UNIVERSITÉ DE NANTES

Polytech'Nantes
École d'ingénieurs de l'université de Nantes

STAGE DE MASTER 2R

SYSTEMES ELECTRONIQUES ET GENIE ELECTRIQUE

PARCOURS SYSTEMES ELECTRONIQUES

ANNEE UNIVERSITAIRE 2009/2010

3D VISUALIZATION OF MOLECULES BY LATTICE VECTOR QUANTIZATION

SIYA LUO

Encadrants du stage :

Vincent Ricordel (IRCCYN, équipe IVC, Nantes)

Bogdan Cramariuc (CITST, Bucarest, Roumanie)

Stage effectué du 25/02/2010 au 7/07/2010

Abstract

This paper presents a method for 3D visualization of the molecule based on the lattice vector quantization (LVQ). The 3D visualization of the molecules involes executing a merging procedure for the corresponding raw data. The whole procedure makes use of LVQ for designing the codebook. The merging procedure includes calculating the average of the blocks and comparing the values to a threshold. We also use the iso-contours plot to display the molecules. We use the hydrogen fluoride (HF) and water (H_2O) molecules in our experiments. The experimental results are given through the histograms and LVQ plots.

Keywords: vector quantization, lattices, merging, tree structure, molecule,
MED (molecules electron density)

Résumé

Cet article présente une méthode pour la visualisation 3D de molécules basée sur la quantification vectorielle avec lattices (réseaux réguliers de points). La visualisation 3D de molécules nécessite l'exécution d'une procedure qui fusionnent les données. La procédure de fusion implique le calcul de moyennes et d'erreurs sur les blocs, et des comparaisons à un seuil. Nous utilisons également des iso-contours pour afficher les molécules. Nous utilisons pour mos experiences, les molécules HF et d'eau (H_2O). Les résultats des expériences sont données avec des histogrammes et des images résultats (visualisations de molécules)

Mots-clés :quantification vectorielle, lattices (réseaux réguliers de points), fusion,
structure d'arbres, molécules, champ de densité électronique

Contents

1. INTRODUCTION

The context is about the computer modeling of organic molecules in biochemistry. The vector quantization (VQ) is commonly used in telecommunication or compression for the representation of data. Here we want to use the VQ for the 3D visualization of organic molecules. The problem is how to find a good criterion to process the raw data. In practice, the raw data are spread out in the electronic density map that corresponds to the molecule. By using the VQ, we want to merge the 3D space in cells whose size will be function of the local density of the points. Exactly we will use an approach based on a tree-structured lattice VQ. Exactly, we plan to use a merging VQ method for the visualization of the molecules.

This paper mainly contains three chapters. The following chapter is the background that mainly describes currently fashion lattice for VQ such as the cubic lattice and some basic principles. The approaches and implementation we employed are introduced in the second chapter. The third chapter introduces the results of the experiments of different steps. Some conclusions and future works are illustrated in the last chapter.

2. BIBLIOGRAPHY

2.1. Scalar Quantization

Quantization is the heart of analog-to-digital conversion. In its simplest form, a quantizer observes a single number and selects the nearest approximating value from a predetermined finite set of allowed numerical values. Ordinarily the input value is analog; that is, it may take on any value from a continuous range of possible amplitudes; furthermore, the output is digital, being uniquely specified by an integer in the set $\{1, 2, 3, ..., N\}$, where N is the size of the set of output values.

More precisely, we define an N-point scalar (one-dimensional) quantizer Q as a mapping Q: $\mathbb{R} \rightarrow C$ where $\mathbb{R}$ is the real line and

$$C = \{y_1, y_2, y_3, ..., y_N\} \subset \mathbb{R} \qquad (1)$$

is the output set or codebook with size $|C| = N$. The output values y_i, are sometimes referred to as output levels, output points, or reproduction values.

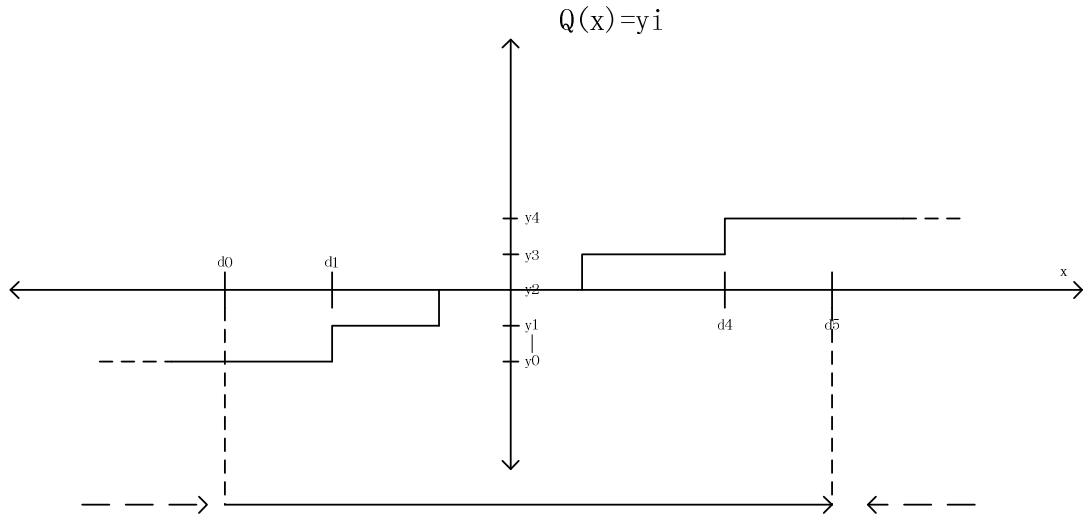


Fig.1 Example of Scalar Quantization

For example: in Fig.1, the input is x, and $\{y_0, y_1, ..., y_i\}$ is the output set or codebook. $\{d_0, d_1, ..., d_i\}$ this set is the boder of the decision cells.

We define the resolution or code rate, r, of a scalar quantizer as $r = \log_2 N$ bit/simples which measures the number of bits needed to uniquely specify the quantized value. The resolution indicates the accuracy with which the original analog amplitude is described. Specifically, if r is an integer, one could assign to each y_i a unique binary r-tuple (by coding the values y_i into binary vectors in an invertible manner). We define the transmission rate, R, of the quantizer as the number of bits transmitted per sample to describe the waveform [1].

2.2. Vector Quantization

VQ has been investigated extensively in recent years. Let $x(n) : (x(1), ..., x(n))$ be a n-component source vector with joint probability density function (PDF) $p_X(x)$. A VQ of dimension N and size L is defined as a function that maps a vector x into one of L reproduction vectors $y_1, ..., y_n$ belonging to $\mathbb{R}^k$. We have:

$$Q : \mathbb{R}^k \rightarrow D$$

$$x \rightarrow Q(x) = y$$

Where D, the set of reproduction vectors or codewords, is the codebook. This chart implicitly determines a partition of $\mathbb{R}^k$ in L non-overlapping Voronoi regions C_i defined by the equation (where $L_p(x) = (\sum_{i=1}^{k} | x_i |^p)^{1/p}$ is the norm):

$$C_i = \left\{ \begin{matrix} x \in \mathbb{R}^k / Q(x) = y_i \\ L_2(x - y_i) \leq L_2(x - y_j), \forall j \neq i \end{matrix} \right\} \quad (2)$$

The total distortion [per dimension] is given by:

$$D = \frac{1}{k} E\{L_2(X - Q(X))\} = \frac{1}{k} \sum_{i=1}^{L} \int_{C_i} L_2(x - y_i) p_X(x) dx \quad (3)$$

In this transmission and storage context, a binary word or index c_i of length b_i bits is assigned to each codeword y_i [1].

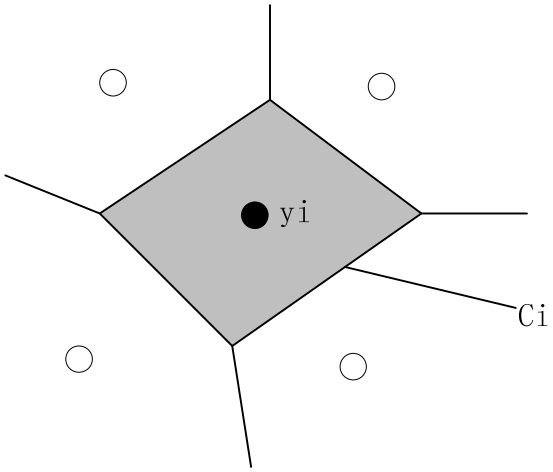


Fig.2 principle of vector quantization [2]

For example, in Fig.2, the Voronoi cells C_i divides the input space into several small spaces, we use y_i stands for the cell C_i .

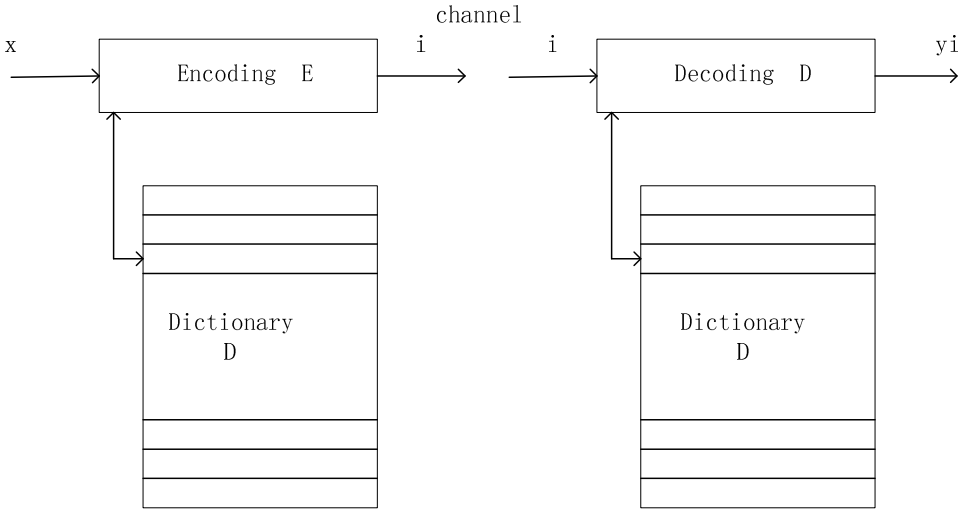


Fig.3 diagram of vector quantizer [2]

In this diagram, when the input vector x is coming, the encoder scans the codebook D, and find the closest vector y_i for the input. Then it just stores the index of y_i, and sends the index to the decoder. When the decoder received the index i, it scans the codebook D, which is the same codebook as before. After that, it finds the vector in the codebook, and sends the output y_i.

