UNIVERSITA' DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

XIII CICLO – NUOVA SERIE

ANNO ACCADEMICO 2013-2014

TESI DI DOTTORATO IN INFORMATICA

Compression and Protection of Multidimensional Data

Tutor
prof. **Bruno Carpentieri**

Candidato
Raffaele Pizzolante

Coordinatore
prof. **Giuseppe Persiano**

Acknowledgements

I would like to deeply thank my advisor Prof. Bruno Carpentieri for its precious and friendly support and guidance during my studies and the Ph.D. programme. I would like to thank also my course chairman, Prof. Pino Persiano.

I am infinitely grateful to my dear friend Arcangelo Castiglione, which is also a colleague and which collaborated with me to most of the experiences related to this thesis. I would like to thank Aniello "Nello" Castiglione, Prof. Francesco Palmieri and Prof. Alfredo De Santis for their valuable collaboration, their suggestions, etc.. I would like to thank also my colleague Ugo Fiore for the useful hints that provided me during my Ph.D. programme.

Finally, I would like to thank ALL who have shared with me also a part of this pathway.

Contents

Chapter – 1

Introduction

Highlights of the chapter

The main purpose of this introductory chapter is to provide a brief review of the basic ideas behind compression techniques. In detail, we start from the main motivations and essence of Data Compression. Subsequently, we briefly introduce the two main categories of data compression techniques: the lossless and the lossy strategies. Finally, we outline our contribution and the organization of this Thesis.

1.1. Introduction

The main aim of data compression techniques is to reduce the space related to the representation of digital information. In particular, the compression process is the process that allows to transform an *input data stream* (a video, an image, an audio file, etc.) from one representation into another (the *compressed data stream*). The size of the compressed data stream is less than the size of the input one. From the compressed data stream, it is possible to either recover the input data stream or an approximation of the latter. The primary ability of a compression algorithm is to exploit the redundancy of the input data stream.

Despite the exponential decrease in the cost of digital storage as well as the growing interest in novel Internet-based technologies (i.e., *Cloud*, *P2P networks*, etc.), it is easy to think that compression strategies could appear to be less relevant than in the past. In fact, the simplest solution, which could appear plausible, is to store an input data stream *"as is"*, in *raw* format, without the application of any compression approaches. However, it is important to consider that even the size of the data has exponentially grown, since new or upgraded technologies have been developed. Clearly, the acquisition technologies, which produce increasingly large data, as well as the decreasing costs of the (remote and local) storage space are continuously evolving. Therefore, Data Compression is a very actual problem, when also considering, for instance, portable devices (with reduced capabilities in terms of storage spaces).

Many research teams are currently involved in the development of novel techniques, which allow to obtain better compression performances as well as compress new types of data.

There are two main compression techniques:

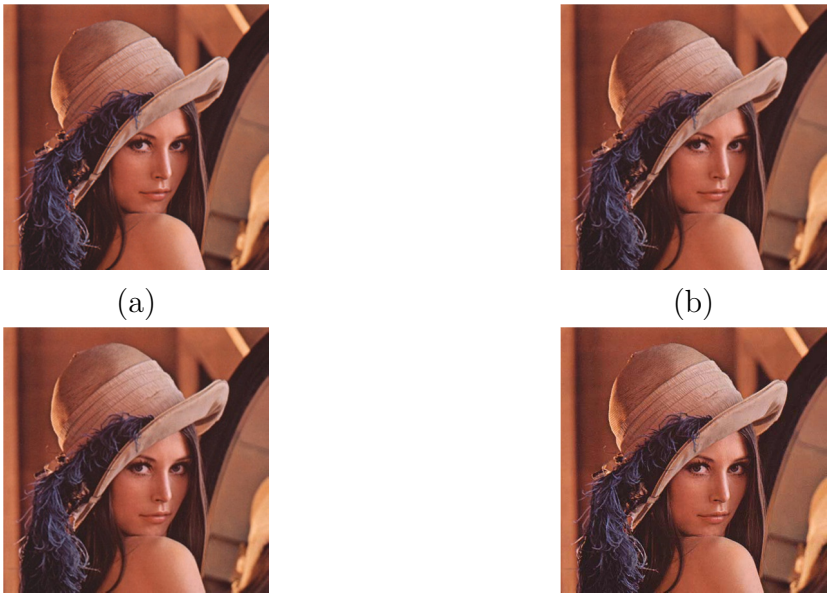
- *Lossy Compression*;
- *Lossless Compression*.

In particular, when using lossy compression techniques, is not possible to recover the original data from the compressed data through the decompression process, but only their approximation.

On the other hand, when considering lossless compression approaches, the original data can be easily extracted from the compressed data by using the decompression algorithm.

1.1.1. Lossy Compression

Lossy compression strategies are widely used for the compression of multimedia data, since data loss can be tolerated. It is important to note that the obtained approximation is *"similar"* to the original data, since only the information, which are not relevant (or not perceptible to the end-users), are not considered. Lossy compression algorithms have generally higher compression performances than lossless ones. Images, videos, audio are just some examples of multimedia data.



(a) (b)

(c) (d)

Figure 1.1: The *Lena* image: (a) uncompressed, JPEG compressed at quality of 100% (b), 50% (c) and 25% (d).

For instance, in the case of the images, one of the most used approaches, especially on the *World Wide Web*, is the *JPEG* (*Joint Photographic Experts Group*) lossy compression algorithm [**56**].

Basically, JPEG uses the *Discrete Cosine Transform* (*DCT*), which permits to convert the image from the *spatial domain* to the *frequency domain* (or *transform domain*). After the conversion, a *quantization* process is performed. Through *quantization*, the high-frequency coefficients are discarded, since this information is not relevant to the *Human Visual System* (*HVS*). The quantized coefficients are finally encoded through a lossless compression algorithm (a *Run-Length Encoding* (*RLE*) schema [**57**].

Generally, JPEG implementations permit to define the quality of the output (see Figure 1.1), which will be achieved after the decompression process. It is

important to note that, in general, outputs with higher quality need more space than outputs with lower quality.

Figure 1.1.a shows the original *"Lena"* (or *"Lenna"*) image, while Figures 1.1.b, 1.1.c and 1.1.d show the output of the decompression process, where the Lena image is compressed through the JPEG compression at 100%, 50% and 25%, respectively.

Various strategies are currently being adopted, in order to compress digital data. For example, *MPEG-1 or MPEG-2 Audio Layer III* (known as *MP3*) [**55**] and *MPEG-4* [**55**] are two of the most popular lossy compression algorithms and are widely diffused for audio and video data.

1.1.2. Lossless Compression

Lossless compression techniques are preferred in all those areas in which any information loss may compromise the value of the data. Through such approaches, the original data can be exactly restored. One of the main goals of lossless compression techniques is related to exploiting the possible redundancies.

For example, consider the following string *s*: *AAAAAXXBBBBBBBZZZZ*. It is easy to note that s has a length of 18 characters. In detail, only 4 symbols are used (i.e., 'A', 'X', 'B' and 'Z') and all of them are repeated in *s*. Supposing that each symbol of s requires 8 bits, then the required space by *s* is 144 bits.

The string *s* can be represented in a more compact form, by preceding each symbol by its number of future occurrences. Therefore, the string *s* can be then

compacted into the new representation as *5A2X7B4Z*. By supposing that an integer can be represented through 8 bits, the required space for the new string is 64 bits. Thus, in this example, the required space for the new representation is 55% less than the space of the original string s.

This idea is exploited by the *Run-Length-Encoding* (*RLE*) algorithm. RLE is a well-suited strategy for the compression of palette-based images (i.e., icons, etc.). Furthermore, the RLE scheme, coupled with other approaches, is used by fax machines, in which the documents produced are generally composed of a white background and some textual information, in black. However, the RLE algorithm is not efficient with static images.

On the other hand, lossless image compression strategies are generally based on a predictive model. A predictive-based strategy consists of two independent and distinct phases:

- *Context-modeling*;
- *Prediction residual coding*.

In the context-modeling phase, the current pixel, $x^{(0)}$, is substantially guessed in a deterministic manner, by considering a subset of previous coded pixels (the *prediction context*). The result of this phase is the predicted pixel, $\hat{x}^{(0)}$.

The *prediction residual* (or *prediction error*), $e^{(0)}$, related to the current pixel, is modeled and encoded by sending it to an entropy coder. It is important to note that $e^{(0)}$ is obtained by means of the equation (1.1).

$$e_{x^{(0)}} = \lfloor x^{(0)} - \hat{x}^{(0)} \rfloor \tag{1.1}$$

Once computed, the prediction error is sent to an entropy or statistical encoder.

The prediction phase plays the main role, since it is delegated to exploiting all the redundancy among the pixels.

1.2. Our Contribution and the Organization of the Thesis

The main goal of this dissertation is to understand and introduce novel approaches for the compression and protection of multidimensional data.

In Chapter 2, first the formal structure of the multidimensional data (Section 2.1) is described, with some synthesized examples: 3-D medical images (Section 2.2), hyperspectral images (Section 2.3), 3-D microscopy images (Section 2.4) and 5-D functional Magnetic Resonance Images (fMRI – Section 2.5).

In Chapter 3, the focus is on the delicate task related to the compression of 3-D medical images. In this case, we review a novel approach, introduced in [**20**] and denoted as *Medical Images Lossless Compression* algorithm(*MILC*). MILC is a lossless compression algorithm, which is based on the predictive model and is characterized to provide a good trade-off between the compression performances and reduced usage of the hardware resources. The results achieved by the MILC approach are comparable with other approaches in the current state-of-art. In addition, the MILC algorithm is suitable for implementations on hardware with limited resources (Section 3.1). It is important to note that in the medical and medical-related fields, the *execution speed* of an algorithm, could be a "critical" parameter. Starting from this

consideration, we review a redesigned and parallelized implementation of the compression strategy of the MILC algorithm, which is referred to as *Parallel MILC* (Section 3.2). In detail, Parallel MILC, introduced in [**47**] exploits the capabilities of the *Parallel Computing* and can be executed on heterogeneous devices (i.e., CPUs, GPUs, etc.). The achieved results, in terms of *speedup*, obtained by comparing the execution speed of Parallel MILC with respect to the MILC, are significant. In addition, the design choices related to the compression strategy of Parallel MILC allow to use the same decompression strategy of MILC. It is therefore possible to compress a 3-D medical image by using the MILC or Parallel MILC algorithms as well as the decompress the coded stream, using the same strategy for both.

In Chapter 4, we consider the important aspect of the protection of two sensitive types of multidimensional data: 3-D medical images and 3-D microscopy images. First, a hybrid approach is reviewed, introduced in [**37**], allowing for the efficient compression of 3-D medical images as well as the embedding of a *digital watermark* (see Section 4.1 for more details on the digital watermark), at the same time as the compression (Section 4.2). Subsequently, we focus on the protection of 3-D microscopy images (Section 4.3). 3-D microscopy images are extremely sensitive, since they can be used in different and delicate contexts (i.e., forensic analysis, chemical studies, etc.). In detail, we review a novel watermarking scheme that allows for the simultaneously embedding of two watermarks, in order to protect the data. It is important to emphasize that, to the best of our knowledge, our approach,

presented in [**8**], is the first that addresses the protection of 3-D microscopy images.

In Chapter 5, we review a predictive structure that can be used for the compression of different types of multidimensional data [**44**]. In detail, we use our predictive structure for the lossless compression of multidimensional data. We successfully carry out our experiments on different datasets of 3-D medical images, hyperspectral images and 5-D fMRI images, which are publicly available. The experimental results show that our approach obtains results in line with, and often better, with respect to other current state-of-art approaches, in the case of both 3-D medical and hyperspectral images. On the other hand, to the best of our knowledge, there are no existing approaches, which are tested on the two datasets we used, in relation to the two 5-D fMRI datasets.

Finally, we draw our conclusions and the future research perspectives, by explaining possible future directions for each one of the discussed techniques.

The description of all the used datasets is provided in Appendix A, by reporting only the useful details for the purpose of the techniques discussed.

Chapter – 2

Multidimensional Data

Highlights of the chapter

During the last decades, digital data are rapidly diffused and used in wide range of areas, ranging from industrial and research contexts to medical applications, etc.. This chapter focusses on the description of multidimensional digital data, which are constituted by a N-dimensional collection of 2-D components. Such components can be images, data matrices, etc..

3-D medical images, 3-D microscopy images, hyperspectral images, and 5-D functional Magnetic Resonance Images are some examples of multidimensional data, which are briefly outlined in this chapter.

It is important to point out that such data need a large amount of memory space for their storage as well as a significant amount of time to be transmitted. In general, multidimensional data are sensitive, expensive and precious.

2.1. Introduction

Informally speaking, we can define a multidimensional dataset as a N-dimensional (with $N \geq 3$) collection of highly-related bi-dimensional *components* [**1**]. It is important to observe that a component can be an image, a data matrix, etc.. In detail, all of these bi-dimensional components have the same size. Formally, we can describe the size of a multidimensional (N-D) dataset by means of Definition 2.1.

Definition 2.1 (*Size of a Multidimensional Dataset*). $< D_1, D_2, ..., D_{N-2}, X, Y >$ is a sequence of integers that is used to define the size of a N-D dataset, where D_k indicates the size of the k-th dimension ($1 \leq k \leq N-2$), X and Y indicate respectively the width and the height of each bi-dimensional component. □

The atomic elements of a multidimensional dataset are the *samples*, which are the elements that compose a component. For example, a sample can be a pixel of an image, an element of a matrix, etc..

Therefore, by considering a scenario in which we have a N-D dataset of size $< D_1, D_2, ..., D_{N-2}, X, Y >$, we can easily observe that each component is composed by $X \times Y$ samples and the whole dataset contains $D_1 \times D_2 \times ... \times D_{N-2} \times X \times Y$ samples.

In particular, it is possible to identify a sample, through its coordinates, and a component, through a vector of $N-2$ elements. In detail, Definition 2.2

and Definition 2.3 define the formal notations we used for the unambiguous identification of a sample and the unambiguous identification of a component, respectively.

Definition 2.2 (*Sample Identification*). $(d_1, d_2, \dots, d_{N-2}, x, y)$ (where $1 \leq d_i \leq D_i$, $1 \leq x \leq X$, $1 \leq y \leq Y$ and $1 \leq i \leq N-2$) are integer coordinates that unequivocally identify a sample, in a N-D dataset of dimensions $< D_1, D_2, \dots, D_{N-2}, X, Y >$. □

Definition 2.3 (*Component Identification*). $[c_1, c_2, \dots, c_{N-2}]$ (where $c_i \in \{1, 2, \dots, D_i\}$ and $1 \leq i \leq N-2$) is a vector that univocally identifies a component in a N-D dataset of dimensions $< D_1, D_2, \dots, D_{N-2}, X, Y >$. □

In the following subsections, we briefly discuss various types of multidimensional data: 3-D Medical Images (Section 2.2), 3-D Microscopy images (Section 2.3), Multispectral and hyperspectral data (Section 2.4) and 5-D functional Magnetic Resonance Images (Section 2.5).

2.2. 3-D Medical Images

Nowadays, medical digital imaging techniques are continuously evolving and most research focusses on the improvement of such techniques, in order to obtain greater acquisition accuracy. It is important to point out that thanks to

the widespread diffusion of inter-connections, new services are provided to medical staff. For examples, the exchange of medical data among different entities/structures connected by networks (e.g. trough *Internet*, *Clouds services*, *P2P networks*, etc.), *telemedicine*, *tele-radiology*, *real-time tele-consultation*, *PACS* (Picture Archiving and Communication Systems), etc..

In such scenarios, one of the main disadvantages is related to the significant amount of storage space required as well as time for the transmission.

It should be noted that such costs are growing proportionally to the size of the data (i.e. images, etc.). It is important to emphasize that the future expectations in medical applications will increase the requests for memory space and/or transmission time.

Different medical imaging methodologies produce multidimensional data. For instance, *Computed Tomography* (*CT*) and *Magnetic Resonance* (*MR*) imaging technologies, which produce three-dimensional data ($N = 3$).

In detail, a 3-D CT image is acquired by means of X-rays, in order to obtain many radiological images. The overall acquisition process is supported by a computer, which is able to obtain different cross-sectional views. 3-D CT images are an important tool for the identification of normal or abnormal structures of the human body. It is important to emphasize that an X-ray scanner allows for the generation of different images, by considering different angles around the body part, which is undergoing analysis. Once processed by the computer, the output is a collection of the cross-sectional images, often referred to as slices.

3-D MR images are an important source of information in different medical applications and, especially, in medical diagnosis (ranging from neuroimaging to oncology). It is important to note that MR images are preferred in most cases. In fact, in the case where both CT and MR images produce the same clinical information, the latter are preferred, since MR acquisitions do not use any ionizing radiation. On the other hand, in presence of subjects with cardiac pacemakers and/or metallic foreign bodies, MR techniques cannot be used.

Figures 2.1 and 2.2 show five slices, respectively, of a 3-D CT (*"CT_carotid"*) image and a 3-D MR image (*"MR_sag_head"*).

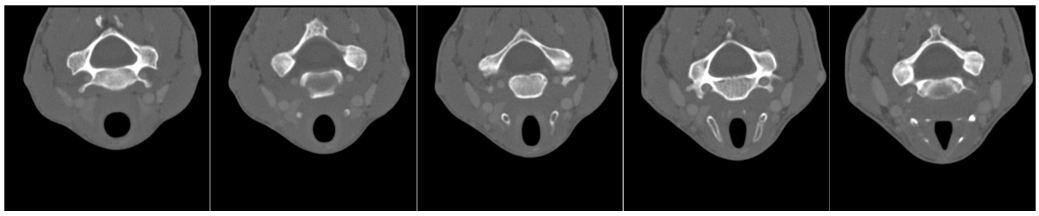


Figure 2.1: Graphical representation of five slices of a CT image.

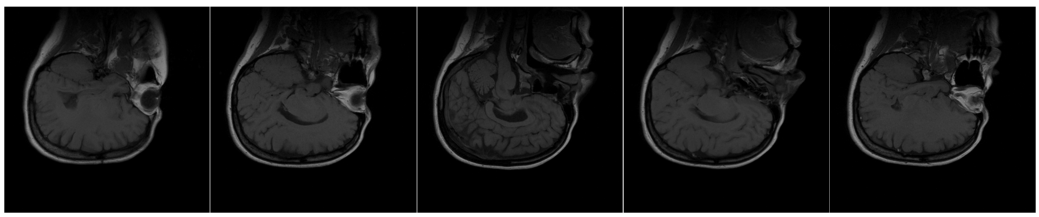


Figure 2.2: Graphical representation of five slices of a MR image.

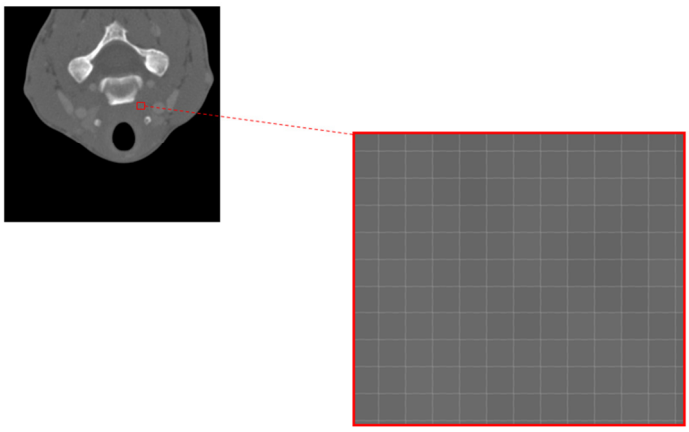


Figure 2.3: A zoomed portion of a slice of a CT image.

Starting from the consideration that medical data need to be managed in an efficient and effective manner, it is clear that data compression techniques are essential, in order to improve the transmission and storage aspects. Basically, due to the importance of such data, the choice of lossless compression strategies is often required and, in many situations, indispensable. In fact, the acquired data are precious or often obtained by means of unrepeatable medical exams. Lossy compression techniques could be considered, but it is necessary take into account that that the lost information due to such methods, might lead to either an incorrect diagnosis or it could affect the reanalysis of data, with future techniques.

It is important to emphasize that 3-D medical images present substantially two types of correlation: *intra-slice correlation* and *inter-slice correlation*.

In particular, it is worth noting that adjacent samples are generally related to the same tissue and may have similar intensity (intra-slice correlation). Figure 2.3 shows a zoomed portion of a slice (of the *"CT_carotid"* image), outlined

with a red border, in which it is possible to observe the similarity of the intensity of the samples. In addition, consecutive slices of a 3-D medical image are generally related (inter-slice correlation).

In Figure 2.4, the Pearson's Correlation [**36**] coefficients for a CT image (*"CT_wrist"*) are graphically reported. In detail, the red color indicates a high correlation, while the violet indicates a low correlation. Furthermore, the slices are indicated on the X and Y-axis. In particular, the color assumed by each point is related to the correlation value, between the slice on the X-axis and the slice on the Y-axis. Thus, it is possible to note that the graph is symmetric on the secondary diagonal. The graph in Figure 2.4, highlights that the consecutive slices are strongly related, observing the area around the secondary diagonal, which is completely red.

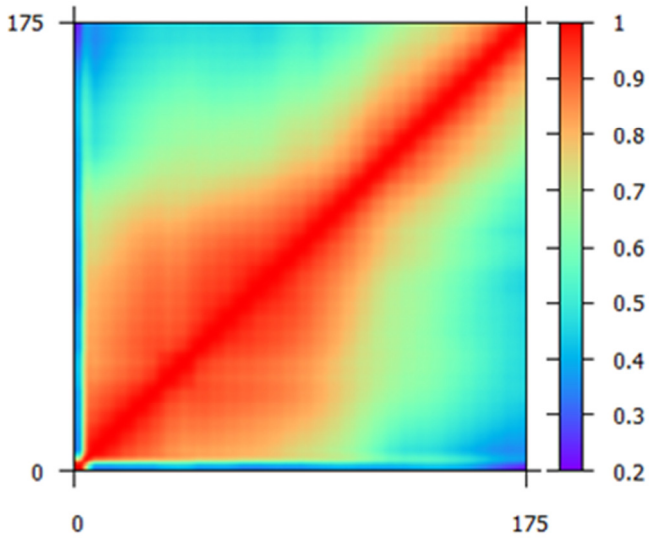


Figure 2.4: A graphical representation of the matrix of the correlation of a CT image.

2.3. 3-D Microscopy Images

The microscope is an essential scientific tool and is extremely used in many fields and for various purposes.

In particular, it is widely used in *chemistry*, especially for the analysis of polymer and plastics, catalyst evaluation and testing, etc.. In the *industrial field* such images are precious and helpful for the designing and development of nanomaterials and biotechnology as well as the analysis of fibres, fabrics and textiles. In addition, such technology plays an important role in *medical* and *biological* fields, in which such data is used in different types of analysis concerning pathologies (*histopathology*, *cytopathology*, *phytopathology*). In *forensic science*, microscopy images are used to study specimens, which are generally acquired at the crime scene.

It is important to highlight that several microscopy imaging techniques commonly produce digital data, namely, images or image sequences that can be processed later. For instance, *confocal microscopy* is an optical imaging technique that can be essential for the study of different structures, since it is possible to obtain their three-dimensional (3-D) representation.

In particular, through conventional *Laser Scanning Confocal Microscope* (*LSCM*) a high intensity source of light is used: a laser. Such a laser is able to excite the molecules of the evaluated sample [**5**]. In addition, the laser light is reflected through a *dichroic mirror*. The reflected light is then directed towards two mirrors, which can rotate.

More precisely, the fluorescent sample is excited by the light, by traversing the microscope objective. In detail, the excited light emitted by the sample passes back through the microscope objective and is then *de-scanned* by the two previously used mirrors.

Subsequently, the emitted light traverses the dichroic mirror onto a pinhole, which is placed onto a conjugate focal plane of the sample under analysis. For these reasons, this kind of microscope is known as *confocal* (*conjugate focal*). Finally, the light reaches a *photomultiplier tube* (a *detector*), which is able to convert the measured light into electronic signals. This detector is connected to a computer, which is able to create the final image by considering one pixel at time. In detail, a confocal microscope is not able to have the complete image of the sample. In fact, only one point is observed at any time. One of the main advantages that a confocal microscope presents is related to its ability in rejecting out-of-focus fluorescent light [**5, 6**], by means of the confocal pinhole, which prevents the out-of-focus light reaching the detector.

2.4. Multispectral and Hyperspectral Data

The main aim of multispectral and hyperspectral imaging is to collect information from a scene through the exploration of the electromagnetic spectrum. Differently to the human eye and traditional camera sensors, which can only perceive visible light, spectral imaging techniques allow to cover a significant portion of wavelengths. In particular, the spectrum is subdivided into different spectral bands.

Thus, it is possible to identify and/or classify materials, objects, etc.. These capabilities are related to the fact that some objects have a unique signature (a sort of *fingerprint*) in the electromagnetic spectrum, which can be employed for identification purposes.

Multispectral and hyperspectral data, produced by airborne and spaceborne remote sensing acquisitions, play an important role in a large and growing range of real-life applications. In particular, such data are used in different fields, varying from environmental studies to mineralogy, from astronomy to physics, etc..

For instance, in some geographical areas, the monitoring of the Earth's surface is provided through the hyperspectral remote sensing technologies coupled with other technologies. Hyperspectral scanning methodologies are also employed in military applications. In particular, for aerial surveillance purposes such types of information can be helpful in many scenarios. Recently, by means of such imaging techniques, it is possible to improve the accuracy in food processing activities.

The multispectral remote sensors are characterized for their capability to acquire information from few spectral bands: from 4 to about 30, for example LANDSAT [**2**], MODIS [**3**]. Instead, the hyperspectral sensors are able to acquire information up to few hundreds of bands. It should be noted that hyperspectral sensors (e.g. AVIRIS, Hyperion, etc. allow for the measurement of narrow and contiguous wavelength bands.

It is important to emphasize that the spectral resolution (i.e. the width of a measured spectral band), is generally one of the most important parameters to

evaluate the precision of a sensor. Nevertheless, also the spatial resolution is a significant aspect and needs to be considered. Informally speaking, the spatial resolution indicates how extensive the geographical area mapped by the sensor into a pixel is. It is also worth noting that it could be difficult to recognize materials from a pixel, if a too wide an area is mapped into it.

Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) hyperspectral sensors, developed by NASA Jet Propulsion Laboratory [**4**], for instance, make it possible to measure from 380 to 2500 nanometers (nm) of the electromagnetic spectrum. In detail, the spectrum is segmented into 224 spectral bands, where each one has a width of about 10 nm.

Figure 2.5 shows an RGB graphical representation of an AVIRIS hyperspectral image (*"Lunar Lake – Scene 03"*), in which the 140-th band is associated to the red component, the 65-th band is associated to the green component and the 28-th band is associated to the blue component. Whereas, Figures 2.6.a, 2.6.b, 2.6.c and 2.6.d show a graphical representation respectively of the 30-th, 100-th, 150-th and 200-th band, of an AVIRIS image ("*Lunar Lake – Scene 03*").

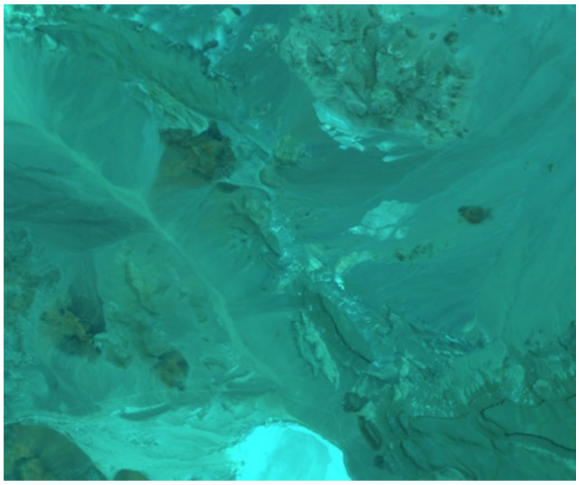
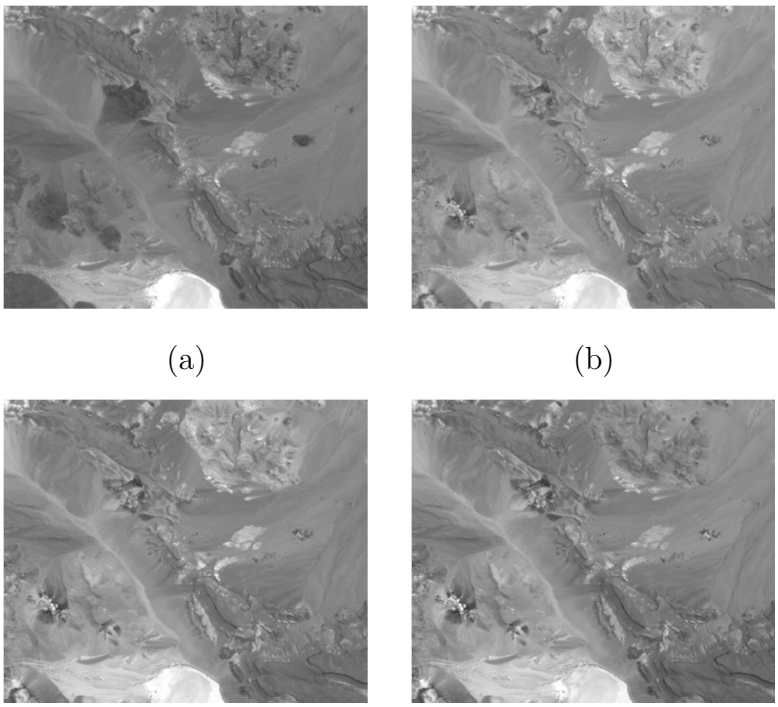


Figure 2.5: A RGB graphical representation of an AVIRIS hyperspectral image.



(a) (b)

(c) (d)

Figure 2.6: A graphical representation respectively of the 30-th (a), 100-th (b), 150-th (c) and 200-th (d) band of an hyperspectral image.

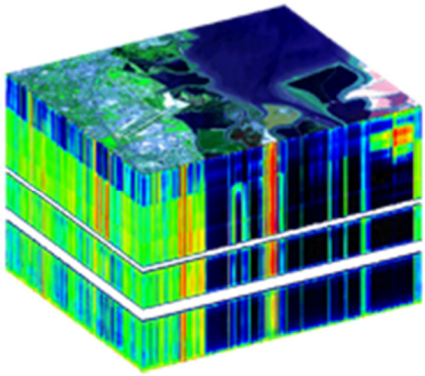


Figure 2.7: A false-color graphical representation of an AVIRIS datacube.

It should be observed that the result of a hyperspectral acquisition is a $3-D$ data collection (often denoted as a *datacube*). In particular, the output is a multidimensional data (where $N = 3$), which is constituted by the collection of bi-dimensional images. It is important to highlight that each image is related to each measured spectral band. In Figure 2.7, a graphical representation (in *false-color*) of an AVIRIS datacube is shown.

Hyperspectral datacubes need a large amount of memory space in order to be stored and transmitted. Due to these implicit costs, in many scenarios, only a subset of bands can be stored directly "on board" and successively analyzed, by limiting the potentiality of hyperspectral remote sensing. For such reasons, efficient data compression techniques are essential, thus allowing for efficient storing and transmission.

Such 3-D data present significant redundancies, which can be exploited by th compression algorithms. In particular, there are two types of correlation: *intra-band correlation* and *inter-band correlation*. Basically, adjacent pixels are associated to adjacent geographical areas, which are, generally, constituted of

the same materials. Thus, adjacent pixel values are effectively correlated in the space (intra-band correlation). Similarly, it is observable that consecutive spectral bands show a high correlation (inter-band correlation). In Figure 2.8, a graph of the Pearson's correlation among all the bands of an AVIRIS hyperspectral image is shown. In detail, on the Y-axis the value assumed by the Pearson's correlation, obtained by considering the i-th and the $(i-1)$-th bands (the bands are reported on the X-axis), is reported.

It is noticeable that the correlation assumes high values (around 0.999) in most of the cases. Only a subset of bands, which are affected by noise, present low correlation values.

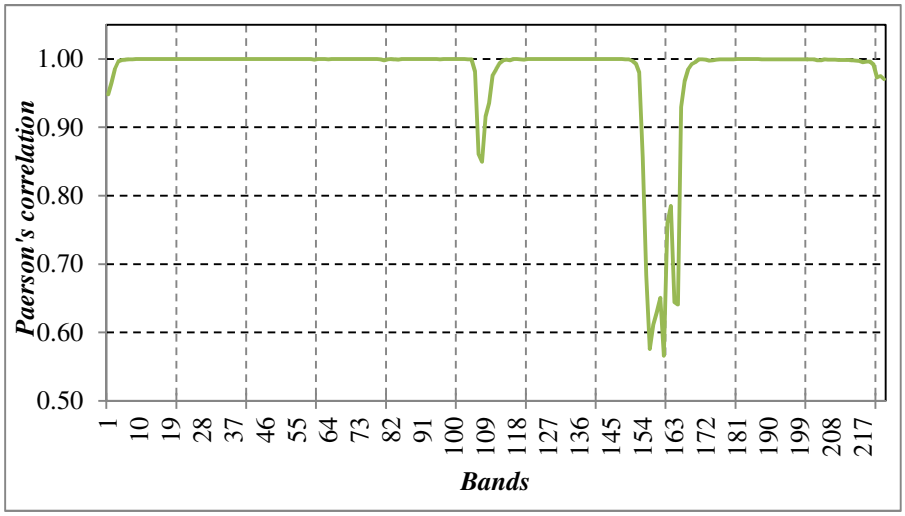


Figure 2.8: The trend of the Pearson's correlation among the spectral band of an AVIRIS datacube (the *"Lunar Lake scene 03"* image).

