Energy Consumption Improvement through a Visualization Software

Benoit Lange, Nancy Rodriguez, William Puech

▶ To cite this version:

Benoit Lange, Nancy Rodriguez, William Puech. Energy Consumption Improvement through a Visualization Software. Zoran Morvaj. Energy Efficiency - A Bridge to Low Carbon Economy, INTECH, pp.161-184, 2012, 978-953-51-0340-0. 10.5772/38794 . lirmm-00808104

HAL Id: lirmm-00808104
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00808104

Submitted on 4 Apr 2013

Energy Consumption Improvement Through a Visualization Software

Benoit Lange, Nancy Rodriguez and William Puech
LIRMM Laboratory, UMR 5506 CNRS,
University of Montpellier II,
France

1. Introduction

In the actual world frame, energy efficiency becomes a necessity. Since 1995, Kyoto protocol has highlighted that humans need to improve their energy consumption and reduce CO_2 signature. Building consumption represent a third of the global energy consumption and Information Technology (IT) equipment is weighing heavily on energy expenses. For example, Google equipment consumes 0,01 percent of the world global Energy, the equivalent of the power usage of a city with 200.000 inhabitants.

Solutions to reduce usage of fossil energy, called "green energies" and supported by standard industries, already exist. For example, faulty processors are removed from production and are re-factored to create solar sensors.The PUE, Power Usage Effectiveness is also a measure created to monitor energy in data centers.

As IT equipment, buildings become greener and smarter. Indeed, buildings are gap of energy and it is necessary to improve them to consume less energy. Governments establish buildings certification to help this development. For example, in France, High Quality Environmental standard has become the actual norm for new or re-factored buildings. Buildings Management Systems (BMS) are widely used to catch data from a set of building sensors. BMS are only reactive systems, there is no prediction. Therefore, it is necessary to propose and develop smarter systems, fully focused in energy improvement.

In this chapter, we present a project called RIDER, for Research for IT Driver EneRgy efficiencies. We are developing a green box composed by a set of IT components; able to generate efficient predictive model and building optimization. The main part of the green box is a pivot model managed by a rule engine and a standard BMS with a specific input/output protocol. Modules can extend the standard green box offering system/user advices, proposing improvements on data model and predicting building behavior.

This chapter introduces the RIDER green box and details the visualization module, which is used to improve model by using user knowledge.

In Section 2, we present the related work on visualization. Section 3 presents a global overview of the RIDER project. Section 4 is dedicated to the visualization method used in this project, whereas Section 5 shows experimental results on real data. Conclusions are presented in Section 6.

2. Related works

Data centers are present in most companies. As they represent a gap of energy, data centers are suitable to be a starting point for energy efficiency research. We work with data extracted from sensors of a dedicated data center located in IBM, Montpellier (France). IBM Montpellier host a specific kind of data center, called "green" (Green Data Center). Effectiveness of a Data center energy is measured with PUE (Power Usage Effectiveness), most of data centers have a high PUE: if value is bigger than two, i.e. a data center consumes twice as much energy as is needed. A data center is told green when its PUE measure is less than 1.5. GDC improvements are developed around management of alleys or usage of a specific kind of server cases. Green Data Center in Montpellier is one of the pilot sites of the RIDER project.

In this chapter, we mainly focus on visualization methods to improve the RIDER model. This section is dedicated to previous visualization methods, level of details and visual analytics domain.

2.1 Visualization

Huge amount of data is produced each day, and visualization is a way to understand this mass. Physics, mechanics, literature and many other domains benefit from visualization.

In physics, scientists use visualization to interpret, understand, render their data, this paradigm is better than spreadsheet. Scientific data contains many dimensions provided by satellites, telescopes, simulations and so on. Scientist's main problem is to give a sense to data, an answer can be given by visualization. Particles are small objects without weight. It is possible to classify particles in three different kinds: static, pre computed or dynamic.

Pre computed particles was used for render explosions in 1960. At each iteration a mathematical function computes location of particles.

Dynamic particles interact with their environment and between them. This method was largely used in fluid simulation, Green (2007) presented results of GPGPU computing [1]: this method use graphic devices as computational device and is based on particles for dynamic fluid simulation. Results are impressive and flow is well rendered. The last kind of particles system is a large static point cloud, they are used to render informations through their colors; cost is low compared to most visualization methods.

Views have to deal with rendering of multivariate and complex data, thus scientists use HPC (High Performance Computing [2]) systems to analyze and produce views from data. These kinds of large computers are gap of energy efficiency and computation time is expensive.

Kapferer & Riser (2008) proposed a method to visualize a large amount of spatial data (clusters of stars) without using expensive HPC. They used a volume rendering method based on particles to render data. They developed their own solution to produce rendering because existent solutions were not efficient. Instead of using traditional HPC method, they used

[1] General-Purpose computation on Graphics Processing Units
[2] Large computer designed to manage large data flow

GPU computing to compute a real time view with a low cost hardware. Authors used GPU computing to apply simulation on data. To display stars, they used a simple point sprite instead of a sphere (spheres are expensive objects in computer graphics). To improve rendering of point cloud, they used lights to create a spherical aspect to these primitives. This solution allows to display more data than full volume representation method. Usage of GPU for astrological data and particle paradigm seems to be well-suited.

Ilcìk (2008) proposed another example of visualization software. In this paper, he analyzed behavior of CAD model within fluid simulation. Particles are used to find weak points of 3D models. When a collision between a particle and fluid occurs, the value of particles change. This method is efficient to analyze CAD models(Indeed, these meshes are well defined and the number of triangles is important). Particles give opportunity to not saturate the object and are an efficient method to understand quickly a problem.

2.2 Level of details

When we focus on 3D visualization, an important question is to fit with hardware's capacities. Current graphic devices are able to render an important quantity of polygons but the number is limited. To solve this problem multi-resolution of meshes appeared. Clark (1976) introduced the concept of level of detail. 3D objects are composed of thousands of polygons. Because a scene is composed of a set of 3D objects, the number of polygons increases quickly. Level of details consists in producing meshes at different resolutions. These different meshes are loaded on the 3D scene depending on camera position. Automatic methods arose in early 90', they were based on usage of different kinds of operators. The first methods were brut forces meshes decimation (Schroeder et al. (1992)). This method use a decimation of vertices as operator but it is not efficient, retriangulation creates some topology mistakes. Luebke (1997; 2001) proposed two overviews of several operators.

But, most of these simplification technics are suitable for triangular meshes and not for volume model. He et al. (1995) proposed a method based on voxel simplification by using a grid for clustering voxels. Lorensen & Cline (1987) have developed marching cube algorithm, used to produce a surface mesh. But this simplification algorithm does not preserve the shape of the mesh. In the work presented in Section 3 and 4, we are looking for a point cloud simplification. Indeed, previous methods dealing with simplification of surface point cloud like Moenning & Dodgson (2004); Pauly et al. (2002); Song & Feng (2009) are not adapted to our case, surface objects do not render inside part of objects.

2.3 Visual analytics

This chapter is focused on visual analytics paradigm proposed by Shneiderman (2001). The goal is to combine two domains: Knowledge Discovery (KD) and Information Visualization (IV). Targets of these domains are similar: extract content from a set of data. Each of these domains uses its own method to reach this goal. KD uses neural network or decisions trees to analyze, simplify and extract interesting features from information. IV is a paradigm where human is the center of the system. Human is used to dig into data instead of an algorithm. Usage of human expertise for digging into data is underlined by Rosen & Popescu (2011). This paper presents an efficient way to count items in a maze; the most important part is to use a smart camera to explore several corridors in the same view. IV and KD have some limits, they can introduce mistakes or misunderstanding on data. Schneiderman had proposed merging of these approaches. Later, Keim et al. (2006) presented a formal definition of this paradigm. In Visual Analytics (VA), automatic methods are used to sort and produce a visualization for user. Then, user can manage data through the visualization and can expose sorting or new content. Often, this data carry different kinds of values, this means that each object can be defined by a set of values: for example, sensor is located by: X, Y and Z coordinates. Visualization tools need to be able to deal with multivariate data, and moreover, it is necessary to give to the user the best method to dig into data: human knowledge can find some unexpected information. To improve its efficiency, user interface needs to be developed around a large set of tools: zooming, panning, scrolling, etc. and also a large set of visualization methods. There are many kinds of visualization to help user's understanding: parallel coordinates, scatter plot, glyph, etc.

