Academie de Montpellier
Université Montpellier II
Sciences et Techniques du Languedoc

MÉMOIRE DE STAGE DE MASTER M2

effectué au Laboratoire d'Informatique de Robotique
et de Micro-électronique de Montpellier

Specialité : **IMAGINA**

Visualisation et analyse d'images 3D de grandes dimensions

par **Anas KHARBOUTLY**

Date de soutenance : **09/07/2013**

Sous la direction de **Gérard SUBSOL**

A mon pays blessé

Syrie

A mon superviseur précédent qui est décédé l'année dernière.
Il sera toujours dans ma mémoire.

Ph.D. Haiyan HOUSROUM

Remerciements

En préambule de ce mémoire, je souhaiterais adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué á l'élaboration de ce mémoire ainsi qu'á la réussite de cette formidable année universitaire.

J'adresse tout d'abord mes sincères remerciements á **M. Gérard SUBSOL**, chercheur CNRS, pour la qualité de l'encadrement dont il m'a fait bénéficié, pour sa grande patience et en particulier pour son attention et sa disponibilité dans la correction de ce mémoire et sans qui ce mémoire n'aurait jamais vu le jour.

Je tiens á remercier très chaleureusement **M. William PUECH**, Professeur á lUniversité de Montpellier 2 et responable de l'équipe ICAR (image and Interaction) du LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier) de m'avoir accueilli au sein de son équipe, ainsi que pour ses remarques, ses conseils et ses suggestions qui m'ont permis d'apporter d'ultimes améliorations.

J'adresse mes plus sincères remerciements á **mes parents**, **ma sœur** et **mes frères** qui m'ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire.

Je remercie tout particulièrement mes amis **Yasser ALMEHIO**, **Ziad ALMAKSOUR**, **Seza ADJOYAN** et **Anas SHATNAWI** pour leur soutien et pour le temps qu'ils m'ont consacré tout au long de cette période, sachant répondre a mes interrogations malgré leur charge académique.
Je remercie **Subhi ISSA** et **Obaida HANTEER** et tous mes amis.

D'une façon plus générale, je remercie l'équipe **ICAR** pour son accueil chaleureux et pour m'avoir accepté en tant que stagiaire pendant ces cinq mois.

En fin, j'aimerais également remercier l'ensemble de l'équipe **pédagogique de la formation Master 2 Informatique spécialité IMAGINA** de l'UM2, pour avoir assuré la partie théorique de ma formation.

Merci á toutes et á tous.

Anas

Resumé

L'imagerie médicale 3D est considérée comme une application importante dans le domaine du traitement d'images. Elle est utilisée pour aider les médecins á améliorer et accélérer le processus de diagnostic en visualisant et en appliquant des opérateurs de traitement a fin de détecter ou de segmenter certains structures dans les images.

Toutefois, certains dispositifs (par exemple, CT et micro-CT) génèrent de très grandes images 3D. Ces images ne peuvent pas tre visualisées ou mme traitées sur les ordinateurs standards. Pour résoudre ces problèmes, plusieurs approches ont été proposées. Néanmoins, elles sont souvent partielles. Dans ce rapport, une approche est présentée pour visualiser et traiter de très grandes images médicales 3D, acquises par tomodensitométrie ou micro-CT scanner, dont les dimensions peuvent atteindre 2000 * 2000 * 2000 voxels. A fin de valider et évaluer l'approche proposée, nous l'avons appliquée sur une image medicale 3D de fœtus. Les résultats montrent une visualisation interactive de ces données et la capacité d'appliquer certains opérateurs de traitement pour une application clinique.

Mots-cles: imagerie médicale 3D, image de très grande taille, visualisation interactive, décomposition en blocs, opérateurs morphologiques

Abstract

Medical image processing is considered an important topic in the domain of image processing. It is used to help the medicines to improve and speed up the diagnosis process by visualizing and applying processing operators in order to detect or segment some objects in such medical images.

However, some devices (e.g. CT and micro-CT) generate very large 3D medical images. These images cannot be visualized or even processed on standard computers. To address this problem, many approaches have been proposed to visualize and process these images. Nevertheless, these approaches did not provide a clear specification or solution for such kind of problems.

In this research, an approach is presented to visualize and process very large 3D medical images, ideally acquired by CT or micro-CT scanner, which dimensions can reach 2000*2000*2000 voxels.

In order to validate and evaluate the proposed approach, we applied it onto an 3D medical image of foetus. The results show an interactive visualization of this foetus and the ability to apply some processing operators.

Keywords: medical image processing, very large 3D medical images, interactive visualization, block decomposition, morphological operation, overlapping

Contents

1 Introduction

Medical imaging has become nowadays an essential organ in the world of medicine, which refers to the techniques that can be used to have a look inside the body without need to be opened up surgically.

