of the point cloud stream is similar to the first segment, which may not be true in practical settings if the content changes substantially.

Besides profiling-based approaches, Yuan et al. [27] proposed a non-profiling based solution that using the differential evolution (DE) algorithm to find a good qp combination. It is the first work that proposes to assign different qp values to different frames in a segment. Given that no pre-encoding is involved, it can save time. However, there exists other limitations. In their experiments, to get the qp values for 4 frames, the population size (NP) is set to 50, and the number of iteration is 75. This is a reasonable setting since NP is typically set to 10 times of the result dimension [4]. However, a point cloud segment usually contains many more frames than just 4. This means the NP needs to be very large in practical scenarios. If given a normal segment with 32 frames, the NP needs to be over 300, and the times of iteration goes much larger. The large-scale DE is no longer a lightweight solution.

3 DESIGN OF VQBA

3.1 Segment Profiling

Discrete cosine transform (DCT) is commonly used in existing video coding standards. Today, it is still being used as the basic algorithm in new generations of video coding such as the upcoming VVC/H.266 standard. By applying DCT to signals, different frequency components of the signal can be obtained. In this work, we propose to use DCT for lightweight profiling. Unlike image and video encoding that apply DCT to 8x8 blocks, we perform DCT on the whole video frame to profile the full frame content. Next, we explain how we use DCT for frame profiling and segment profiling.

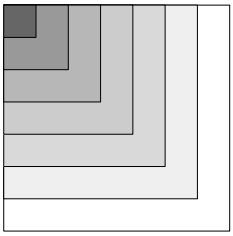


Figure 2: We perform frame profiling by applying band-pass filter to the DCT result with a given stride S, as in Equation1, and summing up DCT coefficient magnitudes in each band. In this way, we obtain a 1D vector indicating the amount of different level of frequencies.

3.1.1 Frame Profiling. Considering 2D DCT, low-frequency signals are located in the upper-left corner. Given $u = v = \xi$, then any other points $u' <= \xi$ and $v' <= \xi$ will not contain higher frequency components than the point at (u, v). So we can apply a low-pass filter by cropping a sub-matrix with the upper-left corner at $(0, 0)$ and the lower-right corner at (u, v) where $u = v$.

Considering the principle of the current lossy video coding standard, low-frequency information is retained, and high-frequency part is dropped/discarded. Usually, the magnitude of low-frequency component's DCT coefficient is higher than the high-frequency component. If we record the sum of DCT coefficient magnitudes in each frequency range as shown in Figure 2 using Equation 1 below, we can obtain a 1D vector, P, where elements with lower indices have greater values compared to elements with higher indices.

$$P(i) = \sum_{u=i\times S}^{(i+1)\times S-1} \sum_{v=0}^{(i+1)\times S-1} |C(u,v)| + \sum_{u=0}^{i\times S-1} \sum_{v=i\times S}^{(i+1)\times S-1} |C(u,v)| \quad (1)$$

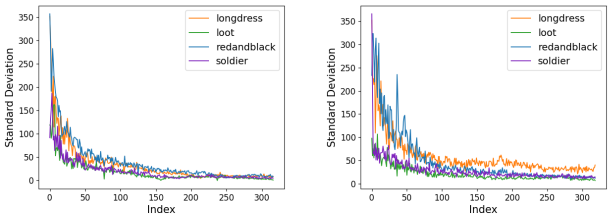
Given stride S and frequency range i, we obtain $P(i)$ by summing up the DCT coefficient magnitudes in the frequency range. The stride parameter S allows us to adapt the profiling vector's granularity. A large value of S decreases the output vector length.

3.1.2 Segment Profiling. For videos and dynamic point cloud streams, inter-frame compression is typically applied for exploiting similarities among a series of continuous frames. In video compression literature, a series of continuous frames is referred to as a "segment". It is know that content movements over time can significantly affect motion vectors, which are used for inter-frame compression. Thus, movements can affect the compression efficiency much more than the static content. This motivates us to profile the segment by calculating the standard deviation of each element in the profiling vectors of frames in the segment as in Equation 2.

$$V(i) = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(P_n(i) - \hat{P}(i))^2} \quad (2)$$

This results in a vector V with the same length as single frame profiling vectors. Each element in the resulting variance vector indicates the standard deviation of DCT coefficient magnitude in specific frequency ranges among frames in the segment. N indicates the number of continuous frames in consideration. For example, consider a V-PCC sub-stream with 32 frames that contains 1280 × 1280×32 pixels, after segment profiling, we can describe this content with just a 1D vector.