Many solutions developed around VA paradigm have emerged from scientific software. Liu et al. (2010) developed Netclinic: a software to find culprit on network. Failure of networks is critical and it is important to find the reason of problems. Netclinic is based on an automatic approach to extract well-known causes of trouble and then software renders view of data designed for non expert users. Human proposals can be analyzed and validated by an automatic method.

Another example of VA paradigm application was proposed by Migut & Worring (2010). Authors explored a method to evaluate risks of a wrong decision from a set of person. They used a machine learning method to extract standard profile of dangerous people. Decision made can be critic and it was necessary to remove false positive. A semantic visualization was given to user to resolve a part of this trouble. The solution in this paper provides a visualization method to understand results from a classifier output using a simple multidimensional space visualization. The software here is composed by a visualization data mining method and a machine learning method. Experts can drive the visualization with a 3D view, based on a scatterplot. This kind of view is presented by Konyha et al. (2009) and a support vector machine provides data filtering. Authors used Voronoï cells to extract borderline of each plot of scatterplot. To improve visualization, pictures and colors were added. Several interaction paradigms (zooming, panning and selection) were implemented to help exploration of data in an efficient way. User case study has been established on Forensic Psychiatry. Experts were able to identify if a patient share same characteristics with a previous patient.

Another data visualization software for multidimensional data is Caleydo. Lex et al. (2010) gave an overview of this software to refine medical diagnostic. Gene exploration is a hard task due to the large amount of multidimensional data. It is not trivial to present a coherent visualization of gene relation. In literature, this problem was apprehended by virtual reality methods. Authors wanted to develop a standard computer method using multiple views and a 2.5D view. The goal was to visualize genes, relations were displayed through the bucket and linked views. Authors used parallel coordinates (Hauser et al. (2003)) or heat map for the visualization of genes. This application was designed around several features: bucket, zooming and links. Their proposal is mainly concentrated on the bucket: it is a special kind of arrangement for multiple views. Each view is mapped to a face of a cube. With this method, links between views can be understood efficiently and interactions give a method to follow the gene path.

3. RIDER project

The RIDER project is a collaborative research project led by IBM France. A consortium around several French industries and universities is in charge of development and research of this project. The goal is to produce a platform, called **green box**, designed to improve energy consumption of a building based on recommend. These are given to monitor, analyze or predict events (temperature increase, crowd, etc). This green box has to be scalable to fit with different building topologies (fit on different kind of buildings: service, industrial or residential buildings).

Buildings become smarter each year. Sets of sensors are used to help monitoring buildings and results are exploited by the building manager. Unfortunately, these results are not easy to understand and they produce huge amount of data. Depending from building, granularity of sensor meshes is different, for example, we can find sensors for: CO_2, temperature, pressure, etc. All these sensors are placed to learn building life, ubiquitous computing is more and more used to help human decisions. But this fast growing brings a large flow of data and actual system does not use all of them. It is necessary to find an efficient way to manage and understand these data. In a second way, it is also necessary to deal with building granularity of sensors. Buildings have not the same number of sensors, some rooms are fully equipped in high resolution, while other rooms are only equipped by a small number of sensors.

This data is managed by a Building Management System (BMS), which only provide analyzes of the current situation and not needs of future. For example, if the temperature increases in a room, the BMS will adapt air conditioner speed depending on the user desire. To improve energy in building, it is necessary to deal with building inertia and adapt air conditioning to the different temperature (external and neighborhood temperature). Improvements on this part are important, because they allow managing building energy in an efficient way. Our green box is designed to understand, anticipate and react to any situation with specific adapted predictive models. This box is composed by a set of tools: a pivot model and recommend modules. Building's model contains by sensors, building's data and external data (from different sources: weather, personal calendar, etc ...).

The global architecture of our system is developed in Figure-1. The heart of our green box is developed around a model driven methodology and a software product line approach for the deployment of "smart buildings" control software. For this purpose, new ways to model and optimize physical infrastructures from an energy-efficiency point-of-view are explored. Also new ways to model software product lines and their variability artifacts are investigated.
The global objective of modeling is to capitalize in a dedicated software product line the architectural knowledge. It aims to study generic model and customizable architecture for a software product line. The idea is to develop modeling tools that will establish new ways to instrument, interconnect and optimize neighborhood or buildings. This method is also developed to facilitate the link between the physical and the computer world. The model will enable creation of real-world aware systems in a context of event-driven computing. New UML models will describe the physical architecture with a higher semantic level to handle the energetic efficiency domain.

A first component is developed around decision making. It aims to build weak dependency solution. This method was developed around learning technics to construct physical and preferences models. The construction of these informations are generated from interviews of final users.

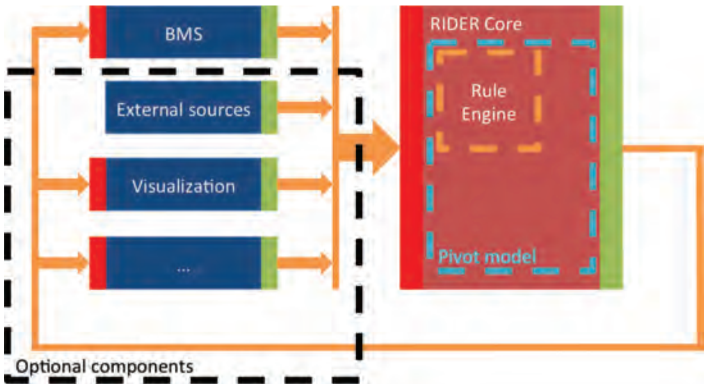


Fig. 1. RIDER architecture.

The main goal of the building manager model is to develop a grey box model for the control and supervision of exchanges and uses of energies between different buildings, including classic and renewable energies, through the use of IT. Energetic problems such as energetic losses will be dealt with information technology in order to bring smarter solutions.

A data mining component has goal to dig into data. It is based on statistical model provided by statistic engines. Informations are extracted from this engine using sensors data. Automatic data mining is done with this component, this allows to validate advices given by the different components.

A visual engine component is also developed. The goal is to propose a mining tool using user knowledge. User is able to feed system when an automatic method fails.

Entire system runs as follow: BMS records data from sensors located in building, then it sends information through a TCP connection with a XML protocol to RIDER Core, this input is the same for each component. RIDER core instantiates the pivot model and applies dedicated rules. Finally it sends results to all subscribers (each component can become a subscriber). If a subscriber wants to propose new data, it can use the same process as the BMS update.

4. Visualization method

In the context of this chapter, interest is on visualization and advices propose to global system. Most of visualization software is developed to render data and not to propose content. Our solution is developed to provide a understandable visualization and also a way to create content. We will present an architecture based on client/server paradigm, different kinds of visualizations used and a digging tool.

4.1 Architecture

Visualization software is based on a client / server paradigm. An HPC (High Performance Computing) is used to produce data, whereas another HPC is used to render this data.

HPC has become a major actor in scientific visualization, unfortunately this kind of computer is expensive and is a gap of energy. HPC are large computers developed for parallel computing, thus it is necessary to use specific algorithms. Instead of using large computer, it is also possible to use an architecture based on graphical engine to execute algorithms. It consists in using a graphic card to compute algorithms in parallel. This section introduces our algorithm used for rendering, and our proposition of a benchmark based on energy consumption of different kinds of devices (a standard computing, parallel computing, ...).

The base of our data is a set of sensors and a coarse building topology. Sensors type, location and value are well known. With all these informations, we want to produce a volumetric view for each room of a building. To produce 3D volume object, voxels and point clouds are the most common methods. Voxels are volumetric pixels, they are also primitives of 3D volume objects. We have adapted the second solution based on point cloud, this method is mostly used in 2D through scatter plot visualization. Piringer et al. (2004) presented a 3D scatter plot method. Primitive used for rendering is a low cost primitive, the goal is to reduce rendering cost (this part will be explained later). This point cloud can be considered as a particle set, as stated previously, particles used in this work are static.

