POINT CLOUDS QUALITY ASSESSMENT: EXPLORING THE POWER OF LEARNING-BASED TECHNIQUES WITH A PERCEPTUAL-DRIVEN HEURISTIC

MATEUS VIEIRA GONÇALVES
JOHANN HOMONNAI

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Point Clouds Quality Assessment: Exploring the power of learning-based techniques with a perceptual-driven heuristic

Mateus Vieira Gonçalves
Johann Homonnai

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM ENGENHARIA ELÉTRICA.

APROVADA POR:

Brasília/DF, 17 Fevereiro de 2023.

FICHA CATALOGRÁFICA

GONÇALVES, MATEUS VIEIRA; HOMONNAI, JOHANN
Point Clouds Quality Assessment: Exploring the power of learning-based techniques with a perceptual-driven heuristic. [Brasília/DF] 2023.
viii, 70p., 210 x 297 mm (ENE/FT/UnB, Bacharel, Trabalho de Conclusão de Curso, 2023).
Universidade de Brasília, Faculdade de Tecnologia, Departamento de Engenharia Elétrica.
Departamento de Engenharia Elétrica

1. Point Cloud Quality Assessment	2. Point Cloud
3. Virtual Reality	4. Mixed Reality
5. Perceptual Index	6. Learning-Based
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

GONÇALVES, M. V.; HOMONNAI, J (2023). Point Clouds Quality Assessment: Exploring the power of learning-based techniques with a perceptual-driven heuristic. Trabalho de Conclusão de Curso, Publicação PPGEE.XXXXX/2023, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 70p.

CESSÃO DE DIREITOS

Johann Homonnai; Mateus Vieira Gonçalves
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
Faculdade de Tecnologia - FT
Departamento de Engenharia Elétrica(ENE)
Brasília - DF CEP 70919-970

Stop counting only those things you have lost. What is gone, is gone. So ask yourself this: What is there... That still remains to you? — Eiichiro Oda

ACKNOWLEDGEMENTS

MATEUS VIEIRA GONÇALVES

Minha graduação se completa agora, depois de uma jornada de cinco anos, repleta de momentos bons, momentos ruins e momentos desafiadores. Entre a vida cotidiana e uma pandemia que nos obrigou a ficar isolados por praticamente dois anos, completo essa etapa muito feliz com tudo que passei, uma vez que nossa vida é composta do acumulado de todos os pequenos momentos: daquilo que fazemos ou deixamos de fazer, bem como de todas as responsabilidades e consequências que enfrentamos. Passei por muita coisa para me tornar o que eu sou hoje e, certamente, no futuro, olharei para trás e verei como minha jornada estava só começando.

Percebo, logicamente, que ter chegado até aqui só seria possível com a ajuda daqueles que me cercam e cabe a mim mostrar-lhes minha gratidão. Começo por Deus, Aquele que me deu a vida e tudo que eu tenho, agradeço a Ele por me abençoar e proteger, por me dar forças para sempre seguir e não me deixar perder nos caminhos do mundo, sem Ele nada seria possível e todos que citarei a seguir foram obras dEle, por sua infinita bondade, em minha vida.

Agradeço à minha família, a base de tudo que vivi e conquistei, sem eles eu não seria nada. À minha mãe, Maria Rosângela, por ser a mãe mais incrível que eu poderia ter, que doa, desde o princípio, a sua vida pela minha, que não hesita em servir o próximo, seja qual for a circunstância, pois o ser humano só passa a ter significado quando se coloca a serviço do próximo, sendo ela o meu maior exemplo de humildade e simplicidade, e que, com meu pai, me ajudou a ser quem estou me tornando. Ao meu pai, Paschoal, por sempre estar presente e fazendo o que estivesse ao seu alcance para me proporcionar a melhor vida possível, e que hoje, apesar dos desafios inerentes à condição humana, não hesita em continuar firme olhando por mim e me apoiando em todas as minhas decisões, mesmo que não sejam as que ele esperava. À minha irmã, Júlia, que esteve ao meu lado desde minha primeira lembrança, que me atura em todos os meus momentos, sejam os alegres, tristes, estressantes ou infantis, você é fundamental na minha vida e um dos motivos pelos quais eu consigo continuar seguindo. Ao restante da minha família: minha avó, tios e tias, primos e primas; obrigado por fazerem parte da minha vida em independentemente do momento, certamente sem vocês eu não estaria onde estou hoje. Amo vocês infinitamente.

Agradeço aos meus amigos, Felipe e Miguel, que carrego no peito desde a época da escola com o Messias, Tucano, Ferraz, Victor e Shiro. Muito obrigado por estarem comigo, por compartilhar tantos momentos e histórias, obrigado por sempre entenderem quando precisei negar estar presente em função de outras responsabilidades, vocês fazem parte da minha vida e eu não poderia estar mais feliz com isso.

Agradeço também aqueles que a universidade colocou em minha vida, em especial meus amigos Londe, Paulinho e Seu Pedro, que, desde o primeiro semestre, compartilham momentos difíceis e momentos alegres comigo, mas sempre juntos. Agradeço ao Ivan por ser uma pessoa fenomenal em tudo que faz e o motivo de eu estar dentro dessa área que eu gosto tanto. À minha amiga Isabel, que o IEEE-CIS nos reuniu, obrigado por sempre fazer o possível para me ajudar, por me escutar e obrigado pela sua companhia. Ao Guilherme, agradeço pelo companheirismo, pelas longas conversas e pela parceria sempre que precisei, certamente você, um grande ser humano, desempenhou um papel também fundamental na minha caminhada. Ao Pedro Ferreira, uma das pessoas mais emblemáticas que a UnB colocou na minha vida, obrigado pela solicitude, pelas extensas conversas e discussões, pelos conselhos, correções e críticas, obrigado por servir de inspiração, me puxando cada vez mais ao meu limite, sempre me tirando da minha zona de conforto, fazendo com que eu amadurecesse constantemente, sei que ainda tenho um longo caminho pela frente, mas obrigado, principalmente, pela sua amizade.

Obrigado à minha orientadora, Mylène, que esteve comigo desde o PIBIC, e que por um longo período me ajudou e apoiou em minha caminhada acadêmica, seja nas aulas ou nos projetos, obrigado pela paciência e pelos ensinamentos. Obrigado ao meu co-orientador, Pedro Garcia, por ser uma pessoa solícita e disponível, que nos ajudou nas incontáveis reuniões. E obrigado ao Johann, meu colega neste trabalho, que lutou nas trincheiras ao meu lado, obrigado pelo companheirismo em prol de um objetivo maior e obrigado pela paciência durante o processo, te desejo muito sucesso nas suas próximas etapas.

JOHANN HOMONNAI

Chegar ao fim da minha graduação é uma situação um tanto quanto complexa. Foi uma experiência difícil, desafiadora e, finalmente, recompensadora. A diferença entre a pessoa que era quando entrei e a pessoa que sou agora que estou sendo graduado não pode ser subestimada, com o crescimento e amadurecimento que me trouxe. Se eu hoje consegui atingir feitos de que me orgulho, isso se deve aos gigantes em que me apoiei para chegar até aqui. Por essas e outras, fica aqui meu agradecimento a todos que fizeram parte dessa jornada.

É necessário primeiro agradecer a Deus, por todas as graças, benção e provações que Ele me forneceu durante toda a minha vida. Sei que as pessoas que tanto me apoiam também fazem parte de Sua Graça. Obrigado pela força para seguir, pela enorme proteção a mim concedida. Obrigado por ter me dado esperança e motivos para perseverar em meio às dificuldades, por fortalecer meu espírito.

Gostaria de agradecer à minha família, por todo o apoio e incentivo que sempre recebi. Aos meus pais, por serem dois verdadeiros exemplos do que é esforço e dedicação e sua importância na vida. À minha mãe, Márcia, por ter me ensinado a importância do carinho e de não julgar. Por mostrar que ser forte não significa ser rígido, e sim ser fiel aos seus sentimentos. Obrigado por me ensinar a ajudar o próximo, a ter empatia. Ao meu pai, Johann Júnior, obrigado por me mostrar que o talento por si só não é suficiente para ser algo nem causar verdadeiro impacto no mundo, e que isso não me faz melhor que ninguém. Por me ensinar a tratar todos com respeito e a valorizar os estudos, a perseverança nos seus sonhos e a dedicação continuada para ser um ser humano melhor. Pelo apoio e crença nas minhas capacidades. À minha irmã, Catarina, por sempre mostrar estar presente, por nunca duvidar de mim, e também pelos meus sobrinhos, Bernardo e Benjamim, dois seres de luz e queridos.

Quero agradecer também à minha "segunda" família, família de minha namorada, que tanto me acolheu e me apoiou. Edmilson, Francisco Antônio, muito obrigado pelo apoio e preocupação demonstrados. Mirian, sua bondade é ímpar e seu amor transborda em tudo que faz. Nonato e Beatriz, vocês são como avós para mim, e valorizo demais as histórias, a convivência e o tanto com que sei que se importam comigo. Carmelina, obrigado por ser esse exemplo de mulher, e também por toda a demonstração de apoio, incentivo, carinho e proximidade. Vocês todos são muito queridos para mim. Obrigado por fazerem eu me sentir em casa.

À Anna Luísa, minha namorada, minha parceira. Acredito que meras palavras não seriam suficiente para que eu pudesse agradecer por tudo que fizeste por mim. Você sabe melhor do que ninguém o quanto foi difícil chegar até aqui. Honestamente, eu não teria chegado se não fosse por você. Obrigado por me fazer ser uma pessoa melhor a cada dia, por ter paciência e me ajudar a consertar meus defeitos, por ser essa verdadeira companheira de vida que és para mim. Obrigado por me mostrar o mundo de uma forma que eu jamais poderia ver sozinho. Muito, muito obrigado por tudo.

Aos meus amigos de escola, Vinícius, Pedro, Luigi, por seguirem próximos e terem me ajudado nos momentos sombrios e alegres que passamos. Fico feliz de seguirmos próximos e de estarmos chegando a outra fase de nossas vidas. Sem vocês, também não teria chegado aqui. Cada um de vocês pensa diferente de mim e um do outro, e creio que isso seja parte dos motivos pelos quais aprendi a respeitar a vivência de cada um. Ver cada um seguindo seus caminhos me ajuda a trilhar os meus próprios, e fico feliz com cada conquista de vocês. Ao Lucas Trindade, meu verdadeiro "companheiro de guerra"nessa batalha da UnB, você foi um dos principais motivos para eu ter ficado no curso e agora finalizá-lo. Obrigado por todos os estresses, trabalhos, listas de exercícios, provas, aulas, piadas, momentos engraçados e tristes, por trilhar esse caminho tão tortuoso da universidade, para o bem e para o mal. É bem estranho pensar que essa fase terminou, mas sou grato por passar ao seu lado.

Gostaria de agradecer por todas as experiências que tive e por todas as pessoas com que convivi na UnB. Tudo isso foi extremamente importante para que eu crescesse e me desenvolvesse como indivíduo e como profissional. Gostaria de agradecer aos professores que me mostraram novos caminhos, novas formas de pensar, e desafios a superar. Agradeço por mostrar áreas completamente desconhecidas por mim e por ver novas perspectivas que nunca havia pensado.

Por último, mas não menos importante, agradeço às pessoas que estiveram diretamente envolvidas na produção deste TCC. À minha orientadora, Mylène, obrigado pelos ensinamentos, pela paciência, e por ajudar a tornar esta caminhada mais tranquila. Ao Pedro Garcia, meu co-orientador, por seu interesse genuíno em nossa pesquisa e sua disposição a nos ajudar. Obrigado aos dois por todo conhecimento e oportunidades oferecidas. Ao Mateus, minha dupla nessa última batalha, o TCC, obrigado por toda a paciência e solicitude, pela ajuda com seus conhecimentos e experiências. Por ter se engajado e ter sido um companheiro para atingirmos nosso propósito. Este trabalho não teria saído de forma alguma sem você. Espero que você

tenha muito sucesso em sua vida profissional e em quaisquer empreendimentos realizados daqui para frente. Acredito que você vai longe.

ABSTRACT

Advances in the manipulation of tridimensional (3D) content have allowed new possibilities for what can be done with these technologies. Applications range from virtual reality to autonomous driving. A common data structure used for this kind of problem is Point Cloud (PC). This data structure is composed of points in the 3D Cartesian space that can hold several attributes, such as color. This makes it quite useful for studying land forms and other physical structures. The application of PCs has gained traction in recent years, leading to the necessity of transmitting, reading, rendering, and using data, and therefore compression.

To truly be impactful and effective, compression techniques must evaluate the quality of the resulting data. In consequence, the field of Point Cloud Quality Assessment (PCQA) has gained more attention recently. New techniques and heuristics have been explored to try and build an objective methodology for quality assessment that correlates well with the perceived quality by human beings. The usual approaches for Point Cloud Quality Assessment (PCQA) are related to color and geometry, and novel projection-based techniques have been developed to leverage existing and robust Image Quality Assessment (IQA) metrics.

In this work, we explore the power of Neural Networks and Machine Learning to create a perceptual-driven, full reference metric for PCQA. We first create 2D projections, which are regular images, from the 3D Point Clouds (PCs). Then, we pass these images through a recently discovered Image Quality Assessment (IQA) metric known as 'DISTS' and deposit the generated scores in a vector. Finally, this vector is used as input in a regressor model to predict the final quality score. We demonstrate that our model is competitive in relation to state-of-the-art metrics, and our results suggest that further improvements can be achieved in future works. Consequently, we demonstrated the power of projection techniques along with texture evaluation by generating a score that correlates well with the human ground truth.

Keywords: Point Cloud, Quality Assessment, Virtual Reality, Mixed Reality, Perceptual Index, Learning-Based

RESUMO

Os avanços na manipulação de conteúdo 3D permitiram novas possibilidades de utilização dessas tecnologias. As aplicações variam de realidade virtual a direção autônoma e escaneamento de comunidades como a Rocinha, essas duas últimas se beneficiando do uso de tecnologia LiDAR. A estrutura de dados padrão *de facto* para esse tipo de problema é a Point Cloud (PC - Nuvem de Pontos). Uma PC é uma estrutura de dados composta de pontos espalhados pelo espaço cartesiano tridimensional que podem apresentar diferentes atributos, como intensidade de luz e cor, além das suas coordenadas. Essa estrutura de dados permite mapear, entre outras coisas, distâncias, alturas, volumes, o que a torna bastante útil para estudo de relevos e outras estruturas físicas. A aplicação de PCs vem ganhando tração em anos recentes, e com isso vem a necessidade de transmitir, ler, renderizar, utilizar e, portanto, comprimir.

Para que técnicas de compressão sejam verdadeiramente impactantes e efetivas, é crucial avaliar a qualidade dos dados resultantes. Por consequência, o campo de Avaliação de Qualidade de Point Clouds (PCQA) tem recebido mais atenção recentemente, e novas técnicas e heurísticas têm sido exploradas para tentar construir uma metodologia objetiva para avaliação de qualidade que correlacione bem com a qualidade percebida por seres humanos. As abordagens usuais para PCQA são relacionadas a cor e geometria, e novas técnicas baseadas em projeções foram desenvolvidas para aproveitar métricas robustas de Avaliação de Qualidade de Imagens (IQA) já existentes, com o padrão de Compressão de Point Clouds baseada em Vídeo (V-PCC) estabelecido pelo MPEG se aproveitando dessa abordagem.

Neste trabalho exploramos o poder de Redes Neurais e Aprendizagem de Máquina para criar uma heurística movida por percepção, referência-completa, para PCQA. Primeiro geramos projeções 2D, as quais são imagens regulares, a partir de PCs. Então, passamos essas imagens pela métrica de IQA DISTS movida por textura desenvolvida por Meynet et al., combinamos as pontuações geradas em um vetor e finalmente utilizamos isso para alimentar um regressor que irá predizer a pontuação de qualidade final. Nós mostramos que nosso modelo é competitivo com métricas estado-da-arte, e sugerimos trabalhos futuros e melhorias que podem ser feitos para explorar adiante a heurística e aprimorar sua desempenho. Consequentemente, demonstramos o poder de técnicas de projeção concomitantes com avaliação de textura ao gerar uma pontuação que correlaciona bem com a dada por humanos.

Palavras-chave: Avaliação de Qualidade de Point Cloud, Point Cloud, Realidade Virtual, Realidade Mista, Índice Perceptivo, Baseado em Aprendizagem

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

2D	Bidimensional
3D	Tridimensional
3DoF	Three Degrees of Freedom
6DoF	Six Degrees of Freedom
ACR	Absolute Category Rating
AI	Artificial Inteligence
CNN	Convolutional Neural Network
DISTS	Deep Image Structure and Texture Similarity
DL	Deep Learning
DPC	Dynamic Point Cloud
DSIS	Double Stimulus Impairment Scale
FM	Feature Map
FNN	Feedforward Neural Network
FR	Full-Reference
G-PCC	Geometry-based Point Cloud Compression
HEVC	High Efficiency Video Coding
HMD	Head-Mounted Display
HVS	Human Visual System
IQA	Image Quality Assessment
ISO	International Organization for Standardization

ITU	International Telecommunication Union
JPEG	Joint Photographics Experts Group
LBP	Local Binary Pattern
LCP	Local Color Pattern
LF	Light Field
LoD	Level of Detail
LOOCV	Leave-One-Out Cross-Validation
M-PCCD	MPEG Point Cloud Compression Dataset
ML	Machine Learning
MLP	Multilayer Perceptron
MLPRegressor	Multi-Layer Perceptron Regressor
MOS	Mean Opinion Scores
MSE	Mean Squared Error
MPEG	Moving Picture Experts Group
MPEG-I	MPEG Immersive Media
NN	Neural Network
NNI	Neural Network Intelligence
NR	No-Reference
NuSVR	Nu Support Vector Regression
P2Pl	Point-to-Plane
P2Po	Point-to-Point
PC	Point Cloud
PCC	Pearson's Correlation Coefficient
PCQA	Point Cloud Quality Assessment

Pl2pl	Plane-to-Plane
PSNR	Peak Signal-to-Noise Error
QoE	Quality of Experience
ReLU	Rectified Linear Unit
RMSE	Root-Mean-Square Error
RNN	Recurrent Neural Network
RMS	Root Mean Squared
RR	Reduced-Reference
SGD	Stochastic Gradient Descent
sklearn	Scikit-Learn
SROCC	Spearman's Rank Correlation Coefficient
SSIM	Structural Similarity
SVR	Epsilon-Support Vector Regression
TMC2	Test Model Category 2
V-PCC	Video-based Point Cloud Compression
V-SENSE VVQDB	V-SENSE Volumetric Video Quality Database
VAE	Variational Autoencoder
VQA	Video Quality Assessment
WPC	Waterloo Point Cloud database

CHAPTER 1

INTRODUCTION

Advances in capture, storage, and display technologies for 3D content continue to push the limits of what can be achieved with this type of media. Point Clouds (PCs), which are crucial for most applications in this field, being the data structure chosen by several immersive media applications, are formed by a collection of points in the 3D space. Each point contains the coordinates (x, y, z) to represent the geometry of the content and the point's respective attributes, such as color or reflectance. In such a manner, PCs can have numerous points, compromising the viability of the entire application due to the increased storage size and the large bandwidth needed for transmission. Therefore, the necessity for compression as an enabling technology arises. Due to the degradation it causes, the respective quality assessment of the content is needed to provide a good metric able to evaluate different compression techniques.

Examples of applications that use PCs range from autonomous driving [1] to virtual reality [2]. To generate the necessary data, capture technologies such as RGB-D cameras [3] and light detection and positioning (LiDAR) sensing [4] come in handy. Miranda *et al.* [5] explores the power of the LiDAR technique is explored by Miranda *et al.* [5] by the collection of data in Rocinha, the largest favela in Rio de Janeiro, Brazil. The favela is mapped and morphologically analyzed using PCs. The resulting data are composed of more than one million points per second, demonstrating how costly some PCs can become to storage, transmission, and rendering, underscoring the necessity for compression. Figure 1.1 from the project site [1] shows an example of the result of Miranda *et al.* work.