2.5. 5-D Functional Magnetic Resonance Imaging (fMRI)

Over recent years, many methods have been proposed with the main objective being to investigate the functioning of the human brain. Mainly, the research activities have focused on specific brain regional and functional features. One of the main objectives is to identify and, consequently, evaluate the distribution of the neural activities in the brain as a whole at a given moment. *Functional Magnetic Resonance Imaging* (*functional MRI* or *fMRI*) allows to measure the hemodynamic response (change in blood flow), related to neural activity in the brain. In detail, through fMRI techniques, it is possible to observe the neuronal activities, characterized by neuroactivation task, which need metabolic oxygen support. It is important to point out that a fMRI scanner is a type of specialized MRI scanner.

The fMRI technique is a fundamental tool for assessing the neurological status and neurosurgical risk of a patient and, through its capabilities, the brain anatomical imaging is extended. Furthermore, by analyzing these data, it is possible to determine the regions of the brain that are activated by a particular task. In particular, a fMRI scanner is able to map different structures and specific functions of the human brain.

Different clinical applications use fMRI for the localization of brain functions, the searching of markers of pathological states, etc.. In particular, fMRI techniques are helpful in identifying preclinical expressions of diseases and developing new treatments for them [**7**].

Consequently, a fMRI scanner produces a dataset, which is composed of a collection of 3-D data volumes (T dimension). Each volume is substantially a collection (on the Z dimension) of bi-dimensional images (X and Y dimensions). In general, multiple trials of observation are performed (R dimension), in order to improve the accuracy of the examination. It is evident that such data can be viewed a multidimensional data (where N can be equal to 4 or equal to 5).

Chapter – 3

Low-Complexity Lossless Compression of 3-D Medical Images

Highlights of the chapter

In this chapter, we describe a predictive-based approach for lossless compression of 3-D medical images (such as 3-D MR, 3-D CT, etc.). In detail, our method is characterized by the low usage of computational resources and the parsimonious usage of memory. Furthermore, it is easily implementable and provides a good trade-off between computational complexity and compression performances.

By considering the medical and medical-related contexts, in which the time could be a critical parameter, we focus on the improvement of the execution time of our approach. In particular, we exploit the capability of the parallel computing, in order to design a parallelized version of the compression strategy of the proposed method.

3.1. Lossless and Low-Complexity Compression of 3-D Medical Images

In different medicine and healthcare scenarios, it could be relevant to consider the efficiency of an algorithm in terms of execution time, which is involved in the analysis and/or processing of medical data. In particular, this aspect might be fundamental in many medical applications. Generally, algorithms with a low usage of computational resources are efficient in terms of execution time. In the case of lossless compression algorithms, it is important to take into account the trade-off between the compression performances and the execution time/computational complexity.

Considering the delicate medical circumstances, during the design phases of a compression scheme, it is essential to take into account and examine, which strategy, between lossless and lossy, could be employed. In particular, the lossy compression strategies are used only in a few cases, while lossless compression techniques are generally preferred. In fact, starting from the coded data, it is possible to get back the original data through lossless strategies.

In this section, we describe a lossless compression technique for 3-D medical images, we referred to it as *Medical Images Lossless Compression* algorithm (*MILC*). In detail, MILC uses limited resources in terms of computational power and memory.

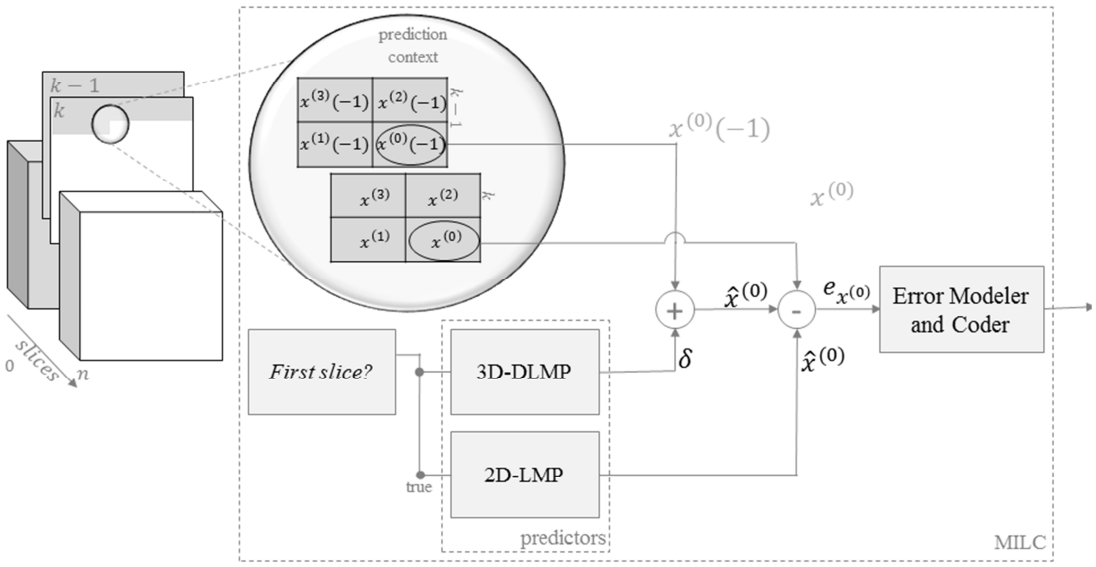


Figure 3.1: The MILC diagram block (adapted from [**20**]).

Figure 3.1 shows the MILC diagram block, with our approach being based on the predictive model and using two predictors:

- *2-D Linearized Median Predictor* (*2D-LMP*);
- *3-D Distances-based Linearized Median Predictor (3D-DLMP).*

In particular, MILC processes each sample, $x^{(0)}$, of the input 3-D medical image. It should be noted that the 2D-LMP predictor (described in Section 3.1.1) is used only for the first slice, which has no previous reference slices. Thus, the 2D-LMP exploits only the intra-slice correlation.

On the other hand, the 3D-DLMP predictor (explained in Section 3.1.2) is used for all the samples of all the slices (except for the first). It is important to point out that both the intra-slice and inter-slice correlations are exploited by the 3D-DLMP.

Finally, once a sample is predicted, the related prediction error is computed by calculating the difference between the sample, $x^{(0)}$, and the predicted

sample, $\widehat{x}^{(0)}$. Then, the prediction error is modeled and coded, as explained in Section 3.1.3.

3.1.1. The 2-D Linearized Median Predictor (2D-LMP)

The 2D-LMP predictive structure uses a prediction context composed only by samples, which belong to the same slice of the current sample, $x^{(0)}$.

In particular, three neighboring samples of $x^{(0)}$ are used, namely, $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$. Figure 3.2 shows a graphical example of the prediction context used by the 2D-LMP. It should be noted that the light blue samples are already processed and coded.

It is important to point out that the prediction of $x^{(0)}$ is obtained by means of the equation (3.1).

$$\widehat{x}^{(0)} = \frac{2\left(x^{(1)} + x^{(2)}\right)}{3} - \frac{x^{(3)}}{3} \tag{3.1}$$

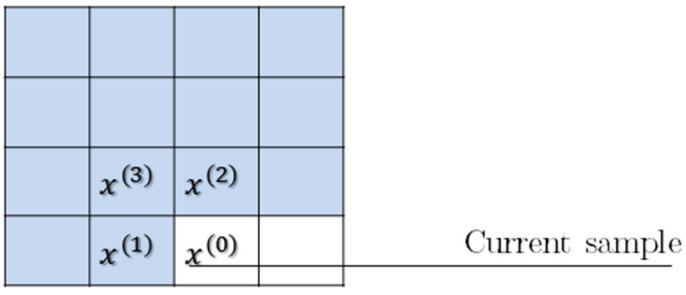


Figure 3.2: A graphical example of the prediction context used by the 2D-LMP predictor.

In detail, this predictive structure is derived from the well-known predictive structure of the Median Predictor [**24, 38**], used in the JPEG-LS algorithm.

$$\hat{x}^{(0)} = \begin{cases} \min\left(x^{(1)}, x^{(2)}\right) & if\ x^{(3)} \geq \max\left(x^{(1)}, x^{(2)}\right) \\ \max\left(x^{(1)}, x^{(2)}\right) & if\ x^{(3)} \leq \min\left(x^{(1)}, x^{(2)}\right) \\ x^{(1)} + x^{(2)} - x^{(3)} & otherwise \end{cases} \tag{3.2}$$

As may be observed from the equation (3.2), which reports the predictive structure of the Median Predictor, one of three possible options is used by the latter predictor for the computing of the prediction. The capability of the 2D-LMP predictor is to combine all of these three options, as explained by equation (3.3).

$$\begin{aligned} \hat{x}^{(0)} &= \frac{1}{3}\left(\max\left(x^{(1)}, x^{(2)}\right) + \min\left(x^{(1)}, x^{(2)}\right) + \left(x^{(1)} + x^{(2)} - x^{(3)}\right)\right) = \\ &= \frac{1}{3}\left(\left(x^{(1)} + x^{(2)}\right) + \left(x^{(1)} + x^{(2)} - x^{(3)}\right)\right) = \\ &= \frac{1}{3}\left(2\ \left(x^{(1)} + x^{(2)}\right) - x^{(3)}\right) = \\ &= \frac{2\left(x^{(1)} + x^{(2)}\right)}{3} - \frac{x^{(3)}}{3} \end{aligned} \tag{3.3}$$

It is noticeable that the third line of the equation (3.3) is obtained because $\min\left(x^{(1)}, x^{(2)}\right) + \max\left(x^{(1)}, x^{(2)}\right) = x^{(1)} + x^{(2)}$.

3.1.2. The 3-D Distances-based Linearized Median Predictor (3D-DLMP)

The 3D-DLMP predictor exploits the inter-slice and intra-slice correlations. In particular, the used prediction context is composed by neighbors of the current sample, $x^{(0)}$, in the current slice as well as the previous slice. In detail, three adjacent samples of the current sample in the same slice (i.e., $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$) and three adjacent samples in the previous slice (i.e., $x^{(1)}(-1)$, $x^{(2)}(-1)$ and $x^{(3)}(-1)$), compose the prediction context. Moreover, also the sample, with the same spatial coordinates of the current sample of the previous slice ($x^{(0)}(-1)$), is used.

Figure 3.3 shows a graphical representation of the prediction context, used by the 3D-DLMP. It is important to note that the light blue samples are already coded.

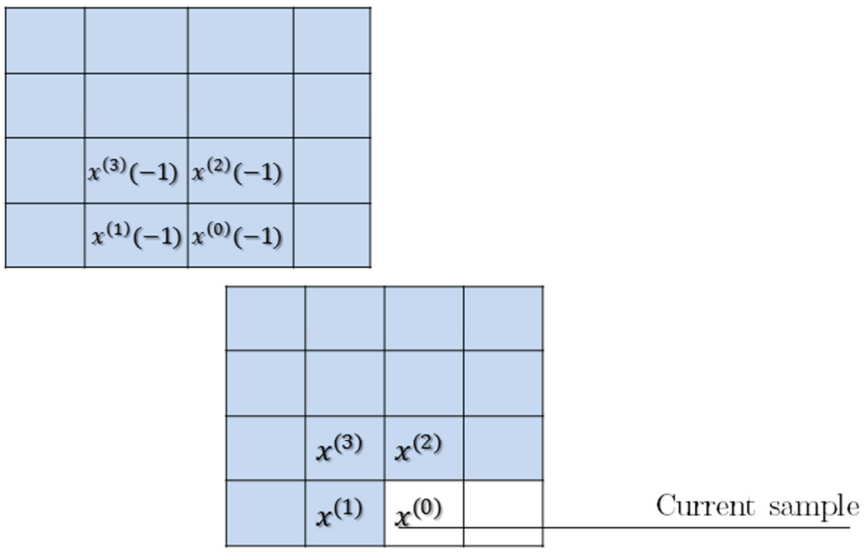


Figure 3.3: A graphical example of the prediction context used by the 3D-DLMP predictor.

Basically, three main steps are needed to implement the 3D-DLMP:

1) Computing of the distances;
2) Computing of a sort of *average distance*;
3) Computing of the prediction.

In step (1), the distances between the samples of the current slice and the samples of the previous slice are computed, by means of the equations (3.4), (3.5) and (3.6).

$$\delta_{x^{(1)}} = x^{(1)} - x^{(1)}(-1) \tag{3.4}$$

$$\delta_{x^{(2)}} = x^{(2)} - x^{(2)}(-1) \tag{3.5}$$

$$\delta_{x^{(3)}} = x^{(3)} - x^{(3)}(-1) \tag{3.6}$$

These differences are used in step (2), in which a sort of *"average distance"*, we denoted as δ, is obtained. The equation (3.7) defines how δ is computed.

$$\delta = \frac{2(\delta_{x^{(1)}} + \delta_{x^{(2)}})}{3} - \frac{\delta_{x^{(3)}}}{3} \tag{3.7}$$

In step (3), the prediction of the current sample is performed, as explained in the equation (3.8). In detail, the above computed distance, δ, is added to the sample $x^{(0)}(-1)$.

$$\hat{x}^{(0)} = x^{(0)}(-1) + \delta \tag{3.8}$$

It is easy to note that the 3D-DLMP predictor is based on the 2D-LMP predictive structure, which is described in the previous subsection. Moreover, the 3D-DLMP predictor can be further optimized, in terms of the number of operations related to the prediction of a sample.

In equation (3.9), we report the optimized predictive structure, in which only 1 division and 7 additions/subtractions are involved, by also taking into account the computing of $\delta_{x^{(1)}x^{(2)}}$.

$$x^{(0)} = x^{(0)}(-1) + \frac{\delta_{x^{(1)}x^{(2)}} + \ \delta_{x^{(1)}x^{(2)}} - \delta_{x^{(3)}}}{3} \tag{3.9}$$

In particular, the computation of $\delta_{x^{(1)}x^{(2)}}$ is performed by means of the equation (3.10).

$$\delta_{x^{(1)}x^{(2)}} = \ \delta_{x^{(1)}} + \delta_{x^{(2)}} \tag{3.10}$$

3.1.3. Error Modeling and Coding

The prediction error related to the current sample, $x^{(0)}$, is obtained by means of the equation (3.11).

$$e_{x^{(0)}} = \lfloor x^{(0)} - \widehat{x}^{(0)} \rfloor \tag{3.11}$$

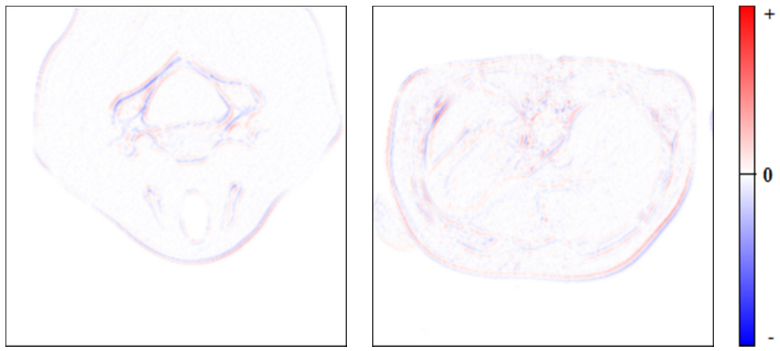
In detail, the prediction error is first mapped, by using the mapping function of the equation (3.12), and then the mapped error is encoded through the *Prediction by Partial Matching with Information Inheritance* (PPMII or PPMd) encoding scheme [**29**].

$$m(e_{x^{(0)}}) = \begin{cases} 2 \times |e_{x^{(0)}}| & if \ e_{x^{(0)}} > 0 \\ 2 \times |e_{x^{(0)}}| - 1 & otherwise \end{cases} \tag{3.12}$$

It is important to remark that the PPMd algorithm improves the efficiency of PPM [**30**]. In particular, its complexity is comparable with other compression schemes, for example LZ77 [**31**], LZ78 [**32**], BWT Transform [**33**].

It is important to emphasize that all the prediction residuals constitute a *residual image* [**55**]. Figures 3.4.a and 3.4.b show two graphical examples of residual images in *false-color*, of a slice of a CT image (*"CT_carotid"*) and of a MR image (*"MR_liver_t2e1"),* respectively. Such residual images are obtained by using the 3D-DLMP predictive structure. The positive errors are represented through a gradient of color, which varies from white to red, and the negative errors are represented through a gradient, which varies from white to blue. It is important to point out that the color intensity proportionally grows with respect to the value of the prediction error.

In Figure 3.5, we graphically report the distribution of prediction errors related to a slice of a MR image (the "*MR_liver_t1*" image). It should be noted that such distribution follows a skewed Laplacian-like distribution, centered on zero. In general, we obtain similar distributions for all the slices of a 3-D medical image, by using our predictive structures. In literature, Laplacian-like distributions of prediction errors are efficiently modeled and coded [**18**].



(a) (b)

Figure 3.4: A false-color residual image of a slice of a CT image (a) and of a MR image (b) (from [**20**]).

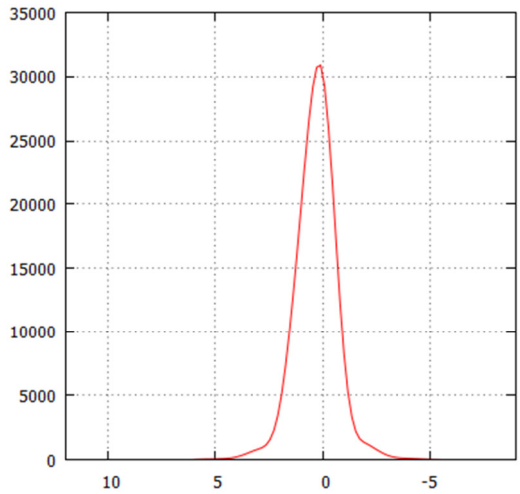


Figure 3.5: The prediction error distribution of a slice of a MR image (from [**20**])

3.1.4. Experimental Results

This section focusses on the experimental results achieved by using our approach on the dataset described in Appendix A.1, which is composed of four 3-D CT images and four 3-D MR images. It is important to point out that we

experimentally performed our approach by using two different orders (i.e., 2 and 4) for the coding of the prediction errors through the PPMd algorithm.

In detail, Table 3.1 reports the experimental results, in terms of BPS, for each one of the CT images. In particular, the first column reports the images, while the second and third columns report the results by using the order equal to 2 and the order equal to 4 for the coding of the prediction errors, respectively. Analogously to Table 3.1, in Table 3.2 we report the achieved experimental results on the MR images.

It is easy to note that our approach obtains better results when the order is set to 4. However, the computational complexity of PPMd is affected by increasing the order.

In addition, we analyzed the coding of prediction errors in terms of required memory through the PPMd scheme, for both the used orders.

Table 3.1: Experimental results achieved on the CT images.

Images	**MILC (PPMd o = 2)**	**MILC (PPMd o = 4)**
CT_skull	2.0683	2.0306
CT_wrist	1.0776	1.0666
CT_carotid	1.4087	1.3584
CT_Aperts	0.8473	0.8190
Average	*1.3505*	*1.3187*

Table 3.2: Experimental results achieved on the MR images.

Images	**MILC (PPMd o = 2)**	**MILC (PPMd o = 4)**
MR_liver_t1	2.1839	2.1968
MR_liver_t2e1	1.7749	1.7590
MR_sag_head	2.1201	2.0975
MR_ped_chest	1.6612	1.6556
Average	*1.9350*	*1.9272*

Table 3.3: Memory usage for the coding of prediction errors on the CT images.

Method / Images	**CT_skull**	**CT_wrist**	**CT_carotid**	**CT_Aperts**	***Average***
PPMd (order = 4)	9.90	0.70	2.50	1.40	*3.63*
PPMd (order = 2)	0.40	0.10	0.30	0.10	*0.23*

Table 3.4: Memory usage for the coding of prediction errors on the MR images.

Method / Images	**MR_liver_t1**	**MR_liver_t2e1**	**MR_sag_head**	**MR_ped_chest**	***Average***
PPMd (order = 4)	1.50	3.60	4.10	0.70	*2.48*
PPMd (order = 2)	0.10	0.20	0.30	0.10	*0.18*

In Tables 3.3 and 3.4, we report the used memory (in terms of Megabytes), related to the coding of the errors. In particular, the first column indicates the

used order for the PPMd scheme, the columns from the second to the fifth indicate the images and the last column indicates the average.

In Figures 3.6 and 3.7, we graphically represent the information contained in Table 3.3 and Table 3.4, respectively.

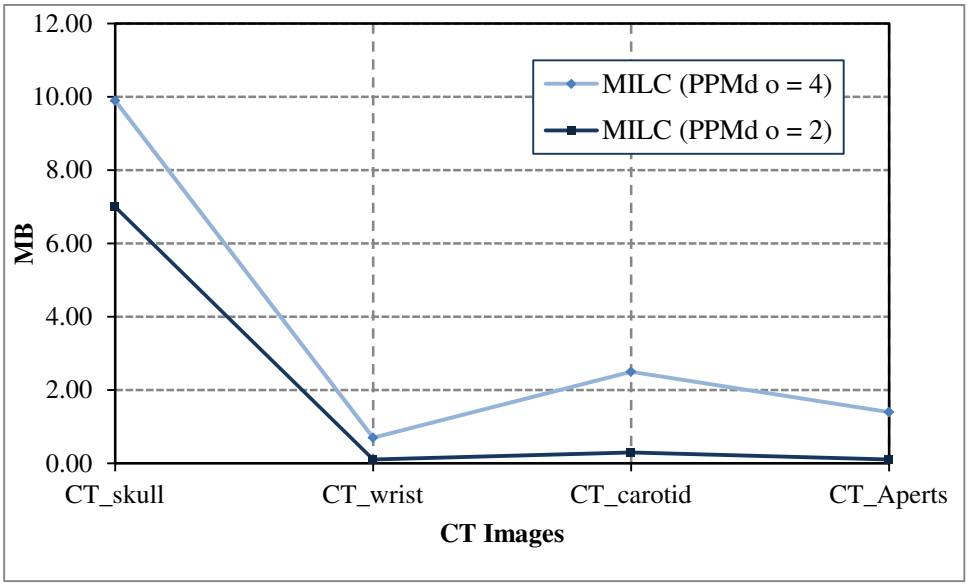


Figure 3.6: A graphical representation of Table 3.3.

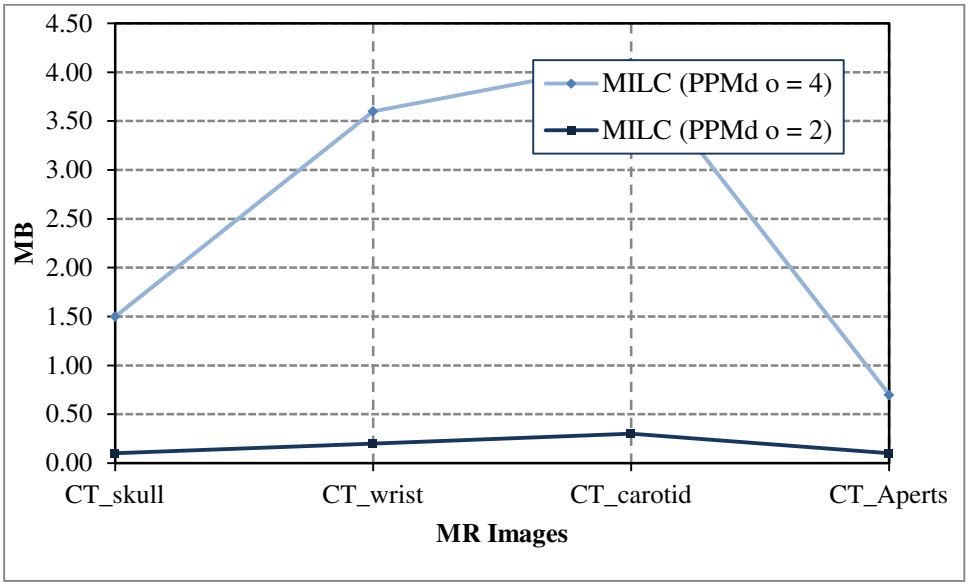


Figure 3.7: A graphical representation of Table 3.4.

The figures show how by setting the order equal to 4 for the PPMd scheme, MILC uses 16 times (CT images) and 14 times (MR images) more memory with respect to the case in which the order is set to 2.

3.1.5. Discussion of Results

We compare the results achieved by using the MILC algorithm with respect to other currently used approaches.

In Table 3.5, we report the results in terms of *bits-per-sample* (BPS), for each one of the CT images (the columns from the second to the fifth), obtained by using different methods (first column) and, in the last column, the average of the results related to a method is reported.

Similarly to Table 3.5, in Table 3.6 we report the achieved results for the MR images.

Table 3.5: Memory usage for the coding of prediction errors on the MR images.

Methods	**CT_skull**	**CT_wrist**	**CT_carotid**	**CT_Aperts**	***Average***
3D-ESCOT [**22**]	1.8350	1.0570	1.3470	0.8580	*1.2743*
MILC (PPMd o = 4)	2.0306	1.0666	1.3584	0.8190	*1.3187*
MILC (PPMd o = 2)	2.0683	1.0776	1.4087	0.8473	*1.3505*
AT-SPIHT [**15**]	1.9180	1.1150	1.4790	0.9090	*1.3553*
3D-CB-EZW [**13**]	2.0095	1.1393	1.3930	0.8923	*1.3585*
DPCM+PPMd [**11**]	2.1190	1.0290	1.4710	0.8670	*1.3715*
3D-SPIHT [**22**]	1.9750	1.1720	1.4340	0.9980	*1.3948*
3D-EZW [**13**]	2.2251	1.2828	1.5069	1.0024	*1.5043*
JPEG-LS [**14**]	2.8460	1.6531	1.7388	1.0637	*1.8254*

Table 3.6: Memory usage for the coding of prediction errors on the MR images.

Methods	**MR_liver_t1**	**MR_liver_t2e1**	**MR_sag_head**	**MR_ped_chest**	***Average***
3D-ESCOT	2.0760	1.5100	1.9370	1.6180	*1.7853*
MILC (PPMd o=4)	2.1968	1.7590	2.0975	1.6556	*1.9272*
MILC (PPMd o=2)	2.1839	1.7749	2.1201	1.6612	*1.9350*
3D-SPIHT	2.2480	1.6700	2.0710	1.7420	*1.9328*
3D-CB-EZW	2.2076	1.6591	2.2846	1.8705	*2.0055*
DPCM+PPMd	2.3900	2.0250	2.1270	1.6890	*2.0578*
3D-EZW	2.3743	1.8085	2.3883	2.0499	*2.1553*
JPEG-LS	3.1582	2.3692	2.5567	2.9282	*2.7531*

As can be observed from the tables, the results achieved, on average, for the 3-D CT images are slightly worse than the 3D-ESCOT approach, which is the most performing schema. In detail, our approach outperforms 3D-ESCOT only in the case of the *"CT_Aperts"* image. Regarding the case of the 3-D MR images, the results achieved by the MILC algorithm, are significantly worse with respect to the 3D-ESCOT algorithm.

On the other hand, the MILC approach presents different advantages related to the trade-off between the computational complexity and the compression performances. In particular, the possibility to easily implement our approach and the low resources usage, with our approach being preferable in various scenarios, even in the case of hardware restrictions.

3.2. Parallel Low-Complexity Lossless Compression of 3-D Medical Images

In this section, we focus on the review of a novel design and implementation for the MILC compression algorithm, which is denoted as "*Parallel MILC*" [**15**]. In particular, Parallel MILC is able to exploit the features and capabilities of the parallel computing paradigm. It is important to consider that our approach can be executed on several heterogeneous devices, which support the OpenCL framework. Currently, there are many types of devices that support the OpenCL framework, for example *Central Processing Units* (*CPUs*), *Graphic Processing Units* (*GPUs*), *Field Programmable Gate Arrays* (*FPGAs*), etc.. The main aim of Parallel MILC is to improve the execution time of the compression algorithm.

In detail, our objective is the redesigning of the MILC compression strategy, according to the OpenCL framework. The key points of this framework are briefly reviewed in the following subsection.

3.2.1. Review of the OpenCL Framework

The *Open Computing Language* (*OpenCL*) allows to design parallelized applications. An Open CL-based application can be executed across heterogeneous platforms, i.e., CPUs, GPUs, FPGAs as well as many others. It is important to point out that OpenCL supports both task-based and data-based parallelism.

Regarding the OpenCL platform model, it is easy to note that only a single *Host entity* is included in the model. In detail, as shown in Figure 3.8, the Host is connected to one or more device(s). The devices connected to the Host, need to support the OpenCL framework, and are denoted as *OpenCL devices*.

In this case, the Host serves as a sort of *"connection point"* between the OpenCL devices and the external environment (i.e. I/O, interactions with the end-user, etc.). On the other hand, an OpenCL device is able to execute a stream of instructions or a function (denoted as the *kernel*).

By focusing on the logical architecture of an OpenCL device (graphically represented in Figure 3.9), it is easy to find out that such device can be composed by one or more *Compute Units* (*CUs*). Furthermore, a CU can be composed of different *Processing Elements* (*PEs*).

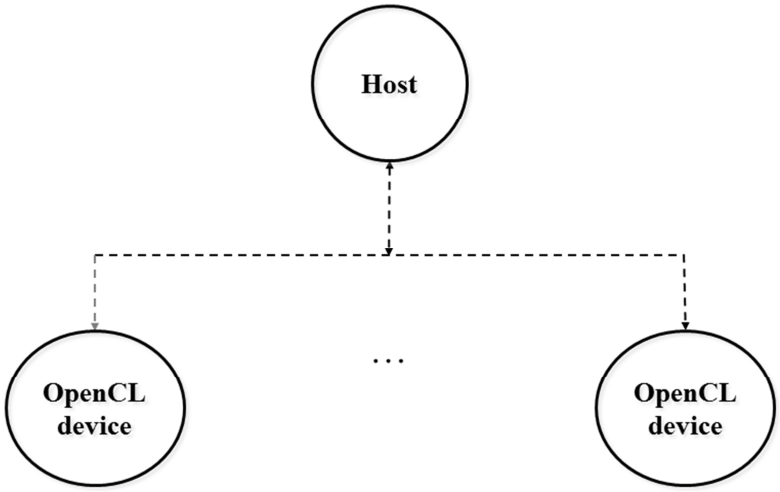


Figure 3.8: Graphical representation of the OpenCL platform model.

Generally, an application based on the OpenCL framework is characterized by two main components:

- The *host program*;
- One or more *kernels*.

The host program has to be executed on the Host entity, while the kernel(s) have to be executed on the OpenCL device(s). In detail, the OpenCL standard has no explicit definitions on how a host program works [**39**]. Nevertheless, the interactions with the OpenCL objects are well defined [**39**]. While, the kernel is substantially a sort of function, which is charge of implementing all (or a significant portion) of the application logic of a program. In particular, through the kernel is possible to perform the logical work of such programs [**39, 40**].

Two main types of kernels are defined by the OpenCL framework, namely, *OpenCL kernels* and *native kernels*. An OpenCL kernel is characterized by the possibility to program it through the embedded C-based programming language (i.e. the OpenCL C language). Furthermore, it can be compiled through an OpenCL compiler and executed over any OpenCL device. Regarding a native kernel, it can be composed of one or more external functions, which can be accessed by OpenCL through a specific function pointer [**39**].

It is important to take into account another fundamental aspect of the OpenCL framework, relating to the manner in which a kernel is executed. Considering a basic example, in which a host program has been defined a kernel. The host program is able to invoke, by using a proper command, the execution of the kernel on one or more OpenCL devices.

In particular, once an OpenCL device has received such a command, the relative "*runtime environment*" enables it to create an integer index space,

which is denoted as *NDRange* [**39**]. It is important to emphasize that an instance of NDRange characterizes a N-dimensional index space. The current version of the OpenCL framework (1.2) allows to create an N-dimensional instance of NDRange, with $N \in \{1, 2, 3\}$. Thus, the simplest type of instance of NDRange can describe a 1-dimensional integer space ($N = 1$). In this case, the instance of NDRange is defined as a sort of linear array of a given size.

Subsequently, in each "*point*" of the instance, a kernel is instantiated. It should be noted that a kernel under execution is denoted as "*work-item*", and can be addressed through its N global identifiers (*global IDs*), where N is the number of the dimensions of the instance. For example, in a bi-dimensional NDRange (where $N = 2$), it is possible to address a work-item through two global IDs (each one related to a dimension).

The work-items can be organized into groups (*work-groups*), which can be identified by their group identifier (*work-group ID*).

Another main point of OpenCL is the *memory model*. Substantially, two main types of objects are defined by the framework, namely, the *buffer* and the *image* objects. Basically, a buffer object can be viewed as a portion of memory provided by the kernel, which the programmer can use as wanted (i.e. to store data structure, arrays, matrices, etc.). While an image object can be used for the management of images.