For segment profiling experiments, we use four segments from different point cloud streams in the `8iVFBv2` dataset [13]. Among the four streams, `longdress` has complicated textures and is with



(a) Profiling result for geometry segments (b) Profiling result for color segments

Figure 3: Results of segment profiling.

frequent motions, `loot` has simple texture and is with few motions, `redandblack` is much similar to `longdress`, and `soldier` has complicated textures but is with few movements. The profiling results are shown in Figure 3. In this figure, the x-axis represents values of the profiling vectors with the index ordered from low to high, and the y-axis represents the standard deviation (variance) calculated by Equation 2. We find that the curves of `longdress` and `redandblack` have similar distributions, and so do `loot` and `soldier`. `longdress` has the most high-frequency color information, making it the most difficult stream to be encoded (Figure 4 in Section 3.2 also confirms this). The 1D vector obtained from segment profiling allows us to perform rate estimation. We discuss rate estimation in Section 3.3.

3.2 Joint Geometry and Attribute Visual Quality

In the current V-PCC standard, the quality metrics for geometry information and color information are separated. PSNR-based geometry distance is used for measuring volumetric distortion [24], and traditional PSNR is used for color frames. However, the different calculation processes of the two metrics make it very hard to present a fair evaluation for the decoded point cloud.

It is also very hard to describe the quality of a point cloud using a simple PSNR metric. In Figure 4, we can see the PSNR values for geometry data and attribute data fall into different ranges. For geometry sub-streams, the PSNR values are all over 60, while the values for the color sub-streams are mainly below 40. Some research groups have proposed models that assign different weights for geometry and color and combine the two metrics, e.g., [15, 17]. However, as in Figure 4, the gradient of geometry quality and color quality are very close, which means the linear combination of the two metrics will still produce a linear gradient.

Instead of using the metrics described above, we use a viewport-quality-based metric that more closely relates to the users' true viewing experience for evaluation. Because true 3D displays are rare and expensive, today, people still have to use 2D displays for watching any 2D or 3D content. That means 3D content must be rendered on a 2D display to be viewed by users. We thus argue that directly using a metric that quantifies the quality of the rendered viewport would more closely relate to the user's true viewing experience. In our study, we used views rendered from the original, uncompressed point cloud as the ground-truth and compared them against the views rendered from compressed and decoded point cloud data to obtain the viewport-PSNR results. The resolution of rendered views is 960 × 960. The idea of studying viewport-based quality is also used in existing subjective visual quality studies. For example, Zerman et al. [28] studied the subjective visual quality of

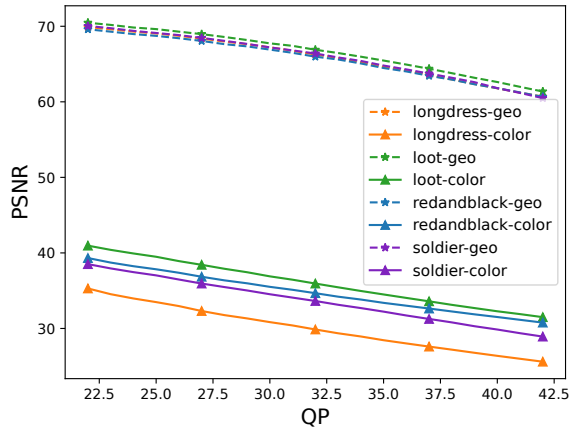


Figure 4: The figure shows the PSNR values with different quantization parameters (qp). "geo" represents the geometry data, and "color" represent the color/attribute data.

3D representations that are compressed using different methods. They used Blender for rendering the viewports and FFmpeg for compressing the rendered viewports to videos. The subjective study was conducted by having subjects watch FFmpeg-encoded videos of viewports.