For the generation of the point cloud, we have developed two methods: a regular grid and a concentric pattern. As stated before, our input is a set of sensors and the room's characteristics. Rooms are designed by three measures: length ($l \in \mathbb{R}^+$), width ($w \in \mathbb{R}^+$) and height ($h \in \mathbb{R}^+$). Figure-2 gives a schema of a room with layered sensors. Sensors are located at determinate height in space. Sensors set is defined by: $S = \{s_1, ..., s_{nbS}\}$, where nbS is the number of sensors and $nbS \in \mathbb{N}$. Each sensor is defined by three coordinates: $s_i = \{x_i, y_i, z_i\}$, $i \leq nbS$, where $0 \leq x_i \leq l$, $0 \leq y_i \leq w$, $0 \leq z_i \leq h$. Particles are defined by: $P = \{p_1, ..., p_{nbP}\}$, where nbP is the

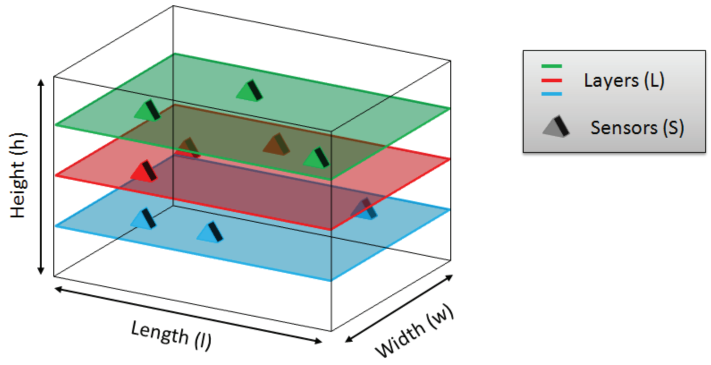


Fig. 2. Topology information of rooms.

number of particles and $nbP \in \mathbb{N}$. Each particle own its coordinates: $p_k = \{x_k, y_k, z_k\}$, $k \leq nbP$, where $0 \leq x_k \leq l$, $0 \leq y_k \leq w$, $0 \leq z_k \leq h$.

For the regular grid, coordinates of each particle can be calculated with:

$$p_k = \begin{cases} x_k = \rho + b \cdot \rho, \\ y_k = \rho + c \cdot \rho, \\ z_k = \rho + d \cdot \rho \end{cases} \tag{1}$$

where b $\in \mathbb{N}$, $0 \leq$ b $< \frac{l}{\rho}$, c $\in \mathbb{N}$, $0 \leq$ c $< \frac{w}{\rho}$, d $\in \mathbb{N}$, $0 \leq$ d $< \frac{h}{\rho}$. ρ represents space between two particles and is defined by depending from resolution desired. We use this measure to produce a multi-resolution mesh designed for different devices. If we want a coarse resolution, we use a high value for ρ, at opposite, we use a low value for ρ to get a high resolution. Each device can visualize a limited number of particles before loosing real time visualization. We can calculate the number of particles for the regular grid by the formula:

$$nbP = \frac{((l-\rho) \times (h-\rho) \times (w-\rho))}{\rho^3}. \tag{2}$$

This grid is regular and fills the entire structure of the room. Figure-3(a) presents a regular grid in a room. Multi resolution can be used to provide the right visualization for each device, simple computer cannot render the same number of particles as a large screen display, see Figure-3(b).

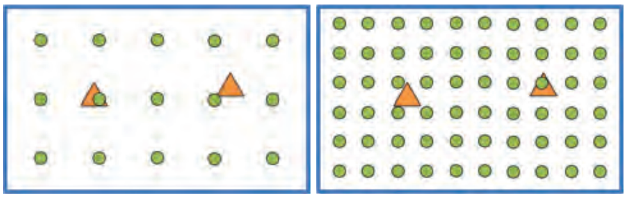

(a) Coarse resolution of a regular grid model.
(b) Higher definition of a regular grid model.

Fig. 3. Regular grid model used to render volume of a room. Dots represent particles, triangles represent sensors.

Another possible construction method for particles grid is a concentric pattern. Particles are dropped in rooms with a concentric sphere centered on sensors. Coordinates of these particles can be computed using the following method:

$$p_k = \begin{cases} x_k = S_{xi} + r \cdot \sin\theta \cdot \cos\phi, \\ y_k = S_{yi} + r \cdot \sin\theta \cdot \sin\phi, \\ z_k = S_{li} + r \cdot \cos\theta \end{cases} \tag{3}$$

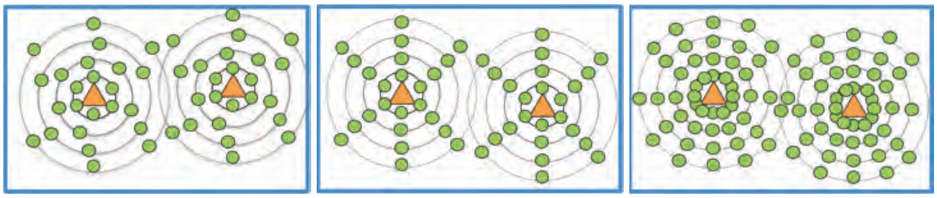
with

$$\begin{cases} r = k \cdot \rho, & k \in \mathbb{N}, k \leq l, \\ \theta = k \cdot \rho\theta, & k \in \mathbb{R}^+, k \leq w \text{ and } 0 \leq \theta \leq 2\pi, \\ \phi = k, & k \in \mathbb{R}^+, k \leq h \text{ and } 0 \leq \phi \leq \pi, \\ \rho \in \mathbb{R}^+ \end{cases} \tag{4}$$

The number of particles can be estimated using the following formula:

$$\begin{aligned} nbP &\leq nbX \times nbY \times nbZ, \\ &\leq l^3 \times w^3 \times h^2. \end{aligned}$$

To produce a desired point cloud, it is also possible to define the desired number of particles, and algorithm solves this problem. Figures-4 illustrates concentric repartition.



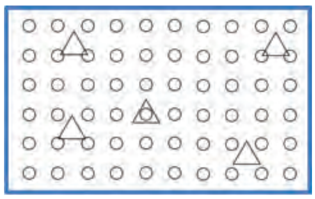
(a) Wiggle model. (b) Coarse model of concentric representation (angle is the same for each particle). (c) Higher model of concentric representation (angle is the same for each particle).

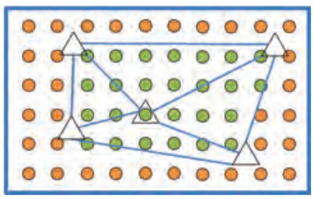
Fig. 4. Different concentric methods used. Dots represent particles, triangles represent sensors.

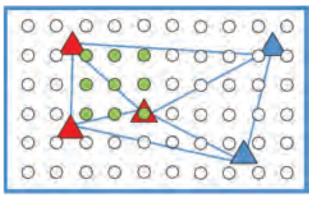
These two kinds of volumetric rendering are suitable for specific visualization. Moreover, particles need to be weighted depending on sensor location. For this, we use two different methods to compute location of particles: a Delaunay triangulation and Voronoï cells. These two methods were presented byAvis & Bhattacharya (1983). The goal of Delaunay triangulation is to produce a set of triangles from a set of vertices; circumcircle of each of these triangles does not own any other point. Thus in 3D, tetrahedrons are extracted using spheres instead of circles. Delaunay tetrahedrons are used to locate particles inside a sensor mesh. The first step is to compute all tetrahedrons, we use the algorithm proposed by Barber et al. (1996). This algorithm is one of the fastest implementation to compute Delaunay triangulation and moreover, it works for any dimensions. We feed its input with coordinates of the sensors and output is a set of tetrahedrons. Then, for each particle, it is necessary to locate it compared to each tetrahedron. To reduce the computation time, we have reduced the number of tested particles using a AABB (axis-aligned bounding box). This method was introduced by Van Den Bergen (1998) to compute hull of mesh using minimum and maximal points coordinates. The final goal is to manipulate only the particle which are inside this box. For each tetrahedron we have four vertices. To compute location of particles, we take three vertices (called a face) from a tetrahedron, then we cast a ray from a particle to the remaining vertex of tetrahedron, and so on. If rays casted from the same particle has collided with a minimum of three faces, this particle is inside tetrahedron. The ray casting method is inspired from works initiated by Snyder & Barr (1987). Authors introduce a method to compute light on a triangular mesh. After this method, some particles have to be located. Most of them are outside the sensor mesh. We use Voronoï cells to compute their location. This method is developed in many algorithms: Qhull by Avis & Bhattacharya (1983) or Vorro++ by Rycroft (2009). Voronoï diagrams are used to partition space in several dimensions. In our case we have used our own method, we locate for each particle the nearest sensors. Figure-5 presents the pipeline of algorithm in 2D. After these steps, particles know their dependency from sensors, this information can be represent by a weight. We compute an invert function based on distance for each particle:

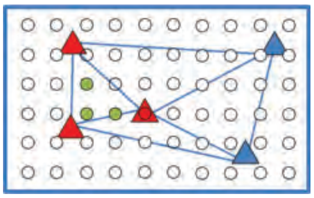
$$\forall t \in T, \forall p \in P^t, W_p = \sum_{s \in S^t} (V_s \cdot (1 - \frac{d(p,s)}{\sum_{s \in S^t} d(p,s)})), \tag{5}$$

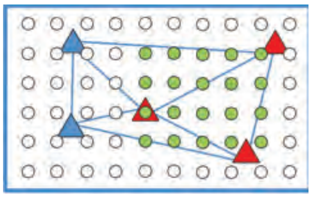
with T is the soup of tetrahedrons, P^t represents the set of particles inside the t-indexed tetrahedron and S^t the set of the internal sensors of t-indexed tetrahedron. *d()* is a weight function based on euclidian distance. The entire algorithm is presented in Algorithm-1.

