DISSERTAÇÃO DE MESTRADO

SALIENCY-DRIVEN DYNAMIC POINT CLOUD CODING USING PROJECTIONS ONTO IMAGES

Victor Fabre Figueiredo

Brasília, 25 de junho de 2021

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO

SALIENCY-DRIVEN DYNAMIC POINT CLOUD CODING USING PROJECTIONS ONTO IMAGES

Victor Fabre Figueiredo

Dissertação de Mestrado submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Ricardo Lopes de Queiroz, Ph.D., PPGEE / UnB
Orientador

Prof. Eduardo Peixoto Fernandes da Silva, Ph.D., PPGEE / UnB
Examinador Interno

Prof. Luciano Volcan Agostini, Ph.D., UFPel
Examinador Externo

FICHA CATALOGRÁFICA

FIGUEIREDO, VICTOR FABRE
SALIENCY-DRIVEN DYNAMIC POINT CLOUD CODING USING PROJECTIONS ONTO IMAGES [Distrito Federal] 2021.
xvi, 58 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2021).
Dissertação de Mestrado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. point cloud	2. saliency map
3. region of interest	4. RAHT
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

FIGUEIREDO, V.F. (2021). *SALIENCY-DRIVEN DYNAMIC POINT CLOUD CODING USING PROJECTIONS ONTO IMAGES*. Dissertação de Mestrado, Publicação: PPGENE.DM-XXX/AA, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 58 p.

CESSÃO DE DIREITOS

AUTOR: Victor Fabre Figueiredo
TÍTULO: SALIENCY-DRIVEN DYNAMIC POINT CLOUD CODING USING PROJECTIONS ONTO IMAGES.
GRAU: Mestre em Engenharia Elétrica ANO: 2021

Victor Fabre Figueiredo
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

To my parents,
Alice and Ricardo.

ACKNOWLEDGMENTS

Throughout the writing of this work, I have received a great deal of support and assistance.

I would first like to thank my my parents, my brothers and my family for their unconditional support. Without them, I could not have completed this thesis.

In addition, I would like to thank my advisor, Professor Ricardo de Queiroz, whose expertise was invaluable in formulating the research questions and methodology. His insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to acknowledge my colleagues for their collaboration. Particularly, André, Davi, Filipe, Igor, Sandri, Tomás and Tonin. I would particularly like to thank the University of Brasília (UnB) for providing me with the necessary tools for the development of this work.

Finally, I would like to acknowledge the support of my girlfriend, Victoria, who cheered, celebrated and believed in me every day.

Victor Fabre Figueiredo

Work partially supported by Coordenação de Aperfeiçoamento de Pessoal Superior - Brasil (CAPES) - Finance Code 001

ABSTRACT

Regions of interest (ROI) have been used in traditional image and video coding to improve image quality in certain regions, like faces, at the expense of other areas. Nevertheless, ROI in point cloud compression have not been properly addressed, nor has the creation of saliency maps. Both points are addressed in this work. It is hard to directly identify features such as faces in unconnected point clouds and an alternative method to do so was developed. Orthographic projections in 2D planes which are subject to well established computer vision algorithms are used. Once an image region is identified, their pixels are back-projected onto the corresponding *voxels*. By repeating the projections over many orientations, the information of the many back projections is fused to form a collection of *voxels* believed to contain the ROI or to be the most salient. As an unsupervised method, it was devised an algorithm to search the projection orientations for the best views, which include temporal consistency information which is inherited from one frame to another. Face detection algorithms, such as Viola-Jones, were used to determine the 2D ROI and well established saliency map creation algorithms were also used in the 2D image case. In order to use ROI for compression, it was developed an encoding strategy based on a modified distortion criterion that can be applied to many coders and is naturally applicable to the region-adaptive hierarchical transform (RAHT) based coder, which is being adapted into compression standards. In essence, bits (and quality) are shifted towards the ROI from non-ROI areas, assuming non-ROI parts are visually less important and have lower salience values. Results reveal large overall subjective improvement by greatly improving the ROI at the expense of a small degradation of textured regions of lower salience.

RESUMO

As regiões de interesse (ROI) têm sido utilizadas na codificação tradicional de imagens e vídeos para melhorar a qualidade do quadro em certas regiões, como rostos, em detrimento de outras áreas. No entanto, a ROI na compressão de nuvens de pontos não foi amplamente abordada, assim como a criação de mapas de saliência. Ambos os pontos são abordados neste trabalho. É difícil identificar diretamente atributos como rostos em nuvens de pontos esparsas e foi desenvolvido um método alternativo para o fazer. São utilizadas projeções ortográficas em planos 2D que são submetidas a algoritmos de visão computacional bem conhecidos. Uma vez identificada uma região de interesse, os seus pixels são retroprojetados nos *voxels* correspondentes. Ao repetir as projeções ao longo de muitas vistas, a informação de múltiplas projeções é agregada para formar um conjunto de *voxels* que se acredita conter a ROI ou serem os com maior valor de saliência. Como método não supervisionado, foi concebido um algoritmo para procurar as melhores vistas para projeções,

utilizando informação de consistência temporal que é herdada de um quadro para outro. Foram utilizados algoritmos de detecção facial, tais como Viola-Jones, para determinar a ROI 2D e foram também utilizados algoritmos de criação de mapas de saliências bem estabelecidos para imagens bidimensionais. A fim de utilizar a ROI para compressão, foi desenvolvida uma estratégia de codificação baseada num critério de distorção modificada que pode ser aplicado a muitos codificadores e é naturalmente aplicável ao codificador que utiliza a transformação hierárquica por região adaptável (RAHT). Na sua essência, os bits (e a qualidade) são deslocados para a ROI a partir de áreas não-ROI, assumindo que as partes não-ROI são visualmente menos importantes e têm valores de saliência inferiores. Os resultados revelam uma grande melhoria subjetiva global ao melhorar consideravelmente o ROI à custa de uma pequena degradação das regiões de menor saliência.

CONTENTS

1 INTRODUCTION 1
1.1 MOTIVATION 1
1.2 PROBLEM STATEMENT 2
1.3 OBJECTIVES 2
1.4 DERIVED WORK 2
1.5 MANUSCRIPT PRESENTATION 3

2 LITERATURE REVIEW 4
2.1 POINT CLOUDS 4
2.1.1 Octree 5
2.2 REGION-ADAPTIVE HIERARCHICAL TRANSFORM (RAHT) 6
2.3 REGION OF INTEREST 8
2.4 SALIENCY MAPS 8
2.5 FACE DETECTION ALGORITHMS 10
2.5.1 The Viola-Jones Algorithm 10
2.5.2 The Facial Landmarks Algorithm 15
2.6 RELATED WORK 17

3 PROPOSED METHOD 20
3.1 ROI-WEIGHTED DISTORTION MEASURE AND MEASURE-THEORETIC RAHT 22
3.2 PROJECTION-BASED ALGORITHM 24
3.2.1 Point Cloud Projection 24
3.2.2 Face Detection 26
3.2.3 Saliency Map Creation 29
3.3 TEMPORAL CONSISTENCY 29
3.4 ENCODING POINT CLOUDS WITH REGIONS OF INTEREST 30
3.5 USING SALIENCY MAPS AS SOFT REGIONS OF INTEREST 31

4 EXPERIMENTAL RESULTS 34
4.1 BINARY REGION-OF-INTEREST CODING 34
4.2 SOFT-REGION-OF-INTEREST CODING 43

5 CONCLUSIONS 51
5.1 FUTURE WORK 51

REFERENCES 53

LIST OF FIGURES

2.1 Point Clouds RomanOilLight and Stanford Bunny 4
2.2 Octree scanning of *voxels*..... 6
2.3 RAHT diagram for a $2 \times 2 \times 2$ block..... 7
2.4 Region of Interest highlighted in the image by yellow rectangles. 9
2.5 Example of a saliency map. 10
2.6 Example of feature rectangles relative to the detection window. 12
2.7 The sum of the pixels within rectangle D can be computed with four array references. 12
2.8 Cascaded classifiers diagram..... 15
2.9 Facial landmark results..... 19

3.1 Diagram of the proposed algorithm for region of interest detection in point clouds. 21
3.2 Region of interest detection in point clouds using projections and image identification algorithms..... 25
3.3 Steps for the creation of a saliency map using two-dimensional projection of a point cloud. 26
3.4 Orthogonal projections of a point cloud. 27
3.5 Representation of a point cloud and its elevation angle and azimuth. 28
3.6 Artefacts in the region of interest identification caused by the obliquity of the projections and the Morton-code-based dilation for different cube widths..... 28
3.7 Point cloud "Soldier" (frame 695) and its sub-point clouds. 32

4.1 ROI identified in frames 1, 101 and 201 of each sequence. The image shown on the second line and second column represents a case when the algorithm was not able to detect the ROI. 35
4.2 Average rate-distortion curves for all frames of the sequence Longdress. The PSNR is computed only for (a) voxels in the ROI and (b) voxels outside the ROI. 37
4.3 Average rate-distortion curves. 39
4.4 Average rate-distortion curves for the sequence "Loot" using the weighted PSNR .. 40
4.5 Point cloud Thaidancer (N_{vox} = 689953) coded with different weights for *voxels* in the ROI. The Q_{step} was adjusted to result in similar file sizes..... 41
4.6 Detected region of interest and close up in the face of the reconstructed point clouds shown in Fig. 4.5 41
4.7 Frames encoded with w_{ROI} equals to 1 and 16..... 42
4.8 A view of point cloud "David" and its saliency and hot-cold maps. 44
4.9 A view of point cloud "Boxer" and its saliency and hot-cold maps..... 44
4.10 A view of point cloud "Loot" and its saliency and hot-cold maps..... 45
4.11 A view of point cloud "Longdress" and its saliency and hot-cold maps. 45
4.12 A view of point cloud "Soldier" (frame 537) and its saliency and hot-cold maps. 46

4.13 A view of point cloud "Soldier" (frame 695) and its saliency and hot-cold maps. 46
4.14 Rate-distortion curves for the point cloud "Longdress" for different values of the ROI weights. The distortion is computed using the weighted PSNR in (a) and using the standard PSNR in (b). 48
4.15 PSNR for the *voxels* inside the ROI with $\alpha = 32$ for the "Longdress" point cloud... 49
4.16 Point cloud Longdress coded with different weights for *voxels* in the ROI. The Q_step was adjusted in order that the files would have similar sizes. 49
4.17 Close up in the face of the reconstructed point clouds shown in Fig. 4.16 50

5.1 Comparison between the projection-based method used in this work and the preliminary hybrid method. 52

LIST OF TABLES

4.1 Correlation coefficient between $PSNR_Y$ and the other metrics for all point clouds tested 36
4.2 Average BD-PSNR and BD-rate in the range from 0.08 and 1 bpov with the PSNR computed inside and outside the ROI, compared to the PSNR of all voxels. 38
4.3 Average BD-PSNR (dB) for all the 5 point clouds, for each $S(v_i)$ comparing the curves with $\alpha > 1$ against those when $\alpha = 1$ for all point clouds tested in this work. 47

SYMBOLS

V	The geometry of a point cloud, a list of 3D positions, i.e. occupied voxels.
C	List of colors associated with the occupied voxels from V.
S	Saliency level in a quantized saliency map.

DEFINITIONS

octree	A tree data structure in which each internal node has exactly eight children.
pixel	Picture element.
voxel	Volume element.

ACRONYMS

1D	One-dimension
2D	Two-dimensions
3D	Three-dimensions
4D	Four-dimensions
AR	Augmented reality
bpov	Bits per occupied voxel
CfP	Call for Proposals for Point Cloud Compression
DC	Direct Current, represents coefficients with zero frequency
DCM	Direct Coding Mode
fps	Frames per second
G-PCC	Geometry-based Point Cloud Compression
MPEG	Moving Pictures Expert Group
MR	Mixed reality
MSE	Mean-Squared Error
MVUB	Microsoft Voxelized Upper Body
PCC	Point Cloud Compression
PSNR	Peak Signal-to-Noise Ratio
RAHT	Region-Adaptive Hierarchical Transform
RGB	Red, Blue and Green color space
ROI	Region of Interest
TMC13	Test Model Categories 1 and 3
VR	Virtual Reality
YUV	Luma, Chrominance blue, and Chrominance red color space, following the BT.709 HDTV standard

1 INTRODUCTION

In the last few years, there was a great revolution in the possible ways to represent the world. Advances in computer graphics created technologies like virtual reality (VR), augmented reality (AR) and mixed reality (MR) [1]. These technologies use three-dimensional models to represent scenes and objects. The 3D representations of scenes, or objects, can be obtained using scanning technologies such as optical laser-based, structured-light and LIDAR (Light Detection And Ranging), or passive methods such as multi-view stereo.

Point clouds are a common representation for the three-dimensional world. Nevertheless, point clouds demand a large amount of resources since a typical point cloud may contain millions of points. A major challenge is to efficiently transmit and/or store high-quality point clouds. Moreover, in order to facilitate inter-operation between production and consumption, compression standards for point clouds are being devised by The Moving Picture Experts Group (MPEG) and the Joint Photographic Experts Group (JPEG) [2], [3]. In 2017, MPEG issued a Call for Proposals for Point Cloud Compression (PCC) [4].

One point cloud can be interpreted as the 3D equivalent of an image and a sequence of point clouds can be interpret as the 3D equivalent of a video. Like image and videos, different information of the represented scene may be needed depending on the application, some methods to obtain it are,

- Region of interest extraction;
- Colour segmentation;
- Object identification;
- Face recognition;
- Saliency map acquisition.

These are well known in the two-dimensional case [5]–[8]. Nevertheless, to obtain such information from point clouds is still a challenge and the literature on such methods is still scarce.

1.1 MOTIVATION

Human vision relies on two mechanisms. In the first, basic "objects", like animals, faces, plants, etc., are categorized, and in the second, perception of the objects is enriched through conscious examination, such that the observer's attention is concentrated on regions of interest (ROI) that are relevant to the observer [9]. Like real scenes, point-cloud representations of scenes naturally

have ROI due to their salience. As a ROI attract attention, preserving its quality is important. For images and video, ROI-driven compression, or *ROI coding*, is well studied [10].

Saliency maps in 2D have been studied for many years [8], [11]–[14]. They were developed with the purpose of identifying, in images, regions that receive greater attention in human visualization. Therefore, it is possible to use saliency maps as regions of interest.

However, for point clouds, the literature on saliency maps creation and ROI coding is scarce. Thus, not only creating the ROI is challenging enough, but encoding it is still a problem.

1.2 PROBLEM STATEMENT

Many computer vision algorithms have been developed and are extensively studied in a two-dimensional space, including those to create saliency maps, regions of interest and encode them. However, the literature is scarce in this field for point clouds. Develop solutions that directly act on a sparse three-dimensional space is challenging. It is possible to borrow solutions for the problem in 2D space and adapt it to the 3D space.

1.3 OBJECTIVES

The main objectives of this work are to develop a framework that applies 2D computer vision algorithms in a 3D sparse space and to use the results to propose a method for ROI coding a point cloud.

1.4 DERIVED WORK

The research developed in this work resulted in the publication of four conference papers and one award of best paper of signal processing in the conference. In order of publication the papers are:

1. Point Cloud Compression Incorporating Region of Interest Coding, International Conference on Image Processing, 2019. [15]
2. Compressão de Nuvem de Pontos Incorporando Codificação de Região de Interesse, Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, 2019. (Award winning paper) [16]
3. Saliency Maps for Point Clouds, International Workshop on Multimedia Signal Processing, 2020. [17]

4. Mapa de Saliência para Nuvem de Pontos usando Projeções, Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, 2020. [18]

1.5 MANUSCRIPT PRESENTATION

In Chapter 2, a literature review is presented in order to explain the concepts required to produce this work. Then, in Chapter 3, the methodology used is presented alongside with a detailed explanation of the developed framework. The results obtained are discussed in Chapter 4. Finally, in Chapter 5, general conclusions are given.

2 LITERATURE REVIEW

As well as images and videos, point clouds often have Regions of Interest (ROI) that have special meaning or relevance - for example, faces - for various applications in the areas of data compression, computer vision, medical imaging, autonomous vehicle navigation, among others.

For images and videos, the use of regions of interest is already widespread in several areas [10], [19]. However, for point clouds there are still some areas, such as compression, where the literature available is scarce on the use of regions of interest. For face detection and object classification in point clouds there is a vast literature available with different detection methods [20]–[24].

In this project, we have chosen a new approach for the detection of regions of interest in point clouds. We opted for a 2D project-based calculation of the regions of interest.

2.1 POINT CLOUDS

With the arising of three-dimensional digital content, it was necessary to develop methods to represent it that would allow its direct consumption by humans. One of the methods developed to perform such representation is the point cloud.

A point cloud is a set of points in space represented in a three-dimensional (X, Y, Z) coordinate system. It commonly serves the purpose of representing the outer surface of an object or scene. An example of a 3D captured object in its point cloud format is shown in Figure 2.1.

Figure 2.1: Point Clouds RomanOilLight [25] and Stanford Bunny [26], [27].