2.3. Codebook Design

2.3.1. The LBG （Linde,Buzo,Gray）method

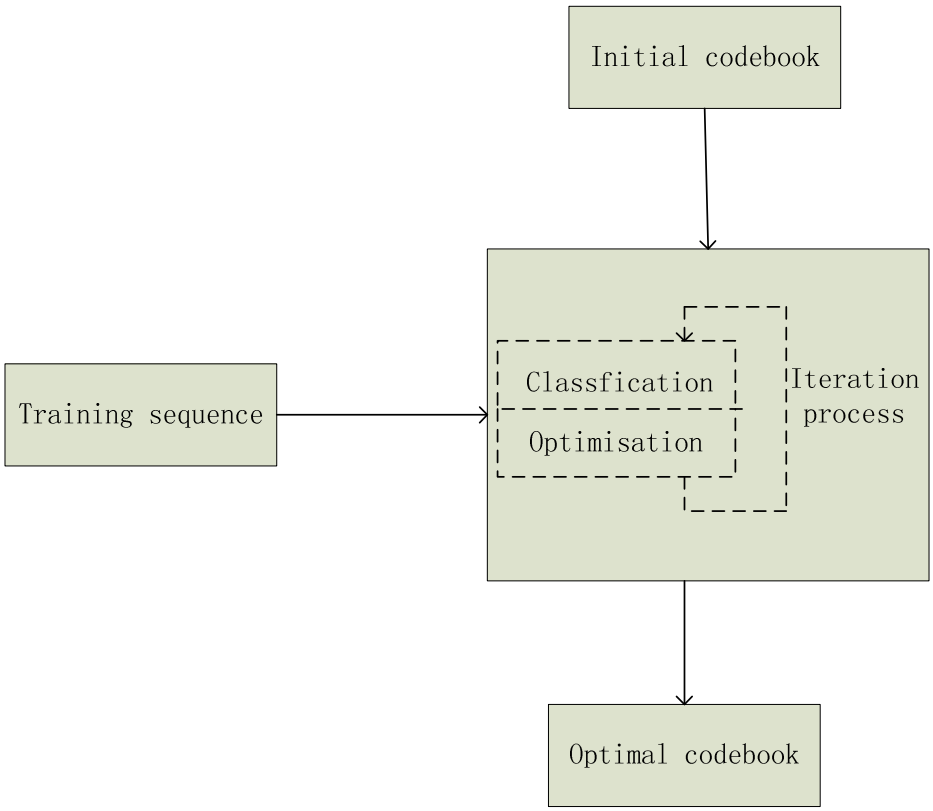


Fig.4 scheme operating in the LBG algorithm

LBG algorithm [2] is a optimal method for the codebook design from a training sequence. As the Fig.4 shows, the algorithm starts with a training sequence (TS). After a long reduplicate processing, the codebook will be optimized.

There are two defects, one is the computation that is very large. Another is because this algorithm

can just produce an optimal codebook for the TS, if we change the TS, the VQ is no longer optimized.

2.4. Lattice Description

The LVQ is based on the lattices. We are going to describe the lattices. At the beginning, the lattices were defined to solve mathematical problems like sphere packing problem, kissing number problem, and the covering problem.

2.4.1. Sphere packing problem

The classical sphere packing problem, still unsolved even today for high space dimensions, is to find out how densely a large number of identical spheres can be packed together. The cubes fit together with no waste space in between, we can fill essentially one hundred percent of the space. But spheres do not fit together so well as cubes, and there is always some wasted space in between.

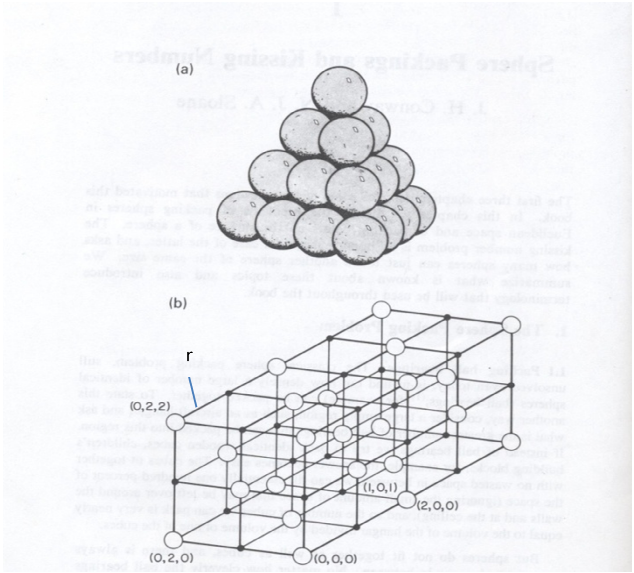


Fig.5 two views of the most familiar sphere packing [9]

Two views of the most familiar sphere packing, in which the centers form the face-centered (fcc) lattice. This is the packing usually found in fruit stands, in piles of cannon balls on war memorials, or in crystalline argon. The spheres occupy 0.7405… of the space, and each sphere touches 12 others. (b) shows only the centers of the spheres (the open circles). These centers are obtained by

taking those points of the cubic lattice Z^3 whose coordinates add up to an even number. The packing radius is ρ [9].

2.4.2. Kissing number problem

In three dimensions this asks how many billiard balls can be arranged so that they all just touch, or kiss, another billiard ball of the same size. The difficulty arises because the arrangement is not unique; in fact there are infinitely many ways to arrange 12 billiard balls around another. For example, if the 12 balls are placed at position corresponding to the vertices of a regular icosahedron concentric with the central ball, the twelve outer balls do not touch each other and many all be moved freely [8].

2.4.3. The covering problem

Another problem is a kind of dual to the packing problem, and asks for the most economical way to cover n-dimensional Euclidean space with equal overlapping spheres.

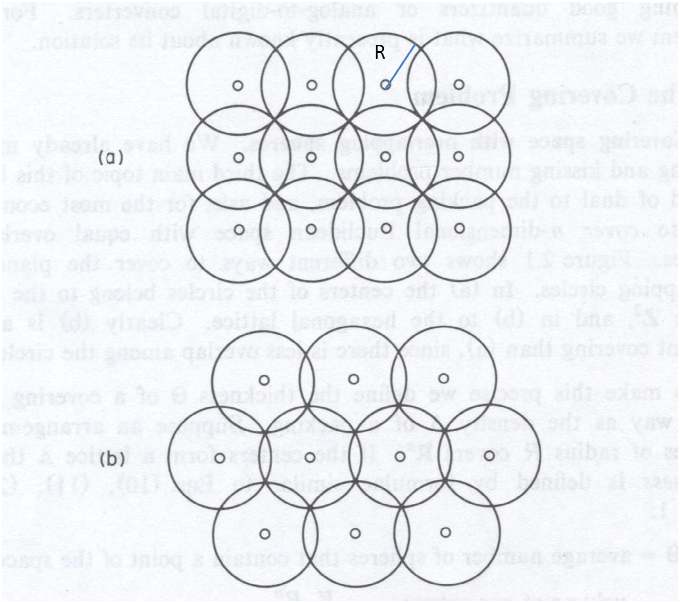


Fig.6 Covering the plane with circles. In (a) the centers belong to the square lattice Z^2, in (b) they belong to the hexagonal lattice (b) is a more efficient or thinner covering [8].

In three dimensions the best covering known, which Bambah showed is optimal among tree-dimensional lattice coverings, is the body-centered cubic lattice. This is at first sight surprising, since the densest lattice packing is a different lattice, the fcc lattice. In fact the bcc is

the dual of the fcc lattice. The covering radius is R [8].

2.4.4. Important lattices for the VQ

Some lattices are important for the VQ because fast quantizing algorithms have been defined for them.

- The lattice Z^n

The set of integers…,-2,-1,0,1,2,3,…is denoted by Z, and

$$Z^n = \{(x_1, ..., x_n) : x_i \in Z\} \tag{4}$$

is the n-dimensional cubic or integer lattice. (Z^2 is better called the square lattice, as seen in ordinary graph paper.) Kissing number τ =2n, and the minimal vectors are (0,…,-1,..,0). The packing radius ρ =1/2, the covering radius R= $\sqrt{\pi}$ /2= $\rho\sqrt{n}$, the density $\Delta = V_n 2^{-n}$ and the center density $\delta = 2^{-n}$. Thus Z has density Δ =1, but the densities of Z^2, Z^3 and Z^4 are only $\pi/4$ =0.785…, $\pi/6$ =0.524… and $\pi^2/32$ =0.328… . A typical deep hole is (1/2,1/2,…,1/2), and the Voronoi cell are cubes. Z^n is self-dual. Its group consists of all permutations and sigh changes of the coordinates, and has order $2^n n!$.

·The lattice A^n

For $n \geq 1$, $\quad A^n = \{(x_0, x_1, ..., x_n) \in Z^{n+1} : x_0 + ... + x_n = 0\}$ (5)

which uses n+1 coordinates to define an n-dimensional lattice.

Of course $A_1 \cong Z$. A_2 is equivalent(or similar) to the familiar hexagonal lattice. The hexagonal lattice may be spanned by the vectors (1,0) and (-1/2, $\sqrt{3}$ /2). Both A_3 and D_3 are equivalent to the face-centered cubic lattice (of fcc), illustrated in every chemistry textbook, and found in the pyramids of oranges on any fruit stand. The lattice dual to A_n is

$$A_n^* = \bigcup_{i=0}^{n} ([i] + A_n) \tag{6}$$

both A_3^* and D_3^* are equivalent to the body-centered cubic lattice (or bcc), also familiar from chemistry.

·The lattice D_n

For $n \geq 3$, $D_n = \{(x_1, ..., x_n) \in Z^n : x_1 + ... + x_n even\}$, (7)

or in other words D_n is obtained by coloring the points of Z^n alternately red and white with a checkerboard coloring, and taking the red points. D_n is sometimes called the checkerboard lattice. [9]

2.4.5. The fast quantizing algorithms

The algorithm for finding the closest point of Z^n to an arbitrary point $x \in R^n$ is particularly simple. For a real number x, let

$$f(x) = \text{closest integer to } x .$$

In case of a tie, choose the integer with the smallest absolute value. For $x = (x_1, ..., x_n) \in R^n$, let

$$f(x) = (f(x_1), ..., f(x_n)) .$$

For the flowing use we also define $g(x)$, which is the same as $f(x)$ except that the worst coordinate of x (that furthest from an integer) is rounded the wrong way. In case of a tie, the coordinate with the lowest subscript is rounded the wrong way [10].

• Algorithm 1: to find the closet point of Z^n to x.

Given $X \in R^n$, the closest point of Z^n is f(x). (If x is equidistant from two or more points of Z^n , this procedure finds the one with the smallest norm). To see that the procedure works, let $u = (u_1, ..., u_n)$ be any point of Z^n . Then

$$N(u-x)=\sum_{i=1}^{n}(u_i-x_i)^2 \tag{8}$$

which is minimized by choosing $u_i = f(x_i)$ for $i=1,...,n$.

·Algorithm 2: to find the closet point of D_n to x.

Given $x \in R^n$, the closest point of D_n is whichever of f(x) and g(x) has an even sum of coordinates.

If x is equidistant from two or more points of D_n this procedure produces a nearest point having the smallest norm.

The procedure works because f(x) is the closest point of Z^n to x and g(x) is the next closest. f(x) and g(x) differ by 1 in exactly one coordinate, and so precisely one of $\sum f(x_i)$ and $g(x_i)$ is even and the other is odd.

·Algorithm 3: to find the closet point of A_n to x.

Step1. Given $x \in R^{n+1}$, computes $s=\sum x_i$ and replace x by

$$x' = x - \frac{s}{n+1}\times(1,1,...,1) \tag{9}$$

Step2. Calculate $f(x') = f(x_0'),..., f(x_n'))$ and the deficiency $\Delta = \sum f(x_i')$

Step3. Sort the x' in order of increasing values of $\delta(x_i')$. We obtain a rearrangement of the number $0,1,...,n$ say $i_0,...,i_n$, such that

$$-1/2 \le \delta(x_{i0}') \le ... \le \delta(x_{in}') \tag{10}$$

Step4. if Δ =0, $f(x')$ is the closest point of A_n to x.