Figure 1: Micro-CT

Medical image processing by the way, has become also a common technique in the domain of image processing which benefits from technology development and image processing field in the medical sides. Brain segmentation, vascular network segmentation, cancer detection and so many other applications have been basics in everyday medical treatments that require from the other hand a continuous development in this domain. The processing of medical images isn't limited to the small medical images any more especially in the presence of medical imaging devices like micro-CT in figure (1) and CT-scan which provide very large or huge 3D medical images that may reach to 1000 voxels in each dimension or even more. Consequently, the size of our image may reach 1.8 Gb and more.

The main role f these medical images is to be used in diagnosis and treatment sides after segmentation operations or detect tumors, but these huge images can't be visualized or processed on standard computers, so we are in need for other or specific techniques for visualizing and processing these images. We want to visualize a very large 3D medical image by size around 2000*2000*2000 voxels in nearly real time and make the basic processing operators.

The very large 3D image as mentioned cannot be visualized on a standard computer because of existence of some obstacles like the required memory, required time for visualization and the required field of view since we cannot load or visualize these images, we cannot go forward with processing operators. These standard computers have small memory and so it is required smaller data to be loaded, visualized and processed.

From this concept and need we go toward reconstructing smaller data that we called blocks decompositions and down-sampling where we construct a low resolution version of the original image that may be loaded and visualized, but by the way we are in need for visualizing the original high resolution so we decomposed the image into sub blocks that may be loaded and visualized separately. We work on smaller data and for a global

interactive visualization , we use down-sampled data by the possibility of getting access to any part of the original data in order to have a focused and accurate visualization or to apply image processing operators.

We work on very large 3D medical image of fetus which was generated by a micro-CT scan at the University of Montpellier 2 by size 2048 * 2048 * 2740 isotropic voxels of 36 microns. We want to visualize this 3D image as MPR (Multi-planar reconstruction) mode that reconstruct the 3D view by stacking the 2D slices by Z axis, we can go throw the slices along each axis to visualize the three dimensions views, figure (2).

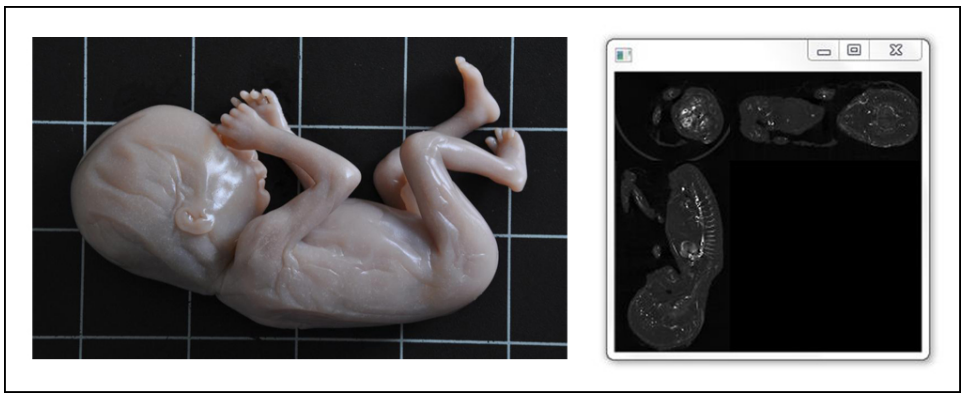

Figure 2: Foetus and its image in MPR mode

Here is an example of two slices from the same previous foetus image, figure(3):

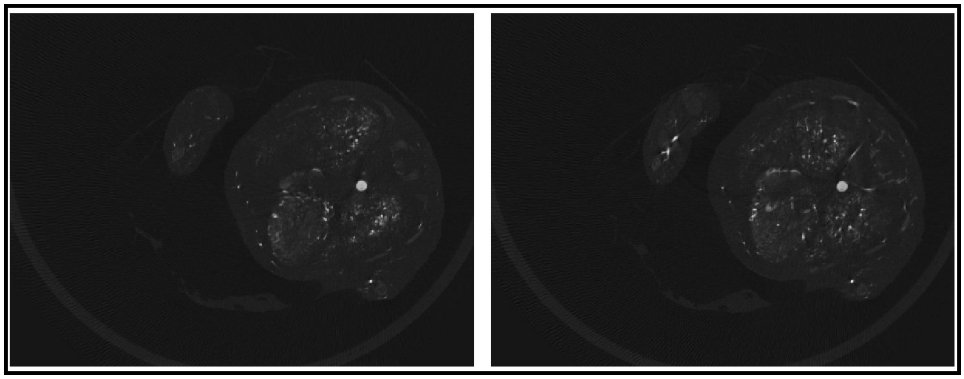

Figure 3: Two scaled slices from a foetus image by size 2048 * 2048

After visualizing the very large 3D medical image, we want to be able to apply some segmentations operations:

- Skin and vascular network segmenting
  Due to the high-contrast, a thresholding should be enough.