Point clouds consist of geometry and all its attributes [28]. The geometric part of a point cloud

with N points, can be described by a set V containing the coordinates of all points, such that:

$$V = \{v_1, v_2, \ldots, v_n\} = \left\{ \begin{array}{c} (x_1, y_1, z_1) \\ (x_2, y_2, z_2) \\ \vdots \\ (x_n, y_n, z_n) \end{array} \right\} \tag{2.1}$$

where $n = 1, \ldots, N$ and $v_i = (x_i, y_i, z_i)$ defines the position of the point p_i.

Attributes can be represented in a similar way by a set C where each entry in that set has D attributes per point:

$$C = \{c_1, c_2, \ldots, c_n\} = \left\{ \begin{array}{c} (a_{11}, \ldots, a_{1D}) \\ (a_{21}, \ldots, a_{2D}) \\ \vdots \\ (a_{n1}, \ldots, a_{nD}) \end{array} \right\} \tag{2.2}$$

Commonly, attributes include color components, but may also include transparency, normal vectors, motion vectors, and so forth. Once the geometry is given, the attributes may be thought of as a signal defined on a set of points.

The point in the point cloud is the primitive notion on which geometry is built. It has no length, area or volume, it is used as a unique location in Euclidean space. Thus, for rendering, points are commonly represented as spheres. Employing *voxelized* point clouds (VPC) is more convenient for representing such data.

A *voxel* is the 3D extension of a pixel, while the pixel is a square and the fundamental element of a 2D image, the *voxel* is a cube and is the fundamental element of a three-dimensional object.

We assume that the object represented is contained in a cube of size $L \times L \times L$, where L is a positive integer. We can divide this cube into the three dimensions L times, obtaining L^3 cubes of dimension $1 \times 1 \times 1$, which are by definition *voxels*. The advantage of such approach is that the position of each *voxel* is discrete rather than continuous and the fundamental element is no longer dimensionless.

Different from the 2D image case, in the 3D representation by *voxels* the vast majority of them must be transparent. Also, those *voxels* in the interior of the objects we are representing may also be transparent because normal techniques can not capture information in those regions and/or because that information is not relevant, reducing the number of data required to represent an image. Those *voxels* that are not transparent are referred as occupied *voxels* and those that are transparent are commonly referred as non occupied *voxels*.

2.1.1 OCTREE

In computer science, a tree is a data structure in which its elements, called nodes, hierarchically related to each other. They are composed of an initial node, called the root, plus its children nodes.

Each subsequent node can have one or more children. When a node has no children, it is called a leaf node [29].

An octree is a tree data structure in which each internal node has exactly eight children [30], it is the 3D extension of a 2D quad-tree. They are most often used to partition a three-dimensional space and have been shown to be very efficient when encoding a point cloud [31].

To explain the octree scanning process, illustrate in Figure 2.2, we start with a cube with dimensions $L \times L \times L$ where ours *voxels* lay. Then we have a list of *voxels* inside our cube, the cube is then divided in half along one of its dimensions (i.e. x axis). At this stage we say that we travelled one level on this octree. The list of occupied *voxels* is divided in two lists: one for the *voxels* laying in the leftmost half and another list for the rightmost half. We continue our division along another dimension (y axis), dividing in four regions and producing 4 *voxel* lists. Finally, repeating the process for the remaining dimension (z axis), we obtain 8 cubes similar to the original one, but with a width of $L/2$. At this stage we say that we have travelled one depth on this octree, where 3 levels are the equivalent of 1 depth.

We can continue with this process, further dividing the space. At depth d we will have cubes of size $L/2^d$. If L is chosen to be a power of 2, at the depth d where $2^d = W$, the resulting cubes will have size $1 \times 1 \times 1$ and will accommodate one single *voxel*. For this reason, for most of voxelized point clouds L is chosen to be a power of 2. The higher the value of L, the more detailed the point cloud can be as it comports more *voxels*.

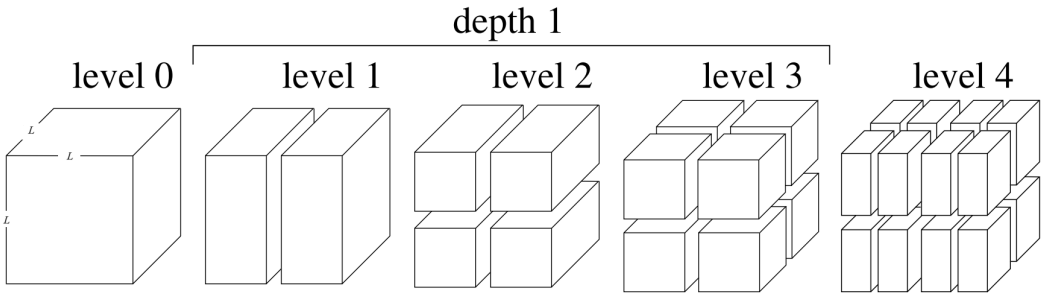


Figure 2.2: Octree scanning of *voxels* [32].

2.2 REGION-ADAPTIVE HIERARCHICAL TRANSFORM (RAHT)

Most point cloud codecs found in the literature first encode the geometry, and then encode the attributes conditioned on the geometry. Typical approaches to attribute coding include transform coding using the Graph Fourier Transform (GFT) [33]–[38], the Gaussian Process Transform (GPT, which is the KLT of a Gaussian Process) [39], [40], and the Region-Adaptive Hierarchical Transform (RAHT) [28], [41]. RAHT, unlike the GFT or GPT, does not require an eigen-decomposition, and has been one of the transforms initially adopted into MPEG PCC [2].

The RAHT is a hierarchical orthogonal subband transform. It is a variation of the Haar transform that takes the data sparsity into account, it accomplishes this by using adaptive weights to consider different regions with empty or occupied *voxels*.

In the RAHT the weight is set to be 1 to all occupied *voxels* and 0 for void *voxels* initially, then the *voxels* are combined two by two, along each dimension. Let us denote $F^{\ell}_{i,j,k}$ the color of the *voxel* at the position $x = i, y = j, z = k$ at level l and by $w^{l}_{i,j,k}$ its weight. Along the x-dimension, the RAHT is given by:

$$\begin{bmatrix} F^{\ell}_{i,j,k} \\ G^{\ell}_{i,j,k} \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} F_{2i,j,k} \\ F^{\ell+1}_{2i+1,j,k} \end{bmatrix}, \tag{2.3}$$

where

$$a^2 = \frac{w^{\ell+1}_{2i,j,k}}{w^{\ell+1}_{2i,j,k} + w^{\ell+1}_{2i+1,j,k}}, \quad b^2 = \frac{w^{\ell+1}_{2i+1,j,k}}{w^{\ell+1}_{2i,j,k} + w^{\ell+1}_{2i+1,j,k}}. \quad a, b \geq 0. \tag{2.4}$$

and at least one of the *voxels* in the pair is occupied. The inverse transform is given by:

$$\begin{bmatrix} F^{\ell+1}_{2i,j,k} \\ F^{\ell+1}_{2i+1,j,k} \end{bmatrix} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} F^{\ell}_{i,j,k} \\ G^{\ell}_{i,j,k} \end{bmatrix}. \tag{2.5}$$

In RAHT the high-pass coefficient is not subject to further processing along the other dimension. They are sent directly to the next compression steps. The low-pass coefficients, on the other hand, are treated as new *voxels*. The process is repeated along each dimension (x, y and z) until reaching the level where only one low-pass coefficient remains, the DC coefficient. The DC coefficient is sent to the decoder along with all the high-pass coefficients generated in the process. In figure 2.3 is shown a diagram for the RAHT implementation for a $2 \times 2 \times 2$ block.

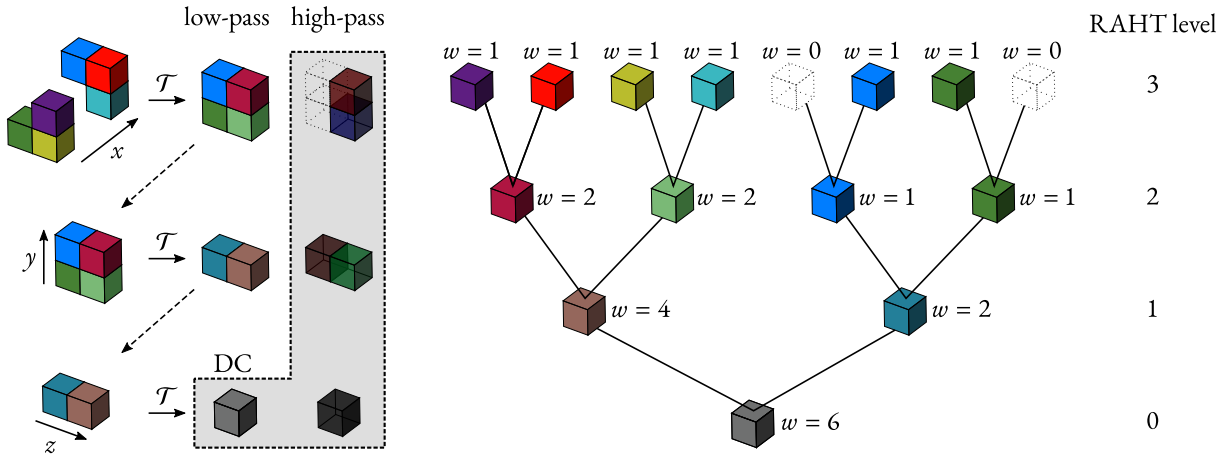


Figure 2.3: RAHT diagram for a $2 \times 2 \times 2$ block. [42]

2.3 REGION OF INTEREST

A Region of Interest (ROI) is defined as samples of a data set identified for a particular purpose [19]. The concept of a ROI is commonly used in many application areas, for example, computer vision, medical imaging, optical character recognition (OCR), geographical information systems and others. In computer vision the ROI usually defines the limits of the area under consideration of the image or video.

The region of interest can be specified in a one-dimensional, two-dimensional (see Fig.2.4 for an example), three-dimensional or four-dimensional dataset. Examples of regions of interest of each of these types are given below.

- 1D data set: a frequency interval of a waveform;
- 2D data set: the contour of an object in an image;
- 3D data set: the contour of a surface of an object in a volume;
- 4D data set: the contour of an object during a time interval in a volume of time.

A ROI is an extra information expressed in a structured form that needs to be encoded. It can usually be encoded as an integral part of the data set as a masking value which tags individual data cells; as a graphic information such as drawing elements; as a set of spatial and/or temporal coordinates.

Regions of interest can be generated manually, semi-automatically, or automatically via software. To distinguish:

- Manual ROI: the user manually defines an area using a keyboard, mouse or other accessory that enables the delimitation;
- Semi-automatic ROI: the software assists the user in drawing or delimits the region based on parameters provided by the user;
- Automatic ROI: the software determines the boundary criteria without user intervention.

Regions of interest are widely used in image compression, such as in the standard *JPEG2000* [43], and in videos [10]. With the detection of regions of interest in point clouds it is possible to use it to preserve the quality of some regions in the compression process.

2.4 SALIENCY MAPS

Saliency map creation can be considered one of the ways of segmenting images and videos. In computer vision, segmenting an image is the process of dividing a digital image into several segments consisting of groups of pixels. The goal of this process is to create a simplified visualization

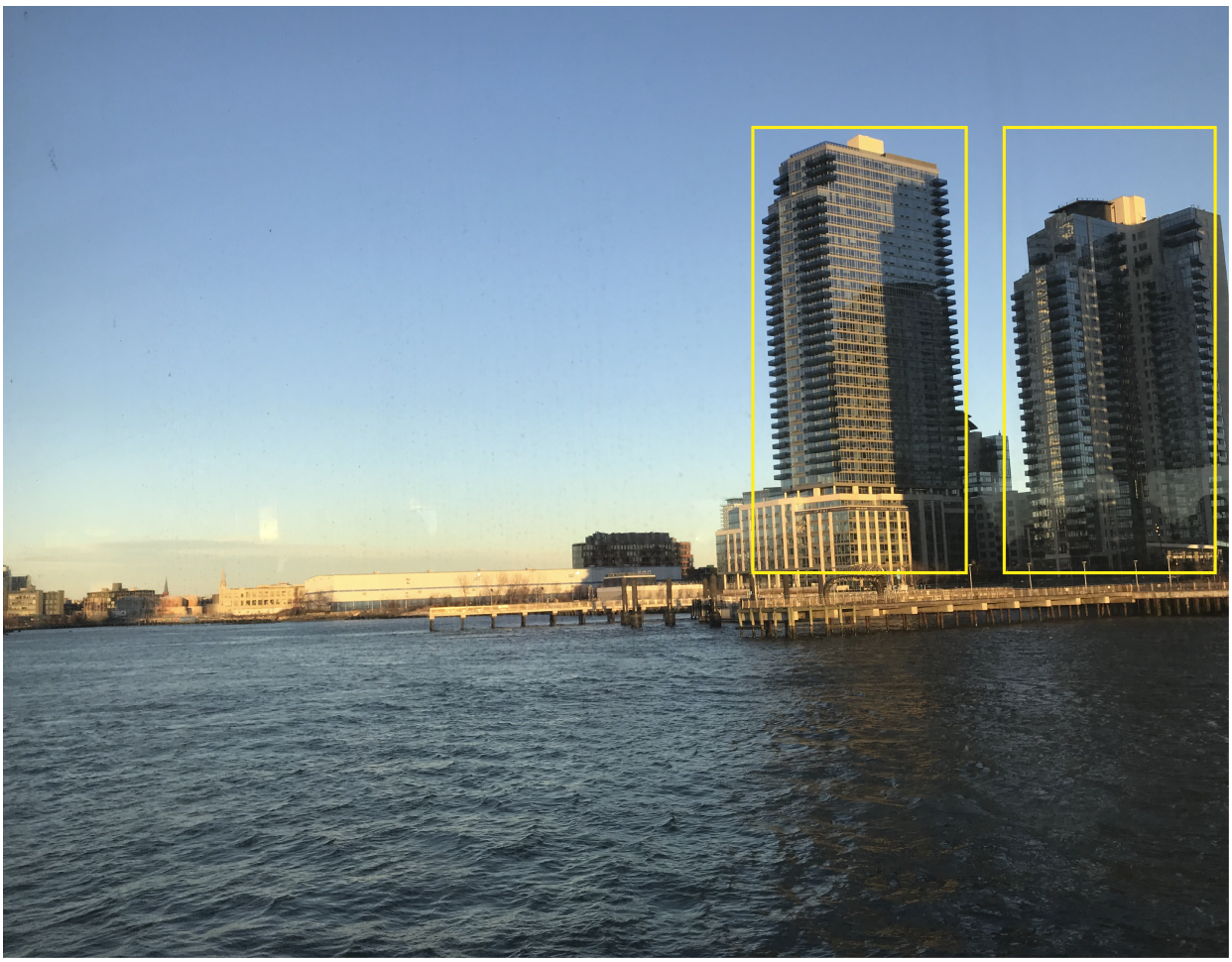


Figure 2.4: Region of Interest highlighted in the image by yellow rectangles.

or to change the representation of the image so that it has a more relevant meaning to the user. Segmentation in images is typically used to find objects or edges [13]. A saliency map is a representation of an image that shows the unique quality of each pixel [44].

Saliency maps in 2D images have been studied for many years [8], [11]–[14], including technologies such as neural networks and others. They were developed with the purpose of identifying, in images, regions that receive greater attention in human visualization.

One of the first groups of neuroscientists to study and define a saliency map states in their work [45], [46] that the purpose of a saliency map is to represent the visibility, or salience, at all locations in the visual field by a scalar quantity and to guide the selection of observed locations, based on the spatial distribution of the salience.

The salience is defined as how different a location is from its surroundings in terms of color, orientation, movement and depth.

Thus, it can be summarized that the general idea of a saliency map is that it is a topographically organized map that indicates the location of salient objects in the visual field and not what such objects are. Therefore, a saliency map created from a digital image should be able to compute the

visibility of each pixel of the image and select the areas that have the most prominence [46]. In Figure 2.5, it is shown an image and its corresponding saliency map.

(a) Input image.

(b) Saliency map.

Figure 2.5: Example of a saliency map. [47].

2.5 FACE DETECTION ALGORITHMS

There are many different algorithms for face detection in the two-dimensional scenario of computer vision. In this work, we opted to use the well known Viola-Jones Algorithm [5] and the Facial Landmarks Algorithm described in [48]. It is not the objective of this work compare the two different algorithms, each one was used associated with a different method of adapting the 2D information for the 3D point clouds.

2.5.1 THE VIOLA-JONES ALGORITHM

The Viola-Jones object detection framework was the first object detection framework to provide competitive real-time object detection rates. It was proposed by Paul Viola and Michael Jones in 2001 [5].

The problem to be solved is the detection of faces in an image, a task easily performed by a human being, but that for a computer requires precise instructions and constraints. To reduce complexity, the framework requires full-view frontal faces, i.e. the face must point at the camera and it cannot be inclined.

This framework is also known for its low computational cost and high rate of true positives with low false positives.