If Δ >0, the closest point is obtained by subtracking 1 form the coordinates $f(x_{i0}'),..., f(x_{i\Delta-1}')$

If Δ <0, the closest point is obtained by adding 1 to the coordinates $f(x_{in}'),..., f(x_{i\Delta-1}')$

2.5. The Lattice VQ (LVQ)

The LVQ [3][4][5](lattice vector quantization) has been successfully introduced in order to overcome the LBG-type algorithm [6] drawbacks. The encoding, based on rounding and scaling operations, is simple and independent of the codebook size. There is no need to seek among all the reproduction vectors, and practically no norms are computed. Because of the predefined structure of the lattice, there is no need to transmit the codebook and no training procedure is required to design it. But a lattice can only optimally quantize uniform source and, because of its infinite size, it has to truncate to index the codewords. Thus the LVQ of a non-uniform source becomes complex. We distinguish the different steps:

1. The lattice choice which defines the space filling advantage (the property is linked with the sphere packing problem) of the LVQ. Some best lattices are known for the dimensions 2, 4, 8, 16 and 24, because they offer the smallest distorsion when quantizing an uniform source associated with the lattice Z^k, D^k, E_8 and Λ_{16}, are the main criterion for the selection;

2. The determination of the codebook shape. When truncating the lattice, the aim is to map the most probable distribution of the vector source. The number of the remaining lattice points in the codebook depends on the allocated rate;

3. The source normalization. The fast quantization algorithm is used, once the source normalization has been achieved. We can describe this operation as a projection of the input vectors into the truncated lattice volume. A scaling function takes place before the quantization module.

4. The labeling of the lattice codebook points. If the encoding step is simpler with a lattice, the last step of indexing the codewords becomes complicated when the codebook size is large. The index has to be calculated, and the methods aim to realize the best tradeoff between the calculation cost and the conversion table storage. If no training procedure has been required yet, it is ususally used

in order to carry out the entropic index associated with the lattice points.

The inverse quantization procedure consists in decoding the received index, finding the corresponding lattice point and re-normalizing it.

The choice of the metric L_2 (respectively L_1) permits to shape the codebook into an hyper-sphere (respectively an hyper-pyramid) and to count the points. As a result the basic LVQ is only adapted for symmetric and Gaussian (respectively Laplacian) source distributions which map such a codebook. This restrictive modelization of the source offers some accurate and sophisticated methods to achieve the LVQ design, but the well-perform condition collapses considering a complex source coding at low bit rate. With the TSLVQ, based on an embedding lattice strategy, simple procedures are implemented to overcome these drawbacks.[7]

2.6. Tree-Structured LVQ (TSLVQ)

We are going to use a particular LVQ, namely the TSLVQ. So we will describe it.

2.6.1. Hierarchical set of embedded lattices

The goal is to use the hierarchical set of embedded lattices which is achieved such as it is possible to embed a lower scale truncated lattice into a cell of the next higher scale truncated lattice. So the scaling factor between two consecutive lattices of the hierarchy is b (see Fig.7).

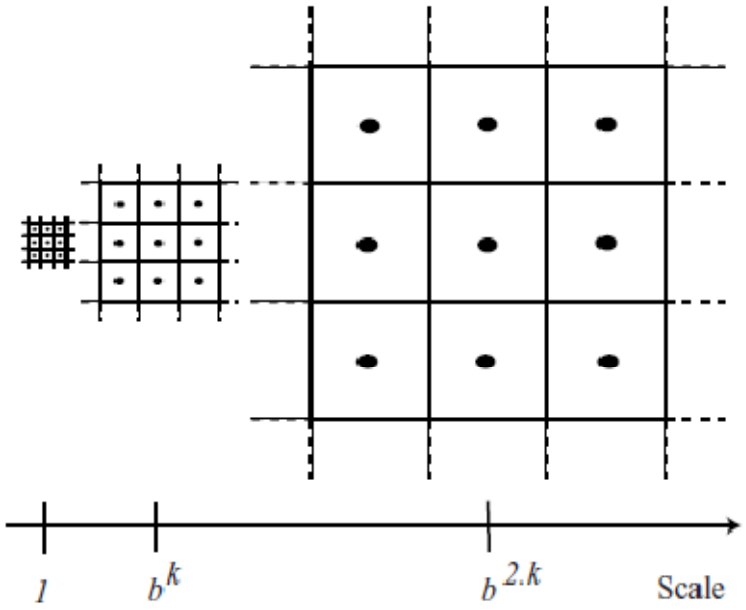


Fig.7 hierarchical set corresponding to the squared lattice (b=3) [7]

2.6.2. Quantization procedure

The principles of the quantization are: [7]

1. a source vector is projected into a first truncated lattice;
2. to get a finer quantization, another lower scale truncated lattice is embedded into the Voronoi cell where lies the input vector;
3. the previous operation can be repeated. Obviously it is more convenient to deal with the input vector scale than to use several lattices with different scales.

The Fig.8 illustrates the resulting multi-stage quantization procedure using successive scaling and translating operators. We have:

·the scaling factor used to project the input vector x into the first truncated lattice:

$$F = \frac{b \times \rho}{L_{2\max}} \tag{11}$$

where $L_{2\max}$ is the maximal L_2 norm of x, this constant energy can be estimated from a training sequence. So all the inputs are projected into a hyper-sphere whose radius equals $b \times \rho$ because:

$$\sum_{i=1}^{k}(F\times x_i)^2 = F^2\times\sum_{i=1}^{k}x_i^2 = (b\times\rho)^2\times\frac{L_2(X)}{L_{2\max}}\le(b\times\rho)^2 \tag{12}$$

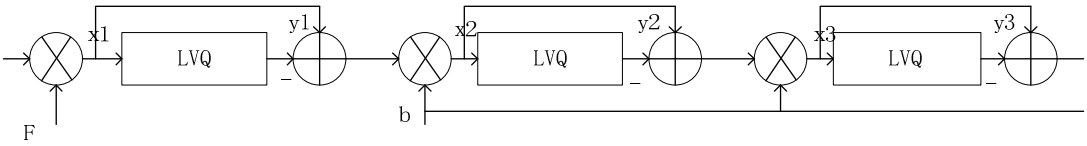
·in the normalized space, the scaling factor used to project each translated vector into the next truncated lattice of the hierarchy: b.

·the reproduction vector of the truncated lattice for the j-th stage: y_j.

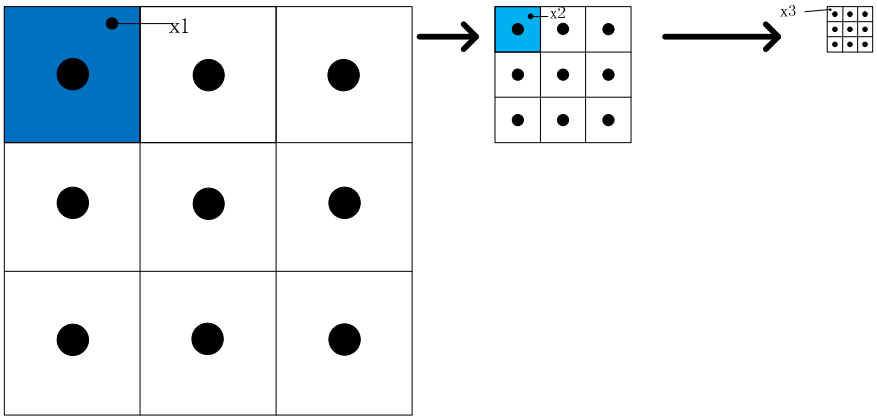
The final value of the reproduction vector associated with x will be:

$$y = \frac{1}{F}\times\sum_j\frac{y_j}{b^{j-1}} \tag{13}$$

where j points out the number of the stage. Note that at each step the same fast quantization algorithm is used.



(a) the principle of the TSLVQ [7]



(b)

Fig.8 Quantization scheme

In Fig.7, we use different lattices with different scales. In Fig.8, for the different stages, we use the same lattice with the same scale, but different input vectors with different scales.

2.7. Tree-structured Codebook

2.7.1. Tree-structured codebook

The codebook has a B-ray tree structure, where B corresponds to the number of points of the basic truncated lattice. Each lattice point (i.e. each reproduction vector) and its Voronoi cell are associated with a tree node. The 0 vector and the whole source space are associated with the root tree. The possible children of a node are the points of the lattice which is embedded into the parent Voronoi cell. A stage in the tree corresponds with a scale in the lattice hierarchy: the deeper is the tree, the finer is the resolution. The final codebook is the set of terminal nodes or leaves.[7]

2.7.2. Unbalanced tree-structured codebook

Two classical strategies have been explored in order to design an unbalanced tree according to a distortion v.s rate tradeoff: a tree pruning, or a tree growing approach. It aims to split the vector space region where the distortion is high, and to avoid simultaneously a prohibitive cost in rate.[7] To our work, we are going to use a tree pruning procedure to design an unbalanced tree-structured codebook which is adapted for the visualization of the molecule data.

3. 3D VISUALIZATION OF MOLECULES USING VQ

3.1. Description of Testing Data

At the beginning of the experiment, we should choose the appropriate testing data, it is described as follows.

3.1.1. General presentation

A molecular property of great interest to chemists is the electronic charge density ρ. MED (molecular electron density), $\rho(r)$, is a real, non-negative and continuous scalar function of the position vector **r.** Further, it can also be determined experimentally using a combination of X-ray and neutron diffraction experiments. MED is also employed as a basic variable in the density functional formalism. It is a usual practice to plot contour maps of $\rho(r)$ for its visual interpretation. However, for quantitative characterization of this scalar field, one has to take recourse to locating and characterizing the respective CP's (critical points). The structure and

reactive sites of a molecule can be well-described by MED and its associated scalar fields such as its Laplacian ($\nabla^2\rho(r)$) and energy-density distribution. Bader et al. pioneered such studies and have successfully employed the topography of $\rho(r)$ for an exhaustive study of molecular structures. Their investigations have revealed that a bond in a molecule is generally represented by a saddle in $\rho(r)$. Other parameters such as bond ellipticity and bond orders which are expressed respectively in terms of eigenvalues and MED values at bond CP are also quite useful in the study of phenomena like charge delocalization and have been applied for a wide variety of aromatic systems. The position of the bond CP has been used as a measure of its polarity. In summary, MED has been widely used for the study of molecular structure. [11]

In fact, the electron density is nowadays regarded as the fundamental observable of the molecular universe.[12][13] We have a example for the hydrogen fluoride (HF) molecule as follows:

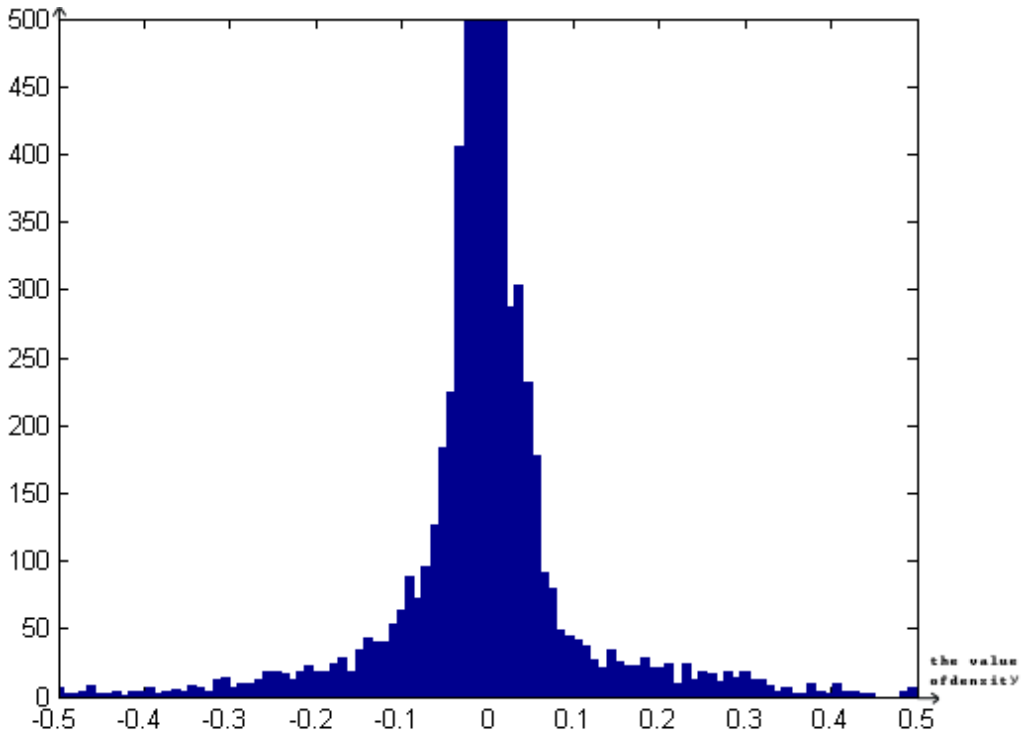


Fig.9 (density distribution of the HF molecule)

3.1.2. How is the data produced

Data is in our case produced by computation. Density functional theory (DFT) was used to solve self-consistently the time-independent Schrödinger equation within the Gaussian quantum chemistry computational package [14]. The calculations require an initial structure that is represented in Cartesian coordinates in an input file which can be constructed with a number of graphical molecular building and visualization tools. Following energy minimization of the input geometry, the electron density is obtained from single point calculations.

The Gaussian computational quantum chemistry package is a popular simulation package due to the fact that it is implementing a wide variety of computational approaches such as semi-empirical, Hartree-Fock (HF), DFT (density function theory), and several correlated post-HF methods. Gaussian can be used to model molecular energies and structures (transition states, reaction products), molecular orbitals, atomic charges and electrostatic potential, vibrational frequencies, and NMR (nuclear magnetic resonance) properties. Calculations can be carried out on systems in the gas phase or in solution, and in their ground state or in an excited state. Gaussian was used to calculate the electron densities of some test molecules.

In the last few years, methods based on Density Functional Theory have gained steadily in popularity [15]. The best DFT methods achieve significantly greater accuracy than Hartree-Fock theory at only a modest increase in computational cost. They do so by including some of the effects of electron correlation much less expensively than traditional correlated methods.

DFT methods compute electron correlation via general functionals, *i.e.* functions of the electron density. DFT functionals partition the electronic energy into several components which are computed separately: kinetic energy, the electron-nuclear interaction, the Coulomb repulsion, and an exchange-correlation term accounting for the electron-electron interaction divided into separate exchange and correlation components.

3.1.3. Data format

The cube file describes volumetric data as well as atom positions, it originates from the Gaussian software package. The file consists of a header which includes the atom information and the size as well as orientation of the volumetric data. This is followed by the volumetric data, one scalar per voxel element. All aspects of the file are text (human readable), originally the numerical values were 5 wide for integers that started each header line (after the first) and floating point values were formatted 12.6, that is, 12 characters wide with 6 decimal places.

The first two lines of the header are comments, they are generally ignored by parsing packages or used as two default labels. The third line has the number of atoms included in the file followed by the position of the origin of the volumetric data. The next three lines give the number of voxels along each axis (x, y, z) followed by the axis vector. Note this means the volume need not be aligned with the coordinate axis, indeed it also means it may be sheared although most volumettric packages won't support that. The length of each vector is the length of the side of the voxel thus allowing non cubic volumes. If the sign of the number of voxels in a dimension is positive then the units are Angstroms, if negative then Bohr. The last section in the header is one line for each atom consisting of 5 numbers, the first is the atom number, second (un-used), the last three are the x,y,z coordinates of the atom center.

Example

In the following example the volumetric data is a 40 by 40 by 40 grid, each voxel is 0.283459 units wide and the volume is aligned with the coordinate axis. There are three atoms.

```
CPMD CUBE FILE [16].
OUTER LOOP: X, MIDDLE LOOP: Y, INNER LOOP: Z
    3    0.000000    0.000000    0.000000
   40    0.283459    0.000000    0.000000
   40    0.000000    0.283459    0.000000
   40    0.000000    0.000000    0.283459
    8    0.000000    5.570575    5.669178    5.593517
```

```
     1      0.000000      5.562867      5.669178      7.428055
     1      0.000000      7.340606      5.669178      5.111259
-0.25568E-04   0.59213E-05   0.81068E-05   0.10868E-04   0.11313E-04   0.35999E-05
       :             :             :             :             :             :
       :             :             :             :             :             :
       :             :             :             :             :             :
         In this case there will be 40 x 40 x 40 floating point values
       :             :             :             :             :             :
       :             :             :             :             :             :
       :             :             :             :             :             :
```

3.2. First Visualization

We take the HF and H_2O molecules for the example.

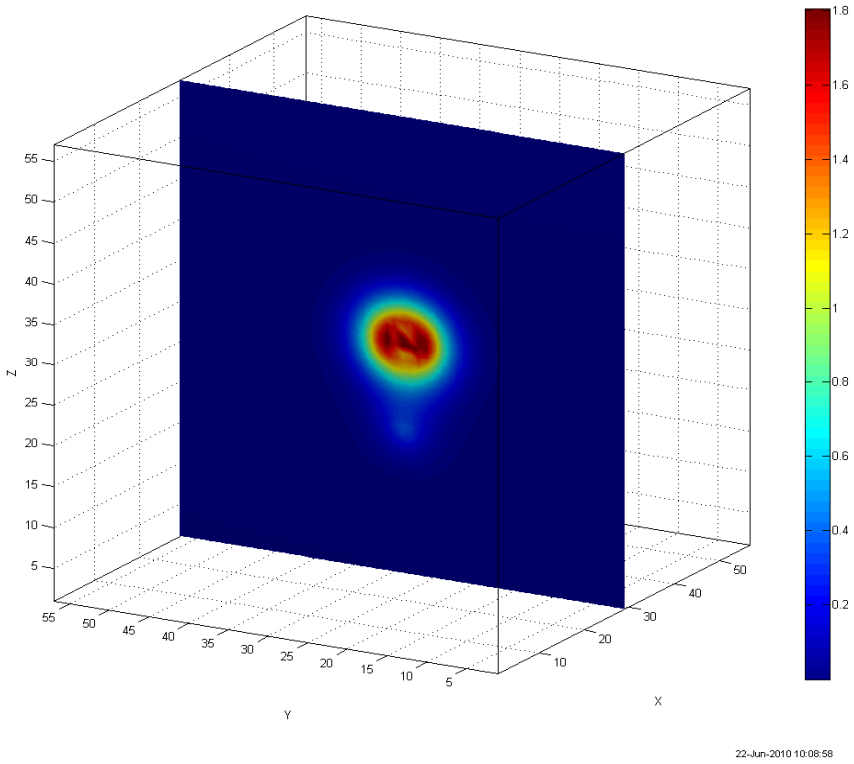


Fig.10.1 (HF molecule)

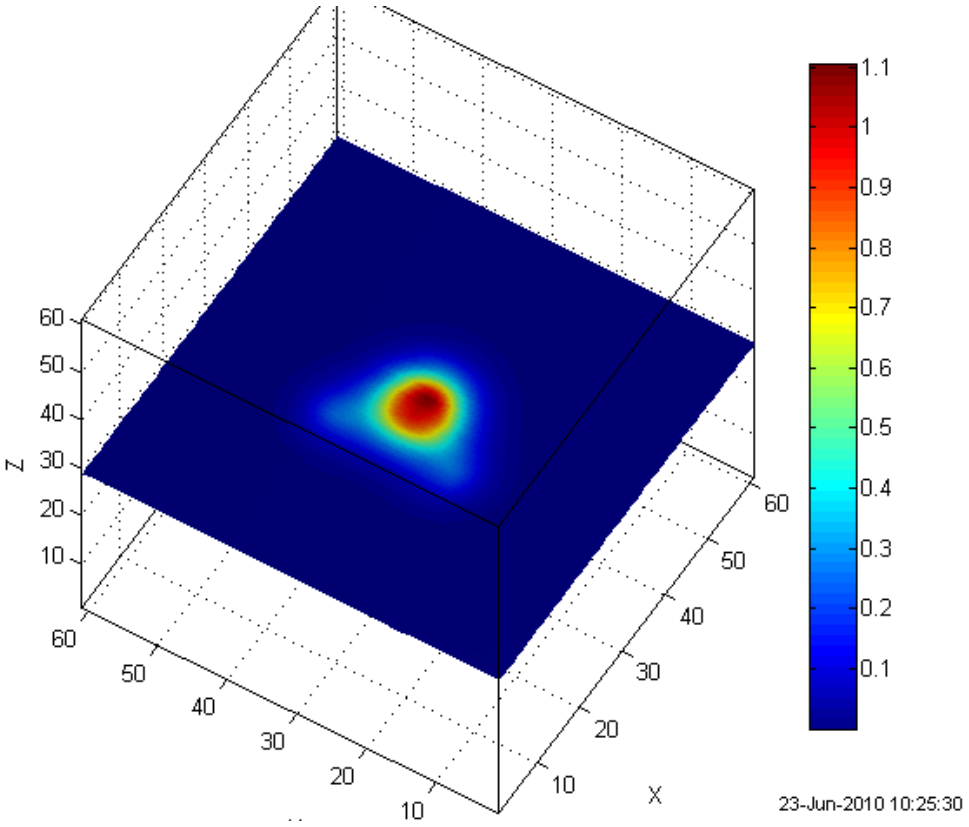


Fig.10.2 (H_2O molecule)

In Fig.10, we can see the shape of the two molecules very clearly. The Fig.10.1 shows a plane at x=29 in the Cartesian coordinates for the HF. Accordingly, the Fig.10.2 displays the plane at z=29 for he H_2O . With different colors, it means the different electron density distribution (EDD). So, for the H_2O molecule, the highest density occurs around the Oxygen (O) atom.

