Università di Pisa

Corso di Laurea in Informatica

Tesi di Laurea

NEUROLOGICAL MODEL APPLICATIONS IN IMAGE PROCESSING

Relatore:
Prof. *Vincenzo Gervasi*

Candidato:
Mario Viti

Anno Accademico 2014-2015

Contents

Chapter 1

Introduction

Maria M. Del Viva[1], Giovanni Punzi[2], Daniele Benedetti[3] presented a neurological model to describe the general workings of vision [3]. Vision is presented as a two phases process: a low level early vision phase in which images are acquired and an high level cognitive phase in which visual stimuli are assigned meaning as objects. This study shows that the difference between the bandwidth of photoreceptors in early vision and the rate at which neuron spikes in the cognitive phase highlights the existence of a bottleneck, this bottleneck can be understood in terms of information compression on fast streams of data. In applying this lossy compression method to image processing, only meaningful visual patterns are conserved.

1.1 Neurological model and computer science

The presented model of vision links to various areas of computer science. The notion of bandwidth is related to computer's architecture design and hardware units communication and lossy compression is a methodology touching signal processing and information theory. The very principle of the presented model is the meaningful information definition by means of Shannon's entropy.

[1]NEUROFARBA Dipartimento di Neuroscienze, Psicologia, Area del Farmaco e Salute del Bambino Sezione di Psicologia, Universita‘ di Firenze, Firenze, Italy, IMB University of Chicago, Chicago, Illinois, United States of America

[2]Dipartimento di Fisica "E. Fermi" Universita‘ di Pisa, Pisa, Italy, Fermi National Accelerator Laboratory, Batavia, Illinois, United States of America

[3]NEUROFARBA Dipartimento di Neuroscienze, Psicologia, Area del Farmaco e Salute del Bambino Sezione di Psicologia, Universita‘ di Firenze, Firenze, Italy, Dipartimento di Fisica "E. Fermi" Universita‘ di Pisa, Pisa, Italy

1.2 The experiment

The image processing capabilities of the presented model are investigated at INFN[4]. The Team experimenting on FTK[5] at the University of Pisa has been researching on image processing for an new possible applications outside of the main HEP[6] goals project[7]. An interesting field in which technology developed for HEP may be successful is *medical imaging*, particularly cerebral MRI[8]. This project has been carried out during the period of one month, over which, our contribute was to generate software implementation of the presented method as an "offline" simulation of the hardware logic experimenting on MRI images.

1.3 LossLess and Lossy compression

This work also aims to collect some flavours of different approaches to compression. There are mainly two families of methods: lossy and lossless. Lossless compression exploits pattern's redundancy of data layout. These methods are general and can be applied to multimedia in a straightforward fashion. Also Lossy compression's methods exploits redundancy of patterns, applying statistical analysis in order to minimize space and using ad hoc metrics decide whether a pattern has a negligible contribution to the overall information content; if so it is discarded hence the method is *lossy*. Usually ad hoc metrics are bound to media format therefore this methods do not share the property of generality with the lossless family. The statistical knowledge can be acquired on seen/known uncompressed data to compress unseen/unknown data. The presented model describes a recipe for lossy compression but without any constraints on the choice of ad hoc metrics. Lately in HEP[9] it has been introduced the notion of lossy compression to deal and analyze the massive streams of data produced by the LHC[10].

[4]Istituto Nazionale di Fisica Nucleare
[5]FastTracker a project woriking on experiments at the Large Hadron Collider
[6]High Energy Physics
[7]Fast Tracker for Hadron Collider Experiment, FP7-PEOPLE-2012-IAPP An FP7 IAPP project (February 1, 2103 - January 31, 2017): Grant Agreement Number 324318
[8]Magnetic Resonance Images
[9]High Energy Physics
[10]Large Hadron Collider

Chapter 2

Compression Methods

Claude Shannon gave his contribution to the problem of compression by providing an hard limit for signal processing and data compression [7]. This study also found many application in other fields as it provided the general definition of *information*: a shared concept among scientific fields.[1].

2.1 Entropy

The notion of entropy in information theory provides an unambiguous definition for information. Entropy measures information exactly and quantifies the uncertainty in occurring of a specific message over a communication, avoiding fuzzy notions.

2.1.1 Definition

Let X being a d.r.v $X \in \chi$, $X \sim p$. The entropy of the d.r.v. X is denoted by $H(X) = -\sum_{x \in \chi} p(x) \log_b p(x)$.

The unit of measure of H in case of $b = 2$ is *bits*, in such case communication uses a minimal alphabet of just two symbols $\{0, 1\}$. H is used to predict the length in bits of a particular message in a communication given the $p(x)$.

2.2 Loss less compression

Lossless compression produces perfectly identical data as the original, without loosing any bit. As the *encoding* of data decides a fixed number of bits to represent

[1] It has been used in literature experimental studies to measure obsolescence of texts

a symbol composing a message and because symbols in messages may reoccur: a simpler more minimal encoding can be created to save data. The lossless compression methods are also known as *entropy encoding*.

2.2.1 Huffman coding

This method has been proven to be optimal in coding an ensemble of messages consisting of a finite number of symbols. The minimum redundancy code is constructed in such way that it minimizes the number of coding digits per message [5] and that is as close as possible to the minimum bound: entropy.

Procedure To compress a message, first the set of symbols or *Alphabet* have to be produced, this is the set of different symbols used in the message. The simplest procedure to obtain the Huffman tree has a $\Theta(n + logn) = \theta n$ complexity, with n being the length of the message, it is attained by using a priority queue where the node with lowest frequency is given highest priority: steps are:

1. create the queue as list of symbols
2. while there's more than one node in the queue
   2.1. create a new node having as children 2 nodes with the lowest frequency (highest priority) and add it to the queue
   2.2. remove children from the queue (pointers preserved in the node)

This encoding is proven to be optimal as it is the shortest path on the Huffman tree (Fig.2.1) and is the minimum encoding that can be unambiguously interpreted. The compressed data is therefore attained by then replacing original symbols with the ones generated by the encoding. The decompression phase is done by means of the compression tree, pre pending the tree to the compressed data may introduce a significant spatial overhead in some instances. The worst case is the encoding of an dictionary (ASCII) in which all symbols have uniform probability (or frequency) the output returned minimal encoding will be the same as the one provided as input. The main issue with this method is that it is static and to be adaptive meaning every time data grows a new tree has to be computed, this could imply a non negligible overhead in time computation, this is known as *offline* or *static* compression method.

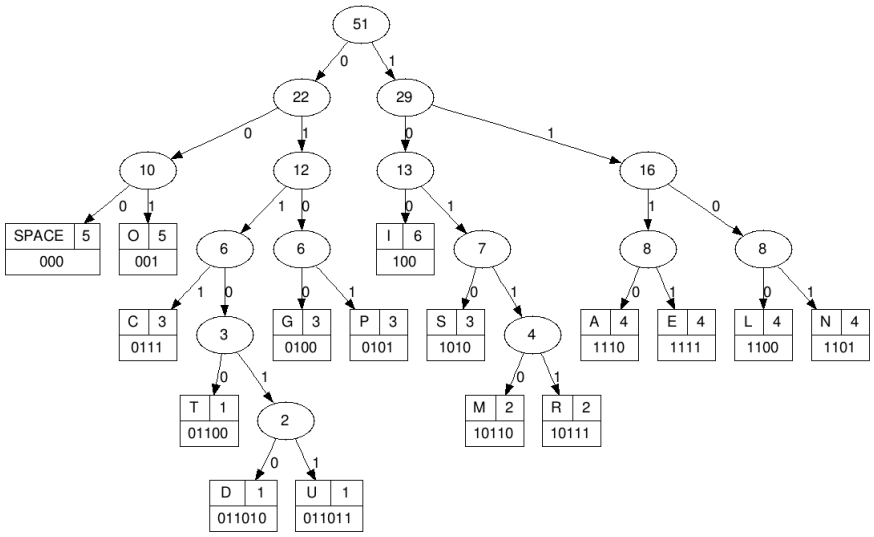

Figure 2.1: Huffman tree encoding of the prhase "Neurological Model Applications in Image Processing"

2.2.2 LZW compression

This method has been presented in 1984 as an automated method to compress data by exploiting the redundancy in files present on *computer systems.*[8]. The study supporting this technology highlighted the fact that only 3/4 of the ASCII character set were mostly used, and therefore the flexibility of the encoding comes at the expense of wasting space on hard disks all over the world, the strategy of the LZW method implied that 1/4 of space used to store documents could be saved. In the paper, it is shown that using Huffman encoding could have some restriction in terms of decompression overhead and lacks generality as the frequency has to be known a priory for a particular type of message (statistical encoders). The most important part of the LZW compression is the substring identification method. The greedy approach of parsing all the sub strings as they occur in a stream of data is the main difference with Huffman trees, as a new substring is encountered new codes can be created incrementally, making LZW more adaptive then Huffman encoding.