- Segmenting of bone
  Due to partial voluming, the voxels located around the vessels have the same intensity as those of the bones. In order to remove them, we are going to perform a dilation after vessel segmentation and delete these voxels. So, we need mathematical morphology operators, figure (4).

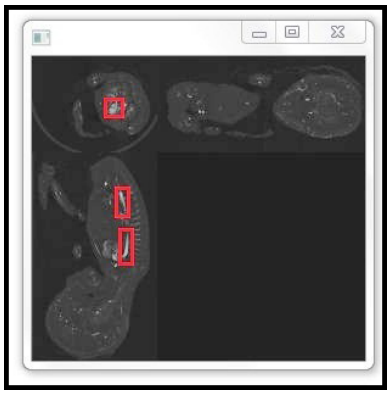

Figure 4: Example of required bone segments

In the coming section we will preview the previous works which related to visualization and processing, then we will provide our contribution in the third section, then we will show our results in the fourth section and finally, we will give our vision in the last section under future work.

2 State-of-the-art overview and analysis

2.1 Visualization of Large 3D Image

These very large 3D images that may reach to about 2000 voxels in each dimension can't be loaded normally in the presence of many limitations like a memory or field of view that requires in two cases smaller data to be loaded, visualized and processed, in order to get smaller data two previous approaches were proposed: Decomposition into blocks and multi-scale coding.

2.1.1 Decomposition into sub-blocks

Since the very large 3D image can't be visualized on standard computers, we need a smaller data that may be visualized, in order to get a smaller data we have to break down the very large 3D image data set into sub-blocks and reconstruct 3D volumes that may be loaded into limited memory and visualized onto standard computers.

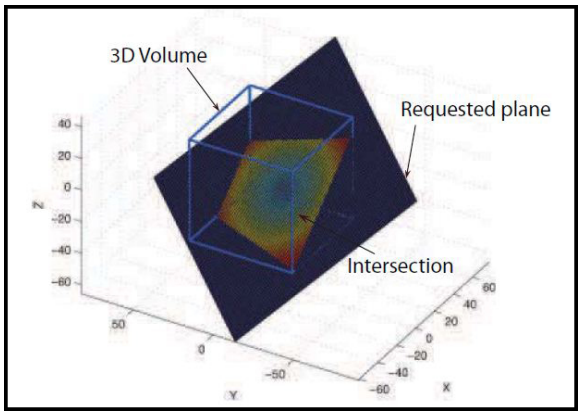

Figure 5: Reconstructed plane

May decomposition into blocks be the optimal solution for such visualization problem as this, but these needs to be studied and managed care as well in order to get an optimal performance in visualization that could be used in the second section of processing operations. In the paper "Optimization of Overlapped Tiling for Efficient 3D Image Retrieval" [1] a client-server method was proposed for visualization and rendering using limited memory on a personal computer, where the client requests a specific plane view from the server that in his turn construct the required view and provide it to the client.

Actually, this operation is not simple as it seems, but many operations are involved during the request view and visualization of this view in addition to that these operations need to be done nearly in real time. When the large image is decomposed into sub-blocks it will not be large any more, these sub-blocks which constitute the large image are stored on the server, as the client request a specific plane that means some blocks need to be loaded (sent to

client) to construct the required plane as shown in figure (5), these blocks need to be selected according to the position of requested plane view in addition to the intersection between blocks which construct the exact field of view.

In blocks decomposition and load these blocks to visualize the requested plane view, two main issues should be taken into consideration: First, the requested plane by the user will be different in each request that requires different blocks to be loaded into memory at each request consequently random access to the requested blocks. Second, the transmission data rate between the client and the server that should be minimized.