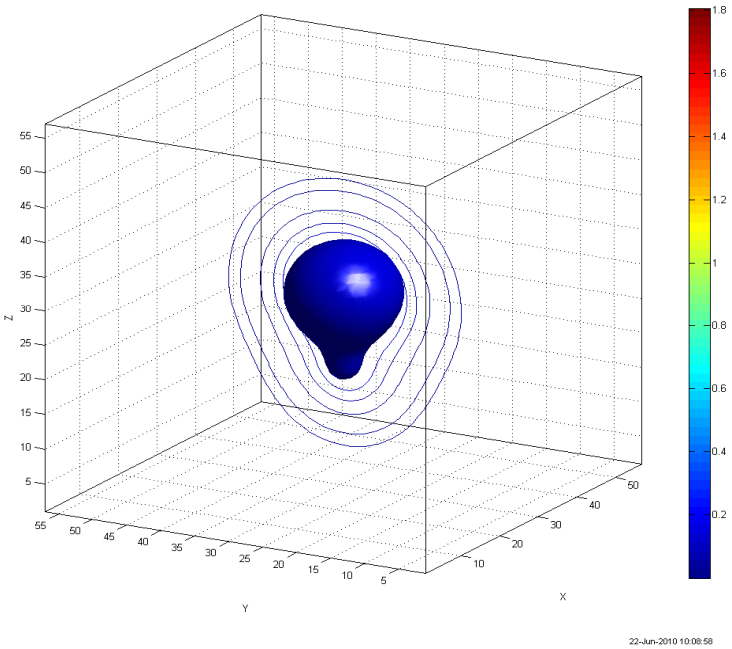


Fig.11.1 (HF molecule)

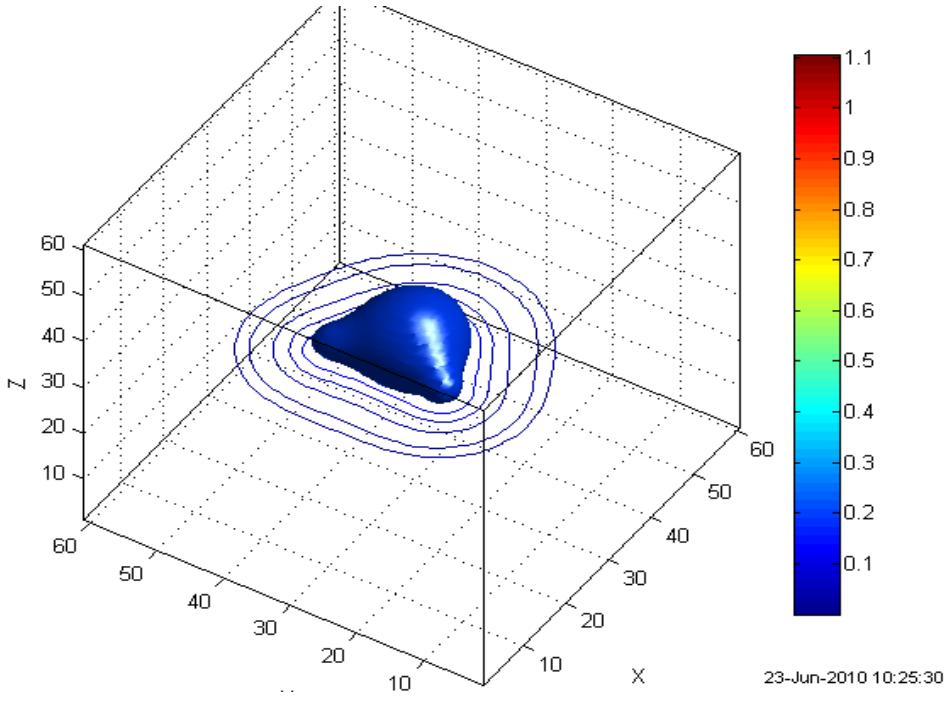


Fig.11.2 H_2O molecule

In Fig.11, we can use the iso-surface (density) plot to display the molecule. Around the molecule, there are some iso-contours which is at [0.1 0.05 0.02 0.005 0.002] with the density equals 0.20.

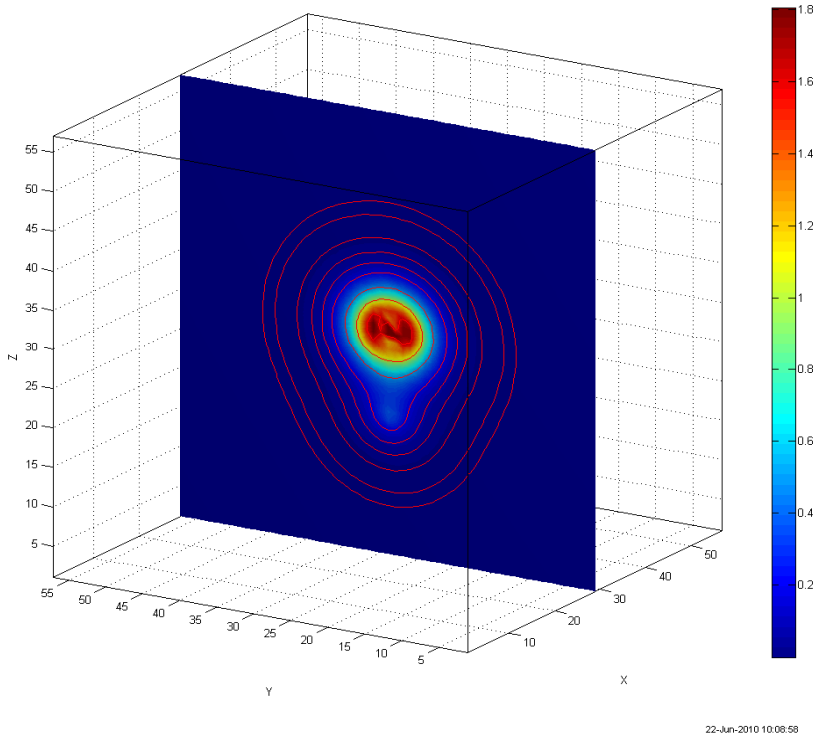


Fig.12.1 HF

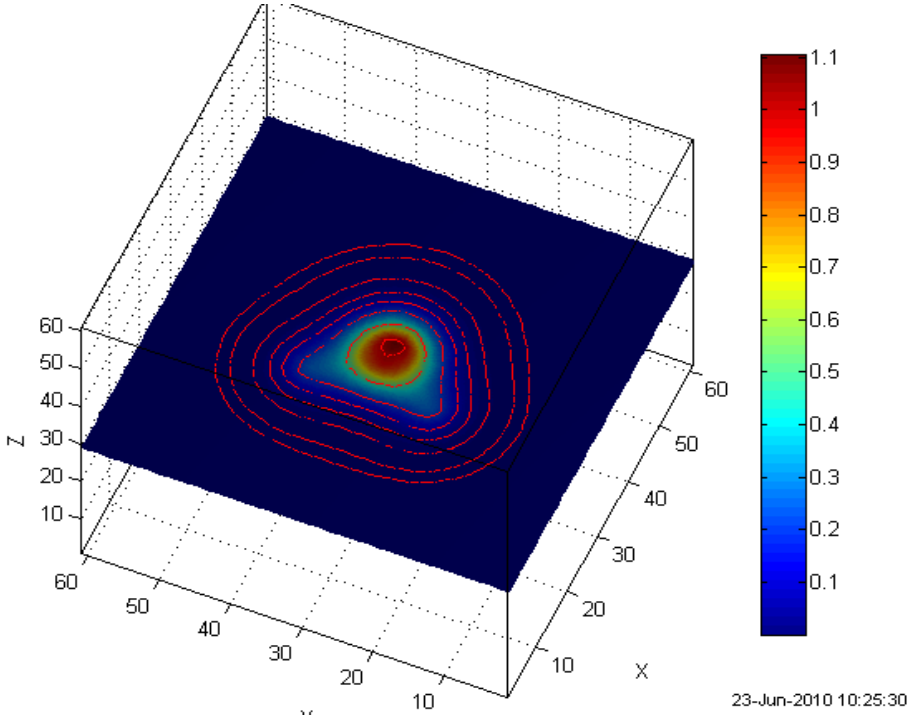


Fig.12.2 H_2O

In Fig.12, we combine the Fig.10 and Fig.11. We create a slice and set the iso-contours equals [1.6 1 0.5 0.2 0.1 0.05 0.02 0.005 0.002].

3.3. The Flow Chart of The Method

In this part, we will introduce what we plan to do.

3.3.1. The flow chart of the method

After preparing for the testing data, we need do some necessary tasks to processing the data. As follows, we give the flow char of the method:

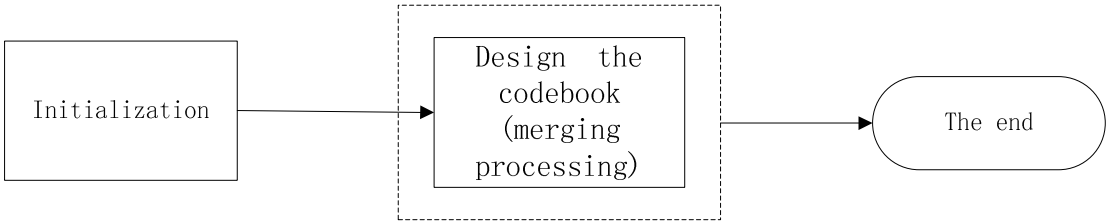


Fig.13 the whole procedure

For example, we load the density of the HF molecule. Therefore, we get a 57*57*57 3D matrix. And then, we should extend the matrix to the 81*81*81, because in the next processing, every 3^{i-1} blocks will be merged into one. So the length of the matrix in each direction must be the multiple of 3. We can give a more clear explanation in the next section.