To generate the viewports, we render the point cloud frame both before encoding and after decoding. For these point clouds, we use the virtual camera implemented by the Open3D library [29] to take pictures of them (i.e., render the viewport based on the viewing position and camera direction). We can then compare the visual quality of these pictures, i.e., pictures generated using the decoded point cloud vs. ground-truth pictures generated using the uncompressed point cloud. We selected 16 vantage points, 8 on the equator (e.g., the waist area of the person in the dataset), 4 with 60 degrees latitude (e.g., the shoulder area), and 4 with -60 degrees latitude (e.g., the leg area). We also adjusted the focal length of the virtual camera to reduce the blank area while leaving no content out of the view. If a given point cloud data has a large volume, we can increase the number of vantage points, so that they are more uniformly distributed. Comparing to existing quality metrics that are based on the geometry distance [24] and the decoded frames (not after reconstruction), our visual-based quality metric directly measures the quality of the rendered views.

We encoded four streams in the dataset with 30 different combinations of qps with $Q_g \in \{32, 28, 24, 20, 16\}$ and $Q_c \in \{42, 38, 34, 30, 26, 22\}$. We then rendered the decoded and reconstructed point cloud frames. For each reconstructed point cloud frame, we took 16 pictures of them from different vantage points as described above and calculated the mean value of the viewport-PSNRs. The results are shown in Figure 5. In this figure, dots of the same color represent results from the same point cloud streams in the dataset. We find that the PSNR gradient for geometry data is much greater than color information. At any point, given a qp combination (Q_g, Q_c), decreasing the qp value for geometry map Q_g gives a better visual quality than Q_c. This indicates we can decrease Q_g for the geometry data greedily and choose Q_c for attribute information with the remaining available bitrates/bandwidth.

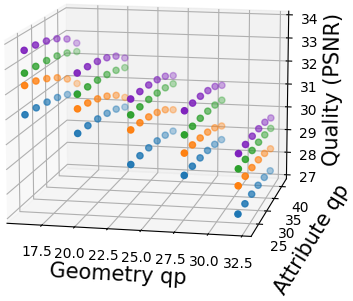


Figure 5: Viewport-PSNR values under different combinations of quantization parameters (qp).

3.3 Video-Level Bit Allocation

3.3.1 Rate Estimation. CTC by MPEG uses five fixed qp combinations for encoding geometry map and attribute map. These fixed setups are problematic and can be improved if the Q_g (qp for the geometry map) and Q_c (qp for the attribute map) can be flexibly selected. However, given a point cloud segment and a target bitrate, it is very challenging to decide the qp values for encoding. If the qp is chosen too small, and the encoded segments have much larger bitrate than the available network bandwidth, the segment needs to be re-encoded, which can incur additional delay, or the segment may be dropped. If the qp is chosen too large, causing too much high frequency details to be discarded, then the visual quality will suffer. It is thus very important to choose appropriate qp values for point cloud streaming, especially under low-latency scenarios.

Within the video encoding pipeline, motion estimation and quantization substantially affect the encoded bitrate and quality. Quantization filters the high frequency signals and noises. Motion estimation is needed for inter-frame prediction and is the most time-consuming component. If we can profile a segment and estimate the motion across frames, then it is possible to quickly estimate the encoding complexity and encoded bitrate.

By using the segment profiling method described in Section 3.1, we obtain a 1D vector where each element indicates the variance of the specific frequency range among the frames from the segment. When quantization is applied to DCT coefficients as with typical compression, a higher qp discards more information, while achieving better compression ratio; with a lower qp , frames retain more details, but result in a larger encoded bitrate. Thus, different qps directly affect the variance vector. Given a point cloud segment, we can calculate a profiling vector under the specific qp .

To do so, we modified Equation 1 by adding the quantization step to simulate the effect of different qp values for encoding. The resulting equation is shown in Equation 3. We calculated the sum of the profiling vector using Equation 4 and plotted them in Figure 6. This figure shows a linear relationship between the profiling vector and the encoded bitrate of the segment. We can see that by simply summing up elements in the profiling vector, we can formulate a roughly flat rate-variance plane. This indicates that by converting the segments into profiling vectors, it is now possible to find a cross-content model for estimating the encoded bitrate with no pre-encoding needed.