LZW algorithm

A string s code is created by encoding recursively the substring $s = \omega K$ with K being the rightmost extension character. Assuming there is an encoding table for a particular set of sub strings, which size depends on bits used to address such table,

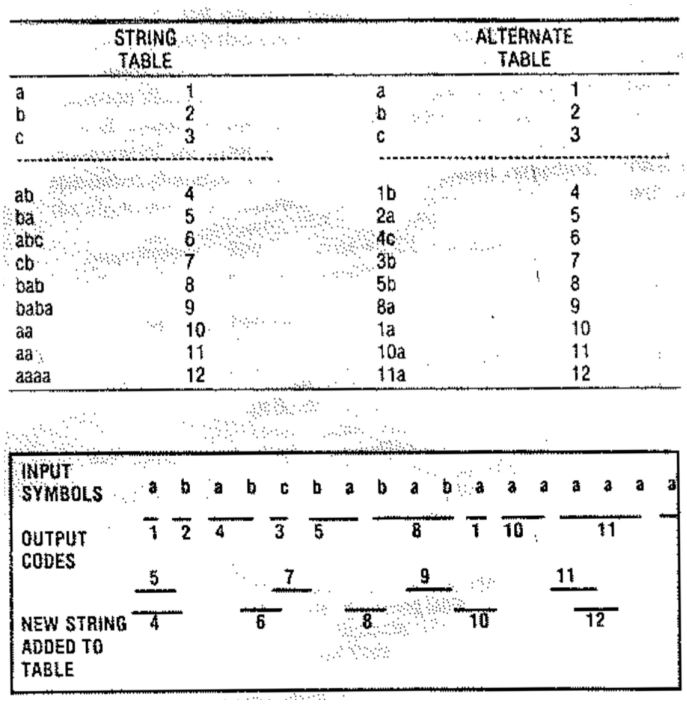

Figure 2.2: LZW compression example shows the STRING TABLE on the left and the logically equivalent ALTERNTIVE TABLE on the right.

the recursive prefix plus extension character can use less codes to represent the same data. In ordinary implementation, the ωK encoding takes up to 12 bits (8 for the char K) and 4 for possible substrings prefixes ω.

Compression Every time the substring is recognized from a stream, the code in the table is replaced with the recursive method, so for example *ab* is recognized only because *a* was already been encoded. The principal concern of this method is storing the string table and to make this tractable sub strings in the table are recorded using the prefix plus extension character encoding, so that each entry has fixed length.

Decompression The decompression phase logically uses the *String table* shown (Fig.2.2)

LZ 77 There are many variants of this method, a particular one does not need the String table to be stored in memory all at once but it is created as data is decompressed, therefore there's an order of decompression, although this might

save space it has a drawback, data has to be stored following an order, therefore if I'd like to decompress only a portion of a file I have to decompress the whole file till the desired point.

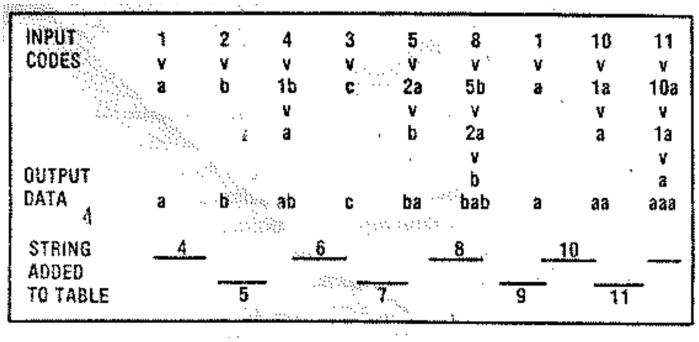

Figure 2.3: LZW decomression example shows the decoding of the outputn data, each code is translated by recursive replacement of the code with a prefix code and extension character

2.3 Lossy compression

Lossy compression inherit most of its theoretical background from signal processing and it is has been particularly successful for multimedia audio and video compression. What is compressed is the amount of data needed to reconstruct a signal and what is preserved is the *energy* of the signal.

2.3.1 DCT discrete cosine transform

DCT compression is based on transforming the space of the image s in the space of an image S where each pixel value represent the intensity contribute of a particular component[2] to the overall image. In it's simplest form a generic transformation has a complexity of ΘN^{43} with N being $N \times N = \#\text{ s} = \#\text{ S}$ on images:

$$S(k_1, k_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} s(n_1, n_2) g(n_1, n_2, k_1, k_2) \tag{2.1}$$

the g function is the Kernel on which depends the component that will be represented in S. Transformation are requested to be invertible by means of the

[2]depending on the transformation, DCT has cosines as basic components

[3]FFT has a computational complexity of $\Theta nlogn$ for the one dimensional case

inverse kernel h

$$s(n_1, n_2) = \sum_{k_1=0}^{N} \sum_{k_2=0}^{N} S(k_1, k_2) h(n_1, n_2, k_1, k_2) \tag{2.2}$$

This step is not lossy, it is the quantization step taken on the S space that is lossy.

$$g(n_1, n_2, k_1, k_2) = g_0(n_1, k_1) g_1(n_2, k_2), g_i(n_i, k_i) = \cos(\frac{\pi(2n_i + 1)k_i}{2N}) \tag{2.3}$$

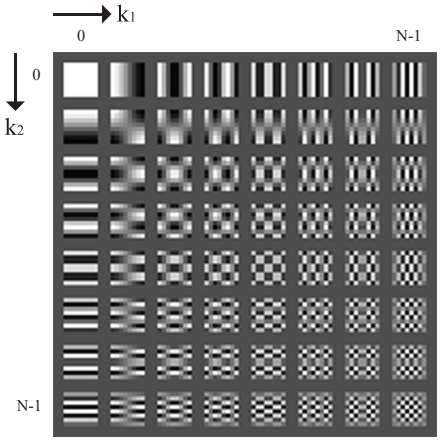

Figure 2.4: frequency space diveided into 8x8 basis images or Kernel Images, this is the most used configuration in jpg images ($N = 8$) Fig.

This S space derived from the kernel 2.3 is attained by means of $\frac{k_i}{N}$ space of discrete cosines 2d waves lowest to the highest, the actual transformation is the response or convolution of the image over the frequency space, from Fig. 2.4 can be also derived the inverse kernel h Fig. 2.2.

The h kernel can be seen as a set of H_{k_1,k_2} matrices each of which will concur to compose a particular value $s(n_1, n_2)$, this matrices are called **basis images** and they correspond to the images in Fig.2.4. Each pixel in the S image will be multiplied for one of the basis images NxN.

Complexity This method of transformation have a ΘN^4 complexity that can be reduced to $\Theta(logNN^3)$ thanks to matrix factorization. In practice the strategy of applying DCT to 8×8 independent patches, reduces "wall clock" time of

computation by exploiting parallel computation.

JPG DCT transform reformulate the image without applying any compression. Compression however takes place in the S space in order to have less space compsumption at the cost of loosing only visual details. Once the image s is transformed into the S image some of the component are cut out because of their low energetic contribution to the overall image. This depends on the image but its statistically probable that natural images have low contribution in high frequencies [6]). DCT is preferred in images in JPG [11] to DFT transformation because the most of the *energy* or emphsignal is concentrated in low frequencies (bulk).This non sparse area is easy to isolate in the left top corner of the S space (origin). JPG used till the latest standard[4] DCT transformation as a first step also in color space: grey images have only compression in the luminosity channel whilst colored images are compressed also in HSV space[5]). Energetic coefficient in S are the cut out using tables of compression depending on the desired quality using a *ladder* approach. In order to be time efficient and to exploit parallelism, this steps are taken separately for a size of the image 8×8. This characteristic introduces artifacts: as the DCT is applied block wise, creating a discontinuity between blocks, this discontinuity is visible in the image and it is known as *blockiness*. (block boundary artifacts, sometimes called macroblocking, quilting, or checkerboarding). The higher the quality of the image the less this discontinuity will be visible.