3.3.2. The chart of the merging processing

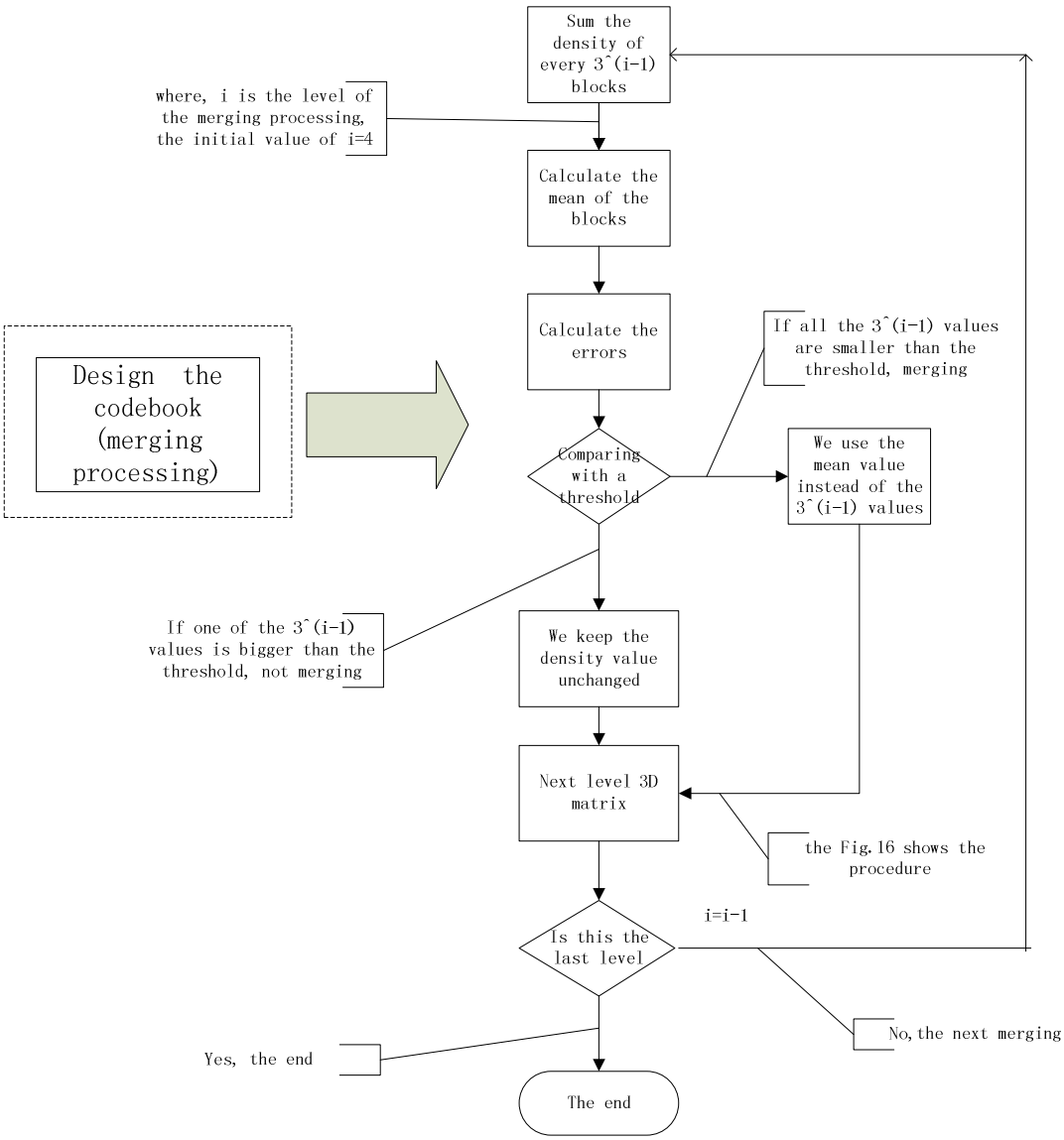


Fig.14 the procedure of merging

The procedure for one loop is as follows:

1. The input is a $3^{i-1} * 3^{i-1} * 3^{i-1}$ matrix, we call it as d_k^A, we sum the every 3^{i-2} blocks, then calculating the average $d_j^B = \frac{\sum d_k^A}{\sum k}$, we can get a $3^{i-2} * 3^{i-2} * 3^{i-2}$ matrix.

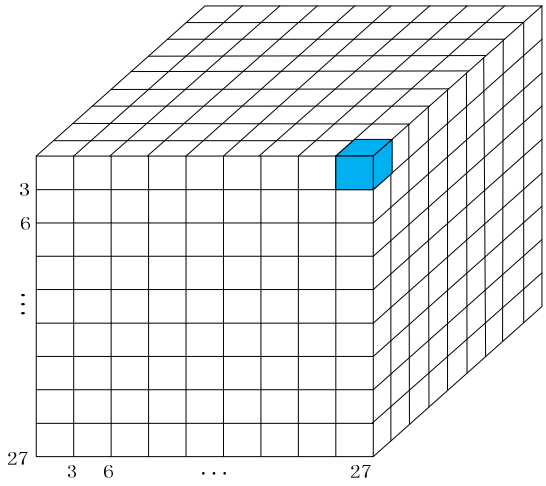


Fig.15 the procedure of summing

2. We calculate the error: $e_k = d_j^B - d_k^A$.

3. If all the values of e_k in one block (see Fig.16) are smaller than the threshold, we will merge the blocks into a big block; on the contrary, if one of the e_k is bigger than the threshold, we will keep the density value unchanged.

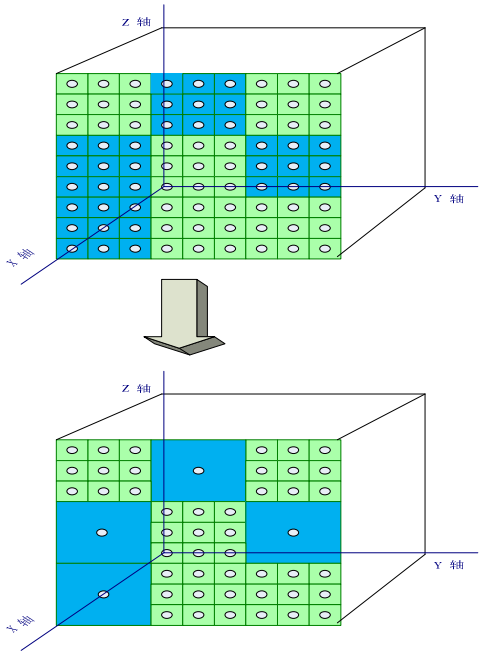


Fig.16 merging or not merging

4. We get a new 3D matrix, and continue the next loop.

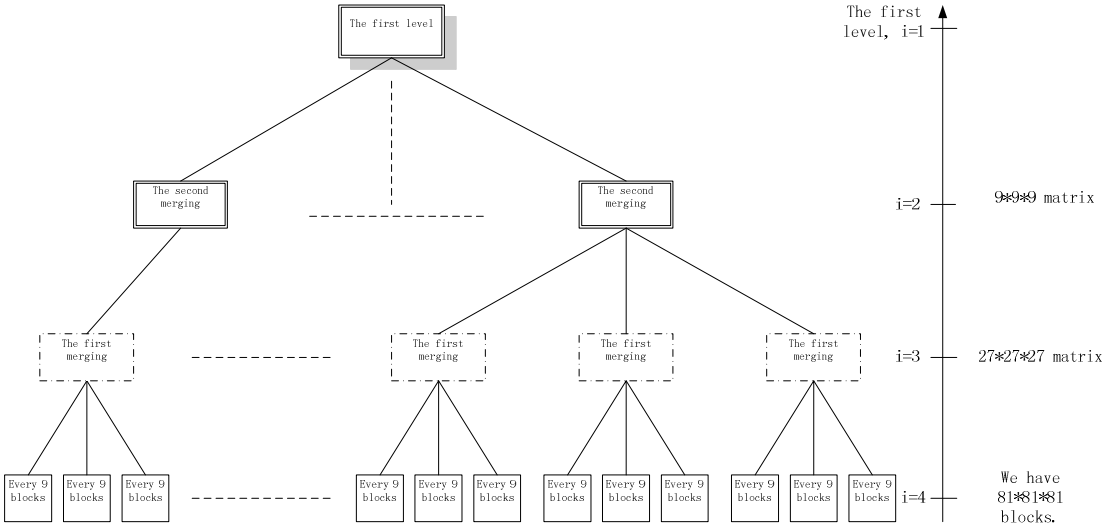
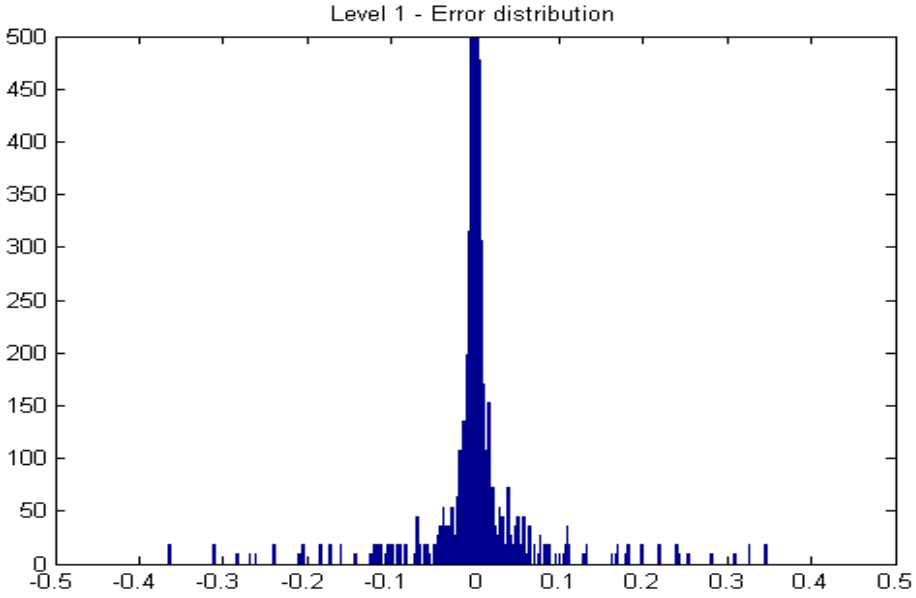


Fig.17 tree structure of merging

4. EXPERIMENTS AND RESULTS

4.1. Set Up For The Merging Algorithm



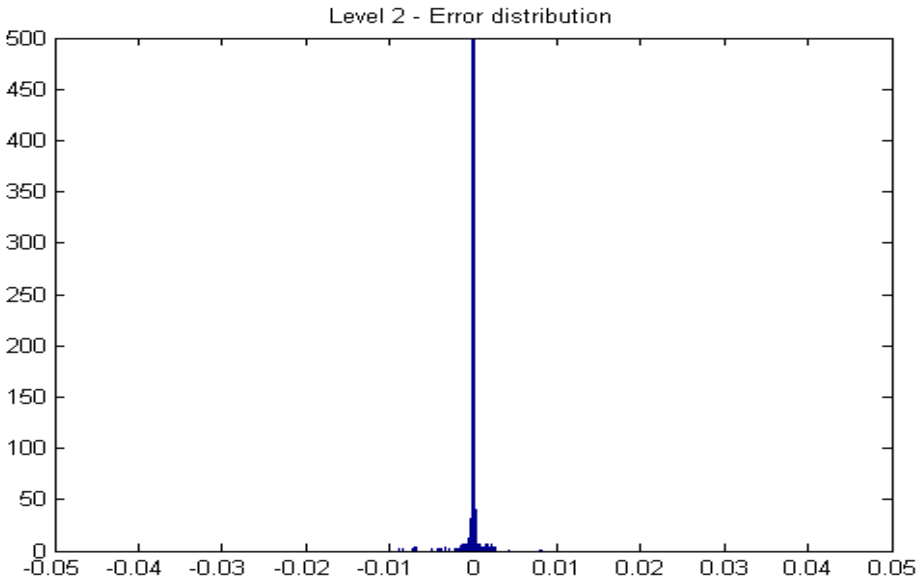
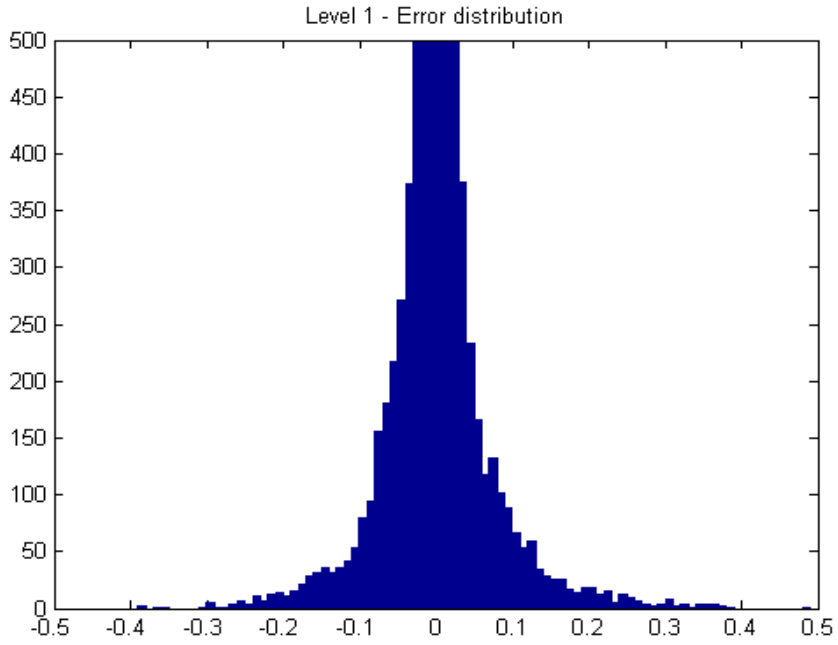


Fig.18.1 error distribution for *HF*

In Fig.18.1, we can see the distribution of e_k. The threshold of the 4th level of the merging processing is 0.1; the threshold of the 3rd level of the merging processing is 0.01. So we can deduce that the threshold of the 2nd level is 0.001.