The Viola-Jones algorithm has some attributes that qualify it as a good detection algorithm, they are:

- Robustness: high detection rate with few false positives and negatives;
- Real-time: it can detect faces at 15 frames per second;
- Face detection only: the goal is to decide between faces and non-faces.

It runs in three different stages:

1. Feature Selection;
2. AdaBoost Training;
3. Cascade of classifiers.

A brief description of each stage of the algorithm is provided in the following.

2.5.1.1 FEATURE SELECTION

The object detection procedure of the algorithm classifies images based on the value of simple features, see figure 2.6. The main motivation for using features instead of pixels directly is that features encode ad-hoc domain knowledge which is difficult to learn using a finite amount of training data. Another justification by the authors for using features was that a features-based system is much faster than a pixel-based system. The features used have similarities with Haar basis functions that have been used in other articles for object detection [49].

The authors used what was called an image integral to rapidly compute rectangular features. A point (x, y) in an integral image contains the sum of the pixels above and to the left of it, as shown in equation 2.6, where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i\left(x', y'\right) \tag{2.6}$$

Using equations 2.7 and 2.8, the integral image can be calculated with just one pass in the original image.

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{2.7}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{2.8}$$

where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$ and $ii(-1, y) = 0$.

The use of integral images is important because it allows that any rectangular sum can be calculated with only four array references. See figure 2.7.

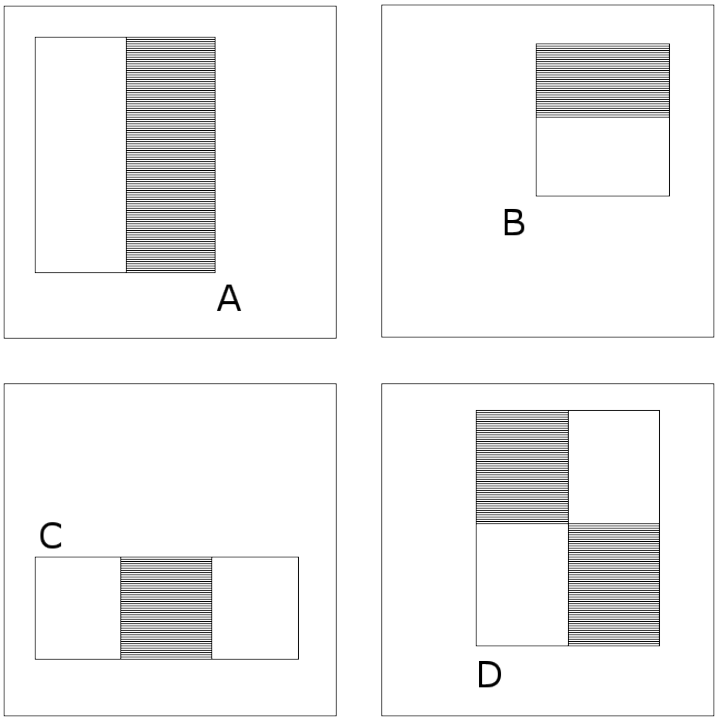


Figure 2.6: Example of feature rectangles relative to the detection window. The sum of pixels in the white rectangle is subtracted from the sum of pixels in the gray rectangle [50].

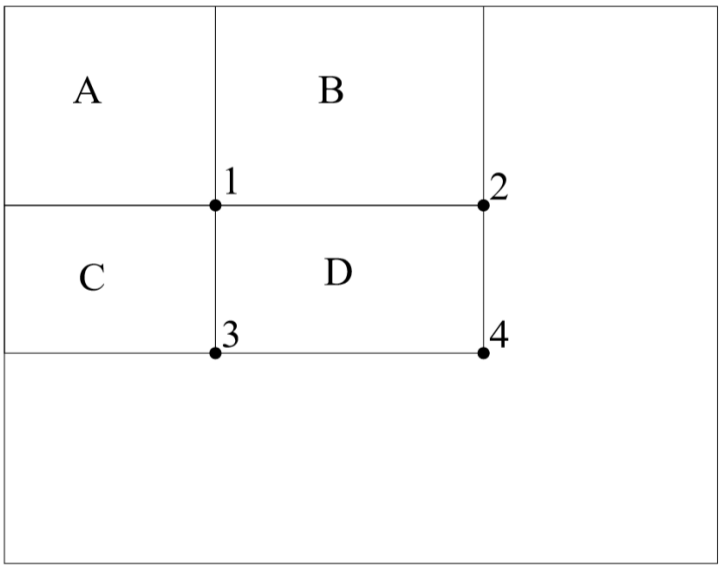


Figure 2.7: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at 3 is $A + C$, and at 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$ [50].

2.5.1.2 ADABOOST TRAINING

The system implemented by Viola and Jones uses a variant of AdaBoost to select the features and train the classifiers [51]. In its original form, the AdaBoost learning algorithm is used to improve the performance of a simple learning algorithm. It does this by combining a collection of weak classification functions to build a strong classifier.

As there are more than 180 thousand rectangle features associated with each image sub-window, it is proposed the hypothesis that a small number of these features can be combined to form an effective classifier. The biggest challenge is to find such features.

To do this, the authors designed the weak learning algorithm to select the single rectangle feature that best separates the positive and negative examples. For each feature, it is determined the optimal threshold classification function, such that the minimum number of examples are misclassified. A weak classifier $h_j(x)$ is made of an feature f_j, a threshold θ_j and a parity p_j indicating the direction of the inequality sign, as seen in Equation 2.9.

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

A simplified version of the learning algorithm is presented as follows:

Algorithm 1: The AdaBoost algorithm for classifier learning

Result: Selection of one feature from all the potential features.

Given examples images $(x_1, y_1), \ldots, (x_n, y_n)$, where $y_i = 0, 1$ for negative and positive examples respectively;

Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ to $y_i = 0, 1$, respectively, where m and l are the number of negatives and positives respectively;

for $t = 1, \ldots, T$ **do**

Normalize the weights

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{2.10}$$

so that w_t is a probability distribution;

for *each feature* j **do**

Train a classifier h_j which is restricted to using a single feature;

Evaluate the error with respect to w_t,

$$\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right| \tag{2.11}$$

end

Choose the classifier, h_t, with the lowest error ϵ_t;

Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \tag{2.12}$$

where $e_i = 0$ if example $x_i = 0$ is classified correctly, $e_i = 1$ otherwise and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$;

end

The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

where $\alpha_t = \log \frac{1}{\beta_t}$;

2.5.1.3 CASCADE OF CLASSIFIERS

To be able to improve face detection rates and decrease false positive rates, the authors have developed a system of cascading classifiers so that simpler classifiers are used to reject most negative images before more complex classifiers are used to achieve low false positive rates.

The general form of the detection process is that of a degenerate decision tree, as it can be seen in Figure 2.8. A positive result of the first classifier triggers the evaluation by a second classifier, that as also adjusted to reach high detection rates. A positive result in the second classifier triggers the evaluation by the next classifier and so on. A negative result in any link of the chain leads to

immediate rejection of the sub-window.

The classifiers were built, using AdaBoost, adjusting the threshold to minimize false negatives.

Cascading the classifiers allows a lower computational cost since most of the evaluated sub-windows will have negative results, so they are eliminated at the beginning of the processing, avoiding as much as possible that one sub-window is evaluated by all the classifiers.

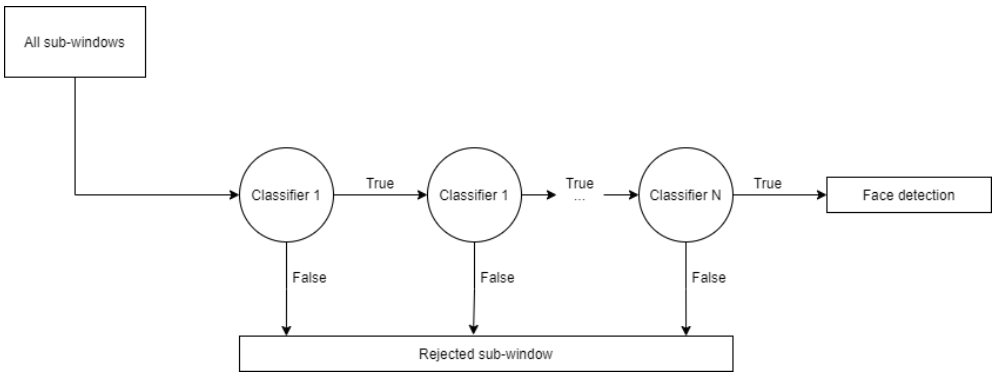


Figure 2.8: Cascaded classifiers diagram.

2.5.2 THE FACIAL LANDMARKS ALGORITHM

The paper written by V. Kazemi and J. Sullivan [48] presents a method for detecting faces even if they are not facing the camera in a upfront position. They demonstrate how an ensemble of regression trees can be used to estimate the face landmark positions directly from a sparse subset of pixel intensities in a computationally efficient way.

Similar to previous works this method uses a cascade of regressors and some facial landmarks, or features, to identify the faces in an image. A brief description of the framework is given bellow.

2.5.2.1 CASCADE OF REGRESSORS

Let $\mathbf{x}_i \in \mathbb{R}^2$ be the x, y-coordinates of the ith facial landmark of an image I. Then the vector $\mathbf{S} = \left(\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_p^T\right)^T \in \mathbb{R}^{2p}$ denotes the coordinates of all the p facial landmarks in I. $\hat{\mathbf{S}}$ is used to denote the estimate of $\mathbf{S}$. Each regressor, r_t, in the cascade predicts an update value from the image and $\hat{\mathbf{S}}$ that is added to the current $\mathbf{S}$ estimate to improve it:

$$\hat{\mathbf{S}}^{(t+1)} = \hat{\mathbf{S}}^{(t)} + r_t\left(I, \hat{\mathbf{S}}^{(t)}\right). \tag{2.14}$$

The main point of the cascade in that the regressor makes its predictions based on features computed from I and indexed relative to the current $\hat{\mathbf{S}}$. To train each r_t it is used a gradient tree boosting algorithm with a sum of square error loss [52].

Assuming a training data $(I_1, \mathbf{S}_1), \ldots, (I_n, \mathbf{S}_n)$ where I_i represents a face image and $\hat{\mathbf{S}}_i$ its shape vector. To learn the first regression function r_0, it is created data triplets of a face image, an initial

shape estimate and the target update step $\left(I_{\pi_i}, \hat{\mathbf{S}}_i^{(0)}, \Delta\mathbf{S}_i^{(0)}\right)$, with that:

$$\begin{aligned} \pi_i &\in \{1, \ldots, n\} \\ \hat{\mathbf{S}}_i^{(0)} &\in \{\mathbf{S}_1, \ldots, \mathbf{S}_n\} \setminus \mathbf{S}_{\pi_i} \text{ and} \\ \Delta\mathbf{S}_i^{(0)} &= \mathbf{S}_{\pi_i} - \hat{\mathbf{S}}_i^{(0)} \end{aligned} \tag{2.15}$$

for $i = 1, \ldots, N$ and $N = nR$ where R is the number of initializations used per image. From this data the regression function r_0 is learned using gradient tree boosting with a sum of square error loss. The set of training triplets is then updated to provide the training data, $\left(I_{\pi_i}, \hat{\mathbf{S}}_i^{(1)}, \Delta\mathbf{S}_i^{(1)}\right)$, for the next regressor r_1 in the cascade by setting

$$\begin{aligned} \hat{\mathbf{S}}_i^{(t+1)} &= \hat{\mathbf{S}}_i^{(t)} + r_t\left(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)}\right) \\ \Delta\mathbf{S}_i^{(t+1)} &= \mathbf{S}_{\pi_i} - \hat{\mathbf{S}}_i^{(t+1)} \end{aligned} \tag{2.16}$$

with $t = 0$. This process is iterated until a cascade of T regressors $r_0, r_1, \ldots, r_{T-1}$ is learned which when combined give a sufficient level of accuracy. Included in the statement of the algorithm is a learning rate parameter, or shrinkage factor, $0 < \nu \leq 1$. The algorithm in 2 summarizes the process of learning each regressor in the cascade.

Algorithm 2: Learning r_t in the cascade.

Result: Regressor r_t

Have training data $\left\{\left(I_{\pi_i}, \hat{\mathbf{S}}_i^{(1)}, \Delta\mathbf{S}_i^{(1)}\right)\right\}_{i=1}^{N}$;

$0 < \nu < 1$;

Set

$$f_0\left(I, \hat{\mathbf{S}}^{(t)}\right) = \arg\min_{\gamma \in \mathbb{R}^{2p}} \sum_{i=1}^{N} \left\|\Delta\mathbf{S}_i^{(t)} - \gamma\right\|^2 \tag{2.17}$$

for $k = 1, \ldots, K$ **do**

  for *each feature* j **do**

    Set

$$\mathbf{r}_{ik} = \Delta\mathbf{S}_i^{(t)} - f_{k-1}\left(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)}\right) \tag{2.18}$$

  end

  Fit a regression tree to the targets $\mathbf{r}_{ik}$ giving a weak regression function $g_k\left(I, \hat{\mathbf{S}}^{(t)}\right)$;

  Update

$$f_k\left(I, \hat{\mathbf{S}}^{(t)}\right) = f_{k-1}\left(I, \hat{\mathbf{S}}^{(t)}\right) + \nu g_k\left(I, \hat{\mathbf{S}}^{(t)}\right) \tag{2.19}$$

end

Output $r_t\left(I, \hat{\mathbf{S}}^{(t)}\right) = f_K\left(I, \hat{\mathbf{S}}^{(t)}\right)$;

2.5.2.2 TREE-BASED REGRESSOR

The core of each regression function r_t is the tree based regressors fit to the residual targets during the gradient boosting algorithm. At each split node in the regression tree a decision is made based on thresholding the difference between the intensities of two pixels. The pixels used in the test are at positions $\mathbf{u}$ and $\mathbf{v}$ when defined in the coordinate system of the mean shape. For a face image with an arbitrary shape, the points that have the same position relative to the shape as $\mathbf{u}$ and $\mathbf{v}$ have to the mean shape are indexed.

Let $k_{\mathbf{u}}$ be the index of the facial landmark in the mean shape that is closest to $\mathbf{u}$ and define its offset from $\mathbf{u}$ as:

$$\delta\mathbf{x}_{\mathbf{u}} = \mathbf{u} - \overline{\mathbf{x}}_{k_{\mathbf{u}}}. \tag{2.20}$$

For $\mathbf{S}_i$ defined in image I_i, the position in I_i that is qualitatively similar to $\mathbf{u}$ in the mean shape image is described by:

$$\mathbf{u}' = \mathbf{x}_{i,k_{\mathbf{u}}} + \frac{1}{s_i} R_i^T \delta\mathbf{x}_{\mathbf{u}}, \tag{2.21}$$

where s_i and R_i are the scale and rotation matrix of the similarity transform which transforms $\mathbf{S}_i$ to $\overline{\mathbf{S}}$. The scale and rotation are used to minimize the sum of squares between the mean shape facial landmark $\overline{\mathbf{x}}_j$, and those of the warped shape, that is given by:

$$\sum_{j=1}^{p} \left\| \overline{\mathbf{x}}_j - \left(s_i R_i \mathbf{x}_{i,j} + \mathbf{t}_i \right) \right\|^2. \tag{2.22}$$

In the end, each split is a decision involving three parameters $\theta = (\tau, \mathbf{u}, \mathbf{v})$ and is applied to each training and test example as shown:

$$h\left(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)}, \theta\right) = \begin{cases} 1 & I_{\pi_i}(\mathbf{u}') - I_{\pi_i}(\mathbf{v}') > \tau \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

where $\mathbf{u}'$ and $\mathbf{v}'$ are defined using the scale and rotation matrix which best warp $\hat{\mathbf{S}}_i^{(t)}$ to $\overline{\mathbf{S}}$.

The results presented by the authors show that the framework developed is capable of detecting faces even when they are not facing the camera in an upfront position or when there is an occlusion of some part of the face. Some of the results can be seen in Figure 2.9.

2.6 RELATED WORK

There are some previous work on determining saliency maps and detecting faces in 3D objects [20], [21], [53].

The saliency map creation proposed by T. Zheng *et al.* [53] is based on the importance of the point for a model-recognition loss. Thus, their saliency map explicitly explains which points are the

key for model recognition. Furthermore, aggregations of highly-scored points indicate important segments in a point cloud.

The method developed by P. Nair *et al.* [20] is based on a 3D point distribution model (PDM), which is fitted with structure and position information, and candidate points. Face detection is performed by classifying transformations between model points and candidate vertices based on the upper bound of the deviation of the model parameters. In the paper a 99.6% face detection rate is reported. The method was developed to work with face meshes and rely on curvature-based feature maps, so it is not directly applicable on point clouds.

In the work of A. Colombo *et al.* [21] a feature-based approach is combined with a holistic approach for three-dimensional face detection. Features such as the eyes and nose are detected through an analysis of the surface curvature. Each candidate trio of eye and nose pair is processed by a classifier trained to distinguish between faces and non-faces. The results presented by the author indicate a hit rate of 96.85%.