[4] JPG 2000 uses Wavelets transform

[5] Hue Saturation Value: one of the most used cylindrical model to represent RGB (Red Green Blue) space

Chapter 3

Compression of Big Streams of Data in HEP

3.1 HEP experiments

An HEP[1] detector is similar to a digital camera, in cameras the captured or interesting particles are photons, in case of a detector we have a *multiple layers cameras* mostly interested in capturing quarks or leptons, the product of *protons* clashing[2]. As the theoretical background to understand quark studies is beyond the scope of this work, we'll simply highlight that the particles' s clash creates sub particles that spreads out in many different directions and trajectories. All originate at the collision points IP[3] hence creating a lot of experimental data to be analyzed. From the study of these trajectories, merged with other information, the mass of particles can be inferred so those can be identified. The probability of identify interesting particles is bound to the energy of quarks in the collision which it's not possible to control with current technology. This fact requires experiments to be repeated massively in order to be able to capture interesting yet rare phenomenons estimated from theoretical studies. This very simple description however shows that data produced by an event is massive, but the interesting information is hidden in a small subset, therefore there's a "needle in a haystack" problem to deal with.

[1] high energy physics
[2] This depends on the experiment, LHC have a proton/proton functioning
[3] interaction point (event): the predicted point of collision of quarks

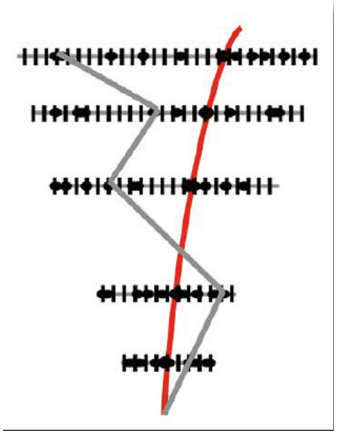

Figure 3.1: A trajectory highlighted by the detector

3.1.1 LHC data analysis and filtering

The most interesting processes generated at LHC[4] are very rare and hidden in an extremely high level of background noise. The LHC generates this processes by accelerating charged *bunches*[5] of particles into counter rotating circular beams, at a rate of 40MHz [6]. In order to observe the products of a collision during a bunch crossing a general-purpose LHC experiment is made of many detecting elements or detector. The data flow is massive (1.7 MB/25 ns, $\sim$ 80 TB/s) and only a very small fraction of the events can be permanently stored as data for further analysis.[1] A *multi-level Trigger* is designed to filter unrelevant events read by detectors, this multilevel-trigger can be described as a stack: the first custom hardware layer L1[7] reduce the event rate from 40 MHz to 100 KHz (1,7MB/10 μs, $\sim$ 170 GB/s), upper layers 2 and 3 both included in the HLT[8] are instead cpu based and refine raw spatial inputs (hits) into more significant and representative objects like the tracks describing the trajectories of particles produced in a collision. The HLT's task of finding trajectories or *track fitting* can be described as a **filtering** process Fig.3.1. This **filter** could greatly benefit from a smaller amount of data provided as input. **Compression** comes into play at this point: cpu based nature is indispensable for flexibility during analysis but it comes at the cost of a limited

[4]Large Hadron Collider

[5]bunch: technical term to indicate packets of protons traveling togheter in a portion of the beam

[6]this rate depends on the capabilities of the LHC (luminosity), one bunch crossing every 25 ns corresponds to the latest upgrade

[7]Level One

[8]High Level Trigger

acceptance bandwidth when confronted with L1 throughput. FTK proposed an upgrade unit between L1 and HLT based on a compression method based on the presented neurological model.

3.1.2 Similarities with the neurological model

An HEP experiment layered architecture share similarities with the neurological phases in the vision process: the levels closer to the event have a very rapid and simple activity, upper levels have a more complex and slower activity of **filtering** similar to the post processing cpu activity in the HLT. The model describes the gap between image acquisition and cognitive processing in vision as an hardware bottleneck and provides a solution based on compression. A key point is that this similarity is strengthened by the use of the metrics of information as it is known in computer science and information theory: Shannon's entropy provides the bridge to implement an algorithm to model this neurological model.

3.2 M. Del Viva Neurological Model

The visual system needs to **filter** the most important elements of the external world from a large flux of information in a short time for survival purposes. It is widely believed that in performing this task, it operates a strong data reduction at an early stage, by creating a compact summary of relevant information that can be handled by further levels of processing [3]. This study shows that the difference between the bandwidth of photoreceptors in early vision and the rate at which neuron spikes in the cognitive process highlights the existence of a bottleneck, this bottleneck can be understood in terms of information compression on fast streams of data.

3.2.1 Model

Under this assumptions the model aims to mimic this neurological behavior by maximizing the information entropy given the resource constrains of a limited output bandwidth: W as the expected value of the frequency of the selected patterns/salient features*features can be patterns but not vice versa, in the context of these works these terms will be used as they were the same*, among these only a limited number can be stored: N as the storage capacity. The first constraint is bound to the context of the application (hardware bottleneck and different IO

bandwidth), the second constraint provides the size of the set of salient features that are to be preserved from the original input and therefore their cost in terms of memory allocation. It is defined the set of all possible configuration of fixed portions of the input (fixed size scrolling window) Q and p_i as the probability of a portion of the input to match an element in Q such that $\sum_{i \in Q} p_i = 1$. The average output information is measured in entropy $\sum_{i \in Q} -p_i \log p_i$ the model therefore maximizes this measure to preserve information: a trivial solution is $N = \#Q$, this would only result in a change of format of the input leaving the size of the input the same as the output. In order to compress the input, constraints must be introduced in the formula. By imposing only constraint N highest entropy patterns would be selected, but as entropy peaks at $1/2$ when information is encoded in bits, and because this is a large probability, that will result in a large output, the second constraint W will ensure that output rate is bounded. This is represented by the bandwidth cost of each pattern p_i/W with W being the accepted bandwidth of the output. The main cost of each pattern is therefore:

$$f(p) = \frac{-p \log p}{\max(1/N, p/W)} \tag{3.1}$$

The optimal configurations for the provided constraints N, W is obtained by imposing a threshold for the cost $f(p) > c$ which is attained by

$$\int_{f(p)>c} \delta(p) dp < N \tag{3.2}$$

$$\frac{1}{\#Q} \int_{f(p)>c} p\delta(p) dp < W \tag{3.3}$$

This kind of selection is very powerful as patterns variance can have high magnitude and frequencies could mostly accumulate peripherally to the distribution**??**, therefore simply choosing highest entropy could lead to low compression levels as

3.2.2 Algorithm

There's therefore an unambiguous and general recipe to determine the optimal set of patterns that a generic pattern filtering system should use in order to achieve maximum information preservation under the given constraints. By tweaking the parameters N and W we can choose the cusp maximum and consequently the patterns Fig.3.2. There can be identified phases 3 phases: a **learning** phase

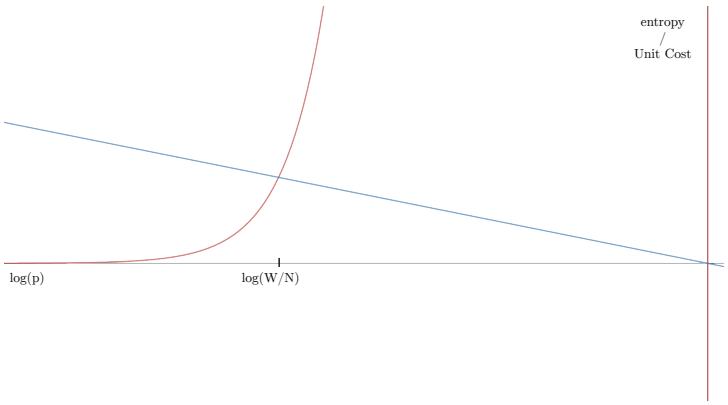

Figure 3.2: function 3.1 plotted with respect to a change of variable in Horizontal axis. Blue curve: limited bandwidth and unlimited pattern storage capacity ($W = 0.001, N = \infty$); Red curve: limited storage and unlimited bandwidth ($N = 100, W = \infty$); Red and Blue curve: limited bandwidth and storage ($N = 100, W = 0.001$).