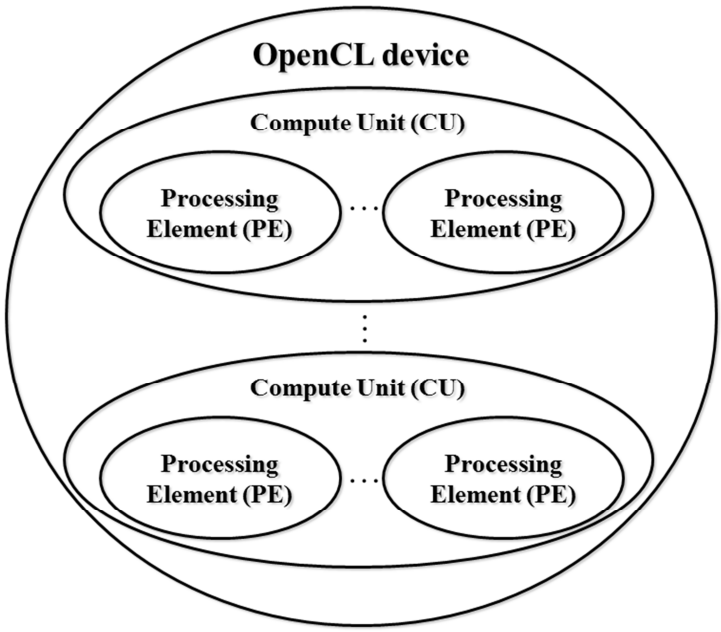


Figure 3.9: Graphical representation of the logical architecture of an OpenCL device.

Furthermore, five memory regions are defined by the memory model and can be used in OpenCL-based applications:

- *Host Memory;*
- *Private Memory;*
- *Local Memory;*
- *Global Memory;*
- *Constant Memory.*

In particular, the Host Memory is a memory region accessible and modifiable only by the host. Whereas, the other regions are accessed by the OpenCL devices. The Private and Local Memory can be only accessed/modified respectively by a work-item and a work-group. Furthermore, the Global Memory is accessible/modifiable by all the work-items. Analogously to the

Global Memory, the Constant Memory is accessible by all the work-items with the limitation that this region is not modifiable.

3.2.2. Description of the Parallel MILC

As discussed in Section 3.1, the MILC algorithm processes each sample of the input 3-D medical image, in the *raster-scan* order. In particular, each sample is predicted and its related prediction error is mapped and sent to the PPMd encoding scheme. On the other hand, from the coded stream the MILC decompression algorithm is able to process the next prediction error (after it has been performed both the PPMd decoding and the inverse mapping phases). By using the obtained prediction error, the inverse prediction is performed and the reconstructed sample is then written to the decoded stream.

Thus, the final decoded stream will be equal to the original uncompressed image. It is worth noting that the order in which the prediction errors are stored is essential, because is the same as the original uncompressed image.

The basic idea behind the Parallel MILC compression strategy is related to the assumption that each sample can be "*independently predicted*", provided that the whole 3-D medical image has been acquired. In such scenario, it is noticeable that several samples are processed at the same time, in different manner with respect to the MILC. Thus, it is not possible an "a priori" estimation in relation to the order, in which the prediction errors will be generated. However, by using some information related to the data format (i.e.

about the storing order of the samples) of the input 3-D medical image, such an issue can be solved.

In detail, through the coordinates of a sample, it is possible to store its related prediction error into a specific position (p) of a temporary buffer. This buffer maintains the residual image. Once the prediction of all the samples is computed, the whole temporary buffer is sent to the PPMd scheme.

It is important to emphasize that MILC does not need any information about the data format and does not require any buffer for the maintaining of the residual image. The latter are the main differences and drawbacks of Parallel MILC with respect to MILC.

Since it is generally easy to identify the data format of the input 3-D medical image, and, consequently, the processing order of the samples, the first one should not always be considered a drawback. It is important to remark that in medical and medical-related environments in which time may be a "critical" parameter, the effective speedup obtained in the Parallel MILC execution performance could justify such issues.

An important aspect related to the operation logic of the Parallel MILC decompression algorithm is that it is the same as the one used by MILC. Thus, it is possible to encode a medical image transparently, by using MILC or Parallel MILC. Thus, the coded stream can be decoded always in the same way.

It is important to note that Parallel MILC is an OpenCL application, therefore, it is characterized by the host program and an OpenCL kernel.

The following subsections focus on the details related to the novel compression algorithm, especially emphasizing the design and implementation details of both the above defined Parallel MILC components.

3.2.3. The Host Program

Despite most of the logical work is being in charge of the OpenCL kernel, the host program plays a significant role. In detail, this program is able to locate the OpenCL device(s) and permits to define the characteristics of an instance of the NDRange. The main steps of the host program of Parallel MILC can be synthesized in the following (adapted from [**47**]):

1. Detection and identification of the platforms supporting OpenCL;
2. If no platforms are found, notification of the end-user and execution of the MILC scheme;
3. Selection of the preferred OpenCL platform among the several available ones. Since a platform may contain more than one device (e.g. multiple CPUs or multiple GPUs), such a choice can be automatically made by estimating the best-performing device among all the available platforms. Alternatively, the user can select a specific device of a given platform through the appropriate parameters;
4. Selection of the OpenCL device;
5. Reading from the file and loading in the Host Memory of the three-dimensional medical image (MI) taken as input;

6. Creation of a buffer object on the OpenCL device. In detail, MI is copied from the Host Memory to the Constant Memory of the OpenCL device;
7. Allocation of a sufficient-sized buffer object in the Global Memory of the OpenCL device. Such buffer maintains the residual image;
8. Retrieving from the OpenCL device of the work group maximum size ($maxSizeWG$). Such information is used in order to optimize the number of work-items per work-group;
9. Enqueuing of a three-dimensional instance of the NDRange (where $N = 3$) of the same size of MI, to be instantiated on the OpenCL device. In particular, in this case, the NDRange is logically mapped onto the whole MI. Thus, each sample is independently processed by a single work-item. In detail, MI is subdivided into several "*sub-cubes*", each of them is mapped into a work-group, as shown in Figure 3.10. Therefore, the dimensions of a sub-cube depend on the $maxSizeWG$ parameter. For example, if $maxSizeWG$ is equal to 256, then the dimensions of a sub-cube are of 8 as width, 8 as height and 4 relating to the third dimension ($8 \times 8 \times 4 = 256$).
10. Copying of the residual image from the Global Memory of the OpenCL device to the host program once the execution of the kernel is completed, and sending of such image to the PPMd encoding scheme.

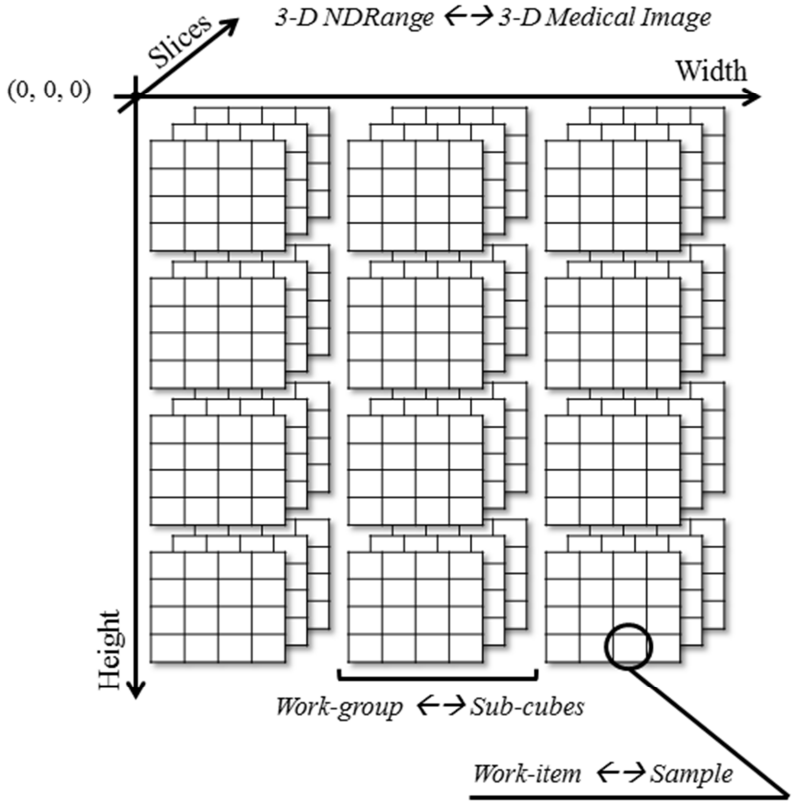


Figure 3.10: Graphical representation of the mapping of *MI* into work-groups (from [**47**]).

3.2.4. The OpenCL Kernel

The OpenCL device is in charge of carry out all the computational work of the Parallel MILC. As previously discussed, a work-item executes an instance of the kernel.

In our approach, each work-item independently performs the prediction of the relative associated sample. Essentially, since the whole input medical image is mapped into the instance of the NDRange, the sample, associated to the work-item, is the one that has the same coordinates of the work-item.

In the following, we describe the most important steps, which are performed by the OpenCL kernel (adapted from [**47**]):

1. Obtaining of the x, y and z coordinates of the work-item wi in the instance of the NDRange. Such coordinates univocally identify the associated sample of MI, denoted as $x^{(0)}$. Thus, $x^{(0)}$ is processed by wi;
2. Retrieving of $x^{(0)}$;
3. If z is equal to 0, then retrieving of the appropriate bi-dimensional prediction context, since $x^{(0)}$ is a sample of the first slice of MI, and performing of the 2D-LMP predictor; Go to step 5;
4. Otherwise, retrieving of the three-dimensional prediction context and performing of the 3D-DLMP predictor;
5. Computing of the prediction error $e_{x^{(0)}}$;
6. By using the coordinates of wi and the information of the data format of MI, locating of the position p in which the prediction error $e_{x^{(0)}}$ will be stored.
7. Storing of the prediction error into the buffer at position p. The buffer is used for maintaining the residual image and it is located in the Global Memory.

3.2.5. Experimental Results

In this subsection, we focus on the experimental results, which are obtained by the testing of the Parallel MILC algorithm. All the experiments are performed

on the dataset, which is outlined in Appendix A.1. It is important to remark that the main aim of such testing is to show the speedup provided by Parallel MILC, with respect to MILC, in terms of time execution performance.

In detail, we implemented the MILC algorithm and the host program for Parallel MILC through the C programming language and both are compiled by using the same environment. Furthermore, both the implementations manage the input data and the prediction errors in the same manner. The main objective of this design choice is to consider only the application logic of both algorithms during the testing.

Thus, the errors are sent to PPMd in a non-optimized way. Thus, we used PPMd with an order equal to 8, instead of using the order equal to 4 (default order), as described in Section 3.1.3.

In order to evaluate the effectiveness of Parallel MILC, each image of the dataset was first compressed by using the MILC and the Parallel MILC approaches. In particular, Parallel MILC is executed over the CPU and the GPU on three hardware configurations.

In Tables 3.7, 3.8 and 3.9 the used configurations are respectively described, denoted Configuration 1, Configuration 2 and Configuration 3. It is important to consider that during the testing phase, we estimated the time execution in terms of milliseconds.

Table 3.7: Description of Configuration 1.

Configuration 1	
Secondary Memory	*HDD 1000 GB*
Operating System	*Microsoft (TM) Windows (R) 8.1 - 64 bit*
OpenCL version	*1.2 (CPU and GPU)*
CPU	
Type	Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz
RAM	8 GB DDR (PC3-12800)
Max CUs	4
Max Work Group Size	1024
GPU	
Type	Intel(R) HD Graphics 4600
RAM	2 GB DDR3 (Shared)
Max Clock Frequency	600 MHz
Max CUs	20
Max Work Group Size	512

Table 3.8: Description of Configuration 2.

Configuration 2	
Secondary Memory	eMMC 32 GB
Operating System	Microsoft (TM) Windows (R) 8.1 - 32 bit
OpenCL version	1.2 (CPU and GPU)
CPU	
Type	Intel(R) Atom(TM) CPU - Z3740 @ 1.33GHz
RAM	2 GB DDR3
Max CUs	4
Max Work Group Size	1024
GPU	
Type	Intel(R) HD Graphics
RAM	1 GB DDR3 (Shared)
Max Clock Frequency	200 MHz
Max CUs	4
Max Work Group Size	256

Table 3.9: Description of Configuration 3.

Configuration 3	
Secondary Memory	HDD 500 GB (5400 RPM) + SSD 8 GB
Operating System	Microsoft (TM) Windows (R) 8.1 - 64 bit
OpenCL version	1.2 (CPU and GPU)
CPU	
Type	Intel (R) Pentium(R) 3556U @ 1.70GHz
RAM	4 GB DDR3 (PC-12800)
Max CUs	2
Max Work Group Size	1024
GPU	
Type	Intel(R) HD Graphics
RAM	2 GB DDR3 (Shared)
Max Clock Frequency	1000 MHz
Max CUs	10
Max Work Group Size	256

Table 3.10 reports the experimental results achieved by Parallel MILC in terms of execution performances over Configuration 1. The table is logically subdivided into three "macro sections", each of them is related to one of the main phases of the MILC algorithm and the Parallel MILC algorithm, namely, the prediction and the error coding phases.

In particular, regarding the MILC, the execution time (evaluated as milliseconds (ms)), the relative throughput (evaluated in megabytes per seconds (MB/s)) are reported for each image of the dataset (first column), in the second and third columns, respectively. Whereas, regarding Parallel MILC, the execution time, the throughput and the relative speedup (with respect to MILC) are respectively reported in the fourth, fifth and sixth columns in

relation to the execution over the CPU and are respectively reported in the seventh, eighth and ninth columns in relation to the execution over the GPU.

Furthermore, since the prediction errors are managed in the same manner by both MILC and Parallel MILC, the phase related to the error coding has substantially similar execution times in both cases. Therefore, in the tenth column a single execution time is reported, concerning the error coding phase of both algorithms.

Analogously to Table 3.10, in Table 3.11 and Table 3.12, the results achieved by the execution of MILC and Parallel MILC, respectively on Configuration 2 and Configuration 3 are reported.

Table 3.13 focuses on the achieved results related to the execution performances of the decompression algorithm of Parallel MILC/MILC. In detail, for each image of the dataset (first column), the execution time (in ms) of the error decoding, the execution time and the processed megabytes per second (MB/s) of the inverse prediction phase are reported. In particular, these results are obtained by using Configuration 1 (from the second to the fourth column), Configuration 2 (from the fifth to the seventh column) and Configuration 3 (from the eighth to the tenth column) respectively.

Table 3.10: Achieved results by the compression algorithm on Configuration 1.

Images	***Prediction Phase***								***Errors Coding***
	MILC		**Parallel MILC**						
			CPU			**GPU**			
	Execution (ms)	*MB/s*	*Execution (ms)*	*MB/s*	*Speedup*	*Execution (ms)*	*MB/s*	*Speedup*	*Execution (ms)*
CT_skull	6906	1.74	578	20.76	12	344	34.88	20	1032
CT_wrist	6157	1.79	594	18.52	10	265	41.51	23	422
CT_carotid	2235	1.79	390	10.26	6	125	32.00	18	265
CT_Aperts	3423	1.75	422	14.22	8	156	38.46	22	422
MR_liver_t1	1672	1.79	376	7.98	4	125	24.00	13	250
MR_liver_t2e1	1688	1.78	359	8.36	5	110	27.27	15	266
MR_sag_head	1672	1.79	375	8.00	4	125	24.00	13	328
MR_ped_chest	2219	1.80	375	10.67	6	141	28.37	16	250

Table 3.11: Achieved results by the compression algorithm on Configuration 2 (adapted from [**47**]).

Images	***Prediction Phase***								***Errors Coding***
	MILC		**Parallel MILC**						
			CPU			**GPU**			
	Execution (ms)	*MB/s*	*Execution (ms)*	*MB/s*	*Speedup*	*Execution (ms)*	*MB/s*	*Speedup*	*Execution (ms)*
CT_skull	17376	0.69	1907	6.29	9	2641	4.54	7	3047
CT_wrist	15720	0.70	1640	6.71	10	1719	6.40	9	1406
CT_carotid	5782	0.69	1219	3.28	5	687	5.82	8	813
CT_Aperts	8704	0.69	1328	4.52	7	984	6.10	9	859
MR_liver_t1	4297	0.70	1187	2.53	4	720	4.17	6	672
MR_liver_t2e1	4375	0.69	1219	2.46	4	704	4.26	6	890
MR_sag_head	4282	0.70	1219	2.46	4	860	3.49	5	985
MR_ped_chest	5704	0.70	1187	3.37	5	735	5.44	8	656

Table 3.12: Achieved results by the compression algorithm on Configuration 3 (adapted from [**47**]).

Images	***Prediction Phase***								***Errors Coding***
	MILC		**Parallel MILC**						
			CPU			**GPU**			
	Execution (ms)	*MB/s*	*Execution (ms)*	*MB/s*	*Speedup*	*Execution (ms)*	*MB/s*	*Speedup*	*Execution (ms)*
CT_skull	12204	0.98	844	14.22	14	391	30.69	31	1688
CT_wrist	11157	0.99	813	13.53	14	313	35.14	36	672
CT_carotid	4015	1.00	625	6.40	6	172	23.26	23	453
CT_Aperts	6016	1.00	672	8.93	9	203	29.56	30	453
MR_liver_t1	3016	0.99	610	4.92	5	157	19.11	19	329
MR_liver_t2e1	3015	1.00	610	4.92	5	172	17.44	18	469
MR_sag_head	3015	1.00	609	4.93	5	156	19.23	19	531
MR_ped_chest	4015	1.00	641	6.24	6	172	23.26	23	343

Table 3.13: Achieved results by the decompression algorithm on Configurations 1, 2 and 3 (adapted from [**47**]).

Images	**Configuration 1**			**Configuration 2**			**Configuration 3**		
	Errors Decoding	***Inverse Prediction Phase***		***Errors Decoding***	***Inverse Prediction Phase***		***Errors Decoding***	***Inverse Prediction Phase***	
	Execution (ms)	*Execution (ms)*	*MB/s*	*Execution (ms)*	*Execution (ms)*	*MB/s*	*Execution (ms)*	*Execution (ms)*	*MB/s*
CT_skull	1203	7047	1.70	3516	18235	0.66	1906	12578	0.95
CT_wrist	562	6438	1.71	1704	16673	0.66	953	11516	0.96
CT_carotid	328	2360	1.69	969	5969	0.67	500	4187	0.96
CT_Aperts	343	3485	1.72	1094	8766	0.68	531	6328	0.95
MR_liver_t1	265	1751	1.71	781	4375	0.69	422	3141	0.96
MR_liver_t2e1	359	1735	1.73	1000	4422	0.68	516	3126	0.96
MR_sag_head	407	1750	1.71	829	4547	0.66	546	3125	0.96
MR_ped_chest	297	2375	1.68	1024	5938	0.67	453	4187	0.96

Figures 3.11, 3.12 and 3.13 graphically represent the information of Tables 3.10, 3.11 and 3.12, respectively. It is important to point out that each graph reports the execution time and speedup of Parallel MILC over the CPU (orange column and dark red line) and over the GPU (light blue column and violet line), respectively by using Configuration 1 (Figure 3.11), Configuration 2 (Figure 3.12) and Configuration 3 (Figure 3.13).

The figures highlight how the speedup provided by Parallel MILC varies from a minimum of 4 times faster to 36 times faster than MILC.

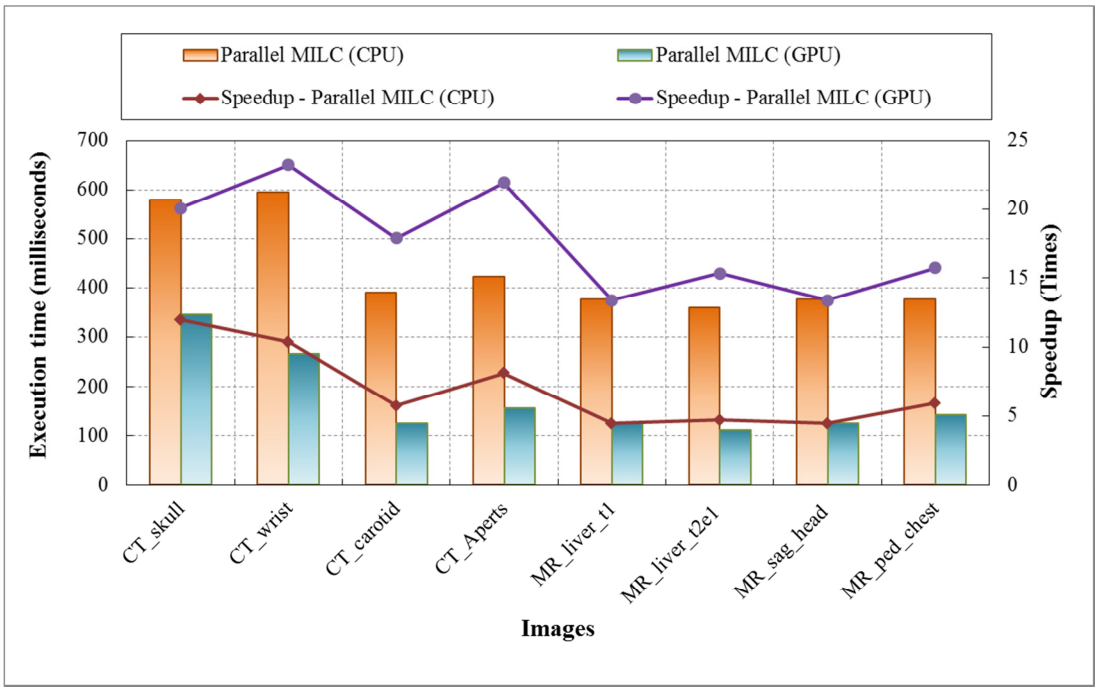


Figure 3.11: Graphical representation of Table 3.10.

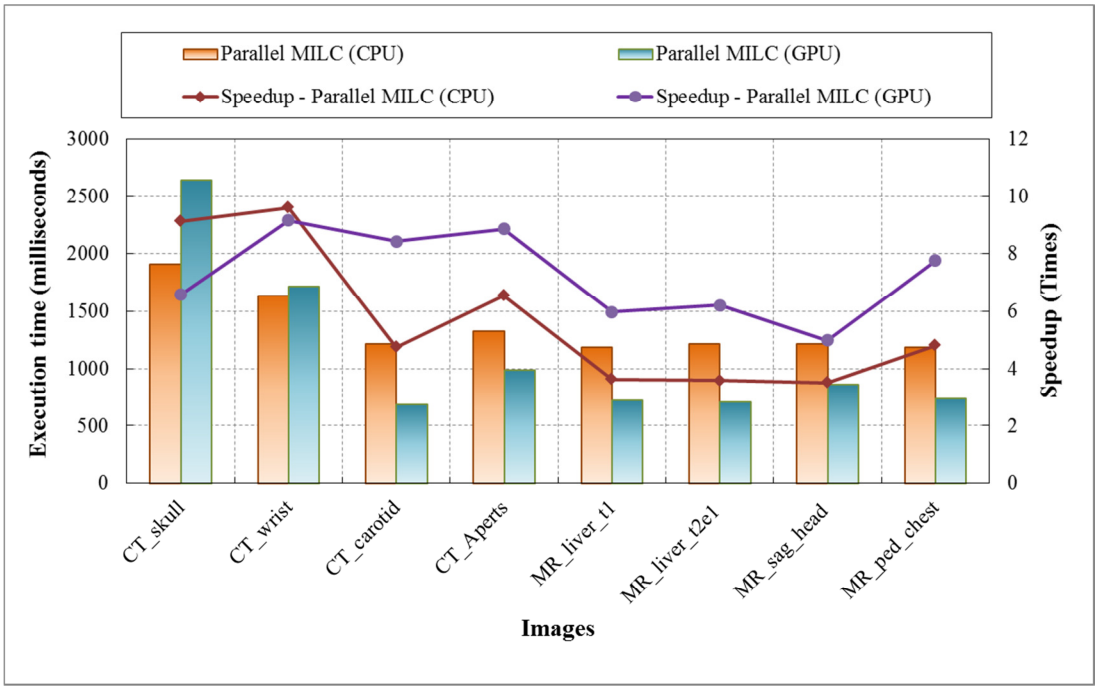


Figure 3.12: Graphical representation of Table 3.11.

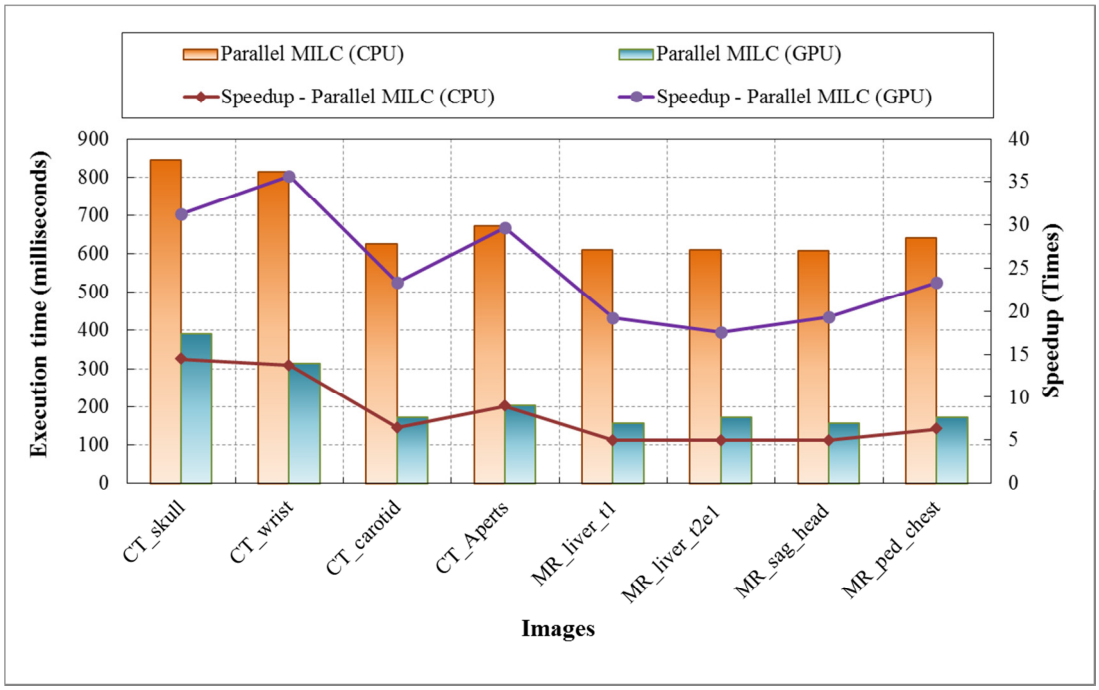


Figure 3.13: Graphical representation of Table 3.12.

Chapter – 4

Protection of 3-D Medical and 3-D Microscopy Images

Highlights of the chapter

This chapter focuses on the protection, through Digital Watermarking techniques, of sensitive multidimensional data (e.g. 3-D medical images, 3-D microscopy images, etc.).

First, we investigated the possibility to protect and, simultaneously, compress 3-D medical images. Therefore, we presented a novel hybrid approach that is suitable for the compression of such data, and, optionally, permits the embedding, at the same time of the compression.

Subsequently, we investigated the possibility to protect the copyright of 3-D microscopy images. It should be noted that microscopy images can be used for different sensitive applications (e.g. research studies, digital forensic, etc.). Starting from these considerations, we presented a protection schema, based on watermarking techniques.

4.1. Introduction

In this section, we explore the basic ideas upon which digital watermarking techniques rely. Such techniques can be used on multimedia objects (i.e., images, videos, etc.) for their protection as well as to hide information in the data.

4.1.1. Review of Watermarking Techniques on Images

With the increasing diffusion of the *Internet*, *Cloud Computing* and other networking technologies, the problem of image protection has been particularly felt. In fact, the intellectual property rights of multimedia digital objects (i.e. images, videos, etc.) can be easily violated, if adequate protections are not adopted. In such scenarios, digital watermarking techniques play an important role, in order to protect and preserve the copyright and value of an image. In detail, the main objective of such techniques is related to hiding data in digital objects (i.e. images, videos, audio, etc.), so as to protect their value.

We can define a digital watermark W as a sequence of N symbols (bits). For instance, a textual message or a simple logo can be characterized as a short sequence of bits.

A watermark can be classified in terms of its "visibility", if the embedded object is visible or not to the end-user. In particular, a digital watermarking technique falls under two main categories:

- *Visible watermark;*

- *Invisible watermark;*

In detail, a visible watermark is perceptible to the end-user, who is able to easily determine whether a watermark is present or not. Whereas, invisible watermarks are not visible by the end-user. In particular, if an object is affected by an invisible watermark, it is necessary to use an appropriate algorithm to extract/detect it. In this manner, the owner can prove his ownership. It is important to point out that if a digital object, in which a watermark is embedded, is copied unauthorized, the watermark is also maintained on the copy.

Several watermark attacks have been currently proposed. The main objective of a watermark attack is to compromise the embedding algorithm of the invisible watermark, by trying to remove/modify the embedded watermark, with minor alterations of the perceptual quality of the image, in order to not invalidate its value.

In [**48**] and [**49**] a benchmark suite, which simulates the most common watermark attacks, is proposed. In detail, such a suite simulates several attacks, such as *geometrical transformation* (i.e. *rotation*, *scaling*, etc.), *lossy compression*, *filter applications* and many others.

It is important to note that an invisible watermark can be categorized in relation to its resistance against a watermark attack. The main categories are the following:

- *Fragile watermark;*
- *Robust watermark.*

If an image, affected by a fragile watermark, is altered the embedded watermark cannot be extracted, even by using the extraction algorithm. In other words, a fragile watermark is "defeated", if any manipulations occur on the image **[50]**. Generally, a robust watermark is resistant against a subset of the aforementioned attacks. It is important to note that if the image, in which a robust watermark is embedded, is heavily altered, the watermark may not be resistant enough. However, it is easily noticeable that heavy alterations drastically alter the image and its value.

It is possible to measure the robustness of a robust watermark, by verifying the "similarities" between the original watermark W and the watermark W^*. In particular, W^* is the watermark string extracted from the watermarked image which suffered a watermark attack. Different similarities measures have been introduced, we considered the one described in **[51]** and outlined by the equation (4.1).

$$sim(W, W^*) = \sum_{i=1}^{N} \frac{W(i) \times W^*(i)}{\sqrt{W(i) \times W^*(i)}} \tag{4.1}$$

It is important to note that $W(i)$ and $W^*(i)$ indicate the i-th bit of W and of W^*, respectively. In addition, N is the length of W and W^*.

It should be noted that invisible watermarks can be classified in relation to the way in which a watermark can be extracted, once embedded into an image. We can denote a watermark as a *blind watermark*, if its extraction, from a watermarked image, is permitted without the original image. Otherwise, if the

extraction is possible only with the original image, then such a watermark can be called a *non-blind watermark*.

4.2. Protection and Compression of 3-D Medical Images

In this section, we review the approach we introduced in [**37**] and [**52**], which permits to compress 3-D medical images and optionally embed, at the same time of the compression process, a digital watermark. In particular, we outline the compression strategy adopted (Section 4.2.1) and the watermarking scheme (Section 4.2.2).

4.2.1. The Compression Strategy

Regarding the compression strategy, the compression algorithm is based on the predictive model. In particular, a novel configurable inter-slice prediction model is used, which takes advantage of the three-dimensional redundancy.

In detail, for the prediction of the first slice of the image taken as input, we use the well-established 2-D *Median Edge Detector* (*MED*) predictor (reviewed in Section 4.2.1.1). In addition, the other slices are predicted by using the proposed inter-slice prediction model (described in Section 4.2.1.2), which is derived by the *Linear Predictor* (*LP*) (introduced in [**9**]).

4.2.1.1. The MED (Median Edge Detector) Predictor

The MED Predictor uses a context of three neighboring pixels, namely, $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$, of the current pixel $x^{(0)}$, as shown in Figure 4.1. The prediction, $\hat{x}^{(0)}$, is performed by means of the equation (4.2).

$$\hat{x}^{(0)} = \begin{cases} \min(x^{(1)}, x^{(2)}) & if\ x^{(3)} \geq \max(x^{(1)}, x^{(2)}) \\ \max(x^{(1)}, x^{(2)}) & if\ x^{(3)} \leq \min(x^{(1)}, x^{(2)}) \\ x^{(1)} + x^{(2)} - x^{(3)} & otherwise \end{cases} \tag{4.2}$$

In detail, if one of the two conditions: $x^{(3)} \geq \max(x^{(1)}, x^{(2)})$ or $x^{(3)} \leq \min(x^{(1)}, x^{(2)})$, is verified, it means that the MED predictor has detected a horizontal or vertical edge, thus, it is highly probable that $x^{(0)}$ follow the identified trend. Otherwise, if no edge is detected, the MED predictor returns a combination of the three neighboring pixels intensity.