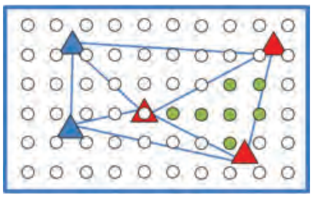

(a) Original model, sensors are represented by triangles and particles are represented by circles.

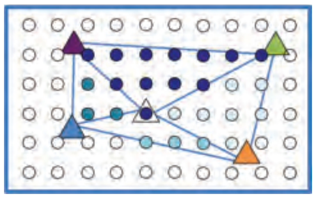

(b) Mesh produced from sensors: green particles are used by Delaunay algorithm while orange ones are used by the Voronoï method.

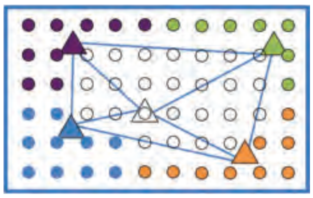

(c) First hull (based on AABB Van Den Bergen (1998)).


(d) Particles inside the triangle.


(e) Selection of the second hull.


(f) Particles inside the second triangle.


(g) Final result of Delaunay triangulation.


(h) Final result of Voronoï cells.

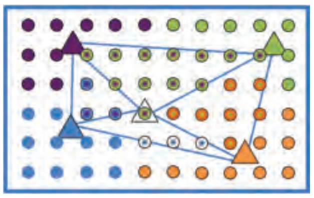

(i) Final result after weight function applied on particles.

Fig. 5. These schemas present a graphical representation of the method used to locate particles (representation is in 2D). Sensors are represented by triangles thus particles are dots.

In this chapter, we present a benchmark of different architecture to compute our algorithm. The goal is to find the best energy efficiency hardware. This benchmark is composed from standard computing method and parallel processing method. We present results of this benchmark (execution time) focusing on energy consumption. Parameters are: resolution of the point cloud, particles repartition schema, the compiler used and the method used for parallel processing.

Our visualization method is designed to used in different devices. The first parameter resolution, allows visualization needs to fit with abilities of hardware for rendering. The second parameter is volume creation, we define two kinds of point cloud construction: a regular method and a concentric method. Each of these methods produces a specific kind of render for the visualization and their construction time is also different because the number of particles produced by these two methods is different. Third parameter is compiler: we want to highlight if an important difference between a generic compiler and a dedicated compiler

Algorithm 1 The benchmark algorithm.

```
$S$ = AddSensors();
$P$ = AddParticles();
$T$ = TetrahedronExtraction($S$);
for all $T$ such as $t_i$ do
    for all $P$ such as $p_j \in$ AABB($t_i$) do
        if Locate($p_j$, $t_i$) $\geq 3$ then
            $p_j$.sensors.add($t_i$)
            $P = P - p_j$
        end if
    end for
end for
for all $P$ such as $p_i$ do
    $p_i$.distance = $+\infty$
    for all $S$ such as $s_j$ do
        if $p_i$.distance $>$ distance($p_i$, $s_j$) then
            $p_i$.distance = distance($p_i$, $s_j$)
            $p_i$.sensors.add($s_i$)
        end if
    end for
end for
ParticlesPonderation();
```

exists. The fourth and last parameter is based on different parallel processing methods, we have tested standard CPU parallel methods and GPU parallel methods.

This benchmark is focused on construction methods of the point cloud. We change the parameter to produce a multi resolution model of the room: from 0,01 to 5 (0,01 is a high definition mesh and 5 is a coarser mesh). Result is given in Table 1, we can see execution time for creation of the point cloud and number of particles produced. This first experimentation gives better results with concentric models than with a regular grid (for the high resolution mesh), but the number of particles produced by a regular grid is higher than concentric method. For very high resolution, object size does not fit in memory for the regular grid. For coarse resolution, we can see that computation on regular grid is the fastest.

	0,01	0,05	0,25	0,5	1	2	5
RG (time in seconds)			39,288	4,763	0,550	0,034	0,002
RG (mesh size)			6481032	780912	90552	9884	840
CS (time in seconds)	52,395	10,512	2,116	1,073	0,548	0,280	0,071
CS (mesh size)	8330976	1668563	335851	169333	86006	44260	19370

Table 1. Time results and number of particles for two methods, RG means Regular grid and CS is for concentric sphere.

Next benchmark is focused on parallel processing. Parallel processing methods can be implemented with OpenMP, MPI, on HPC, on FPGA, and so on. We have used OpenMP library to compile in parallel with GCC (Gnu Compiler Collection) and with ICC (Intel C++ Compiler), moreover, we have used a method based on GPU with OpenCL. Then, we have used a hybrid parallel method using OpenMP and OpenCL. And finally we have benched our algorithm on a HPC. Results are shown in the Table 2. Lower resolutions of the mesh (step at 0,01 and 0,05) swap memory and software becomes slower, this resolution produce an huge structured data is produce at this resolution. Moreover, the computation time is better compared to a standard computer for high resolution. But if the resolution becomes coarser, the laptop is faster. The results of the HPC in parallel shows that this method is really efficient compared to a standard programming.

	0,25	0,5	1	2	5
GCC 1	39,288	4,763	0,550	0,034	0,002
GCC with OpenMP 2	29,737	3,431	0,412	0,038	0,013
ICC 3	21,938	2,739	0,313	0,012	0,001
ICC with OpenMP 4	18,947	2,341	0,278	0,020	0,013
OpenCL 5		2,466	0,304	0,044	0,020
OpenMP and OpenCL 6		3,256	0,416	0,063	0,037
HPC 7	29,281	3,521	0,394	0,029	0,014

Table 2. Execution time in seconds on a: standard computer using GCC (1), standard computer using GCC and OpenMP (2), standard computer using ICC (3), standard computer using ICC and OpenMP (4), standard computer using OpenCL (5), standard computer using GCC with OpenMP (6) and OpenCL and finally using a HPC with GCC and OpenMP (7).

Next step is focused on energy consumption of each solution, parallel computing is a gap of energy and it is necessary to find the less consuming method. Energy formula is:

$$E = P \times T, \tag{6}$$

where E is the energy in joules, P the power in watts and T is the time in seconds. For this benchmark we have measured the energy consumed by the different devices during execution time. Figure-6 present energy results of the different executions: two main point can be extract, a standard computing is the best solution for a coarse resolution, and HPC is always the worst solution.

We can notice that solution using a spherical method is more efficient than the solution using a regular grid: the number of particles is denser, the resolution of the mesh is more important. None of other method provides bigger resolution for the regular grid on a standard computer. Another important point is that parallel processing is bigger consumer than standard CPU programming. Time of standard programming is longer but the energy used for multi core programming is more important. With a specific compiler: ICC, results are improved for the energy consumption and runtime execution compared to GCC. For the GPU programming we can see that the energy consumption is more efficient than the ICC compiler for high resolution, unless the memory of the graphic card is not enough and does not allow to produce a high-resolution mesh. Finally for the MacBook results, we see that the hybrid solution is not a good choice, exchange between memory and graphic memory produced latency on execution time, and using multi core programming and GPU programming give best results in terms of energy consumption. For the HPC results, energy is higher than the previous solutions. Professional CPUs are less efficient to save energy than standard CPU. The better solution for saving energy is to use a single core application but if we want to have a low computation time, parallel methods using GPU are more efficient for high resolution.