There is no work available in the literature regarding the encoding of point clouds using regions of interest and neither about a projection-based region of interest creation for point clouds. With this, the method proposed aggregates these two ideas.

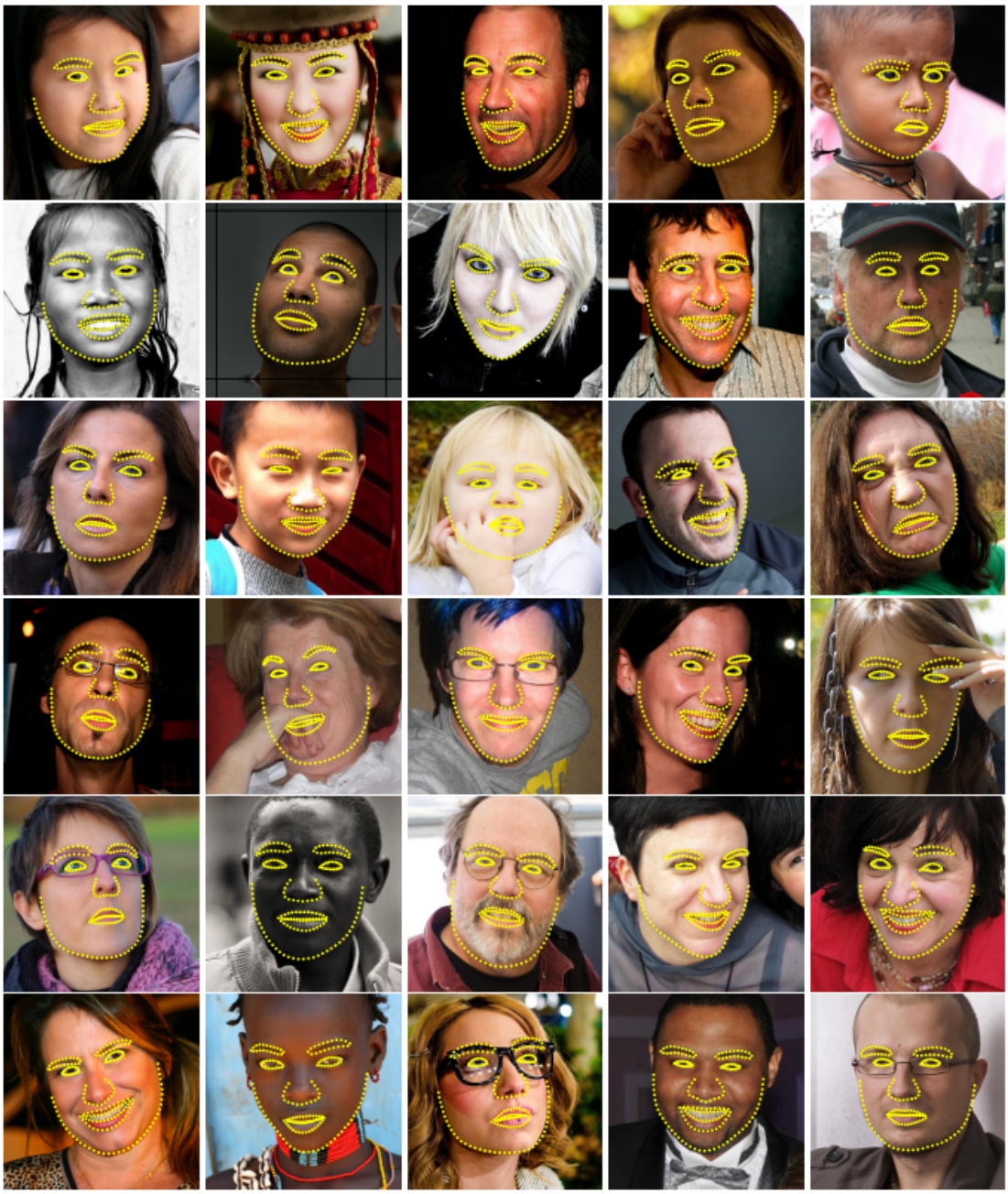


Figure 2.9: Results presented in [48].

3 PROPOSED METHOD

This chapter presents the methodology of the work described in this dissertation as well as a detailed description of the algorithms used to implement the final solution.

We divided our study into two parts:

1. Face detection as a ROI in a point cloud;
   (a) Single frame;
   (b) Frame sequence;
2. Saliency map calculation as a soft ROI in a point cloud.

Both types of regions of interest were further used to encode the point clouds. The face detection method was divided in two different approaches, the single frame and the frame sequence, which are detailed next:

- **Region of interest detection in a single frame of a point cloud:** this method searches for the region of interest in only one frame of a point cloud. The input of this method is a *voxelized* point cloud from which no information about the location of the ROI or its existence is available. Its output is the locality of the *voxels* of the ROI obtained.
- **Region of interest detection in a sequence of frames of a point cloud:** This method searches for the ROI in a frame sequence of a point clouds making use of temporal consistency to optimize the search and to decrease the computational cost. The input of this method is a frame sequence of a *voxelized* point cloud where there is no information about the location of the ROI or its existence in the first frame. The information of the previous processed frame is used to process the current frame. Outputs the location of the *voxels* in the obtained ROI for each frame.

The diagram in Figure 3.1 represents the developed algorithm for ROI detection and extraction in a point cloud.

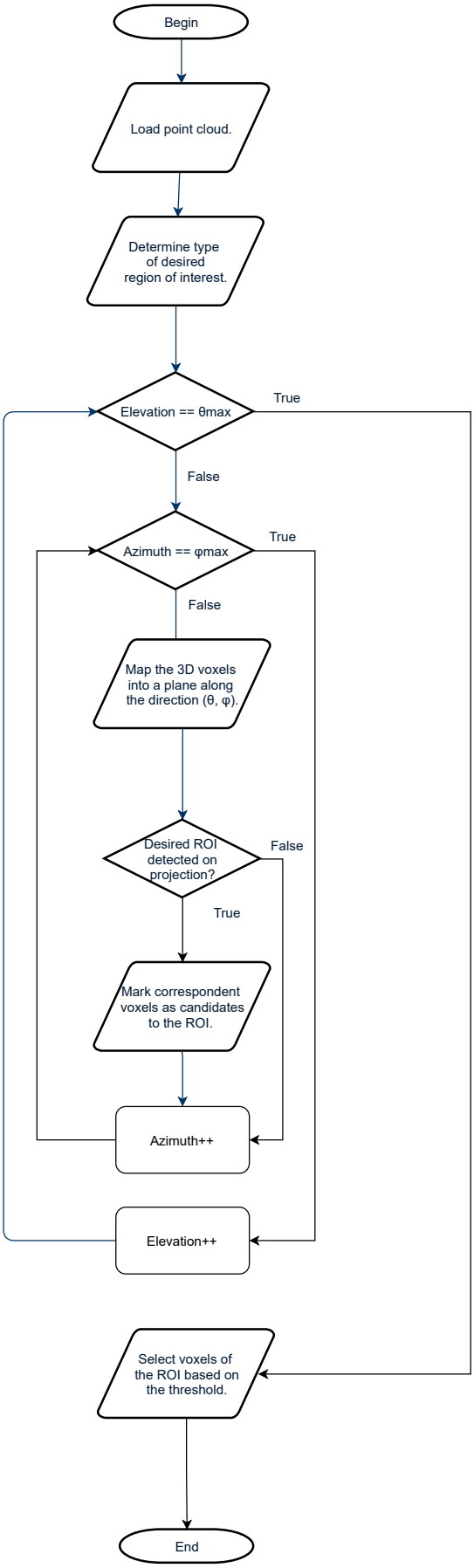


Figure 3.1: Diagram of the proposed algorithm for region of interest detection in point clouds.

3.1 ROI-WEIGHTED DISTORTION MEASURE AND MEASURE-THEORETIC RAHT

In lossy compression, artefacts introduced in some regions can highly influence the subjective perception of quality. The squared error may not correlate well with subjective perception of quality.

Let a generic attribute $\mathbf{X} = \{X_i\}$ be considered, where X_i is the attribute value associated with the i-th *voxel* of a total of N occupied *voxels*, and let $\hat{\mathbf{X}} = \{\hat{X}_i\}$ be its reconstruction. A better way to account for the relevance is to consider a *weighted squared error*, defined as:

$$d(\mathbf{X}, \hat{\mathbf{X}}) = \sum_i w_i (X_i - \hat{X}_i)^2, \tag{3.1}$$

where w_i are the *weights* associated with each *voxel* and reflects its semantic or perceptual importance.

A region of interest (ROI) can be defined via the weights w_i. A natural choice is to set $w_i = 1$ for all *voxels* outside the ROI and $w_i > 1$ for *voxels* inside the ROI. A codec that minimizes this distortion measure subject to a rate constraint will tend to reproduce X_i as $\hat{X}_i$ with squared error inversely proportional to w_i. It is also possible to define a smooth transition between the ROI border or define multiple ROI with different weights.

The weights can be used to define a measure on the set of *voxels* composing the point cloud. A measure on a set is a systematic way to assign a number to each suitable subset of that set, intuitively interpreted as its size. As each *voxel* has a weight associated with it, we define the measure of a subset S of *voxels* by:

$$\mu(S) = \sum_{i \in S} w_i. \tag{3.2}$$

The measure is non-negative, as all weights are non-negative and the measure of two disjoint subsets is equal to the sum of their measure.

The definition of measure (3.2) induces the definition of the integral,

$$\int f(\mathbf{x}) d\mu(\mathbf{x}) = \sum_i w_i f(\mathbf{x_i}) \tag{3.3}$$

because

$$\frac{d\mu}{d\mathbf{x}} = \sum_i w_i \delta\left(|\mathbf{x} - \mathbf{x_i}|^2\right) \tag{3.4}$$

since *voxels* are dispersed in discrete positions $\mathbf{x_i}$.

The definition of the integral (3.3) in turn induces the definition of the inner product,

$$\langle f, g \rangle = \int f(\mathbf{x}) g(\mathbf{x}) d\mu(\mathbf{x}) = \sum_i w_i f(\mathbf{x_i}) g(\mathbf{x_i}), \tag{3.5}$$

which in turn induces the definitions of orthogonality, $f \perp g \Leftrightarrow \langle f, g \rangle = 0$, and norm, $||f|| = (\langle f, f \rangle)^{1/2}$. Altogether, these induce a Hilbert space. The weighted squared error given by (3.1) is precisely the squared norm $||f - \hat{f}||^2$ of this Hilbert space, where $f_i = X_i$ and $\hat{f}_i = \hat{X}_i$.

RAHT is region-adaptive to remain orthonormal regardless the locations of the points. Recently RAHT has been shown to be interpretable as a separable piecewise constant spline wavelet that is orthonormal with respect to the inner product $\langle f, g \rangle$ defined by the weights w_i [54]. Thus, if the weights are set to the weights in the ROI-weighted distortion measure, the transform will remain orthonormal, and moreover uniform scalar quantization of the transform coefficients with the quantization stepsize set to a constant will minimize the ROI-weighted distortion measure, at least at high rates.

To be specific, let $\mathbb{R}^3$ be uniformly partitioned into cubes of size $2^{-m} \times 2^{-m} \times 2^{-m}$, half-cubes of size $2^{-m} \times 2^{-m} \times 2^{-(m+1)}$, and quarter-cubes of size $2^{-m} \times 2^{-(m+1)} \times 2^{-(m+1)}$, and let $\mathcal{F}_{3m}$, $\mathcal{F}_{3m+1}$, and $\mathcal{F}_{3m+2}$ be the spaces of all functions $f_\ell : \mathbb{R}^3 \to \mathbb{R}$ that are piecewise constant on these blocks, for $\ell = 3m$, $3m+1$, and $3m+2$, respectively. The nested sequence of function spaces $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_\ell \subseteq \mathcal{F}_{\ell+1} \subseteq \cdots$ approximates ever more finely (with respect to the norm, i.e., the weighted squared error) the space of piecewise continuous functions.

Now let $B_{\ell,n}$ denote a block at level ℓ indexed by n, let $1_{B_{\ell,n}}(\mathbf{x})$ be its indicator function, and let $w_{\ell,n} = \mu(B_{\ell,n})$ be its measure. Then $\mathcal{F}_\ell$ is spanned by the basis functions

$$\phi_{\ell,n}(\mathbf{x}) = w_{\ell,n}^{-1/2} 1_{B_{\ell,n}(\mathbf{x})}, \tag{3.6}$$

which are orthogonal to each other and are normalized with respect to the inner product and norm induced by the weighted measure. Similarly, let $B_{\ell+1,n_0}$ and $B_{\ell+1,n_1}$ denote the sub-blocks of $B_{\ell,n}$, and let $\mathcal{G}_\ell$ be the orthogonal complement of $\mathcal{F}_\ell$ in $\mathcal{F}_{\ell+1}$. Then $\mathcal{G}_\ell$ is spanned by the basis functions

$$\psi_{\ell,n}(\mathbf{x}) = \frac{-w_{\ell+1,n_0}^{-1} 1_{B_{\ell+1,n_0}}(\mathbf{x}) + w_{\ell+1,n_1}^{-1} 1_{B_{\ell+1,n_1}}(\mathbf{x})}{(w_{\ell+1,n_0}^{-1} + w_{\ell+1,n_1}^{-1})^{-1/2}} \tag{3.7}$$

which are orthogonal to each other and to the functions (3.6), and are normalized, as can be verified by the diligent reader. Thus any function $f_{\ell+1} \in \mathcal{F}_{\ell+1}$ can be written as:

$$f_{\ell+1}(\mathbf{x}) = \sum_n F_{\ell,n} \phi_{\ell,n}(\mathbf{x}) + \sum_n G_{\ell,n} \psi_{\ell,n}(\mathbf{x}), \tag{3.8}$$

where the $F_{\ell,n} = \langle f_{\ell+1}, \phi_{\ell,n} \rangle$ are known as low-pass coefficients and the $G_{\ell,n} = \langle f_{\ell+1}, \psi_{\ell,n} \rangle$ are known as high-pass coefficients. After some algebraic manipulation, (3.6) and (3.7) can be expressed recursively as the "two-scale equations"

$$\phi_{\ell,n}(\mathbf{x}) = a\phi_{\ell+1,n_0} + b\phi_{\ell+1,n_1} \tag{3.9}$$

$$\psi_{\ell,n}(\mathbf{x}) = -b\phi_{\ell+1,n_0} + a\phi_{\ell+1,n_1}, \tag{3.10}$$

where $a = \frac{\sqrt{w_{\ell+1,n_0}}}{\sqrt{w_{\ell,n}}}$ and $b = \frac{\sqrt{w_{\ell+1,n_1}}}{\sqrt{w_{\ell,n}}}$. Substituting these into the definitions of $F_{\ell,n}$ and $G_{\ell,n}$, we obtain

$$\begin{bmatrix} F_{\ell,n} \\ G_{\ell,n} \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} F_{\ell+1,n_0} \\ F_{\ell+1,n_1} \end{bmatrix}, \tag{3.11}$$

which is a Givens rotation whose angle of rotation depends on the relative weights of the sub-blocks. RAHT applies (3.11) recursively to expand $f_L \in \mathcal{F}_L$ as

$$f_L(\mathbf{x}) = \sum_n F_{0,n}\phi_{0,n}(\mathbf{x}) + \sum_{\ell=0}^{L-1}\sum_n G_{\ell,n}\psi_{\ell,n}(\mathbf{x}), \tag{3.12}$$

where L is chosen large enough so that each cube $B_{L,n}$ contains at most a single point, say $\mathbf{x}_i$ with value $f_i = f(\mathbf{x}_i)$. The number of coefficients is N, i.e., RAHT is critically sampled. (For details, see [54].) Note that $\phi_{L,n}(\mathbf{x}) = w_i^{-1/2}1_{B_{L,n}}(\mathbf{x})$, and therefore $F_{L,n} = \langle f, \phi_{L,n}\rangle = w_i^{1/2}f_i$. This generalizes RAHT in [40], for which $w_i = 1$ for all points $i = 1, \ldots, N$.

The RAHT coefficients are uniformly scalar quantized with stepsizes $\Delta(F_{0,n})$ and $\Delta(G_{\ell,n})$, $\ell = 0, \ldots, L-1$, and are entropy coded. Because Givens rotations are orthonormal, norm is preserved. Thus the squared quantization error is

$$\sum_n (F_{0,n} - \hat{F}_{0,n})^2 + \sum_{\ell=0}^{L-1}\sum_n (G_{\ell,n} - \hat{G}_{\ell,n})^2 = \sum_{i=1}^{N} w_i(f_i - \hat{f}_i)^2, \tag{3.13}$$

which is the same as the ROI-weighted distortion measure. Since a constant step-size $\Delta = Q_{step}$ minimizes the squared quantization error subject to an entropy constraint, at least at high rates [55], setting the step-sizes of the RAHT coefficients to a constant also minimizes the ROI-weighted distortion measure desired for ROI coding [15].

In summary, with RAHT, at the encoder, *voxels* in ROI should have initial weights set to $w_i = w$ and initial attributes scaled by $\sqrt{w}$. The decoder should scale back the attributes.