Another application of PCs can be seen in virtual reality, in which users wear a Head-Mounted Display (HMD) to enter an immersive experience, as shown in Figure 1.2. The use of HMDs can increase the user's awareness of content degradation since the display is very close to the eye. Therefore, this application calls for even more powerful quality assessment techniques

[1]<https://senseable.mit.edu/favelas/>

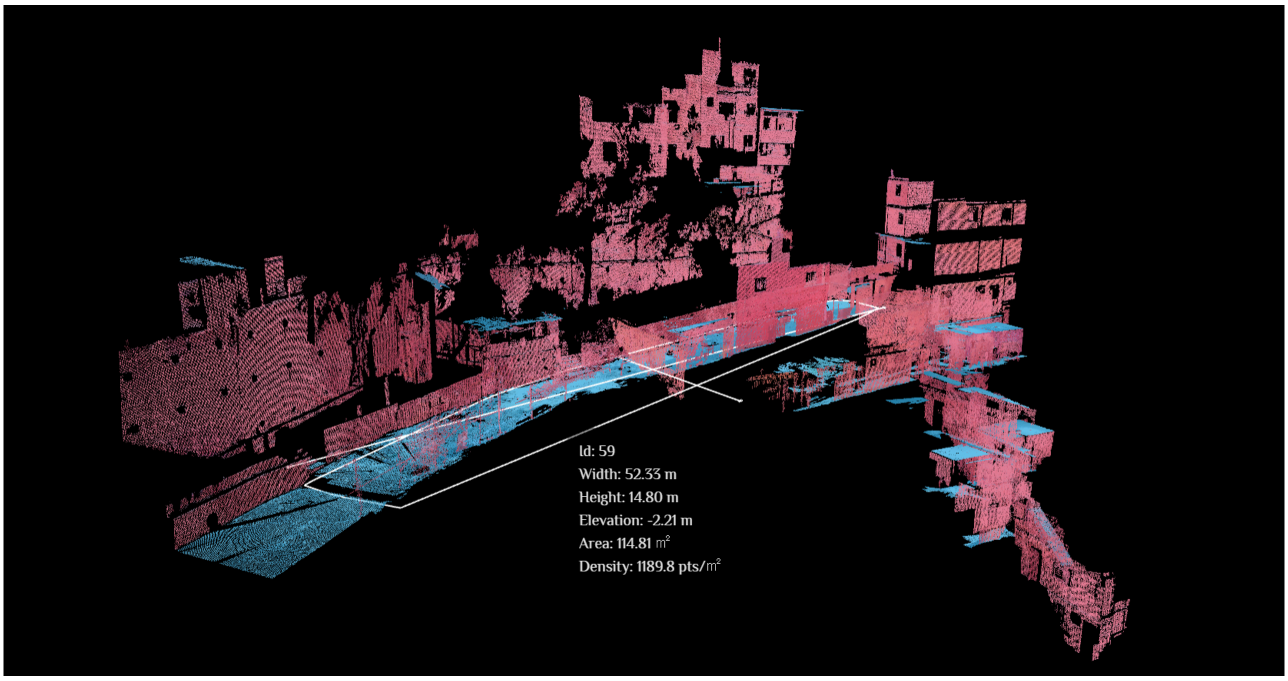

Figure 1.1. Point Cloud representation of a street in Rocinha, obtained with LiDAR technology. Courtesy of *Senseable City Lab* and available at https://senseable.mit.edu/favelas/.

that would be able to objectively assess the quality of the content. The provided score must have a strong correlation with the final user perception, serving as a guide to the development of codecs that provide improved content quality.

However, compression of PC can become a very complex task, not only due to the really large number of points, but also due to some intrinsic characteristics of the content, such as irregularity and sparsity. The representation of 3D content with PCs has two main aspects: geometry (how the points are arranged in space) and attributes (color, in most cases). In compression, only the geometry aspect can be addressed, only the attributes, or both, as in Meynet *et al.* [6]. Not only does compression suffer from the complex nature of PCs, but also PCQA has to address the intrinsic complexities of the data to provide a good quality score.

SUMMARY OF THE CONTRIBUTIONS

At the beginning of our research, we conducted an extensive review of the state of the art literature to identify and solve some pertinent challenges related to dynamic PCs. The main goal of this initial development was to contribute to the advancements of the techniques proposed by Diniz *et al.* [7, 8, 9, 10, 11, 12, 13] and to adapt these techniques for dynamic PCs purposes. A

Figure 1.2. Example of a Head Mounted Display. Source: https://www.arvigbusiness.com/for-home/are-virtual-reality-headsets-still-a-things

dynamic PC is a PC that has a time dimension, is composed of a combination of frames formed by static PCs and can be analogous to a bidimensional (2D) video. As part of this collaboration, we investigated the added value of incorporating temporal pooling in PCQA of dynamic PC using metrics designed for static PCs. This investigation was carried out considering both full reference and no reference PCQA approaches and we found that the performance of temporal pooling is consistently better when a temporal variation model (specifically detailed in [14]) is used. This finding is in part already published in top-tier conferences, the first page of the articles is depicted in the Appendix A.

However, the main contribution of this dissertation includes the discovery of a novel learning-based metric based on PC projections. Projection-based approaches have already been explored and some techniques leverage its benefits in diverse applications, such as PC compression (e.g., MPEG's Video-based Point Cloud Compression (V-PCC) standard codec for Dynamic Point Clouds (DPCs) [15], where projections are generated per frame and compressed using a 2D video compression technique). In the scope of PCQA, we adopted a projection-based approach in which we generate six projections for each PC. Each projection is a conventional digital image, removing the irregularity and sparsity of the content. This use of projections also makes it possible to take advantage of the power of IQA techniques, such as powerful Neural Networks (NNs) trained on large datasets that already have a good generalization capacity, such as the VGG16 [16]. From that, to assess the quality of the content, the task is to find a way to map from an IQA technique to a suitable score for PCQA. Furthermore, after generating the projections, cropping and padding were performed to mitigate distraction effects that background (nonoccupied area) may cause to the final score, as proposed by [17].

Despite its benefits, there is a trade-off in projection-based techniques: While they overcome irregularity and sparsity, they fail to capture local displacement errors in 3D space, which may result in a lower correlation with the user's perception. To balance this trade-off, it is important to generate good features to feed the PCQA model. Inspired by the work of [18], we chose to follow a perceptual approach to extract the features from the projections, instead of performing a pixel-by-pixel comparison, since it fails to capture the real perception that the user would have. In this sense, our approach aims to drive the nature of the features to better represent the user's final perception by combining texture and structure characteristics from the reference and degraded images obtained with a Deep Learning (DL)-based approach using the VGG16 Convolutional Neural Network (CNN) [16]. These characteristics are combined into a weighted sum, generating one output value for a single projection (view). As there are six projections (views) per PC, a vector v of six values is generated, representing our final "perceptual features". Finally, we map the vector v to the final score using a Machine Learning (ML) regression model called the Support Vector Regressor (SVR). These steps produce an efficient quality metric for static PCs that, based on the results disclosed in Chapter 4, is probably the new state of the art in PCQA.

ORGANIZATION OF THIS DISSERTATION

This dissertation is organized into 5 chapters: this introduction plus four chapters. Chapter 2 presents a brief review of the PC data structure and the intrinsic problems that arise when dealing with it. Later, it also provides the main concepts involved in the DL field, as well as some points that need attention when using learning-based techniques. Chapter 3 covers the techniques chosen to generate the projections, extract the perceptual features, and map from the features to the final quality score, as well as an overview of the final proposed architecture. Chapter 4 describes the experiments, the configurations used, and the results obtained. Finally, Chapter 5 provides our conclusions about the work.

CHAPTER 2

THEORETICAL FRAMEWORK

Working in the field of Point Cloud (PC) requires a knowledge of the main concepts and techniques to understand its problems and the proposed solutions. On top of that, Deep Learning (DL) is also a field of study that has a wide set of concepts and techniques. This chapter establishes the main knowledge for both fields, seeking to provide a consistent theoretical basis that will allow us to jointly understand both fields and guide our work.

2.1 POINT CLOUDS OVERVIEW

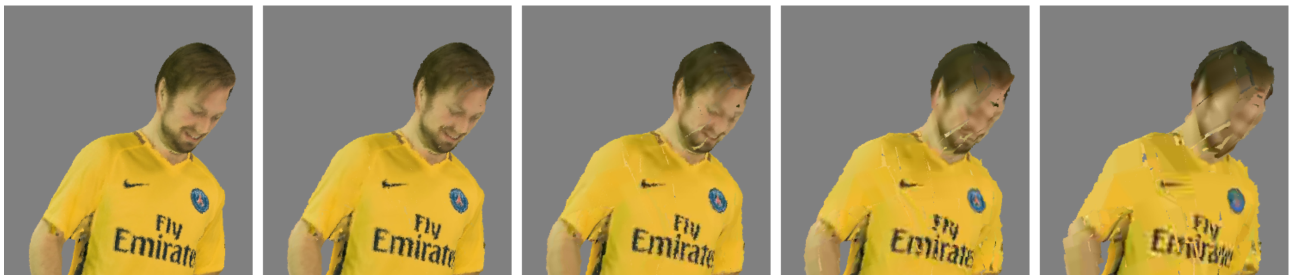

Figure 2.1. Example of a PC present in VSENSE/VVQDB database [19] and its variations with different levels of degradation.

PCs have a variety of applications, ranging from representing industrial tridimensional (3D) CAD models to autonomous driving [1] and virtual reality [2]. Among these types of usage, we will address one in particular, which is the new types of imaging technologies that are based on PCs and are called immersive imaging. Three data structures, PCs, holograms, and light fields, are useful for representing this type of content, and each of them has its particularities. Holograms, which were introduced by Gabor [20], consist of a form of recording light fields instead of simply recording an image formed by the lens. Light fields, on the other hand, collect radiance from rays in all directions, demultiplexing the angular information lost in conventional photography [21] and describing the distributions of light rays in space.

These new types of immersive imaging technology use more dimensions of the plenoptic

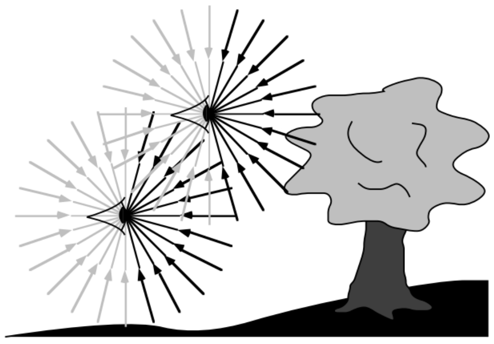

Figure 2.2. From Bergen & Adelson[22]: the plenoptic function describes the information available to an observer at any point in space and time. A real observer cannot see the light rays coming from behind, but the plenoptic function does include these rays.

illumination function than traditional representations, although, due to the high dimensionality of the function, they represent only an approximation. According to [22], the plenoptic function aims to represent what can potentially be seen; it is a systematic way to describe how the visual elements are related to the visual information. Therefore, it takes into account all possible viewpoints (V_x,V_y,V_z), the path of light rays at any possible angle (θ,ϕ), for every wavelength λ, at every time t, as shown in Figure 2.2. Therefore, the resulting function takes the following form:

$$P = P\left(\theta,\phi,\lambda,t,V_x,V_y,V_z\right). \tag{2.1}$$

Point clouds have gained greater acceptance for volumetric visual representation, compared to other immersive imaging technologies [23]. One of the reasons comes from the fact that surface reconstruction does not need to be performed when obtaining a PC, which is not true, for example, for polygon meshes 3D. In such a manner, PCs are a more direct and compact representation of immersive media, which makes them a more computationally efficient way to represent this type of content. This is a very relevant aspect of real-time immersive media systems.

Another relevant aspect of these three data structures is the support for Six Degrees of Freedom (6DoF). According to Diniz [24], degrees of freedom, in this context, represent the possible types of movement a body can make in the 3D space, which is divided into three translations: forward/backward (surging), up/down (elevating), and left/right (strafing). In addition, there are three rotations: yaw, pitch, and roll. Figure 2.3, taken from [25], illustrates these movements. In this manner, 6DoF gives the user the freedom to look in any direction

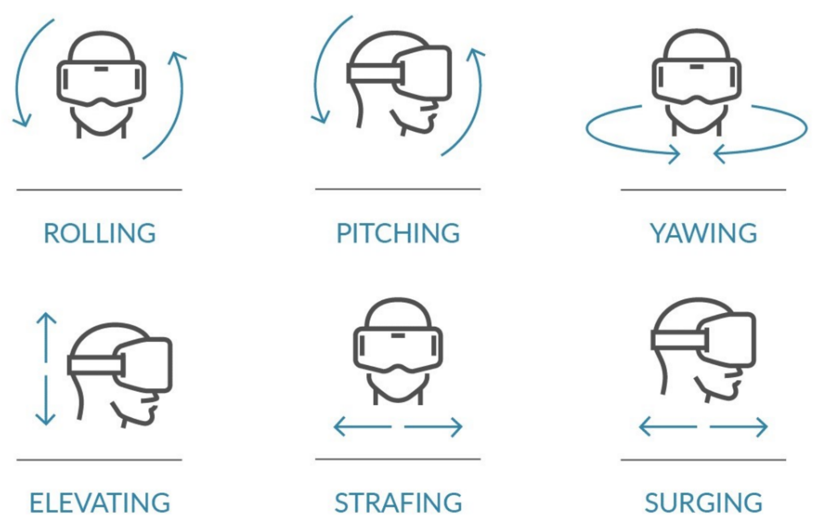

Figure 2.3. The first row illustrates the 3DoF possible movements, whereas the second row illustrates the additional possible movements for 6DoF. Source: Foundry 45.

and the ability to walk virtually through the scene, making 6DoF a prerequisite for immersive media systems.

Having support for 6DoF is another reason why PCs is gaining more and more relevance in the immersive media context. On the other hand, their representation requires a large number of points and, consequently, a large computational complexity as well, which limits their use in real applications [26]. Fortunately, researchers in the field are rapidly advancing [27, 28, 29] to enable the viability of this data structure by developing compression techniques.

In addition to advances in the compression research field, recent advances in 3D data capture technologies, such as camera arrays [30], RGB-D cameras [3], and Light Detection And Ranging (LiDAR) sensing [4], are also facilitating the viability of the use of PCs by making the process of obtaining this data structure easier and cheaper. It should also be noted that each type of capture technology has its particularities that can be used to process the final content.

Point Clouds vs 2D Images

One of the most popular ways to represent media is the digital image. Digital images are a set of pixels organized in a bidimensional (2D) matrix grid of a given number of rows, height h and a given number of columns, width w. Each unit value of this matrix represents a pixel. Pixels are the smallest elements in the digital image, and each of them has its intensity value

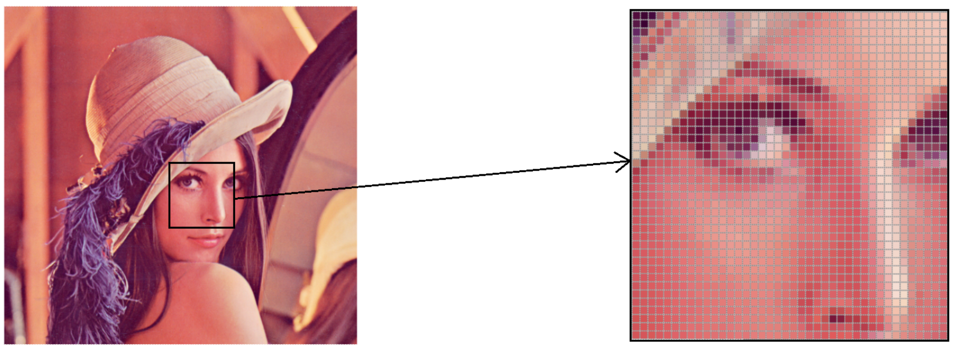

Figure 2.4. The Lenna image and a sample of it showing the corresponding pixels.

commonly represented by a byte (8 bits), thus each pixel has a range of 2^8 possible values of intensity. A visual representation of the pixels in an image can be seen in Figure 2.4. In this manner, a pixel is defined by the function $A = f(x,y)$ [31], where x and y represent the spatial information of where the pixel is located in the matrix and A is the corresponding discrete intensity. A single-channel image is called a grayscale image. A possible way to represent a colored image is by using the RGB format. In this case, the image is made up of three channels: one for the intensity of red, one for the intensity of blue, and one for the intensity of green. Thus, colored digital images in the RGB format are made up of three overlapping 2D matrices, each representing one color channel.

As mentioned above, PCs are collections of points in the 3D space. However, contrary to the images, PCs do not have a regular grid on which to lay, so they are said to be irregular data structures. Each point has spatial coordinates (x, y, z) and the corresponding attached information can be represented by $A = f(x, y, z)$. It follows that when storing a PC, in addition to storing the information of each point (the color channel intensities in the case of images), one must also store the spatial coordinates of each point (x, y, z), which can be floating point numbers, given the continuous nature of space.

This leads to two key aspects of a PC: the spatial information of each point, which is said to be the geometry aspect, and the corresponding information of each point, known as the attribute. Thus, irregularity is a characteristic of the geometry aspect of a PC. On top of that, within the total space that a PC represents, just a small portion is occupied by points, which leads to another important characteristic of a PC: the sparsity of the geometry. In short, irregularity is the lack of a regular grid to form the support for the attributes, and sparsity is the global low density of the total space given the small number of occupied points.

The lack of regularity naturally increases the storage size of PCs due to the need to encode the geometry information together with the attributes. This problem does not affect images, since by knowing the image size in both dimensions one can encode the intensities of the pixels sequentially and still be able to correctly rebuild the original image. More details of how this affects the usage of PCs will be described in the next section.

2.1.1 The Compression Task

With all that said, the large space in a disk that PCs requires compromises not only the storage but also the transmission since channel limitations arise when transmitting above a certain rate [32]. These implications call for the necessity of data compression to make the usage of PCs viable in real applications. In that sense, compression appears to be an enabler technology, making the usage of not only PCs but also of many types of data viable in real-world applications by compressing them to a level where devices can handle transmission and storage.

As stated by Sayood [33], when a compression technique or compression algorithm is mentioned, one refers to two algorithms. The first is the compression algorithm (encoder) that takes an input X and generates a representation X_c that requires fewer bits, and the second is a reconstruction algorithm (decoder) that operates on the compressed representation X_c to generate the reconstruction Y, together they form the codec, the pair coder-decoder.

The main challenge then is to develop a codec that, understanding the aspects of the data, can provide a good spot in the trade-off compression rate vs. content quality. When compressing a PC a large number of points is not the main problem, since one can have, for example, a video in 4K resolution (3840 x 2160 pixels) that occupies a huge amount of space on the disk in raw format and still be able to compress it to make its usage viable, while preserving an acceptable quality for the content. On the other hand, the increased complexity of PC compression comes from the fact that the information lies on a sparse subset of an irregular space, contrary to images that have a regularly sampled grid. This makes the neighborhoods irregular and compromises the relationship among close points and, consequently, the usage of context in compression techniques.

Many techniques have been developed to work around these innate characteristics of PCs. Among them, a widely used technique for PC coding, known as voxelization, addresses the irregularity problem by quantizing the space, allowing us to represent it regularly. The voxelization groups point together into voxels, which are a set of unit cubes, providing a sense of resolution to PC. For example, 8 bits of resolution mean 2^8 cubes in each dimension. As defined by Xu *et al.* [34], voxels, similar to pixels in an image, are abstracted 3D units with predefined volumes, positions, and attributes, which can be used to structurally represent discrete points in a topologically explicit and information-rich manner.

After the voxelization process, an occupied voxel is a voxel that contains at least one point inside. Voxelization allows PC to be represented in quantized space or through the octree representation. In such a manner, this technique helps in the problem of geometry irregularity, since, by quantizing the space and giving it a sense of resolution, the attributes now have a regular grid to lay on. However, sparsity remains a problem since only a small portion of the voxels are occupied. Figure 2.5 shows the effect of the voxelization process in PC.

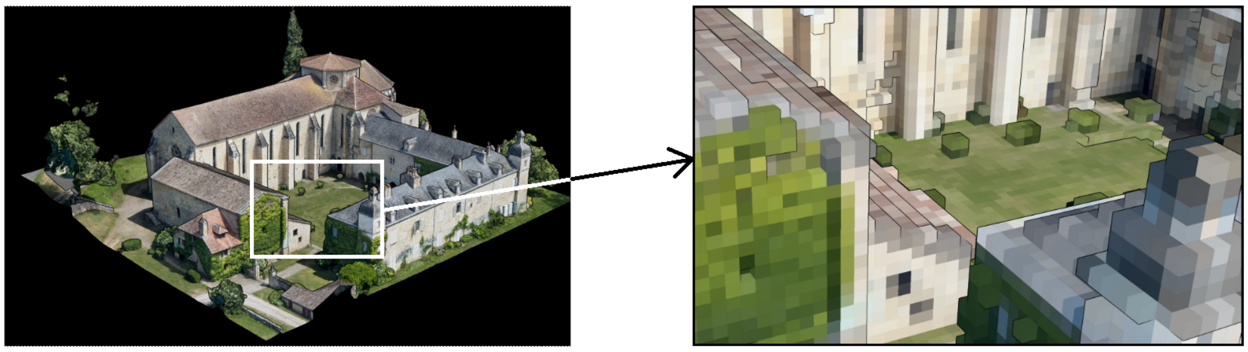

Figure 2.5. Visualization of a portion of a PC after the voxelization process.

In addition to tackling the irregularity problem, voxelization brings a new set of possibilities to approach the PC compression task. The geometry can now be seen as a binary signal over a regular grid in a way that the possible states of a voxel are occupied and not occupied. Therefore, geometry compression can be seen as a predictive problem, where one can build a probability distribution of occupancy and predict if a voxel is occupied or not based on context, since the space is now regularly sampled.

Octrees are another important technique in the context of PC compression that comes with voxelization and were introduced by Meagher [35] in 1982. Octree-based methods hierarchically partition the space following a tree-based data structure. For the tree, each node has exactly eight children; in this manner, the space is divided into eight equal cubes, the major space is the node, and the cubes are the children. If a cube contains a point inside, this cube is divided again into eight new cubes. The process goes on recursively until the desired Level of Detail (LoD) is achieved — simply put, LoD can be seen as the density of a PC region; the higher the density, the higher the precision and richness of details and vice versa — or until the new cube does not contain any point inside. Therefore, octree-based methods are a good approach to the problem of sparsity in PCs, since when a cube is empty, all its subcubes are also empty, which makes the octree only represent the occupied portion of space. Figure 2.6, taken from [36], illustrates the recursive partitioning process of a PC using the octree-based method.