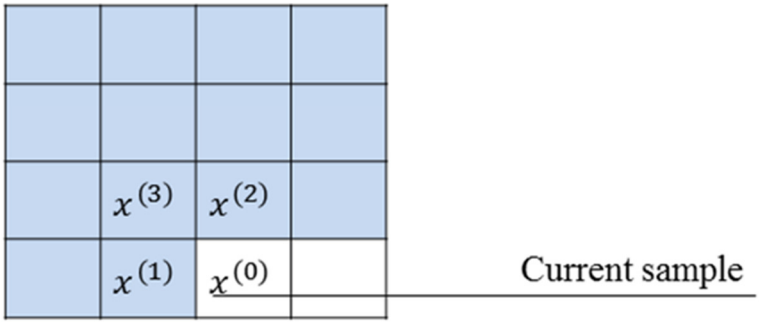


Figure 4.1: A graphical example of the prediction context used by the MED predictor.

4.2.1.2. The Adaptive Inter-slice Predictive Model

The proposed adaptive inter-slice predictive model uses a three-dimensional prediction context composed by $2 \times N + 1$ pixels (where $3 \leq N \leq 8$), in order to exploit the significant correlation between two consecutive slices. Such a context is composed of N neighboring pixels of $x^{(0)}$, in the current slice (denoted as the k-th slice), and $N + 1$ pixels, in the previously coded reference slice (denoted as the r-th slice). It is important to point out that N of the $N + 1$ pixels of the r-th slice have the same spatial coordinates of the ones of the k-th slice.

Furthermore, also the sample with the same spatial coordinates of $x^{(0)}$, in the r-th slice, is used. An example of the prediction context, by considering $N = 8$, is shown in Figure 4.2.

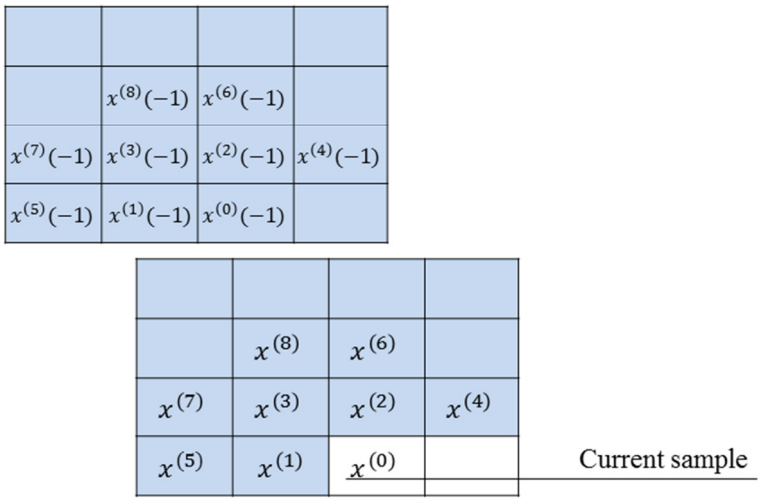


Figure 4.2: A graphical example of the prediction context used by the proposed inter-slice predictor.

The prediction $\hat{x}^{(0)}$, of the current sample $x^{(0)}$, is performed by means of the equation (4.3).

$$\hat{x}^{(0)} = x^{(0)}(-1) + \left(\frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - x^{(i)}(-1))\right) \tag{4.3}$$

The basic idea upon which such predictive structure relies is to consider the distances between the pixels of the prediction context in the k-th slice and the ones in the r-th. Afterwards, the average of such distances is computed. Such an average is then added to $x^{(0)}(-1)$, which is the pixel which has the same spatial coordinates of $x^{(0)}$, in the r-th slice.

4.2.1.3. Error Modeling and Coding

The error distributions produced by the two outlined predictive structures generally follow the Laplacian one. As previously mentioned, such errors can be efficiently managed.

In particular, prediction errors are mapped by using an invertible mapping function, in order to have only non-negative values. It should be noted that that the redundancy among errors is not altered, even if a mapping is applied. Finally, each mapped prediction error is coded through the PPMd scheme.

4.2.2. The Embedding and the Extraction of the Watermark

The reviewed embedding procedure is based on the one introduced in [**54**]. In detail, our procedure is suitable for bi-dimensional images instead of three-dimensional ones.

Such a design choice is derived from the consideration that only a subset of slices can be significant and can be, consequently, extracted from the considered 3-D medical image. Through the proposed approach when such cases occur, the hidden information can be still independently extracted even by a single slice. In fact, we enable the embedding of a watermark inside all the slices or in a user-defined subset of them. In addition, it should be noted that the low computational complexity required to embed a watermark and the possibility to embed it during the compression process, without any significant overhead, makes our approach helpful in many scenarios.

Algorithm 4.1 synthesizes the phases of the proposed approach. In particular, we assume that all the underlined functions in such algorithm use the *readPixelValue* auxiliary procedure, which is delegated to read a specific sample value. In addition, the $2D\_MED$ and the $3D\_InterSlicePreditor$ procedures implement the MED predictor (reviewed in Section 4.2.1.1) and our adaptive inter-slice predictor (explained in Section 4.2.1.2), respectively.

As highlighted by line 2, the side information (i.e., N and r) are written to the OS output stream. In particular, such data are the first information of the stream. In this manner, the decoder is able to perform the decompression.

Subsequently, the coordinates in which the bits of ws (the watermark string) will be embedded are randomly generated, by using the Pin as a seed (lines from 3 to 5).

By means of the if statement (line 9), Algorithm 4.1 checks if the sample, which is being processed, is delegated or not to hide a bit of ws. In particular, if such a check is verified, the *Least-Significant-Bit* (*LSB*) of the current sample is altered, by substituting such bit with the one of ws.

It should be noted that the modified value of the current sample needs to be used for the future predictions of the other samples, due to the fact that this sample may be a neighbor of other samples and the decoder has only the modified information (line 12). Therefore, the future invocation of the *readPixelValue* procedure, on the current sample, returns the modified value (simulating the decompression algorithm behavior), instead of the original one.

On the other hand, if the condition is not satisfied, then the value of the current sample can be extracted from the image without any further processing (lines from 14 to 16).

The user-defined set of slices, denoted as *watermarkSlicesSet* in Algorithm 4.1, contains the indices of the slices, in which ws will be embedded. Therefore, only the slices that belong to such subset will be affected by the watermark.

In Algorithm 4.2, we outline the extraction procedure, which allows to extract the watermark string from watermarked data. In detail, such a procedure needs to know the Pin (the same used for the embedding), in order to identify the coordinates of the samples that hide the watermark of a given slice (*sliceIndex*), as highlighted in line 2.

Subsequently, each bit of the watermark is extracted and concatenated to ws, which is initially empty (lines from 3 to 7).

It is important to note that the raster scan order is used to generate the coordinates of the *prCoordinates* set, in the same way as with the embedding procedure. Furthermore, the extraction procedure can be easily integrated within the decompression algorithm.

Algorithm 4.1 (a): Input and Output of the HYBRID procedure for the compression and extraction (adapted from [**52**]).

Input:

- **MI**: 3D medical image.
- **N**: Number of pixel for 3D context.
- **r**: Reference slice used for the 3D prediction.
- **watermarkSlicesSet**: Set of integer indices representing the slices which the user wants to affect by digital invisible watermark.
- **ws**: Watermark string to embed.
- **Pin**: Number needed for the embedding of the watermark and the relative extraction.

Output:

- **MI_w**: Coded output stream.

Algorithm 4.1 (b): The HYBRID procedure for the compression and extraction (adapted from [**52**]).

```
1: procedure HYBRID(MI, N, r, watermarkSlicesSet, ws, Pin)
2:     Write side information to the OS output stream (i.e. N, r, etc.);
3:     if watermarkSlicesSet is not empty then
4:         Pseudo-randomly generate a set of length(ws) elements, containing unique spatial coordinates (prCoordinates), by using the Pin as seed;
5:     end if
6:     for k = 1 to NumberOfSlices do
7:         wIdx = 1;
8:         for each (i, j) in the k-th slice of MI do
9:             if k ∈ watermarkSlicesSet and (i,j) ∈ prCoordinates then
10:                currentPixel = readPixelValue(MI, i, j, k);
11:                y = Apply_IDW_LSB(ws, wIdx, currentPixel);
12:                Set the readPixelValue function to return the modified value, y, in correspondence to the coordinates (i, j) of the k-th slice of the image MI. (Simulating the decompression algorithm, which have only the modified values);
13:                wIdx = wIdx + 1;
14:            else
15:                y = readPixelValue(MI, i, j, k);
16:            end if
17:            if k = 1 then
18:                ŷ = 2D_MED(MI, i, j, k);
19:            else
20:                ŷ = 3D_InterSlicePredictor(MI, i, j, k, N, r);
21:            end if
22:            residual = ⌊y − ŷ⌋;
23:            Send Mapping(residual) to the PPMd (k-th order) coder for the writing to the OS output stream. Let MI_w the output of such phase;
24:        end for
25:    end for
26:    return MI_w;
27: end procedure
```

Algorithm 4.2 (a): Input and Output of the WATERMARK procedure for the extraction (adapted from [**52**]).

Input:

- **MI_w**: 3D medical image.
- **sliceIndex**: Index of the slice, in which is embedded the watermark string.
- **lengthWs**: Length of the watermark string.
- **Pin**: Pin previously used for the embedding.

Output:

- **ws**: Embedded watermark string.

Algorithm 4.2 (b): The WATERMARK procedure for the extraction (adapted from [**52**]).

```
1: procedure WATERMARK(MI_w, sliceIndex, lengthWs, Pin)
2:     Pseudo-randomly generate a set of length(ws) elements,
   containing unique spatial coordinates (prCoordinates), by
   using the Pin as seed;
3:     for wIdx = 1 to lengthWs do
4:         (i, j) = nextCoordsInRasterscanOrder(prCoordinates);
5:         y = readPixelValue(MIw, i, j, sliceIndex);
6:         ws(wIdx) = leastSignificantBit(y);
7:     end for
8:     return ws;
9: end procedure
```

4.2.3. Experimental Results

We performed our experiments on the dataset, described in Appendix A.1. In particular, Tables 4.1 and 4.2 report the experimental results achieved by considering different values for the N parameter (i.e., from 3 to 8), on the CT and MR images, respectively. In our experiments, the previous slice (the $(k-1)$-th slice), with respect to the current (the k-th slice), is used as the reference slice ($r = k - 1$).

In detail, in the first column of Tables 4.1 and 4.2 the value of the N parameter is indicated, while, in the columns from the second to the fifth, the achieved results are reported for each one of the CT images (Table 4.1) and MR images (Table 4.2). In both the tables, the last column reports the average results. It is important to note that all the results are reported in BPS.

Table 4.1: Experimental results achieved on the CT images (without the embedding of a watermark).

N	**CT_skull**	**CT_wrist**	**CT_carotid**	**CT_Aperts**	***Average***
3	2.1039	1.0599	1.4318	0.8296	*1.3563*
4	2.0720	1.0260	1.3942	0.8038	*1.3240*
5	2.1075	1.0351	1.4251	0.8289	*1.3492*
6	2.1168	1.0612	1.4447	0.8328	*1.3639*
7	2.1441	1.0733	1.4727	0.8515	*1.3854*
8	2.1563	1.0869	1.4880	0.8578	*1.3973*

Table 4.2: Experimental results achieved on the MR images (without the embedding of a watermark).

N	MR_liver_t1	MR_liver_t2e1	MR_sag_head	MR_ped_chest	*Average*
3	2.2727	1.8985	2.0755	1.6909	*1.9844*
4	2.2132	1.8670	2.0283	1.6706	*1.9448*
5	2.2587	1.8989	2.0448	1.6906	*1.9733*
6	2.2914	1.9165	2.0481	1.7203	*1.9941*
7	2.3243	1.9384	2.0625	1.7387	*2.0160*
8	2.3430	1.9572	2.0705	1.7530	*2.0309*

Figures 4.3 and 4.4 show the histogram related to the average BPS (on the X-axis), by considering the tested values of the N parameter (on the Y-axis) for the tested CT and MR images, respectively.

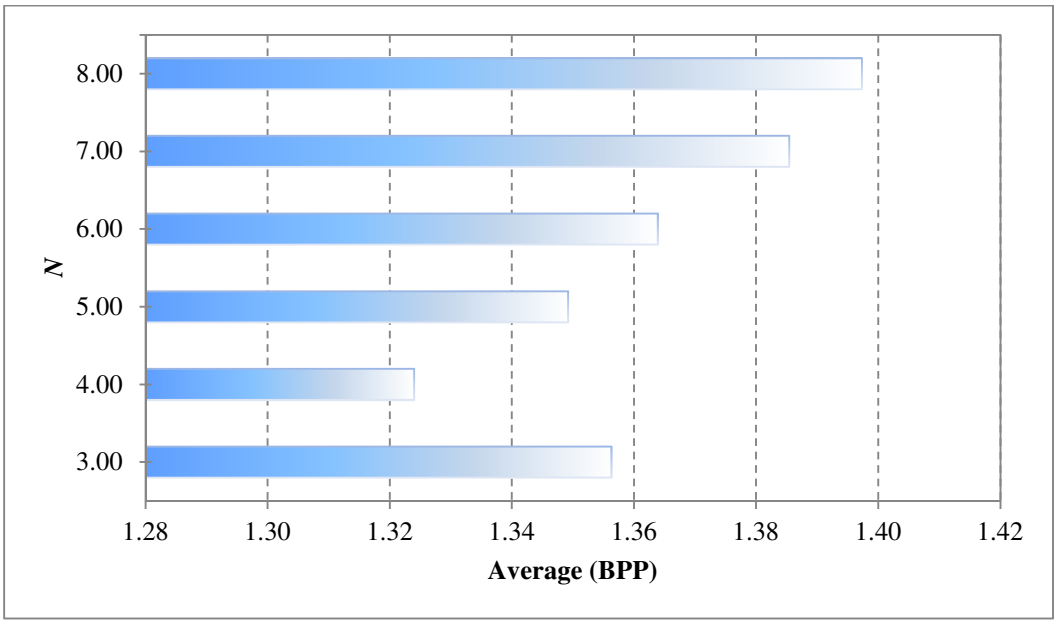


Figure 4.3: Histogram of the average results (CT images).

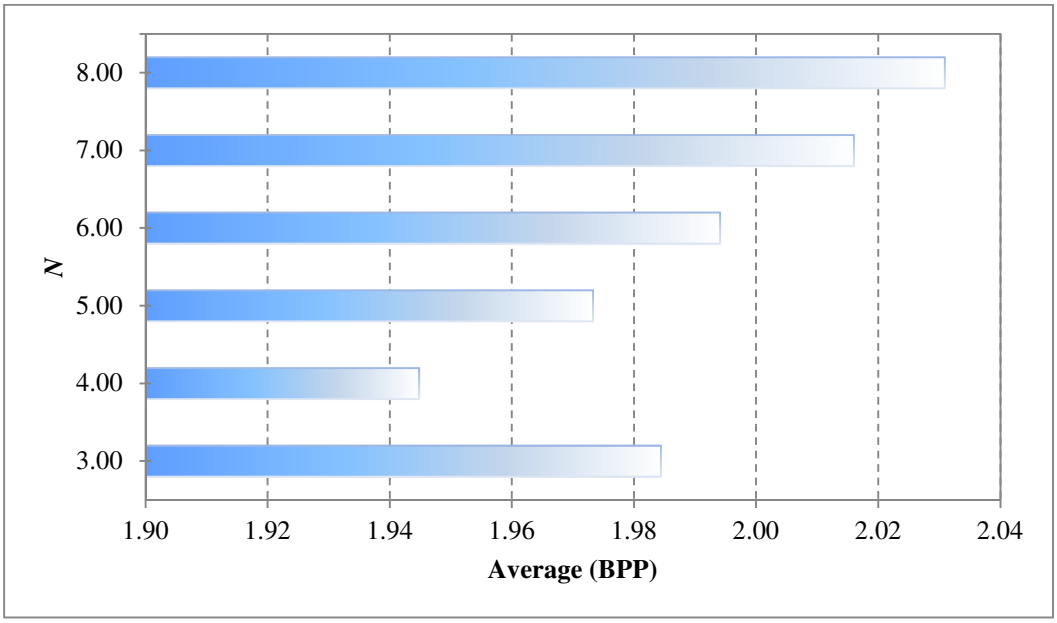


Figure 4.4: Histogram of the average results (MR images).

It is evident from the figures that the best results, for both types of 3-D medical imagery, are achieved when the N parameter is equal to 4. It should be observed that the results are slightly worse with respect to MILC. However, the proposed predictor is configurable and can be adapted to different hardware characterizations.

Furthermore, we performed experiments of our approach by considering the compression and simultaneous embedding of a watermark. In detail, for testing purposes, we used two different randomly generated watermark strings, composed of 100 and 200 bits, respectively. In addition, we set $N = 4$, $r = k - 1$ (where k is the currently analyzed slice) and the numeric *Pin* is set to 12345. The results, in terms of BPS, we achieved from such experiments are very similar to the ones obtained without the embedding of a watermark string.

It is important to note that once a medical image is affected by a watermark, is obviously slightly different with respect to the original. Therefore, we used the *Peak Signal-to-Noise Ratio* (PSNR) metric to measure the distortion between the original image and the watermarked one, obtained through the decompression process.

In Figures 4.5 and 4.6, we plot the histograms related to the slice-by-slice average PSNR. In detail, on the X-axis the CT and MR images are reported, while, on the Y-axis the average PSNR is reported. The cyan and red columns indicate the average (slice-by-slice) PSNR obtained, by comparing the original image with respect to the watermarked one, affected by the watermark of 100 bits and 200 bits, respectively.

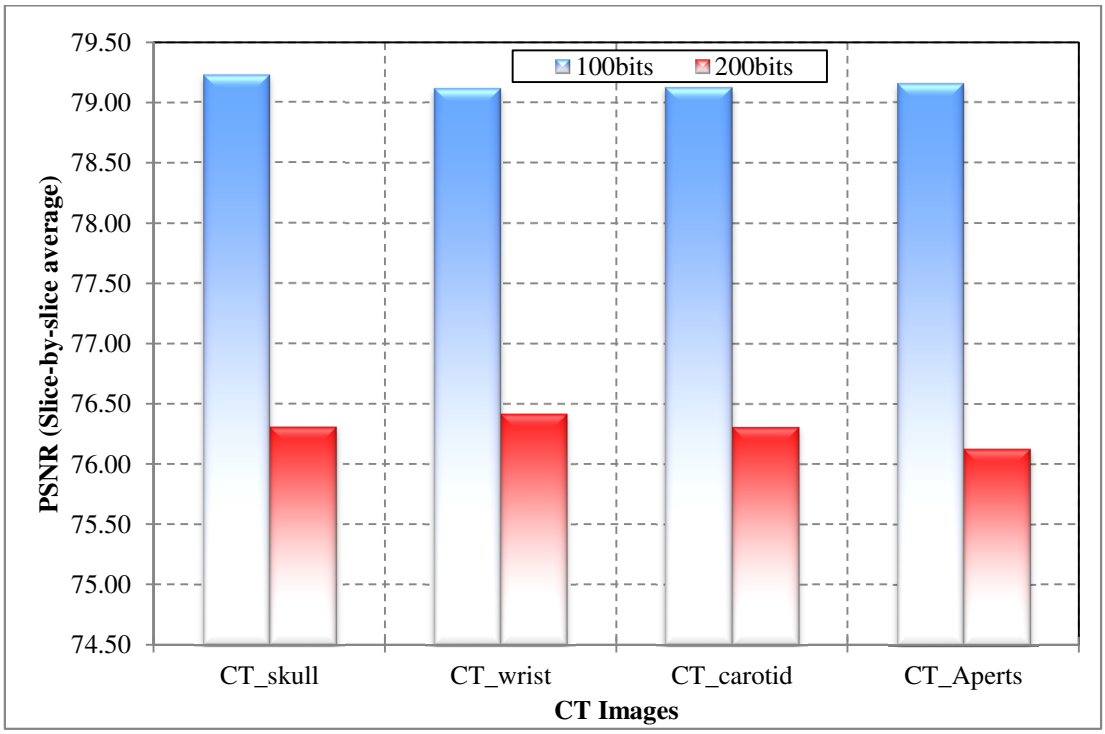


Figure 4.5: Histogram of the slice-by-slice average PSNR (MR images).

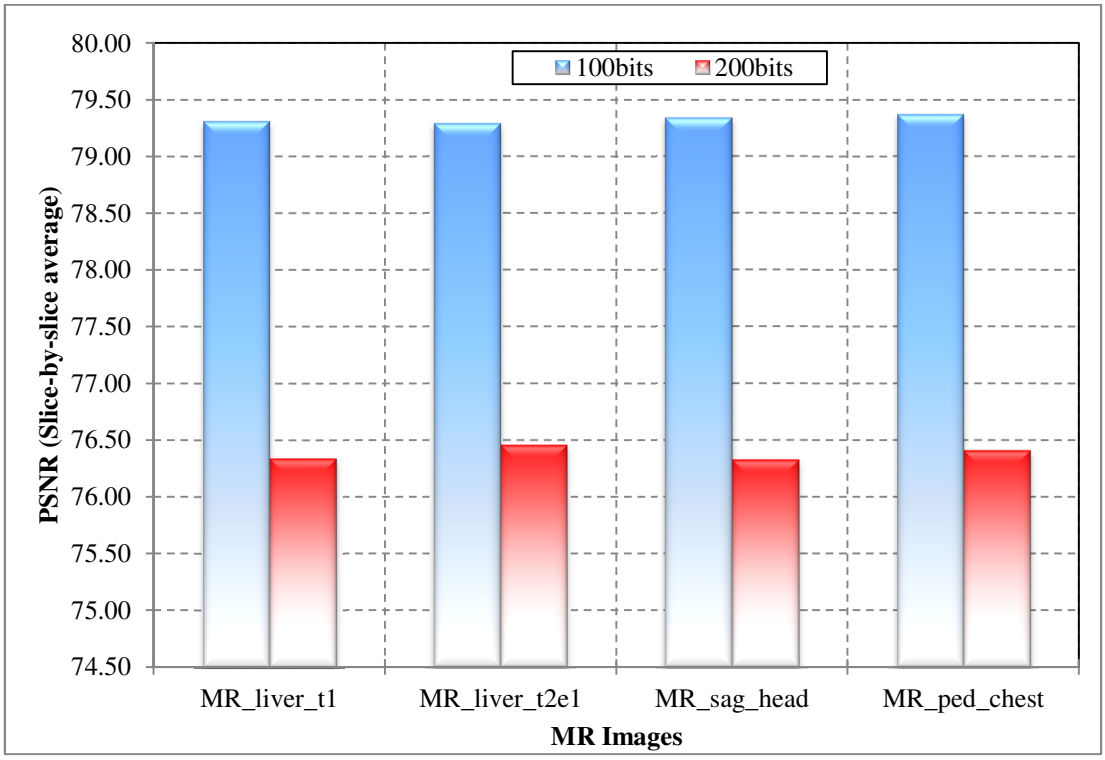


Figure 4.6: Histogram of the slice-by-slice average PSNR (MR images).

From the figures, it is possible to observe that the average PSNR is sufficiently high in all the cases. This aspect guarantees a high *fidelity* of the watermarked image, with respect to the original one. In particular, the average PSNR is about 79 and about 76 for the 3-D medical images in the case in which the embedded watermark is composed of 100 and 200 bits, respectively. It is important to note that, through our technique, the hidden information is not perceptible by humans. However, the end-users need to be conscious of such modifications, in order to extract the watermarked information.

4.3. Protection of 3-D Microscopy Images by using Digital Watermarking Methods

A 3-D microscopy image can be considered a 3-D datacube, composed of a collection of 2-D slices. Starting from the consideration that in these types of data, only a subset of slices can be relevant and extracted from the datacube, we observed that it is important to preserve the integrity of the whole 3-D image as well as of a single slices.

The approach we proposed permits to simultaneously protect both the characterizations of 3-D microscopy images, by embedding two invisible watermarks, one of them is embedded into the whole 3-D microscopy image, while the other one is independently embedded into each slice.

The design choices for our approach are outlined in order to permit an easy implementation as well as use less resources as possible, in terms of CPU computations and memory. In this manner, even devices with limited hardware capabilities (i.e., industrial devices, etc.) are suitable for the implementation of our approach.

4.3.1. The Embedding Procedure

In Algorithm 4.3, we highlight the key points related to the embedding procedures of our scheme. In detail, two invisible watermarks are embedded: a

3-D watermark, embedded in the whole 3-D microscopy image, and a 2-D watermark, independently embedded in each slice.

In particular, the former is intended to protect the whole 3-D microscopy image, while the latter is intended to independently protect each slice. The design choice, which considers the protection of each slice, is motivated by the fact that only a subset of slices might be important and, consequently, extracted from the whole 3-D microscopy image. These watermarks are embedded into the spatial domain and are sufficiently robust against lossy compression attacks. It is important to note that both of them can be extracted through a blind extraction approach, which permits, in different scenarios, to save storage space, by avoiding to memorize the original image.

As it is possible to observe from Figure 4.7, several procedures are used by the EMBEDDING procedure. Such a procedure takes the following parameters as input:

- MI: 3-D microscopy image;
- Key: Key used for the embedding of 3-D watermark;
- c: Parameter used for the embedding of the 3-D watermark;
- α, T: Parameters used for the embedding of the 2-D watermark.

First, in the whole MI a 3-D watermark is embedded, through the 3-D EMBEDDING procedure (described in Algorithm 4.4). It is important to point out that MI_{3D} is the output of the 3-D EMBEDDING procedure. Subsequently, each slice of MI_{3D} ($slice_{W3D}$) is extracted through the EXTRACTSLICE procedure. The EXTRACTSLICE procedure extracts a specific slice from MI_{3D}, we do not report its pseudo-code, since such a procedure is

often implemented by ad-hoc libraries for the management of microscopy images or its implementation can vary depending to the microscopy image format. Furthermore, by means of the 2-D EMBEDDING procedure (described in Algorithm 4.5), a 2-D watermark is embedded into the considered slice, $slice_{W3D}$. It is important to note that the 2-D EMBEDDING procedure outputs in $slice_{W3D_{W2D}}$. In the last step, the final output, $MI_{W3D_{W2D}}$, is returned by the EMBEDDING procedure.

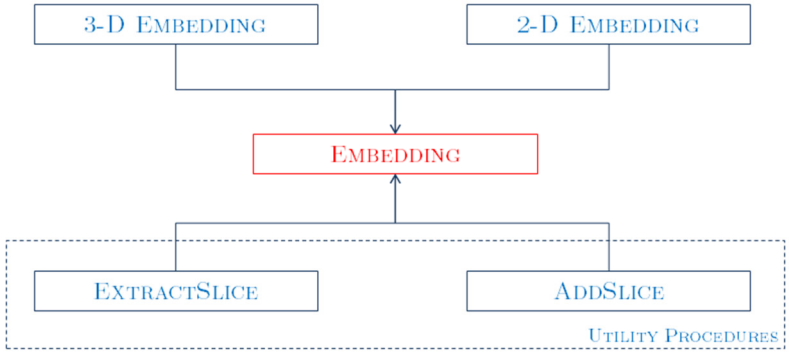


Figure 4.7: The used procedures by the EMBEDDING procedure.

Algorithm 4.3: The EMBEDDING procedure (from [**52**]).

1: **procedure** EMBEDDING(MI, Key, c, α, T)
2: MI_{W3D} = 3D Embedding(MI, Key, c);
3: **for** $sliceIdx = 0$ **to** $MI.NumOfSlices - 1$ **do**
4: Set $slice_{W3D}$ = ExtractSlice(MI_{W3D}, $sliceIdx$);
5: Set $slice_{W3D_{W2D}}$ = 2D Embedding($slice_{W3D}$, Key, α, T);
6: $MI_{W3D_{W2D}}$ = AddSlice($slice_{W3D_{W2D}}$, $sliceIdx$);
7: **end for**
8: **return** $MI_{W3D_{W2D}}$;
9: **end procedure**

4.3.2. The 3-D EMBEDDING procedure

The 3-D EMBEDDING procedure, highlighted in Algorithm 4.4, takes as input the following parameters:

- MI, Key: 3-D microscopy image and key used for the embedding of 3-D watermark;
- c: Parameter used for defining the robustness of the embedding.

In particular, the 3-D EMBEDDING procedure is composed by two main phases:

1. The *generation* phase;
2. The *embedding* phase.

Algorithm 4.4: The 3-D EMBEDDING procedure (from [**52**]).

procedure 3-D EMBEDDING(MI, Key, c)
    Convert all the pixels of MI from the *RGB* domain to the *YUV* domain;
    Set $sliceSize$ = $MI.Width \times MI.Height$;
    Set $NPixels$ = $sliceSize \times MI.NumOfSlices$;
    Generate a watermark string $ws \in \{1, -1\}^{NPixels}$, by using PRG and Key. ws will be embedded into MI.
    for each pixel $P(MI, x, y, s)$ of MI **do**
        Set $currentWSSym$ = ExtractSymbol(ws, x, y, s);
        Set Y($P(MI_{W3D}, x, y, s)$) = Y($P(MI, x, y, s)$) + $c \times currentWSSym$;
        Set U($P(MI_{W3D}, x, y, s)$) = U($P(MI, x, y, s)$);
        Set V($P(MI_{W3D}, x, y, s)$) = V($P(MI, x, y, s)$);
    end for
    Convert all the pixels of MI_{W3D} from the *YUV* domain to the *RGB* domain;
    return MI_{W3D};
end procedure

In the generation phase, the watermark is generated, by using a *Pseudo-Random Generator* (*PRG*) [**53**]. However, as explained in [**54**], a *Chaotic Map* can also be used.

Formally, let $l(\bullet)$ be a polynomial and let G be a *deterministic polynomial-time algorithm*. In detail, for any input string $s \in \{0,1\}^n$ the G algorithm outputs a string of $s' \in \{0,1\}^{\ln(n)}$. It is possible to say that G is a PRG if the following two conditions are held:

1. *Expansion*: $\forall n$ it holds that $l(n) > n$;
2. *Pseudorandomness*: For all probabilistic polynomial-time distinguishers D, there exists a negligible function *negl* such that: $|Pr[D(r) = 1] - Pr[D\big(G(s)\big) = 1]| \leq negl(n)$, where r is chosen uniformly from $\{0,1\}^{l(n)}$ and s is the seed, which is chosen uniformly at random from $\{0,1\}^n$. The probabilities are taken over the random coins used by D and the choice of r and s.

The function $l(n)$ is referred to as the "*expansion factor*" of G. Thus, starting from an initial "*seed*", through a PRG, it is possible to obtain an output sequence of a specified size. It is important to note that in the scheme we propose, the seed is represented by the input parameter key (i.e., *Key*). Furthermore, this output sequence is "*not invertible*", therefore, it is not possible to reconstruct the key even in the case of this sequence being known.

Regarding the 3-D EMBEDDING procedure, the watermark string is generated in order to take into account the whole MI: $ws \in \{-1,1\}^{NPixels}$ (where $NPixels$ is the number of the samples of MI).

In the embedding phase, the effective embedding of ws into MI is performed. In particular, all the samples of MI are converted from the RGB to the YUV format. After that, the luminance component (the Y component) of each sample is modified, according to the current symbol of ws. In detail, each pixel is modified, by adding the product between the current symbol of ws and c. In order to define what the current symbol of ws,is $currentWSSym$, it is important to specify an appropriate order, in which ws is considered (i.e. the raster scan order, etc.) [**54**].

It is important to note that the function P permits to obtain the triple related to the Y, U and V components, of a specified sample. The functions Y, U and V are used respectively for the extraction of the luminance and the chrominance components of a given triple.

4.3.3. The 2-D EMBEDDING procedure

In Algorithm 4.6, we outline the pseudo-code related to the 2-D EMBEDDING procedure. This procedure takes as input the following input parameters:

- *Slice*: Slice of a 3-D microscopy image;
- *Key*: Key used for the embedding of the 2-D watermark;
- α, T: Parameters used for defining the robustness of the embedding.

The 2-D EMBEDDING procedure is based on the procedures described in [**37**] and [**45**]. The basic idea behind such a procedure is that each symbol of the watermarks string, ws (generated through a PRG along with the relative Key), is diffused over the whole $Slice_{W2D}$. In detail, the first step is related to the copying of $Slice$ into $Slice_{W2D}$. $Slice_{W2D}$ will store the watermarked slice and will be returned at the end of the procedure. After that, all the sample of $Slice_{W2D}$ are converted from the RGB to the YUV domain.

Subsequently, each bit of ws will be embedded into each one of the pseudo-randomly selected 8×8 blocks. It is important to note that the JPEG lossy compression algorithm is "simulated", in order to improve the robustness of the watermark against lossy compression attacks. In particular, one of the aforementioned blocks is selected (denoted as B in Algorithm 4.5). Therefore, the quality of the block B is reduced, by using the REDUCEQUALITY procedure (explained in Algorithm 4.5). The REDUCEQUALITY procedure takes as input the block B and a quantization matrix Q and outputs a reduced-quality block, referred to as B^I. Subsequently, the product $\alpha \times B^R$, where B^R is a pseudo-randomly generated block through the GENERATEPSEUDORANDOMBLOCK procedure (of the same dimensions of B and each entry can assumes 0 or 1 as value) and α is an integer parameter (used for defining the robustness of the watermark), is added or subtracted, depending on the value of the bit. Until the specified threshold T is reached (if the bit assumes 1 as value) or $-T$ (if the bit assumes 0 as value), the described operations are repeated.