3.2 PROJECTION-BASED ALGORITHM

Many computer vision algorithms have been extensively studied for the 2D image case, including those to create saliency maps and for face recognition. For point clouds, however, literature in the subject is scarce. Our goal is to find ROI such as faces and saliency maps in unstructured point clouds. In these, relative position among *voxels*, normals etc. may have to be derived, and efficient algorithms for feature identification are still being developed [56]–[58].

Our solution uses projections onto 2D images, in order to recognize the features in 2D space, and to map back the image pixels to the 3D *voxels*. In other words, computer vision tasks on 3D point clouds are performed on a 2D projection with the aid of a back-projection of 2D pixels onto 3D *voxels*. The data from many projections are fused to find the *voxels* of interest to us. The idea is illustrated in Figures 3.2 and 3.3.

3.2.1 POINT CLOUD PROJECTION

Orthographic projection is a type of parallel projection where the projection lines are drawn parallel to each other and perpendicular to the chosen plane of projection.

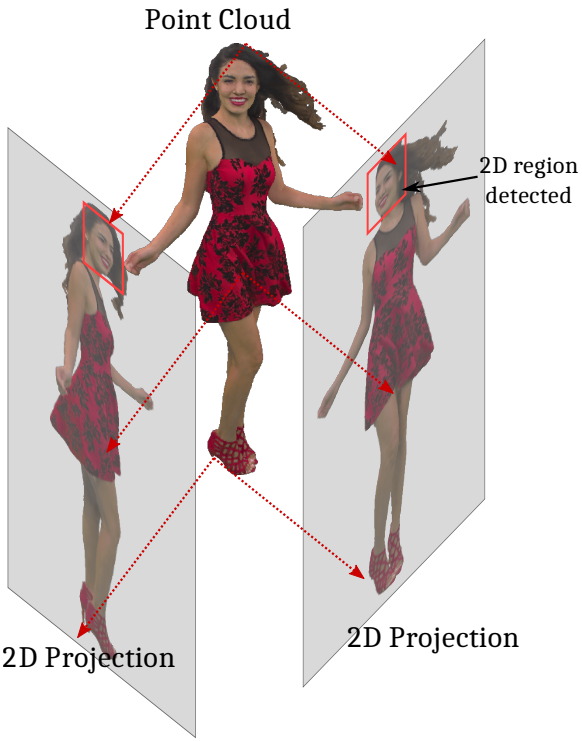


Figure 3.2: Region of interest detection in point clouds using projections and image identification algorithms. Detected pixels are mapped back onto *voxels*.

The algorithm begins by orthographically projecting the point cloud P onto a 2D plane I. If we imagine a *voxel* as a 3D cube and a pixel as a square element the size of the cube side, and, if we orthogonally project the cube into any of its six faces, we may be able to uniquely map the *voxel* face to a pixel in the 2D projection plane. Hence, the $P \rightarrow I$ mapping would be reversible, Fig. 3.4 illustrates this case.

If the projection is at any other oblique direction, the cube projection would not be a square, but a more complex polygon. Such a projection would not fit into a square pixel and would partially project onto many adjacent pixels. In order to cope with that situation, there are many solutions with varying degrees of accuracy and complexity. In $P \rightarrow I$ and $I \rightarrow P$, one solution is to compute the *voxel* or pixel color by linear combinations of the various partial projections. An alternative is to increase resolution by replicating *voxels* and pixels and simply assigning the *voxel* color to the pixel with the largest corresponding projection area. In the back-projection $I \rightarrow P$ we can mark the *voxel* whose center is the closest to the projection line from the center of a marked pixel in the 2D projection plane. After all *voxels* are marked, the point cloud should be reduced (down-sampled or averaged) to the correct resolution. Similar interpolation issues arise if one does not assume cubic *voxels* nor square pixels. Nevertheless, one should make sure it is possible to map *voxels* to pixels and to map specific pixels back to *voxels*.

The projection-based ROI determination algorithm works as:

- Map the 3D *voxels* into a plane along the direction (θ, ϕ), where $-90° \leq \theta \leq +90°$ is the elevation and $0° \leq \phi \leq 360°$ is the azimuth, see Fig 3.5.

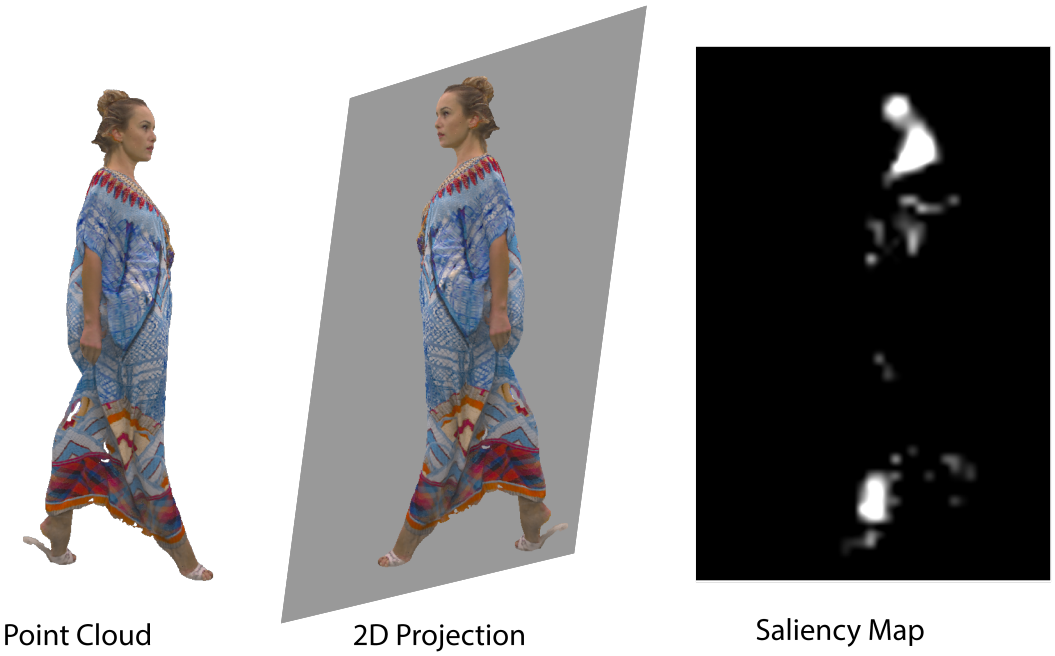


Figure 3.3: Steps for the creation of a saliency map using two-dimensional projection of a point cloud. Saliency maps are derived for images and the saliency pixels are mapped back onto *voxels*.

- Identify the ROI in 2D, marking the pixels that belong to it.
- Map the marked pixels back to the *voxels* in 3D. As a pixel may map to multiple *voxels*, one may use rounding or other decision process.

3.2.2 FACE DETECTION

With the algorithm described in section 3.2.1, given a *voxelized* point cloud and a pair (θ, ϕ) it is obtained a set of identified (marked) *voxels*. However, many directions are tested, since it is not known which orientation would be the best to identify the feature. The (θ, ϕ) space is scanned by spanning θ from θ_{min} to θ_{max} in steps of $\Delta\theta$ and ϕ from ϕ_{min} to ϕ_{max} in steps of $\Delta\phi$. If we have N_θ elevation and N_ϕ azimuth points, we end with $N_a = N_\theta N_\phi$ projection and back-projection cycles.

A region of interest is not always successfully identified. For example, only at a few viewpoints can we identify faces of a person. Let us say that out of N_a projections we identify the ROI N_R times. Then for each of the N_R projection cycles, we record the *voxels* that are marked "of interest". If a given *voxel* is marked "of interest" N_i times, we will then consider that particular *voxel* as belonging to the ROI if $N_i/N_R > \tau$, where τ is a given threshold. This method is information fusion by voting on *voxels*.

The algorithm is generic in nature but the focus here is in identifying faces of human point clouds as regions of interest. For that, it is used the algorithms described in section 2.5 for face detection in images. This work is not about new face detection algorithms and we simply used well

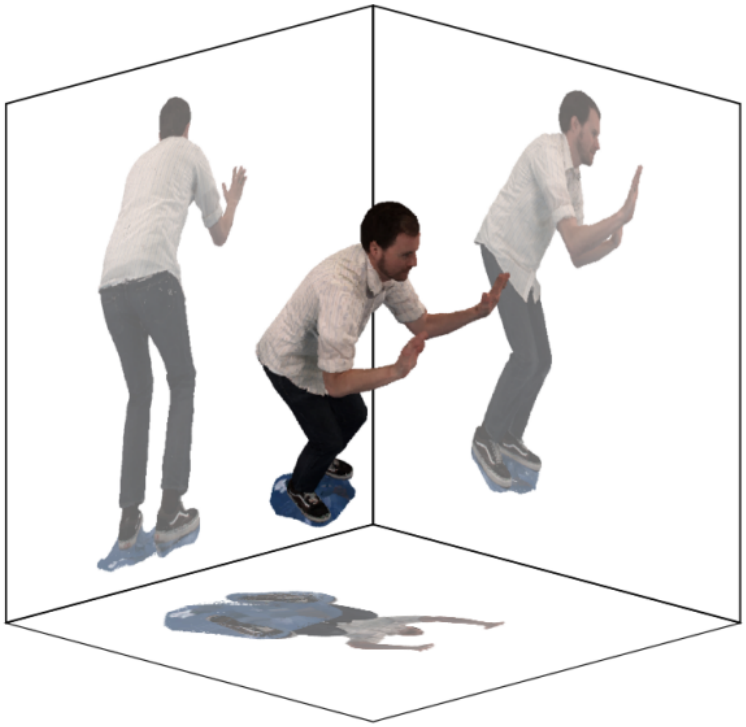


Figure 3.4: Orthogonal projections of a point cloud in three different views [59].

established algorithms to keep the focus on the projections and the compression. In the examples in this work, we used $\theta_{min} = -70°$, $\theta_{max} = 90°$, $\phi_{min} = 0°$, $\phi_{max} = 359°$, $\Delta\theta = \Delta\phi = 10°$. Hence, $N_\theta = 17$ and $N_\phi = 36$, so that $N_a = 612$ projections are performed for each point cloud. Detection is made with $\tau = 0.3$, i.e. the *voxel* is assigned as a face *voxel* if it has been identified as such in at least 30% of the projections.

Face detection may be problematic when viewing the face sideways, which causes gaps in the face region of interest in the cheeks and ears. To overcome this problem one can dilate the ROI mask in a special way. Each *voxel* in an integer grid can be addressed by its Morton code [60]. Morton codes are obtained by interleaving the binary representation of the *voxel* xyz integer coordinates, from the most to the least significant bit. If the point cloud has depth d, the Morton code has $3d$ bits. The first $3(d - w)$ bits of the code are an address to cubes of $2^w \times 2^w \times 2^w$ *voxels*.

The dilation algorithm works with cubes of width 2^w, such that if a *voxel* in the cube is deemed in the region of interest, all other occupied *voxels* in the cube are also made part of the ROI. These other *voxels* in the cube are easily found since they all share the same $3(d - w)$ bits of their Morton codes. In other words, if a *voxel* is deemed in the ROI all other *voxels* with the same $3(d - w)$ bits prefix in their Morton codes are also made part of the ROI. Fig. 3.6 shows results of expanding the region of interest using different widths (2^w).

In summary, the algorithm can be described as in Alg. 3.

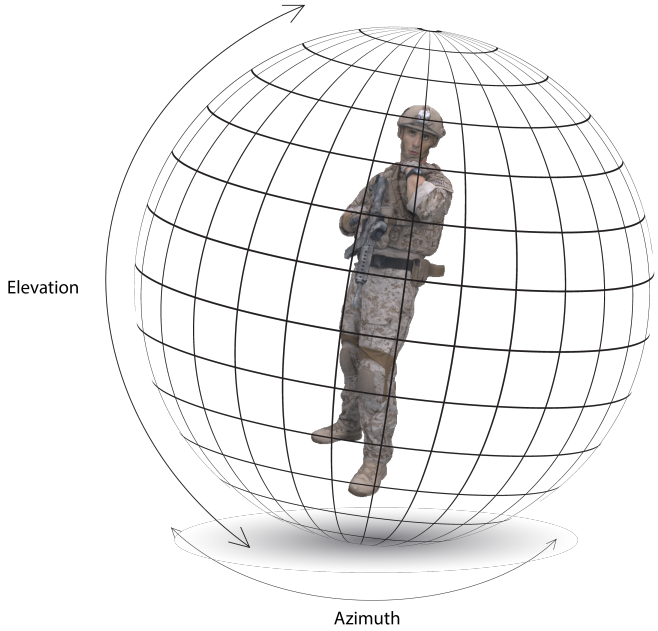


Figure 3.5: Representation of a point cloud and its elevation angle and azimuth.

(a) *width = 1* (b) *width = 2* (c) *width = 4* (d) *width = 8*

Figure 3.6: Artefacts in the region of interest identification caused by the obliquity of the projections and the Morton-code-based dilation for different cube widths.

Algorithm 3: Detecting the region of interest in a point cloud.

Result: Voxels belonging to the desired ROI

for $\theta = \theta_{min} : \Delta\theta : \theta_{max}$ **do**
  for $\phi = \phi_{min} : \Delta\phi : \phi_{max}$ **do**
    Project the point cloud onto direction (θ, ϕ);
    Run 2D object detection algorithm (e.g. Viola-Jones);
    Back-project "marked" pixels as "marked" *voxels*;
    Count N_R;
    Count N_i for each *voxel*;
  end
end
for *each voxel* **do**
  Mark it as ROI if $N_i / N_R > \tau$;
end
Dilate ROI using blocks (e.g. 8-sided, or $w = 3$);

3.2.3 SALIENCY MAP CREATION

The method developed for the saliency map creation for point clouds is very similar to the one described in section 3.2.2 for face detection in point clouds. Since the algorithm described in section 3.2.1 is generic by nature, the algorithm developed by Walther and Koch [61] was used to construct the saliency maps in the 2D projections. The same parameters of $\theta_{min} = -70°$, $\theta_{max} = 90°$, $\phi_{min} = 0°$, $\phi_{max} = 359°$ and $\Delta\theta = \Delta\phi = 10°$ were used, hence we perform $N_a = 612$ projections for each point cloud, resulting in the same number of 2D saliency maps.

After each projection, the salience value of each pixel is re-projected onto the corresponding *voxel* and added to the already existing *voxel* salience value. After the 612 projections, the salience value of a *voxel* is the sum of the salience values of the corresponding pixels in each projection. This method for merging the attributes is similar to the one presented by Niebur and Koch in their work [46].

At the end of all projections, the *voxels* salience values are normalized to a continuous range from 0.0 to 1.0. In order to smooth the transition between the salient region and the non-salient region, it is proposed a spatial low-pass filter with a cubic kernel of size $9 \times 9 \times 9$. The algorithm described in Algorithm 4 summarizes the proposed method.

Algorithm 4: saliency map creation for a point cloud.

```
Result: 3D saliency map.
for θ = θ_min : Δθ : θ_max do
    for φ = φ_min : Δφ : φ_max do
        Project the point cloud onto direction (θ, φ);
        Run saliency map creation algorithm;
        Re-project the salience value of the pixels onto the voxels;
        Sum the current salience value of the voxel with the correspondent pixel salience
         value;
    end
end
Normalize the salience values to a continuous range of 0.0 to 1.0;
Filter the saliency map with a low-pass filter;
```

3.3 TEMPORAL CONSISTENCY

In videos, temporal consistency is defined as the resemblance between one frame and the subsequent frame. This consistency is strongly explored in different ways such as for more efficiently compressing videos [62], object tracking [63], disparity estimation [64], etc. In dynamic point clouds, it is assumed that objects and regions do not change too much from one frame to another. For that, it was adopted a speed-up to explore the temporal consistency among frames of a sequence.

For a given frame, it is calculated the preferred viewing direction, or a center mode (θ_c, ϕ_c), in the (θ, ϕ) space. The goal is to find the direction with the most identified *voxels*, assuming the projection region with the most pixels and back-projected *voxels* may correspond to a frontal unobstructed view of the face or of other feature is being identified. The weighted average (centroid) of the distribution of views is taken. If at the i-th view direction at position (θ_i, ϕ_i) the algorithm identified n_i *voxels* then

$$\theta_c = \frac{\sum_i n_i \theta_i}{\sum_i n_i} \quad , \quad \phi_c = \frac{\sum_i n_i \phi_i}{\sum_i n_i}. \tag{3.14}$$

Assuming the center mode would not vary much from one frame to another, the search is concentrated for the next frame using new values of $\Delta\theta$, $\Delta\phi$, τ, while using $\theta_{min} = \theta_c - \theta_s$, $\theta_{max} = \theta_c + \theta_s$, $\phi_{min} = \phi_c - \phi_s$, $\phi_{max} = \phi_c + \phi_s$.

Our tests, the inter-frame mode uses $\theta_s = \phi_s = 20°$, $\Delta\theta = \Delta\phi = 4°$, and because of the more focused search space ($N_a = 121$) the threshold is raised to $\tau = 0.5$. Whenever θ_c and ϕ_c are void, *i.e.*, there was no previous ROI, the azimuth varies from 0° to 360°, in steps of 10°, and the elevation angle is varied from −70° up to 90° in steps of 10°.