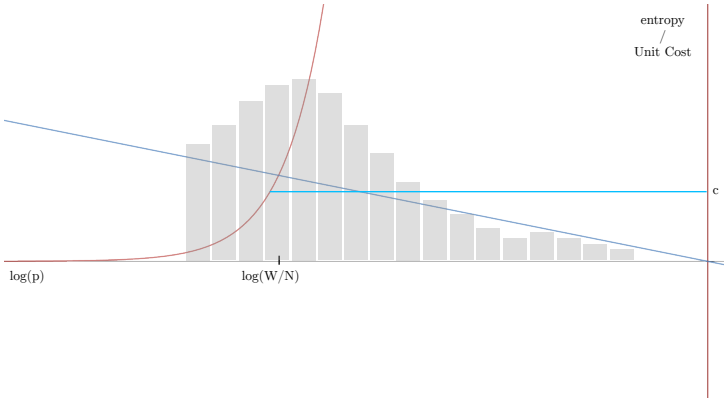

Figure 3.3: grey histrogram $\delta(p)$ is a unimodal fictional distribution chosen for illustration purposes, the selection window is defined by $f(c)$ by formla 3.2 and 3.3

in which all possible feature are sampled from uncompressed input streams, a **selecting** phase in which the salient pattern or features are extracted by applying Eq.3.2, 3.3 and a **compression** phase in which new unseen input streams are filtered using the set of extracted features.

3.3 Implementations

Such algorithm can be implemented in many ways, but still it has to be able to cope with the problem that has been exposed earlier: a filtering system with a limited acceptance bandwidth must process data coming from an high throughput source and the chosen implementation must operate accordingly to such constraints.

3.3.1 Complexity

The fundamental operation in this compressing method is **pattern matching**. A simple implementation as array access yields a time complexity of $O(1)$, spatially this implementation might need an exponential number of entries as the probability space could be any combination of the input, therefore space complexity is $O(2^n)$ and $2^n = \#(addrspace)$[9] with n being the number of bits for each pattern.

3.3.2 Algorithms for pattern matching

We cannot rely on this theoretical worst case scenarios, the pattern matching problem has to be faced using different approaches:

- Binary search: An order is defined on the the set of meaningful patterns thus a match can be acquired via binary search. With $n = \#(patternbank)$, this method matches with a number of comparison $O(log_2 n)$ both with binary search and binary heap.

- Hash Tables: hash tables can be used to efficently store and condense sparse keys into a fixed size array, this is a static case as *patternbank* is learned once: the simplest case to analyze is hash tables with linked list to handle collisions or hashing with caching [2]. In our case there is a $\#(patternbank) = n = O(m)$, with m being the number of entries to store n elements, and by having $\alpha = \frac{n}{m} = \frac{O(m)}{m} = 1$ it yields an average complexity of $O(1 + \alpha) = O(1)$ accesses for search. The pattern bank is known not being a uniform distributed function among the $addressingspace$ therefore n must be set to the nearest bigger prime to have a uniform hashing. By using the simple hash function $h(k) = kmod(n)$ the number of collisions can be in the worst case $\frac{\#(addrspace)}{n}$ as there are $\forall k \in (addrspace), \exists K = \{k_1, ..., k_m\} | \forall k_i \in K, h(k_i) = h(k), m = \left\lfloor \frac{\#(addrspace)}{n} \right\rfloor$ the probability of having only a collision for a particular slot is $\frac{1}{m}$ so even without perfect hashing functions we can be confident of good performance.

- Bloom filter: an algorithm that can be used to infer membership checking of an element to a set: For a given set $X, X \subset U$ a set of hash functions $K = \{h_i : U \mapsto \mathbb{N}, i \in \{1, .., k\}\}$ are selected, each $x \in X$ is hashed multiple

[9] will refer to the space of all possible patterns as *addrspace* to be consistent with the hardware implementation in which each pattern is referenced by using an address in the memory

times, a boolean table of size m is used to store a 1 at positions $h_1(x), ..., h_k(x)$. This method is sensible to *False Positives* and the uncertainty can be measured analytically: $m >= n \log_2 e \log_2 \frac{1}{\epsilon}$, with $n = \#(patternbank)$, m = number of bits used for the Bloom filter, ϵ = probability of *False Positive*, we assume that the cost to compute the k hash functions is $O(k) = O(1)$.

- Mixed: By taking $X = (patternbank)$ and $U = (addrspace)$, we can join a bloom filter and an hash table, the case of $x \notin X$ is handled by the Bloom filter, the complementary case will be handled by a lookup in the hash table, the overall cost can be analyzed:

$$X = \begin{cases} c_s & p = \frac{\#X}{\#U} + \epsilon \\ O(1) & o.w. \end{cases} \tag{3.4}$$

c_s is the cost of a lookup in the hash table.

$$E[X] = c_s(\frac{\#X}{\#U} + \epsilon) + O(1)(1 - (\frac{\#X}{\#U} + \epsilon)) \tag{3.5}$$

3.3.3 CAM in general purpose architecture

The term associative memory or AM is widely used in Pyhsics, Computer Science and electrical engineering. Generally speaking it indicates an architecture or function that given a key returns a value: $AM : K \mapsto V, key \in K, value \in V$[10]. This technology is used on the CPU chips to handle logical to physical address translation (MMU) and to address cache memories. A theoretical model of a computer architecture has:

- Address size w_{size} = number of bits per word
- Main memory addressing space $Maddr_{space} = 2^{w_{size}}$
- Cache memory addressing space $Ccaddr_{space} << Maddr_{space}$

To use caches to implements efficient pattern matching $Ccaddr_{space}$ needs to cover the entire pattern bank, this could be achieved by assuming for simplicity an architecture with words of $w_{size} = patternsize$. As mentioned before the set of meaningful patterns could be a sparse subset of the probability space therefore might

[10] Also hardware look up table or LUT has a similar functioning

not references contiguous addresses: cache memory uses *set associative* strategies to remap physical address in main memory to a memory block in the cache3.4, therefore there might be no possible way to control remapping of the address. However this must be investigated case by case are there could be workarounds on some architectures that provide a static use of caches. To match an address generated from the processor, caches are implemented in hardware with CAM that take one cycle to match the key at input port to the matching location. Caches are organized in hierarchical levels growing in size and time of access: the lower the number the closer to the processor unit. This logic takes a setup time of ~ 3 cycles of CPU clock at cache L1 (MMU translation from logic address is considered) up to ~ 100 cycles in L3.

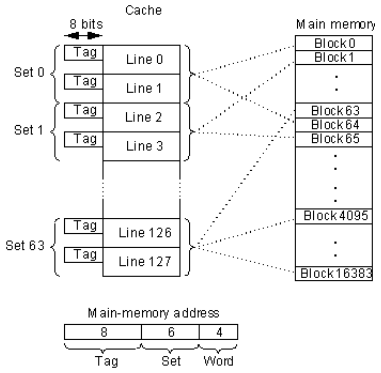

Figure 3.4: K-way set associative cache memory can map sparse addresses in memory, In this example its shown a 2way set associative cache.

K-way caches For $Ccaddr_{space}$ to cover the entire pattern bank means that it should be stored all in cache memory. The widest memory in the hierarchy are L3 caches[11]. For example a 6 MB, 8 way set associative cache means $\sim \frac{2^{23}}{2^6} = 2^{17}$lines of cache, if we assume a 64B memory block grouped in 2^3 ways memory yields a $\sim \frac{2^{17}}{2^3} = 2^{14}$sets. This implies 14 bits address for each set and 3 bits for tag parallel matching. L3 cache is usually shared among cores, and introduces a ~ 100 cycles latency[12]. With this kind of ideal dedicated caching a pattern could match at a speed of ~ 24 Mhz ($\frac{2.4\text{Ghz}}{100}$). Furthermore L3 caches adopts prefetching policy, as it might not be possible to circumvent this logic there's a considerable overhead of unuseful overhead to be considered.