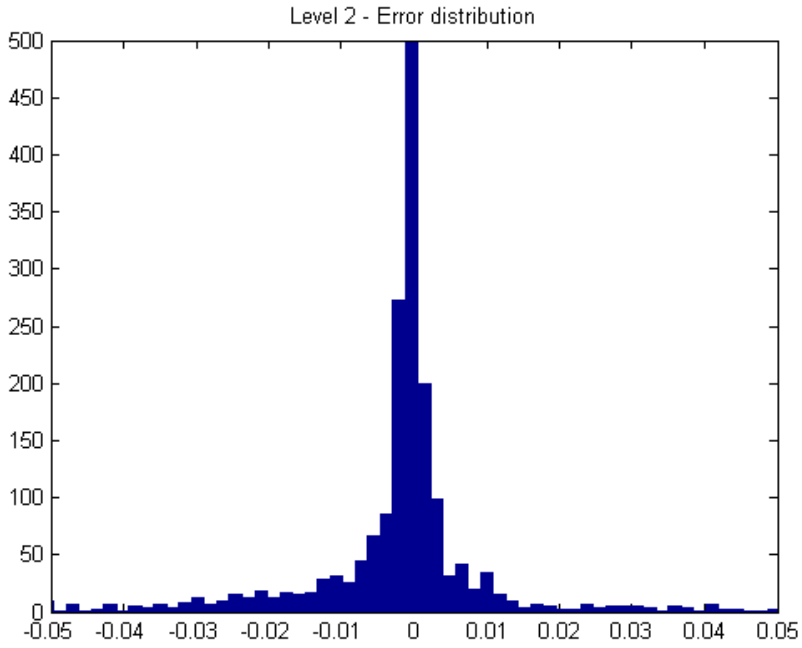


Fig.18.2 error distribution for H_2O

In Fig.18.2, we can see the distribution of e_i' (the error distribution for H_2O). The threshold of the fourth level of the merging processing is 0.2; the threshold of the third level of the merging processing is 0.02. So we can deduce that the threshold of the second level is 0.002.

4.2. The Results

4.2.1. The visualization of the original density

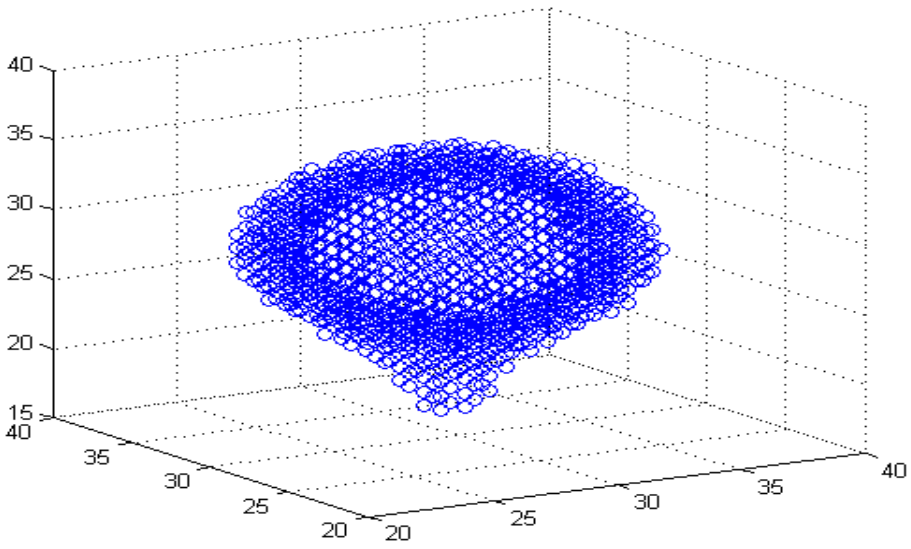


Fig.19.1 (the plot of density distribution of HF)

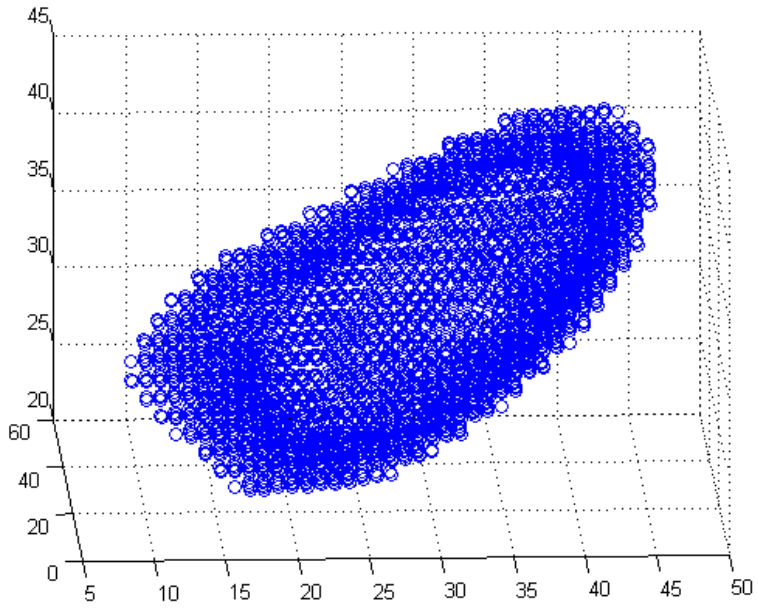


Fig.19.2 (the plot of density distribution of H_2O)

In Fig.18, we can observe that most of the values of density are very small. Therefore, for visualizing the original density, we just visualize the points whose values are higher than a threshold. For the HF molecule, the threshold is 0.35. For the H_2O molecule, the threshold is 0.75. Fig.19.1 and Fig.19.2 are the visualization of the original density.

4.2.2. The visualization of the first and second merging

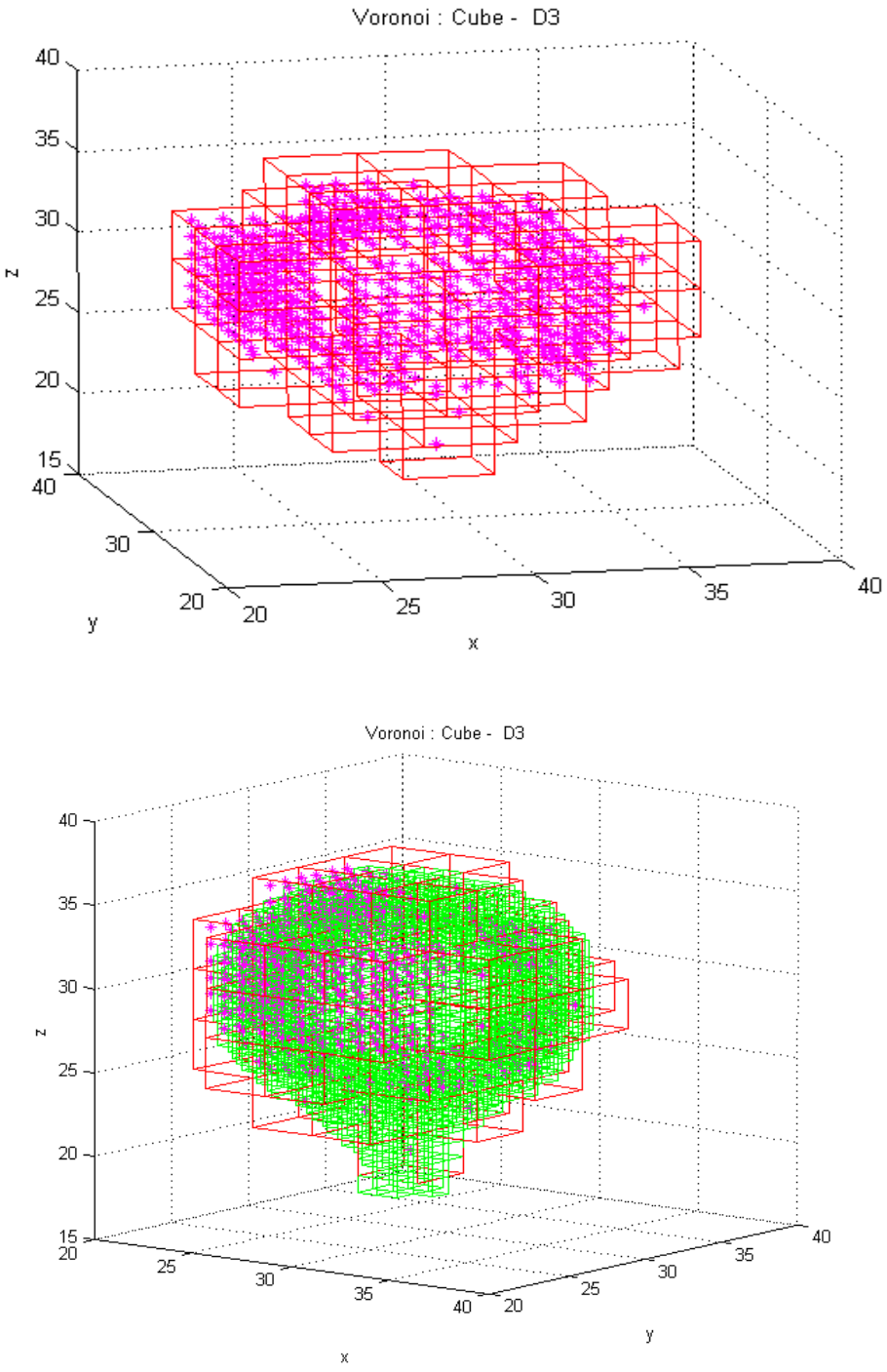


Fig.20 the first and second the merging procedure for *HF*

Fig.20 shows the plot for the merging procedure using the cubic lattice (*HF*).