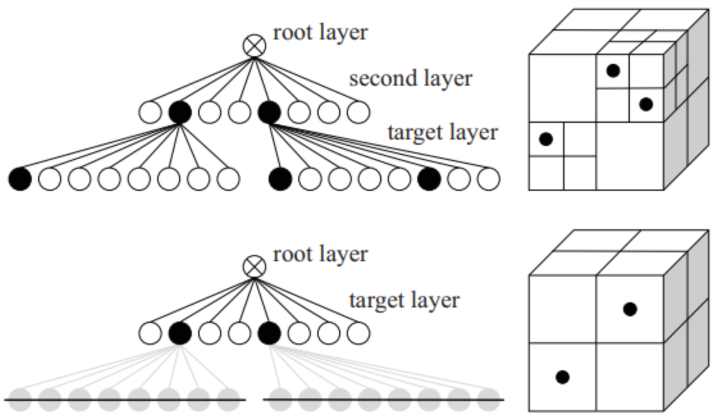

Figure 2.6. Octree partition example. The first representation shows a partition that goes on until every point is represented, whereas the second represents the partition that goes on until the desired LoD is reached. Figure taken from [36].

Considering the possibility of applying voxelization to a PC, the compression problem can now be addressed in two ways: with Point-Based or Voxel-Based approaches. Point-based techniques process PC as a raw set of points. On the other hand, voxel-based techniques consider PC as a binary occupancy signal on a regular voxel grid, as commented before. Point-based approaches are typically more suitable for sparse PCs, while Voxel-based approaches tend to perform better on denser PCs. Each approach also has disadvantages; Point-Based approaches may have to deal with a very large number of points, while Voxel-based approaches may have to deal with very high dimensionality in data caused by the regularization of the space. When choosing the best technique to use, the upsides and downsides of each need to be taken into consideration. Also, the nature of the PC depends on the method used to obtain the data, and this prior information can help in choosing the correct way to address the compression problem.

In addition to Voxel-based or point-based approaches, one can choose to address the compression of PCs by its aspects. In this manner, it is possible to compress each aspect individually, and the compression problem can be addressed only for the geometry aspect, only for the attribute aspect, or both. Some peculiarities need attention when compressing the aspects separately, such as transferring the attributes when performing lossy compression for geometry.

Recent advances in the field have brought widely-known techniques and standards. Among them, MPEG has released Geometry-based Point Cloud Compression (G-PCC) and Video-Based Point Cloud Compression (V-PCC). According to Schwarz *et al.* [37], G-PCC takes advantage of native 3D representation while decomposing the 3D space into a hierarchical structure, that is, octree-based division, while encoding each point as an index of the cube to which it belongs. On the other hand, V-PCC tries to focus on the problem of compressing Dynamic Point Clouds (DPCs), taking advantage of already existing video coding technologies, MPEG-4, HEVC, etc., by projecting points onto 2D frames.

Point Cloud compression is still an open problem, and several works in the field leverage powerful approaches, either more traditional, such as octree-based ones, or more modern, such as data-driven approaches. The compression problem for PCs has a challenging complexity, and the algorithms proposed to address it are progressively following this complexity to gain performance [27, 28, 29]. This gain in complexity compromises the usage of better techniques in real-world applications, leaving space for simple but already very good approaches such as G-PCC and V-PCC.

2.1.2 Point Clouds Quality Assessment (PCQA)

Along with the necessity of compressing PCs to enable its use in real-world applications, comes the necessity of evaluating the resulting content. This is done to ensure the best possible Quality of Experience (QoE) and to maintain acceptable degradation levels of the PCs coding techniques. Point Clouds Quality Assessment (PCQA) is the field that takes this responsibility and allows one to assess the quality of content by developing quality metrics, which can be objective or subjective. Users will typically consume 3D PC content with displays very close to the eyes, through Head-Mounted Displays (HMDs), so PC visual impairments can easily degrade the user experience, confirming the need for ways to assess content quality during the compression process.

Subjective metrics are provided by human assessment based on the scores given to the content. They provide ground truth for objective metrics since they are the best representation of the quality perceived by the final user. Objective metrics, according to Diniz [24], automatically evaluate the quality of content based on algorithms that optimally have the highest possible correlation with subjective ground truth.

Quality metrics are also used to optimize, evaluate, or tune different compression techniques to ensure that their final result is the best possible QoE. In this manner, as subjective metrics are dependent on human evaluation, they are very expensive and difficult to obtain, making their usage in this type of application compromised. Therefore, objective metrics are useful, as they provide a fast and objective assessment of content that correlates well with human evaluation, allowing compression techniques to be optimized to generate the final content that will provide the best possible QoE.

2.1.2.1 Subjective Quality Assessment

Subjective methods used to evaluate PCs take advantage of some protocols used to evaluate 2D images and videos. As currently there is no specific standard or recommendation for subjective quality assessment [38], questions related to how content should be processed and displayed still cause uncertainties for PC subjective quality assessment.

So far, the proposed PCQA methods use standardized methodologies for subjective quality assessment, such as Absolute Category Rating (ACR) and Double Stimulus Impairment Scale (DSIS) [39, 40]. The difference between these methods lies in the fact that the content has two possible states: the reference state and the degraded state. When subjects are evaluated with the ACR methodology, the content is shown individually, so the user does not have a way to

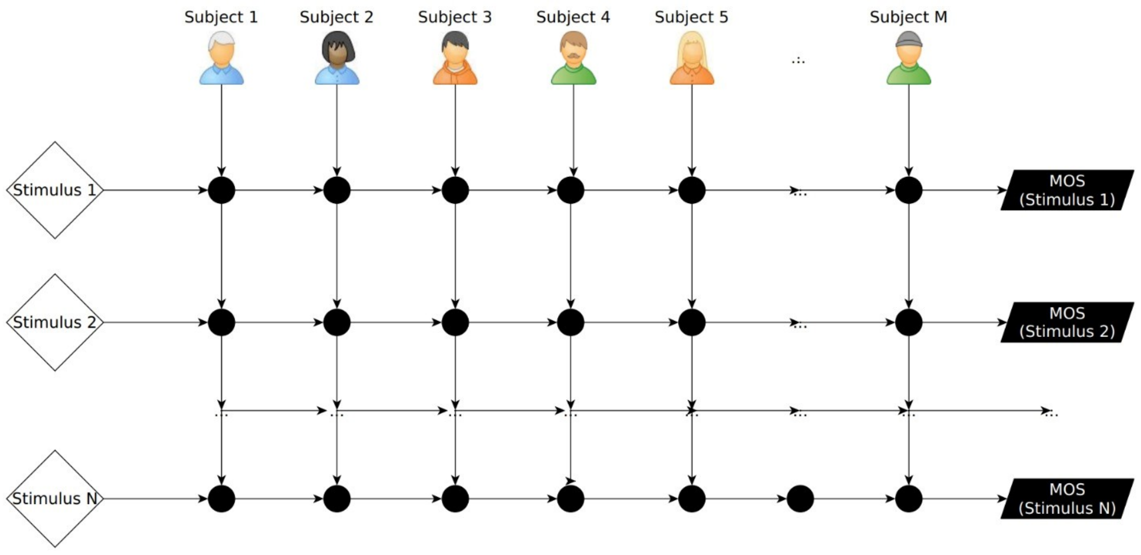

Figure 2.7. The subjective process of evaluating the quality of content. Note that each subject evaluates every stimulus and the score of each stimulus is the aggregation of the scores. Courtesy of [44].

compare the content shown with a reference, which is a single stimulus method. On the other hand, the DSIS methodology shows the reference content and the degraded content; in such a manner, the user can compare both contents, characterizing a double stimulus method. The subjects' scores are typically mapped onto the Mean Opinion Scores (MOS) scale, ranging from 1 (very bad) to 5 (very good). Figure 2.7 illustrates the process of subjectively assessing the quality of content.

Subjective tests were conducted to compare quality perception on different types of displays [41, 42]. These tests only considered geometry-related types of distortion and found that human perception of distortions has a high correlation between different visualization devices, and the rendering method of PC may influence the perceived quality. There are also works, such as the one from Zhang *et al.* [43], that suggest that human perception is less tolerant to geometry degradation than color degradation on PCs, which is a very relevant fact for the PC compression problem.

Despite the uncertainty that permeates the rendering of PCs before displaying it, the works on PC subjective quality assessment are at a satisfactory level where they can support the development and improvement of both objective metrics and compression techniques.

2.1.2.2 Objective Quality Assessment

Subjective quality assessment is an expensive and time-consuming task. Applications that strongly depend on its compression techniques, such as video-streaming services, also depend on providing the content with the best possible quality for a given compression rate. In these cases, subjectively assessing the quality of numerous videos with different compression codecs and parameters is infeasible. Therefore, there is a need to automatically extract the quality of the content. This is the role of objective metrics, as they provide a quality score using only content information. In that sense, they are divided into Full-Reference (FR), Reduced-Reference (RR), and No-Reference (NR). FR metrics rely on the complete content of the reference to estimate the quality metric for the impaired content, while NR uses only the features of the impaired content without reference information. RR uses only a portion of the reference content to estimate the quality metric.

Given that PC is a relatively new type of immersive media format, FR methods for objectively estimating content quality are the most chosen in the field of PCQA. This is due to the fact that having both reference and altered content allows the algorithms to work without prior knowledge of the content [24]. The works that built the foundation for this field introduced the famous point-based metrics: Point-to-Plane (P2Pl) [45], Point-to-Point (P2Po) [46], Plane-to-Plane (Pl2pl) [47], which is a variation of point-based metrics, using the angular distance between the tangent planes of each pair of points of a reference and the test PCs to estimate the perceptual distortion.

According to Diniz [24], point-based methods use a relation between the points of reference and the test PCs. Given a point k in reference PC R and the closest point i in the test PC D, a distance measure is used to estimate the error in the test content. Euclidean or Hausdorff distances are commonly chosen distances used in this context and are performed symmetrically, that is, the distances from the reference PC to the test PC and from the test PC to the reference PC. After all distance computations, the cumulative error of all points is used to estimate the quality metric.

As also stated by Diniz [24], the metrics P2Po, P2Pl, and Pl2pl are types of calculation errors between points that can estimate the geometry degradation in PCs. For each of them, first, one needs to find a point-to-point correspondence between PCs, normally this is computed using the nearest neighbor search based on the coordinates of the points. After finding the pair of points that are correspondent $\{k \in R,\ i \in D\}$ the geometry error can be calculated as:

$$d(k, i) = \sqrt{(k - i)^2}. \tag{2.2}$$

After the distance-based error has been calculated for each pair of points, the global error is estimated using the Mean Squared Error (MSE) technique [48]:

$$P2Po_{MSE} = \frac{1}{N} \cdot \sum_{k=1}^{N} (d(k, i))^2\,, \tag{2.3}$$

or with Peak Signal-to-Noise Error (PSNR):

$$P2po_{PSNR} = 10 \cdot log_{10} \frac{3p^2}{P2po_{MSE}}, \tag{2.4}$$

where N is the number of points in the reference PC and p is $2^{pr} - 1$ bits, which is the peak distance with pr being the dynamic range of the PC coordinates [39]. The quality metric is estimated by the maximum error between the reference content and the test content, and vice versa [24].

To estimate the color error, metrics based on P2Po can also be used following the same procedure for geometry, but with color distance formulas instead. Using the ITU-R Rec. BT.709 colorimetry equations, the points' colors are converted to the YCbCr color space, and, for each component of the color channel, the measures MSE and PSNR are calculated.

So far, we have presented scores that are based only on point information. However, we can see PCs as representations of surfaces given a set of points [24]. Taking advantage of this, P2Pl uses the projection of a correspondent point i in a degraded PC to the plane perpendicular to the normal of the point k in the reference PC together with the P2Po distances to estimate the quality metric. Pl2pl still leverages plane information but estimates the quality score with angular similarity differences between the tangent plane of the point k in the reference PC and the tangent plane of its correspondent point i in the test PC.

Before computing the plane metrics, the plane information must be obtained, which is done using local neighborhoods and the calculation of the normal vector of each point [45]. These vectors represent a local surface plane and, according to Diniz [24], points in the test PC that are close to the reference plane will lead to small errors even if they are far from the reference point. The P2Po, P2Pl and Pl2pl metrics depend on the relationship and the ability to establish reliable connections between PCs. However, the irregular aspect of PCs causes the data to be unstructured, significantly increasing the difficulty of establishing the necessary relations between the reference and the test PCs. Additionally, there are many types of coding errors that must be considered when processing a PC so that these metrics do not sense properly. They also fail to assess both geometry and color. All these facts together compromise the performance of the aforementioned metrics. However, these metrics are still widely used by MPEG [49] and some are even considered state-of-the-art, such as [39].

Together with advances in the field of PC compression, data-driven approaches are very popular and are providing astonishing results in their applications. Gao *et al.* work [29] take advantage of the power of data-driven techniques with the use of Variational Autoencoders (VAEs) combined with neural graph sampling to provide a powerful PC compression technique. There are also some works in the field of PCs that use data-driven techniques, such as the approach from Bello *el al.* [50], which points out that local point relationships are more effective for modeling a PC data-driven approach, providing other works with useful knowledge when modeling this type of technique. In the field of PCQA, Liu *et al.* [27] proposed the first NR method that uses a data-driven approach by applying a type of architecture known as Convolutional Neural Network (CNN).

2.2 DEEP LEARNING OVERVIEW

Machine Learning (ML) is the field of study to find out how computers learn from data [51]. It has become some sort of buzzword today and can be treated as a new programming paradigm [52]. If in traditional programming the inputs are the data and the rules to output the answers, in machine learning, what is sought is to discover the rules that are intrinsic to the data.

However, in this dissertation, we also use Deep Learning (DL) techniques. In general, Deep Learning comes from the need to solve some problems that classical machine learning arose regarding the curse of dimensionality and regularization (see [51] and [53]) and it provides a very powerful framework for supervised learning.

2.2.1 What are neural networks?

NNs are the foundation for deep learning. When we talk about a deep learning model, we are necessarily talking about neural networks. They can come in many forms, such as Convolutional, Recurrent, and so on. Neural networks are made up of layers and each layer is made up of neurons, also called *units* or *nodes*. Each neuron has an activation function that dictates its output. This is based on the idea that a biological neuron receives input and activates itself on the basis of this input value. However, neural networks are more function-approximating machines than a representation of the human brain [51].

The most common type of NN is Feedforward Neural Network (FNN), which means that the layers do not have any feedback connection. For an illustrated example, see Figure 2.8. FNNs are also called Multilayer Perceptrons (MLPs). As mentioned above, NNs aims to approximate (or represent) functions. The family of functions that a network can represent is called the *capacity* of the network. Usually, we want this family of functions to include nonlinearities, to represent more complex problems. Furthermore, NNs should select the most appropriate input according to the properties of the data.

It is important to define some terms: we say that the first layer of a NN is *input layer* and the last layer is *output layer*. All the other layers in between are called *hidden layers*, as shown in Figure 2.8. We can say that a NN with one hidden layer is a *shallow neural network*, and a NN with two or more hidden layers is called a *deep neural network*. Neural Networks also have a set of hyperparameters. They are the variables that dictate the learning algorithm and so they are not learned by the algorithm. These variables are set before the training process.

2.2.2 Activation Functions

The objective of FNN is to approximate an unknown function. To achieve this, multiple operations are performed and passed through, layer by layer. Each layer is composed of multiple units, and each unit receives the output of a computation. Usually, one largely used model is linear regression, given by:

$$f(\mathbf{X};\mathbf{W},\mathbf{B}) = \mathbf{X}^{\mathbf{T}}\mathbf{W} + \mathbf{B}, \tag{2.5}$$

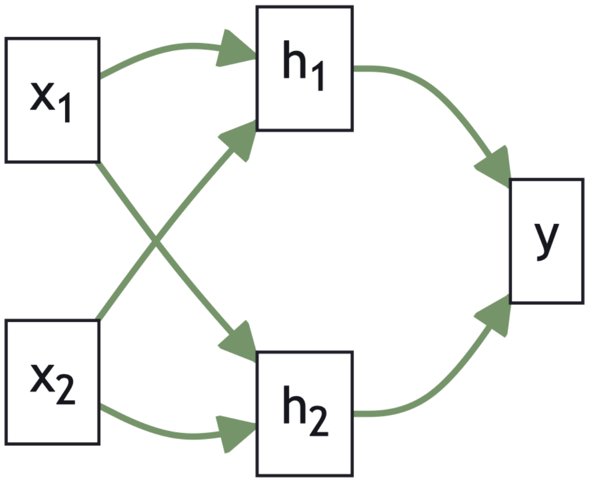

Figure 2.8. Example of a simple feedforward neural network with one hidden layer.

where W are *weights* and B are *biases.* Then, each column of these matrices is used for each unit, and so the operation that each unit of a layer will perform is $f(x_i; W_i, B_i)$. Every unit of the network is a linear regression model. The network then updates the parameters W and B to achieve the best possible approximation of the desired function.

So, suppose that a given FNN has 2 hidden layers that perform the linear regression model. The input of the second hidden layer is exactly $f^{(1)}(\mathbf{X}; \mathbf{W}, \mathbf{B})$, so the output of the network is $f^{(2)}(\mathbf{H}; \mathbf{w}, \mathbf{C})$, where $\mathbf{H}$ is the output of the first layer, $\mathbf{W}$, $\mathbf{w}$ are the weights and $\mathbf{B}$, $\mathbf{C}$ are the biases. Therefore, this yields the following.

$$f(\mathbf{X}; \mathbf{W}, \mathbf{B}, \mathbf{w}, \mathbf{C}) = f^{(2)}(f^{(1)}(\mathbf{X}; \mathbf{W}, \mathbf{B}); \mathbf{w}, \mathbf{C}). \tag{2.6}$$

Since the composition of two linear functions is still a linear function, a neural network that performs this model will necessarily be linear. Although linear operations are desirable for ease of computation, this severely limits the capacity of the network.

To tackle this problem, we use **activation functions**, which applies a nonlinear transformation to the result of the operation performed by the NN unit, for example, the one provided by linear regression. This way, in the network described in the example above, a node of the first hidden layer will follow these steps:

1. Compute the model equation, such as $\mathbf{h} = \mathbf{x}^\mathbf{T}\mathbf{w} + \mathbf{b}$;

2. Apply the activation function g to the result: $l = g(\mathbf{h})$;

3. Return l as the output of the node.

In such a manner, the network gains the ability to select its inputs and also benefits from the advantages of nonlinearity, e.g. representing more complex problems. Arguably, the most popular activation function is the Rectified Linear Unit (ReLU):

$$g(x) = max(0, x). \tag{2.7}$$

One of the great benefits of ReLU is that it does not have *gradient saturation* [51], which means that its gradient does not reduce significantly and close to zero as x grows. This allows for much faster convergence of gradient descent, which is described in Section 2.2.3.2. Furthermore, the function ReLU is easy to compute, which gives it an advantage over other activation functions, such as the Sigmoid function, which requires exponentials to be calculated.

2.2.3 The Process of Learning

It is important to define exactly what learning means in the context of ML and DL. As defined by Mitchell [54], a computer program is said to learn from experience E on some class of tasks T and performance measure P, if its performance in tasks in T, as measured by P, improves with experience E.

2.2.3.1 Optimization

As stated above, traditional programming takes data as input and passes them through a set of rules designed by the programmer to obtain the output. In supervised ML and DL we take the input and present to the computer the expected outputs; the task now is to find the rules that guide the inputs to the expected outputs, and this usually requires a lot of numerical computation. As Goodfellow *et al.* [51] described and as the previous definition of learning from Mitchell [54], the problem of finding this set of rules is mathematically modeled and solved by means of an iterative process rather than analytically finding the solution and providing a formula to it.

In the context of ML and DL, to learn the correct rules that lead to the expected outputs, one needs to tell the computer if it's guess was correct or not and, most importantly, tell how much it was wrong, allowing the definition of how to improve, so it can guide itself to the expected direction. This is done by the definition of a function known as a cost function, a loss function, or an error function. As it is a measure of error, optimization is performed to find the arguments that minimize the cost function [51], as it forces the computer to get closer to the expected output, and this is the learning process.

However, this task can get very complicated very easily. In the context of NN training, optimization can become a very challenging process, since cost functions can be very complex with a huge set of parameters and have some fancy representations that add more complexity to the process. To contour that, we usually do not care about finding the exact minimum of a function, but we seek to sufficiently reduce its value to obtain a good generalization error [51], that is, a good capacity. How exactly this is done is covered in more detail in the following sections.

For now, it is important to point out that it is common to have three different slices of data for training and testing a DL algorithm: the training set, the validation set, and the test set. They can be drawn from one or more datasets. This division is made in order to prove the capacity of the model in unseen data. First, the model is trained in the training set, usually the larger one. After that, it is common to use the validation set to tune the model's hyperparameters. Lastly, the test set is used to calculate the generalization error of the model. This division between test set and validation set is made in order to avoid using the test set to make any decision about the model, thus increasing the reliability of the generalization results generated on the test set.

2.2.3.2 Gradient Descent

To minimize a function $f(x)$ by its parameters, Goodfellow *et al.* [51] points out that one needs to know how to change x to x' in a way that $f(x') < f(x)$. The derivative $f'(x) = \frac{dy}{dx}$ gives the slope of $f(x)$ at the point x. In such a manner, the derivative describes how a change in the inputs would affect the output of a function: $f(x + \epsilon) = f(x) + \epsilon f'(x)$. However, a very useful characteristic of the derivative is that it tells us how to change x to increase $f(x)$, allowing one to minimize $f(x)$ by changing x in the opposite direction of $f'(x)$ and this technique is called gradient descent [55].