[11] 6 MB, 8 way associative cache of i7-3630QM Processor

[12] any kind of smart cache technology will not be discussed

Complementary overhead The complementary case of a pattern not matching cannot be handled without encountering complications. In case of cache miss an access to main memory has to be performed, this may occur if pattern bank is all stored in cache as there's no dedicated logic to this case. This might cause the loading and the replacing of a memory block into cache memory depending on the replacing algorithm (LRU). This side effect degrades predicted matching performances even more, and has to be investigated case by case: impact of cache miss rate are unknown a priori as this depends on the probability of a pattern occurring in an instance.

Custom hardware benefits vs general purpose General purpose could in theory be suitable to deal with the presented problem, but a smaller addressing space and the lack of dedicated logic, makes it hard to predict performances with precision. Hence in order to have performance under control custom hardware is essential, especially in online computation when this task is a part of a bigger pipeline like the*multi-level Trigger* and loss of data is costly in terms of energy consumption of LHC. A middle way solution would be to implement pattern matching on a Many Integrated Core Architecture like the *Xeon phi*.

3.3.4 Custom Hardware

Implementing a powerful real-time selections is essential to filter background noise and record only meaningful information. FTK is a high-performance embedded system based on the combination of two widley used technologies: FPGAs working with standard-cell ASICs, the Associative Memory (AM) chips[4]. At the core of this system is the high parallelism provided by the AM chips, these chips share the same pattern matching capabilities of CAM[13] but the design is conceptually different.

AM chip

FTK specifies a particular type of inputs and output: *HITS* are the raw spatial data provided as low resolution track candidates, *ROAD* are the intermediate products returned as output to the later stages in the tracking system.

AM chip matches the inputs with patterns stored in a *patternbank* in parallel. Each row is composed of 8 words, each words is 16 bits long. A typical setup

[13]Content-Addressable Memories

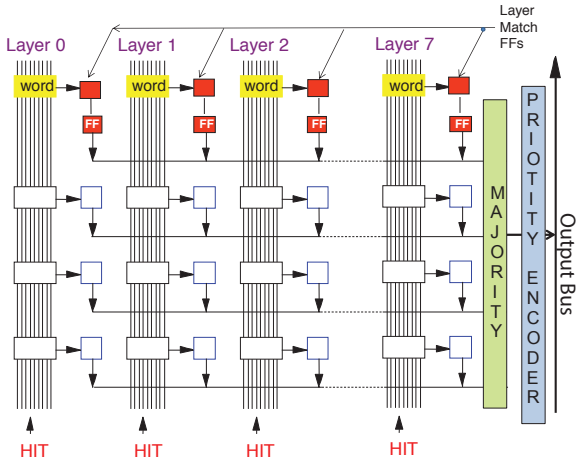

Figure 3.5: The AM chip architecture.

is to store a pattern in one or more rows (Fig.3.5) The 8 words buses in input carry the HIT coming from different levels of the detector[14]. Every HIT at the input is matched on columns against an independent CAM bus (Xor + RAM). The bitwise comparison's result is then stored in a flip flop, a voting systems sync the independent CAM buses. In (Fig.3.5) *FF* means that all 8 words matched and 8 ones have been written on the bus that routes the flip flops to the majority unit. If a pattern matches or "fires" the input HIT becomes a ROAD and the address of the matched pattern as row index is provided as output. AM chip fast pattern matching capabilities are achieved thanks to the parallel architecture. In the latest version[15] 128 000 patterns are matched at a rate of 100 Mhz. The entire FTK trigger has 128 boards with 64 AM chips each. FTK is very efficient as the throughput of HITs is on average 50 MHZ.

- pattern = 128 bit
- pattern bank per chip = 128 $\times 10^3$ pattern
- matching capabilities per chip: $128 \times 10^{11} \frac{\text{pattern comparison}}{\text{sec}}$
- matching capabilities per board = $3.276810^{15} \frac{\text{pattern comparison}}{\text{sec}}$

[14] complexity of a detector may introduce delays among the different layers. AM chip hardware stores the result of the temporary match in the flip flop looking for correlation with other layers HIT

[15] AM chip 06

Hardware and Software implementations.

This model can be implemented in hardware using the AM chip. Once the patterns are selected and loaded into the CAM buses the compression phase can be performed with parallel pattern matching. On the software side the simulation of the algorithm is an important step in the developing of a technology, therefore even if custom solution have better performances also software implementation must be explored in order to highlight technological differences and validate the hardware results.

Chapter 4

Image compression with Del Viva Model

As described in [3] images are processed by extracting the optimal visual patterns from natural Image statistics. The first attempt has been made with binary images. Binary images are obtained by applying a suppressing threshold at the mean luminance value of greyscale images. frequencies are sampled by superimposing each pattern at all possible position on the sample image, the frequencies are then clustered in bins from which the distribution δ in 3.2 3.3 is obtained.

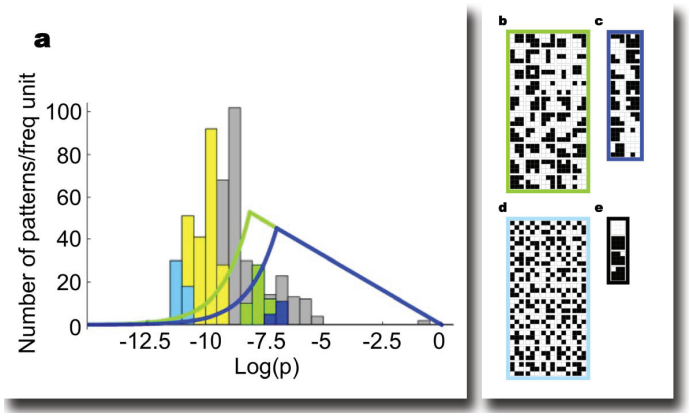

Figure 4.1

In applying this model to vision, the simplest possible set Q of base patterns have been considered, defined as all possible configurations of 3×3 square pixel matrices in black-and-white images (1-bit depth) yielding $\#Q = 512$ possible patterns. The set of meaningful patterns has been extracted form the probability

distribution of the patterns in a set of natural images. For this purpose a public database of 560 calibrated natural pictures has been used. The choice of the algorithm parameters (N, W) was based on the following considerations. Since the algorithm revolves on the idea of a strong compression at the minimum possible computational price, compression parameters have been set to $W = 0.05$ and $N = 16$ as a "bare minimum" to be able to handle at least a few different spatial orientations. A reasonable upper bound has been taken: N is a value of 10% of all possible distinct patterns. Given that only 512 total distinct patterns are possible in our basic 3×3 model, N = 50 is the limit.4.1

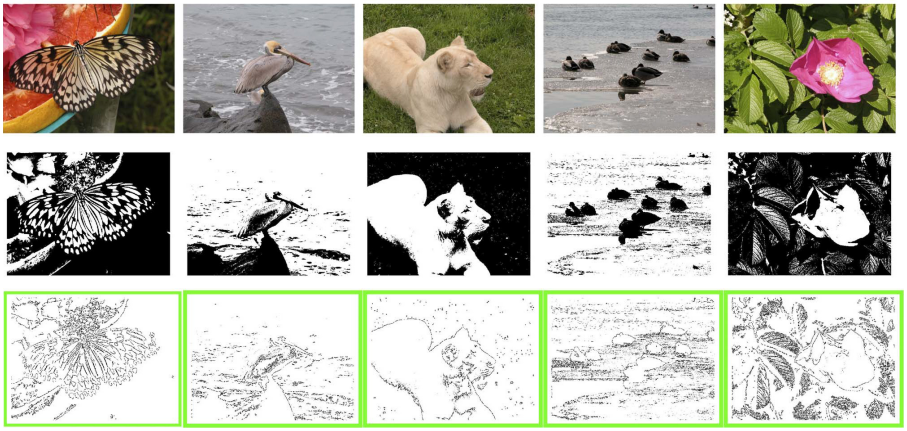

Figure 4.2: A series of "natural" images compressed using the descripted method in which only 10% of the overall patterns were preserved

4.1 Experimenting on Medical Images

The image processing capabilities of the model are investigated further as an experimentation project during the period of development of this work, roughly one month span. Our contribution was to generate an implementation of the presented method and to experiment on MRI images.

4.1.1 Medical Imaging

MRI images are defined in terms of:

- Voxels: the basic unit of an MRI image is a pixel with 3D coordinates: it differs from the graphical unit point or vector as it has 3 positive integer indexes.