Algorithm 4.5: The 2-D EMBEDDING procedure (from [8]).

procedure 2-D EMBEDDING($Slice$, Key, α, T)
  Create a copy of $Slice$, denoted as $Slice_{W2D}$;
  Convert all the pixels of $Slice_{W2D}$ from the *RGB* domain to *YUV* domain;
  Generate a watermark string $ws \in \{0,1\}^{2DWatermarkLength}$, by using PRG and Key. The ws will be embedded into $Slice_{W2D}$;
  Subdivide $Slice_{W2D}$ into M blocks of 8×8 pixels;
  for idx = 0 **to** $2DWatermarkLength$ **do**
    $currentWSBit$ = ExtractBit(ws, idx);
    Select a block B from the M blocks by using PRG and Key;
    Set B^R = GeneratePseudoRandomBlock(Key);
    Set I_0 = Compute$I$(B, B^R, 0);
    Set I_1 = Compute$I$(B, B^R, 1);
    Set B^I = ReduceQuality(B, Q);
    Set $D = I_1 - I_0$;
    Set I_0^I = Compute$I$(B^I, B^R, 0);
    Set I_1^I = Compute$I$(B^I, B^R, 1);
    Set $D^I = I_1^I - I_0^I$;
    if $currentWSBit$ == 1 **then**
      if $D < T$ **or** $D^I < T$ **then**
        B = B + $\alpha \times B^R$;
        The steps from 10 to 29 are repeated until $D > T$ and $D^I > T$;
      end if
    else
      if $currentWSBit$ == 0 **then**
        if $D > -T$ **or** $D^I > -T$ **then**
          B = B - $\alpha \times B^R$;
          The steps from 10 to 29 are repeated until $D < -T$ and $D^I < -T$;
        end if
      end if
    end if
  end for
  Convert all the pixels of $Slice_{W2D}$ from the *YUV* domain to the *RGB* domain;
  return $Slice_{W2D}$;
end procedure

Algorithm 4.6: The ReduceQuality procedure (from [8]).

```
1: procedure ReduceQuality(B, Q)
2:     B_Coefficients = DCT(B);
3:     B_QuantizedCoeffs = Quantize(B_Coefficients, Q);
4:     B_InverseQuantizedCoeffs = InverseQuantize(B_QuantizedCoeffs, Q);
5:     B^I = InverseDCT(B_InverseQuantizedCoeffs);
6:     return B^I
7: end procedure
8: procedure ComputeI(B, B^R, value)
9:     Set sumValues = 0;
10:    Set numValues = 0;
11:    for x to 8 do
12:        for y to 8 do
13:            if B^R[x][y] == value then
14:                Set sumValues = sumValues + Y(P(B, x, y));
15:                Set numValues = numValues + 1;
16:            end if
17:        end for
18:    end for
19:    Set I = sumValues / numValues;
20:    return I;
21: end procedure
```

4.3.4. The Detection Procedures

The 3-D DETECTION procedure is outlined in Algorithm 4.7. In particular, the procedure takes the following parameters as input:

- MI_{W3D}: 3-D microscopy image (in which embed a 3-D watermark is embedded through the EMBEDDING procedure);
- Key: Key used by the EMBEDDING procedure.

Basically, as can be noted by its name, the procedure is delegated to detect the 3-D watermark, by considering the whole MI_{W3D}.

In particular, once all the samples of MI_{W3D} are converted from the RGB domain to the YUV domain, the procedure generates the watermark string, ws again. Clearly, the watermark string is the same as the one generated through the EMBEDDING procedure (in presence of the same Key).

Algorithm 4.7: The 3-D DETECTION procedure (from [**8**]).

```
1: procedure 3-D DETECTION(MI_W3D, Key, 3-D_Threshold)
2:     Set R = ComputeR(MI_W3D, Key);
3:     if R ≥ 3-D_Threshold then
4:         return true;
5:     else
6:         return false;
7:     end if
8: end procedure
```

Algorithm 4.8: The COMPUTER procedure (from [**8**]).

procedure COMPUTER(MI_{W3D}, Key)
    Convert all the pixels of MI_{W3D} from the *RG*
domain to the *YUV* domain;
    Set $sliceSize = MI_{W3D}.Width \times MI_{W3D}.Height$
    Set $NPixels = sliceSize \times MI_{W3D}.NumOfSlices$;
    Generate a watermark string $ws \in \{1, -1\}^{NPixels}$, b
using PRG and Key. The ws should be embedded int
MI_{W3D}, by the *Embedding* procedure;
    Set $sum_{S0} = 0$, $sum_{S1} = 0$, $N_{S0} = 0$, $N_{S1} = 0$;
    for *each* pixel $P(MI_{W3D}, x, y, s)$ of MI_{W3D} **do**
        Set $currentWSSym$ = ExtractSymbol(ws, x, y, s);
        if $currentWSSym == 1$ **then**
            Set $sum_{S0} = sum_{S0}$ + Y($P(MI_{W3D}, x, y, s)$);
            Set $N_{S0} = N_{S0} + 1$;
        else
            Set $sum_{S1} = sum_{S1}$ + Y($P(MI_{W3D}, x, y, s)$);
            Set $N_{S1} = N_{S1} + 1$;
        end if
    end for
    Set $R = \frac{1}{N_{S0}} \times sum_{S0} - \frac{1}{N_{S1}} \times sum_{S1}$;
    return R;
end procedure

It should be noted that the 3-D DETECTION procedure uses the COMPUTER procedure. In particular, the COMPUTER procedure (outlined in Algorithm 4.8) is in charge of implementing the function R, defined in equation (4.4).

$$R(MI_{W3D}, ws) = \left(\frac{1}{|S_0|} \times \sum_{P(MI_{W3D},x,y,k)\in S_0} Y\big(P(MI_{W3D}, x, y, k)\big)\right) - \left(\frac{1}{|S_1|} \times \sum_{P(MI_{W3D},x,y,k)\in S_1} Y(P(MI_{W3D}, x, y, k))\right) \quad (4.4)$$

S_0 and S_1 are two sets, explained through the following equations, respectively:

$$S_0 = \{P(MI_{W3D}, x, y, s) \mid ExtractSymbol(ws, x, y, s) == 1\}$$
$$S_1 = \{P(MI_{W3D}, x, y, s) \mid ExtractSymbol(ws, x, y, s) == -1\}$$

The function permits to detect the presence of a watermark or not. As explained in [**54**], if the output of the R function is nearly zero ($R(\bullet) \approx 0$), the watermark is not detected. On the other hand, if the output of the function is nearly the double of the c parameter ($R(\bullet) \approx 2 \times c$), the watermark is detected.

In our 3-D Detection procedure, in order to verify if a watermark is detected or not, we use a threshold (denoted as $3-D_{Threshold}$). In particular, if the output of the ComputeR procedure is greater than the $3-D_{Threshold}$, the watermark is detected (in this case the procedure returns *true*). Otherwise, the watermark is not detected (in this case the procedure returns *false*).

Algorithm 4.9 outlines the pseudo-code of the 2-D Extraction procedure, which takes the following parameters as input:

- $Slice_{W2D}$: Slice of a 3-D microscopy image, watermarked through the Embedding procedure;
- *Key*: Key used for the embedding.

The 2-D Extraction procedure extracts the 2-D watermark from a watermarked slice ($Slice_{W2D}$). In particular, all the samples of $Slice_{W2D}$ are

converted from the RGB to the YUV domain. Analogously to the 2-D EMBEDDING procedure, a sequence of 8×8 blocks is pseudo-randomly generated, by using a PRG coupled with the input key, Key. Once a block B is selected, the block B^R (of the same dimensions of B) is pseudo-randomly generated (through the GENERATEPSEUDORANDOMBLOCK procedure). By considering the blocks B and B^R, the I^1 and I^0 averages and their difference, D, are computed. D is used for the extraction of the bit of the watermark string, which is embedded into B. In this case, if D is greater than zero, the bit has 1 as a value. On the other hand, if D is less or equal than zero, the bit has 0 as a value. Finally, the extracted watermark string (referred to as $extratecedWS$) is returned.

Through the 2-D DETECTION procedure (Algorithm 4.10), the watermark string is extracted from a watermarked input slice ($Slice_{W2D}$), through the 2-D EXTRACTION procedure. Afterwards, by considering the sim metric (defined in the equation (4.1)), the extracted watermark is compared with the embedded one, which is generated again. Finally, if the result of the comparison lies within the $2 - D_{Threshold}$, then $true$ is returned by the procedure (the watermark is detected). Otherwise, $false$ is returned, the watermark is not detected.

Algorithm 4.9: The 2-D EXTRACTION procedure (from [8]).

procedure 2-D EXTRACTION($Slice_{W2D}$, Key)
    Convert all the pixels of $Slice_{W2D}$ from the *RGB* domain to the *YUV* domain;
    Subdivide $Slice_{W2D}$ into M blocks of 8×8 pixels;
    for idx = 0 **to** $2DWatermarkLength$ **do**
        Select a block B from the M blocks by using PRG and Key;
        Set B^R = GeneratePseudoRandomBlock(Key);
        Set I_0 = Compute$I(B, B^R, 0)$;
        Set I_1 = Compute$I(B, B^R, 1)$;
        Set $D = I_1 - I_0$;
        if $D > 0$ **then**
            Set $extractedWS[idx]$ = 1;
        else
            Set $extractedWS[idx]$ = 0;
        end if
    end for
    return $extractedWS$;
end procedure

Algorithm 4.10: The 2-D DETECTION procedure (from [8]).

procedure 2-D DETECTION($Slice_{W2D}$, Key, $2-D_{Threshold}$)
    Set $extractedWS$ = 2-D Extraction($Slice_{W2D}$, Key);
    Generate a watermark string $ws \in \{0,1\}^{2DWatermarkLength}$, by using PRG and Key. The ws should be embedded into $Slice_{W2D}$, by the *Embedding* procedure;
    Set Ret = sim(ws, $extractedWS$);
    if $Ret \geq$ 2-D$_{Threshold}$ **then**
        return $true$;
    else
        return $false$;
    end if
end procedure

4.3.5. Experimental Results

We experimentally performed the testing phase of our method on a dataset composed of five 3-D confocal images of cells, which is described in Appendix A.2.

The parameters used in our experiments are the following:

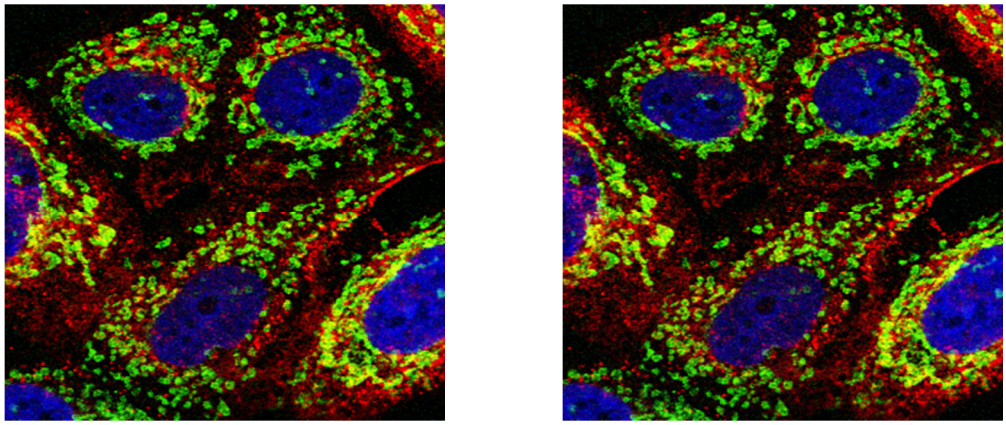
- $Key = 12345$;
- $c = 3$;
- $2DWatermarkLength = 20$;
- $\alpha = 8$;
- $T = 1$.

In Table 4.3, we report the obtained slice-by-slice average PSNR value (second column), for each one of the images (first column).

Table 4.3: The PSNR achieved between the watermarked and the original microscopy image.

Images	**PSNR**
Image1	40.51
Image2	40.89
Image3	40.35
Image4	40.54
Image5	40.64

Figure 4.8.a reports a portion of a slice of a tested microscopy image, while, Figure 4.8.b reports the same portion but of a watermarked microscopy image.



(a) (b)

Figure 4.8: A portion of a slice extracted from (a) an unaltered microscopy image and from (b) a watermarked and JPEG compressed microscopy image.

Furthermore, all the slices of the evaluated images are compressed by using the JPEG algorithm, with the *quality factor* parameter equal to 100%. Thus, for each 3-D microscopy image, we tried to extract the watermark related to the whole image, by using the 3-D Detection procedure, as well as from each slice composing the image, by using the 2-D Detection procedure. In particular, in the testing, we used the following parameters:

- $2 - D_{Threshold} = 0.75$;
- $3 - D_{Threshold} = 3$ (the same of the c parameter).

In all the cases, as can be seen in Table 4.4, the watermark is successfully detected.

In Figure 4.9, we plot the results achieved by observing the behavior of the 3-D Detection procedure, which is applied to each one of the evaluated images (reported on the X-axis). It is important to note that on the Y-axis the value is the value of the variable R (the light blue point), returned by the

execution of the COMPUTER procedure (Algorithm 4.8). In particular, the value of R is graphically compared with $3 - D_{Threshold}$ (the green dotted line).

From the figure, it is noticeable that the difference between the variable R and the $3 - D_{Threshold}$ varies by about 0.50 to 1.00. It is important to consider that by setting a $3 - D_{Threshold}$ to a value smaller than c, the 3-D watermark can be made robust even when the JPEG compression quality factor decreases.

Table 4.4: Achieved results for the DETECTION procedures.

Images	**2-D Detection**	**3-D Detection**
Image1	*Detected in all the slices*	*Detected*
Image2	*Detected in all the slices*	*Detected*
Image3	*Detected in all the slices*	*Detected*
Image4	*Detected in all the slices*	*Detected*
Image5	*Detected in all the slices*	*Detected*

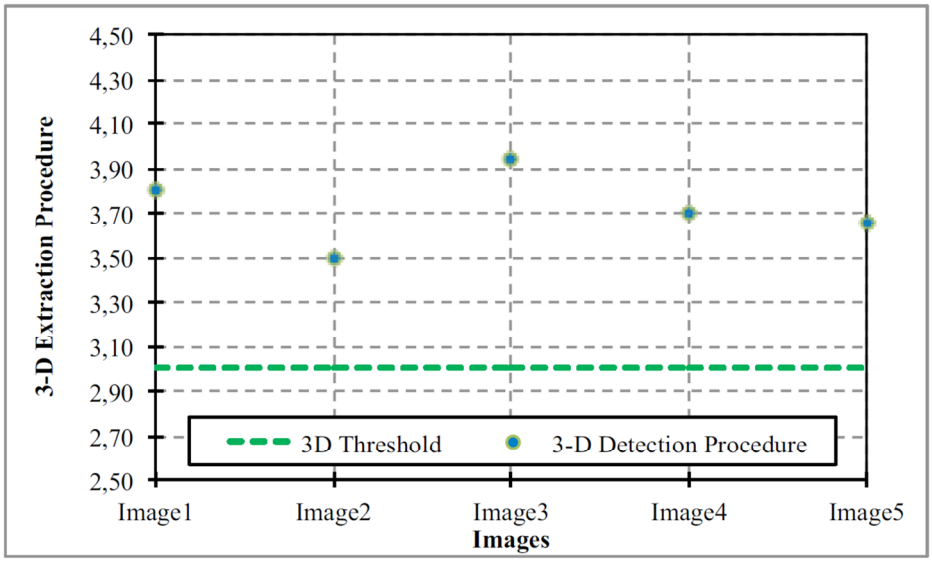


Figure 4.8: Graphical representation of the 3-D EXTRACTION Procedure results (from [**8**]).

Chapter – 5

Lossless Compression of Multidimensional Data

Highlights of the chapter

In this chapter, we focus on the description of a predictive structure well suited for the efficient and lossless compression of multidimensional data. As previously mentioned, a multidimensional dataset can be defined as a N-dimensional (with $N \geq 3$) collection of highly-related 2-D components. It is important to highlight that a component can be an image, a data matrix, etc..

We implemented our approach and experimentally tested its effectiveness, by considering different scenarios for the coding of prediction errors. In our experiments, we consider different types of N-data: 3-D medical images, hyperspectral images and 5-D functional Magnetic Resonance Images (fMRI).

5.1. The Predictive Structure for Multidimensional Data

The predictive model we propose is based on the least squares optimization technique. In particular, in order to perform the prediction of the current sample, a prediction context, composed of the neighboring samples of the current component and one (or more) reference component(s) is used. It is important to point out that that the reference component(s) can be of different dimension(s), with respect to the current component. Thus, the prediction is performed by using a multidimensional prediction context.

5.1.1. Definitions and Notations

Without loss of generality, for the following definitions, we assume that the N-D dataset, which we have to compress, has the following size $< D_1, D_2, \dots, D_{N-2}, X, Y >$. In addition, the sample that is under process and will be predicted (referred to as the *current sample*) has the following coordinates $(d_1, d_2, \dots, d_{N-2}, x, y)$ and the component to which it belongs (referred to as the *current component*) is identified through the following vector $[d_1, d_2, \dots, d_{N-2}]$ (where $1 \leq d_i \leq D_i$, $1 \leq x \leq X$ and $1 \leq y \leq Y$).

Example 5.0: Assuming that we have a 3-D dataset (where $N = 3$) of size $< 30, 100, 100 >$, where $M_1 = 30$ (Z dimension), $X = 100$ and $Y = 100$. Furthermore, the current sample has $(10, 20, 30)$ as coordinates and the current

component is identified through the vector [10]. The characterization of this dataset, the current component and the current reference are used in the following examples of this section. □

As outlined above, our predictive model uses one or more reference components, which will be specified through the *Sets of References*, described in Definition 5.1.

Definition 5.1 ***(Sets of References)***: A *Set of References* is denoted for each one of the $N-2$ dimensions. In particular, the Set of References related to the i-th dimension is denoted as $R_i = \{r_1^i, r_2^i, \dots, r_{t_i}^i\}$, where $r_j^i \in \{1, 2, \dots, D_i\} \cup \{-1, -2, \dots, -D_i\}$, $t_i = |R_i|$ and $1 \leq j \leq t_i$. It is important to point out that $\left|\bigcup_{i=1}^{N-2} R_i\right| > 0$ is always verified. Thus, at least one of the Sets of References should be not empty.

In order to univocally associate a component of the multidimensional dataset to an element of a Set of References, we adopt the following notation: the reference component is identified through the vector $[d_1, d_2, \dots, d_{i-1}, r_j^i, d_{i+1}, d_{N-2}]$ when $r_j^i > 0$, while it is identified through the following vector $[d_1, d_2, \dots, d_{i-1}, d_i - |r_j^i|, d_{i+1}, d_{N-2}]$ when $r_j^i < 0$ (where $r_j^i \in R_i,\ 1 \leq i \leq N-2$). It is important to observe that in both cases, the vectors related to the reference components are obtained by the vector that identifies the current component. □

We can highlight the concepts related to the Sets of References by reporting three examples, Examples 5.1.a, 5.1.b and 5.1.c. These examples show the effective use of the sets in three scenarios, in which the characterizations of the dataset, the current sample and component, specified in Example 5.0, are used. In detail, Example 5.1.a reports a simple instance in which all the elements of the Set of References, R_1 (related to the M_1 dimension), are less than zero, and Example 5.1.b reports a similar scenario in which all the elements of R_1 are greater than zero. Whereas, Example 5.1.c reports a scenario in which some elements of R_1 are less than zero and some elements are greater than zero.

*Example 5.1.*a: For example, if we set up the Set of References, related to the M_1 dimension, as $R_1 = \{-1, -2, -3\}$, it means that for the prediction, the reference components identified respectively through the vector [9], [8] and [7] are used. It is important to note that such vectors are obtained through the elements of the R_1 set. In particular, the vector [9] is obtained as $[10 - |-1|]$, by using the first element of R_1 (which is $-1 < 0$). Analogously, the vectors [8] and [7] are obtained by using respectively the second, -2 (< 0), and the third element, -3 (< 0), of R_1. □

Example 5.1.b: Assuming that we set up the Set of References, for the M_1 dimension, as $R_1 = \{5, 6\}$. In this scenario, the reference components used for the prediction are respectively identified through the vectors [5] and [6], which are obtained respectively by using the first element, 5 (> 0), and the second element, 6 (> 0), of the R_1 set. □

Example 5.1.c: For example, consider a Set of References defined as $R_1 = \{-1, 3, 7\}$. In this scenario, the prediction context is formed by using the reference components identified through the vectors [9], [3] and [7], which are obtained respectively by using the elements of R_1: -1 (< 0), 3 (> 0) and [7] (> 0). □

In order to refer to a sample without the use of its coordinates, we define an enumeration (Definition 5.2). Its main objective is the relative indexing among all the samples (or a subset of them) of the same component. In particular, by fixing a sample, namely the *reference sample*, all the other samples of the component will be indexed with respect to it. Therefore, it is possible to address a sample by using its relative index. The relative indexing of the samples is used for the definition of the multidimensional prediction context involved in our predictive model. In Example 5.2, we consider an example of a simple enumeration, which uses the current sample as a reference sample.

Definition 5.2 (*Enumeration*): $E^{(a_1,a_2,\dots,a_{N-2},l,m)}$ is a bi-dimensional enumeration that is used to define a relative integer indexing of the neighboring samples with respect to a specified *reference sample*, which have $(a_1, a_2, \dots, a_{N-2}, l, m)$ as coordinates (where $1 \leq a_i \leq D_i$, $1 \leq l \leq X$, $1 \leq m \leq Y$ and $1 \leq i \leq N-2$). It is important to point out that the neighboring and reference sample can belong to the same component. In detail, an enumeration needs to satisfy two requirements:

- The reference sample has 0 as an index;
- If two samples have different coordinates, then they have different enumeration indices. □

Example 5.2: In Figure 5.1, we graphically report an example of the enumeration $E^{(10,20,30)}$ (for the first three samples), which uses the current sample as the reference sample (identified by its relative index equal to zero and highlighted in the figure). According to the enumeration, for example, the sample that has $(10, 19, 30)$ as coordinates, can also be referred to its relative index, 1, with respect to the current sample. Analogously, the samples with coordinates $(10, 20, 29)$ and (10,19,29) can be identified by their relative indices (respectively of 2 and 3), with respect to the current sample. □

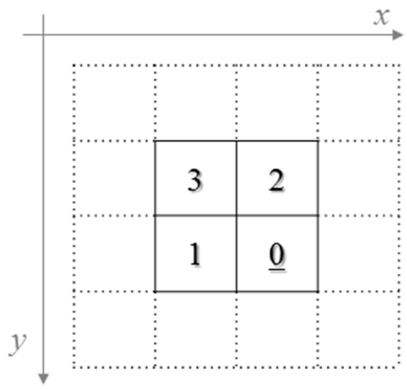


Figure 5.1: A graphical example of an enumeration.

Through Definitions 5.3.a and 5.3.b, we introduce the formal notations by means of which it is possible to refer to samples by using their relative indexing according to an enumeration E, with respect to a reference sample. In

Example 5.3, we focus on these latter definitions and report an example of their use.

Definition 5.3.a: Let $x_j^{(e)}(r_s^j)$ (where $r_s^j \in R_j$) identifies the sample that have e as an index, according to the enumeration $E^{\left(d_1,d_2,\ldots,d_{j-1},r_s^j,d_{j+1},\ldots\ d_{N-2},x,y\right)}$ when $r_s^j > 0$, and according to the enumeration $E^{\left(d_1,d_2,\ldots,d_{j-1},r_s^j,d_{j+1},\ldots\ d_{N-2},x,y\right)}$ when $r_s^j < 0$. It is important to emphasize that, in both cases, the coordinates of the reference samples related to the enumerations are obtained by using the coordinates of the current sample. □

Definition 5.3.b: Let $x^{(e)}$ identifies the sample that has e as an index, according to the enumeration $E^{(d_1,d_2,\ldots\ d_{N-2},x,y)}$, where $(d_1, d_2, \ldots, d_{N-2}, x, y)$ are the coordinates of the current sample. It is worth noticing that $x^{(0)}$ precisely identifies the current sample, according to Definition 5.2. □

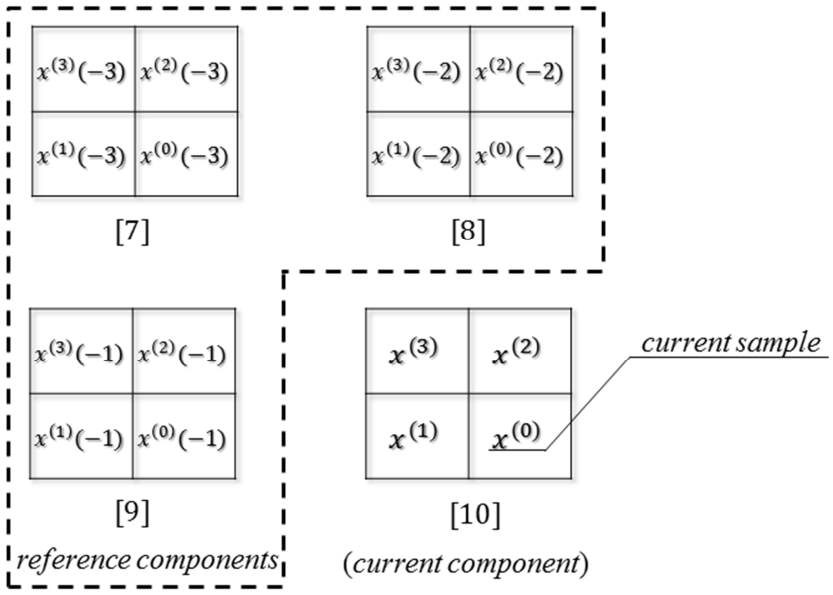


Figure 5.2: A graphical example of the use of an enumeration.

Example 5.3: In Figure 5.2, we graphically show an explicative scenario, in which the use of Definitions 5.3.a and 5.3.b is involved. In particular, we consider the enumeration $E^{(10,20,30)}$, graphically defined in Example 5.2, and the Sets of References, specified in Example 5.1.a.

It is possible to observe from Figure 5.2, $x^{(0)}(-3)$ refers to the sample that has the same spatial coordinates, x and y (respectively of 20 and 30), of the current sample, in the component identified through the vector [7]. In detail, the coordinates of $x^{(0)}(-3)$ are $(7, 20, 30)$.

Whereas, as can be noted in the figure, $x^{(1)}$ addresses a sample that is a neighbor of the current sample, in the same component. In particular, the coordinates of $x^{(1)}$ are $(10, 19, 30)$. □

5.2. The Predictive Structure

According to the definitions and notations of Section 2.1 and Section 5.1, we introduce how the prediction is performed.

Analogously to Section 2.1.1 and without loss of generality, we assume that we have a N-D dataset of size $< D_1, D_2, \dots, D_{N-2}, X, Y >$, the current sample (referred to as $x^{(0)}$) has $(d_1, d_2, \dots, d_{N-2}, x, y)$ as coordinates and then the current component is identified through the vector $[d_1, d_2, \dots, d_{N-2}]$ (where $1 \leq d_i \leq D_i$, $1 \leq x \leq X$ and $1 \leq y \leq Y$).

Furthermore, assuming that the Sets of References and the enumeration $E^{(d_1,d_2,\dots,d_{N-2},x,y)}$ are previously defined. In particular, both, which can be user-specified and need to be set up before computing the prediction.

In detail, the T-order prediction is performed by means of the equation (5.1), in which $\widehat{x}^{(0)}$ denotes the prediction of the current sample (where $T = \sum_{i=1}^{N-2} t_i = \sum_{i=1}^{N-2} |R_i|$). It is important to point out that T indicates the number of the reference components which are involved in the prediction.

$$\widehat{x}^{(0)} = \sum_{i=1}^{N-2} \sum_{r_j^i \in R_i} \alpha_i^j x_i^{(0)}(r_j^i) \tag{5.1}$$

In detail, the coefficients $\alpha_0 = \left[\alpha_1^1, \dots, \alpha_1^{t_1}, \dots, \alpha_i^1, \dots, \alpha_i^{t_i}, \dots, \alpha_{N-2}^1, \dots, \alpha_{N-2}^{t_{N-2}}\right]^T$ are chosen with the aim to minimize the energy of the prediction error, which is obtained by means of the equation (5.2).

$$P = \sum_{i=1}^{H} (x^{(i)} - \hat{x}^{(i)})^2 \tag{5.2}$$

It is important to point out that the H parameter denotes the number of samples for each one of the reference components and the current component, which will be used for the prediction.

Similarly to [**9**], the optimal coefficients, α_0, are obtained through the optimal linear prediction method and, in particular, as previously mentioned, through the least squares optimization technique. In detail, it is possible to rewrite the equation (5.2) in the form of the equation (5.3), by using the matrix notation. $\boldsymbol{C}$ and $\boldsymbol{X}$, which are respectively a matrix and a vector, are defined by means of the equation (5.4).

$$(\boldsymbol{C}^T \boldsymbol{C})\alpha_0 = (\boldsymbol{C}^T \boldsymbol{X}) \tag{5.3}$$

$$\boldsymbol{C} = \begin{bmatrix} x_1^{(1)}(r_1^1) & \cdots & x_1^{(1)}(r_{t_1}^1) & \cdots & x_i^{(1)}(r_1^i) & \cdots & x_i^{(1)}(r_{t_i}^i) & \cdots & x_{N-2}^{(1)}(r_1^{N-2}) & \cdots & x_{N-2}^{(1)}(r_{t_{N-2}}^{N-2}) \\ \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots \\ x_1^{(H)}(r_1^1) & \cdots & x_1^{(H)}(r_{t_1}^1) & \cdots & x_i^{(H)}(r_1^i) & \cdots & x_i^{(H)}(r_{t_i}^i) & \cdots & x_{N-2}^{(H)}(r_1^{N-2}) & \cdots & x_{N-2}^{(H)}(r_{t_{N-2}}^{N-2}) \end{bmatrix},$$

$$\boldsymbol{X} = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(H)} \end{bmatrix} \tag{5.4}$$

It is worth noting that by computing the vector α_0, which solves the system of linear equations (5.3), is possible to obtain the prediction of the current sample $\hat{x}^{(0)}$, through the equation (5.1). It is important to outline that $H \times (T+1) + H$ samples are used to perform the prediction. In all the cases,

if our predictive structure involves only by past information, there is no need to send any side information to the decompression algorithm.

5.3. Complexity Analysis

The computational complexity of the prediction is related to the two configurable parameters: H and the Sets of References. In detail, it is possible to model the multidimensional prediction context, by specifying its wideness and the number of the reference components. Thus, it is possible to define a prediction context in order to minimize as much as possible the use of the computational resources or refine the accurateness of the prediction, by using more computational resources. It is important to take into account that, in the first scenario, our approach is suitable even in scenarios in which the resources can be limited or restricted.

However, it is important to point out that the most important computational costs are due to the linear system of the equation (5.3), in Section 5.2. There are many techniques that are able to solve linear systems. For example, by using the *normal equation method* [**16**], our linear system can be solved through $(H + T/3) \times T^2$ operations.

5.4. The Exceptions

It is important to note that, in some situations, our predictive structure can be ineffective. In particular, when the linear system of equations (5.3) cannot be

solved, if it has no solutions. Furthermore, the linear system can have infinitely many solutions. In these scenarios, referred to as *exceptions*, the predictive structure is not able to perform the prediction.

In presence of a sample that cannot be predicted through the proposed predictive structure (because an exception is verified), an alternative predictive structure (for example, Median Predictor, etc.) can be used.

Furthermore, it is important to take into account that our predictor needs to use at least one reference component. The samples that belong to a component with no reference component (for example, the first component of a multidimensional dataset) need to be predicted by using an appropriate bi-dimensional predictive structure.

5.5. Error Modeling and Coding

The prediction error (or prediction residual) related to the current sample is obtained by means of the difference between the current sample, $x^{(0)}$, and its prediction, $\widehat{x}^{(0)}$, by means of the equation (5.5).

$$e^{(d_1,d_2,\ldots,d_{N-2})} = x^{(0)} - \widehat{x}^{(0)} \tag{5.5}$$

It should be noted that a prediction error can assume a positive or negative value. It can optionally use a mapping function [**18**], in order to have only non-negative values.

An example of mapping function, $m(e)$ outlined by the equation (5.6).

$$m(e) = \begin{cases} 2 \times |e| & if\ e > 0 \\ 2 \times |e| - 1 & otherwise \end{cases} \tag{5.6}$$

Finally, the error (or eventually the mapped error) is coded by using an entropy coder.

5.6. Experimental Results

In this section, we will report the experimental results achieved by using our predictive structure on different multidimensional datasets:

- 3-D Medical Images (Section 5.6.1);
- Hyperspectral Images (Section 5.6.2);
- 4-D and 5-D functional Magnetic Resonance Images – fMRI (Section 5.6.3).

For each dataset, we used different configurations in terms of the H parameters and the Sets of References.