$$P(i) = \sum_{u=i\times S}^{(i+1)\times S-1} \sum_{v=0}^{(i+1)\times S-1} \left|\frac{C(u,v)}{Q}\right| + \sum_{u=0}^{i\times S-1} \sum_{v=i\times S}^{(i+1)\times S-1} \left|\frac{C(u,v)}{Q}\right| \quad (3)$$

$$Variance = \sum V(i) \quad (4)$$

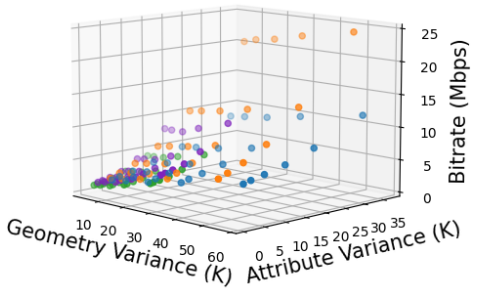


Figure 6: The figure shows the linear correlation between encoded rates and segment profiling result.

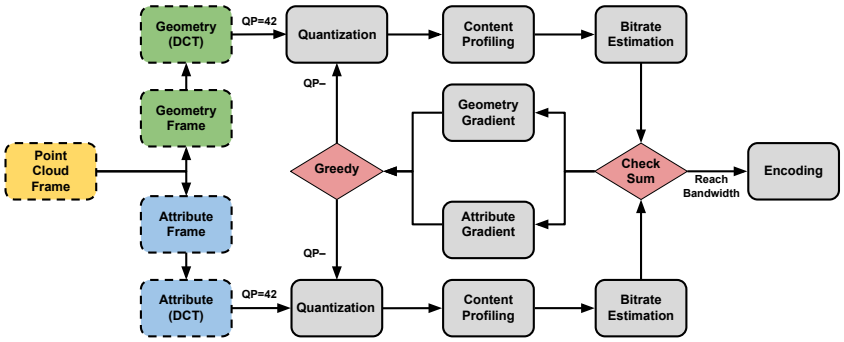


Figure 7: Overall VQBA bit allocation workflow.

We trained two simple linear-regression-based models for fitting the geometry-rate curve and attribute-rate curve, respectively, for encoded bitrate estimation. The linear model is described in Equation 5. We used two separate models because in the V-PCC pipeline, the geometry sub-stream and the color sub-stream are encoded separately and packed together with other sub-streams after encoding.

$$\text{Estimated Bitrate} = \sum w_i \cdot V(i) + b \tag{5}$$

3.3.2 Geometry-Greedy Strategy. Given that the point cloud is an emerging form of media and has not yet been widely used, there is no existing qp adaptation scheme that can be used as baseline. Instead, we can refer to the scheme used in WebRTC [7], which is a real-time communication system, since real-time scenarios has the similar constraints as low-latency streaming. In WebRTC, for any new stream, qp starts from the value of 106. If the encoded stream cannot consume all the available bandwidth (bitrate), the qp value can be reduced, which results in higher encoded bitrate. Otherwise, qp is increased until hitting the largest value possible. If the resulting encoded stream still consumes too much bandwidth (bitrate), then future frames will be scaled to a smaller frame resolution. For point cloud streaming, we can use a similar strategy. To reach a close-optimal result, the quantization steps for both the geometry map and the attribute map can start from the biggest value and reduce to a smaller value subject to bandwidth constraints.

Since our goal is to improve the visual quality under limited available bandwidth (encoding target bitrate), we aim to allocate more bits to the part that can result in more visual quality increases. Figure 5 shows that with the same change in qp (i.e., $\Delta Q_g = \Delta Q_c$), reducing Q_g can improve the visual quality in PSNR more than the other way. This motivates us to propose a geometry-greedy strategy: given a bandwidth/bitrate limit, we prioritize reducing the geometry qp , Q_g. We use the bitrate estimation model for quickly checking if the selected qp combination satisfies the given bandwidth constraint.

For example, in the beginning, both Q_g and Q_c are set to 42 (the highest qp value used in CTC), which corresponds to quantization step size of 80. By using the rate estimation models, we can check if the sum of the predicted geometry rate and the color rate is smaller than the available bandwidth. If it does, we can decrease the quantization step size. We decrease the geometry qp first until running out of the bandwidth or hitting the lowest qp cap. After that, we can assign the remaining bandwidth to color information until the color qp hits the cap. The workflow of our bit allocation scheme is presented in Figure 7. We discuss the implementation details in the next section.