- Tissue Signal Characteristics: T1, T2 viscosty wieghtings [12] are standaridized grey scales luminosity values representing a tissue consistency. Weighting may depends on the MRI scanner used in the analysis.
- Affine: 4x4 matrix (12 degree of freedom) representing the affine transformation of the origin absolute point of view (register) in an Anatomical locations system.
- Terms of Anatomical location system: RAS is the nomenclature of 3 origin orthogonal axes: left to Right, posterior to Anterior and inferior to Superior, respectively. RAS+ meaning that Right, Anterior, Posterior are all positive values on these axes, also known as "real-world" coordinates or neurological convention.

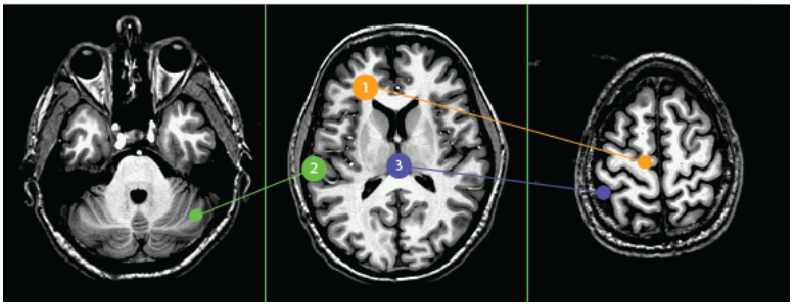

Figure 4.3: T1 weightings, 1 white matter, 2 grey matter, 3 cerebrospinal fluids

Voxels luminance values are representative of the density of the tissue located in at a given position. In cerebral MRI there are 3 to 4 main areas of grey. Such areas correspond roughly to 4 tissue consistencies in T1 viscosity scale: cerebrospinal fluid(CSF), grey matter(GM), white matter (WM), bones and vessels, Fig.4.3. This noticeable difference between grey levels induced researchers at INFN experimenting on AMchip implementation of the image compression method designed by Del Viva, Punzi et al. to evaluate the possible application for MRI imaging as with such a low bit depth patterns can be used efficiently.

4.2 Data Analysis methods and key words

Patches The *patternbank* elements or patterns have the same dimensions as the sampling unit, more simply called image *patches*.

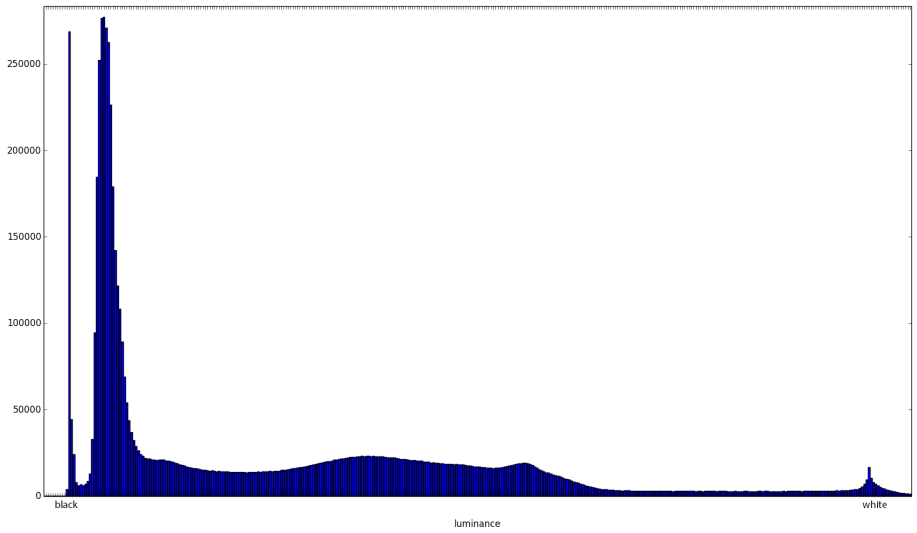

Figure 4.4: Grey level historgram computed for a patient: two pillars at the left periphery and a 3 modal plateu (respectively CSF, GM, WM) in the mid low values can be identified, the small peak of wihte at the right periphery is bones adn vessels value

Sampling The frequency of a pattern is sampled from an instance image: each pattern is stored with the relative frequency sampled to create the *frequency histogram*.

Quantization After frequencies are sampled and *patternbank* is selected a new unseen images data is quantized to a series of *pathces*.

Merging By merging the average value of many *frequency histogram* instances we calculate a *probability histogram*

Binning *probability histogram* uses the $\log_e$ scale, these values are floored according to a parameter *bin factor*, the sum of all patterns having the same *bin* value yield a distribution δ.

4.3 Implementation and format

For this experiment a set of images Nifti-1[1] was used: this format is the new standard in medical imaging (previously ANALYZE 7.5 was used but independent updates made this format ambiguous), it is comprensive of a .nii storage that

[1] http://nifti.nimh.nih.gov/nifti-1

contains: the header .hdr and the voxels .img: the header contains the relevant information for MRI scan: affine, weighting(T1 or T2), and orientation (RAS or others).

Software Used

- Python 2.7: scripting language
- Numpy: python package for multidimensional array handling
- OpenCV 2: python package for image processing
- Nibabel: python package for Nifti handling, provides Python interface as multidimensional array.
- Pickle: python packag providing serialization of simple data structure.

Software Developed

- *MDVcomp*: a package for python capable of: sampling 2D and 3D images, a merger used to perform analysis on data (binning), depends on:
  - Numpy, OpenCV 2, Pickle
- *niiImaging*: a package for converting Nifti1 .nii data into PNG images, depends on:
  - Numpy, Nibabel, OpenCV 2

4.3.1 Compression and performances

A full 3D filtering will be the next step, for now an experiment has been run on 2D slices of MRIs to evaluate the results on smaller instances. From a *probability histogram* the patterns with values in the bin adjacent the maximum were selected by highest entropy Fig 4.6. The *middle left to right* image is seen from the *left to Right* axis with coordinates at half of the maximum value on such axis (mid): in Fig 5.1 first image on the left is provided as input, the second is the uncompressed thresholded image and the third is the compressed image.

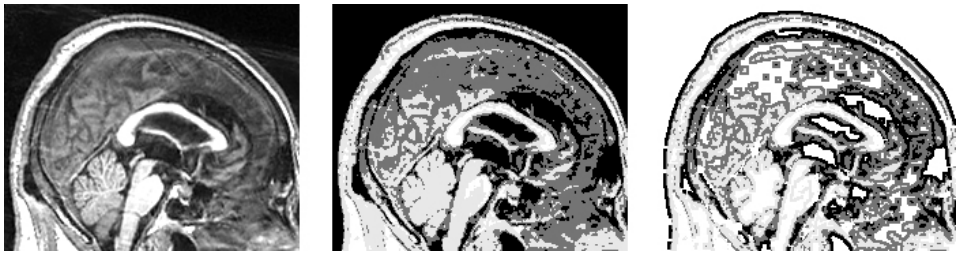

Figure 4.5: From the original central image to the compressed version

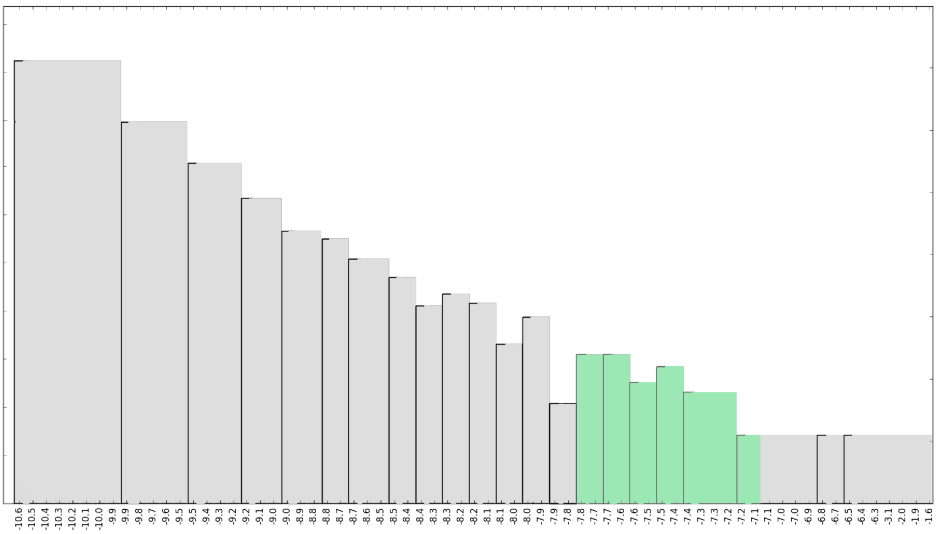

Figure 4.6: $\delta(p)$ learned *binning* from the 8 patient set, *patternbank* was selected from one bin adjacent the maximum (RED) $logW/N = -7.5$, vertical axis representation is scaled in *log* for figurative purposes

Compressiopn method

The image is sampled with a *patch* 3×3 pixels $\times 2$ bits depth greyscale, the ($addrspace$) in this case has size 2^{18}, with such a wide range we put heavy constraints, the parameters used for compression are N = 180 as 5% of all the possible patterns sampled, and W = 0.1. The *patternbank* is learned from 2D images taken from the same mid left to Right position on a set of 8 patients Fig4.6.