In Figure 5.3, we graphically describe the enumeration E used, related to the first 32 samples with respect to the current sample, which is identifiable through its index equal to zero (in parenthesis).

Three different schemes are used for the coding of prediction errors:

- The *Prediction by Partial Matching with Information Inheritance* scheme (*PPMd* or *PPMII*);
- The *PAQ8* scheme [**17**];
- The *Arithmetic Coding* scheme (*AC*) [**55**].

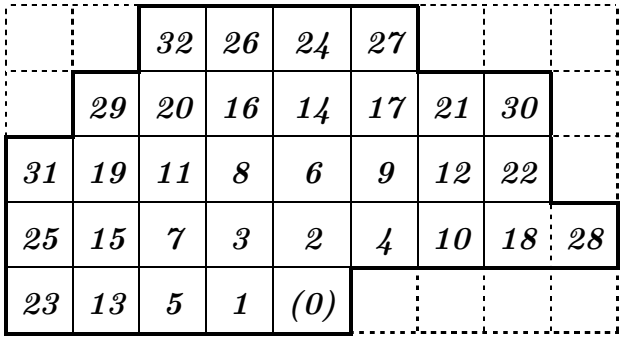


Figure 5.3: A graphical representation of the used enumeration.

5.6.1. 3-D Medical Images

We performed our experiments on a dataset composed of eight 3-D Medical Images, described in Appendix A.1.

In Tables 5.1 and 5.2, we respectively report the achieved experimental results on the CT data and MR data, by considering the PPMd encoding scheme (with default order equal to 4) for the coding of prediction errors. It is important to note that we experimentally tested our approach by using different values for the H parameter (i.e. 8, 16, 24 and 32) and several values for the Set of References. It should be noted that all the results are reported in *bits-per-sample* (*BPS*).

In particular, the first column indicates either the CT image (Table 5.1) or MR image (Table 5.2), from the second to the fifth columns report the achieved results for each of the tested Sets of Reference. The results are reported for each used value for the H parameter, which characterizes the vertical subdivision of the tables.

Table 5.1: Achieved results on the 3-D CT images. The coding of prediction errors is performed through the PPMd scheme.

CT Images	$\boldsymbol{R_Z = \{-1\}}$	$\boldsymbol{R_Z = \{-1,-2\}}$	$\boldsymbol{R_Z = \{-1,-2,-3\}}$	$\boldsymbol{R_Z = \{-1,-2,-3,-4\}}$
		$\boldsymbol{H = 8}$		
CT_Aperts	0.8507	0.7870	0.8140	0.8557
CT_carotid	1.4535	1.4208	1.4130	1.4849
CT_skull	2.1417	1.7159	1.7260	1.8255
CT_wrist	1.0958	1.0562	1.0521	1.1194
Average	1.3854	1.2450	1.2513	1.3214
		$\boldsymbol{H = 16}$		
CT_Aperts	0.8646	0.7768	0.7850	0.7988
CT_carotid	1.4770	1.4128	1.3650	1.3816
CT_skull	2.1552	1.6604	1.6237	1.6532
CT_wrist	1.1109	1.0129	0.9674	0.9770
Average	1.4019	1.2157	1.1853	1.2027
		$\boldsymbol{H = 24}$		
CT_Aperts	0.8705	0.7757	0.7788	0.7874
CT_carotid	1.4799	1.4046	1.3478	1.3552
CT_skull	2.1547	1.6350	1.5855	1.6018
CT_wrist	1.1144	0.9954	0.9429	0.9437
Average	1.4049	1.2027	1.1638	1.1720
		$\boldsymbol{H = 32}$		
CT_Aperts	0.8751	0.7778	0.7786	0.7836
CT_carotid	1.4850	1.4052	1.3455	1.3478
CT_skull	2.1603	1.6287	1.5735	1.5832
CT_wrist	1.1129	0.9895	0.9344	0.9314
Average	1.4083	1.2003	1.1580	1.1615

Table 5.2: Achieved results on the 3-D MR images. The coding of prediction errors is performed through the PPMd scheme.

MR Images	$\boldsymbol{R_Z = \{ -1 \}}$	$\boldsymbol{R_Z = \{ -1, -2 \}}$	$\boldsymbol{R_Z = \{ -1, -2, -3 \}}$	$\boldsymbol{R_Z = \{ -1, -2, -3, -4 \}}$
		$\boldsymbol{H = 8}$		
MR_liver_t1	2.2970	2.0224	2.0722	2.2136
MR_liver_t2e1	1.9721	1.4332	1.4240	1.5076
MR_ped_chest	1.6736	1.5245	1.5197	1.5969
MR_sag_head	2.0916	1.7127	1.7061	1.5969
Average	2.0086	1.6732	1.6805	1.7288
		$\boldsymbol{H = 16}$		
MR_liver_t1	2.3295	1.9804	1.9563	1.9977
MR_liver_t2e1	2.0014	1.4073	1.3619	1.3976
MR_ped_chest	1.6856	1.4587	1.3956	1.4091
MR_sag_head	2.0992	1.6750	1.6308	1.6563
Average	2.0289	1.6304	1.5862	1.6152
		$\boldsymbol{H = 24}$		
MR_liver_t1	2.3460	1.9713	1.9281	1.9450
MR_liver_t2e1	2.0118	1.3949	1.3423	1.3674
MR_ped_chest	1.6908	1.4350	1.3542	1.3529
MR_sag_head	2.1017	1.6561	1.6025	1.6194
Average	2.0376	1.6143	1.5568	1.5712
		$\boldsymbol{H = 32}$		
MR_liver_t1	2.3618	1.9731	1.9231	1.9298
MR_liver_t2e1	2.0186	1.3930	1.3366	1.3573
MR_ped_chest	1.6952	1.4282	1.3391	1.3318
MR_sag_head	2.1049	1.6477	1.5892	1.6031
Average	2.0451	1.6105	1.5470	1.5555

Similar to Tables 5.1 and 5.2, Tables 5.3 and 5.4 synthesize the experimental results achieved by using the PAQ8 algorithm for the coding of prediction errors.

Table 5.3: Achieved results on the 3-D CT images. The coding of prediction errors is performed through the PAQ8 scheme.

CT Images	$\boldsymbol{R_Z = \{-1\}}$	$\boldsymbol{R_Z = \{-1,-2\}}$	$\boldsymbol{R_Z = \{-1,-2,-3\}}$	$\boldsymbol{R_Z = \{-1,-2,-3,-4\}}$
		$\boldsymbol{H = 8}$		
CT_Aperts	0.7829	0.7268	0.7501	0.7889
CT_carotid	1.3838	1.3456	1.3376	1.4055
CT_skull	2.0291	1.6139	1.6191	1.7132
CT_wrist	1.0496	1.0066	0.9998	1.0612
Average	1.3114	1.1732	1.1767	1.2422
		$\boldsymbol{H = 16}$		
CT_Aperts	0.7968	0.7198	0.7261	0.7393
CT_carotid	1.4060	1.3417	1.2930	1.3084
CT_skull	2.0365	1.5645	1.5247	1.5525
CT_wrist	1.0645	0.9691	0.9244	0.9326
Average	1.3260	1.1488	1.1171	1.1332
		$\boldsymbol{H = 24}$		
CT_Aperts	0.8019	0.7187	0.7204	0.7291
CT_carotid	1.4075	1.3342	1.2767	1.2828
CT_skull	2.0330	1.5423	1.4901	1.5054
CT_wrist	1.0667	0.9539	0.9021	0.9017
Average	1.3273	1.1373	1.0973	1.1048
		$\boldsymbol{H = 32}$		
CT_Aperts	0.8063	0.7205	0.7198	0.7253
CT_carotid	1.4116	1.3343	1.2739	1.2755
CT_skull	2.0372	1.5366	1.4786	1.4880
CT_wrist	1.0646	0.9486	0.8935	0.8898
Average	1.3299	1.1350	1.0915	1.0947

Table 5.4: Achieved results on the 3-D MR images. The coding of prediction errors is performed through the PAQ8 scheme.

MR Images	$\boldsymbol{R_Z = \{ -1 \}}$	$\boldsymbol{R_Z = \{ -1, -2 \}}$	$\boldsymbol{R_Z = \{ -1, -2, -3 \}}$	$\boldsymbol{R_Z = \{ -1, -2, -3, -4 \}}$
		$\boldsymbol{H = 8}$		
MR_liver_t1	2.2013	1.9443	1.9870	2.1173
MR_liver_t2e1	1.8760	1.3437	1.3311	1.4102
MR_ped_chest	1.5801	1.4576	1.4577	1.5272
MR_sag_head	1.9606	1.5960	1.5888	1.6571
Average	1.9045	1.5854	1.5912	1.6780
		$\boldsymbol{H = 16}$		
MR_liver_t1	2.2304	1.9062	1.8823	1.9223
MR_liver_t2e1	1.9051	1.3196	1.2739	1.3072
MR_ped_chest	1.5869	1.3917	1.3406	1.3548
MR_sag_head	1.9676	1.5609	1.5179	1.5431
Average	1.9225	1.5446	1.5037	1.5319
		$\boldsymbol{H = 24}$		
MR_liver_t1	2.2436	1.8962	1.8547	1.8715
MR_liver_t2e1	1.9140	1.3092	1.2559	1.2794
MR_ped_chest	1.5898	1.3668	1.2985	1.2986
MR_sag_head	1.9697	1.5438	1.4903	1.5078
Average	1.9293	1.5290	1.4749	1.4893
		$\boldsymbol{H = 32}$		
MR_liver_t1	2.2568	1.8973	1.8485	1.8560
MR_liver_t2e1	1.9201	1.3079	1.2504	1.2697
MR_ped_chest	1.5932	1.3591	1.2822	1.2766
MR_sag_head	1.9720	1.5360	1.4768	1.4917
Average	1.9355	1.5251	1.4645	1.4735

Similar to Tables 5.3 and 5.4, Tables 5.5 and 5.6 report the experimental results achieved by using the AC encoding scheme for the coding of prediction errors.

Table 5.5: Achieved results on the 3-D CT images. The coding of prediction errors is performed through the AC scheme.

CT Images	$\boldsymbol{R_Z = \{-1\}}$	$\boldsymbol{R_Z = \{-1,-2\}}$	$\boldsymbol{R_Z = \{-1,-2,-3\}}$	$\boldsymbol{R_Z = \{-1,-2,-3,-4\}}$
		$\boldsymbol{H = 8}$		
CT_Aperts	1.3088	1.0223	1.0352	1.0808
CT_carotid	2.0882	1.8385	1.7573	1.8312
CT_skull	2.7651	1.9926	1.9539	2.0497
CT_wrist	1.6744	1.2757	1.2383	1.3023
Average	1.9591	1.5323	1.4962	1.5660
		$\boldsymbol{H = 16}$		
CT_Aperts	1.3772	1.0394	1.0195	1.0287
CT_carotid	2.1729	1.8861	1.7324	1.7432
CT_skull	2.8392	1.9747	1.8708	1.8897
CT_wrist	1.7779	1.2588	1.1709	1.1728
Average	2.0418	1.5398	1.4484	1.4586
		$\boldsymbol{H = 24}$		
CT_Aperts	1.4270	1.0579	1.0246	1.0239
CT_carotid	2.2181	1.9112	1.7304	1.7309
CT_skull	2.8808	1.9746	1.8472	1.8492
CT_wrist	1.8497	1.2591	1.1568	1.1481
Average	2.0939	1.5507	1.4398	1.4380
		$\boldsymbol{H = 32}$		
CT_Aperts	1.4638	1.0743	1.0327	1.0260
CT_carotid	2.2509	1.9311	1.7389	1.7349
CT_skull	2.9127	1.9824	1.8437	1.8380
CT_wrist	1.8937	1.2650	1.1553	1.1428
Average	2.1303	1.5632	1.4427	1.4354

Table 5.6: Achieved results on the 3-D CT images. The coding of prediction errors is performed through the AC scheme.

MR Images	$\boldsymbol{R_Z = \{ -1 \}}$	$\boldsymbol{R_Z = \{ -1, -2 \}}$	$\boldsymbol{R_Z = \{ -1, -2, -3 \}}$	$\boldsymbol{R_Z = \{ -1, -2, -3, -4 \}}$
		$\boldsymbol{H = 8}$		
MR_liver_t1	2.6977	2.0652	2.0787	2.2012
MR_liver_t2e1	2.5707	1.7326	1.6953	1.7867
MR_ped_chest	1.8374	1.5481	1.5341	1.6070
MR_sag_head	2.4853	1.8932	1.8606	1.9401
Average	2.3978	1.8098	1.7922	1.8838
		$\boldsymbol{H = 16}$		
MR_liver_t1	2.8301	2.0679	1.9993	2.0289
MR_liver_t2e1	2.6616	1.7243	1.6395	1.6766
MR_ped_chest	1.8979	1.4949	1.4203	1.4315
MR_sag_head	2.5419	1.8745	1.7881	1.8115
Average	2.4829	1.7904	1.7118	1.7371
		$\boldsymbol{H = 24}$		
MR_liver_t1	2.9242	2.0888	1.9897	1.9941
MR_liver_t2e1	2.7164	1.7303	1.6284	1.6520
MR_ped_chest	1.9418	1.4806	1.3850	1.3807
MR_sag_head	2.5843	1.8733	1.7668	1.7784
Average	2.5417	1.7933	1.6925	1.7013
		$\boldsymbol{H = 32}$		
MR_liver_t1	2.9963	2.1107	1.9959	1.9895
MR_liver_t2e1	2.7561	1.7414	1.6297	1.6476
MR_ped_chest	1.9723	1.4799	1.3735	1.3633
MR_sag_head	2.6168	1.8785	1.7610	1.7678
Average	2.5854	1.8026	1.6900	1.6921

5.6.1.1. Results Discussion

In this section we focus on the analysis of the experimental results, by considering the average results of the two whole datasets. In detail, in Tables 5.7 and 5.8, we outline the average results of all the CT images as well as all the MR images, respectively. The results are obtained by considering the PPMd scheme for the coding of prediction errors. Furthermore, the *2-D Linearized Median Predictor* (*2D-LMP*), described in Section 4.1.1, and the *3-D Distances-based Linearized Median Predictor* (*3D-DLMP*), described in Section 4.1.2, are used for the prediction of samples of the first slice and the management of the exceptions, respectively.

In particular, the first column indicates the H parameter and from the second to the forth columns show the average results for each tested Set of References, respectively.

Table 5.7: Average results on the CT Images. The coding of prediction errors is performed through the PPMd scheme.

H	$R_Z = \{ -1 \}$	$R_Z = \{ -1, -2 \}$	$R_Z = \{ -1, -2, -3 \}$	$R_Z = \{ -1, -2, -3, -4 \}$
8	1.3854	1.2450	1.2513	1.3214
16	1.4019	1.2157	1.1853	1.2027
24	1.4049	1.2027	1.1638	1.1720
32	1.4083	1.2003	1.1580	1.1615

Table 5.8: Average results achieved on the MR Images. The coding of prediction errors is performed through the PPMd scheme.

H	$R_Z = \{ -1 \}$	$R_Z = \{ -1, -2 \}$	$R_Z = \{ -1, -2, -3 \}$	$R_Z = \{ -1, -2, -3, -4 \}$
8	2.0086	1.6732	1.6805	1.7288
16	2.0289	1.6304	1.5862	1.6152
24	2.0376	1.6143	1.5568	1.5712
32	2.0451	1.6105	1.5470	1.5555

Figures 5.4 and 5.5 plot the information of Tables 5.7 and 5.8, respectively. In detail, on the X-axis, the tested Set of References are indicated, while on the Y-axis the average BPS is indicated. The blue, dark red, green and violet lines highlight the trend in correspondence to the average results when $H = 8$, $H = 16$, $H = 24$ and $H = 32$, respectively.

As can be noted in the figures, the best trend of the average results is delineated when the H parameter is equal to 32, except for the configuration in which $R_Z = \{-1\}$. In particular, the best results are obtained when the previous three slices are used (i.e. when the Set of References is defined as $R_Z = \{-1, -2, -3\}$).

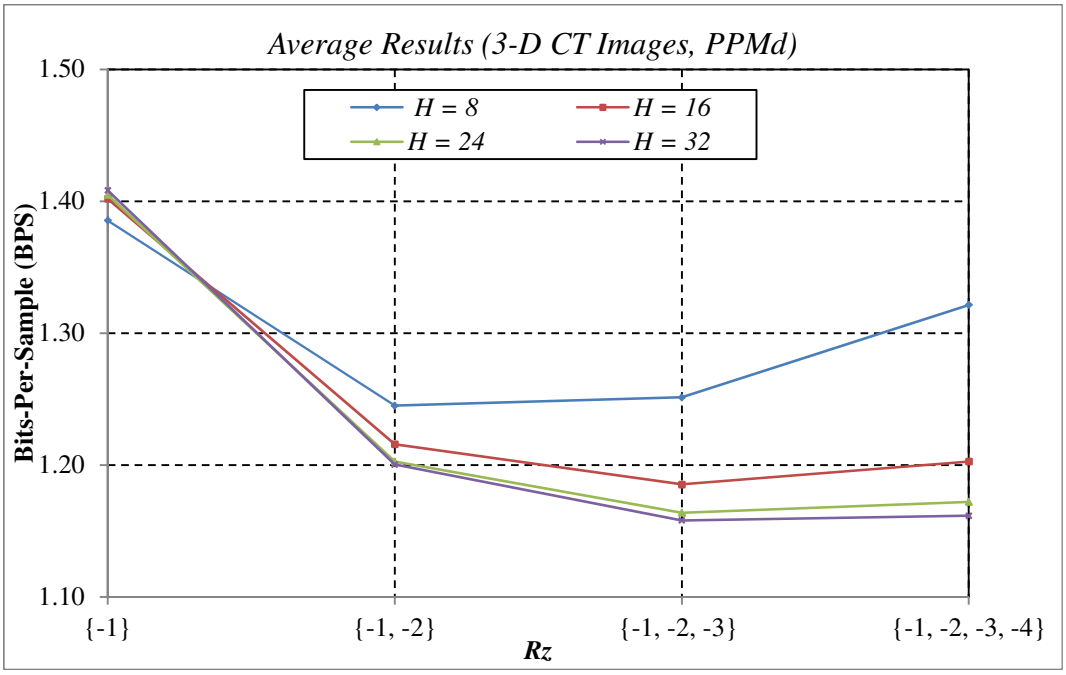


Figure 5.4: A graphical representation of Table 5.7.

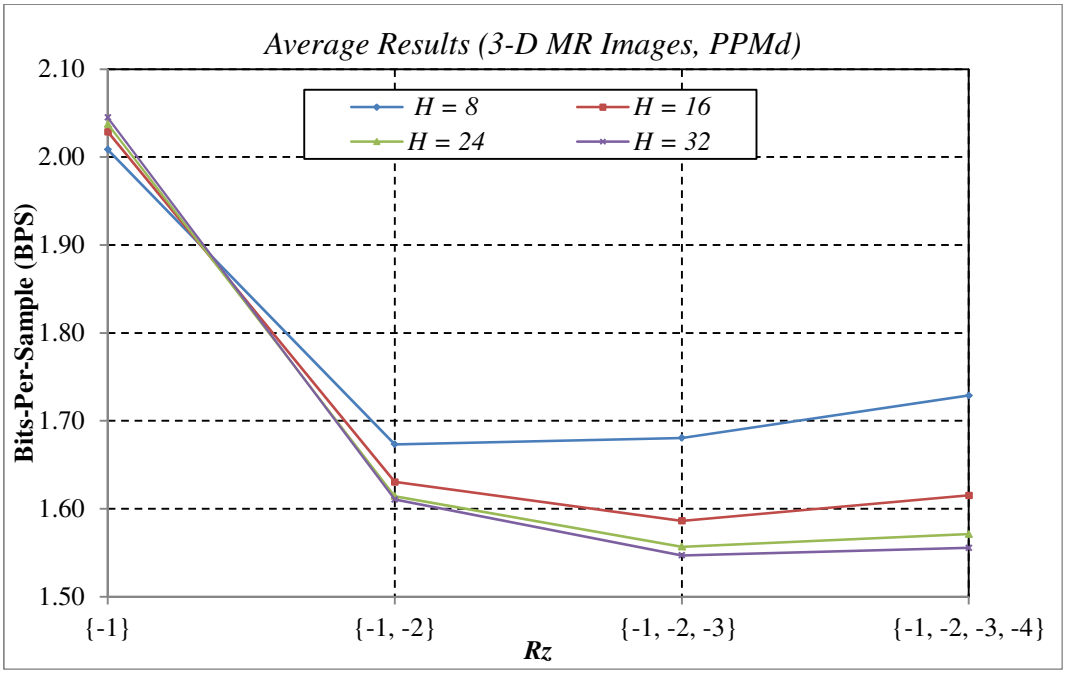


Figure 5.5: A graphical representation of Table 5.8.

Tables 5.9 and 5.10 (similarl to Tables 5.7 and 5.8) emphasize the average results achieved by considering the PAQ8 approach for the coding of prediction errors. It should be noted that thanks to the general best compression performances (paid with higher computational costs), the average results are better than the ones obtained by using the PPMd scheme.

Table 5.9: Average results on the CT Images. The coding of prediction errors is performed through the PAQ8 scheme.

H	$R_Z = \{ -1 \}$	$R_Z = \{ -1, -2 \}$	$R_Z = \{ -1, -2, -3 \}$	$R_Z = \{ -1, -2, -3, -4 \}$
8	1.3114	1.1732	1.1767	1.2422
16	1.3260	1.1488	1.1171	1.1332
24	1.3273	1.1373	1.0973	1.1048
32	1.3299	1.1350	1.0915	1.0947

Table 5.10: Average results on the MR Images. The coding of prediction errors is performed through the PAQ8 scheme.

H	$R_Z = \{ -1 \}$	$R_Z = \{ -1, -2 \}$	$R_Z = \{ -1, -2, -3 \}$	$R_Z = \{ -1, -2, -3, -4 \}$
8	1.9045	1.5854	1.5912	1.6780
16	1.9225	1.5446	1.5037	1.5319
24	1.9293	1.5290	1.4749	1.4893
32	1.9355	1.5251	1.4645	1.4735

In Figures 5.6 and 5.7, we graphically show the information of Tables 5.9 and 5.10, respectively. In detail, also in this scenario and in both the cases, it is possible to note that the best average results trend is obtained when the H parameter is equal to 32 (except for the case in which $R_Z = \{-1\}$) and, the best results, are obtained when $R_Z = \{-1, -2, -3\}$.

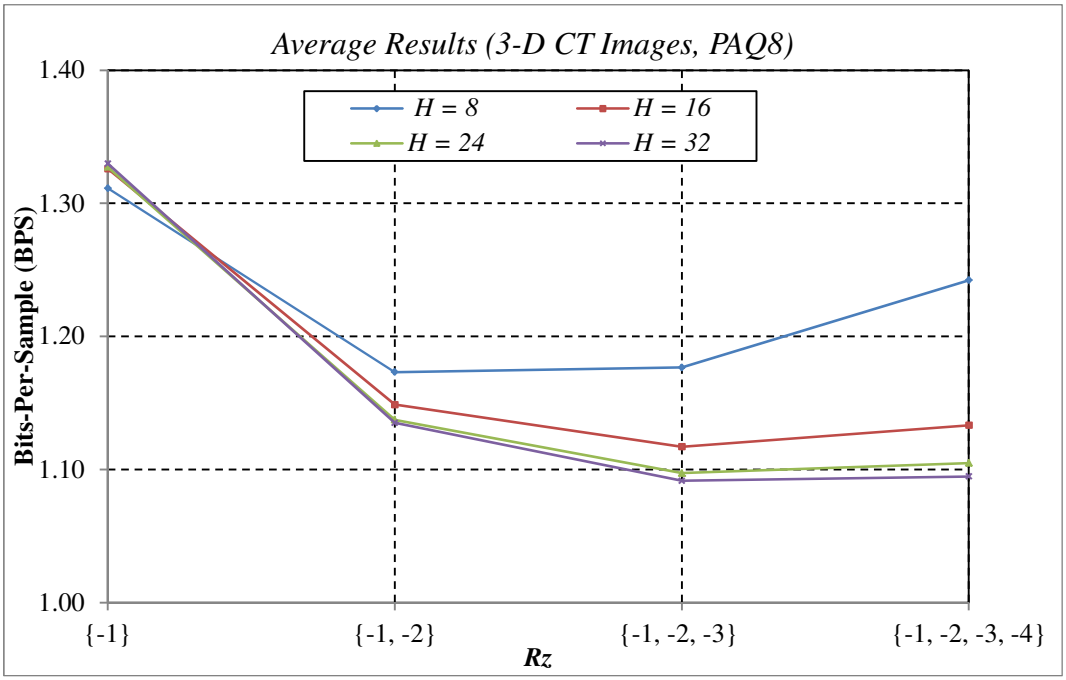


Figure 5.6: A graphical representation of Table 5.9.

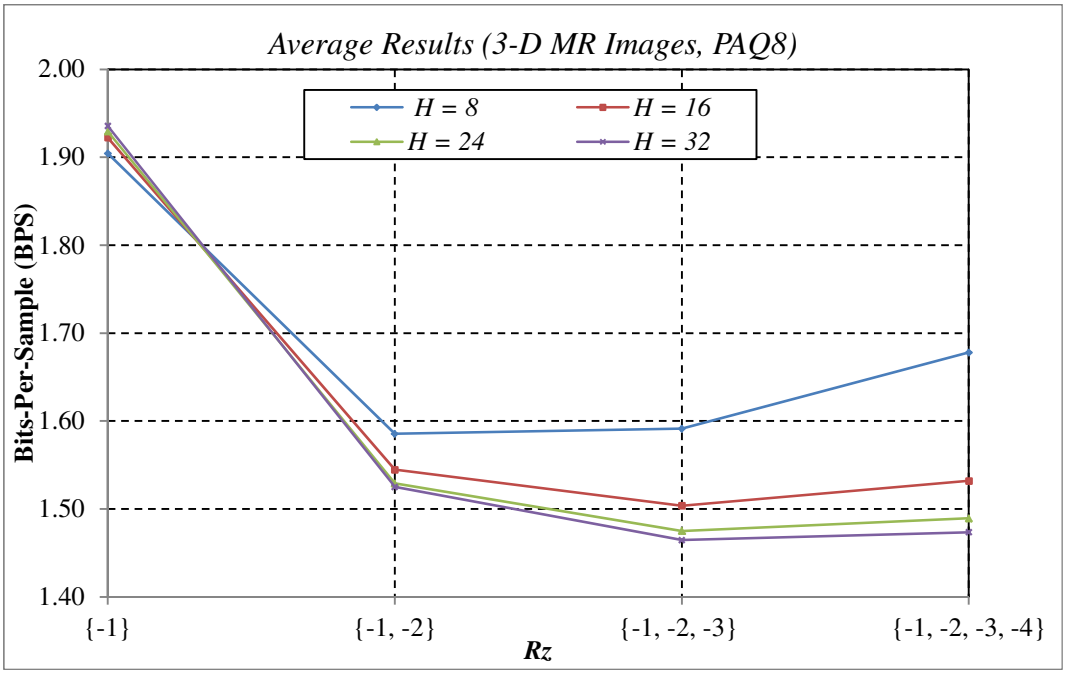


Figure 5.7: A graphical representation of Table 5.10.

Tables 5.11 and 5.12 report the average results concerning the coding of prediction errors through the AC scheme. In the scenario, the average results are worse than the coding of the prediction errors via PPMd and PAQ8.

Table 5.11: Average results on the CT Images. The coding of prediction errors is performed through the AC scheme.

H	$R_Z = \{-1\}$	$R_Z = \{-1, -2\}$	$R_Z = \{-1, -2, -3\}$	$R_Z = \{-1, -2, -3, -4\}$
8	1.9591	1.5323	1.4962	1.5660
16	2.0418	1.5398	1.4484	1.4586
24	2.0939	1.5507	1.4398	1.4380
32	2.1303	1.5632	1.4427	1.4354

Table 5.12: Average results on the CT Images. The coding of prediction errors is performed through the PAQ8 scheme.

H	$R_Z = \{-1\}$	$R_Z = \{-1, -2\}$	$R_Z = \{-1, -2, -3\}$	$R_Z = \{-1, -2, -3, -4\}$
8	2.3978	1.8098	1.7922	1.8838
16	2.4829	1.7904	1.7118	1.7371
24	2.5417	1.7933	1.6925	1.7013
32	2.5854	1.8026	1.6900	1.6921

Figures 5.8 and 5.9, which graphically show the information of Table 5.11 and Table 5.12, highlight that the best average results are obtained with the following configuration: the H parameter equal to 32 and $R_Z = \{-1, -2, -3, -4\}$ (for the CT images) and $R_Z = \{-1, -2, -3\}$ (for the MR images).

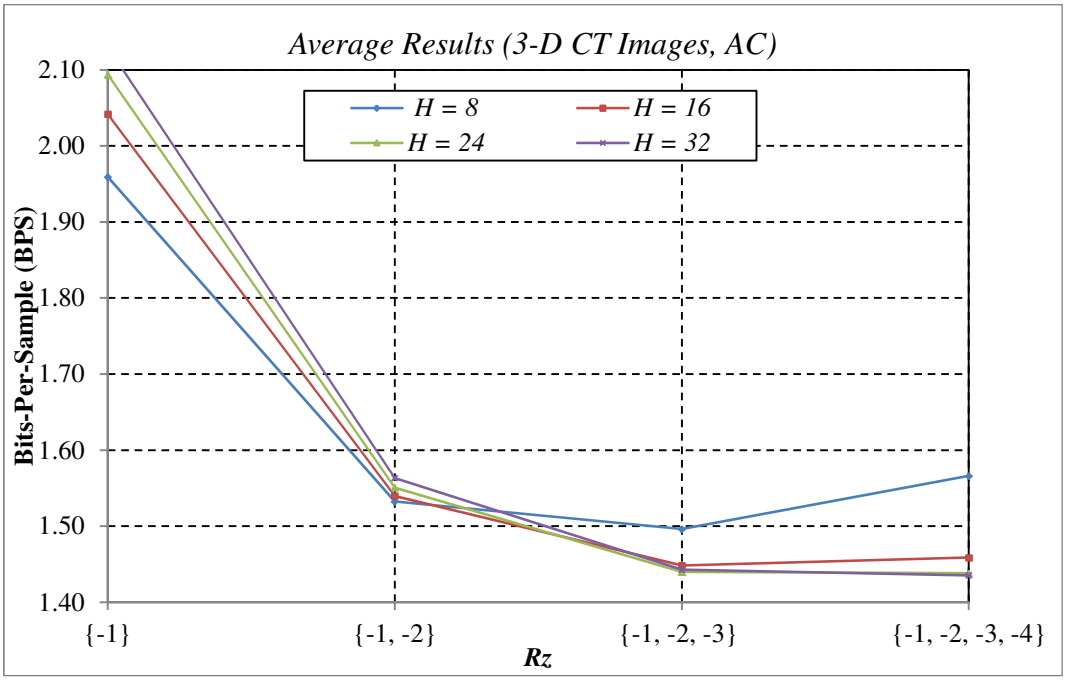


Figure 5.8: A graphical representation of Table 5.11.

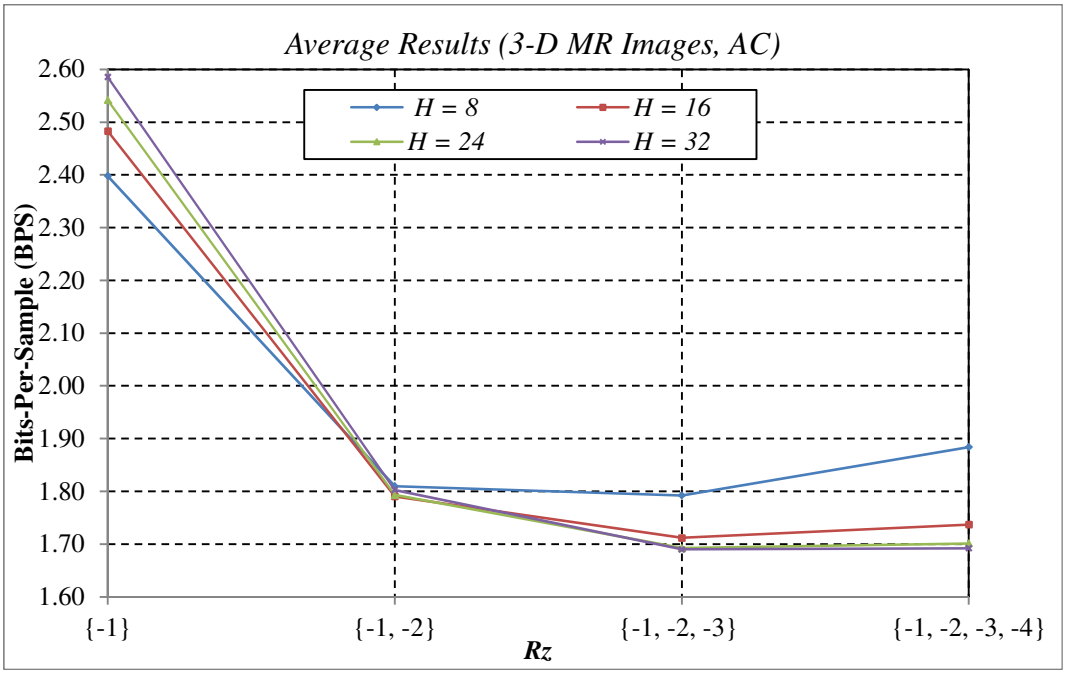


Figure 5.9: A graphical representation of Table 5.12.