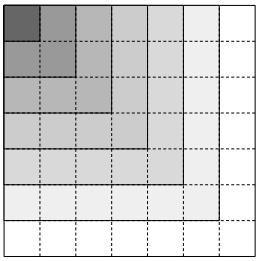


Figure 8: The figure shows how we perform block operation for frame profiling with a given 2D DCT frame.

4 IMPLEMENTATION

Block Operation. To profile a segment, we need to first apply quantization to the DCT matrix and then perform non-uniform additions. How additions are performed can significantly affect the performance. In our implementation, we used the block operation APIs from Eigen [10]. The DCT matrix is divided into blocks with size of $S \times S$ and divided by a quantization step corresponding to the qp . After quantization, the values within the same frequency range (band) will be added together to formulate a 1D vector from a 2D DCT frame. By default, the divisions and additions are operated pixel-by-pixel and line-by-line, which is very slow as each frequency (band) needs to scan the whole frame once. Instead, as shown in Figure 8, we divide the 2D frame into blocks with the same dimension. In this way, all the calculations are done in a block-by-block manner. Compared to the naive implementation, the processing time is reduced from 780ms to about 100ms by applying the block operation, for a single-time inference.

Segment Profiling Vector. Based on DCT calculation, we know that the low-frequency components are concentrated on the upper-left corner. With full-frame DCT implementation used by us, the resolution of the output must be the same size as the input. For values around the lower-right corner (i.e., high-frequency information), however, they are not as important. To decrease computing overhead, in our implementation, we only calculate variance vector for the upper-left corner with the resolution of 320×320. Even so, we observe that after quantization is applied to the DCT results,

most values of the 320×320 matrix are 0, and most non-zero values are around the upper-left corner.

In the V-PCC implementation (`mpeg-pcc-tmc2`) (shown in Figure 1), the pixel width of the packed frame is fixed to 1280, and the lowest frame resolution is 1280×1280. If it is not enough for large, complicated scenes, only the height will be extended to cover the whole point cloud. Considering the DCT function, calculating only a 320×320 matrix is safe and efficient.

Linear Regression Model. We use the linear regression model from the `scikit-learn` library [20] for estimating the encoded bitrate given the variance vector. We trained two models for bitrate estimation for geometry and color data, respectively. The trained model can work for different point cloud streams, which is more desirable compared to per-stream model training.

For training, due to the slow speed of `mpeg-pcc-tmc2`, it takes several days to get a few data points for a segment. So we have to validate our idea with limited number of data. To prevent over-fitting, we shrink the segment profiling vector length and the model size. To keep the frame information while shrinking the length of the output vector, we set the stride parameter $S = 64$. The resulting profiling vector length used in experiments is only 4, which is a more appropriate size for the 84 data points in total. Each linear model only contains 5 parameters.

Workflow. Figure 7 shows the overall workflow of our proposed scheme. Every time a new frame arrives, the cropped DCT result is calculated. Q_g and Q_c are initialized to be 42. For a single iteration, 32 DCT frames are divided by the corresponding quantization step block by block. Then the rest segment profiling processes will be applied. After we get the segment profiling vector, we use the rate estimation model for predicting the encoded bitrate under $Q_g = 42$ and $Q_c = 42$. If the sum of the predicted geometry bitrate and predicted color bitrate is greater than the given bandwidth, the iteration will be stopped, and the encoder will begin encoding with the selected `qp` combination. If there is still bandwidth available, the geometry-greedy strategy will reduce Q_g first. After that the next round of the iteration will start with updated `qp` combination.

Binary Search. In our initial implementation, we used a complete searching strategy, checking for `qp` values one by one to find the result. We can see from Figure 5 that both the geometry-rate curve and color-rate curve are monotone. This allows us to apply a binary search strategy to find the result in a faster way, which reduces the number of iterations from $O(n)$ to $O(\log n)$ where n indicates the scale of the `qp` candidate, e.g., if the highest `qp` is set to 42, and the lowest `qp` is 16, then $n = 27$.