Dictionary Encoding In the quantization phase, each pattern that matches is stored in coordinate list at the *patternbank* index, with the relative 2d coordinate (x, y) . For every new occurrence of the same pattern, only the couple (x, y) is appended to the coordinates list, hence index pattern is stored only once.

Offset Encoding An alternative encoding is done by saving for each position in the image the index of the pattern that matched that portion of data. This implementation however never showed significant performances.

SwitchedOffPixels Encoding This type of encoding considers the "edgy" nature of the image, by saving only pixel on edges it can be saved a lot of space. However in order to store the offset each pixel has to be stored with $n + 1$ bits, with n being the grey scale bit depth. This needs to be done in order to tell offsets from pixels.

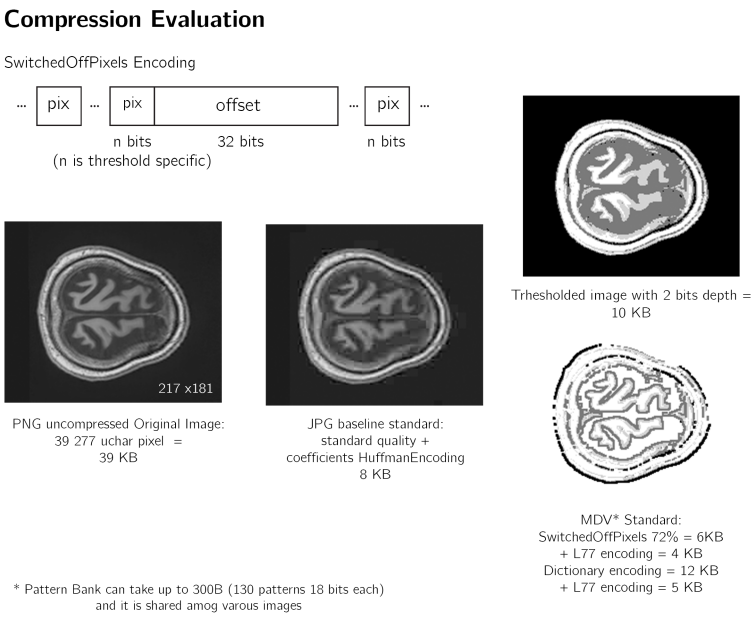

Figure 4.7

Drawbacks The best compression is achieved with the SwitchedOffPixels encoding. However there are drawbacks. Dictionary encoding is simpler for the hardware to implement, whilst SwitchedOffPixels may need more complex hardware as the image has to be reconstructed with the encoding. A dictonary encoding could also be used for progressive dencoding as patterns are replaced sparsely over the image.

file format

- header: *patternbank* : $pattern_0$,..,$pattern_{N-1}$
- data options:
  - Dictionary: $(x_0, y_0, x_1, y_1, ., x_j, y_k\ )$, ... , $(x'_0, y'_0, x'_1, y'_2, ., x'_j, y'_k))$, index of the list correspond to index of the pattern in the *patternbank*.
  - Offset: $pattern_{index_i}$,off_delim,ofs,off_delim,$pattern_{index_j}$.... , with ofs being the number of empty pixels to a jump.
  - SwitchedOffPixels: ... , offset,$pixel_{i,j}$, ..., $pixel_{i+l,j+k}$, offset, ... , with offset being = overflow pixel, int

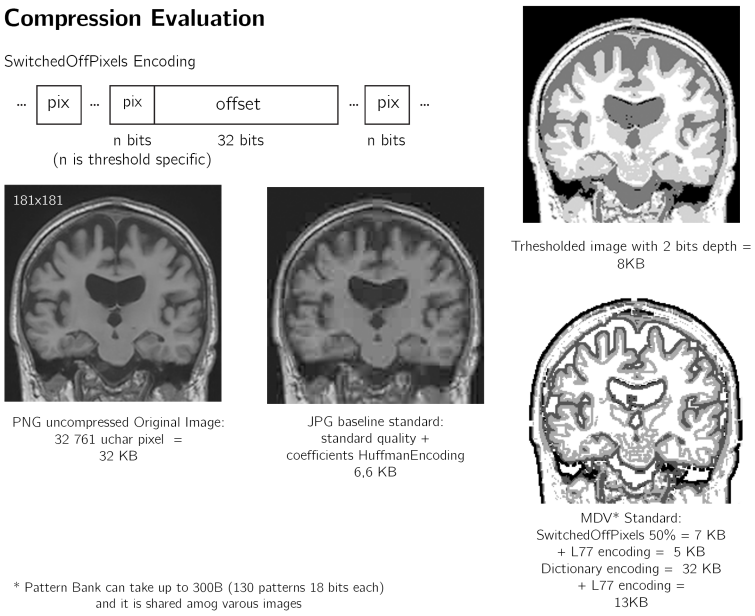

Figure 4.8: as shown in the images above compression ratio depends on the image itself, sparse images are common in some MRI areas, but also very busy regions are present near center of the image

Chapter 5

Conclusions

5.1 Compression results

The comp ratio vary on the type of image: as an edge detector very sparse images can be compressed very efficiently whilst others "busy" images might not be as suitable candidates for this approach. Infact there are cases in which images are actually inflated as it will be discussed later in the section about overlapping patterns.

5.2 Possible image processing applications

This experimentation done on medical images highlighted that the method is capable of edges detection. This simple approach is faster than other method, like cv2.Canny[1]. Another possible application regarding MRI processing is the possibility to exploit the edge detection capabilities on 3D features to make surface mappings (MESH)5.2 easier, this could help to make a brain's atlas.

Canny Complexity

- convolution pass: *Sobel*= $N \times M^2$, whit N being the number of pixels per image and M the size of the *Sobel* Kernel.
- Canny complexity = 2 convolution pass *Sobel* + 2 N (computation of the gradient vector∇)

[1] openCV http://opencv.org/

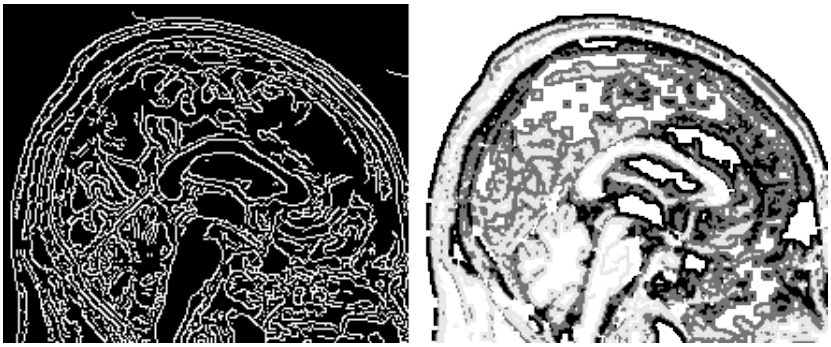

Figure 5.1: On the left the edge detector cv2.Canny has been applied, in this case discreete derivative approach is cutting off relevant data, different grey levels are meaningful and they're conserved in the right image.