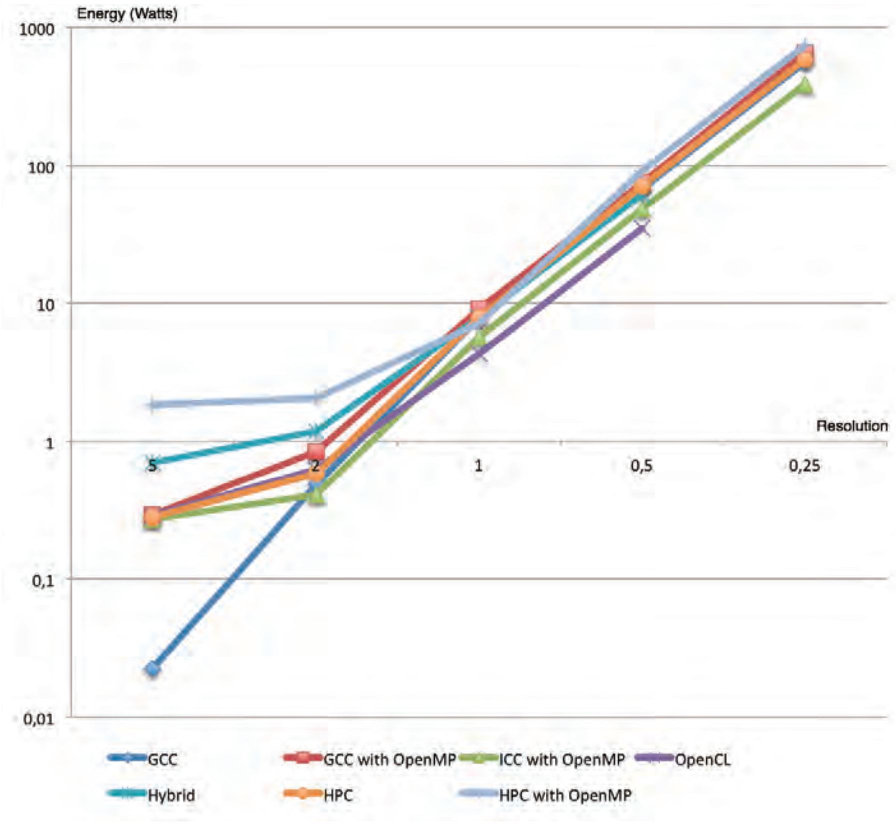


Fig. 6. Results of parallel benchmark.

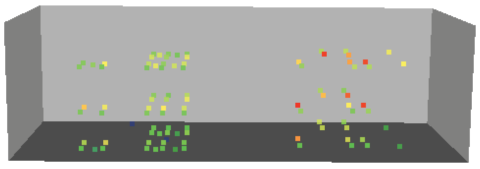
4.2 One dimension visualization

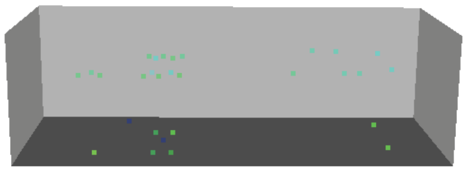
In this Section, we focus on the rendering engine. It is an important task to render content in an efficient way: data needs to be understandable and their render have to be in real time. In this part we present a method to produce visualization and we explain effectiveness. Moreover, sensors can produce multivariate data, but this part is dedicated to visualization of only one kind.

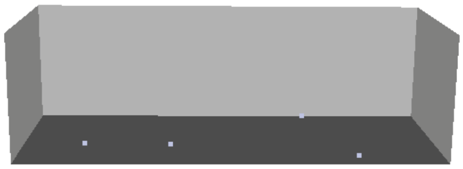
Data are provided by IBM, Montpellier, France. This data contains value of a set of sensors, located in the IBM Green Data Center (GDC). Sensors are set as layers, and they are multivariate: some of them own temperature, pressure or hygrometry. Another main information is the topology of the rooms. This information can reconstruct virtual rooms with sensors location.

Figure-7 presents the different sensors layers. We use temperature sensors as example, the content produced by this sensors is the most volatile sensors value.

The first step is to view the point cloud produced by the server. This point cloud is based on a simple point sprite primitive, with this method we are able to render a large set of vertices


(a) Location of temperature sensors.


(b) Location of pressure sensors.


(c) Location of hygrometry sensors.

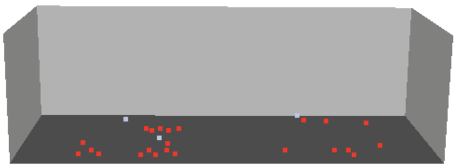
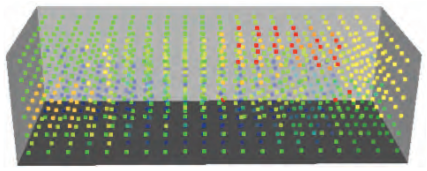

(d) Location of APS sensors.

Fig. 7. Location of the different kind of sensors.

in real time. Figures-8(a) gives a representation of the regular grid pattern and Figure-8(b) is a view of spherical method for the particle representation. This kind of primitive is efficient to navigate into data and understanding is not easy. point clouds have some issues, depth is not easy to catch and moreover aliasing produced by this view is a weak point for the usage of this visualization.


(a) point cloud visualization of a regular grid.

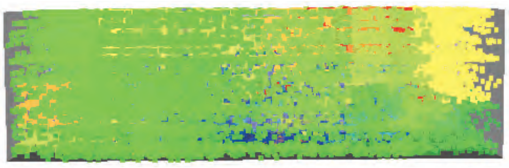
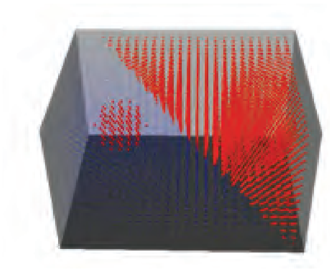

(b) point cloud visualization for a concentric model.

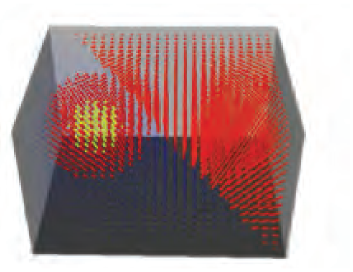
Fig. 8. Raw visualization using point sprite representation.

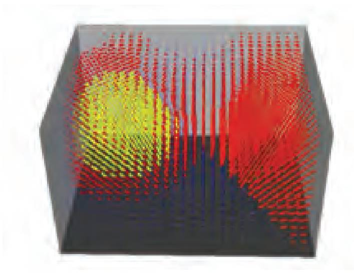
To smooth result of the previous method, we implement a convolution method. Convolution is used in 2D imaging to smooth an image. It works by using a kernel function applied on each pixel. Results smooth the entire room volume, unfortunately same issues than the previous method persists, visualization is hard to understand and computation time increases. Result seems to be more realistic, without convolution an aliasing color effect is produced by particle, from cells or tetrahedron border, with convolution this brutal change is smoothed.

This kind of view has another important issue, data are dynamic, and it is hard to catch an update. It is necessary to produce efficient method to identify updates. To reduce computation, bandwidth of client, data transmission between server and client is done through LOD paradigm, update is realized on sensors and then on the first neighborhood of it, the graphical result of this method is presented Figure-9.

But the previous method is only a method to understand the update of data, real value is not visible on updated particles. To update and catch this modification, we propose to change


(a) Neighbor update at T = 0.


(b) Neighbor update at T = 1.


(c) Neighbor update at T = 2.

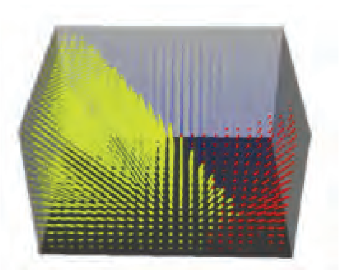
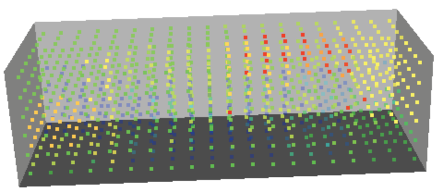

(d) Neighbor update at T = 4.