5 EVALUATION

For experiments, we used a machine with an Intel i5-8600K CPU and a GPU of NVIDIA GEFORCE 1080. `HM-16.21` with `SCM-8.8` is used for point cloud compression [1]. The coding structure is set as random access (RA). The group of frame size is set to 32, which is same as the number of frames in a segment in CTC. The occupancy precision is set to 4. All experiment data is from the `8iVFBv2` dataset [13].

5.1 Bitrate Estimation

Since we increased the stride parameter to $S = 64$ to avoid over-fitting during the training processes due to limited data points, the length of the calculated segment profiling vector, from a 320×320 DCT frame, is only 4. We always drop the first component of the profiling vector since it contains the direct current (DC) component. Thus, the linear regression model for bitrate estimation only contains 5 parameters. If the size of data points grow as we obtain a larger dataset, we can increase the model size to fit the data scale.

Table 2: The average error rate for the predicted/estimated bitrate vs. the real bitrate obtained after encoding.

	longdress	loot	red-black	soldier	All (1:27)	All (1:41)
Geo.	5.6%	6.6%	6.9%	8.5%	7.9%	9.2%
Attr.	9.5%	22.0%	16.0%	30.3%	18.0%	17.8%

Table 3: Viewport quality and corresponding segment size by Liu et al. [17] vs. the proposed geometry-greedy strategy. By simply reducing the geometry `qp` and increasing the attribute `qp`, the visual quality can be improved while reducing the encoded segment size in all but one scenario. Note that Liu et al. [17] did not provide results for `redandblack` under $180kbpmp$, we thus are unable to provide the comparison results here.

Method	Viewport Quality / Segment Size (PSNR (dB) / KBytes)		
	longdress	loot	soldier
Liu [17]	28.87 / 380.7	30.91 / 290.3	31.42 / 314.0
GGS-1	29.06 / 374.5	31.19 / 290.3	31.15 / 317.3
GGS-2	29.32 / 356.4	31.24 / 268.2	31.71 / 303.4

For evaluation, we tested our method using cross-validation over all the four streams that we can use in the `8iVFBv2` dataset. For each set of experiments, bitrate estimation is done separately for the geometry sub-stream and the color information, considering the V-PCC workflow. The results are shown in Table 2. For each stream, each rate estimate model is trained with data from the remaining three streams and validated on the untrained stream itself. In addition, we also present results of cross-content performance in the last two columns. With $training : validation = 1 : 27$, the model is trained using three segments randomly selected from the four streams, which is about 3.6% of all the segments, and validated with the rest segments. With $training : validation = 1 : 41$, only two randomly selected segments are used for training. With more different streams and segments used for training, we believe the performance gaps (caused by over-fitting) between the different models will be much smaller. Table 2 shows that the estimated bitrate for geometry sub-streams is more accurate than attribute/color sub-streams. We conjecture that this is because we only applied our profiling method on the Y-channel, which contains the illumination information, from the 3-channel (YUV) data. So the color information contained in the other two channels is not profiled, causing the lower prediction accuracy.

5.2 Geometry-Greedy Strategy

In this part, we validate our proposed geometry-greedy strategy. We selected a group of `qp` combinations from the work done by Liu et al. [17] with target bitrate of $180kbpmp$. We applied our geometry-greedy strategy (GGS) to the given `qp` combination. In Table 3, GGS-1 means we reduce Q_g by 1 while increasing Q_c by

Table 4: Results under $2.5Mbps$ bandwidth/bitrate. Left: qp combinations calculated by Liu et al. and our method. Right: corresponding viewport quality and encoded segment sizes.

Method	Q_g, Q_c ($2.5Mbps$)			
	longdress	loot	redandblack	soldier
Liu [17]	30, 38	26, 28	N/A	28, 32
Ours	30, 42	22, 34	28, 40	24, 39

Method	Viewport Quality/Segment Size (PSNR (dB) / KBytes)			
	longdress	loot	redandblack	soldier
Liu [17]	30.23/380.7	31.73/290.3	N/A	32.65/314.0
Ours	30.02/304.6	33.89/265.5	30.49/333.2	34.08/287.0

Table 5: Results under $5Mbps$ bandwidth/bitrate. Left: qp combinations calculated by Liu et al. and our method. Right: corresponding viewport quality and encoded segment sizes.