The notation $f'(x)$ represents the derivative of a single variable function. For multivariate functions, the concept of partial derivatives is used in the way that the partial derivative $\frac{\partial}{\partial x_i} f(x)$ describes how f changes as only the variable x_i increases at the point x. The gradient denoted as $\nabla_x f(x)$, is the generalization of the derivatives of a multivariable function and is represented as a vector that contains the partial derivatives of the function. As in the single variable derivative, the gradient points in the direction of a larger growth; therefore, one can minimize f by iteratively following the opposite direction $(-\nabla_x f(x))$, which is exactly the gradient descent approach for multivariate functions.

Therefore, optimization occurs by applying the gradient descent algorithm, which implies that the function parameters need to be updated following the negative of the gradient scaled by a factor known as the learning rate, denoted as ϵ, which determines the step size at each update. In this manner, the new value of x, denoted x', can be obtained as

$$x' = x - \epsilon \nabla_x f(x). \tag{2.8}$$

The learning rate ϵ is one of the NN hyperparameters (as commented before, they are the set of parameters that dictate the behavior of the learning algorithm). Usually, it is a small constant, and one good way to tune it is to try different values and compare the different behaviors of the loss functions over time; the ϵ that gives the best learning curve should be chosen.

Unlike most ML algorithms, the non-linearity of the DL models can cause the cost functions to be non-convex [51]. Hence, the task of finding the global minimum becomes very complex, making the gradient descent algorithm drive the cost function to a low value, but not necessarily the lowest one. This goes in contrast to the convex case, where any local minimum is guaranteed to be a global minimum.

Moreover, the complexity of the cost function makes the optimization problem very sensitive to the initial value of the function's parameters; if they are initialized in a good spot, the algorithm can converge to a local minimum that represents a satisfactory low value. However, if the parameters are initialized in a bad spot, the algorithm could lose itself in the iterations and not even converge. In such a manner, the gradient descent continues iteratively until convergence or until a stop criterion is met, since convergence is not guaranteed in some cases.

Non-convex functions have not only local minimums, but also local maximums and saddle points, and they are all known as critical points since the derivative at these points is zero. Saddle points represent the points whose neighborhoods have points that are higher and points that are lower than themselves. Furthermore, Dauphin *et al.* [56] points out that the existence of saddle points contributes negatively to training speed, since their neighborhoods have high error plateaus, giving an illusory impression of a local minimum. Complex cost functions have many critical points and flat regions, which explains such a complexity in the task of finding the function's global minimum.

Even with all this complexity involved, DL models can be trained to perform very well in the training set and have a very good generalization capacity. Some works, such as [57, 56], start to suspect that the search for a global minimum is not that important, since most local minimums already represent a point in the parameter space with a significantly low-cost function value, making the model output sufficiently close to the expected outputs and promoting a good generalization error.

In this sense, gradient descent is the process in which NN effectively learns, since the parameters, weights, and biases are tuned in a way that minimizes the cost function, making the outputs as close as possible to the expected one. The combination of weights and biases dictates the intrinsic rules learned from the training data and, in the process, NN also gains the power of generalization, making it possible to generalize the learned information for data that were never seen before. Gradient descent relies on the derivatives to correctly update the weights and biases at each iteration; these derivatives with respect to each weight or bias are calculated using an algorithm called *backpropagation*. In short, it computes the derivatives for each training sample after a forward pass using the calculus chain rule in a backward pass. After each forward pass of the training sample, it averages the derivatives and uses them to update the parameters, completing one iteration of the training process. More details about backpropagation are covered in Section 2.2.5.

This reinforces the idea that training NNs is more an iterative task than an analytical search for the solution to the problem. Also, we point out that by averaging the derivatives for every training sample, the algorithm updates the parameters more efficiently, helping to converge the cost function. However, this also drastically slows down the training process since one needs to pass through all the training sets to perform a single update in the parameters. Furthermore, this also scales with a large amount of data normally used to train a NN, making techniques such as Stochastic Gradient Descent (SGD) very helpful in the process.

There are also some concerns that one should have when training a deep NN with gradient descent. The application of the calculus chain rule can cause two problems that for a long time were a hindrance to NN gradient descent training. The so-called vanishing or exploding gradients that arise in the deep NNs can make learning infeasible. The vanishing gradient comes from the chain multiplication of small factors (< 1) which makes the gradient very small at some point of iteration and, as a consequence of equation 2.8, the update of the parameters becomes insignificant. On the other hand, exploding gradients come from chain multiplication of large values (> 1), making learning unstable due to the large value of the gradient. In this manner, the problem of vanishing/exploding gradients during training should be avoided by choosing the most suitable technique to initialize the NN parameters or by applying some technique that acts directly on the gradient value, such as gradient clipping for exploding gradients or techniques such as the one proposed by Srivastava *et al.* [58] for vanishing gradients.

2.2.3.3 Stochastic Gradient Descent (SGD)

As exposed in Section 2.2.3.2, training a NN to obtain good generalization requires a large amount of data, and each full pass through the data to update the parameters is time-consuming and computationally expensive. In addition, some training sets may have billions of examples, which makes the time for a single update prohibitively long.

As stated by Goodfellow *et al.* [51], the main insight of Stochastic Gradient Descent (SGD) is that the gradient is an expectation, which can be approximated using a small set of samples. This implies that training can be performed using a minibatch of m' samples drawn uniformly from the training set that contains all m samples. The common values for m' range from 1 to a few hundred and remain constant as m increases. It is pivotal to draw minibatch samples randomly; the independence among the samples is crucial for the gradient to be computed in an unbiased manner and, normally, it is sufficient to just shuffle the dataset before storing it.

An important property of SGD is the fact that the computation time per update does not increase with increasing number of training samples m. In this way, convergence can be reached even if m is very large, and SGD algorithms may converge to some criterion even before the entire training set has been processed [51]. Goodfellow *et al.* also highlights that convergence in traditional optimization problems is different from convergence in the models ML and DL. Traditional optimization stops iteration when a local minimum is reached, while optimization based on SGD in ML and DL does not. In turn, it continues to try to minimize the cost function until the convergence criterion based on early stopping is satisfied. Typically, the criterion is satisfied once the overfitting starts, which implies that the algorithm may stop when the derivatives still have large values. Overfitting and Underfitting are covered in more detail in Section 2.2.6.

SGD is an algorithm that takes advantage of the fact that one can see the gradient of the cost function. It statistically approximates the gradient by computing the derivatives using random subsets of the training set. Computing the gradient using only subsets of the whole dataset significantly reduces the number of computations required at each batch and the total training time. In that sense, SGD is the technique that made the training of NNs with huge amounts of data possible.

2.2.4 Batches

This text has briefly discussed “training” and “generalization”. It is mandatory in ML and DL to *train* and then *test* the model/NN one is building to obtain the desired results. For that, the data are separated into two subsets: the training set and the test set. It is important to shuffle and sample to maintain the statistical characteristics of the dataset. Then, one set, larger, is used for training, and the other, smaller, is used to test the model/architecture.

The more training data, the better the algorithm can generalize. However, the amount of data one can have is limited. Also, more data means that the algorithm will be more computationally expensive, and hardware is also limited for several reasons. Therefore, different techniques have emerged to overcome these limitations, and some of them are data augmentation and SGD, as discussed in Section 2.2.3.3. This section focuses on batch and mini-batch techniques, along with the use of epochs to train a NN.

As mentioned above, the SGD algorithm consists of taking batches of data from the training dataset and computing the gradient descent over these samples. How NN is trained with this algorithm iteratively is by separating the training data with batches of a specified size and then passing these batches through the network. Every time a batch is propagating, NN is updated. When NN has passed through all the data, **epoch** was done. The number of epochs and the size of the batches can vary, but it is common practice to select batches in powers of 2 since this increases memory efficiency. Also, if the number of samples in the whole data is not divisible by the batch size, the last batch will be smaller than the rest.

Training NN in batches can reduce the convergence time. Of course, this comes at the expense of less accuracy, but separating the data allows for parallelized implementations and higher speed. This is the advantage of using batches. For epochs, their usefulness is related to the convergence of the gradient. Increasing the number of totals passes through the network enables more iterations and tackles underfitting. However, it is important to find a balance to this quantity to avoid overfitting.

2.2.5 Backpropagation

During a NN training process, information flows in two directions. The first one, called the forward pass, takes the input x and passes it through the network connections performing the corresponding operations with the weights and biases. This process continues until the network generates the outputs $\hat{y}$ and the corresponding cost function $J(\theta)$. The second, called the backward pass, or backpropagating algorithm [59], takes the information at the end of the network and backpropagates it to compute the gradient numerically.

Note that the gradient is computed numerically instead of finding the gradient analytically and then computing it, which is a computationally expensive process [51]. The backpropagation algorithm, which is the algorithm that computes the gradient, together with gradient descent or SGD is the core of the learning algorithm used to update the parameters of NN at each iteration.

Using the chain rule of calculus, the gradient $\nabla_x f(x,y)$ for a function f composed of other functions that have known derivatives. In the context of DL, the function of interest is the cost function $J(\theta)$ whose gradient for the parameters θ is $\nabla_\theta J(\theta)$. To compute the gradient numerically, backpropagation is the algorithm that performs the chain rule of calculus in a highly efficient manner [51]. The main concept behind the chain rule can be understood intuitively when we have the functions $y = f(x)$, $z = h(f(x)) = h(y)$, the chain rule states:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}. \tag{2.9}$$

In the context of DL we are often not interested in the scalar case, but in the case where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, causing $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $h : \mathbb{R}^m \mapsto \mathbb{R}$. Consequently, the derivatives become partial derivatives and are denoted as follows.

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}. \tag{2.10}$$

Moreover, in DL context, vector notation might be required to better represent the equations, therefore, the derivatives in vector notation become:

$$\nabla_x z = (\frac{\partial y}{\partial x})^\top \nabla_y z. \tag{2.11}$$

The computation of the gradient in vector notation using the chain rule of calculus depends on the computation of $\frac{\partial y}{\partial x}$, a $m \times n$ matrix known as the Jacobian matrix of f, which is:

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}_{m\times n} \tag{2.12}$$

Therefore, we can see that the gradient of a variable x obtained by the chain rule by backpropagation is computed by multiplying the Jacobian matrix $\frac{\partial y}{\partial x}$ by the gradient $\nabla_y z$. The notation presented above helps us to understand the steps required to compute the derivatives. However, despite the complexity of the notation, backpropagation computes the derivatives numerically, leveraging the use of computational graphs to do that in a very efficient way.

In computational graphs, each node indicates a variable that may be scalar, vector, tensor, or other convenient type. Variables are associated with operations, which are just functions. Moreover, if a variable y is calculated by evaluating a function in a variable x, the variables x and y are connected by a directed edge that goes from x to y. Figure 2.9, taken from [51], helps visualize a computational graph and the relations between variables. This is the technique used in libraries such as Torch7 [60] and TensorFlow [61].

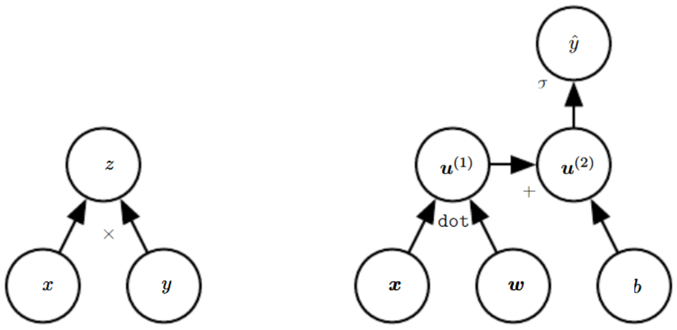

Figure 2.9. Example of a computational graph. Note that they can be used to represent simple operations like the one on the left that is just a multiplication, as well as more complex operations like the one on the right $\hat{y} = \sigma(x^\top w + b)$. Source: Goodfellow *et al.* [51].

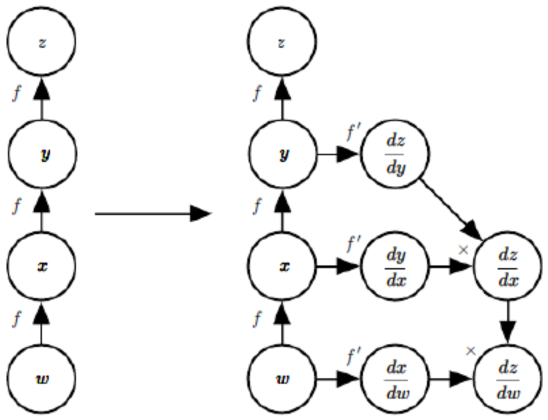

Figure 2.10. Computing derivatives using the symbol-to-symbol approach on computational graphs. Source: Goodfellow *et al.* [51].

In summary, backpropagation is the algorithm that computes the derivatives of the output concerning the NN parameters. Gradient descent uses these derivatives to update the NN parameters, which minimizes the cost function, thus completing the learning process. NNs are represented by computational graphs whose connections define the path that information in the input must take to generate the output and the associated operations performed at each step. Furthermore, by using the computational graph representation, new nodes can be created in the graph to indicate how to compute the derivatives, as in Figure 2.10, without accessing any numerical values, avoiding the necessity of computing the same subexpressions several times and improving computational performance by saving time and memory.

2.2.5.1 Parameters Initialization

When training the DL models, before feeding the NN with data and starting the forward pass and the gradient descent algorithm with backpropagation, the NN parameters must be initialized. Given the complexity of the functions involved, such as the cost function and its gradient, thend a large amount of computation required, the task of training a NN is inherently challenging. Furthermore, as Goodfellow *et al.* [51], the fact that the learning process is iterative gives the initial point a strong impact on the training result. The initial point is set by the initialization of the parameters, and it can determine whether the algorithm converges or not. Furthermore, if the algorithm converges, the initial point can determine how quickly it converges and if it converges to a high- or low-cost function value, which also affects the final generalization capacity of NN after training.

A naive way to get an idea about the effect of initialization can be made with the help of Figure 2.11. As the objective is to iteratively minimize a cost function, if the initial point is set to a, the algorithm quickly converges to point a', however, this point does not give the lowest cost function value, therefore it is not optimal. In the case where the initial point is set to b, the algorithm converges to the point b' in a slower process, but, at the same time, the point b' represents a lower value of the cost function compared to the point a', representing a better final result. Finally, when the algorithm starts at point c, it continues until it converges at some point lower than b', which can take a longer (or shorter) time to reach. Therefore, the initial point dictates the training result and needs to be chosen carefully.

The initialization strategies used today are simple and heuristic, designed to achieve some desired properties when the NN parameters are initialized and prevent unwanted ones [51]. This simplicity comes from the fact that the optimization of NN is yet primitive. Normally, the parameters are randomly selected from a Gaussian or uniform distribution and scaled by a sufficiently small factor, since initializing everything as zero would make the results of the iterations all null and the learning process would be compromised. Random selection is performed to 'break the symmetry' between different units, as two units with the same initial parameters would be updated in the same way given the deterministic nature of the cost model, which in turn would compromise learning. There are also initialization techniques that leverage pretraining of NN and improve generalization by learning quality features from data [62].

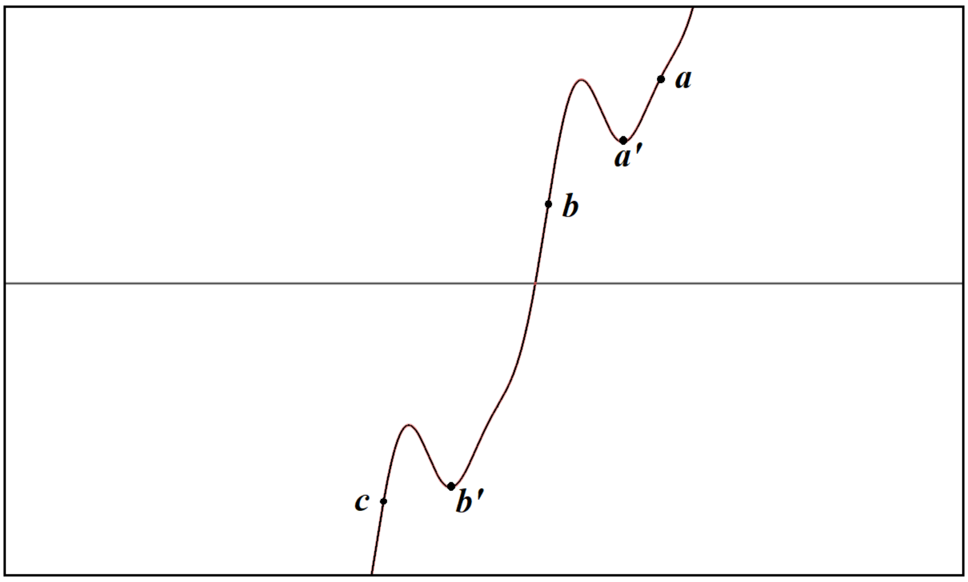

Figure 2.11. Illustrative function to visually show the impact of the initial parameters on the training results.

Here, a general overview of parameter initialization is given, emphasizing its importance in the NN training process. Other works, such as Narkhede *et al.* [63], perform a broad survey on the importance of weight initialization and the various techniques used in this context.

2.2.6 Underfitting and Overfitting

When talking about DL, there are some metrics and terms of great importance. In this section, we are going to talk about **underfitting** and **overfitting**. These two circumstances are related to the training and generalization error of NN, more specifically, according to the gap between these two regimes [51].

The training error of NN is an error measure calculated in the training set. The generalization error, also called the test error, is a measure of how well the model behaves according to previously unseen data [51]. It is very important to use the same error measure in both training and the test set. This metric can vary depending on the chosen training strategy, and a common one is Mean Squared Error (MSE), given by equation 2.3. Other distance metrics can be used, mostly in the case of supervised learning, where we know the target value of the input data that are fed to the network. The idea is to verify how distant the value predicted by NN is from the real target value. With these metrics, two challenges arise:

- Keep the training error as low as possible;

- Keep the gap between the training and generalization error as small as possible.

The underfitting regime starts when the first item is not met; it is the regime in which the model is not able to correctly represent the training data. However, if we achieve the first condition but fail to achieve the second, an overfitting regime occurs. This means that NN behaves well in the training set, but cannot generalize.

These situations are related to the capacity of the model, that is, the set of functions that the learning algorithm can choose to represent the data [51]. It can be tempting to create a highly complex model to achieve the largest possible capacity, but, as shown in Figure 2.12, taken from [51], the larger the capacity of a model, it tends to overfit. On the contrary, the lower the capacity, the more it tends to overfit.

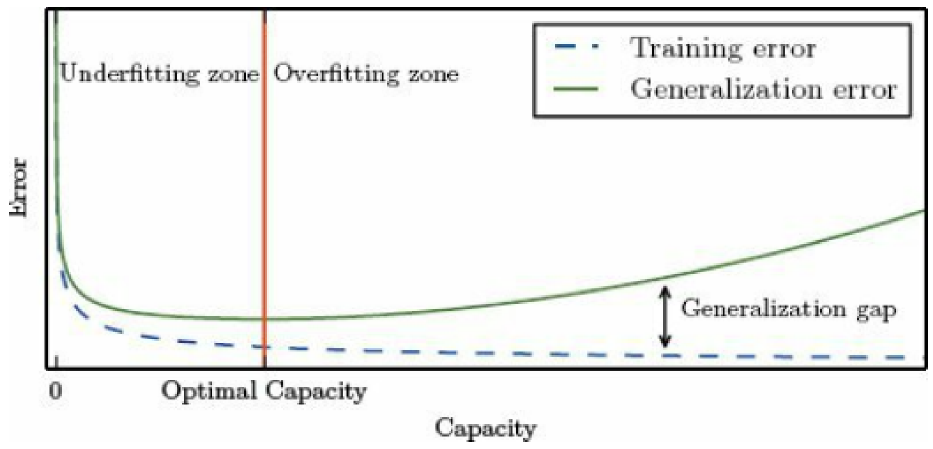

Figure 2.12. Underfitting and overfitting zones. Image taken from [51].

The reality of solving deep learning problems is that the real complexity of the problem will never be known, *by default*. The data generation process will often be much more complex than we can ever represent [51]. Therefore, in general, the goal of a NN architecture is to find the optimal capacity that best represents the data.

It was stated above that it is not recommended to just pick a model with a very large capacity to try and fit the data. However, it is considered good practice to start with a model that has a large capacity and then use different techniques to reduce the family of functions that the model can choose from. In such a manner, it is more feasible to tune the model to achieve the objective.

The process by which we take the model from overfitting to the scenario in which it seems closest to the data-generating process is called **regularization**. There are several ways of regularizing NN, and there are also ways to express the beliefs we have about what our data mean. By increasing the penalty on certain weights, we can, in a certain way, tell the model to prefer one kind of function over another.

2.3 SUPPORT VECTOR REGRESSION

Epsilon-Support Vector Regression (SVR) is a technique belonging to one of the most important supervised learning methods, Support Vector Machines (SVM) [64]. Although not classified as a Deep Learning method, in this work, we applied SVR, a classic machine learning method, as a regression model in the output of our architecture, which will be discussed in Section 3.4.

The idea of Epsilon-Support Vector Regression (SVR) is to fit a model while keeping the target data within a certain distance ϵ from the predicted function. This predicted model will take the form $f(\mathbf{x}) = k(\mathbf{w}, \mathbf{x})$. The weight vector $\mathbf{w}$ belongs to the space of the feature vectors $\mathbf{x}$. The function $k(\mathbf{w}, \mathbf{x})$ is *kernel function*.