Fig. 9. Update processes based on neighbor exploration.

the size of updated particles. Thus, updated particles are smaller than fixed value particles. Updated data is now easier to catch, as shown in Figure-10. The render cost and render issue are the same than a standard point sprite visualization.


(a) Particle view at T = 0.

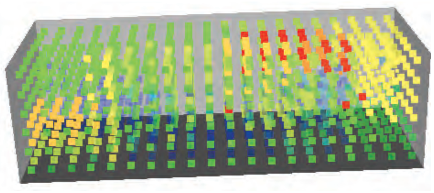

(b) Particle view at T = 1.

Fig. 10. Updated particles have a bigger size than static value particles.

Another possible solution is to use a method based on convolution algorithm. Updated particles are moved from their original location depending from the result of the convolution kernel. This method gives a simple view of influence of each particle. Airflow of rooms can be model by this method as presented in Figure-11.

To improve the previous solution instead of point cloud, vectors are used to represent results of the convolution mask. Origin of the vector is the location of particle, and the end point is the location of the previous method as illustrated Figure-12. Has we see using vectors is not an efficient method to understand data.

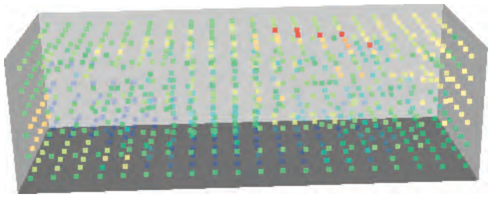


Fig. 11. Particles are moved depending from result of convolution mask.

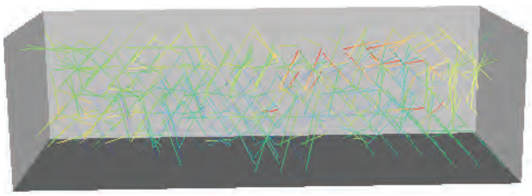
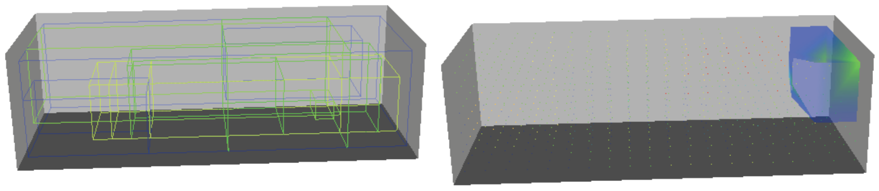


Fig. 12. V-neck direction of most influent sensors for each particle.

An important point in data visualization is to extract quickly the main information. Main information is variable and depends of the topic. In our case, we produce a thermal visualization and try to extract the relevant temperature area. To do this, we define methods based on boundary extraction or graph theory.

The first method extracts bounding box from a set of particles. The goal is to produce hull of a 3D set of points, we use AABB hull due to the low cost of computation. In our case, data can be aggregated by values, and components are only managed by a delta value. First results are presented in Figure-13.



(a) Segmentation of space using extraction of component by value at T = 0. (b) Triangulation of the biggest component at T = 1.

Fig. 13. Segmentation based on AABB algorithm.

We can also convert our point cloud to a graph. Each neighborhood vertex with a similar value can be connected together. This method creates a set of connected components. Formally, this method is defined by the graph: $G = (V, E)$, E the set of edges and V the set of vertices. G is composed by subgraphes: $G = \{G_1, ..., G_N\}$, G_j where $G_i \cap G_j = \emptyset$, $G_j = (V_j, E_j)$, $V_j \subset V$, $E_j \subset E$ and $\forall j$, G_j is connected. Then, we can use the same bounding method: AABB for each component. Results produce a messy visualization, accuracy of component is more realistic, but the number of boxes is too important. This issue can be visualized on Figure-14, it is also possible to reduce this kind of noise by increasing the delta value of each component.

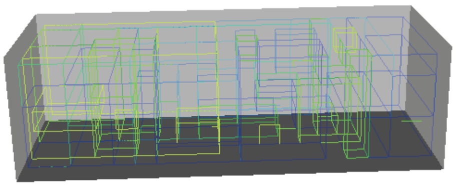


Fig. 14. Segmentation of particle cloud with a simple extraction of connected component.

Finally, to resolve aliasing issue, we focused on rendering of volume. point cloud is not the best method to render a volume, so we have tested several other methods to give user the best point of view.

Our first method is based on triangulation. We take the point cloud and apply a marching cube method, this method is introduced by Lorensen & Cline (1987), the goal is to produce a surface mesh from a point cloud set. Results are given in Figure-15(a). We can see some visual issues due to the triangulation. But the main problem is computation time, indeed, methods used are not in real time for large set of points.

To produce triangulation of the point cloud, we also used the Delaunay Algorithms, Figure-15(b) gives a representation of this method, as we see, this algorithm simplify the point cloud mesh and only extremal points are conserved.

Another idea is based on extraction of the connected component, when we triangulate them, rendering time is too long; result is given in Figure-15(c), some issue appear for color mapping.

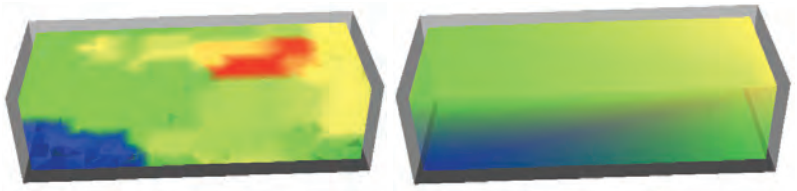
This part will present voxel view of rooms. Unfortunately, the rendering cost of these primitives is too important compared to a standard point cloud visualization, the results is presented Figure-15(d). This method is well suited for a volume rendering compared to point cloud, but update of each frame take too mush time.

Last method used is shader programming language. With this method, we get best results and can explore data in an efficient way. Shaders are used after the rendering of the 3D images, computation is done on GPU on a rasterized 2D image. Figure-16 gives some results of this method, different shaders have been used to render each of these figures. Exploring data and understanding is easier than all solutions proposed before, shader methods are mostly used to render flow visualization.

4.3 Data mining through visualization

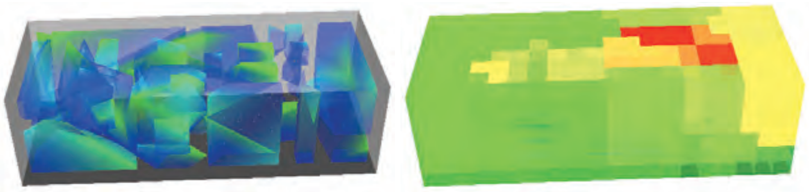
In this section, we present a method used to dig into data content, the goal is to propose content for the whole system and optimize consumption of a building. We focus on method used for annotate the building model and ways to add content.

We propose some interaction tools to give user abilities to add, update and delete content from current instance of the green box model. Original model extracted from the BMS does not contain enough information, thus it is necessary to add these missing informations. Moreover, buildings have their own life such as the topology and the environment. These informations always changed and BMS are not developed to catch these updates. The interaction tool we propose is composed of two shapes: a sphere and a box. This method gives to the user the



(a) Marching cube method.

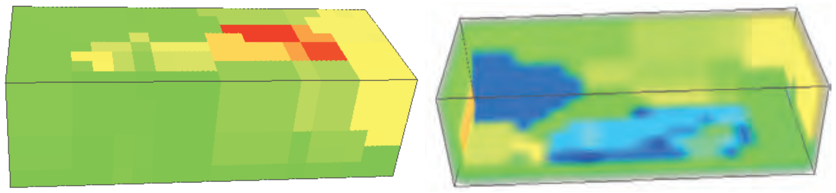
(b) Triangulate mesh of point cloud surface.



(c) Triangulate meshes of all component.

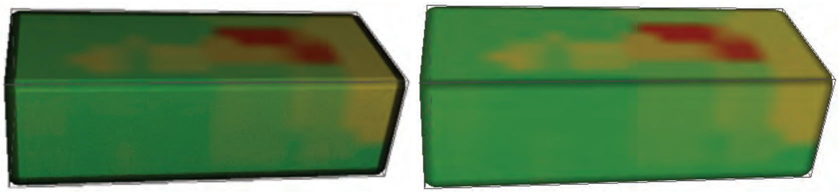
(d) Voxel based visualization.