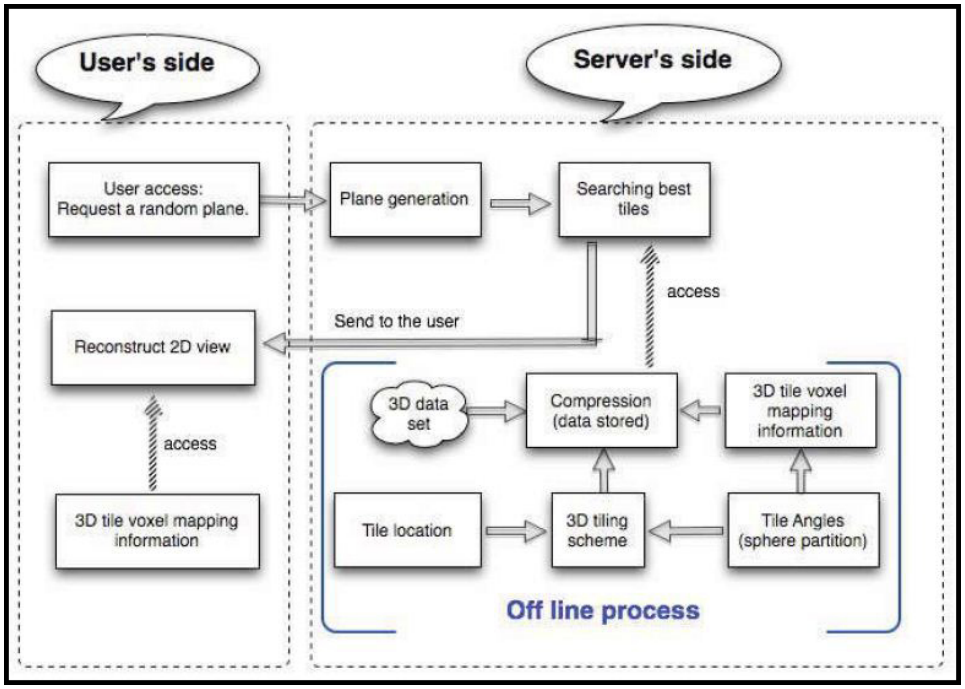

Figure 6: Components of visualization process

The used system as shown in figure (6) is consisted of five components which manage the visualization operation starting by user request and end by reconstruct the 2D plane view: (1) 3D blocking scheme that defines each blocks direction and the related rotation center, (2) mapping algorithm, (3) block searching algorithm that define the required blocks to be loaded into memory to visualize a specific plane view, (4) 3D compression that is applied on each block and stored on the server, (5) determine the blocks intersection in order to reconstruct and display a specific plane view. The basic production

of this paper is the prediction model that will predict the transmission rate between client and server side in addition to the required memory for storing the blocks, this model is defined using the mean and standard deviation of the number of blocks per voxel and its main role is providing a specific information that concern the performance and define the required parameters for decomposition and visualization operations according to:

- Develop the existing 3D blocking scheme which in his turn will verify the prediction model.
- Increase the server side storage which in its turn will decrease the client side storage.
- A minimum data transmission rate between client and server that leads to a speed access to the decomposed blocks.
- Decrease the required run-time memory on the client side

2.1.2 Multi-scale coding

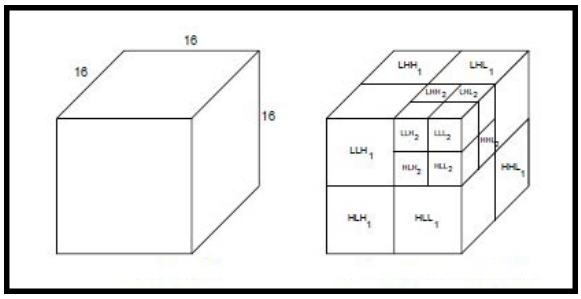

Figure 7: Unit block and decomposed unit block

As mentioned previously since a very large 3D image cant be loaded and visualized on standard computers that require smaller data, this smaller data can be either small in size as previous states that lead us to block decomposition or smaller in resolution that may be achieved in constructing a low resolution image by multi-scale coding. In the paper "Wavelet-Based 3D Compression Scheme for Interactive Visualization of Very Large Data" [2] a method-based wavelet transformation was proposed for 3D compression scheme. Wavelets transformation [2] is about the mathematical approach which provides hierarchy representation and decomposing function in the multi-resolution form. Actually, this method consists of three stages: (1) 3D wavelet transformation, (2) encoding wavelets coefficients and (3) reconstruction voxel values.

In the 3D wavelet transformation, a breakdown structure for the very large 3D image is applied into sub-blocks by size 16*16*16 which called unit block as shown in figure (7), and then a wavelet transformation is applied

two times along three axes (x, y, z). The used transformation was based on the Haar wavelet transformation whose eight-band filter bank is expressed as follow:

1. clll= (c1 + c2 + c3 + c4 + c5 + c6 + c7 + c8)=8
2. cllh = (c1 + c2 + c3 + c4 - c5 - c6 - c7 - c8)=8
3. clhl = (c1 + c2 - c3 - c4 + c5 + c6 - c7 - c8)=8
4. clhh = (c1 + c2 - c3 - c4 - c5 - c6 + c7 + c8)=8
5. chll = (c1 - c2 + c3 - c4 + c5 - c6 + c7 - c8)=8
6. chlh = (c1 - c2 + c3 - c4 - c5 + c6 - c7 + c8)=8
7. chhl = (c1 - c2 - c3 + c4 + c5 - c6 - c7 + c8)=8
8. chhh = (c1 - c2 - c3 + c4 - c5 + c6 + c7 - c8)=8

Where ci, 0¡i¡9, on the right side are eight coefficients in each 2*2*2 sub-region of a unit block, clll is their average value and the rest on the left side relate to the filtering sequence for example Clhl is the expression of applying a low pass filter, high pass filter and then low pass filter. Finally, keep the coefficients that are greater than a specific threshold and replace the rest by zero so the final result represents the smallest number of non-zero coefficients. Second section, is encoding the wavelet coefficients which necessary information to be stored in a smaller number of bits. Then, wavelets coefficients encoding which explained in [2] where most of the information loosening happened and finally reconstructing voxel values as shown in figure (8) where the voxel values reconstruction started by the first level (0) which has the lowest resolution the using the level 0 we can construct the second level of resolution, but as mentioned in the previous step that there are data loosing. Consequently, the last level reconstructed will not contain the information which contained in the original image because of

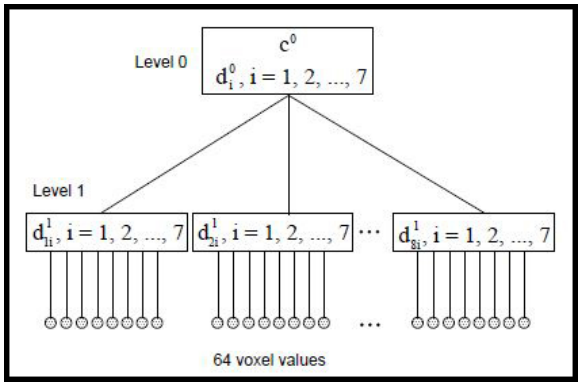

Figure 8: Multi levels representation

data loosening. Actually, maybe this approach is useful for compression and provide many levels of reconstruction with multiple resolutions, but its not convenient for our concern of visualization or processing where we are in need for zero data loosening and the exact values must be reached.

2.2 Processing Large 3D Images

As we decomposed the very large 3D image into sub-blocks and visualize the required region of interest, we have to process this region in order to segment the vascular network. In the paper "Multi-generational analysis and visualization of the vascular tree in 3D micro-CT images" [3] a method was proposed consisting of several processing sections that lead to segment the vascular network.

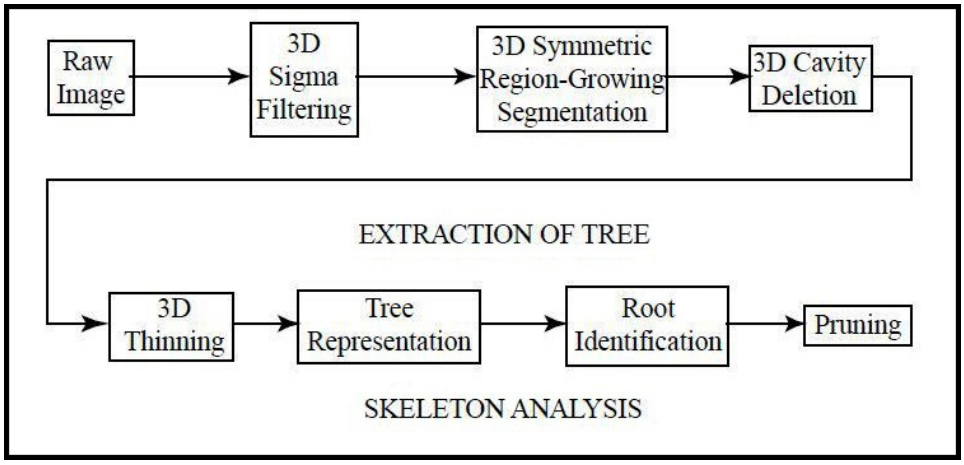

Figure 9: Vascular network segmentation steps

These sections as shown in figure (9) are consisted of: (1) Apply 3D sigma filter which will smooth the image by taking an average over the neighboring pixels including the pixels values that are not preserved from the current pixel by more than the range, which is defined by standard deviation of the pixel values within the neighborhood so, it the number of pixels included in this range is too small then average all the neighboring pixels. For the outliers, average all neighboring pixels excludes the center pixel. So, outliers having a value very different from the surrounding are not included in the average and. Consequently, completely eliminated that will remove the noise from this image, (2) 3D symmetric region-growing segmentation which can initiate the segmentation process anywhere within an image, especially that it can begin to grow a region from the first point it reaches, the