This boundary for the maximum error allowed is the *margin* of our model. The goal of SVR is to minimize $\frac{1}{2}||\mathbf{w}||^2$, *i.e.*, the norm of $\mathbf{w}$. As implied above, there are some constraints to this; there is a trade-off between this norm and the tolerance to deviations of the model. Although ideally we would not like to admit deviations larger than ϵ, in practice this is almost impossible without setting a value too large for our purpose. To counteract this, there are also what are called *slack parameters* ξ, the deviation from the margin ϵ. These deviations are also minimized to keep the error at a minimum. Finally, this results in the model shown in

$$\min(\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i) \tag{2.13}$$

and

$$\begin{aligned} |y_i - f(x_i)| &\leq \epsilon + |\xi_i| \\ |y_i - k(w_i, x_i) - b| &\leq \epsilon + |\xi_i|. \end{aligned} \tag{2.14}$$

The hyperparameter C illustrates the tolerance to points outside of the margin. The larger C, the larger the tolerance. The value n is the size of our training set. Figure 2.13 illustrates an example for SVR.

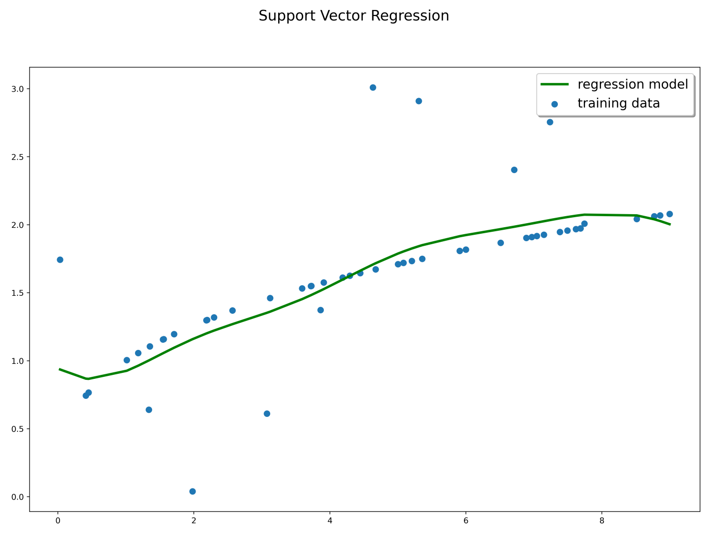

Figure 2.13. An example of SVR.

CHAPTER 3

METHODOLOGY

The compression task, detailed in Section 2.1.1, involves many aspects that are intrinsic to Point Clouds (PCs), such as irregularity and sparsity. To help compress this type of media, Point Cloud Quality Assessment PCQA is the field that tries to assess the content to evaluate the performance of different codecs, providing a way to compare them. To be able to predict a quality score from the data, one needs to design a set of rules that takes the data as input and gives the score as output. However, as exposed in Section 2.2, the power of supervised data-driven approaches comes from the fact that they take data as input and, knowing the expected outputs, try to find the set of rules that would guide the input to the correct outputs. Deep Learning (DL) techniques are the most powerful nowadays for computer vision, and there are already works that join the fields of PCQA and DL such as Lui *et al.* [27]. Our work also focuses on leveraging the power of DL in the PCQA field while designing an architecture that can provide a quality metric that correlates well with the ground truth, the human perceived quality.

3.1 PROJECTION-BASED APPROACH

PCQA techniques also face irregularity and sparsity problems, since one needs to find a mathematical relationship in the content, in our case the PCs, to be able to predict a quality score. Furthermore, the use of DL techniques faces its challenges, considering that it is not easy to find the intrinsic set of rules of a dataset, and usually a large dataset is needed to obtain a good result. However, the PCs datasets are very small, which compromises learning the Neural Network (NN) parameters from scratch and, in turn, places the final result in a spot where it cannot generalize well what the model has learned.

We choose to follow a projection-based approach to address the problems cited above.

Projections are a way of representing PCs as common 2D images. By doing so, we can overcome irregularity and sparsity, since the projections are in the form of 2D structured images. Furthermore, we can maximize the power of existing Convolutional Neural Networks (CNN) architectures trained with images, which are much more advanced than those trained with PCs content. In other words, we can overcome the initialization problem using a NN that already has a set of parameters optimized for a given task. Normally, these NNs are good at evaluating previously unseen data.

The process of generating projections associated with a PC can be done by drawing a bounding cube around the PC and projecting each side of the PC onto the six faces of the cube. This results in six projected images, as shown in Figure 3.1. After generating the projections, one can perform any image process method since they are nothing more than regular images. Therefore, we can use projections to assess the quality of PCs by applying Image Quality Assessment (IQA) techniques, eliminating some of the problems intrinsic to the geometry of PCs.

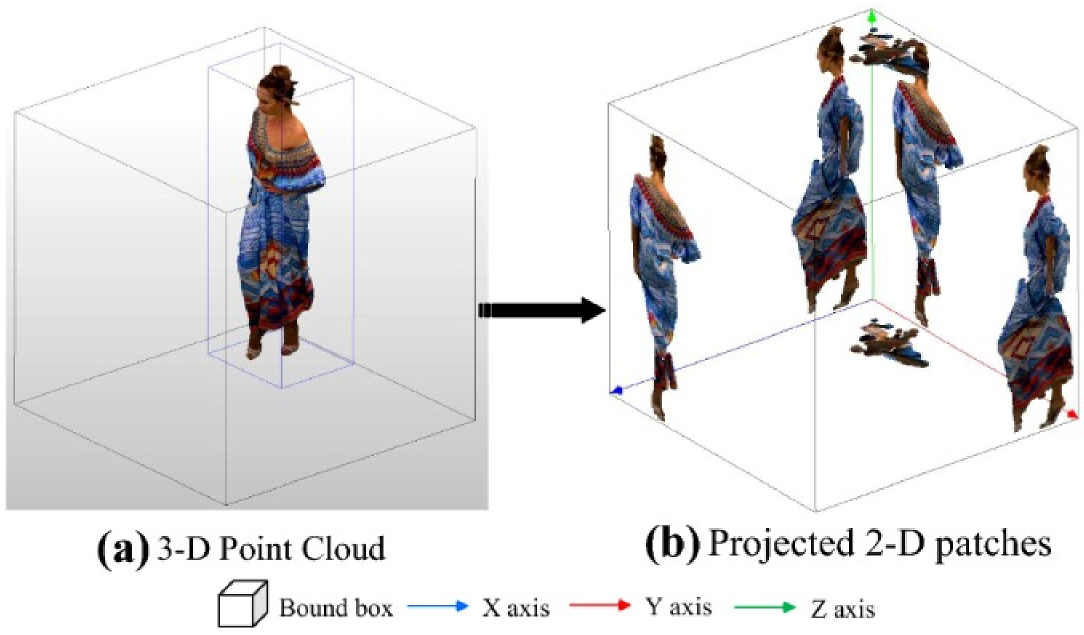

Figure 3.1. To generate the PC projection, first we draw a bounding cube around the PC, as in (a). After, we orthographically project the views on the six faces of the cube, as in (b). Courtesy of [65].

However, a trade-off arises. The cost of being able to treat the PCQA problem as an IQA problem is a direct consequence of treating PCs as images. As Javaheri *et al.* [17] stated, IQA techniques do not efficiently handle local displacement errors, as pixel-level comparisons are the way to go in these techniques. Lossy PC coding can cause geometry degradations and, in turn, cause displacements at the pixel level, resulting in lower correlation performance. In summary, treating a 3D structure as 2D leads to a faulty capture of the geometry perception of the user and may result in a misjudgment of the degree of tolerance the user has to geometry-degraded regions. Additionally, the resulting projections depend on the number of PC points, as a different number of points results in different projections. When comparing the reference and the degraded PCs projections, the compression technique can create a difference in the number of points. This causes one projection to have a pixel occupied while the other has the same pixel not occupied, leading to large pixel mismatches [17].

Some techniques have already used projection-based approaches for PCQA. The first, proposed by de Queiroz and Chou [66], uses a bounding cube around PC to generate the six projections corresponding to the faces of the cube, concatenates them, and measures 2D PSNR between references and degraded projections. More recently, Video-based Point Cloud Compression (V-PCC), proposed by MPEG, is a standard codec focused on the problem of compressing Dynamic Point Clouds (DPCs). It uses already existing video coding techniques, such as MPEG-4 and High Efficiency Video Coding (HEVC) by projecting points into 2D frames and is still widely used [49], proving the relevance of this approach. To generate projections of each PC, we used the technique developed by Javaheri *et al.* [17], in which they orthographically project the PC onto the six faces of the bounding cube. We decided to follow this approach, since it aims to mitigate problems that arise when representing PCs as images, helping to improve the overall performance of the architecture.

First, after generating the raw projections, a cropping operation is performed, since projecting PC may generate a large background area compared to the image portion representing the occupied points. This area comes from the lack of occupied points in the PC, which makes the background, which is also composed of non-occupied pixels, act as a distractor for the chosen IQA technique. In the cases that the background is composed of the same color, the distracting effect is worsened, since pixel-level comparisons are misleading due to the non-degraded portions of the background in contrast with the occupied and degraded ones, resulting in a poor metric correlation with subjective assessment. In this sense, occupancy maps are used to crop the projections and reduce the background area to mitigate its effects, as shown in Figure 3.2.

After cropping, we still have areas in the projected images that are not occupied, either

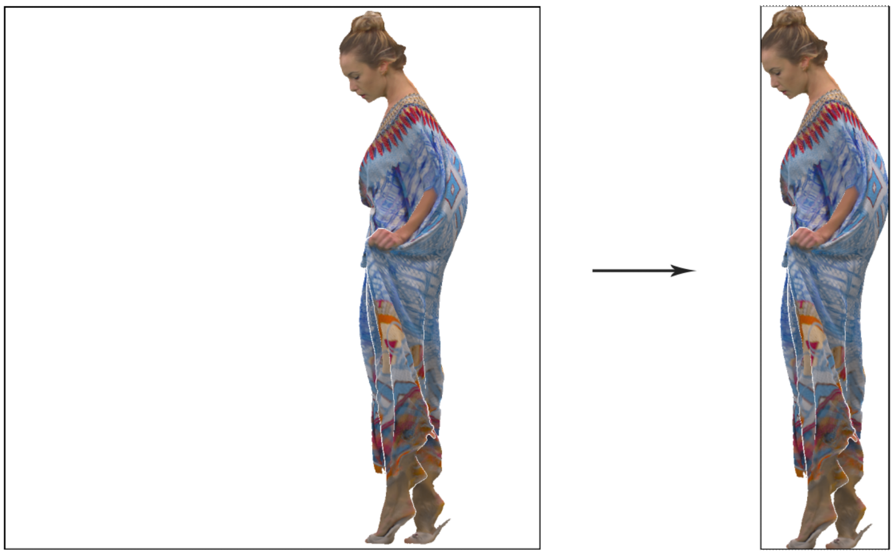
Figure 3.2. Result of the cropping operation in a raw projection.

in the remaining background or within the occupied area. These areas continue to represent a distractor, even if mitigated, to the IQA technique. Therefore, Javaheri *et al.* [17] applied a padding operation that fills the non-occupied portions of the projections using a technique derived from fluid dynamics called Navier-Strokes [67]. The result can be seen in Figure 3.3. Together, cropping and padding operations result in a powerful distractor mitigation effect, improving the performance of the IQA technique.

3.2 PERCEPTUAL-DRIVEN FEATURE EXTRACTION

The next step after generating the projections is to compute a metric score for the images that correlates well with the quality of the content, as perceived by the user. For that, there are some techniques in the field of IQA that are well known, such as MSE [48], that have a low correlation with user perception, and Structural Similarity (SSIM) [68] that provide a better correlation with the final user quality perception. However, using these methods can become problematic. As pointed out by Ding *et al.* [18], they rely on both the reference and the degraded images to be perfectly aligned, as the comparison is made point-by-point between pixels. This can cause two samples of the same texture that appear the same to the user to differ in the arrangement of their features since texture images are especially homogeneous and consist of repeated elements, often subject to some randomization in their location, size, color,

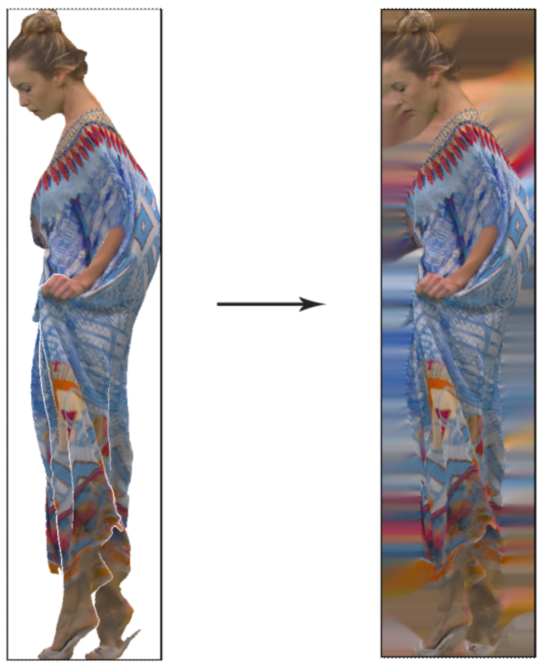

Figure 3.3. Result of the padding operation in a cropped projection.

orientation, etc., as defined by Portilla and Simoncelli [69]. This randomization, despite not changing much of the user's perception, causes the point-by-point metrics to fail in capturing texture similarity, and Figure 3.4 shows the impacts of using a pixel-by-pixel approach on the estimated quality perception. Therefore, we chose to follow the approach of Ding *et al.* [18], in which they developed a full reference metric, Deep Image Structure and Texture Similarity (DISTS), which combines both structural and texture perception, increasing the correlation with perceived quality.

The initial step of their method is a transformation $f : \mathbb{R}^n \mapsto \mathbb{R}^r$ whose purpose is to map the reference and distorted images to perceptual representations, by converting the pixel representation to a space that is more perceptually uniform [18]. This transformation is performed using the VGG16 CNN [16] pretrained for object recognition [70] on the ImageNet database [71]. It is worth noting that Zhang *et al.* [72] showed the effectiveness of VGG's pretrained deep features when used as a substrate to quantify perceptual quality. Using pre-trained features from VGG16 CNN is the representation of the second topic on the benefits of using a projection-based approach, as discussed in Section 3.1. Using a pre-trained CNN can be seen as learning something by reading a specialized book instead of trying to learn it from scratch by yourself; which is the benefit of leveraging already existing knowledge by starting from a good checkpoint instead of the 'start line'.

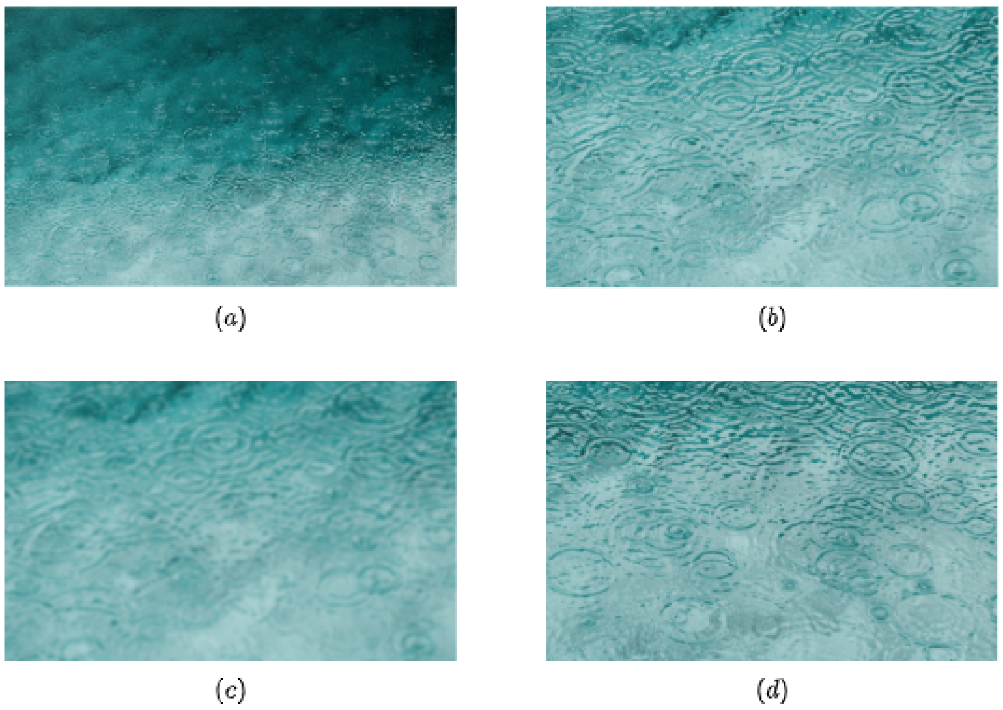

Figure 3.4. As to show how point-by-point comparisons negatively impact the resulting metric, we provide these four images. **(a)** is the original full image. **(b)** is a sample of **(a)**. **(c)** is **(b)** after a Gaussian Blur filter application and **(d)** is a different sample of **(a)**. In a point-by-point approach, like PSNR or SSIM, that compares **(c)** and **(d)** with **(b)**, the resulting quality of **(c)** would be greater than the quality of **(d)**, given the matches in pixel-level. However, for humans, the quality of **(d)** would be superior, given the similar texture and perception.

However, the VGG architecture, as it is by default, is not adequate for the task. Therefore, Ding *el al.* [18] modified its architecture. The first modification aims to make the initial transformation alias-free, providing a better substrate for texture resampling. This was done by changing the Max Pooling layers by L2 Pooling layers, as suggested by Hénaff and Simoncelli [73], who point out that the Max Pooling layers are responsible for introducing aliasing. Therefore, the operation performed in the L2 Pooling layers is described as follows:

$$L_2(x) = \sqrt{g * x^2}, \tag{3.1}$$

where the square and square root operations are pointwise, and the blurring kernel $g(\cdot)$ is chosen as a Hanning window that approximately enforces the Nyquist criterion. The second modification aims to make the transform f injective, where distinct inputs are mapped to different outputs. This ensures that the final measure is a metric in the mathematical sense, as this property has proven itself useful in perceptual optimization. This is implemented by simply including the input image as an additional feature map. The representation then consists of the input image concatenated with the convolution responses of five VGG layers.

So far, Ding *et al.* just adapted the VGG16 pre-trained architecture to generate "perceptual features". The link between features and quality metric is made by designing a mathematical relation that can be optimized to maximize the correlation between the two. This relation is designed under the premise that the visual appearance of textures is often characterized in terms of sets of local statistics [74]. As described by Ding *et al.* [18], after extracting features in both reference and degraded images, they combine two terms of general feature maps, one to compare spatial averages and the other to compare structural characteristics. Therefore, the final score is a weighted sum of these two terms.

Both terms are obtained after first feeding the adapted VGG16 model with the reference and degraded images, x and y, resulting in the reference features and the degraded features, $\tilde{x}$ and $\tilde{y}$, respectively. Then, a measurement for the texture and a measurement for the structure was defined, using the features' global means and correlations, respectively:

$$l(\tilde{x}_j^{(i)}, \tilde{y}_j^{(i)}) = \frac{2\mu_{\tilde{x}_j}^{(i)}\mu_{\tilde{y}_j}^{(i)} + c_1}{\left(\mu_{\tilde{x}_j}^{(i)}\right)^2 + \left(\mu_{\tilde{y}_j}^{(i)}\right)^2 + c_1}, \tag{3.2}$$

and

$$s(\tilde{x}_j^{(i)}, \tilde{y}_j^{(i)}) = \frac{2\sigma_{\tilde{x}_j\tilde{y}_j}^{(i)} + c_2}{\left(\sigma_{\tilde{x}_j}^{(i)}\right)^2 + \left(\sigma_{\tilde{y}_j}^{(i)}\right)^2 + c_2}, \tag{3.3}$$

where $\mu_{\tilde{x}_j}^{(i)}$, $\mu_{\tilde{y}_j}^{(i)}$, $\left(\sigma_{\tilde{x}_j}^{(i)}\right)^2$, $\left(\sigma_{\tilde{y}_j}^{(i)}\right)^2$, $\sigma_{\tilde{x}_j\tilde{y}_j}^{(i)}$ represent the global means, the global variances, and the global covariance of $\tilde{x}_j^{(i)}$ and $\tilde{y}_j^{(i)}$, respectively. $\tilde{x}_j^{(i)}$ denotes the j-th feature map at the i-th convolution layer; c_1 and c_2 are small constants included to avoid instability when the denominator is close to zero. Normalization in both equations serves to equalize the magnitudes of feature maps at different stages. Lastly, the DISTS model combines the quality measurements of different convolution layers in a weighted sum:

$$D(x,y;\alpha,\beta) = 1 - \sum_{i=0}^{m}\sum_{j=1}^{n_i}\left(\alpha_{ij} l(\tilde{x}_j^{(i)}, \tilde{y}_j^{(i)}) + \beta_{ij} s(\tilde{x}_j^{(i)}, \tilde{y}_j^{(i)})\right), \tag{3.4}$$

where α_{ij}, β_{ij} are the weights to be learned in the optimization process, satisfying $\sum_{i=0}^{m}\sum_{j=1}^{n_i}(\alpha_{ij} + \beta_{ij})$ = 1.