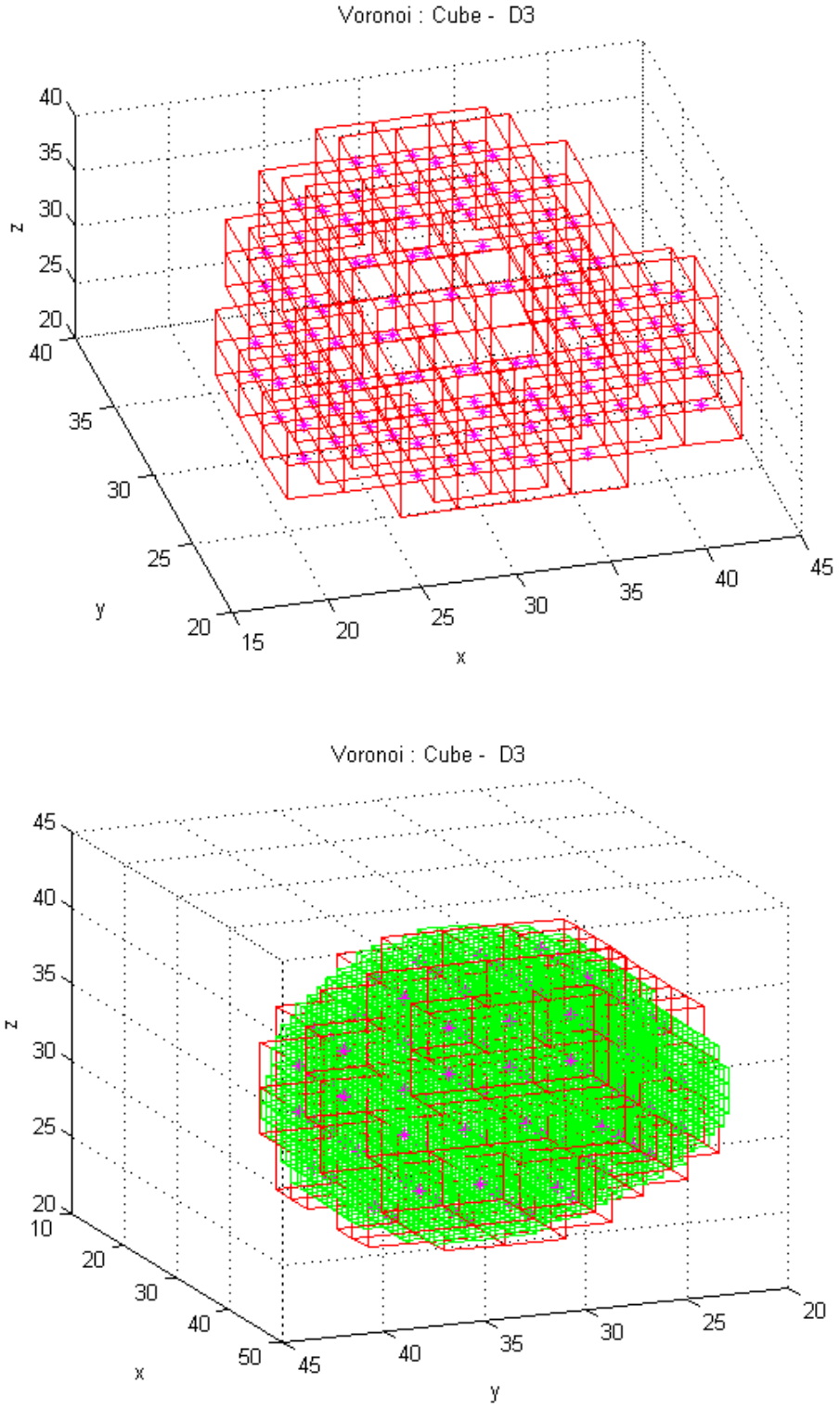


Fig.21 the first and second level of the merging procedure for H_2O

Fig.21 shows the plot for the merging procedure using the cubic lattice (H_2O).

4.2.3. The visualization of the third merging

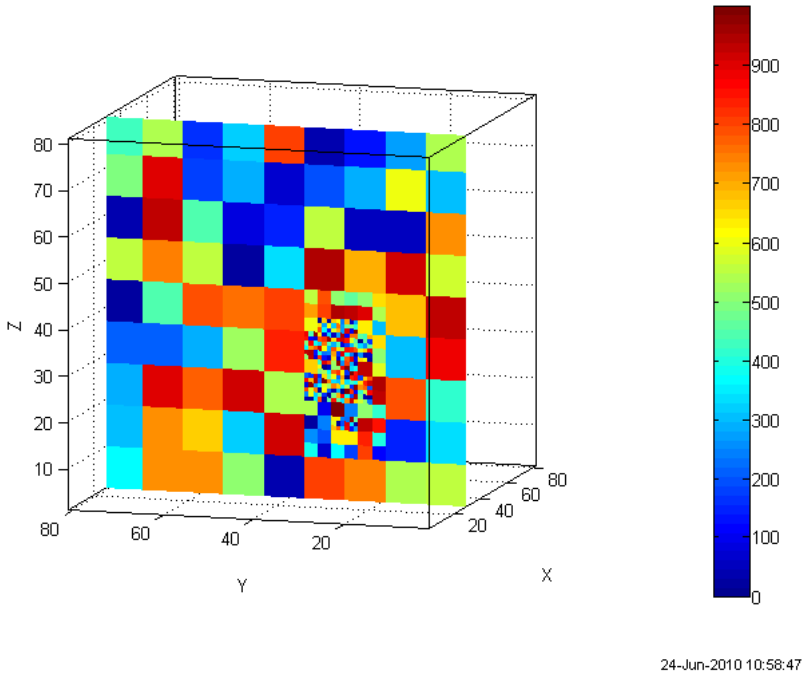


Fig.22.1 (the slice at x =29 of the third merging procedure for HF)

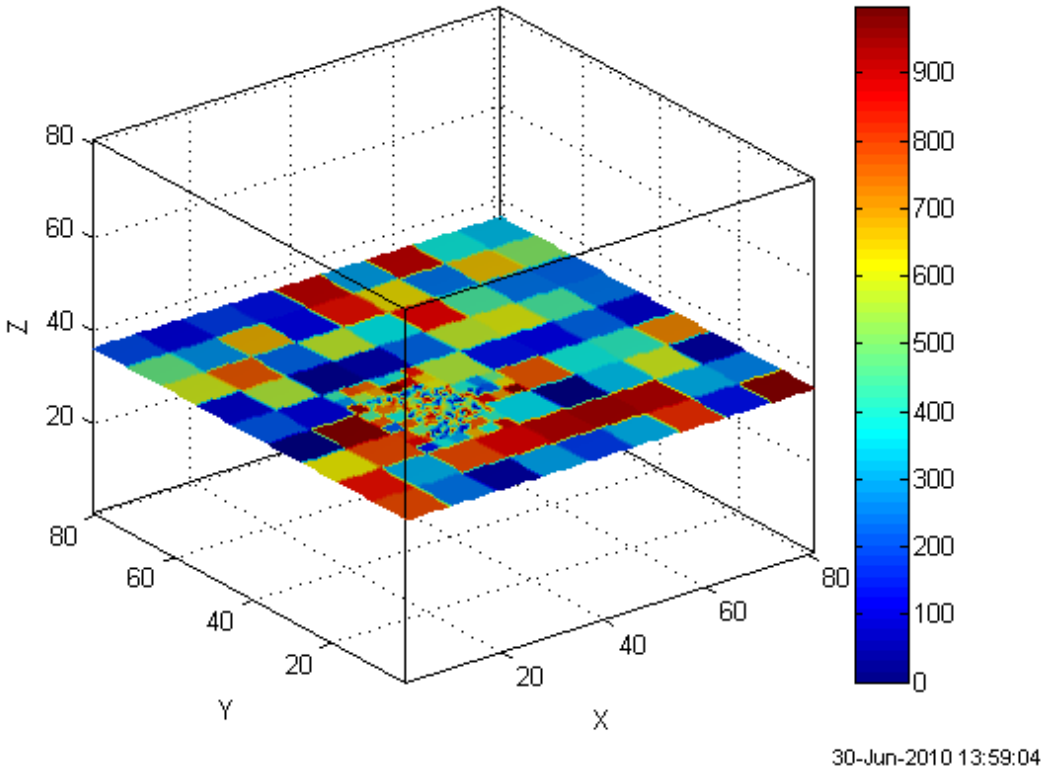


Fig.22.2 (the slice of z =29 of the third merging procedure for H_2O)

In Fig.22, there are three types of cubic. The smallest cubic stands for that the values of the density in the cubic are very high. Accordingly, the values of the density in the biggest cubic are very small.

4.3. Conclusion and Future works

Several experiments were done in this paper. Our contribution mostly focuses on using the LVQ to visualize the molecule using a merging procedure.

There are 2 steps in the experiment. The first step is to get the testing data, and understand the format of the data. Then we should display the distribution of the testing data. In the procedure, we use the HF and H_2O molecules in our experiments and analyze the distribution by using the histograms. The second step is to find a method for merging the matrix. We calculate the mean of the small block, judging merging or not. So in each level we get a bigger block until the procedure stops.

Therefore, it is easy for us to analysis the density distribution of the molecule. And it helps us to find the regularity. So it's a good method to deal with the molecule. Certainly, we can do more work in the future. For example, we can lead a new parameter which describes the distance between the two points that takes into account of the procedure, or improve the criterion of the merging processing.

5. ACKNOWLEDGEMENTS

I greatly appreciate the very efficient assistance of Mr. Vincent Ricordel and Mr. Bogdan Cramariuc in putting together this volume and especially in the tedious task of preparing the work. Thank you for Vincent’s tropical heart and Bogdan’s optimistical mind during the stage.

REFERENCE

[1] Allen Gersho, and Robert M.Gray, “Vector quantization and signal compression”, Kluwer Academic publishers, Boston, 1992

[2] Vincent Ricordel,” Etude d'un schéma de quantification vectorielle algébrique et arborescente. Application à la compression de séquences d'images numériques, Thesis, Université de Rennes1, Laboratoire IRISA, December 1996

[3] T.R. Fisher, “A pyramid vector quantizer," IEEE Transactions on Information Theory, vol. 32, no. 4, pp.568-583, July 1986.

[4] D.G. Jeong and J.D. Gibson,”Lattice vector quantization for image coding," Proc. of International Conference on Acoustics, Speech, and Signal Processing ICASSP, pp. 1743-1746, May 1989.

[5] M. Barlaud, P. Sol_e, T. Gaidon, M. Antonini, and P. Mathieu,“Pyramidal lattice vector quantization for multiscale image coding," IEEE Transactions on Image Processing, vol. 3, no. 4, pp.367-381, July 1994.

[6] Y. Linde, A. Buzo, and R.M. Gray, “An algorithm for vector quantizer design," IEEE Transactions on Communications, vol.28, pp. 84-95, 1980.

[7] Vincent Ricordel, Claude Labit and Moncef Gabbouj,” Tree-Structured Lattice Vector Quantization for Video Coding” IEEE Transactions on Image Processing. 1997

[8] J.H.Conway and N.J.A. Sloane,”Sphere Packings, Lattices and Groups.” pp.1-6,31-33

[9] J.H.Conway and N.J.A. Sloane,”Sphere Packings, Lattices and Groups.” pp.84-118

[10] J.H.Conway and N.J.A. Sloane,”Sphere Packings, Lattices and Groups.” pp.443-448

[11] J.S.Murray and K.Sen,”Molecular Electorstatic Potentials Concepts and Applications” pp.228

[12] Kohn W, Becke AD, Parr RG,” Density functional theory of electronic structure.”J Phys Chem 1996;100:12974–12980.

[13] Kohn W. Nobel lecture: Electronic structure of matter-wave functions and density functionals. Rev Mod Phys 1999;71:1253–1266.

[14] Gaussian 09, Revision A.1, Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.& al. Gaussian, Inc., Wallingford CT, 2009.

[15] Wolfram Koch, Max C. Holthausen, A Chemist’s Guide to Density Functional Theory, 2nd Ed, 2001 Wiley-VCH Verlag GmbH.

[16] http://www.citst.ro/forum/viewforum.php?f=26