In summary, with inter-frame consistency the (θ, ϕ) search space is narrowed around the previous-frame centroid (θ_c, ϕ_c), with smaller $\Delta\theta$, $\Delta\phi$ and larger τ.

3.4 ENCODING POINT CLOUDS WITH REGIONS OF INTEREST

When compressing a point cloud, there is always a trade off between the number of bits spent to encode the point cloud and the quality of the reconstructed point cloud at the decoder. The higher the quality, the more bits are necessary. Salient regions are supposed to have a higher semantic or perceptual significance than the rest of the point cloud. Therefore, an encoder that prioritizes the quality of salient regions or regions of interest, in detriment to other regions, tends to produce reconstructed point clouds with a better subjective quality, when compared to an encoder that treats all regions equally, for the same number of bits.

With the explanation given in section 3.1, one can try to encode a point cloud using regions of interest and the region adaptive hierarchical transform (RAHT) by raising the initial weights of those *voxels* marked as belonging to the ROI. In order to accomplish that, the region of interest location for a given frame needs to be conveyed to the decoder.

Let $\mathbf{b} = \{b_i\}$ be a binary vector with values indicating whether occupied *voxels* belong to the ROI or not, at that given frame. *Voxels* are sorted according to their associated Morton codes. Since neighbouring *voxels* have a high probability of belonging to the same category (ROI or non-ROI) and the Morton codes tend to preserve neighbourhood, one can convert vector $\{b_i\}$ in a differential vector $\{\overline{b_i}\}$ where

$$\overline{b_i} = \begin{cases} b_0 & i = 0 \\ 1 & b_{i-1} \neq b_i, i > 0 \\ 0 & b_{i-1} = b_i, i > 0 \end{cases} . \tag{3.15}$$

The vector $\{\overline{b_i}\}$ is expected to have long sequences of zeros and is encoded with run-length coding. Since the vector is differential and binary, we only encode the runs of $0s$. The run size is encoded with an adaptive Golomb-Rice code [65]. This is a simple coder. Others, more sophisticated coders, may be used as well.

The number of bits spent signalling the ROI is typically not significant compared to the overall encoding bit rate, around 0.003 bits per occupied *voxel* (bpov).

3.5 USING SALIENCY MAPS AS SOFT REGIONS OF INTEREST

In section 3.4, it was assumed that *voxels* belong to only two regions: ROI and non-ROI. For the saliency maps in this section, there is a smooth transition between *voxels* that are completely salient to those that are completely non-salient. The objective is to allow for the compression of point clouds using saliency maps.

As with the ROI the saliency map also needs to be conveyed to the decoder. In order to reduce the number of bits spent to send the saliency map as side information, we quantized the salience values in L levels, with that the saliency map becomes a discrete saliency map making it possible to send it to the decoder with less bits. In summary, the saliency map is quantized in L levels as

$$S(v_i) = \lfloor \xi(v_i)\, L \rfloor, \tag{3.16}$$

where $\lfloor . \rfloor$ represents the floor operation, $0 \leq \xi(v_i) < 1$ is the saliency value the i-th *voxel* and $S(v_i)$ is the quantized saliency value of the i-th *voxel*. Thus, the saliency map can be represented by integers where $S(v_i) \in \{0, 1, \ldots, L-1\}$.

Let also w_i be the weight assigned to that *voxel* v_i. We assume the same weight is assigned to all *voxels* with the same saliency level, i.e. we define the set of $\{\alpha_k\}$, where α_k is the single weight for all *voxels* such that $S(v_i) = k$.

A point cloud $\mathcal{P}$ is subdivided into L sub-point clouds as

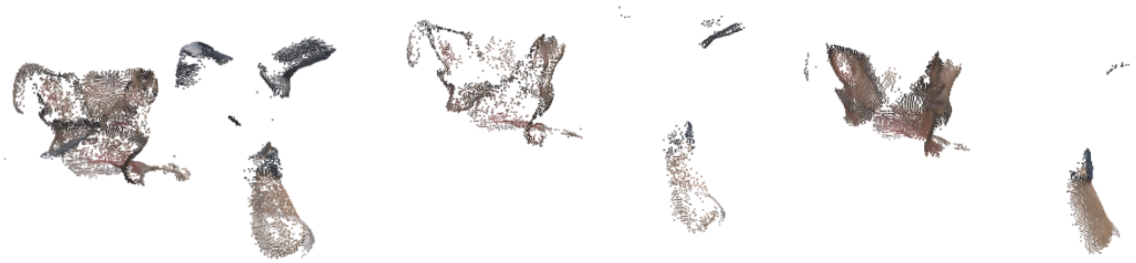
$$\mathcal{P} = \bigcup_{k=0}^{L-1} \mathcal{P}_k \tag{3.17}$$

such that

$$\mathcal{P}_k \equiv \{\, v_i \mid S(v_i) = k \,\}. \tag{3.18}$$

In summary, $\mathcal{P}_k$ is the sub-point-cloud composed by *voxels* with salience level k, for which we apply weights as α_k. Figure 3.7 shows an example of a point cloud and its sub-point-clouds.

(a) Original point cloud ($\mathcal{P}$). (b) $\mathcal{P}_0$. (c) $\mathcal{P}_1$.



(d) $\mathcal{P}_2$. (e) $\mathcal{P}_3$. (f) $\mathcal{P}_4$.

Figure 3.7: Point cloud "Soldier" (frame 695) and its sub-point clouds.

The quantized saliency map is sorted according to the Morton codes of the geometry of the corresponding *voxel* [60]. Morton code sorting preserves neighbourhood. We encode the saliency values with adaptive run-length / Golomb-Rice encoding (RLGR) [65]. RLGR performs better when there are long sequences of zeros. As neighbouring *voxels* tend to have similar salience, we take differences of the quantized saliency map prior to encoding with RLGR as $\overline{S[1]} = S[1]$ and

$$\overline{S[n]} = S[n] - S[n-1], \ \forall \, n > 1, \tag{3.19}$$

where $\overline{S[n]}$ is the n-th differential quantized value.

The encoder in Section 3.4 attributes a weight to each *voxel* as a non-negative integer value. The higher the weight, the better the quality. With the saliency map, the encoder and decoder can compute the weight for each *voxel*. Unoccupied *voxels* have a weight of 0, occupied *voxels* that are completely non-salient have a weight of 1, and completely salient occupied *voxels* have a weight of $\alpha \geq 1$. For *voxels* in the transition, the weight is linearly interpolated. Given the weights for each *voxel*, the encoding of the point clouds follows as in Section 3.4.

4 EXPERIMENTAL RESULTS

In order to objectively compare the results of the proposed coding method with the already existing ones, we use the Bjøntegaard delta rate (BD-Rate) and PSNR (BD-PSNR) [66] that are objective measures used in point cloud compression to compare the rate-distortion performance or compression efficiency of two different codecs or different settings of the same codec over a range of bit-rate or quality values.

When evaluating a point cloud codec performance, we generally obtain a rate-distortion (RD) curve that informs about the quality (dB) at different bit-rates (bpov). The BD-Rate and BD-PSNR metrics use the information from the RD curves and tells how much one codec has improved over the other. We can define the improvement of one codec over another as

$$\Delta_{PSNR} = BDPSNR(BR_1, PSNR_1, BR_2, PSNR_2)\ [dB], \tag{4.1}$$

$$\Delta_{Rate} = BDRate(BR_1, PSNR_1, BR_2, PSNR_2)\ [\%], \tag{4.2}$$

where BR_1 is a vector containing the rate obtained by $codec_1$ for the correspondent quality in vector $PSNR_1$, BR_2 and $PSNR_2$ are the equivalent for $codec_2$, Δ_{PSNR} represents the variation in quality for $codec_2$ over $codec_1$ at equivalent bit-rates and Δ_{Rate} represents the variation in rate for $codec_2$ over $codec_1$ to achieve the same quality.

In all the experiments performed the geometry is considered lossless and geometry compression is outside the scope of this work.

4.1 BINARY REGION-OF-INTEREST CODING

We first analyse the proposed method capability to detect the desired region of interest, which in this work was determined to be the face of a person. In order to test our method 9 point clouds were used: Longdress, Loot, Redandblack and Thaidancer, *voxelized* with a depth of 10 (*i.e.* $1024\times 1024\times 1024$ *voxels*), and Andrew, David, Phil, Ricardo and Sarah, *voxelized* with a depth of 9 (*i.e.* $512\times 512\times 512$ *voxels*), respectively [67]–[69]. The first 3 sequences have 300 frames while the last five sequences have 318, 216, 245, 216 and 207 frames, respectively. The point cloud Thaidancer was analysed only from a single-frame perspective.

Examples of projections of the identified faces for frames 1, 101 and 201 of each sequence are shown in Fig. 4.1. It also includes a failure example, where face detection failed for a particular frame. Subjectively the face is easily found in this case. A similar behaviour can be seen in the whole test dataset.

Encoding performance was evaluated with rate-distortion curves. The $PSNR$ (peak signal to noise ratio) measured in the Y, U and V channels, denoted by $PSNR_Y$, $PSNR_U$ and $PSNR_V$,

Figure 4.1: ROI identified in frames 1, 101 and 201 of each sequence. The image shown on the second line and second column represents a case when the algorithm was not able to detect the ROI.

and also the PSNR measured in all R, G, and B channels ($PSNR_{RGB}$) are highly correlated. The Pearson product-moment correlation coefficient [70] is shown in Table 4.1. It is apparent that the results are very highly correlated and there is not much information to gain by repeating the presentation of results for all channels and color spaces. Therefore, rate-distortion curves are only shown for the Y channel for simplicity. Nevertheless, it reflects, in average, the performance of the other color channels, for the purposes of this work.

Table 4.1: Correlation coefficient between $PSNR_Y$ and the other metrics for all point clouds tested

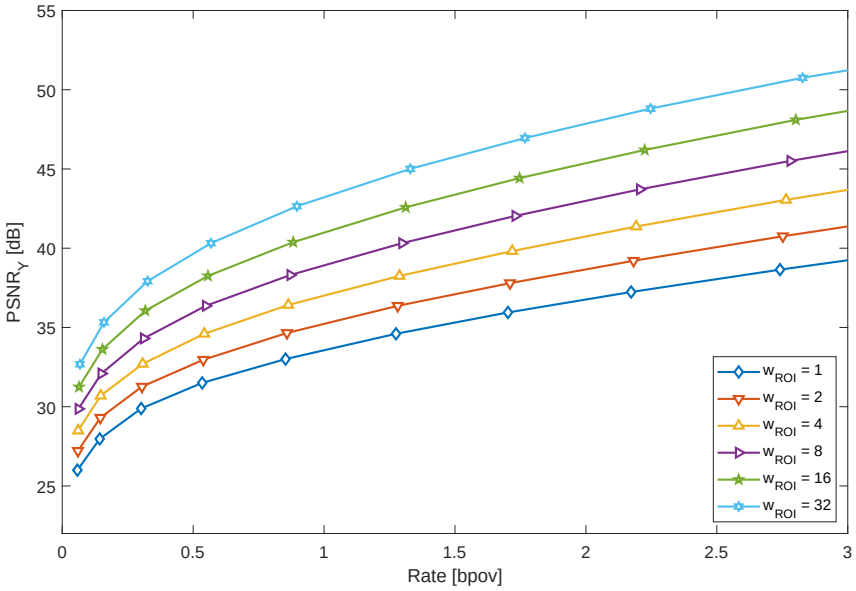
	$PSNR_{RGB}$	$PSNR_U$	$PSNR_V$
maximum	0.9999	0.9990	0.9990
average	0.9996	0.9911	0.9922
minimum	0.9993	0.9807	0.9791

Figure 4.2 shows average rate-distortion curves for sequence Longdress. The bit-rate includes the overhead to signal the region of interest. The weights for *voxels* in the ROI, w_{ROI}, were chosen as 1, 2, 4, 8, 16 and 32 while weights outside the ROI are 1. In Fig. 4.2(a) the PSNR was computed for *voxels* inside the ROI against rate in bits per occupied *voxels* (bpov), while w_{ROI} is varied. As expected when there is an increase in the w_{ROI}, the quality inside the ROI increases. When $w_{ROI} = 1$ all *voxels* have the exact same weight and it is the regular encoder. In Fig. 4.2(b), the PSNR was computed for *voxels* outside the ROI against rate in bits per occupied *voxels*, while w_{ROI} is varied. There is a very small difference among curves, in this case, because the ROI region is typically much smaller than the rest. Re-allocating a very small bit-rate over a large area can largely increase the ROI bit-rate, thus improving the ROI while keeping the degradation to non-ROI to a minimum. As a result, the performance impact of increasing w_{ROI} is not very significant in Fig. 4.2(b).

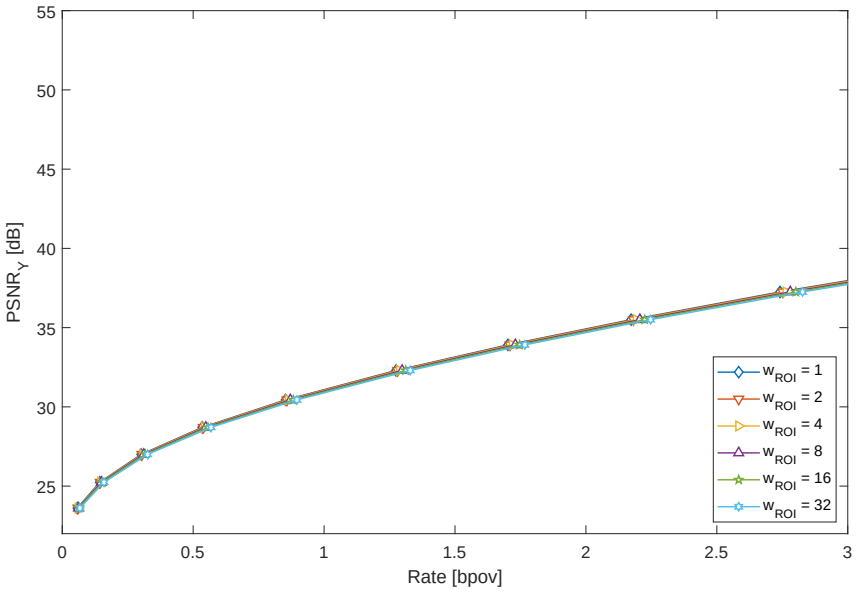
Figure 4.3 shows the average rate-distortion curves computed for one frame of each point cloud sequence in the tested dataset. Bits for the side information are included. In Fig. 4.3(a), the PSNR was computed only for *voxels* inside the ROI. In Fig. 4.3(b), the PSNR was computed for *voxels* outside the ROI. Figure 4.3(c) shows the standard PSNR computed for the entire point cloud and Fig. 4.3(d) shows the weighted PSNR computed for the entire point cloud.

Table 4.2 presents Bjøntegaard-delta PSNR (BD-PSNR) and rate (BD-RATE) [66] for rates from 0.08 to 1.0 bpov. Table 4.2 reflects an average of results for all sequences and frames, uses all *voxels* as reference and compares against either the ROI or non-ROI voxels. The point clouds are coded using the proposed method and compared against the point clouds coded using the traditional (non-ROI) method.

When $w_{ROI} = 1$ the distortion computed for all *voxels* only slightly differs from the ROI and non-ROI, as expected. As the w_{ROI} increases, the distortion computed for *voxels* outside the ROI remain slightly inferior to the one computed for all *voxels*. The performance does not drop as much for *voxels* outside the ROI because the ROI is relatively small. Less than 6%, of *voxels* are in the ROI. For *voxels* in the ROI, in the other hand, the performance significantly increases as w_{ROI} increases. If we look at the average difference between $PSNR_Y$ computed for *voxels* in the ROI



(a) ROI



(b) Non-ROI

Figure 4.2: Average rate-distortion curves for all frames of the sequence Longdress. The PSNR is computed only for (a) voxels in the ROI and (b) voxels outside the ROI.

Table 4.2: Average BD-PSNR and BD-rate in the range from 0.08 and 1 bpov with the PSNR computed inside and outside the ROI, compared to the PSNR of all voxels.

	BD-PSNR (dB)		**BD-Rate (%)**	
w_{ROI}	**ROI**	**non-ROI**	**ROI**	**non-ROI**
2	1.03	0.09	-11.93	0.77
4	2.47	-0.10	-36.68	4.98
8	3.97	-0.24	-53.53	8.52
16	5.53	-0.34	-64.88	11.48
32	7.16	-0.41	-70.18	14.01

and that computed for all *voxels*, we obtain values close to the expected, *i.e.* values computed as $10log_{10}\left(\sqrt{w_{ROI}}\right)$ which are 1.5051, 3.0103, 4.5154, 6.0206 and 7.5257 dB when w_{ROI} is 2, 4, 8, 16 and 32, respectively.