Fig. 15. Severals methods used to produce a surface mesh from our point cloud (visualization not in real time).



(a) Shader visualization, witha fixed shader function.

(b) Shader visualization, based on rayTracing method.



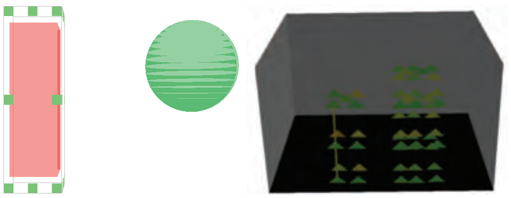
(c) Shader visualization, with isosurface extraction.

(d) Shader visualization, with isosurface extraction and a transfer function based on light intensity.

Fig. 16. Real time visualization of particles using shader programming language.

possibility to improve building by standard modifications. Another feature allows adding information like influence of sensors, furniture location, split and rooms. All these new contents will introduce new relevant informations for the green box. Figure-17(a) presents two shapes used to dig into data, and manipulator used for placement.

Another content which can be added to the pivot model, is the correlation. Correlations are links between data, when a value of data set changes, each link is influenced. Most modules of RIDER green box are developed to find these correlations, unfortunately some of them can be wrong or missing. But user can discover, update and propose new of them. This tool consists in producing links between data using a simple "Line" primitive with color depending from weight of each node. Correlation can be added to different objects in a room such sensors, furnitures and valve. Figure-17(b) presents an overview of this method.



(a) Digging tools to add, update or remove content.

(b) Example of correlation between sensors.

Fig. 17. Tools to dig into data.

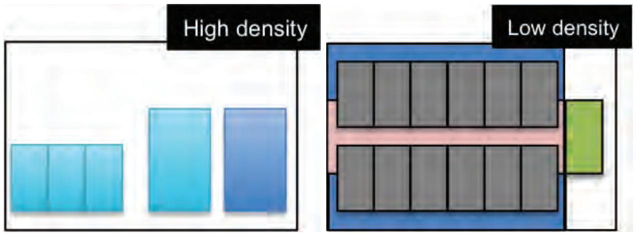
To provide new data in the Pivot model, we use a simple XML message. Messages are composed by unique component's id, and new features. For annotation, feature is a text message. For a wall adding, two XML messages are sent: the first one composed by the old room, with topology switch, the second one composed from a new unique ID and shape of this room. These messages are formatted to fit with input of RIDER model, each plug-in sends messages with this format.

5. Case study

In this section, we present results of the Section 4 methods with real data. We use data provided by our pilot: Green Data Center from IBM Montpellier, France. The goal of this data center is to propose efficient methods to reduce energy consumption. This data center is composed by two rooms. The first room is dedicated to servers with a large number of CPU (cooling is done using specific case), called High density. The second room is for low-density servers with a standard air cooling, air management is done through confined alleys. Figure-18 presents a schema of these two rooms. Our data set is composed by shape of merged rooms and sensors data (location, kinds and value). To help other data mining components, it is necessary to provide more content. In this case study, we will extract two spaces (one for each room), then we propose to analyze rooms, extract content and optimizations.

5.1 visualization of the case study

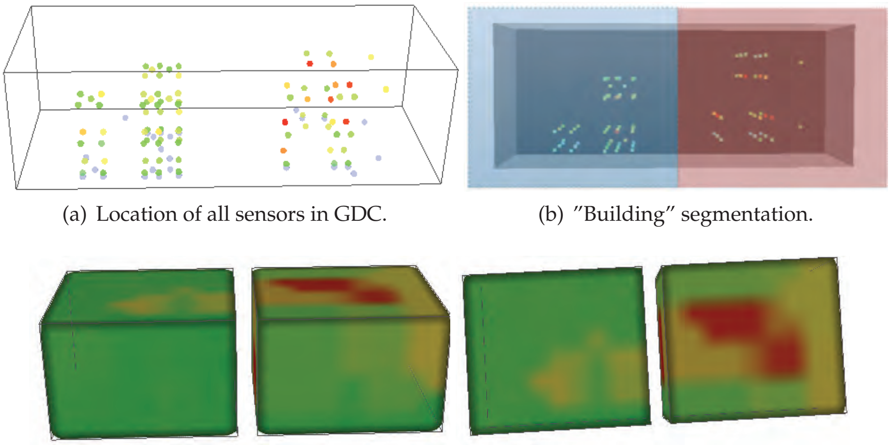
Our first step is to split the two rooms into two spaces. For this, we analyze also sensors location as illustrated Figure-19(a). Obviously, we can find limits of each room, Figure-19(b). It is necessary to identify the content of these rooms. We use the particle view presented Section 4 to identify rooms. We know that the low-density room is organized with area as stated in Figure-18(b). Figure-19(c) and Figure-19(d) present two different views of the room



(a) Schema of high-density room (top view). (b) Schema of low-density room (top view).

Fig. 18. User knowledge on GDC, two rooms composed the green data center: the low-density room and the high-density room.

using a volume method. The right room seems to be segmented in different areas. This means that this room is low density.

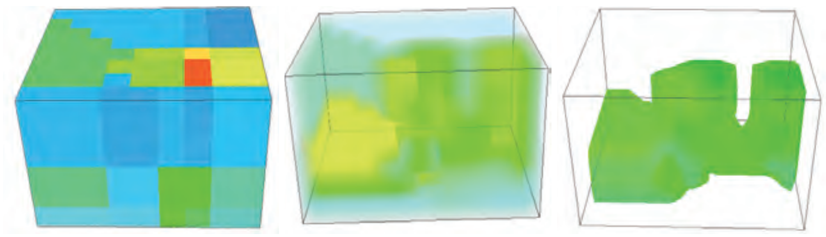


(a) Location of all sensors in GDC. (b) ”Building” segmentation.

(c) Front view of two rooms, temperature view. (d) Top view of two rooms, temperature view.

Fig. 19. Different steps to identify rooms.

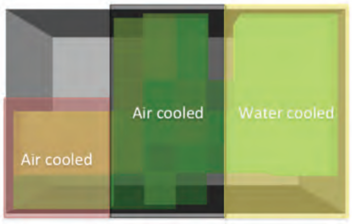
The next goal is to identify content from the high-density room. We analyze data sensors of each kind, using a time function, temperature is most variable value, other kinds are not enough relevant. To find content, we manipulate temperature slider and look for relevant information. Figure-20 gives an overview of different tests done, we found interesting information, for example at center of the room a continuous mass is present. User knowledge is two large cases and a group of servers compose this room, see Figure-18(a). With our flow visualization, we can identify each part of knowledge. But an error occurs on the stack of servers, because they are too low. Another step is to add correlation between stacked sensors. Finally from the green box, we have proposed several contents concerning the location and the size of server (heater part of the room) and moreover some links between sensors.



(a) High density room visualization.

(b) Temperature flow visualization (an alpha function is applied on low temperature).

(c) Extraction of highest temperature, popping of three blobs.



(d) Identification of blobs.

Fig. 20. Visualization mining of high-density room: extraction of servers.

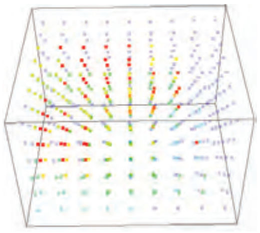
For the low-density room, the goal is to extract alley information, this part of data center is confined to improve energy usage. It is decomposed in three alleys: two cold and a hot alley as stated in 18(b). We use the particle visualization to identify these areas. Figure-21 gives results. Hot alley can be located in center of the room. Another information appears in this visualization, Figure-21(e) presents location of CRAC (CRAC are cooling systems), they are used to extract hot temperature from the alley. As presented before, we can add for each sensor correlation between them (stacked, and alley sensors).

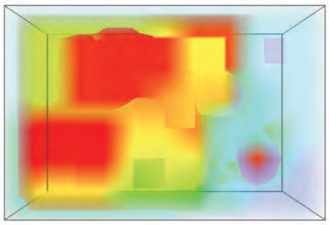
All these new informations are compared to real knowledge from data center and matched. We can now analyze visualization to propose data center optimization. A recurrent issue appears on the low-density room; top servers are too hot during a long while. It is necessary to refactor location of servers to improve their server air flow.