In Tables 5.13 and 5.14, we compare the experimental results achieved by our approach with other state-of-art approaches. In detail, Table 5.13 focuses on the CT images, while Table 5.14 focuses on the MR images. In both the tables, from the second to the fifth columns indicate the achieved results on each of the CT images (Table 5.13) as well as each of the MR images (Table 5.14). The results are reported for each compared method (first column). Whereas, the average results of all the tested images of a specified method is reported in the last column (highlighted in italic). It should be noted that the parameters used for the proposed approach are the following: $H = 32$ and $R_Z = \{-1, -2, -3\}$. In particular, we report the results obtained by using the PPMd, PAQ8 and the AC schemes.

Table 5.13: Comparison with other state-of-art approaches (CT images).

Methods	**CT_skull**	**CT_wrist**	**CT_carotid**	**CT_Aperts**	***Average***
Proposed (PPMd)	1.5735	0.9344	1.3455	0.7786	*1.1580*
Proposed (PAQ8)	1.4786	0.8935	1.2739	0.7198	*1.0915*
Proposed (AC)	1.8437	1.1553	1.7389	1.0327	*1.4427*
3D-ESCOT	1.8350	1.0570	1.3470	0.8580	*1.2743*
MILC	2.0306	1.0666	1.3584	0.8190	*1.3187*
AT-SPIHT	1.9180	1.1150	1.4790	0.9090	*1.3553*
3D-CB-EZW	2.0095	1.1393	1.3930	0.8923	*1.3585*
DPCM+PPMd	2.1190	1.0290	1.4710	0.8670	*1.3715*
3D-SPIHT	1.9750	1.1720	1.4340	0.9980	*1.3948*
3D-EZW	2.2251	1.2828	1.5069	1.0024	*1.5043*
JPEG-LS	2.8460	1.6531	1.7388	1.0637	*1.8254*

Table 5.14: Comparison with other state-of-art approaches (MR images).

Methods	**MR_liver_t1**	**MR_liver_t2e1**	**MR_sag_head**	**MR_ped_chest**	***Average***
Proposed (PPMd)	1.9231	1.3366	1.5892	1.3391	*1.5470*
Proposed (PAQ8)	1.8485	1.2504	1.4768	1.2822	*1.4645*
Proposed (AC)	1.9959	1.6297	1.7610	1.3735	*1.6900*
3D-ESCOT	2.0760	1.5100	1.9370	1.6180	*1.7853*
MILC	2.1968	1.7590	2.0975	1.6556	*1.9272*
3D-SPIHT	2.2480	1.6700	2.0710	1.7420	*1.9328*
3D-CB-EZW	2.2076	1.6591	2.2846	1.8705	*2.0055*
DPCM+PPMd	2.3900	2.0250	2.1270	1.6890	*2.0578*
3D-EZW	2.3743	1.8085	2.3883	2.0499	*2.1553*
JPEG-LS	3.1582	2.3692	2.5567	2.9282	*2.7531*

Figures 5.10 and 5.11 graphically show the histograms related to the average results reported in Tables 5.13 and 5.14, respectively.

From the figures and the aforementioned tables, it is possible to note that the obtained results outperform the other approaches in the state-of-art, when

PPMd and PAQ8 are coupled with our predictive model. In the scenarios, singularly and on average, our approach performs better with respect to the other competitors. Whereas, when the AC scheme is used on the MR images, the results are better than the other state-of-art approaches. On the other hand, the results are significantly worse with respect to the compared approaches on the CT images. For example, in the case of the *CT_Aperts* image, our predictor coupled with the AC scheme achieves 1.0327 (in terms of BPS), while 3D-ESCOT achieves 0.8580, which is significantly better.

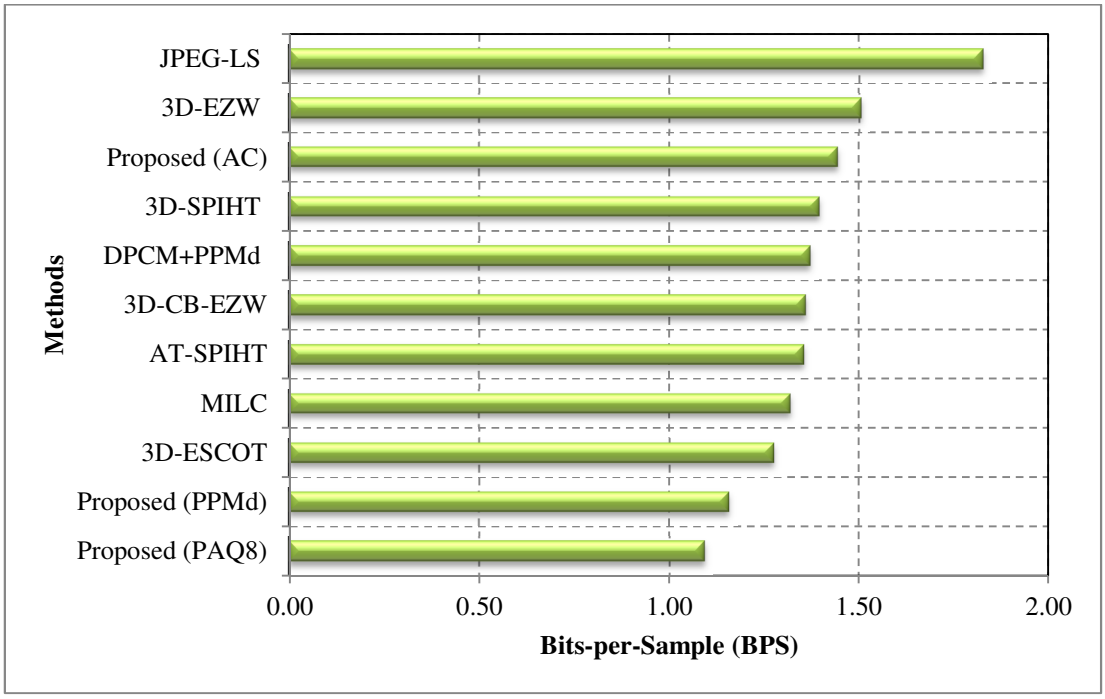


Figure 5.10: Histogram of Table 5.13.

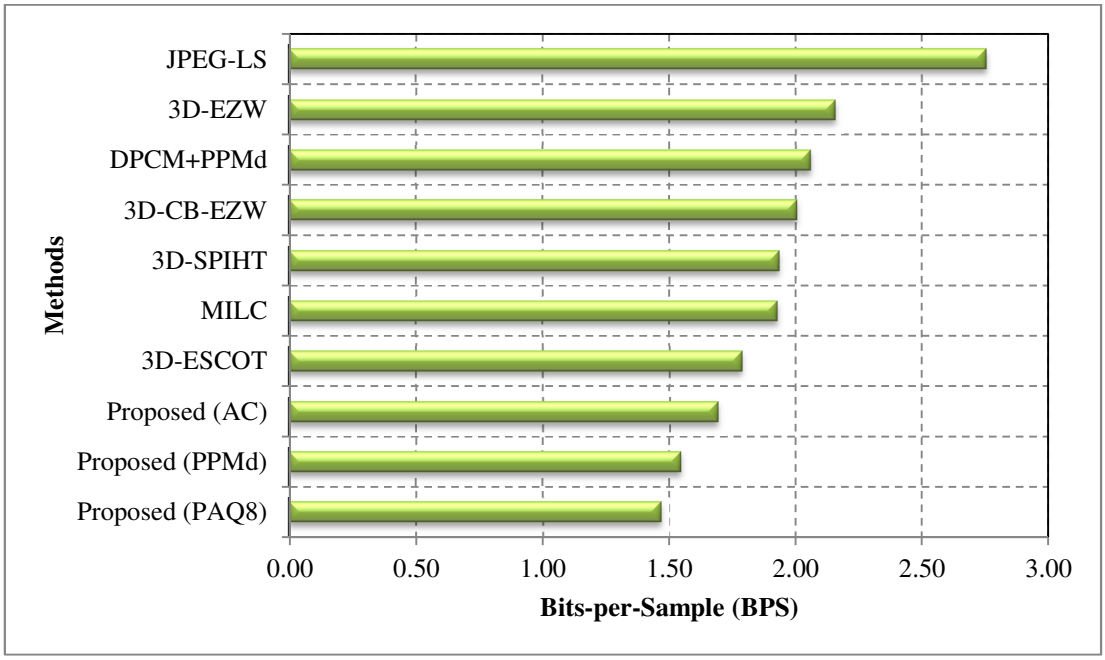


Figure 5.11: Histogram of Table 5.14.

5.6.2. Hyperspectral Images

We performed our experiments on a dataset composed of five *Airborne Visible/Infrared Imaging Spectrometer* (*AVIRIS*) **[23]** hyperspectral images, provided by the NASA's Jet Propulsion Laboratory (JPL) **[4]**. The used dataset is described in Appendix A.3.

In Tables 5.15, 5.16, 5.17, 5.18, 5.19 and 5.20, the achieved experimental results are reported. The results are obtained by using different values for the H parameter (i.e., 8 and 16) and several values for the Set of References (i.e., $R_Z = \{-1\}$, $R_Z = \{-1, -2\}$ and $R_Z = \{-1, -2, -3\}$). In addition, we used the AC scheme for the coding of prediction errors. In detail, in Tables 5.18, 5.19 and 5.20, we report the results achieved by using the H parameter equal

to 8 and $R_Z = \{-1\}$, $R_Z = \{-1, -2\}$ and $R_Z = \{-1, -2, -3\}$, respectively. Whereas, in Tables 5.18, 5.19 and 5.20, the experimental results achieved by using the H parameter equal to 16 are reported.

Table 5.15: Achieved results on the hyperspectral images ($H = 8, R_Z = \{-1\}$).

Scenes	**Lunar Lake**	**Moffett Field**	**Jasper Ridge**	**Cuprite**	**Low Altitude**
Scene 01	5.0560	5.1463	5.0602	4.9699	5.3784
Scene 02	5.0099	5.1023	5.0524	5.0457	5.4015
Scene 03	4.9963	4.9672	5.0987	4.9930	5.3066
Scene 04	*N.P.*	5.1888	5.1160	5.0380	5.3341
Scene 05	*N.P.*	*N.P.*	5.0532	5.0358	5.3810
Scene 06	*N.P.*	*N.P.*	5.0525	*N.P.*	5.3145
Scene 07	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.3110
Scene 08	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.3268
Average	*5.0207*	*5.1012*	*5.0722*	*5.0165*	*5.3442*

Table 5.16: Achieved results on the hyperspectral images ($H = 16, R_Z = \{-1\}$).

Scenes	**Lunar Lake**	**Moffett Field**	**Jasper Ridge**	**Cuprite**	**Low Altitude**
Scene 01	5.0212	5.1407	5.0547	4.9334	5.3658
Scene 02	4.9767	5.1083	5.0485	5.0194	5.3906
Scene 03	4.9645	4.9654	5.0952	4.9579	5.2892
Scene 04	*N.P.*	5.1843	5.1158	5.0064	5.3158
Scene 05	*N.P.*	*N.P.*	5.0488	5.0002	5.3701
Scene 06	*N.P.*	*N.P.*	5.0463	*N.P.*	5.2963
Scene 07	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.2936
Scene 08	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.3131
Average	*4.9875*	*5.0997*	*5.0682*	*4.9835*	*5.3293*

Table 5.17: Achieved results on the hyperspectral images ($H = 8, R_Z = \{-1, -2\}$).

Scenes	**Lunar Lake**	**Moffett Field**	**Jasper Ridge**	**Cuprite**	**Low Altitude**
Scene 01	5.0141	5.0934	5.0140	4.9364	5.3295
Scene 02	4.9689	5.0172	5.0072	5.0326	5.3491
Scene 03	4.9558	4.8890	5.0501	4.9625	5.2648
Scene 04	*N.P.*	5.1254	5.0654	5.0105	5.2881
Scene 05	*N.P.*	*N.P.*	5.0095	5.0012	5.3281
Scene 06	*N.P.*	*N.P.*	5.0162	*N.P.*	5.2707
Scene 07	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.2732
Scene 08	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.2927
Average	*4.9796*	*5.0313*	*5.0271*	*4.9886*	*5.2995*

Table 5.18: Achieved results on the hyperspectral images ($H = 16, R_Z = \{-1, -2\}$).

Scenes	**Lunar Lake**	**Moffett Field**	**Jasper Ridge**	**Cuprite**	**Low Altitude**
Scene 01	4.9175	5.0206	4.9397	4.8387	5.2482
Scene 02	4.8743	4.9513	4.9341	4.9483	5.2689
Scene 03	4.8632	4.8160	4.9755	4.8679	5.1799
Scene 04	*N.P.*	5.0496	4.9940	4.9194	5.2017
Scene 05	*N.P.*	*N.P.*	4.9358	4.9047	5.2491
Scene 06	*N.P.*	*N.P.*	4.9418	*N.P.*	5.1859
Scene 07	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.1883
Scene 08	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.2108
Average	*4.8850*	*4.9594*	*4.9535*	*4.8958*	*5.2166*

Table 5.19: Achieved results on the hyperspectral images ($H = 8, R_Z = \{-1, -2, -3\}$).

Scenes	**Lunar Lake**	**Moffett Field**	**Jasper Ridge**	**Cuprite**	**Low Altitude**
Scene 01	5.1060	5.1815	5.1037	5.0324	5.4238
Scene 02	5.0621	5.0919	5.0968	5.1363	5.4414
Scene 03	5.0479	4.9649	5.1374	5.0618	5.3610
Scene 04	*N.P.*	5.2085	5.1523	5.1084	5.3831
Scene 05	*N.P.*	*N.P.*	5.0999	5.0963	5.4202
Scene 06	*N.P.*	*N.P.*	5.1138	*N.P.*	5.3656
Scene 07	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.3707
Scene 08	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.3918
Average	*5.0720*	*5.1117*	*5.1173*	*5.0870*	*5.3947*

Table 5.20: Achieved results on the hyperspectral images ($H = 16, R_Z = \{-1, -2, -3\}$).

Scenes	**Lunar Lake**	**Moffett Field**	**Jasper Ridge**	**Cuprite**	**Low Altitude**
Scene 01	4.9243	5.0206	4.9413	4.8494	5.2546
Scene 02	4.8822	4.9383	4.9353	4.9667	5.2731
Scene 03	4.8704	4.8052	4.9739	4.8823	5.1887
Scene 04	*N.P.*	5.0445	4.9919	4.9320	5.2093
Scene 05	*N.P.*	*N.P.*	4.9379	4.9149	5.2528
Scene 06	*N.P.*	*N.P.*	4.9510	*N.P.*	5.1935
Scene 07	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.1980
Scene 08	*N.P.*	*N.P.*	*N.P.*	*N.P.*	5.2214
Average	*4.8923*	*4.9522*	*4.9552*	*4.9091*	*5.2239*

5.6.2.1. Results Discussion

In this section, we analyze the experimental results, which are achieved on the used dataset. In particular, in Tables 5.21, the average results of all the tested hyperspectral images are reported.

It is important to point out that all the results are obtained by considering the AC scheme for the coding of prediction errors. It is important to note that the 2D-LMP predictor (described in Section 4.1.1) and the 3D-DLMP predictor (described in Section 4.1.2) are used for the prediction of all the samples of the first band as well as the management of the exceptions, respectively.

In detail, the first column indicates the H parameter and from the second to the fourth columns the average results for each tested Set of References, respectively.

Table 5.21: Average results of the hyperspectral images.

$\boldsymbol{H}$	$\boldsymbol{R_Z} = \{ -\mathbf{1} \}$	$\boldsymbol{R_Z} = \{ -\mathbf{1}, -\mathbf{2} \}$	$\boldsymbol{R_Z} = \{ -\mathbf{1}, -\mathbf{2}, -\mathbf{3} \}$
8	5.1110	5.0652	5.1565
16	5.0936	4.9821	4.9865

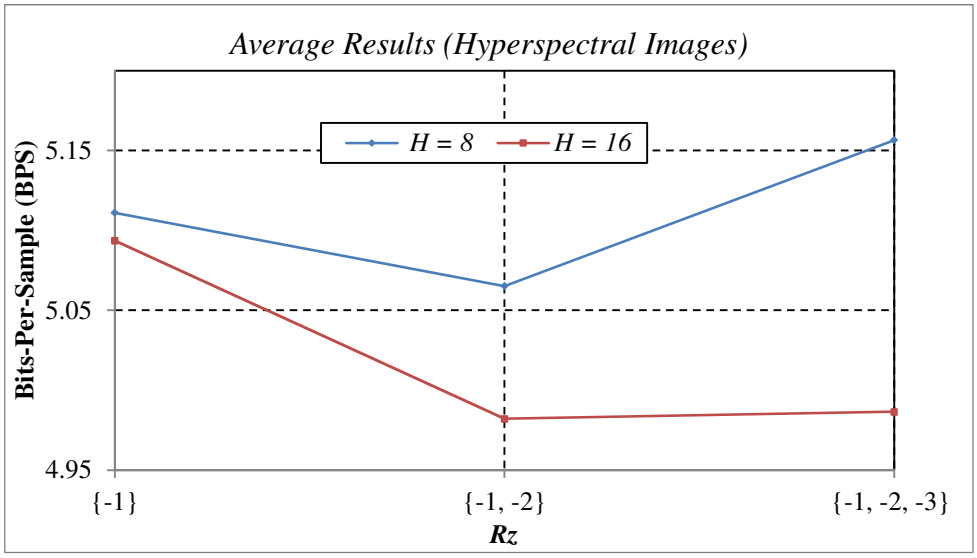


Figure 5.12: Graphical representation of Table 5.21.

From Figure 5.12, it is possible to observe that the best results are achieved when the H parameter is equal to 16 and $R_Z = \{-1, -2\}$. On the other hand, the worst results are obtained in the case in which the H parameter is equal to 8 and $R_Z = \{-1, -2, -3\}$.

We also tested the PAQ8 and the PPMd schemes for the coding of the prediction errors. In detail, by using the PAQ8 scheme, the results are better with respect to those achieved by using the AC scheme. On the other hand, the results obtained by using the PPMd scheme are slightly worse than those obtained by using the AC scheme.

In the case of the *Scene 02* of the *"Cuprite"* image, 4.9693 and 5.1702 are achieved in terms of BPS (with the following parameters, $H = 8$ and $R_Z = \{-1, -2\}$), respectively.

It is important to note that the achieved results are comparable with other state-of-art approaches.

5.6.3. 5-D functional Magnetic Resonance Images (fMRI)

We focus on the experimental testing of our approach on the data produced through the functional Magnetic Resonance Imaging (fMRI) technology. In particular, we perform our experiments on two datasets (referred to as *Dataset 1* and *Dataset 2*, respectively), provided by the *OpenfMRI project* [**19**], which are described in Appendix A.4.

For brevity, in order to denote how many and which dimensions of an fMRI image will be used, by the predictive structure to perform the prediction, we will use a notation similar to the following, 4-D (T, Z). In detail, we can describe this latter configuration as the configuration for the Sets of References, which permits to use the previous slice of each one of four dimensions of a fMRI image, namely, the T dimension and Z dimension (the T and the Z dimensions are highlighted in the parenthesis), the X and the Y dimensions. For the 5-D configuration, there is no need to indicate the used dimensions, in the parenthesis, since all the dimensions of a fMRI image are exploited.

5.6.3.1. Experimental Results on Dataset 1

In this section, we focus on the experimental results achieved by our approach on Dataset 1. In particular, we used the PPMd scheme for the coding of prediction errors and the 2D-LMP is used, for the prediction of all the samples of slices, with no reference slices as well as for the management of the

exceptions. Furthermore, we experimentally tested our method by using different configurations and several values for the H parameter. In detail, Tables 5.22, 5.23, 5.24 and 5.25 report the results obtained by using $H = 8$, $H = 16$, $H = 24$ and $H = 32$, respectively.

The tables are organized as follows: the first column indicates the studied subjects, the columns from the second to the sixth (each one related to a tested configuration of the Sets of References) is subdivided into two sub-columns, concerning the results achieved on the *task001* image as well as the *task002* image. It should be noted that the last row indicates the average results, with all the results being reported in terms of BPS.

Table 5.22: Achieved results on *Dataset 1* ($H = 8$).

Subjects	**3-D (*Z*)**		**3-D (*T*)**		**4-D (*Z*, *T*)**		**4-D (*T*, *R*)**		**5-D**	
	$R_Z = \{ -1 \}$		$R_T = \{ -1 \}$		$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$	
	task001	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*
sub001	6.9997	7.0008	5.6130	5.6096	5.6998	5.6968	5.6225	5.6117	5.7221	5.7100
sub002	7.3306	7.3176	5.6828	5.6796	5.7803	5.7755	5.6739	5.6786	5.7806	5.7814
sub003	7.0545	7.0536	5.6488	5.6057	5.7337	5.6937	5.6490	5.6159	5.7488	5.7170
sub004	7.5926	7.6049	6.0364	6.0532	6.1345	6.1479	6.0381	6.0324	6.1477	6.1427
sub005	7.1849	7.1867	5.7291	5.7098	5.8200	5.8015	5.7578	5.7155	5.8617	5.8195
sub006	7.1406	7.1288	5.7727	5.7776	5.8619	5.8658	5.7868	5.8079	5.8866	5.9097
sub007	7.5192	7.4989	5.7170	5.7308	5.8210	5.8359	5.7366	5.7509	5.8508	5.8648
sub008	7.1334	7.1337	5.5801	5.5773	5.6754	5.6733	5.5801	5.5914	5.6754	5.6981
sub009	7.4563	*N.P.*	5.6954	*N.P.*	5.7991	*N.P.*	5.7305	*N.P.*	5.8448	*N.P.*
sub010	7.2435	7.2387	5.7055	5.6958	5.7979	5.7884	5.7322	5.7353	5.8369	5.8417
sub011	7.0940	7.0854	5.5904	5.5759	5.6823	5.6673	5.6114	5.6340	5.7147	5.7362
sub012	7.3039	7.3253	5.7278	5.7727	5.8281	5.8727	5.7309	5.8094	5.8407	5.9227
sub013	7.1662	7.1592	5.7608	5.7595	5.8453	5.8448	5.7609	5.7749	5.8587	5.8719
sub014	7.2792	7.2987	5.7751	5.7937	5.8649	5.8854	5.7883	5.8322	5.8890	5.9379
sub015	7.0624	7.0582	5.7245	5.7148	5.8102	5.8012	5.7456	5.7469	5.8445	5.8460
Average	7.2374	7.2208	5.7173	5.7183	5.8103	5.8107	5.7296	5.7384	5.8335	5.8428

Table 5.23: Achieved results on *Dataset 1* ($H = 16$).

Subjects	**3-D (*Z*)**		**3-D (*T*)**		**4-D (*Z*, *T*)**		**4-D (*T*, *R*)**		**5-D**	
	$R_Z = \{ -1 \}$		$R_T = \{ -1 \}$		$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$	
	task001	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*
sub001	6.9895	6.9924	5.5733	5.5698	5.5985	5.5950	5.5409	5.5295	5.5891	5.5763
sub002	7.3264	7.3125	5.6435	5.6403	5.6757	5.6715	5.5897	5.6059	5.6409	5.6513
sub003	7.0418	7.0408	5.6133	5.5689	5.6363	5.5949	5.5712	5.5365	5.6187	5.5855
sub004	7.5895	7.5997	5.9977	6.0165	6.0296	6.046	5.9554	5.9483	6.0053	6.0021
sub005	7.1784	7.1809	5.6891	5.6688	5.7159	5.6971	5.6749	5.6314	5.7252	5.6820
sub006	7.1360	7.1217	5.7341	5.7388	5.7562	5.7605	5.7056	5.7258	5.7501	5.7713
sub007	7.5162	7.4948	5.6776	5.6910	5.7142	5.7293	5.6522	5.6665	5.7088	5.7235
sub008	7.1275	7.1242	5.5397	5.5364	5.5705	5.5678	5.5397	5.5068	5.5705	5.5590
sub009	7.4541	*N.P.*	5.6567	*N.P.*	5.6943	*N.P.*	5.6474	*N.P.*	5.7057	*N.P.*
sub010	7.2311	7.2257	5.6646	5.6548	5.6940	5.684	5.6494	5.6513	5.7000	5.7033
sub011	7.0816	7.0742	5.5489	5.5344	5.5764	5.5614	5.5372	5.5478	5.5826	5.5957
sub012	7.2995	7.3227	5.6868	5.7328	5.7225	5.7675	5.6569	5.7256	5.7085	5.7804
sub013	7.1562	7.1476	5.7235	5.7215	5.7467	5.7447	5.6804	5.6932	5.7258	5.7378
sub014	7.2672	7.2886	5.7359	5.7540	5.7628	5.7821	5.7063	5.7497	5.7532	5.8004
sub015	7.0467	7.0403	5.6830	5.6723	5.7060	5.6960	5.6609	5.6614	5.7064	5.7075
Average	7.2294	7.2119	5.6778	5.6786	5.7066	5.7070	5.6512	5.6557	5.6994	5.7054

Table 5.24: Achieved results on *Dataset 1* ($H = 24$).

Subjects	**3-D (*Z*)**		**3-D (*T*)**		**4-D (*Z*, *T*)**		**4-D (*T*, *R*)**		**5-D**	
	$R_Z = \{ -1 \}$		$R_T = \{ -1 \}$		$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$	
	task001	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*
sub001	6.9830	6.9842	5.5554	5.5514	5.5586	5.5545	5.5086	5.4969	5.5281	5.5148
sub002	7.324	7.3088	5.6262	5.6234	5.6359	5.6320	5.5574	5.5775	5.5792	5.5953
sub003	7.0382	7.0364	5.5970	5.5521	5.5983	5.5558	5.5402	5.5049	5.5596	5.5253
sub004	7.5877	7.5981	5.9812	6.0005	5.9889	6.0074	5.9246	5.9179	5.9446	5.9401
sub005	7.1755	7.1778	5.6714	5.6508	5.6762	5.6566	5.6422	5.5986	5.6639	5.6199
sub006	7.1318	7.1186	5.7170	5.7215	5.7173	5.7215	5.6739	5.6936	5.6898	5.7105
sub007	7.5145	7.4923	5.6601	5.6732	5.6733	5.6880	5.6198	5.6340	5.6464	5.6607
sub008	7.1270	7.1219	5.5221	5.5184	5.5295	5.5264	5.5221	5.4737	5.5295	5.4961
sub009	7.4509	*N.P.*	5.6394	*N.P.*	5.6542	*N.P.*	5.6153	*N.P.*	5.6440	*N.P.*
sub010	7.2238	7.2179	5.6477	5.6379	5.6550	5.6451	5.6184	5.6196	5.6401	5.6425
sub011	7.0753	7.0685	5.5323	5.5176	5.5369	5.5217	5.5093	5.5154	5.5269	5.5340
sub012	7.2985	7.3213	5.6706	5.7167	5.6834	5.7289	5.6296	5.6945	5.6536	5.7197
sub013	7.1502	7.1406	5.7061	5.7034	5.7072	5.7045	5.6492	5.6614	5.6650	5.6766
sub014	7.2632	7.2837	5.7200	5.7382	5.7249	5.7439	5.6757	5.7188	5.6939	5.7405
sub015	7.0378	7.0317	5.6655	5.6546	5.6664	5.6561	5.6287	5.6288	5.6454	5.6462
Average	7.2254	7.2073	5.6608	5.6614	5.6671	5.6673	5.6210	5.6240	5.6407	5.6444

Table 5.25: Achieved results on *Dataset 1* ($H = 32$).

Subjects	**3-D (*Z*)**		**3-D (*T*)**		**4-D (*Z*, *T*)**		**4-D (*T*, *R*)**		**5-D**	
	$R_Z = \{ -1 \}$		$R_T = \{ -1 \}$		$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$	
	task001	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*
sub001	6.9843	6.9838	5.5487	5.5449	5.5446	5.5404	5.4970	5.4850	5.5080	5.4942
sub002	7.3265	7.3114	5.6196	5.6168	5.6219	5.6180	5.5452	5.5668	5.5585	5.5763
sub003	7.0380	7.0367	5.5911	5.5460	5.5848	5.5422	5.5290	5.4936	5.5397	5.5052
sub004	7.5893	7.6013	5.9750	5.9945	5.9745	5.9936	5.9129	5.9065	5.9240	5.9200
sub005	7.1770	7.1791	5.6647	5.6438	5.6620	5.6419	5.6303	5.5862	5.6430	5.5986
sub006	7.1327	7.1198	5.7107	5.7152	5.7034	5.7075	5.6623	5.6819	5.6695	5.6902
sub007	7.5165	7.4937	5.6536	5.6667	5.6584	5.6733	5.6079	5.6221	5.6251	5.6397
sub008	7.1303	7.1251	5.5157	5.5118	5.5151	5.5118	5.5157	5.4617	5.5151	5.4749
sub009	7.4549	*N.P.*	5.6331	*N.P.*	5.6400	*N.P.*	5.6038	*N.P.*	5.6234	*N.P.*
sub010	7.2238	7.2183	5.6410	5.6313	5.6407	5.6309	5.6068	5.6077	5.6197	5.6220
sub011	7.0782	7.0694	5.5259	5.5108	5.5224	5.5071	5.4987	5.5033	5.5075	5.5128
sub012	7.2994	7.3238	5.6638	5.7103	5.6687	5.7145	5.6187	5.6828	5.6340	5.6990
sub013	7.1509	7.1422	5.6998	5.6970	5.6935	5.6906	5.6380	5.6500	5.6455	5.6568
sub014	7.2643	7.2823	5.7137	5.7318	5.7112	5.7301	5.6642	5.7074	5.6738	5.7204
sub015	7.0373	7.0314	5.6587	5.6476	5.6518	5.6415	5.6167	5.6165	5.6245	5.6251
Average	7.2269	7.2085	5.6543	5.6549	5.6529	5.6531	5.6098	5.6123	5.6208	5.6239

5.6.3.2. Experimental Results on Dataset 2

We outline the experimental results concerning Dataset 2. In particular, Tables 5.26, 5.27, 5.28 and 5.29 report the results obtained by using $H = 8$, $H = 16$, $H = 24$ and $H = 32$, respectively. The reported results are obtained by using the PPMd scheme for the coding of prediction errors. The tables are organized as follows: the studied subjects are reported in the first column and from the second to the sixth columns, the achieved results are reported, respectively for each of the tested configurations. It is important to note that the last row indicates the average results, with all the results being reported in terms of BPS.

Table 5.26: Achieved results on *Dataset 2* ($H = 8$).

Subjects	**3-D (*Z*)**	**3-D (*T*)**	**4-D (*Z*, *T*)**	**4-D (*T*, *R*)**	**5-D**
	$R_Z = \{ -1 \}$	$R_T = \{ -1 \}$	$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$
sub001	7.0549	5.7922	5.8692	5.8353	5.9310
sub002	7.0856	5.6474	5.7265	5.6794	5.7762
sub003	7.4676	5.7845	5.8843	5.8073	5.9196
sub004	7.4145	5.7471	5.8440	5.7633	5.8750
sub005	7.2733	5.6911	5.7887	5.7257	5.8387
sub006	6.7643	5.3771	5.4652	5.3942	5.4958
sub007	7.0518	5.5917	5.6866	5.6046	5.7124
sub008	7.1476	5.7128	5.8086	5.7234	5.8311
sub009	6.9215	5.5518	5.6373	5.5699	5.6705
sub010	7.3652	5.7123	5.8134	5.7168	5.8300
sub011	7.2508	5.6701	5.7653	5.7115	5.8197
sub012	7.0680	5.6733	5.7614	5.6575	5.7580
sub013	7.5672	5.9481	6.0552	5.9556	6.0758
sub014	7.0996	5.7208	5.8069	5.7299	5.8307
Average	7.1809	5.6872	5.7795	5.7053	5.8118

Table 5.27: Achieved results on *Dataset 2* ($H = 16$).

Subjects	**3-D (*Z*)**	**3-D (*T*)**	**4-D (*Z*, *T*)**	**4-D (*T*, *R*)**	**5-D**
	$R_Z = \{ -1 \}$	$R_T = \{ -1 \}$	$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$
sub001	7.0516	5.7572	5.7733	5.7470	5.7947
sub002	7.0821	5.6136	5.6337	5.5941	5.6437
sub003	7.4694	5.7476	5.7815	5.7176	5.7750
sub004	7.4131	5.7105	5.7421	5.6716	5.7301
sub005	7.2722	5.6545	5.6868	5.6376	5.6952
sub006	6.7544	5.3380	5.3642	5.3047	5.3567
sub007	7.0456	5.5538	5.5845	5.5156	5.5710
sub008	7.1395	5.6751	5.7029	5.6330	5.6871
sub009	6.9110	5.5133	5.5368	5.4786	5.5300
sub010	7.3647	5.6765	5.7113	5.6262	5.6854
sub011	7.2444	5.6330	5.6649	5.6201	5.6769
sub012	7.0564	5.6360	5.6597	5.5670	5.6160
sub013	7.5621	5.9086	5.9427	5.8603	5.9201
sub014	7.0872	5.6808	5.7040	5.6362	5.6871
Average	7.1753	5.6499	5.6777	5.6150	5.6692

Table 5.28: Achieved results on *Dataset 2* ($H = 24$).