The general DISTS workflow takes the reference and degraded images and passes them through the adapted VGG16 model. There are six stages in total, one stage with the original projection that outputs 3 feature maps and the others with 64, 128, 256, and 512 feature

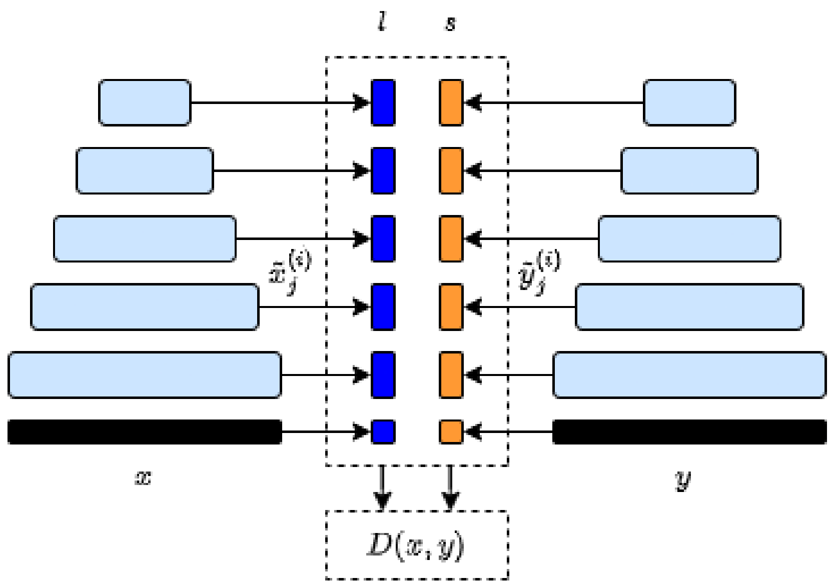

Figure 3.5. Computation diagram for the DISTS model [18] using the pretrained VGG16 CNN layers.

maps, respectively. Together, they form the 1475-parameter texture model that is combined in equation 3.4 and forms the final score DISTS. Figure 3.5 shows the DISTS computation diagram.

The weights $\{\alpha, \beta\}$ in equation 3.4 were optimized to maximize the correlation with human perception of image quality (comparing the output of the model with human ground truth) and invariance to resampled texture patches (minimizing the distance between pairs of patches sampled from the same texture images). As we are dealing with image quality, we minimize the absolute error between model outputs and human ground-truth scores $q(y)$:

$$E_1(x,y;\alpha,\beta) = |D(x,y;\alpha,\beta) - q(y)| \,. \tag{3.5}$$

Then, to optimize for the invariance to the resampled texture patches, they minimized the distance of equation 3.4 between two patches (z_1, z_2) sampled from the same texture image z:

$$E_2(z;\alpha,\beta) = D(z_1,z_2;\alpha,\beta). \tag{3.6}$$

In their work, Ding *et al.* pointed out that they randomly sampled two minibatches Q and T from the KADID-10k [75] and DTD [76] datasets, respectively, and optimized the parameters $\{\alpha, \beta\}$ using a variant of the SGD algorithm and the function:

$$E(Q,T;\alpha,\beta) = \frac{1}{|Q|}\sum_{x,y\in Q} E_1(x,y;\alpha,\beta) + \lambda\frac{1}{|T|}\sum_{z\in T} E_2(z;\alpha,\beta), \tag{3.7}$$

with λ being a constant that controls the trade-off between the terms. Specific details of the training and testing setups used by Ding *et al.* can be found in [18]. Note that Ding *et al.* did not provide how exactly they mapped the output of the model to the actual quality metric, they just provided the final weights $\{\alpha, \beta\}$. More importantly, we are interested in the ability to generate perceptual scores for different images. One can see the DISTS score as a combination of structural and texture comparison between reference and degraded contents, causing the output of the model to give a score that correlates well with human perception.

Taking advantage of this, we use the architecture developed by Ding *et al.* to perform a perceptual-driven feature extraction on the PCs projections, generating, per PC, a vector $v = \{d_0, d_1, \cdots, d_i\}$ with $\{i \in \mathbb{N} \mid 0 \leq i \leq 5\}$, where each value i represents one of the six PC's projections. We chose to use the final outputs of DISTS as features instead of the same features used by Ding *et al.* since the final outputs are already optimized for an IQA task and are well correlated with human perception of digital images. Therefore, we map the vector v to the final PC quality score using a ML algorithm and the ground truth scores, but how exactly we did that will be covered in the next section.

3.3 FEATURE MAPPING

After generating the feature vector v, the next step is to map it to the final quality score. However, the necessity of capturing the vector's intrinsic rules that would perform this mapping increases the complexity of this task significantly. For that, we chose to follow a learning-based approach using a regression model, since it would learn these rules directly from the data and, based on previous assessments of degraded PCs, generate an output that correlates well with human perception. Furthermore, choosing the best regression model to perform the mapping is also a complex task given the large number of possibilities and the specific characteristics of each. We chose to leverage a Python library called 'Lazy Predict', which uses multiple regressors to fit the data and returns a table with the results ranked by performance. This allows us to easily identify the best models for our problem. Details on which dataset was used and how we carried out tests to find the best regressor and its hyperparameters, as well as to prove its generalization capacity after training, will be covered in Chapter 4.

3.4 ARCHITECTURE OVERVIEW

A diagram with an overview of the final proposed architecture is shown in Figure 3.6. As the figure shows, the metric is Full Reference (FR), since it uses both reference and degraded PCs. Also, the diagram is divided into three parts (I, II, and III). The part I is the projection generation process, which is done by leveraging the ProjQM [17] implementation, which orthographically projects the PC views onto the six faces of a bounding cube. After generating the raw projections, cropping and padding are performed to mitigate distraction effects that the background may generate. The part II implements the generation of perceptual features, in which we use the DISTS model [18] to generate a per-view quality score for the PC content. As so, the feature vector v has six quality scores per PC. The process of generating the DISTS score starts with generating 1,475 feature maps extracted from an input image, the projection, after feeding it to the pre-trained VGG16 CNN. From the feature maps, we use equations 3.2 and 3.3 to generate the measurements (coefficients) for texture and structure, respectively. Then, the resulting coefficients are combined into a weighted sum, as described by equation 3.4 where $\{\alpha,\beta\}$, the perceptual weights, are jointly optimized for human perception and texture invariance. For this, we use the already optimized weights from Ding *et al.* [18] that are available on their GitHub page[1]. The result of this weighted sum is the DISTS score d for a single projection. Part III consists of mapping the feature vector v to the final quality score, which is calculated using the SVR model from the Python package Scikit-Learn (sklearn) [77]. We used the Waterloo Point Cloud (WPC) dataset for PCQA [78] to train the regression model and its output is our predicted quality score.

[1]<https://github.com/dingkeyan93/DISTS>

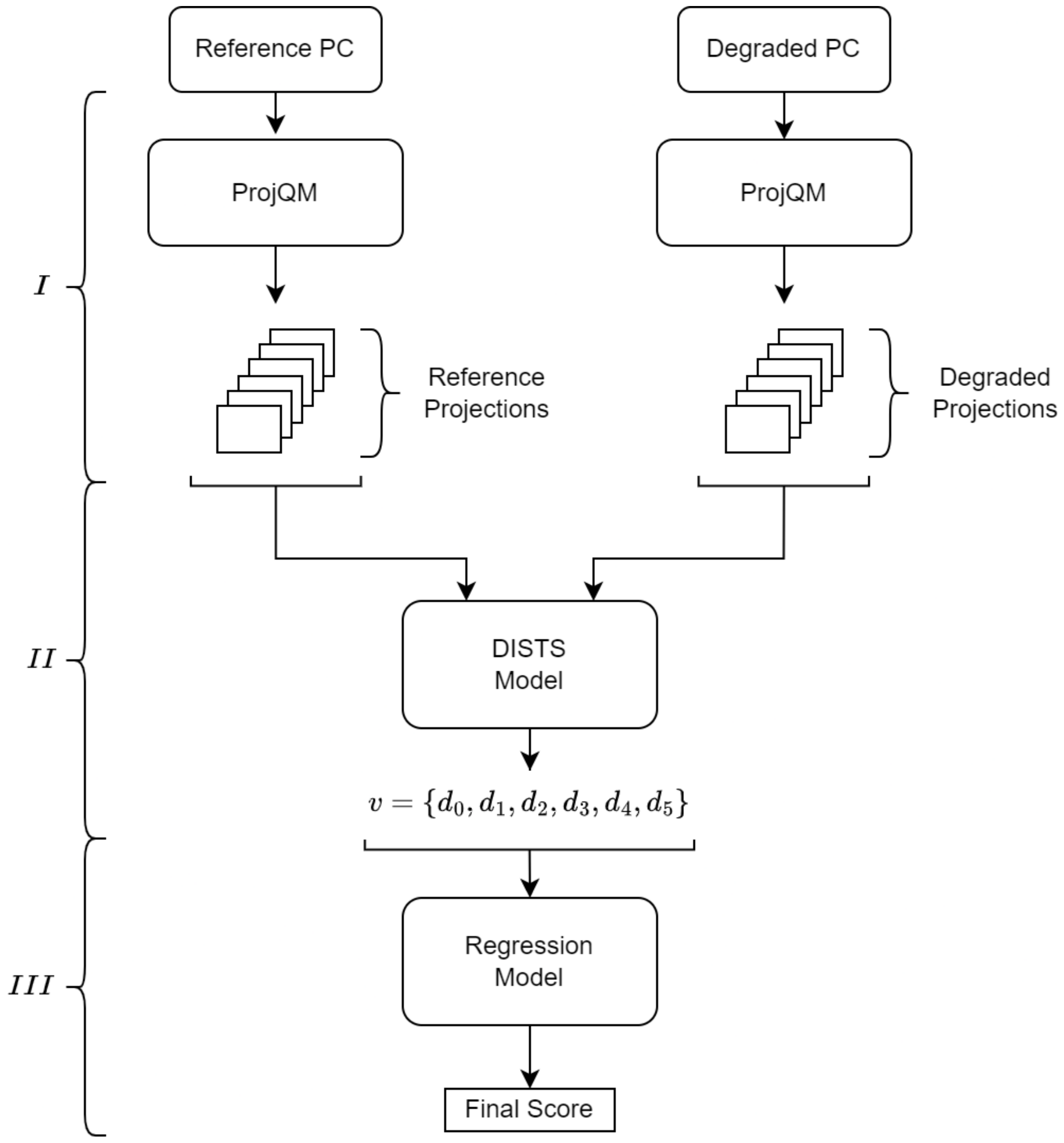

Figure 3.6. Full diagram of the proposed architecture for the FR PC quality metric. The chosen regression model is the SVR from [79] implemented in the sklearn Python package [77].

CHAPTER 4

EXPERIMENTAL SETUP AND RESULTS

In this section, we present the results of our work. First, we describe the datasets in which we trained and tested our models. Then, we explain the approach used in selecting the regression model, as well as the validations used to evaluate the regressors. Lastly, we compare the performance of our model with other PCQA methods on the same datasets.

4.1 DATABASES

Two databases were used for training and validating our model, namely MPEG Point Cloud Compression Dataset (M-PCCD), created by Alexiou *et al* [80], and WPC, created by Liu *et al* [78]. Both datasets were created for purposes of PCQA. The M-PCCD dataset has 8 references, which can be seen in Figure 4.1. Each reference has 29 degraded versions, generated with 5 different techniques, resulting in 232 degraded PCs. For the WPC dataset, it has 20 references that can be seen in Figure 4.2. Each reference has 37 degraded versions, generated with 5 different techniques, resulting in 740 degraded PCs. We will go over more details in the following subsections.

4.1.1 MPEG Point Cloud Compression Dataset

The M-PCCD was created as part of an effort to benchmark the algorithms for geometry and color compression that were standardized by MPEG at the time. That is, Alexiou *et al* [80] used version 6.0-rc1 of G-PCC [81] and version 5.1 of V-PCC [15]. The point clouds in the dataset were extracted from existing databases and the data, which are made up of human figures and objects, are diverse in terms of geometry and color. The goal of the dataset was to evaluate and determine the best practices for rate allocation. Objective and subjective quality assessment methodologies were used to evaluate the scores. An interesting tool that emerged

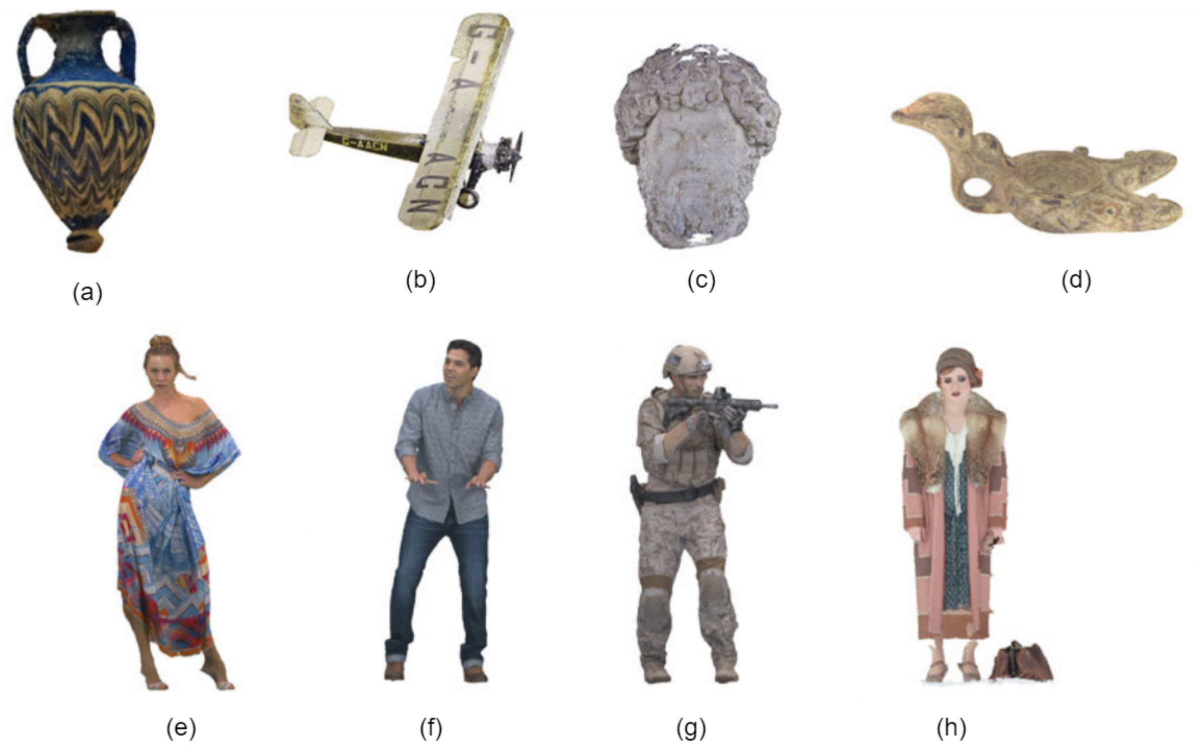

Figure 4.1. Reference PC from the M-PCCD dataset [80]. They are: (a) *amphoriskos*, (b) *biplane*, (c) *head*, (d) *romanoillamp*, (e) *longdress*, (f) *loot*, (g) *soldier* and (h) *the20smaria*. Source: Alexiou *et al.* [80].

from this work [80] was the point cloud web renderer [1]. For more details, refer to the work by Alexiou *et al* [80].

4.1.2 Waterloo Point Cloud Database

The WPC dataset was constructed for PC perceptual quality assessment. It contains 20 point clouds that were generated using a single-lens reflex camera and a turntable in a laboratory. For the techniques used to construct PCs, the reader can refer to the work of Liu *et al* [78]. After generating the references, five types of degradation were applied to each point cloud, each degradation having varying parameters. The final result was, as mentioned above, 740 degraded PCs, with a total of 760 PCs in the dataset. All of these point clouds were normalized to a unit cube of step 0.001.

The degradation types used were *downsampling*, *Gaussian noise contamination*, and three types of compression algorithms [78]: *G-PCC Trisoup*, *G-PCC Octree*, and *V-PCC*. The details are covered in Liu *et al.*[78]. Furthermore, Liu *et al* proposed their model for PCQA, given the

[1] <https://github.com/mmspg/point-cloud-web-renderer>

need for more quality assessment methodologies. This metric was $IW - SSIM_p$ [78], based on projections and weighting of information content, adapting an already existing IQA metric, $IW - SSIM$ [82].

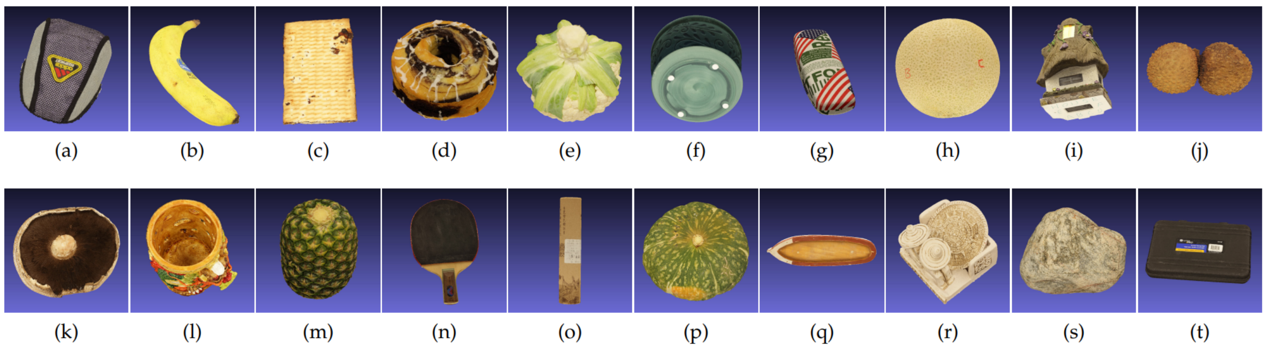

Figure 4.2. Reference PCs from the WPC dataset [78]. They are: (a) *bag*, (b) *banana*, (c) *biscuits*, (d) *cake*, (e) *cauliflower*, (f) *flowerpot*, (g) *glasses_case*, (h) *honeydew_melon*, (i) *house*, (j) *litchi*, (k) *mushroom*, (l) *pen_container*, (m) *pineapple*, (n) *ping-pong_bat*, (o) *puer_tea*, (p) *pumpkin*, (q) *ship*, (r) *statue*, (s) *stone* and (t) *tool_box*. Source: Liu *et al.* [78].

4.2 REGRESSION MODEL SELECTION

To select the best regression model for our architecture, we used a Python package called **lazypredict**. This package allows one to easily fit different regression models in the provided data with its default configurations and return a table with the models ranked by performance. In this manner, using the WPC dataset, we generated projections for every PC and passed them through the DISTS model to obtain the feature vector v for each degraded projection. Our data are then the combination of the 740 feature vectors (since we use the WPC dataset and generate one feature vector for each degraded PC) and the respective labels (subjective scores). Also, we added two new metrics in the LazyPredict class, Pearson's Correlation Coefficient (PCC) and Spearman's Rank Correlation Coefficient (SROCC), since these are metrics that show the correlation of the predictions with the actual quality scores, and our work aims to maximize these values. The PCC coefficient is defined by the following equation, which measures the linear relationship between two sets of data:

$$\text{PCC}(m_i,p_i) = \frac{\sum_i (m_i - m_a)(p_i - p_a)}{\sqrt{\sum_i (m_i - m_a)^2}\sqrt{\sum_i (p_i - p_a)^2}}, \tag{4.1}$$

where m_i is the subjective MOS score, p_i is the predicted score, and m_a and p_a are their averages. For the SROCC, it is a nonparametric measure of the monotonicity of the relationship between two sets of data, and it is defined as:

$$\mathrm{SROCC}(m_i,p_i) = 1 - \frac{6\sum_{i=1}^{L}(m_i - r_i)^2}{L(L^2-1)}, \tag{4.2}$$

where m_i is the subjective MOS score, p_i is the predicted score, r_i is the rank order of p_i and L is the number of test PCs. Root-Mean-Square Error (RMSE) measures not the correlation, but the difference between the predicted values and the actual values, which is the square root of the average squared errors:

$$\mathrm{RMSE}(m_i,p_i) = \sqrt{\frac{\sum_{i=1}^{L}(m_i - p_i)^2}{L}}, \tag{4.3}$$

where m_i, p_i and L are the same as defined before. Since the amount of data is limited, we fed LazyPredict with 90% of the data for training (666 samples) and 10% for testing (74 samples). Table 4.1 shows the results obtained. Note that the top three models were SVR, Nu Support Vector Regression (NuSVR), and the Poisson Regressor.

We picked these three regressors and performed hyperparameter tuning on the WPC dataset to find their best combination with the same training and test sets used to feed the LazyPredict algorithm. For that, we leveraged the RandomSearchCV module from the sklearn Python package [77] with 800 different combinations of hyperparameters and selected the best ones, being:

- SVR:
  - `kernel="rbf", gamma=1, epsilon=1, degree=2, C=5;`
- NuSVR:
  - `kernel="rbf", gamma=1, nu=0.429, degree=2, C=50;`
- Poisson Regressor:
  - `solver="lbfgs", max_iter=100, fit_intercept=True, alpha=0.01.`

After obtaining that set of parameters, we performed four different validations within the WPC dataset: Leave-One-Out Cross-Validation (LOOCV), Leave-One-Group-Out Cross-Validator (LOGOCV) grouping by references, KFold cross-validation with five folds, and finally a cross-database validation where we trained the regressors in a dataset and tested in a different one. We performed these different validations since we had a limited dataset and we wanted to guarantee a good generalization performance. It is important to note that, for LOOCV, the metric used was only RMSE, since both PCC and SROCC cannot be calculated.