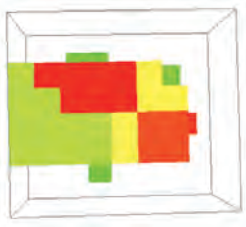
5.2 Case study improvement

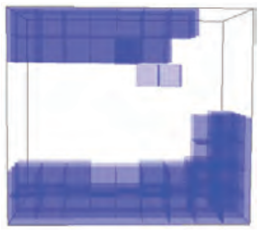
IBM Montpellier GDC has a low PUE near 1.5. Using all presented visualization in subsection 5.1 we can propose some improvements for reduce energy consumption of the data center. Firstly, we will focus on energy usage of the high density room, then on the low-density room.

In the high density room we can highlight some points. As stated in Figure-20(d), we can extract location of different servers, the left group is a group of server stacked on a classic server case. If we analyze results from time-lapse visualization, we can see that hottest servers are mostly on the bottom of server case. This part seems to not need changes. For the second air cooled case (center of Figure-20(d)), using a time based visualization, we can


(a) point cloud view of low-density room.


(b) Top view of low-density room.


(c) Hottest region.


(d) Median temperature, fresh alleys.

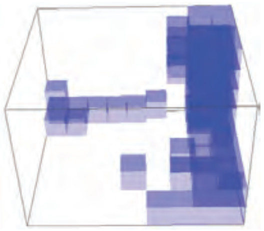

(e) Coldest temperature, identification of CRAC region .

Fig. 21. Visualization method to extract different alleys from low-density room.

find refactoring optimization in this case. Sometime, hottest servers are placed at the top of the case. To reduce cooling, it is necessary to get down these hot blades. For the last case (the water cooled case), we can find the same pattern as the previous one. Top servers become hotter than down servers. It is necessary to bring them to the bottom of the case to reduce the energy consumption of this case. The cooling system of the global room is located in the left back corner of the room, unfortunately, it is seems to only cool a small part of the room and water cooled server capture an hot air.

The second room visualization present a new way to improve data center. The hot alley of the data center are segmented but we can see that computation of servers is done on the top of the stack. Thus, it is necessary to refactor server blade to improve the airflow of the room and reduce the energy consumption. To propose other improvement, it is necessary to get more data dimensions, for example: energy of each server. Scalability features have to be added to be able to manage buildings and group of buildings in the same time.

6. Conclusions

Specials devices can be used to avoid electrical pic of consumption during journey and smooth usage on the electrical network. Electric usage of consumer can be predict (usage of washing machine) and optimization can be done to reduce the network load. Domotic can be used to automate process, for example window awning can be set depending on sunshine or external temperature, the goal here is to reduce usage of cooling system and thereby reduce electric consumption. Smart buildings are able to react when classic buildings do not.

Buildings consume 38 % of the global energy and produce between 25 % and 40 % of the CO_2 emission. Buildings are expensive investment, not only for their construction cost or buildings fees, but also for their ecological cost.

Our project called RIDER has a goal to develop a new information system to optimize energy efficiency of building or group of buildings. To propose optimizations, energies transfer between buildings can be realized. This transfer can be done on different energies: classic, green (extracted from wind system or photovoltaic) or deadly (produce by industry usage). This project is based on development of new data processing method (models, data mining, visualization, IT architecture and communication).

In this chapter, we present a concept of the green box to dig volumetric data for extracting important features of buildings, the goal is to provide some optimizations. Data are provided from a Green Data Center at IBM Montpellier, France. Our visualization software gives the abilities for users to suggest some topology modifications based on room definition.

Walls have to be added, many informations provided by building manager will be improved. Other stuff needs to be added to improve the building model. It is necessary to set a part of furniture to know the influences of model. To propose this model, we use a particle paradigm to visualize the volume of each room. To dig into data, we give to the user the ability to filter data and submit some updates to IT architect.

First global results of the whole architecture can expect that optimization can reduce energy between 30% to 40 % of a building.

7. Acknowledgments

We would like to thanks the PSSC (Products and Solutions Support Center) team of IBM Montpellier, France for having provided the necessary equipment and data needed for this experimentation. We also thank the FUI (Fonds Unique Interministeriel) for their financial support.

8. References

Avis, D. & Bhattacharya, B. K. (1983). Algorithms for computing D-dimensional voronoï diagrams and their duals, *Advances in Computing Research* 1: 159–180.

Barber, C. B., Dobkin, D. P. & Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls, *ACM Trans. Math. Softw.* 22: 469–483.

Clark, J. H. (1976). Hierarchical geometric models for visible surface algorithms, *Commun. ACM* 19: 547–554.

Green, S. (2007). Cuda particles, *NVIDIA Whitepaper, November* .

Hauser, H., Ledermann, F. & Doleisch, H. (2003). Angular brushing of extended parallel coordinates, *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, IEEE, pp. 127–130.

He, T., Hong, L., Kaufman, A., Varshney, A. & Wang, S. (1995). Voxel based object simplification, *Proc. SIGGRAPH Symposium on Interactive 3D Graphics*, pp. 296–303.

Ilcìk, M. (2008). A framework for global scope interactive visual analysis of large unsteady 3d flow data, *CESCG* .

Kapferer, W. & Riser, T. (2008). Visualization needs and techniques for astrophysical simulations, *New Journal of Physics* 10(12): 125008 (15pp).

Keim, D., Mansmann, F., Schneidewind, J. & Ziegler, H. (2006). Challenges in visual data analysis, *Information Visualization, 2006. IV 2006. Tenth International Conference on*, IEEE, pp. 9–16.

Konyha, Z., Matkovic, K. & Hauser, H. (2009). Interactive visual analysis in engineering: A survey, *Posters at SCCG 2009* pp. 31–38.

Lex, A., Streit, M., Kruijff, E. & Schmalstieg, D. (2010). Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context, *Pacific Visualization Symposium (PacificVis), 2010 IEEE*, IEEE, pp. 57–64.

Liu, Z., Lee, B., Kandula, S. & Mahajan, R. (2010). Netclinic: Interactive visualization to enhance automated fault diagnosis in enterprise networks, *IEEE VAST* pp. 131–138.

Lorensen, W. E. & Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm, *SIGGRAPH Comput. Graph.* 21: 163–169.

Luebke, D. (1997). A survey of polygonal simplification algorithms, *IEEE Computer Graphics and Applications* (21): 24–35.

Luebke, D. P. (2001). A developer's survey of polygonal simplification algorithms, *IEEE Computer Graphics and Applications* 21: 24–35.

Migut, G. & Worring, M. (2010). Visual exploration of classification models for risk assessment, *Proceedings of the IEEE Conference on Visual Analytics Science and Technology 2010*.

Moenning, C. & Dodgson, N. (2004). Intrinsic point cloud simplification, *Proc. 14th GrahiCon* 14.

Pauly, M., Gross, M. & Kobbelt, L. (2002). Efficient simplification of point-sampled surfaces, *Visualization, 2002. VIS 2002. IEEE*, IEEE, pp. 163–170.

Piringer, H., Kosara, R. & Hauser, H. (2004). Interactive focus+ context visualization with linked 2d/3d scatterplots, *Coordinated and Multiple Views in Exploratory Visualization, 2004. Proceedings. Second International Conference on*, IEEE, pp. 49–60.

Rosen, P. & Popescu, V. (2011). An evaluation of 3-d scene exploration using a multiperspective image framework, *Vis. Comput.* 27: 623–632.

Rycroft, C. H. (2009). Voro++: a three-dimensional voronoi cell library in c++, *Chaos 19* . Lawrence Berkeley National Laboratory.

Schroeder, W. J., Zarge, J. A. & Lorensen, W. E. (1992). Decimation of triangle meshes, *SIGGRAPH Comput. Graph.* 26: 65–70.

Shneiderman, B. (2001). Inventing discovery tools: Combining information visualization with data mining, *in* K. Jantke & A. Shinohara (eds), *Discovery Science*, Vol. 2226 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 17–28.

Snyder, J. M. & Barr, A. H. (1987). Ray tracing complex models containing surface tessellations, *SIGGRAPH Comput. Graph.* 21: 119–128.

Song, H. & Feng, H.-Y. (2009). A progressive point cloud simplification algorithm with preserved sharp edge data, *The International Journal of Advanced Manufacturing Technology* 45(5-6): 583–592.

Van Den Bergen, G. (1998). Efficient collision detection of complex deformable models using aabb trees, *J. Graphics Tools*.