Subjects	**3-D (*Z*)**	**3-D (*T*)**	**4-D (*Z*, *T*)**	**4-D (*T*, *R*)**	**5-D**
	$R_Z = \{ -1 \}$	$R_T = \{ -1 \}$	$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$
sub001	7.0476	5.7416	5.7357	5.7131	5.7304
sub002	7.0805	5.5986	5.5983	5.5607	5.5815
sub003	7.4699	5.7321	5.7432	5.6840	5.7106
sub004	7.4111	5.6950	5.7044	5.6374	5.6653
sub005	7.2692	5.6380	5.6483	5.6038	5.6313
sub006	6.7500	5.3206	5.3253	5.2699	5.2924
sub007	7.0417	5.5374	5.5459	5.4822	5.5071
sub008	7.1352	5.6581	5.6639	5.5982	5.6220
sub009	6.9074	5.4963	5.4979	5.4436	5.4645
sub010	7.3659	5.6605	5.6727	5.5917	5.6202
sub011	7.2385	5.6161	5.6261	5.5849	5.6112
sub012	7.0504	5.6191	5.6201	5.5317	5.5503
sub013	7.5563	5.8909	5.9009	5.8246	5.8510
sub014	7.0785	5.6629	5.6643	5.5999	5.6205
Average	7.1716	5.6334	5.6391	5.5804	5.6042

Table 5.29: Achieved results on *Dataset 2* ($H = 32$).

Subjects	**3-D (*Z*)**	**3-D (*T*)**	**4-D (*Z*, *T*)**	**4-D (*T*, *R*)**	**5-D**
	$R_Z = \{ -1 \}$	$R_T = \{ -1 \}$	$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$
sub001	7.0477	5.7362	5.7233	5.7010	5.7094
sub002	7.0824	5.5935	5.5861	5.5494	5.5615
sub003	7.4737	5.7266	5.7299	5.6721	5.6896
sub004	7.4146	5.6894	5.6913	5.6254	5.6443
sub005	7.2689	5.6322	5.6352	5.5919	5.6105
sub006	6.7531	5.3145	5.3119	5.2573	5.2712
sub007	7.0429	5.5313	5.5321	5.4700	5.4858
sub008	7.1366	5.6520	5.6505	5.5857	5.6009
sub009	6.9099	5.4903	5.4845	5.4311	5.4431
sub010	7.3692	5.6549	5.6593	5.5796	5.5987
sub011	7.2383	5.6100	5.6122	5.5723	5.5895
sub012	7.0516	5.6128	5.6061	5.5186	5.5282
sub013	7.5576	5.8844	5.8862	5.8113	5.8283
sub014	7.0776	5.6562	5.6497	5.5862	5.5976
Average	7.1732	5.6275	5.6256	5.5680	5.5828

5.6.3.3. Results Discussion

Table 5.30 synthesizes the average results obtained on Dataset 1. In particular, the values of the H parameter are reported in the first column, while from the second to the sixth columns, the average results are reported, respectively for each of the tested configurations. It is important to point out that the latter columns are subdivided into two sub-columns, relating to *task001* image and *task002* image.

Similarly to Table 5.30, Table 5.31 shows the average results of Dataset 2. In detail, the first column indicates the values of the H parameter, while the second to the sixth columns indicate the average results, respectively for each of the tested configurations.

Table 5.30: Average results of *Dataset 1*.

H	**3-D (Z)**		**3-D (T)**		**4-D (Z, T)**		**4-D (T, R)**		**5-D**	
	$R_Z = \{ -1 \}$		$R_T = \{ -1 \}$		$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$		$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$	
	task001	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*	*task001*	*task002*
8	7.2374	7.2208	5.7173	5.7183	5.8103	5.8107	5.7296	5.7384	5.8335	5.8428
16	7.2294	7.2119	5.6778	5.6786	5.7066	5.7070	5.6512	5.6557	5.6994	5.7054
24	7.2254	7.2073	5.6608	5.6614	5.6671	5.6673	5.6210	5.6240	5.6407	5.6444
32	7.2269	7.2085	5.6543	5.6549	5.6529	5.6531	5.6098	5.6123	5.6208	5.6239

Table 5.31: Average results of *Dataset 2*.

H	**3-D (*Z*)**	**3-D (*T*)**	**4-D (*Z*, *T*)**	**4-D (*T*, *R*)**	**5-D**
	$R_Z = \{ -1 \}$	$R_T = \{ -1 \}$	$R_Z = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$	$R_R = \{ -1 \}$, $R_T = \{ -1 \}$, $R_Z = \{ -1 \}$
8	7.1809	5.6872	5.7795	5.7053	5.8118
16	7.1753	5.6499	5.6777	5.6150	5.6692
24	7.1716	5.6334	5.6391	5.5804	5.6042
32	7.1732	5.6275	5.6256	5.5680	5.5828

Figures 5.13 and 5.14 graphically show the information of Table 5.30, for *task001* image and *task002* image of Dataset 1, respectively. Furthermore, Figures 5.15 graphically show the information of Table 5.31.

In particular, the configurations are reported on the X-axis, while on the Y-axis, the average results are reported. The blue, dark red, green and violet lines outline the trend in correspondence to the average results, in the cases in which $H = 8$, $H = 16$, $H = 24$ and $H = 32$, respectively.

From the figures, it is worth noting that the best results are obtained when the H parameter is equal to 32, even if similar results are obtained, but slightly worse (except for the 3-D (Z) configuration) when the H parameter assumes a value equal to 24.

By focusing on the analysis of the average results, it is possible to observe that when only the Z dimension is explored (i.e. the 3-D (Z) configuration), the worst results are obtained.

Regarding the case in which the Z dimension as the third dimension and the T dimension as the fourth dimension are used (i.e. the 4-D (T, Z) configuration), the results are slightly better, with respect to the 3-D (T) configuration, when only the H parameter is equal to 32. For the other values

of the H parameter (i.e. 8, 16 and 24), the results are worse than the 3-D (T) configuration.

On the other hand, the results achieved by the 4-D (T, R) configuration are better than the other configurations. In detail, through the 5-D configuration similar, but slightly worse, results, with respect to 4-D (T, R) configuration, are obtained, especially when the H parameter is set to 32.

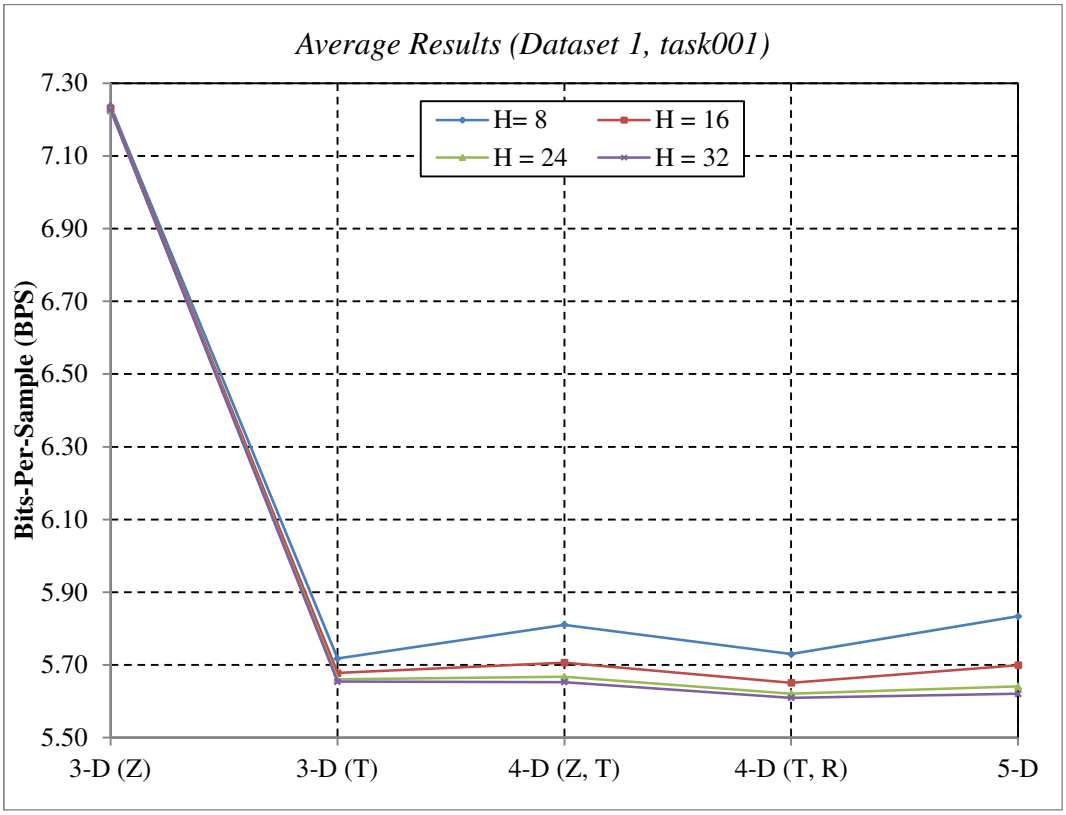


Figure 5.13: Graphical representation of Table 5.30.

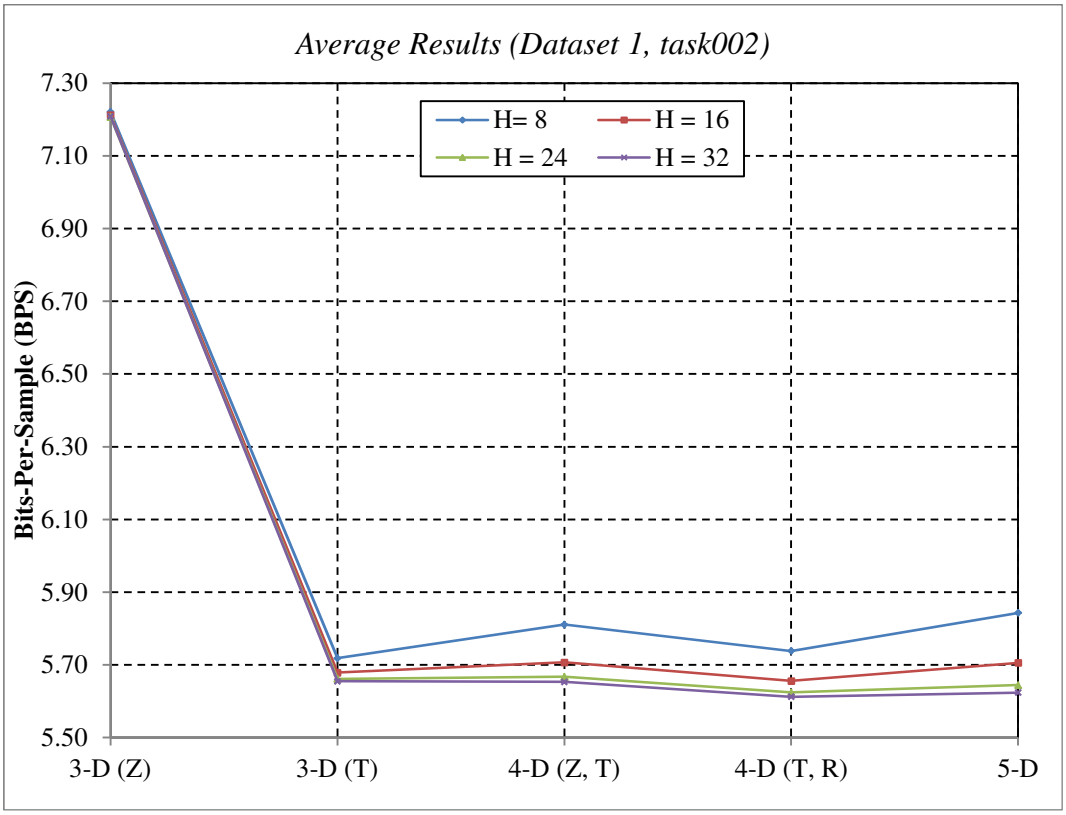


Figure 5.14: Graphical representation of Table 5.30.

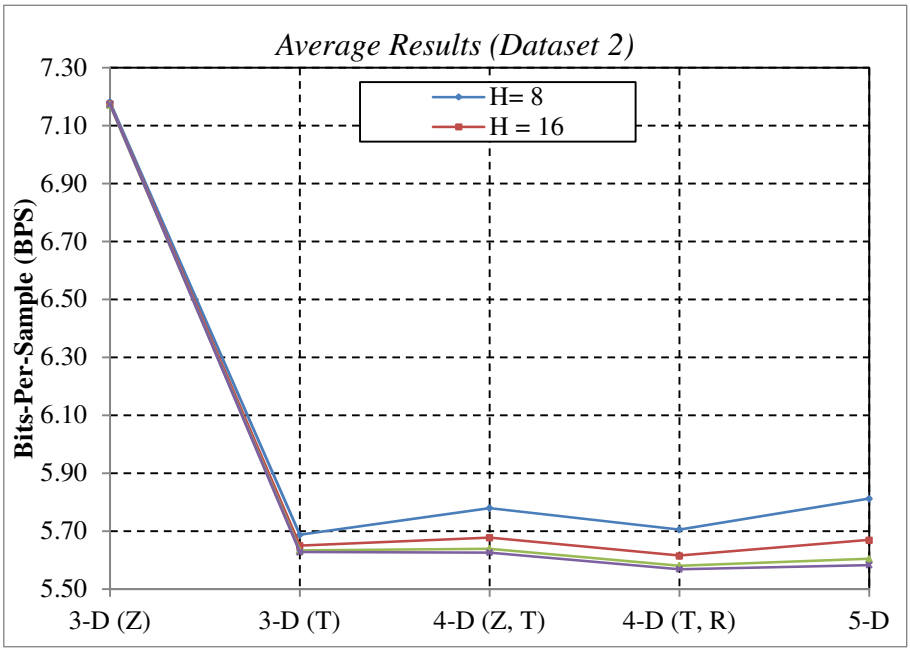


Figure 5.15: Graphical representation of Table 5.31.

Conclusions and Future Works

In this thesis, we reviewed several novel strategies for the efficient compression and protection of different types of multidimensional data. It is important to emphasize that all the ideas of this work have been developed and published on different peer-reviewed international journals as well as the proceedings of various peer-reviewed international conferences, during the PhD programme. The references of all the paper are reported in the Reference section.

First of all, we synthesized the basic aspects and main highlights of the different multidimensional data: 3-D medical images, hyperspectral images, 3-D microscopy images and 5-D fMRI images.

Subsequently, we focused on the lossless compression of medical images, by considering the MILC algorithm, which is designed to obtain a good trade-off between computational complexity and compression performances. In addition, we reviewed a parallelized version of the compression strategy of the MILC algorithm, denoted as Parallel MILC. The main aim of the Parallel MILC algorithm is the improvement of the performance, in terms of execution time. It is important to note that only the compression strategy is modelled in order to exploit the capabilities of the parallel computing. Therefore, it is possible to perform the decompression independently without knowing which compression strategy is used between MILC or Parallel MILC.

Then, we reviewed a novel hybrid approach for the protection, through the embedding of a digital invisible watermark, and the simultaneous compression of 3-D medical images. In many scenarios, the hiding of some information in a 3-D medical image, at the same time as their compression, could be significant.

In addition, we provided a description of the key points of an approach related to the protection of the 3-D microscopy images. This approach is suitable to embed two invisible watermarks: the former is used for the protection of each slice of a 3-D microscopy image, while the latter is used for the protection of the whole 3-D microscopy image. To the best of our knowledge, there are no proposed approaches for the protection of 3-D microscopy images.

Finally, we described a predictive structure, which we used for the efficient and lossless compression of multidimensional data. We experimentally tested our approach, by considering different types of multidimensional data: 3-D medical images, hyperspectral images and 5-D functional Magnetic Resonance Images (fMRI).

There are several future research directions, which might include the following ideas:

- The design of a parallelized version of the decompression strategy of the Parallel MILC algorithm;
- The extension, for the 5-D fMRI data, of the hybrid approach for the embedding of a digital watermark and the simultaneous compression of 3-D medical images;
- The parallelization of the strategies related to the compression and the decompression of multidimensional data;
- The design of approaches for the efficient compression of 3-D microscopy images;
- The design of lossy compression schemes for the compression of multidimensional data.

Appendix – A

Description of the Datasets

A.1. 3-D Medical Images

The dataset used, related to the 3-D Medical Images, is publicly available and composed of eight images. It is important to point out that the images are produced by two different technologies. In particular, the dataset is composed of four 3-D Computed Tomography (3-D CT) images and four 3-D Magnetic Resonance (3-D MR) images.

Tables A.1 and A.2 synthetize the 3-D CT and 3-D MR images of the dataset, respectively.

It is important to emphasize that each slice has 256 columns and 256 lines. In detail, each sample is stored through an 8-bits unsigned integer.

Table A.1: Description of the used dataset (3-D CT images)

3-D Computed Tomography Images		
Description / Age / Gender	**Image Name**	**# of Slices**
Tripod fracture / 16 / M	CT_skull	192
Healing scaphoid dissection / 20 / M	CT_wrist	176
Internal carotid dissection / 41 / F	CT_carotid	64
Apert's syndrome / 2 / M	CT_Aperts	96

Table A.2: Description of the used dataset (3-D MR images)

3-D Magnetic Resonance Images		
Description / Age / Gender	**Image Name**	**# of Slices**
Normal / 38 / F	MR_liver_t1	48
Normal / 38 / F	MR_liver_t2e1	48
Left exophthalmos / 42 / M	MR_sag_head	48
Congenital heart disease / 1 / M	MR_ped_chest	64

A.2. 3-D Microscopy Images

The dataset is composed of five 3-D Confocal Images of cells, which was kindly provided by Dr. Pasquale Del Gaudio (*Department of Pharmacy, University of Salerno, Italy*) and Dr. Veronica Granata (*MUSA Laboratory, SPIN Institute of CNR, Physics Department of the University of Salerno, Italy*).

In detail, Table A.3 reports the description of the structure of the dataset. In the first column the mnemonic name, we denote each 3-D microscopy image with, is reported. The second and third columns indicate the width and height, respectively. Whereas, in the fourth column, the number of slices for each image is reported.

It should be noted that each slice is stored by using 8 bits per RGB channel (24 bits per pixel) and has been stored as the Bitmap Format.

Table A.3: Description of the dataset structure.

Images	**Width**	**Height**	**# of Slices**
Image1	1025	1025	10
Image2	1025	1025	7
Image3	1025	1025	5
Image4	1025	1025	3
Image5	1025	1025	2

A.3. Hyperspectral Images

The dataset used is publicly available and is composed of five *Airborne Visible/Infrared Imaging Spectrometer* (*AVIRIS*) hyperspectral images [**23**]. The data are provided by the NASA's Jet Propulsion Laboratory (JPL) [**4**]. In particular, an hyperspectral image is subdivided into many scenes. Each scene presents 614 columns, 512 lines and 224 spectral bands. It should be noted that the last scene of an image can be composed of less than 512 lines. In addition, each sample is stored as a 16 bits integer.

In Table A.4, we indicate, for each image (in the first column), the number of scenes (in the second column).

Table A.4: Description of the dataset structure.

Images	**# of Scenes**
Lunar Lake	3
Moffett Field	4
Jasper Ridge	6
Cuprite	5
Low Altitude	8

A.4. 5-D fMRI Images

The datasets of fMRI images used are provided by the *OpenfMRI project* site [**19**]. In detail, the used datasets are denoted as "*Stop-signal task with unconditional and conditional stopping*" and "*Living-nonliving decision with plain or mirror-reversed text*" (from now on, we denoted as *Dataset 1* and *Dataset 2*, respectively).

Dataset 1 is composed of thirty 5-D fMRI images and structured as follows: two 5-D fMRI images namely, *task001* and *task002*, are related to each subject. While, Dataset 2 is composed of fourteen 5-D fMRI, each one produced by the analysis of each subject. It is important to note that, in both Dataset 1 and the Dataset 2, each sample is stored by using 16 bits.

Tables A.5.a and A.5.b synthesize only the relevant information, for our experimental testing, related to Dataset 1 and the Dataset 2, respectively. In particular, Table A.3.a reports the sizes of the *task001* image (second column) and the sizes of the *task002* image (third column), of each subject (first column). Table A.3.b shows the sizes of each fMRI image (second column) related to each subject (first column).

Table A.5: Description of the structure of (a) Dataset 1 and (b) Dataset 2.

(a)

Subjects	*task001* $< R,T,Z,X,Y >$	*task002* $< R,T,Z,X,Y >$
sub001	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub002	$< 3,182,30,64,64 >$	$< 2,176,30,64,64 >$
sub003	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub004	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub005	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub006	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub007	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub008	$< 1,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub009	$< 3,182,30,64,64 >$	*Not Present*
sub010	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub011	$< 2,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub012	$< 2,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub013	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub014	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$
sub015	$< 3,182,30,64,64 >$	$< 3,176,30,64,64 >$

(b)

Subjects	$< R,T,Z,X,Y >$
sub001	$< 6,205,25,64,64 >$
sub002	$< 6,205,25,64,64 >$
sub003	$< 5,205,25,64,64 >$
sub004	$< 6,205,25,64,64 >$
sub005	$< 5,205,25,64,64 >$
sub006	$< 5,205,25,64,64 >$
sub007	$< 5,205,25,64,64 >$
sub008	$< 6,205,25,64,64 >$
sub009	$< 6,205,25,64,64 >$
sub010	$< 6,205,25,64,64 >$
sub011	$< 6,205,25,64,64 >$
sub012	$< 6,205,25,64,64 >$
sub013	$< 6,205,25,64,64 >$
sub014	$< 6,205,25,64,64 >$

References

[**1**] H.G. Lalgudi, A. Bilgin, M.W. Marcellin, M.S. Nadar, "Compression of Multidimensional Images Using JPEG2000", *Signal Processing Letters, IEEE*, vol.15, no., pp. 393-396, 2008.

[**2**] LANDSAT Home Page, Online Resource, URL: http://landsat.gsfc.nasa.gov/ (Last Access: February, 2014).

[**3**] MODIS Website Home Page, Online Resource, URL: http://modis.gsfc.nasa.gov/ (Last Access: February, 2014).

[**4**] JPL Home Page, Online Resource, URL: http://www.jpl.nasa.gov/ (Last Access: February, 2014).

[**5**] V. Prasad, D. Semwogerere, and E. R. Weeks, "Confocal microscopy of colloids", *Journal of Physics: Condensed Matter*, vol. 19, no. 11, pp.102-113, 2007.

[**6**] J. Pawley and B. R. Masters, "Handbook of biological confocal microscopy", *Optical Engineering*, vol. 35, no. 9, pp. 2765–2766, 1996.

[**7**] S. A. Huettel, A. W. Song, and G. McCarthy, "Functional magnetic resonance imaging", *Sinauer Associates Sunderland*, MA, 2004, vol. 1.

[**8**] R. Pizzolante, A. Castiglione, B. Carpentieri, A. De Santis, A. Castiglione: "Protection of Microscopy Images through Digital Watermarking Techniques", *The 6-th International Conference on Intelligent Networking and Collaborative Systems*, Salerno, Italia, 2014, pp. 65-72.

[**9**] F. Rizzo, B. Carpentieri, G. Motta, J.A. Storer, "Low-complexity lossless compression of hyperspectral imagery via linear prediction", *Signal Processing Letters, IEEE*, vol.12, no.2, pp. 138-141, Feb. 2005.

[**10**] G. Motta, F. Rizzo, J.A. Storer, "Hyperspectral Data Compression", *Springer Science: Berlin, Germany*, 2006.

[**11**] S. Ait-Aoudia, F. Benhamida, M. Yousfi, "Lossless Compression of Volumetric Medical Data", *Lecture Notes in Computer Science*, vol. 4263/2006, pp. 563-571, 2006.

[**12**] A.R. Aron, T.E. Behrens, S. Smith, M.J. Frank, R.A. Poldrack, "Triangulating a Cognitive Control Network Using Diffusion-Weighted Magnetic Resonance Imaging (MRI) and Functional MRI", *The Journal of Neuroscience*, 4 April 2007, 27(14): 3743-3752.

[**13**] A Bilgin, G. Zweig, and M.W. Marcellin, "Three-Dimensional Image Compression with Integer Wavelet", *Applied Optics*, vol. 39, no. 11, pp. 1799-1814, April 2000.

[**14**] B. Carpentieri, M. Weinberger, G. Seroussi, "Lossless Compression of Continuous Tone Images", *Proceeding of IEEE*, vol. 88, no. 11, pp. 1797-1809, November, 2000.

[**15**] S. Cho, D. Kim, W. A. Pearlman, "Lossless Compression of Volumetric Medical Images with Improved Three-Dimensional SPIHT Algorithm", *Journal of Digital Imaging*, vol. 17, no. 1, pp. 57-63, March, 2004.

[**16**] G.H. Golub, C.F. Van Loan, "Matrix Computations, 3rd ed. Baltimore", *MD: The Johns Hopkins Univ. Press*, 1996.

[**17**] B. Knoll, N. de Freitas, "A Machine Learning Perspective on Predictive Coding with PAQ8", *Data Compression Conference (DCC)*, 2012, pp.377-386, 10-12 April 2012.

[**18**] G. Motta, J.A. Storer, B. Carpentieri, "Lossless Image Coding via Adaptive Linear Prediction and Classification", *Proceedings of the IEEE*, vol. 88, no. 11, pp. 1790–1796, November, 2000.

[**19**] OpenfMRI Home Page, Online Resource, URL: https://openfmri.org/ (Last Access: February, 2014).

[**20**] R. Pizzolante, B. Carpentieri, "Lossless, low-complexity, compression of three-dimensional volumetric medical images via linear prediction", *Digital Signal Processing (DSP)*, 2013, pp.1-6, 1-3 July 2013.

[**21**] R. Pizzolante, B. Carpentieri, "Visualization, Band Ordering and Compression of Hyperspectral Images", *Algorithms*, 2012, 5, 76-97.

[**22**] Z. Xiong, X. Wu, S. Cheng, H. Jianping, "Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms", *IEEE Trans. on Medical Imaging*, vol.22, no.3, pp.459,470, 2003.

[**23**] AVIRIS Airborne Visible / Infrared Imaging Spectrometer Home Page, Online Resource, URL: http://aviris.jpl.nasa.gov/ (Last Access: February, 2014).

[**24**] M. J. Weinberger, G. Seroussi, G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS", *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309-1324, August, 2000.

[**25**] X. Wu, N. Memon, "Context-based, adaptive, lossless image codec", *IEEE Trans. Commun.*, vol. 45, no. 4, pp. 437-444, April, 1997.

[**26**] A. Bilgin, G. Zweig, and M.W. Marcellin, "Efficient Lossless Coding of Medical Image Volumes Using Reversible Integer Wavelet Transforms", *DCC '98 Proceedings of the Conference on Data Compression*, IEEE Computer Society Pub., March, 1998.

[**27**] J.M. Sapiro, "Embedded image coding using zerotrees of wavelet coefficients", *IEEE Trans. On Signal Processing*, vol. 41, no. 12, December, 1993.

[**28**] P.G. Howard, "The design and analysis of efficient lossless data compression systems", *Ph.D. dissertation, Dept. Comput. Sci., Brown Univ.*, Providence, RI, June 1993.

[**29**] D. Shkarin, "PPM: one step to practicality", *Data Compression Conference (DCC)*, 2002, pp. 202–211, April, 2002.

[**30**] J.G. Cleary, I.H. Witten, "Data compression using adaptive coding and partial string matching". *IEEE Trans. on Comm.*, vol. 32, no. 4, pp. 396-402, 1984.

[**31**] J. Ziv, A. Lempel, "A universal algorithm for sequential data compression", *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 337-343, May, 1977.

[**32**] J. Ziv, A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding", *IEEE Trans. Inf. Theory*, vol. IT-24, no. 5, pp. 530-536, September, 1978.

[**33**] M. Burrows, D. Wheeler, "A block sorting lossless data compression algorithm", *Technical Report 124, Digital Equipment Corporation*, 1994.

[**34**] L. Ying, W.A. Pearlman, "Four-Dimensional Wavelet Compression of 4-D Medical Images Using Scalable 4-D SBHP", *Proceedings of DCC 2007*, IEEE Computer Society Pub., March, 2007.

[**35**] H.G. Lalgudi, A. Bilgin, M.W. Marcellin, M.S. Nadar, "Compression of fMRI and ultrasound images using 4D SPIHT", *Proc. of ICIP 2005*, vol. 2, pp. II- 746-9, 2005.

[**36**] K. Pearson, "Mathematical contributions to the theory of evolution-III. Regression, heredity and panmixia", *Philos. Trans. R. Soc. Lond.*, 1896., vol. 187, pp. 253-318.

[**37**] R. Pizzolante, B. Carpentieri, and A. Castiglione, "A Secure Low Complexity Approach for Compression and Transmission of 3-D Medical Images," *in Broadband and Wireless Computing, Communication and Applications (BWCCA)*, 2013 Eighth International Conference on. IEEE, 2013, pp. 387–392.

[**38**] B. Carpentieri, M. J. Weinberger, and G. Seroussi, "Lossless compression of continuous-tone images", *Proceedings of the IEEE*, vol. 88, no. 11, pp. 1797–1809, 2000.

[**39**] A. Munshi, B. Gaster, T. G. Mattson, J. Fung, D. Ginsburg, "OpenCL Programming Guide", *1st ed. Addison-Wesley Professional*, 2011.

[**40**] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems", *Computing in science & engineering*, vol. 12, no. 3, p. 66, 2010.

[**41**] R. Pizzolante and B. Carpentieri, "Band clustering for the lossless compression of aviris hyperspectral images", *ACEEE International Journal on Signal & Image Processing*, 5 (1), 1-14., 2014.

[**42**] M.-S. Hsieh, D.-C. Tseng, and Y.-H. Huang, "Hiding digital watermarks using multiresolution wavelet transform", *Industrial Electronics, IEEE Transactions on*, vol. 48, no. 5, pp. 875–882, 2001.

[**43**] H. Inoue, A. Miyazaki, A. Yamamoto, and T. Katsura, "A digital watermark based on the wavelet transform and its robustness on image compression", *in Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, vol. 2. IEEE, 1998, pp. 391–395.

[**44**] B. Carpentieri and R. Pizzolante: "Lossless Compression of Multidimensional Medical Images for Augmented Reality Applications", *The 1-th International Conference on Augmented and Virtual Reality*, Lecce, Italia, 2014.

[**45**] R. Pizzolante and B. Carpentieri, "Copyright protection for images on mobile devices", *in Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2012 Sixth International Conference on. IEEE, 2012, pp. 585–590.

[**46**] G. Voyatzis and I. Pitas, "Chaotic watermarks for embedding in the spatial digital image domain," *in Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, vol. 2. IEEE, 1998, pp. 432–436.

[**47**] R. Pizzolante, A. Castiglione, B. Carpentieri, A. De Santis, "Parallel Low-Complexity Lossless Coding of Three-Dimensional Medical

Images", *The 17-th International Conference on Network-Based Information Systems*, Salerno, Italia, 2014, pp. 91-98.

[**48**] F. A. P. Petitcolas, "Watermarking schemes evaluation," *Signal Processing Magazine, IEEE*, vol. 17, no. 5, pp. 58–64, Sep 2000.

[**49**] F. A. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Attacks on copyright marking systems", *in Information Hiding. Springer*, 1998, pp. 218–238.

[**50**] J. Nin and S. Ricciardi, "Digital Watermarking Techniques and Security Issues in the Information and Communication Society", *in Advanced Information Networking and Applications Workshops (WAINA)*, 2013, pp. 1553–1558.

[**51**] I. J. Cox, J. Kilian, F. Leighton, T. Shamoon, "Secure spread spectrum watermarking for multimedia", *Image Processing, IEEE Transactions on*, vol. 6, no. 12, pp. 1673–1687, Dec 1997.

[**52**] A. Castiglione, R. Pizzolante, A. De Santis, B. Carpentieri, A. Castiglione, F. Palmieri, "Cloud-based adaptive compression and secure management services for 3D healthcare data", *Future Generation Computer Systems*, Vol. 5, No. 1, pp. 120-134, 2015.

[**53**] J. Katz and Y. Lindell, "Introduction to modern cryptography: principles and protocols", *CRC Press*, 2007.

[**54**] G. Voyatzis and I. Pitas, "Chaotic watermarks for embedding in the spatial digital image domain", *in Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, vol. 2, Oct 1998, pp. 432–436 vol.2.

[55] D. Salomon, G. Motta, "Handbook of Data Compression (5. ed.)", *Springer*, 2010, ISBN 978-1-84882-902-2.

[56] W. B. Pennebaker and J. L. Mitchell, "JPEG Still Image Data Compression", *Van Nostrand Reinhold*, 1992.

[57] S. Golomb, "Run-length encodings (Corresp.)", *IEEE Transactions on Information Theory*, vol.12, no.3, pp.399,401, Jul 1966.