In the modification shown in Section 3.1, the transform minimizes the weighted squared error given by equation (3.1). The PSNR is computed as

$$PSNR(\mathcal{P}, \hat{\mathcal{P}}, \{w_i\}) = 10\ \log_{10}\left(\frac{255^2}{E_{noise}}\right) \tag{4.3}$$

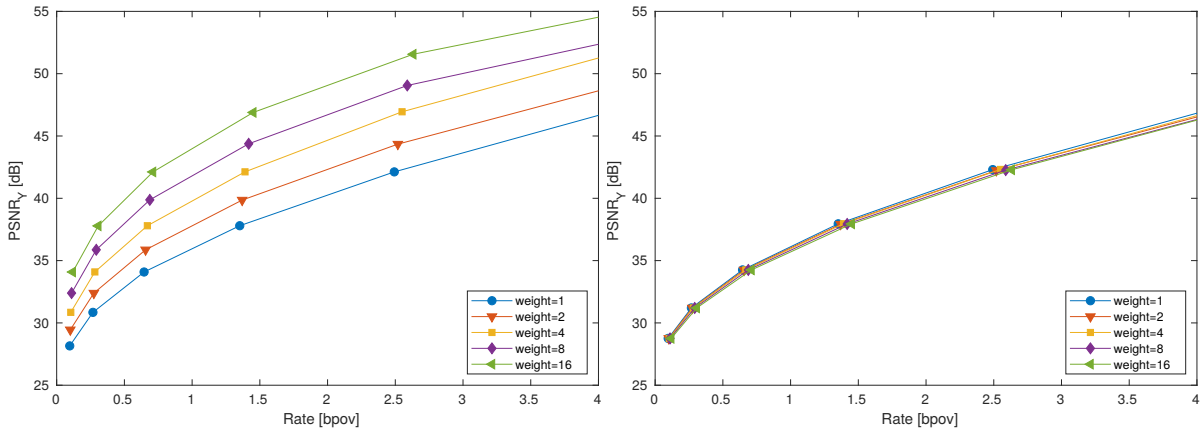
i.e. the ratio between the peak signal and the noise energy E_{noise} that is computed as the average squared error. Energy can be redefined as

$$E_{noise} = \frac{\sum_i w_i \left(Y_i - \hat{Y}_i\right)^2}{\sum_i w_i} \tag{4.4}$$

where w_i is the weight of the *voxel*, $\mathcal{P}$ is the original point cloud, $\hat{\mathcal{P}}$ is the reconstructed point cloud, Y_i is the original value and $\hat{Y}_i$ is the reconstructed value, for the Y channel, for example, to conform to equation (3.1), which implies the definition of a weighted PSNR, where the weight of each *voxel* is w_{ROI} if it is inside the ROI and 1, otherwise. Figure 4.4 compares the average rate-distortion curves for the sequence Loot. When $w_{ROI} = 1$, the weighted and the standard PSNR are equal. However, when $w_{ROI} > 1$ the weighted PSNR shows better results than the standard PSNR for the same data. This was expected since the transform is minimizing the distortion given by equation (3.1). Which means that the transform is prioritizing those *voxels* with higher weights over those with lower weights.

All results so far indicate a sizeable improvement in the ROI at the expense of a small loss to the regions outside the ROI. The subjective analysis of quality is the main motivation behind the use of ROI for compression. In Fig. 4.5, the point cloud Thaidancer was encoded with different ROI weights. Subjectively, Fig. 4.5(b) seems to have a better quality since we are likely to be more sensitive to artefacts in the face than in rest of the scene. Fig. 4.6 shows a close up of the face for the reconstructed point clouds shown in Fig. 4.5.

Figure 4.7 shows a frame of each sequence for the same scenarios shown in Fig. 4.5, being: (a) the ROI, (b)(d) where all *voxels* are equally encoded and (c)(e) where we privilege *voxels* in the ROI



(a) PSNR in the ROI

(b) PSNR outside the ROI

(c) PSNR over all *voxels*

(d) Weighted PSNR over all voxels

Figure 4.3: Average rate-distortion curves.

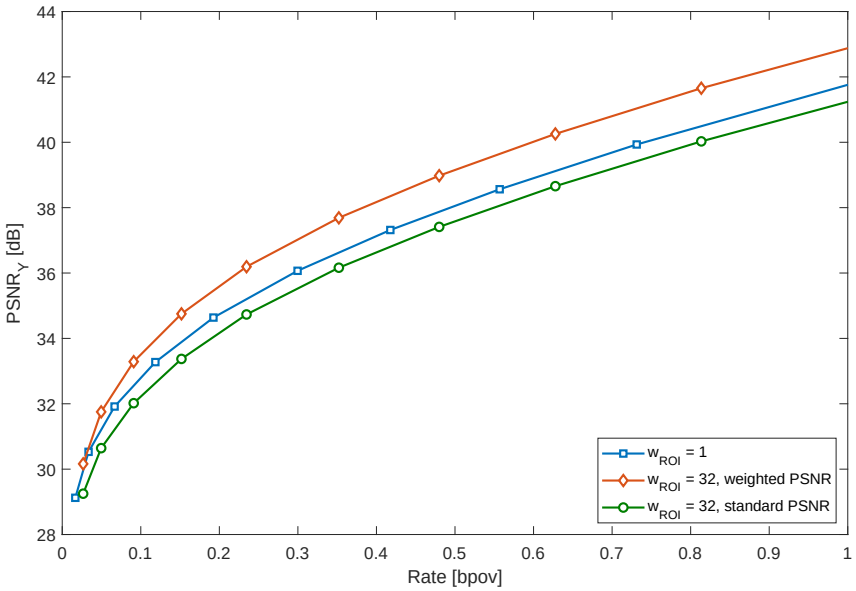


Figure 4.4: Average rate-distortion curves for the sequence "Loot" using the weighted PSNR

(face). Even though the point clouds in (b) have a higher overall PSNR, the point clouds in (c) have a better PSNR for *voxels* in the ROI and seems to have a much higher subjective quality since humans are very sensitive to face artefacts.

Some point clouds in Figure 4.7 (*e.g.* the last three) have a not-so-large PSNR gain in the ROI against the non-ROI. This is the case where there are not many color details outside the ROI, hence there are already fewer bits assigned to it. Therefore, it is more difficult to shift bits from the non-ROI to the ROI *voxels*.

(a) $w_{ROI} = 1$, $Q_{step} = 128$, file size = 11225 $bytes$.

(b) $w_{ROI} = 16$, $Q_{step} = 152$, file size = 11162 $bytes$.

Figure 4.5: Point cloud Thaidancer ($N_{vox} = 689953$) coded with different weights for *voxels* in the ROI. The Q_{step} was adjusted to result in similar file sizes.

(a) Detected ROI.

(b) $w_{ROI} = 1$

(c) $w_{ROI} = 16$

Figure 4.6: Detected region of interest and close up in the face of the reconstructed point clouds shown in Fig. 4.5

(a) (b) (c) (d) (e)

	w_{ROI} 1	16
Qstep	128	133
File Size	150006	148974
$PSNR_Y$ (overall)	25.8	25.7
$PSNR_Y$ (ROI)	28.4	34.1
Qstep	128	155
File Size	69041	70143
$PSNR_Y$ (overall)	31.0	30.7
$PSNR_Y$ (ROI)	28.7	34.2
Qstep	128	161
File Size	32149	31929
$PSNR_Y$ (overall)	31.0	30.5
$PSNR_Y$ (ROI)	28.3	33.3
Qstep	128	197
File Size	47316	47647
$PSNR_Y$ (overall)	26.3	25.0
$PSNR_Y$ (ROI)	28.2	31.9
Qstep	128	206
File Size	33192	33374
$PSNR_Y$ (overall)	25.2	24.4
$PSNR_Y$ (ROI)	29.4	32.9
Qstep	128	401
File Size	14546	14637
$PSNR_Y$ (overall)	32.5	29.9
$PSNR_Y$ (ROI)	29.0	30.0
Qstep	128	402
File Size	10675	10690
$PSNR_Y$ (overall)	32.5	30.0
$PSNR_Y$ (ROI)	29.0	30.8
Qstep	128	430
File Size	14833	14802
$PSNR_Y$ (overall)	31.8	29.8
$PSNR_Y$ (ROI)	29.1	29.8

(f)

Figure 4.7: Frames encoded with w_{ROI} equals to 1 and 16. The quantization step size was adjusted to yield similar encoded file sizes. (a) The ROI; decoded point cloud for (b) $w_{ROI} = 1$ and (c) $w_{ROI} = 16$; (d) close-up of (b); (e) close up of (c); (f) small information table with quantization step sizes, encoded file size (in bits) and $PSNR_Y$ (in dB).

4.2 SOFT-REGION-OF-INTEREST CODING

In order to test the proposed projection-based method for point cloud saliency map creation, 5 point clouds were used: Longdress, Loot, Soldier, Boxer, David, all *voxelized* with depth 10 (i.e. $1024 \times 1024 \times 1024$ *voxels*) [67]–[69].

The results are presented in Figs. 4.8 trough 4.13 as a saliency map (*i.e.* gray scale) and in a hot-cold map, where colors closer to red represent a higher saliency value and colors closer to blue are associated with a lower saliency value.

It is noticeable that, in all the examples, the most salient region contains the face, or parts of it, and, in some cases, (as in Fig 4.11 and Fig. 4.13) a region close to the face is also considered salient.

In the tests carried, the quantization step was varied from 2 to 128 and we used $L = 5$. The ROI weights ($\{\alpha_k\}$) are $\alpha_0 = 1$ and

$$\alpha_k = k\frac{\alpha}{L-1}, \tag{4.5}$$

where α is the maximum weight we apply.

Let the point cloud $\mathcal{P}$ be encoded with an encoder denoted as $COD(\mathcal{P}, S(v_i), \alpha, L)$, using saliency $S(v_i)$ and parameters α and L, the encoder uses the RAHT transform, as described in section 2.2, with RLGR [65]. The reconstructed point cloud is then

$$\hat{\mathcal{P}} = COD^{-1}(COD(\mathcal{P}, S(v_i), \alpha, L)), \tag{4.6}$$

and the rate we achieve is

$$r = RATE(COD(\mathcal{P}, S(v_i), \alpha, L)), \tag{4.7}$$

and the distortion is

$$d_k = PSNR(\mathcal{P}_k, \hat{\mathcal{P}}_k, \alpha_k). \tag{4.8}$$

For the regular encoder we have

$$\hat{\mathcal{P}}' = COD^{-1}(COD(\mathcal{P}, 1, 1, 1)), \tag{4.9}$$

where the parameters are all 1, with the rate being

$$r' = RATE(COD(\mathcal{P}, 1, 1, 1)), \tag{4.10}$$

and the calculated distortion is

$$d'_k = PSNR(\mathcal{P}_k, \hat{\mathcal{P}}'_k, 1). \tag{4.11}$$

The rate is computed as bits per occupied *voxel* (bpov) and the quality of the reconstructed point cloud by the peak signal to noise ratio of the luminance channel ($PSNR_Y$), where $\mathcal{P}_k$ is the original k-th sub-point-cloud, $\hat{\mathcal{P}}_k$ is the k-th sub-point-cloud reconstructed using the proposed method, $\hat{\mathcal{P}}'_k$ is the k-th sub-point-cloud reconstructed using the traditional (i.e. non-ROI coding) method, d_k is the distortion for the encoded k-th sub-point-cloud using the proposed method and d'_k is the

Figure 4.8: A view of point cloud "David" and its saliency and hot-cold maps.

Figure 4.9: A view of point cloud "Boxer" and its saliency and hot-cold maps.

Figure 4.10: A view of point cloud "Loot" and its saliency and hot-cold maps.

Figure 4.11: A view of point cloud "Longdress" and its saliency and hot-cold maps.

Figure 4.12: A view of point cloud "Soldier" (frame 537) and its saliency and hot-cold maps.

Figure 4.13: A view of point cloud "Soldier" (frame 695) and its saliency and hot-cold maps.

distortion for the encoded k-th sub-point-cloud using the traditional method. Both, d_k and d'_k are calculated as in equation (4.3).

In table 4.3, each cell $\gamma_{\alpha,k}$ is calculated as

$$\gamma_{\alpha,k} = BDPSNR(\{r'\}, \{d'_k\}, \{r\}, \{d_k\}) \tag{4.12}$$

where the sets are composed by one point for each quantization step used, each row in table 4.3 corresponds to an α and each column to a sub-point-cloud corresponding to the k-th saliency level.

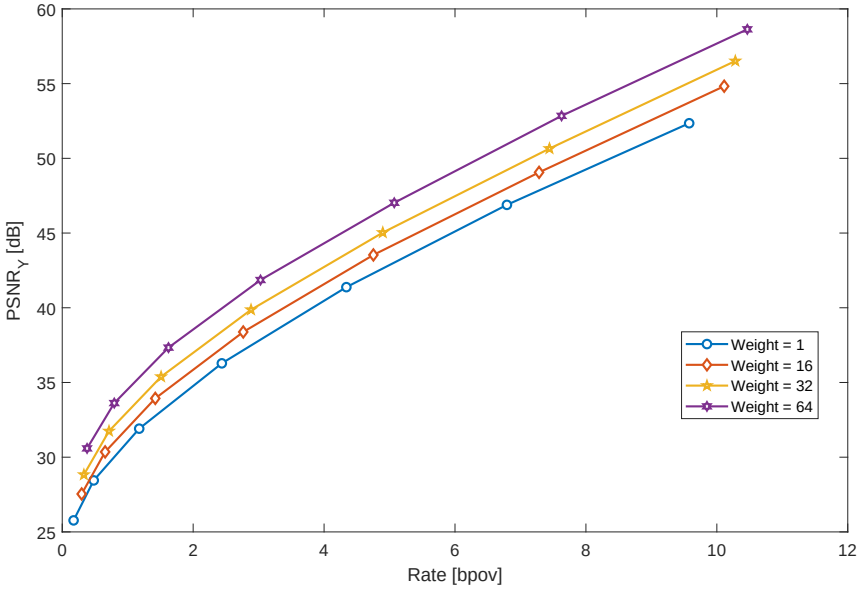
Table 4.3 summarizes the encoder performance using the saliency maps. The results present the average $PSNR_Y$ difference (BD-PSNR) [66] obtained for the point clouds tested in this part of the work, comparing those curves that prioritize the ROI ($\alpha > 1$) against the curves that equally treats all *voxels* ($\alpha = 1$). One can observe that as the α increases, the quality of reconstructed *voxels* that are completely non-salient ($S(v_i) = 0$) decreases, while the quality of those that are salient ($S(v_i) > 0$) increases at a larger rate (as in Fig. 4.14). The gain in quality is higher for higher values of $S(v_i)$, as expected (see Fig. 4.15). The higher the value of α, the more bits are spent to encode the ROI in detriment to non salient *voxels*. As there are fewer *voxels* in the ROI compared to those outside the ROI, a small decrease in the quality of the *voxels* outside the ROI results in a big increase in the quality of those inside. The number of bits spent to encode the saliency map is accounted in the overall bit rate, except when $\alpha = 1$, since there is no need to convey the saliency map to the decoder.

Table 4.3: Average BD-PSNR (dB) for all the 5 point clouds, for each $S(v_i)$ comparing the curves with $\alpha > 1$ against those when $\alpha = 1$ for all point clouds tested in this work.

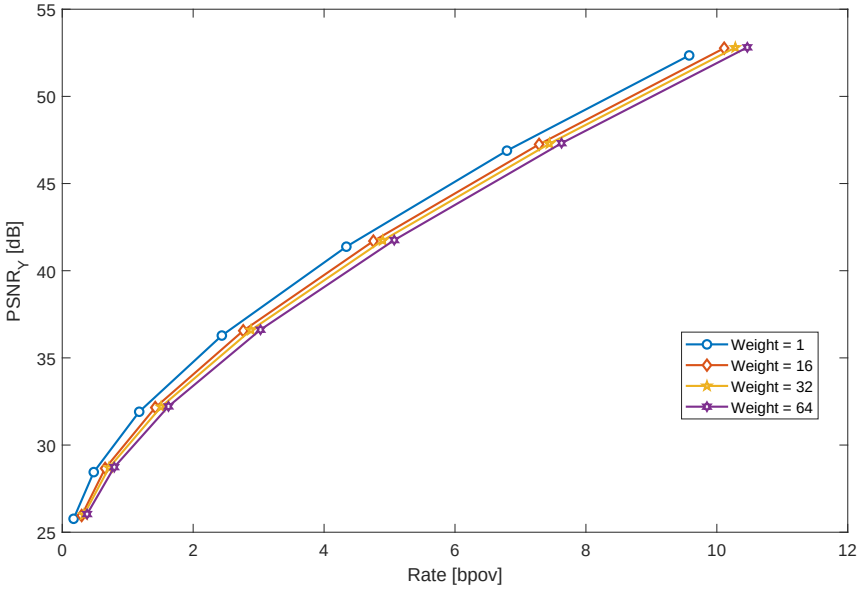
	BD-PSNR (dB)				
α	$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{P}_4$
2	-0.43	-0.48	1.35	1.16	1.20
4	-0.54	1.30	2.43	2.65	3.06
8	-0.65	2.40	3.88	4.39	5.06
16	-0.82	3.83	5.64	7.07	7.42
32	-1.04	5.60	7.82	9.61	10.28
64	-1.31	7.79	10.61	12.23	16.43

Figure 4.14(a) shows results for rate-distortion curves for the point cloud "Longdress" using the weighted $PSNR$. The weighted $PSNR$ uses the weights of each *voxel* to compute the average squared error as it was shown previously. Higher values of α produces better results in the reconstructed point cloud for the *voxels* inside the ROI, as it was shown in section 4.1 that the encoder tries to maximize the weighted $PSNR$ for a given rate.