4.2.1 K-Fold Cross-Validation

K-Fold cross-validation is the most straightforward method. We divide our dataset into K folds of equal size, then separate $K - 1$ folds as our training set, and the remaining fold as our test set. We fit the model in the training set, test it on the test set, and then store the results. This process is repeated until all folds have been used for testing. Finally, we take the average of the scores and set it as our final result. Table 4.2 shows the results for a K-Fold with 5 folds on the WPC dataset.

In this case, the NuSVR performed the best, achieving the highest PCC and lowest RMSE while performing slightly worse than the PoissonRegressor for the SROCC. Curiously, this was the only case in which the worst regressor was the SVR.

4.2.2 Leave-One-Out Cross-Validation

LOOCV is a special case of K-Fold cross-validation in which we set the number of folds equal to the size of our dataset. This means that, for a dataset of size N, the regressor will be fitted to the $N - 1$ samples and then evaluated on the remaining one. For this reason, we were unable to calculate the correlation coefficients for this validation, since they both require their inputs to be greater than or equal to two. Since we only compared two values, the test metric was RMSE. We see that, in this case, SVR granted the best result, as in table 4.3.

4.2.3 Leave-One-Group-Out Cross-Validation

LOGOCV is another special case of a K-Fold, but with some considerations: the size of each fold is not necessarily equal, because we split the data between groups according to its structure. In the case of the WPC dataset, the data was grouped by reference, and one reference was left out of the training and used for testing at a time. It is important to reiterate that every reference has 37 degraded versions related to it, so in the end, we had 20 folds with 37 samples each. The results are displayed in table 4.4.

When analyzing the results, we can see that the score for SROCC was the same at the third decimal place for all three regressors, although the individual values were different. Given that the nature of SROCC is to evaluate how well the relationship between variables could be approximated by a monotonic function, the similarity of the resulting values in our case is expected. We are not comparing the exact values of our model output, but the order in which they can be sorted. This shows that, without considering the accuracy of each regressor, the three were able to predict the same ranking of the input PCs.

As for PCC, we obtained the same scores for both NuSVR and SVR. Taking into consideration the results for both SROCC and PCC, we can conclude that both regressors approximated similar functions when fitted on the WPC dataset and consequently both performed better than the PoissonRegressor, even though it was shown to be able to predict the same order as the Support Vector Regressors.

4.2.4 Cross-Database Validation

This validation is arguably the most important one for a learning-based model, as it proves the ability to generalize the model to previously unseen data. The results of the cross-database validation of our model are shown in 4.5. In this case, the SVR was the regressor that had the best performance, since it achieved the highest SROCC and the second highest PCC, while the other two regressors performed the worst on either of the correlation coefficients. For this reason, we chose the SVR as our final regressor.

4.3 PERFORMANCE COMPARISON AMONG METRICS

Finally, we compare the performance of other metrics in the dataset M-PCCD, ranging from traditional Point-to-Point, Point-to-Plane, and Plane-to-Plane metrics, as described in

Section 2.1.2.2, to projection-based ones that directly compare projections with techniques like PSNR and SSIM, to state-of-the-art ones like PCQM from Meynet *et al.* which is an optimally weighted linear combination of geometry-based and color-based features [6], PointSSIM from Alexiou *et al.*, which aims to capture local changes similarly to SSIM [41], and LBP_N from Diniz *et al.* which is based on descriptors that extract geometry-aware texture information of PC contents [13]. The results for the values of PCC, SROCC, and RMSE are shown in Table 4.6.

4.3.1 Comparison on the M-PCCD dataset

Table 4.6 shows our results compared to other approaches and variants on the M-PCCD dataset. Our model performed generally worse than the descriptor-based approach when considering PCC, although it performed better than the other approaches. An interesting observation is that the best performing were the ones that took a localized approach, which is a desired behavior when taking into consideration textures since images and PCs can have multiple textures.

For SROCC, our model performed reasonably better than all other approaches and respective variants. This shows that our model is very good at ordering and comparing inputs, correlating with the human ground truth. This is useful when one wants to compare the performance of two codecs without knowing the exact quality score each one outputs, for example. Lastly, the results show that our model can generalize well what it has learned and is robust to overfitting, presenting, indeed, a promising result. The implementation used to produce the results is publicly available at [2].

[2]<https://github.com/mateusvgg/TCC>

Table 4.1. Ranking of Lazypredict regression models using the WPC dataset [78]. The Time Taken represents the time to train and test the model.

Model	RMSE	PCC	SROCC	Time Taken (ms)
SVR	**12.926**	**0.821**	0.840	11.408
NuSVR	13.212	0.820	**0.841**	10.666
PoissonRegressor	13.184	0.807	0.838	27.118
HuberRegressor	13.300	0.806	0.834	7.593
SGDRegressor	13.432	0.804	0.835	3.457
BayesianRidge	13.326	0.804	0.836	2.917
RidgeCV	13.351	0.804	0.836	3.736
ElasticNetCV	13.360	0.803	0.836	17.261
KernelRidge	49.730	0.803	0.833	372.410
Ridge	13.317	0.803	0.833	2.809
LinearRegression	13.319	0.803	0.829	2.750
TransformedTargetRegressor	13.319	0.803	0.829	2.949
Lars	13.319	0.803	0.829	21.452
LassoCV	13.393	0.803	0.841	18.537
OrthogonalMatchingPursuitCV	13.354	0.803	0.841	4.208
LassoLarsIC	13.382	0.803	0.841	3.457
LarsCV	13.392	0.803	0.841	27.551
LassoLarsCV	13.392	0.803	0.841	5.088
LinearSVR	13.275	0.803	0.836	2.891
Lasso	13.534	0.803	0.841	4.160
LassoLars	13.535	0.803	0.841	2.825
GammaRegressor	13.885	0.801	0.834	4.546
PassiveAggressiveRegressor	13.750	0.798	0.834	3.019
MLPRegressor	13.774	0.795	0.809	259.384
RANSACRegressor	13.919	0.787	0.809	14.427
ElasticNet	14.215	0.786	0.838	**2.674**
OrthogonalMatchingPursuit	13.806	0.786	0.826	2.691
TweedieRegressor	14.453	0.784	0.834	4.777
AdaBoostRegressor	15.227	0.761	0.784	47.259
RandomForestRegressor	14.431	0.759	0.790	279.203
GradientBoostingRegressor	14.685	0.752	0.758	168.331
ExtraTreesRegressor	14.543	0.751	0.766	97.293
HistGradientBoostingRegressor	14.761	0.751	0.785	3884.656
LGBMRegressor	15.322	0.731	0.773	316.816
BaggingRegressor	15.201	0.729	0.753	20.820
XGBRegressor	16.645	0.689	0.718	454.797
KNeighborsRegressor	16.295	0.687	0.701	2.988
DecisionTreeRegressor	18.684	0.626	0.649	4.799
ExtraTreeRegressor	19.816	0.618	0.632	3.154
GaussianProcessRegressor	72.575	0.257	0.193	65.265

Table 4.2. KFold cross validation with $k = 5$ for WPC dataset [78]. The last line is the mean of every column's values.

	NuSVR			SVR			PoissonRegressor		
Fold	PCC	SROCC	RMSE	PCC	SROCC	RMSE	PCC	SROCC	RMSE
0	0.707	0.708	15.74	0.698	0.698	15.91	0.715	0.713	15.54
1	0.757	0.753	14.55	0.751	0.750	14.62	0.749	0.749	14.83
2	0.752	0.744	15.65	0.743	0.732	15.89	0.750	0.747	15.69
3	0.756	0.735	16.72	0.750	0.728	17.05	0.755	0.739	16.69
4	0.754	0.758	13.98	0.751	0.756	14.03	0.745	0.760	14.27
mean	**0.745**	0.740	**15.33**	0.738	0.733	15.50	0.743	**0.741**	15.40

Table 4.3. Leave-one-out cross-validation for the WPC dataset [78]. For conciseness, only the mean is displayed.

Model	RMSE
SVR	**11.38**
NuSVR	11.42
Poisson Regressor	11.48

Table 4.4. Leave one group out cross-validation leaving one reference out at time for WPC dataset [78]. The last line is the mean of the values of every column.

	NuSVR			SVR			PoissonRegressor		
Reference out	PCC	SROCC	RMSE	PCC	SROCC	RMSE	PCC	SROCC	RMSE
biscuits	0.824	0.840	13.68	0.828	0.839	12.99	0.804	0.840	14.33
banana	0.606	0.552	17.69	0.626	0.579	17.35	0.623	0.570	17.16
puer_tea	0.626	0.672	25.36	0.638	0.671	24.72	0.640	0.651	24.51
glasses_case	0.804	0.812	16.02	0.803	0.809	16.46	0.798	0.811	15.30
litchi	0.783	0.752	16.87	0.791	0.761	16.53	0.777	0.751	17.16
pen_container	0.899	0.922	16.80	0.904	0.924	17.42	0.907	0.923	15.07
house	0.822	0.825	13.79	0.814	0.826	14.01	0.812	0.830	14.53
pineapple	0.777	0.790	13.53	0.779	0.789	13.98	0.778	0.788	13.12
tool_box	0.852	0.860	11.13	0.830	0.852	11.89	0.849	0.874	11.36
stone	0.752	0.762	14.26	0.754	0.774	14.10	0.747	0.772	14.30
statue	0.834	0.828	14.01	0.839	0.823	13.90	0.831	0.831	13.97
ping-pong_bat	0.891	0.866	11.46	0.883	0.864	11.65	0.884	0.876	12.25
cauliflower	0.733	0.748	15.33	0.722	0.743	15.65	0.732	0.757	15.30
honeydew_melon	0.728	0.751	16.44	0.728	0.743	16.35	0.718	0.736	16.93
ship	0.692	0.731	17.89	0.690	0.735	18.07	0.669	0.733	18.70
pumpkin	0.682	0.718	16.38	0.679	0.717	16.41	0.677	0.714	16.69
bag	0.853	0.839	14.21	0.843	0.819	14.67	0.857	0.842	13.51
mushroom	0.820	0.797	13.07	0.836	0.812	12.70	0.807	0.785	13.40
flowerpot	0.883	0.896	12.11	0.885	0.897	12.22	0.872	0.899	13.05
cake	0.752	0.754	17.08	0.746	0.751	16.87	0.737	0.736	16.97
mean	**0.781**	**0.786**	**15.36**	**0.781**	**0.786**	15.40	0.776	**0.786**	15.38

Table 4.5. Cross dataset validation - training with WPC dataset and testing in M-PCCD dataset [80].

NuSVR		SVR		PoissonRegressor	
PCC	SROCC	PCC	SROCC	PCC	SROCC
0.843	0.924	**0.850**	**0.931**	0.868	0.918

Table 4.6. Performance comparison of the proposed metric on the M-PCCD dataset [80].

Approach	Variant	PCC	SROCC
Point-to-point	po2point$_{MSE}$	0.800	0.868
	PSNR-po2point$_{MSE}$	0.503	0.524
	po2point$_{Hausdorff}$	0.218	0.461
	PSNR-po2point$_{Hausdorff}$	0.503	0.487
	Color-YCbCr$_{MSE}$	0.560	0.640
	PSNR-Color-YCbCr$_{MSE}$	0.293	0.500
	Color-YCbCr$_{Hausdorff}$	0.616	0.559
	PSNR-Color-YCbCr$_{Hausdorff}$	0.293	0.288
Point-to-plane	po2plane$_{MSE}$	0.768	0.891
	PSNR-po2plane$_{MSE}$	0.503	0.625
	po2plane$_{Hausdorff}$	0.255	0.545
	PSNR-po2plane$_{Hausdorff}$	0.503	0.487
Plane-to-plane	pl2plane$_{MSE}$	0.711	0.645
	pl2plane$_{RMS}$	0.715	0.650
Projection-based	proj$_{PSNR}$	0.611	0.628
	proj$_{SSIM}$	0.636	0.633
	proj$_{MSSIM}$	0.701	0.752
	proj$_{VIFP}$	0.716	0.742
Descriptor-based	Local Binary Pattern (LBP)	0.903	0.906
	LBP$_R$	0.867	0.857
	LBP$_N$	0.903	0.917
	LBP$_U$	0.895	0.913
	Local Color Pattern (LCP)	0.882	0.897
	PCQM	0.607	0.915
	PointSSIM-Color	**0.910**	0.918
	PointSSIM-Geometry	0.784	0.834
Proposed	Our Metric	0.850	**0.931**

CHAPTER 5

CONCLUSIONS AND FUTURE WORKS

Point Clouds (PCs) are gaining more and more relevance as technologies regarding 3D media are evolving. The PC data structure is composed of geometry — the points' 3D coordinates — and attributes — the information attached to each point — and, depending on the application, it can have a really large number of points, such as in Miranda *et al.* work [5], where the resulting PCs have more than 1 million points per second.

In Section 2.1, we describe the nature of the PC data structure, some of its intrinsic characteristics, and how it differs from other data structures used to represent 3D media, such as holograms and light fields. In addition, we pointed out how it differs from traditional digital images, such as the lack of a regular grid on which to lay the attributes. Apart from a large number of points, this irregularity sums up the sparsity of the space, resulting in a significant increase in the complexity of compression techniques. To understand how the works in the field address this complexity, we also pointed out some techniques commonly used when working with PCs, such as voxelization and octree-based compression. In that sense, PCQA is the field that, facing the same problems as compression, studies different approaches to generate a good quality metric that correlates well with human perception, making it possible to calibrate and compare different compression techniques.

In our work, focused on the development of a quality metric, we avoid irregularity and sparsity by using a projection-based approach, where we project PC onto the six faces of a bounding cube, generating six projections (views), then crop and pad them to mitigate distraction effects that the background may cause. However, there is a trade-off involved when using projections, once there is a loss in the capacity to capture 3D relationships between points, leading to a reduction in the correlation with human perception. To balance this trade-off, we propose a perceptual-driven feature extraction from the projections, leveraging the power of learning-based techniques applied in the field of IQA, since projections are nothing more than digital images. Using projections also avoids training NN from scratch with the limited available PCs databases, allowing greater capacity of CNN.

Inspired by the use of a learning-based technique, Section 2.2 gives an introduction to the main concepts of DL and the concepts described can be expanded to any DL application. In our scenario, we leverage the already pre-trained VGG16 CNN with slight adjustments in the pooling layers to extract feature maps from the projections and generate the perceptual features with a handcrafted approach developed by Ding *et al.* [18]. After obtaining the features, we feed a regression model to generate the final score. Based on several tests, the chosen regressor was SVR and the final architecture can be seen in the diagram in Figure 3.6.

After training the regressor and evaluating the architecture with different validation techniques, the results show that our metric has competitive performance compared to the state-of-the-art metrics such as PCQM [6], two modes of PointSSIM [83] and LBP_N [13] based on the results present in Table 4.6. Therefore, our projection-based approach, together with perceptual-driven feature extraction, proves itself as a metric that correlates well with human perception of quality.

In addition to the promising results, our architecture shows room for more performance gains, mainly due to the learning-based layer, given that these techniques are getting better every day. One could retrain the $\{\alpha,\beta\}$ parameters from equation 3.4 to obtain weights that are more suitable for PCs projections than for conventional images. However, in the perceptual approach, it is possible to test the A-DISTS metric [84], which is an evolution of DISTS that originally makes rather global quality measurements. In this regard, A-DISTS describes a locally adaptive structure and texture similarity index, taking into consideration that natural photographic images are locally structured and textured across space and scale [84]. Also, one could test different NNs architectures such as ImageNET CNN [85] trained on the ImageNET database [86] to extract even more refined features, it should be noted that changing the architecture of the main NN may call for adaptations in the pipeline. Also, one can use a more robust database or a data augmentation technique to increase the model generalization capacity by increasing the size of the training set.

REFERENCES

1 YUE, X.; WU, B.; SESHIA, S. A.; KEUTZER, K.; SANGIOVANNI-VINCENTELLI, A. L. A lidar point cloud generator: from a virtual world to autonomous driving. In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. [S.l.: s.n.], 2018. p. 458–464. Cited 2 times in pages 1 and 5.

2 BRUDER, G.; STEINICKE, F.; NÜCHTER, A. Poster: Immersive point cloud virtual environments. In: IEEE. *2014 IEEE symposium on 3D user interfaces (3DUI)*. [S.l.], 2014. p. 161–162. Cited 2 times in pages 1 and 5.

3 BESL, P. J. Active optical range imaging sensors. In: *Advances in machine vision*. [S.l.]: Springer, 1989. p. 1–63. Cited 2 times in pages 1 and 7.

4 GOYER, G. G.; WATSON, R. The laser and its application to meteorology. *Bulletin of the American Meteorological Society*, JSTOR, v. 44, n. 9, p. 564–570, 1963. Cited 2 times in pages 1 and 7.

5 MIRANDA, A. S.; DU, G.; GORMAN, C.; DUARTE, F.; FAJARDO, W.; RATTI, C. Favelas 4d: Scalable methods for morphology analysis of informal settlements using terrestrial laser scanning data. *arXiv preprint arXiv:2105.03235*, 2021. Cited 2 times in pages 1 and 55.

6 MEYNET, G.; NEHMÉ, Y.; DIGNE, J.; LAVOUÉ, G. Pcqm: A full-reference quality metric for colored 3d point clouds. In: IEEE. *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. [S.l.], 2020. p. 1–6. Cited 3 times in pages 2, 51, and 56.

7 DINIZ, R.; FREITAS, P. G.; FARIAS, M. C. Towards a point cloud quality assessment model using local binary patterns. In: IEEE. *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. [S.l.], 2020. p. 1–6. Cited in page 2.

8 DINIZ, R.; FREITAS, P. G.; FARIAS, M. C. Local luminance patterns for point cloud quality assessment. In: IEEE. *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. [S.l.], 2020. p. 1–6. Cited in page 2.

9 DINIZ, R.; FREITAS, P. G.; FARIAS, M. C. Multi-distance point cloud quality assessment. In: IEEE. *2020 IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2020. p. 3443–3447. Cited in page 2.

10 DINIZ, R.; FREITAS, P. G.; FARIAS, M. A novel point cloud quality assessment metric based on perceptual color distance patterns. *Electronic Imaging*, Society for Imaging Science and Technology, v. 2021, n. 9, p. 256–1, 2021. Cited in page 2.

11 DINIZ, R.; FREITAS, P. G.; FARIAS, M. C. Color and geometry texture descriptors for point-cloud quality assessment. *IEEE Signal Processing Letters*, IEEE, v. 28, p. 1150–1154, 2021. Cited in page 2.

12 FREITAS, X. G.; DINIZ, R.; FARIAS, M. C. Point cloud quality assessment: unifying projection, geometry, and texture similarity. *The Visual Computer*, Springer, p. 1–8, 2022. Cited in page 2.

13 DINIZ, R.; FREITAS, P. G.; FARIAS, M. C. Point cloud quality assessment based on geometry-aware texture descriptors. *Computers & Graphics*, Elsevier, v. 103, p. 31–44, 2022. Cited 3 times in pages 2, 51, and 56.

14 NINASSI, A.; MEUR, O. L.; CALLET, P. L.; BARBA, D. Considering temporal variations of spatial visual distortions in video quality assessment. *IEEE Journal of Selected Topics in Signal Processing*, IEEE, v. 3, n. 2, p. 253–265, 2009. Cited in page 3.

15 MPEG. V-pcc codec description. ISO/IEC JTC 1/SC 29/WG 7 MPEG Output Document N00100, Jun. 2020. Cited 2 times in pages 3 and 45.

16 SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Cited 3 times in pages 3, 4, and 38.

17 JAVAHERI, A.; BRITES, C.; PEREIRA, F.; ASCENSO, J. Joint geometry and color projection-based point cloud quality metric. *IEEE Access*, IEEE, v. 10, p. 90481–90497, 2022. Cited 5 times in pages 4, 35, 36, 37, and 43.

18 DING, K.; MA, K.; WANG, S.; SIMONCELLI, E. P. Image quality assessment: Unifying structure and texture similarity. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, 2020. Cited 9 times in pages 4, 37, 38, 39, 40, 41, 42, 43, and 56.

19 ZERMAN, E.; GAO, P.; OZCINAR, C.; SMOLIC, A. Subjective and objective quality assessment for volumetric video compression. *Electronic Imaging*, Society for Imaging Science and Technology, v. 2019, n. 10, p. 323–1, 2019. Cited in page 5.

20 GABOR, D. A new microscopic principle. *nature*, v. 161, p. 777–778, 1948. Cited in page 5.

21 WU, G.; MASIA, B.; JARABO, A.; ZHANG, Y.; WANG, L.; DAI, Q.; CHAI, T.; LIU, Y. Light field image processing: An overview. *IEEE Journal of Selected Topics in Signal Processing*, IEEE, v. 11, n. 7, p. 926–954, 2017. Cited in page 5.

22 BERGEN, J. R.; ADELSON, E. H. The plenoptic function and the elements of early vision. *Computational models of visual processing*, MIT press, v. 1, p. 8, 1991. Cited in page 6.

23 DOMAŃSKI, M.; STANKIEWICZ, O.; WEGNER, K.; GRAJEK, T. Immersive visual media—mpeg-i: 360 video, virtual navigation and beyond. In: IEEE. *2017 International conference on systems, signals and image processing (IWSSIP)*. [S.l.], 2017. p. 1–9. Cited in page 6.