5.3 Threats

With the potential benefits of this methods, there's still a number of unanswered questions and pitfalls in this research that must be investigated:

5.3.1 Image processing known weakness

A desirable property for a computer vision/image processing method is being rotation invariant, many state of the art methods try to cope with this problem by providing transformed inputs as a pre processing phase. Even thou there are studies on local binary pattern or LBP that are rotation invariant[9] image processing methods suffer from this weakness. This method is no exception but the MRI scan format can mitigate this problem as the .nii format has absolute coordinate system, so data can be aligned to a standard position. The problem is mitigated but not completely solved as the origin of such spatial coordinates system is not in the observed object, but in the observer point of view (the camera or in this case the magnets of the MRI scanner), therefore there's no perfectly aligned absolute head position to be consistent with, but only an average standard positioning that patients have to assume when analyzed and even few degrees may do a big difference.

5.3.2 Overlapping patterns

In this step there's no meaningful compression of data, rather information is compressed as meaningful patterns. In order to simulate the way light interacts with retina photoreceptors, patterns overlaps so that are invariant in the offset.

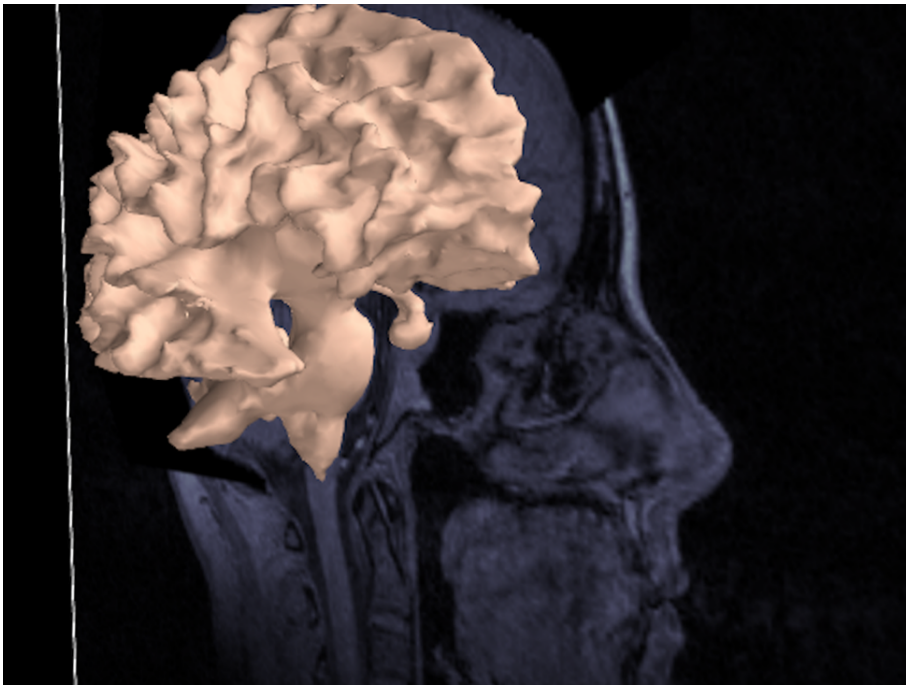

Figure 5.2: A 3d mesh of WM selection

The result may introduce redundant data as the patterns are not an unambiguous encoding of the image, in order to have such code, there should be no sub portion of any pattern overlapping with any other pattern, this could be a next step in the learning process that has not been considered from the model. As shown before for an instance image of size 181× 201 pixels was encoded using 130 patterns. Each pattern is stored alongside with a

5.3.3 Scalability

In order to apply this method to MRI 3D voxel images *pathces* must have different dimensions form the 2D approach, the size of $addrspace$ scales esponentially with the size of the *patch*

$$\#(addrspace) = 2^{dim \times bitdepth} \tag{5.1}$$

although this seems unreasonably huge the chance to sample all the possible pattern decreases as the space increases. As an example if there were to be sampled 3d images $100 \times 100 \times 100$ with a *patch* of dim:($3 \times 3 \times 3$), 2 bits of depth, assuming uniform distributed pattern (which is a true only for noise "salt and pepper images")

the probability of a d.r.v[2] X selecting all different patterns:

$$p(X = \text{sampling all different patterns}) = (\frac{1}{2^{dim \times bitdepth}})^{100 \times 100 \times 100} \tag{5.2}$$

Is the uniformity was not the case, $p(X)$ would be even less as there would be pattern that are more frequent than others.

5.3.4 Overfitting

There has to be considered the *training set* and the *test set* are the same. Images were eventually self learned and therefore there might be a form of overfitting. At the mean time no loss function has been defined to qualitatively evaluate the compression quality. A evaluation of this method could be to train two classification algorithm to try to recognize the same labeled data from a compressed and an uncompressed set.

5.4 Next

In a possible next phase there could be the possibility to apply the presented method to a *labeled* data set and evaluate 2 different *Machine Learning Classifier*. For example *Conv NN*[3] could be trained on two different sets, one compressed and one uncompressed but with the same labeling, and evaluate how the classification performs . This might also give a better evaluation metric for the method itself.

[2] discrete random variable

[3] Convolutional Neural Networks are the standard machine learning approach to image classification and object recognizer

.1 Appendix

.1.1 Function Study

The function 3.1 can be studied in two separate intervals.

$$f(p) = \frac{-p \log p}{\max(1/N, p/W)} \tag{3}$$

$$f(p) = -p \log pN, p < W/N \tag{4}$$

$$f(p) = -\log pW, p > W/N \tag{5}$$

with the change in variable $x = log(p)$ there can be found the equivalent formulas.

$$f(x) = -xW, x > \log W/N \tag{6}$$

$$f(x) = -e^x xN, p < \log W/N \tag{7}$$

the intersection point is shown in Fig 3.2 as the cusp at blue and red curve intersection.

.2 Ringraziamenti

Ringrazio il mio relatore: prof. Vincenzo Gervasi per avermi supportato in questo percorso eterodosso, la prof.sa Paola Giannetti per avermi accolto nel Team di sviluppo e per avere avuto la pazienza di introdurmi a concetti di HEP, ringrazio Hikmat Nasimi per avermi portato a conoscenza di questo progetto interessante e per aver condiviso serate alternative a spulciare la documentazione. Ringrazio tutto il Team di FTK per la calorosa accoglienza, i consueti genitori, i familiari e anche chi è superfluo ringraziare dato che avrà sempre la mia gratitudine.

Bibliography

[1] S. Citraro, N. Biesuz, P. Giannetti, P. Luciano, H. Nasimi, M. Piendibene, C.-L. Sotiropoulou, Member, IEEE and G. Volpi (confirmend 2016) Highly Parallelized Pattern Matching Hardware for Fast Tracking at Hadron Colliders, IEEE Transactions on Nuclear Science

[2] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, Charles E. Leiserson (2001) Introduction to Algorithms

[3] Del Viva , Punzi , Benedetti (2013) Information and Perception of Meaningful Patterns. PLoS ONE 8(7): e69154. doi:10.1371/journal.pone.0069154

[4] M. Dell'Orso and L. Ristori (1989) VLSI Structures Track Finding, Nucl. Instr. and Meth. A, vol. 278, pp. 436-440.

[5] Huffman (1952) A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the IRE 40 (9): 1098–1101. doi:10.1109/JRPROC.1952.273898.

[6] RANDALL C. REININGEK AND JERRY D.GIBSON, MEMBER IEEE (1983) Distributions of the Two Dimensional DCT Coefficients, E TRANSAOCN-TIONS COMMUNICATIONS, VOL. COM-31, NO. 6, JUNE

[7] Claude Shannon (1948) A Mathematical Theory of Communication, 10.1002/j.1538-7305.1948.tb01338.x

[8] Welch, Terry (1984) A Technique for High-Performance Data Compression. Computer 17 (6): 8–19. doi:10.1109/MC.1984.1659158.

[9] G. Zhao, T. Ahonen, J. Matas and M. Pietikainen, (2012) Rotation-Invariant Image and Video Description With Local Binary Pattern Features, in IEEE Transactions on Image Processing, vol. 21, no. 4, pp. 1465-1477

[10] (2013)Fast Tracker for Hadron Collider Experiment FP7-PEOPLE-2012-IAPP An FP7 IAPP project (February 1, 2103 - January 31, 2017) Grant Agreement Number 324318

[11] JPEG ISO/IEC 10918-1 ITU-T Recommendation T.8

[12] http://www.med.harvard.edu/aanlib/basicsMR.html