Method	Q_g, Q_c ($5Mbps$)			
	longdress	loot	redandblack	soldier
Liu [17]	24, 32	22, 22	26, 26	22, 26
Ours	22, 36	16, 28	22, 32	20, 30

Method	Viewport Quality/Segment Size (PSNR (dB) / KBytes)			
	longdress	loot	redandblack	soldier
Liu [17]	32.52/717.0	34.02/606.9	31.54/776.9	33.80/644.6
Ours	33.12/579.7	32.44/527.5	30.96/579.4	33.35/537.8

1; GGS-2 means that we reduce Q_g by 2 while increasing Q_c by 2. The results presented are the visual-based point cloud compression quality metric we proposed in Section 3.2. We can see that by simply applying the geometry-greedy strategy to the baseline results, in most conditions, our GGS results outperform the baseline results with lower bitrates and higher visual quality.

5.3 Bit Allocation

For bit allocation, Liu et al. [17] propose to pre-encode the segments with three different setups in order to calculate the profiling parameters. They solve the optimal qp combination with the given target bitrate and the calculated parameters. In our method, we replace the pre-encoding process with DCT-based profiling, which is a lightweight solution targeting to get the close-optimal solution with fewer calculation.

We validated our method by comparing it with baseline approach by Liu et al. [17], while limiting the total bitrate to $2.5Mbps$ and $5Mbps$. Note that these chosen bitrates match the target bitrate of $180kbpmp$ and $365kbpmp$ in Liu et al. [17]. The results are shown in Table 4 and Table 5. In these tables, the viewport quality results are obtained by placing the camera on the equator and rotate 0.4 degrees per frame. We loop the 32 frames segment to rotate the full 360 degrees. Overall, for each experiment, 900 views are rendered and compared with their corresponding ground-truth views.

We can see that both methods reach similar Q_g and Q_c values under the same setups, while our geometry-greedy approach prefers devoting more available bitrates to the geometry sub-stream. In addition, Tables 4 and 5 also show results about the viewport quality and the encoded segment sizes in KBytes. In all cases, the encoded segment sizes of our approach is smaller than Liu et al. [17]. For all but one, the encoded segment sizes are within the bitrate target. In three cases, our approach results in better viewport quality even with smaller segment sizes. The improvement is most significant for `loot` and `soldier` videos under 2.5 Mbps bandwidth.

5.3.1 Latency. Compared to the approach by Liu et al. [17], we replaced the computing-intensive pre-encoding step with lightweight matrix manipulation. In addition, we further used block operations and the binary search to boost the performance. As a result, we reduce the bit allocation time from tens of minutes to sub-second level (shown in Table 6). We also notice that bit allocation for simpler streams such as `loot` and `soldier` takes longer time than the other more complicated streams. This is because simpler streams require lower qp to fully utilize the available bandwidth, which involves more iterations to find the best qp combinations, even with binary search applied.

Table 6: Average time used for obtaining the qp combinations.

Method	Average Processing Latency			
	longdress	loot	redandblack	soldier
Liu [17]	$5199.87s$	$6196.04s$	$6120.76s$	$9334.95s$
Ours	$861ms$	$912ms$	$871ms$	$997ms$

6 CONCLUSION

In video-based point cloud compression (V-PCC), information about the geometry and attribute (color) of the dynamic point cloud is encoded as separate sub-streams. It is important to determine appropriate encoding parameters for these sub-streams so that the best overall quality of the encoded stream can be obtained. Existing approaches use pre-configured parameters, which may not work well for all contents, or rely on a pre-encoding step, which can be time-consuming.

In this paper, we proposed VQBA – a visual-quality-driven bitrate allocation scheme for dynamic point cloud stream encoding. Unlike existing works that rely on computing-intensive stream pre-encoding, we design a lightweight, DCT-based, segment-profiling method for bitrate estimation. The profiling step generates a 1D vector that characterizes the potential compression efficiency of the point cloud segment, and we use it for encoded bitrate estimation via a simple linear regression method. With fast bitrate estimation, we are able to perform bit allocation in under one second of time, significantly faster than existing methods which can take tens of minutes. Results show that our method can achieve better visual qualities with smaller encoded segment sizes compared to the baseline approach.

ACKNOWLEDGMENTS

We appreciate constructive comments from anonymous referees. This work is partially supported by NSF under grants CNS-2200042 and CNS-2200048.