In Fig. 4.16, the point cloud "Longdress" is encoded with different α. With $\alpha = 1$ the point cloud is encoded with a quantization step of 128. For $\alpha = 16$ the quantization step is adjusted to 212 so that both encoded files have the same bit-rate of 0.175 bpov. Subjectively, Fig. 4.16(b) seems to have a better quality. Figure 4.17 shows a close up of the salient region for the reconstructed point clouds shown in Fig. 4.16.



(a) Weighted PSNR.



(b) Standard PSNR.

Figure 4.14: Rate-distortion curves for the point cloud "Longdress" for different values of the ROI weights. The distortion is computed using the weighted PSNR in (a) and using the standard PSNR in (b).

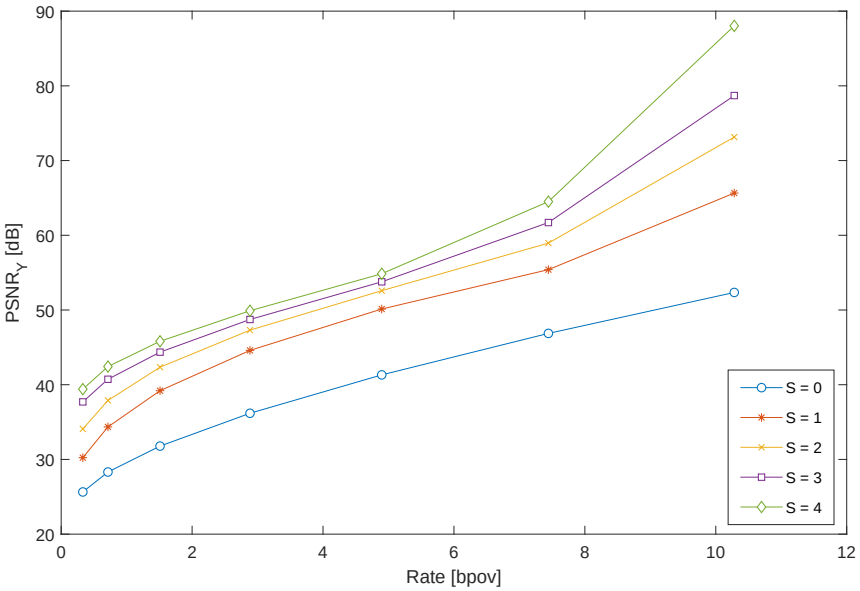


Figure 4.15: PSNR for the *voxels* inside the ROI with $\alpha = 32$ for the "Longdress" point cloud.

(a) $\alpha = 1$. (b) $\alpha = 16$.

Figure 4.16: Point cloud Longdress coded with different weights for *voxels* in the ROI. The Q_step was adjusted in order that the files would have similar sizes.

(a) $\alpha = 1$ (b) $\alpha = 16$

Figure 4.17: Close up in the face of the reconstructed point clouds shown in Fig. 4.16

5 CONCLUSIONS

The development of this project enabled the study of algorithms for the detection of regions of interest in point clouds, more specifically face detection and saliency map creation. It was proposed a framework that explores existing techniques for processing and analysis of two-dimensional images and videos, in order to perform the detection of ROI in point clouds.

There are methods proposed in the literature for identifying ROI, classifying objects in point clouds or calculating saliency maps that make use of various techniques. However, there are none that make use of two-dimensional projections to achieve the proposed objectives. Thus, the suggested method innovates and presented successful results.

The developed framework was able to process on 2D projections and to extrapolate the obtained results to 3D point clouds. It was also able to combine the processing performed from multiple views and to apply the obtained results to the correspondent *voxels*.

By modifying the distortion measure to a weighted-distortion measure, ROI coding was introduced with RAHT transform coding. However, any other point cloud coding method that optimizes the weighted distortion measure could be used. The results show an improvement in the ROI PSNR and an increase in the quality of the *voxels* inside the selective regions of higher levels of interest for the soft ROI. Higher weights results in better quality. The gain in quality was found to be, on average, 1.03, 2.47, 3.97, 5.53 and 7.16 *dB* when the weight attributed to the ROI was 2, 4, 8, 16 and 32, respectively, for the binary regions of interest. For the soft regions of interest, the average gain was 1.20, 3.06, 5.06, 7.42, 10.28 and 16.43 *dB* when the maximum weight attributed to the ROI was 2, 4, 8, 16, 32 and 64, respectively, for the *voxels* with the maximum weight.

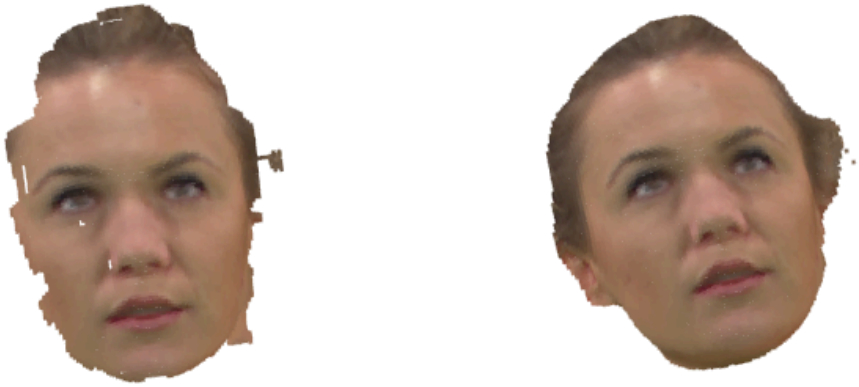
The relatively small ROI sizes allow for a large improvement of ROI *voxels* at the expense of a very small degradation to non-ROI *voxels*. The real motivation is the clearly improved subjective quality of the ROI-coded point clouds, as viewers often prefer to preserve the face quality and have a hard time noticing a slightly higher distortion in textured areas of little interest.

5.1 FUTURE WORK

Future work can include optimizing the ROI weights using perceptual studies, optimizing the side information and applying ROI coding to point cloud geometry. It may also include a way to improve the temporal consistency by using motion vectors to try to further reduce the computational cost when processing a frame sequence.

Other possible improvement to be studied is to fuse 2D processing with 3D processing, making it a hybrid framework in order to reduce the number of undetected faces. For example, this could be done by finding the nose and then generating an sphere around it to get all the *voxels* in the face,

similar to what was done in [20]. Fig. 5.1 shows a comparison of the detected faces between the method used in this work and an initial version the hybrid method.



(a) Standard projection-based method.

(b) Preliminary hybrid method.

Figure 5.1: Comparison between the projection-based method used in this work and the preliminary hybrid method.

REFERENCES

[1] P. Milgram and H. Colquhoun, "A taxonomy of real and virtual world display integration," Jan. 2001.

[2] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2019.

[3] MPEG, "MPEG121 version of MPEG standardisation roadmap," ISO/IEC JTC1/SC29/WG11 MPEG, output document N17332, Tech. Rep. N17332, January 2017.

[4] 3DG, "MPEG 3DG and Requirements: Call for proposals for point cloud compression v2," ISO/IEC MPEG JTC1/SC29/WG11, Hobart, AU, Approved WG 11 document N16763, April 2017.

[5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, Dec 2001, pp. I–I.

[6] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan 1979.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 580–587.

[8] C. Koch and S. Ullman, "Shifts in selective visual attention: Towards the underlying neural circuitry," *Matters of Intelligence: Conceptual Structures in Cognitive Neuroscience*, pp. 115–141, 1987.

[9] T. Leisti, J. Radun, T. Virtanen, R. Halonen, and G. Nyman, "Subjective experience of image quality: Attributes, definitions and decision making of subjective image quality." *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 7242, Jan. 2009.

[10] H. Hadizadeh and I. V. Bajić, "Saliency-aware video compression," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 19–33, 2014.

[11] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *preprint*, Dec. 2013.

[12] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1265–1274.

[13] A. Maity, "Improvised salient object detection and manipulation," *International Journal of Image, Graphics and Signal Processing*, vol. 8, pp. 53–60, Nov. 2015.

[14] E. Mendi and M. Milanova, "Image segmentation with active contours based on selective visual attention," May 2009.

[15] G. Sandri, V. F. Figueiredo, P. A. Chou, and R. de Queiroz, "Point cloud compression incorporating region of interest coding," in *Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, pp. 4370–4374.

[16] G. Sandri, V. F. Figueiredo, P. Chou, and R. de Queiroz, "Compressão de nuvem de pontos incorporando codificação de região de interesse," in *XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2019) (SBrT 2019)*, Petrópolis, Brazil, Sep. 2019.

[17] V. F. Figueiredo, G. L. Sandri, R. L. de Queiroz, and P. A. Chou, "Saliency maps for point clouds," in *Proceedings of the 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, Tampere, Finland, 2020, pp. 1–5.

[18] V. F. Figueiredo and R. de Queiroz, "Mapa de saliência para nuvem de pontos usando projeções," in *XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2020) (SBrT 2020)*, Florianópolis, Brazil, Nov. 2020.

[19] R. Brinkmann, *The Art and Science of Digital Compositing*. Morgan Kaufmann Publishers Inc., 1999, pp. 184–185.

[20] P. Nair and A. Cavallaro, "3D face detection, landmark localization, and registration using a point distribution model," *IEEE Transactions on Multimedia*, vol. 11, no. 4, pp. 611–623, June 2009.

[21] A. Colombo, C. Cusano, and R. Schettini, "3D face detection using curvature analysis," *Pattern Recognition*, vol. 39, pp. 444–455, March 2006.

[22] T. Fabry, D. Vandermeulen, and P. Suetens, "3D face recognition using point cloud kernel correlation," in *Proceedings of the 2008 IEEE Second International Conference on Biometrics: Theory, Applications and Systems*, Sep. 2008, pp. 1–6.

[23] Z. Wang, L. Zhang, L. Zhang, R. Li, Y. Zheng, and Z. Zhu, "A deep neural network with spatial pooling (dnnsp) for 3-d point cloud classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 8, pp. 4594–4604, Aug 2018.

[24] D. Habermann, E. Dranka, Y. Caceres, and J. B. R. do Val, "Drones and helicopters classification using point clouds features from radar," in *Proceedings of the 2018 IEEE Radar Conference (RadarConf18)*, April 2018, pp. 0246–0251.

[25] M. Zuffo, "University of São Paulo point cloud dataset." [Online]. Available: http://uspaulopc.di.ubi.pt/

[26] M. Levoy. (2014) The Stanford 3D Scanning Repository. Stanford University Computer Graphics Laboratory. Accessed: 15-10-2020. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/

[27] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*. ACM Press, 1994.

[28] G. Sandri, R. L. de Queiroz, and P. A. Chou, "Comments on "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform"." [Online]. Available: https://arxiv.org/abs/1805.09146

[29] Y. LANGSAM, M. AUGENSTEIN, and A. M. TENENBAUM, *Data Structures using C and C++*. New Jersey: Prentice Hall, 1996.

[30] D. Meagher, "Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3D objects by computer," Oct. 1980.

[31] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *Proceedings of the 5th High-Performance Graphics Conference*, ser. HPG '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 73–79. [Online]. Available: https://doi.org/10.1145/2492045.2492053

[32] G. L. Sandri, "Compression of Point Cloud Attributes," Ph.D. dissertation, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília–DF, Brazil, 2019.

[33] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE Int'l Conf. Image Processing (ICIP)*, Oct 2014.

[34] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," in *Proceedings of the IEEE Int'l Conf. Image Processing (ICIP)*, Sept 2015.

[35] ——, "Graph-based compression of dynamic 3d point cloud sequences," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1765–1778, April 2016.

[36] E. Pavez and P. A. Chou, "Dynamic polygon cloud compression," in *IEEE Int'l Conf. Acoustics, Speech and Signal Processing (ICASSP)*, March 2017.

[37] E. Pavez, P. A. Chou, R. L. de Queiroz, and A. Ortega, "Dynamic polygon cloud compression," *CoRR*, vol. abs/1610.00402, 2016.

[38] R. A. Cohen, D. Tian, and A. Vetro, "Attribute compression for sparse point clouds using graph transforms," in *IEEE Int'l Conf. Image Processing (ICIP)*, Sept 2016.

[39] P. A. Chou and R. L. de Queiroz, "Gaussian process transforms," in *Proceedings of the IEEE Int'l Conf. Image Processing (ICIP)*, Sept 2016.

[40] R. L. de Queiroz and P. A. Chou, "Transform coding for point clouds using a Gaussian process model," *IEEE Trans. Image Processing*, vol. 26, no. 8, Aug. 2017.

[41] ——, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

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

[43] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[44] T. Kadir and M. Brady, "Saliency, scale and image description," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 83–105, Nov 2001. [Online]. Available: https://doi.org/10.1023/A:1012460413855

[45] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[46] E. Niebur and C. Koch, "Control of selective visual attention: Modeling the 'where' pathway," *Neural Information Processing Systems*, vol. 8, pp. 802–808, 1996.

[47] J. Kuen, Z. Wang, and G. Wang, "Recurrent attentional networks for saliency detection," *CoRR*, vol. abs/1604.03227, 2016. [Online]. Available: http://arxiv.org/abs/1604.03227

[48] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867–1874.

[49] C. Papageorgiou, M. Oren, and T. Poggio, "General framework for object detection," vol. 6:, 02 1998, pp. 555 – 562.

[50] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision - IJCV*, vol. 57, Jan. 2001.

[51] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, pp. 119–, 08 1997.

[52] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: Data mining, inference, and prediction," *Math. Intell.*, vol. 27, pp. 83–85, Nov. 2004.

[53] T. Zheng, C. Chen, J. Yuan, B. Li, and K. Ren, "Pointcloud saliency maps," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1598–1606.

[54] M. Krivokuća, P. A. Chou, and M. Koroteev, "A volumetric approach to point cloud compression," *ArXiv e-prints*, Sep. 2018. [Online]. Available: https://arxiv.org/abs/1810.00484

[55] R. Gray and D. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[56] Z. Wen, Y. Yan, and H. Cui, "Study on segmentation of 3D human body based on point cloud data," in *Proceedings of the 2012 Second International Conference on Intelligent System Design and Engineering Application*, 2012, pp. 657–660.

[57] M. Qiao, J. Cheng, W. Bian, and D. Tao, "Biview learning for human posture segmentation from 3d points cloud," *PloS one*, vol. 9, p. e85811, 01 2014.

[58] P. Mandikal, N. K. L., and R. V. Babu, "3D-PSRNet: Part segmented 3D point cloud reconstruction from a single image," *CoRR*, vol. abs/1810.00461, 2018. [Online]. Available: http://arxiv.org/abs/1810.00461

[59] E. M. Torlig, "Métricas Objetivas Baseadas em Projeções para Avaliação de Qualidade de Nuvem de Pontos," Master's thesis, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília–DF, Brazil, 2018.

[60] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," IBM Ltd., Ottawa 4, Ontario, Canada, Tech. Rep., March 1966.

[61] D. Walther and C. Koch, "Modeling attention to salient proto-objects," *Neural Networks*, vol. 19, pp. 1395–1407, 2006.

[62] H. P. Hariharan, T. Lange, and T. Herfet, "Low complexity light field compression based on pseudo-temporal circular sequencing," in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, June 2017, pp. 1–5.

[63] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," in *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98 (Cat. No.98EX201)*, Oct. 1998, pp. 8–14.

[64] R. Khoshabeh, S. H. Chan, and T. Q. Nguyen, "Spatio-temporal consistency in video disparity estimation," in *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 885–888.

[65] H. S. Malvar, "Adaptive run-length / Golomb-Rice encoding of quantized generalized Gaussian sources with unknown statistics," in *Proceedings of the Data Compression Conference*, March 2006, pp. 23–32.

[66] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," *Visual Coding Experts Group, ITU-T Q6/16 document VCEG-M33*, April 2001.

[67] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies — a voxelized point cloud dataset," ISO/IEC JTC1/SC29/WG1 & WG11 JPEG & MPEG, input documents M74006 & m40059, Jan. 2017.

[68] M. Krivokuća, P. A. Chou, and P. Savill, "8i voxelized surface light field (8iVSLF) dataset," ISO/IEC JTC1/SC29/WG11 MPEG, input document m42914, Jul. 2018.

[69] C. Loop, Q. Cai, S. Escolano, and P. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), input document m38673/M72012, May 2016.

[70] A. G. Asuero, A. Sayago, and A. G. González, "The correlation coefficient: An overview," *Critical Reviews in Analytical Chemistry*, vol. 36, no. 1, pp. 41–59, 2006.