growing process can be performed first and the seed criteria used to identify the final regions of interest can be applied later in order to extract the raw vascular tree, (3) 3D cavity detection which will detect the 3D holes in the previous raw tree then fill these holes in order to get a robust tree, (4) Applying 3D thinning (skeletonization) method to formulate the skeleton structure, (5) tree representation, up to now all the previous processing sections were applied on the image, so this step turn to deal with graph theory to constitute the robust tree structure using a graph data, after building the graph tree, (6) root identification: by user's interaction that will define the root of the tree in order to satisfy the branches allocation and data structure depending on a specific root and finally, (7) pruning: that will correct all of the graph tree structure by removing the unwanted distortion branches, adding another ones as shown in figure (10).

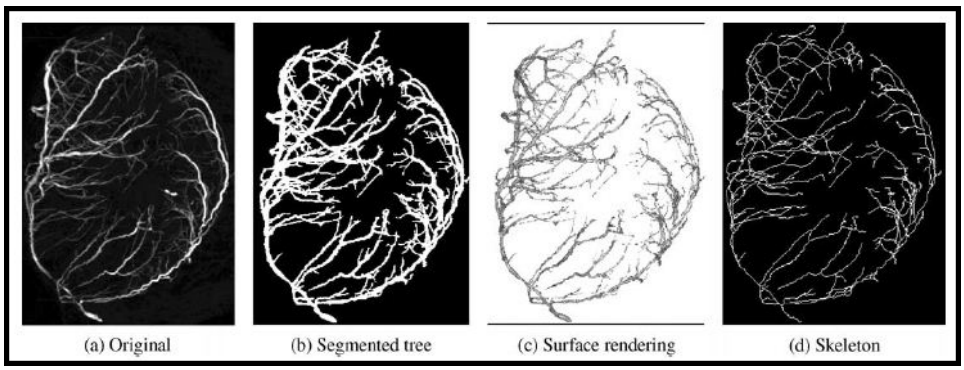

Figure 10: Results of vascular network segmentation

Actually, all the previous processing sections that lead to vascular network segmentation is about basic processing operators in case of filters as sigma filter and cavity detector or thresholding as in region growing segmentation and mathematical morphological dilation operation which used to fill the caps that had detected using cavity in the tree structure. All of these processing operators and morphological operations can be applied simply on normal images and 3D images as well.

But for very large 3D images that are decomposed into blocks, there may be some problems to match the results which are computed in each block separately.

In the paper "Connectivity Analysis in Very Large 3D Micro tomographic Images" [4] a method was proposed to deal with such problems and matching results, they provide a method of labeling connected objects in very large images which are decomposed into sub-blocks, this proposed

method is consisted of three stages:

- The objects are labeled in each block separately.
- Study the interconnections between blocks.
- Relabeling operation is applied according to the interconnections information.

After we get our very large 3D image decomposed into sub-blocks, each block is labeled separately where each block is decomposed into slices and each slice is labels one by one depending on that the voxels is labeled as the minimum label of its 26 neighbors if at least one of them is belong to an object otherwise this voxel is given a new label as shown in figure (11) where on the left side in the plane (n-1) there are two objects labels by 5 and 2 there for the voxel in the plane (n) is labeled according to the minimum value of its neighbors and in case there are no objects in the plane (n-1) consequently the voxel in the plane (n) will take a new label.

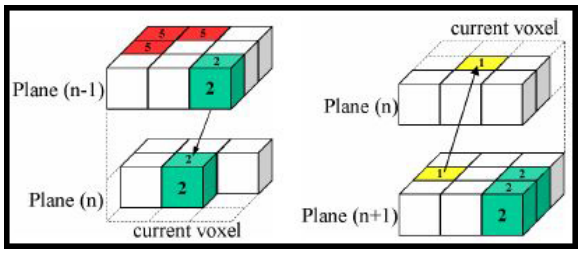

Figure 11: Labeling according to the minimum value

Next stage is to study the interconnections between blocks, the figure (12) shown an example of interconnections between blocks, where sv1 represents the first block and sv2 represents the second one, as a result of previous stage of labeling blocks separately, the objects 1 and 2 in the block sv2 take different labels while they are belong to the same object and the objects 5,1 and 2 all take different labels in the sv1 block while they belong to the same object also and by the way, all of the five objects in both blocks belong to the same object in the large 3D image and here is the importance of studying the interconnections between blocks, to determine the interconnection between sv1 and sv2 objects lets take p1 and p2 planes which shown in figure (13), the voxel 1 in plane p2 is

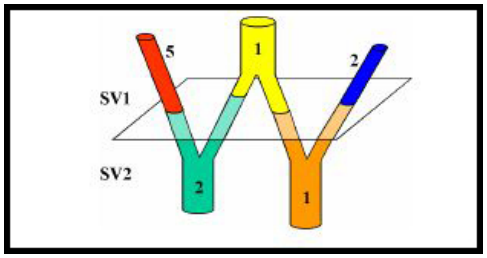

Figure 12: The interconnection between blocks

interconnected with both 1 and 13 objects at plane p1 because there is one minimum label in the voxels neighbors.