24 DINIZ, R. 3d point-cloud quality assessment using color and geometry texture descriptors. 2021. Cited 4 times in pages 6, 13, 15, and 16.

25 BECK, D. *Six Degrees of Freedom: Why More is Better in Virtual Reality*. 2021. <https://foundry45.com/six-degrees-of-freedom-why-more-is-better-in-virtual-reality/>. Cited in page 6.

26 SUGIMOTO, K.; COHEN, R. A.; TIAN, D.; VETRO, A. Trends in efficient representation of 3d point clouds. In: IEEE. *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. [S.l.], 2017. p. 364–369. Cited in page 7.

27 LIU, Y.; YANG, Q.; XU, Y.; YANG, L. Point cloud quality assessment: Dataset construction and learning-based no-reference approach. *arXiv preprint arXiv:2012.11895*, 2020. Cited 4 times in pages 7, 12, 17, and 34.

28 YANG, Q.; MA, Z.; XU, Y.; LI, Z.; SUN, J. Inferring point cloud quality via graph similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 2020. Cited 2 times in pages 7 and 12.

29 GAO, L.; FAN, T.; WAN, J.; XU, Y.; SUN, J.; MA, Z. Point cloud geometry compression via neural graph sampling. In: IEEE. *2021 IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2021. p. 3373–3377. Cited 3 times in pages 7, 12, and 17.

30 STERZENTSENKO, V.; KARAKOTTAS, A.; PAPACHRISTOU, A.; ZIOULIS, N.; DOUMANOGLOU, A.; ZARPALAS, D.; DARAS, P. A low-cost, flexible and portable volumetric capturing system. In: IEEE. *2018 14th international conference on signal-image technology & internet-based systems (SITIS)*. [S.l.], 2018. p. 200–207. Cited in page 7.

31 GONZALEZ, R. C. *Digital Image Processing*. 4th. ed. [S.l.]: Pearson, 2018. Cited in page 8.

32 SHANNON, C. E. A mathematical theory of communication. *The Bell system technical journal*, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948. Cited in page 9.

33 SAYOOD, K. *Introduction to data compression*. [S.l.]: Morgan Kaufmann, 2017. Cited in page 9.

34 XU, Y.; TONG, X.; STILLA, U. Voxel-based representation of 3d point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*, Elsevier, v. 126, p. 103675, 2021. Cited in page 10.

35 MEAGHER, D. Geometric modeling using octree encoding. *Computer graphics and image processing*, Elsevier, v. 19, n. 2, p. 129–147, 1982. Cited in page 11.

36 DRICOT, A.; ASCENSO, J. Hybrid octree-plane point cloud geometry coding. In: IEEE. *2019 27th European Signal Processing Conference (EUSIPCO)*. [S.l.], 2019. p. 1–5. Cited in page 11.

37 SCHWARZ, S.; PREDA, M.; BARONCINI, V.; BUDAGAVI, M.; CESAR, P.; CHOU, P. A.; COHEN, R. A.; KRIVOKUĆA, M.; LASSERRE, S.; LI, Z. *et al.* Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, IEEE, v. 9, n. 1, p. 133–148, 2018. Cited in page 12.

38 DUMIC, E.; DUARTE, C. R.; CRUZ, L. A. da S. Subjective evaluation and objective measures for point clouds—state of the art. In: IEEE. *2018 First International Colloquium on Smart Grid Metrology (SmaGriMet)*. [S.l.], 2018. p. 1–5. Cited in page 13.

39 JAVAHERI, A.; BRITES, C.; PEREIRA, F.; ASCENSO, J. Point cloud rendering after coding: Impacts on subjective and objective quality. *IEEE Transactions on Multimedia*, IEEE, v. 23, p. 4049–4064, 2020. Cited 3 times in pages 13, 16, and 17.

40 ALEXIOU, E.; EBRAHIMI, T. On the performance of metrics to predict quality in point cloud representations. In: SPIE. *Applications of digital image processing XL*. [S.l.], 2017. v. 10396, p. 282–297. Cited in page 13.

41 ALEXIOU, E.; EBRAHIMI, T.; BERNARDO, M. V.; PEREIRA, M.; PINHEIRO, A.; CRUZ, L. A. D. S.; DUARTE, C.; DMITROVIC, L. G.; DUMIC, E.; MATKOVICS, D. *et al.* Point cloud subjective evaluation methodology based on 2d rendering. In: IEEE. *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. [S.l.], 2018. p. 1–6. Cited 2 times in pages 14 and 51.

42 ALEXIOU, E.; EBRAHIMI, T. Impact of visualisation strategy for subjective quality assessment of point clouds. In: IEEE. *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. [S.l.], 2018. p. 1–6. Cited in page 14.

43 ZHANG, J.; HUANG, W.; ZHU, X.; HWANG, J.-N. A subjective quality evaluation for 3d point cloud models. In: IEEE. *2014 International Conference on Audio, Language and Image Processing*. [S.l.], 2014. p. 827–831. Cited in page 14.

44 FREITAS, P. G.; FARIAS, M. C. *Using Texture Measures for Visual Quality Assessment*. Tese (Doutorado) — PhD thesis, University of Brasília, Brasília, 2017. Cited in page 14.

45 TIAN, D.; OCHIMIZU, H.; FENG, C.; COHEN, R.; VETRO, A. Geometric distortion metrics for point cloud compression. In: IEEE. *2017 IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2017. p. 3460–3464. Cited 2 times in pages 15 and 16.

46 MEKURIA, R.; LI, Z.; TULVAN, C.; CHOU, P. Evaluation criteria for pcc (point cloud compression). 2016. Cited in page 15.

47 ALEXIOU, E.; EBRAHIMI, T. Point cloud quality assessment metric based on angular similarity. In: IEEE. *2018 IEEE International Conference on Multimedia and Expo (ICME)*. [S.l.], 2018. p. 1–6. Cited in page 15.

48 WANG, Z.; BOVIK, A. C. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, IEEE, v. 26, n. 1, p. 98–117, 2009. Cited 2 times in pages 16 and 37.

49 MPEG. Common test conditions for point cloud compression. ISO/IEC JTC1/SC29/WG11 Doc. N18474, 2019. 25. Cited 2 times in pages 17 and 36.

50 BELLO, S. A.; YU, S.; WANG, C.; ADAM, J. M.; LI, J. Deep learning on 3d point clouds. *Remote Sensing*, Multidisciplinary Digital Publishing Institute, v. 12, n. 11, p. 1729, 2020. Cited in page 17.

51 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016. Cited 13 times in pages 17, 18, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, and 31.

52 CHOLLET, F. *Deep learning with Python*. [S.l.]: Simon and Schuster, 2021. Cited in page 17.

53 POGGIO, T.; MHASKAR, H.; ROSASCO, L.; MIRANDA, B.; LIAO, Q. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, Springer, v. 14, n. 5, p. 503–519, 2017. Cited in page 17.

54 MITCHELL, T. M. *Machine learning.* [S.l.]: McGraw-hill New York, 1997. v. 1. Cited in page 20.

55 CAUCHY, A. *et al.* Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, v. 25, n. 1847, p. 536–538, 1847. Cited in page 22.

56 DAUPHIN, Y. N.; PASCANU, R.; GULCEHRE, C.; CHO, K.; GANGULI, S.; BENGIO, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, v. 27, 2014. Cited in page 23.

57 SAXE, A. M.; MCCLELLAND, J. L.; GANGULI, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013. Cited in page 23.

58 SRIVASTAVA, R. K.; GREFF, K.; SCHMIDHUBER, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. Cited in page 24.

59 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. Cited in page 26.

60 COLLOBERT, R.; KAVUKCUOGLU, K.; FARABET, C. Torch7: A matlab-like environment for machine learning. In: *BigLearn, NIPS workshop*. [S.l.: s.n.], 2011. Cited in page 27.

61 ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M. *et al.* Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. Cited in page 27.

62 ERHAN, D.; COURVILLE, A.; BENGIO, Y.; VINCENT, P. Why does unsupervised pre-training help deep learning? In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of the thirteenth international conference on artificial intelligence and statistics.* [S.l.], 2010. p. 201–208. Cited in page 29.

63 NARKHEDE, M. V.; BARTAKKE, P. P.; SUTAONE, M. S. A review on weight initialization strategies for neural networks. *Artificial intelligence review*, Springer, v. 55, n. 1, p. 291–322, 2022. Cited in page 30.

64 SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. *Statistics and computing*, Springer, v. 14, p. 199–222, 2004. Cited in page 32.

65 JIA, W.; LI, L.; LI, Z.; LIU, S. Deep learning geometry compression artifacts removal for video-based point cloud compression. *International Journal of Computer Vision*, Springer, v. 129, n. 11, p. 2947–2964, 2021. Cited in page 35.

66 QUEIROZ, R. L. D.; CHOU, P. A. Motion-compensated compression of dynamic voxelized point clouds. *IEEE Transactions on Image Processing*, IEEE, v. 26, n. 8, p. 3886–3895, 2017. Cited in page 36.

67 BERTALMIO, M.; BERTOZZI, A. L.; SAPIRO, G. Navier-stokes, fluid dynamics, and image and video inpainting. In: IEEE. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* [S.l.], 2001. v. 1, p. I–I. Cited in page 37.

68 WANG, Z.; BOVIK, A. C.; SHEIKH, H. R.; SIMONCELLI, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, IEEE, v. 13, n. 4, p. 600–612, 2004. Cited in page 37.

69 PORTILLA, J.; SIMONCELLI, E. P. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, Springer, v. 40, n. 1, p. 49–70, 2000. Cited in page 38.

70 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017. Cited in page 38.

71 DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition.* [S.l.], 2009. p. 248–255. Cited in page 38.

72 ZHANG, R.; ISOLA, P.; EFROS, A. A.; SHECHTMAN, E.; WANG, O. The unreasonable effectiveness of deep features as a perceptual metric. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* [S.l.: s.n.], 2018. p. 586–595. Cited in page 38.

73 HÉNAFF, O. J.; SIMONCELLI, E. P. Geodesics of learned representations. *arXiv preprint arXiv:1511.06394*, 2015. Cited in page 39.

74 JULESZ, B. Visual pattern discrimination. *IRE transactions on Information Theory*, IEEE, v. 8, n. 2, p. 84–92, 1962. Cited in page 40.

75 LIN, H.; HOSU, V.; SAUPE, D. Kadid-10k: A large-scale artificially distorted iqa database. In: IEEE. *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX).* [S.l.], 2019. p. 1–3. Cited in page 41.

76 CIMPOI, M.; MAJI, S.; KOKKINOS, I.; MOHAMED, S.; VEDALDI, A. Describing textures in the wild. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* [S.l.: s.n.], 2014. p. 3606–3613. Cited in page 41.

77 PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Cited 3 times in pages 43, 44, and 48.

78 LIU, Q.; SU, H.; DUANMU, Z.; LIU, W.; WANG, Z. Perceptual quality assessment of colored 3d point clouds. *IEEE Transactions on Visualization and Computer Graphics*, IEEE, 2022. Cited 6 times in pages 43, 45, 46, 47, 52, and 53.

79 DRUCKER, H.; BURGES, C. J.; KAUFMAN, L.; SMOLA, A.; VAPNIK, V. Support vector regression machines. *Advances in neural information processing systems*, v. 9, 1996. Cited in page 44.

80 ALEXIOU, E.; VIOLA, I.; BORGES, T. M.; FONSECA, T. A.; QUEIROZ, R. L. D.; EBRAHIMI, T. A comprehensive study of the rate-distortion performance in mpeg point cloud compression. *APSIPA Transactions on Signal and Information Processing*, Cambridge University Press, v. 8, 2019. Cited 4 times in pages 45, 46, 53, and 54.

81 MAMMOU, K.; CHOU, P. A.; FLYNN, D.; KRIVOKUĆA, M.; NAKAGAMI, O.; SUGIO, T. G-pcc codec description v2. *ISO/IEC JTC1/SC29/WG11 N*, v. 18189, p. 1–39, 2019. Cited in page 45.

82 WANG, Z.; LI, Q. Information content weighting for perceptual image quality assessment. *IEEE Transactions on Image Processing*, v. 20, n. 5, p. 1185–1198, 2011. Cited in page 47.

83 ALEXIOU, E.; EBRAHIMI, T. Towards a point cloud structural similarity metric. In: IEEE. *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW).* [S.l.], 2020. p. 1–6. Cited in page 56.

84 DING, K.; LIU, Y.; ZOU, X.; WANG, S.; MA, K. Locally adaptive structure and texture similarity for image quality assessment. In: *Proceedings of the 29th ACM International Conference on Multimedia.* [S.l.: s.n.], 2021. p. 2483–2491. Cited in page 56.

85 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017. Cited in page 56.

86 DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition.* [S.l.], 2009. p. 248–255. Cited in page 56.

Appendices

APÊNDICE A

PUBLISHED PAPERS RESULTING FROM THIS DISSERTATION

On the Performance of Temporal Pooling Methods for Quality Assessment of Dynamic Point Clouds

Pedro Garcia Freitas*, Mateus Gonçalves†, Johann Homonnai‡, Rafael Diniz§, and Mylène C.Q. Farias¶
*†Department of Computer Science, ‡Department of Electrical Engineering,
University of Brasília, Brasília, Brazil
Emails: *pedrogarcia@ieee.org, §rafaeldiniz@aluno.unb.br, ‡mylene@ieee.org

***Abstract*—Point Clouds (PCs) are collections of points distributed in the 3D space, containing attributes such as color, normals, transparency, and specularity. Dynamic Point Clouds (DPCs) correspond to sequences of points in the 3D space that vary over time like pixels vary over time in a conventional video. Dynamic PCs are a suitable way to represent volumetric videos that can be used in augmented or virtual reality applications. This representation, however, requires a large number of points to achieve a high quality of experience and needs to be compressed before storage and transmission. Therefore, reliable quality metrics are needed in order to automatically estimate the perceptual quality of dynamic PC contents. Since currently there are several quality assessment metrics for static PC, a possible approach solution consists of using temporal pooling functions to combine the quality scores predicted for each of the frames. In this paper, we study the effects of different temporal pooling strategies on the performance of dynamic PC quality assessment methods. Our experimental tests were performed using a recent publicly-available database, demonstrating the efficiency of the evaluated temporal pooling models. More specifically, the work provides a recipe on how to apply a temporal pooling function to combine frame-based quality predictions generated with texture-based static PC quality assessment methods to estimate the quality of dynamic PCs.**

***Index Terms*—Point Cloud Quality Assessment, Temporal Pooling, Memory Effect, Visual Attention, Temporal Visual Masking**

I. INTRODUCTION

Point Cloud Quality Assessment (PCQA) models have gained a lot of attention in the last years, particularly after the release of the recent PC compression standards [1–3]. Many researchers have proposed solutions to the PCQA problem, especially for static PCs that represent a scene by a set of points, with each point being associated with a spatial position and with attributes that describe the surface properties. For instance, Torlig *et al.* [4] developed a folding-based metric that maps tridimensional (3D) volumes onto bidimensional (2D) images. Their method uses orthographic projections in combination with conventional 2D image quality metrics, but it does not exploit the intrinsic 3D structures of PCs. Alexiou and Ebrahimi [5] developed simple metrics to capture the perceived geometric impairments of distorted PCs. Similarly, Javaheri *et al.* proposed a PCQA method based on the generalized Hausdorff distance, which instead of taking the maximum distance over all the distances considers only the K-lowest distance values [6].

Other authors proposed PCQA more complete models that incorporate other types of information, besides geometry. For example, Viola *et al.* [7] proposed a metric that combines color- and geometry-based metrics in order to provide a global quality score. Their metric takes into account the color statistics by analyzing the color histograms and the correlograms. Meynet *et al.* [8] proposed a metric that also takes into consideration geometry- and color-based features, using logistic regression to combine these features and produce a quality estimate. More recently, Alexiou *et al.* [9] proposed a PCQA based on local features extraction. Similarly, Diniz *et al.* [10, 11] have explored the use of texture descriptors to estimate the quality of PC contents, achieving promising results. More recently, Liu *et al.* [12] proposed a method for PCQA that uses a data-driven approach using a Convolutional Neural Network (CNN). Finally, Yang *et al.* [13] proposed a method that uses graph-based relations among points in the PC to estimate quality.

Although the aforementioned papers contribute to important advances in the state-of-the-art, they were proposed to estimate the quality of static PCs. Up to our knowledge, there are not works that tackle the more challenging problem of assessing the quality of DPCs. Therefore, in this work, we seek to fill this gap by conducting a comprehensive evaluation of the use of temporal pooling methods, generally used for Video Quality Assessment (VQA) purposes, to pool quality scores predicted for the individual frames of DPCs and obtain an overall quality estimation. We evaluated the advantages, stability, and generalizability of these pooling mechanisms using four PCQA metrics originally designed for static PCs and 10 temporal pooling functions. We aim to identify statistically valid pooling mechanisms that can be employed to combine scores produced by state-of-the-art PCQA metrics, further improving their prediction performance.

The rest of this work is organized as follows. Section II recaps important related works in the literature, especially those concerned with videos. Section III summarizes the evaluated pooling functions and algorithms used in the benchmarks in our experiments. Section IV describes the experimental setup used in the experiments, including details of parameters used in the pooling algorithms. Experimental results and analysis are presented in Section V. Finally, Section VI presents the conclusions.

Comparative Evaluation of Temporal Pooling Methods for No-Reference Quality Assessment of Dynamic Point Clouds

Pedro G. Freitas
pedrogarcia@ieee.org
University of Brasília
Brasília, Federal District, Brazil

Giovani D. Lucafo
giovanilucafo@usp.br
University of São Paulo
São Carlos, São Paulo, Brazil

Mateus Gonçalves
University of Brasília
Brasília, Federal District, Brazil

Johann Homonnai
University of Brasília
Brasília, Federal District, Brazil

Rafael Diniz
University of Brasília
Brasília, Federal District, Brazil

Mylène C.Q. Farias
mylene@ieee.org
University of Brasília
Brasília, Federal District, Brazil

ABSTRACT

Point Cloud Quality Assessment (PCQA) has become an important task in immersive multimedia since it is fundamental for improving computer graphics applications and ensuring the best Quality of Experience (QoE) for the end user. In recent years, the field of PCQA has made exemplary progress, with state-of-the-art methods achieving better predictive performance at lower computational complexity. However, most of this progress was made using Full-Reference (FR) metrics. Since, in many cases, the reference point cloud is not available, the design of No-Reference (NR) methods has become increasingly important. In this paper, we investigate the suitability of geometric-aware texture descriptors to blindly assess the quality of colored Dynamic Point Cloud (DPC). The proposed metric first uses a descriptor to extract features of the assessed Point Cloud (PC) frames. Then, the descriptor statistics are used to extract quality-aware features. Finally, a machine learning algorithm is employed to regress the quality-aware features into visual quality scores, and these scores are aggregated using a temporal pooling function. Then we study the effects of different temporal pooling strategies on the performance of DPC quality assessment methods. Our experimental tests were carried out using the latest publicly available database and demonstrated the efficiency of the evaluated temporal pooling models. This work aims to provide a direction on how to apply a temporal pooling function to combine per-frame quality predictions generated with descriptor-based PC quality assessment methods to estimate the quality of dynamic PCs. An implementation of the metric described in this paper can be found in https://gitlab.com/gpds-unb/no-reference-dpcqa-temporal-pooling.

CCS CONCEPTS

• **General and reference → Metrics**; **Performance**; • **Information systems → Multimedia information systems**.

KEYWORDS

Point Cloud, Quality Metric, Quality Assessment, QoE Methods

ACM Reference Format:
Pedro G. Freitas, Giovani D. Lucafo, Mateus Gonçalves, Johann Homonnai, Rafael Diniz, and Mylène C.Q. Farias. 2022. Comparative Evaluation of Temporal Pooling Methods for No-Reference Quality Assessment of Dynamic Point Clouds. In *Proceedings of the 1st Workshop on Photorealistic Image and Environment Synthesis for Multimedia Experiments (PIES-ME '22), October 14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3552482.3556552

1 INTRODUCTION

PCQA models have gained a lot of attention in recent years, particularly after the release of recent PC compression standards [1]. Many researchers have proposed solutions to the PCQA problem, especially for static PCs. For example, Torlig *et al.* [2] developed a folding-based metric that maps tridimensional (3D) volumes to bidimensional (2D) images. Their method uses orthographic projections in combination with conventional 2D image quality metrics but does not take advantage of the intrinsic 3D PCs structures. Alexiou and Ebrahimi [3] developed simple metrics to capture perceived geometric impairments in distorted PCs. Similarly, Javaheri *et al.* proposed a PCQA method based on the generalized Hausdorff distance, which, instead of taking the maximum distance over all distances, considers only the K-lowest distance values [4]. Other authors proposed more complete PCQA models that incorporate other types of information besides geometry, remarkably Viola [5], Meynet [6], Diniz [7, 8], Liu [9], Alexiou [10], and Yang *et al.* [11].

Although these works contribute to the state-of-the-art, they were proposed to estimate the quality of static PCs. To our knowledge, there are no works that tackle the more challenging problem of assessing the quality of DPCs. Therefore, in this work, we seek to fill this gap by performing a comprehensive evaluation of the use of temporal pooling methods to pool the predicted quality scores for the individual frames of DPCs and obtain an overall quality estimate. The rest of this work is organized as follows. Section 2 describes the proposed method. Section 3 describes the experimental setup used in the experiments, including details of the parameters used in the pooling algorithms. Section 4 as well as analyses the experimental results. Finally, Section 5 presents the conclusions.