REFERENCES

[1] 2020. HM-16.21+SCM-8.8. https://vcgit.hhi.fraunhofer.de/jvet/HM/-/tree/HM-16.21+SCM-8.8.

[2] 2022. Cyark. https://cyark.org/.

[3] 2022. Video Point Cloud Compression - VPCC - mpeg-pcc-tmc2 test model candidate software. https://github.com/MPEGGroup/mpeg-pcc-tmc2.

[4] 2023. DE. https://en.wikipedia.org/wiki/Differential_evolution.

[5] 2023. Matterport 3D Real Estate. https://matterport.com/industries/real-estate.

[6] 2023. The MPEG-PCC project. https://mpeg-pcc.org/.

[7] 2023. WebRTC. https://webrtc.org/.

[8] Li Cui, Rufael Mekuria, Marius Preda, and Euee S Jang. 2019. Point-cloud compression: Moving picture experts group's new standard in 2020. *IEEE Consumer Electronics Magazine* 8, 4 (2019), 17–21.

[9] D Graziosi, O Nakagami, S Kuma, A Zaghetto, T Suzuki, and A Tabatabai. 2020. An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing* 9 (2020).

[10] Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

[11] Simon NB Gunkel, Rick Hindriks, Karim M El Assal, Hans M Stokking, Sylvie Dijkstra-Soudarissanane, Frank ter Haar, and Omar Niamut. 2021. VRComm: an end-to-end web system for real-time photorealistic social VR communication. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 65–79.

[12] Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno. 2021. Voxelized gicp for fast and accurate 3d point cloud registration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 11054–11059.

[13] Maja Krivokuca, Philip A Chou, and Patrick Savill. 2018. 8i voxelized surface light field (8iVSLF) dataset. *ISO/IEC JTC1/SC29/WG11 MPEG, input document m42914* (2018).

[14] Loic Landrieu and Martin Simonovsky. 2018. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4558–4567.

[15] Li Li, Zhu Li, Shan Liu, and Houqiang Li. 2020. Rate control for video-based point cloud compression. *IEEE Transactions on Image Processing* 29 (2020), 6237–6250.

[16] Jianqiang Liu, Jian Yao, Jingmin Tu, and Junhao Cheng. 2019. Data-adaptive packing method for compression of dynamic point cloud sequences. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 904–909.

[17] Qi Liu, Hui Yuan, Junhui Hou, Raouf Hamzaoui, and Honglei Su. 2020. Model-based joint bit allocation between geometry and color for video-based 3D point cloud compression. *IEEE Transactions on Multimedia* 23 (2020), 3278–3291.

[18] Anh Nguyen and Bac Le. 2013. 3D point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE, 225–230.

[19] Yue Pan, Bisheng Yang, Fuxun Liang, and Zhen Dong. 2018. Iterative global similarity points: A robust coarse-to-fine integration solution for pairwise 3d point cloud registration. In *2018 International Conference on 3D Vision (3DV)*. IEEE, 180–189.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[21] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. 2022. Geometric transformer for fast and robust point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11143–11152.

[22] Sebastian Schwarz, Gaëlle Martin-Cocher, David Flynn, and Madhukar Budagavi. 2018. Common test conditions for point cloud compression. *Document ISO/IEC JTC1/SC29/WG11 w17766, Ljubljana, Slovenia* (2018).

[23] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. 2018. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2018), 133–148.

[24] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. 2017. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3460–3464.

[25] Anh-Vu Vo, Linh Truong-Hong, Debra F Laefer, and Michela Bertolotto. 2015. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing* 104 (2015), 88–100.

[26] H Woo, E Kang, Semyung Wang, and Kwan H Lee. 2002. A new segmentation method for point cloud data. *International Journal of Machine Tools and Manufacture* 42, 2 (2002), 167–178.

[27] Hui Yuan, Raouf Hamzaoui, Ferrante Neri, Shengxiang Yang, and Tingting Wang. 2021. Global Rate-distortion Optimization of Video-based Point Cloud Compression with Differential Evolution. In *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 1–6.

[28] Emin Zerman, Cagri Ozcinar, Pan Gao, and Aljosa Smolic. 2020. Textured mesh vs coloured point cloud: A subjective study for volumetric video compression. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 1–6.

[29] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).