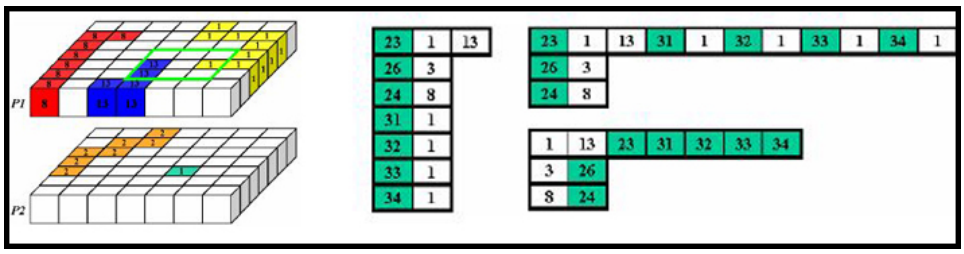

Figure 13: Relabeling the connected components

To analyze this problem, a table of the interconnections between blocks was created which contain the labels that are existed in p2 and followed by labels that are existed in p1 where p2 in the upper plane of second block and p1 is the lowest plane in the first block, so the interconnections information which is extracted here is used to solve the overlapping between these two planes in the last stage of this method which relabeling, where each label of p2 is incremented by the maximum label of p1 as shown in figure (14), so the basic difference between labeling two within the same block and two blocks is that in same block the labeling operation depends on the minimum number of label while it depends on the maximum number of label in two blocks.

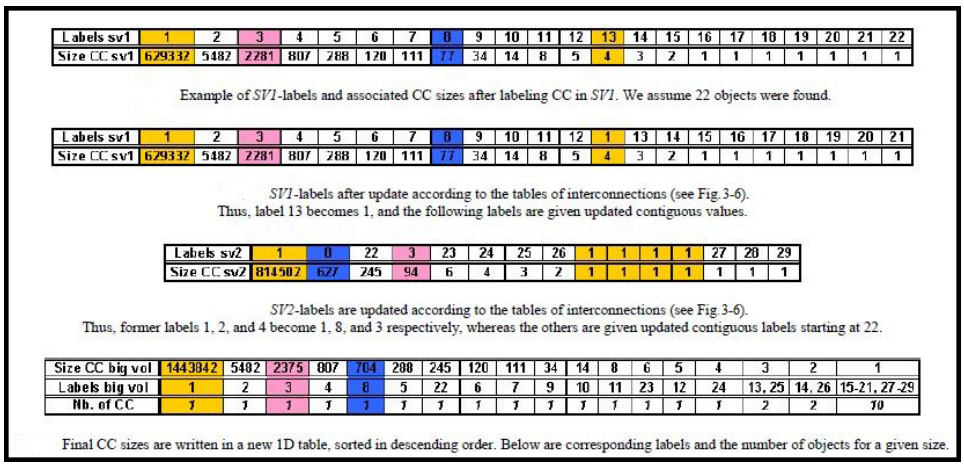

Figure 14: Relabeling according to the maximum value

2.3 Discussion

In our case, we don't have a client-server application since we are working on the same computer, we don't need to zoom into details since we want to visualize the high resolution part only in addition to that there is no limit of disk memory.From the other hand, we want a fast random access to different parts of the original high resolution 3D medical image and visualize it. So the block's decomposition is adequate to our requirements. Then, we are in need to apply the basic processing operators like: Erosion and dilation in addition to thresholding in order to be able to apply some segmentation operations. We need to pay attention because some of these processing operators needs to be studied carefully before being applied on separated blocks especially in the case of overlapping blocks.

3 Contribution

Maybe for the time once this problem looks like it's simple, but actually it's more complex than what it seems, the main goal is to achieve many factors under specific circumstances, we want to:

- Process 3D images by size around 2000 * 2000 * 2000 = 10 GB.
- Work on a standard PC (Not a client-powerful server basis).
- Browse the image interactivity in a MPR visualization mode.
- Have an overview of basic operators as thresholding and mathematical morphology.
- Apply these operators on the original image (in batch mode) without error.
- The final application is to extract the skin, vascular network and bone on the foetus image.

The used tools in application's building are C++ (visual studio compiler) and CImg library (www.cimg.sourceforge.net) that is an open source library for image processing. We choose CImg because of its simplicity, portability not a heavy library. It provides the ability to get deal with various types of images and manage the user interactions with these images.

3.1 Visualization of Large 3D Image

Two versions of the original very large 3D medical image are reconstructed, the first one is the low resolution version and the second one is the blocks version. The user is previewed two views figure (15), the low resolution view and the high resolution one. The low resolution view that is related to the low resolution version, the low resolution version is generated by applying down-sampling on the high resolution original image using down-sampling factor which has a specific value regarding each dimension and the user has to provide this factor as a parameter for the down-sampling operation. When the user chooses a down-sampling factor 4 along X dimension, 3 along Y and 5 along Z that means for X dimension 4 voxel values are taken, the average of their values is computed and considered a new voxel value in the low resolution version, the same thing is happening for the Y and Z dimensions. Right now, we have a new version of the high resolution original image with a smaller data size that could be loaded and visualized on the

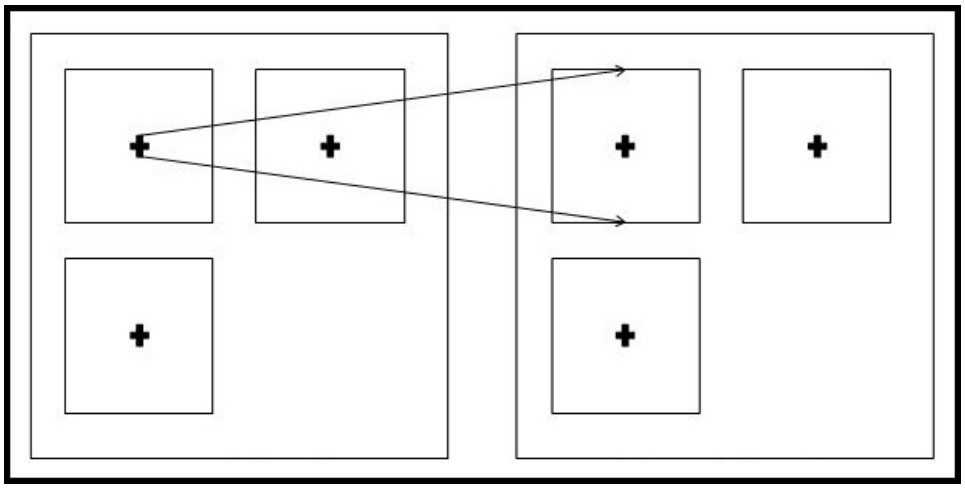

Figure 15: Low resolution and high resolution views

standard computer simply, by the way, this version is in low resolution.

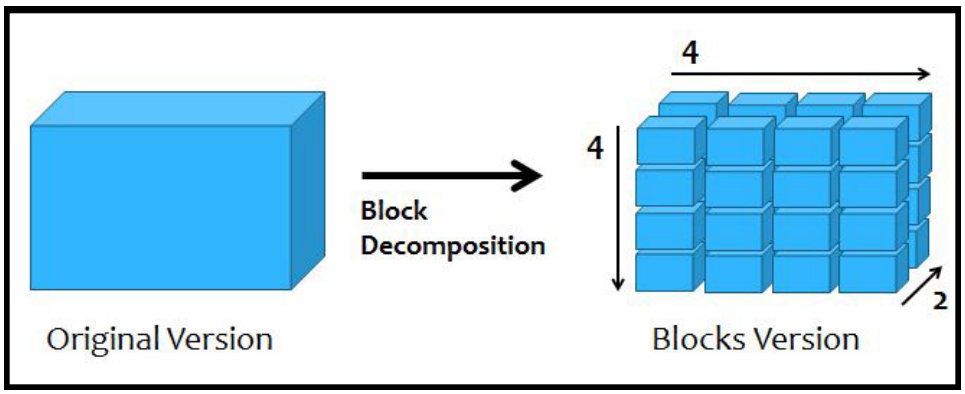

Figure 16: Blocks decomposition by a factor (4,4,2)

Another version as we mentioned previously is reconstructed from the high resolution original image that is blocks version which is generated by applying block-decomposition on the high resolution original image using decomposition factor which has a specific value regarding each dimension and the user has to provide this factor as a parameter for the decomposition operation. In case the user chooses a decomposition factor 4 along X dimension, 4 along Y and 2 along Z. Consequently, the original high resolution image will be decomposed into 4*4*2= 32 high resolution blocks, figure (16). Actually, the linking between the low resolution version and high resolution blocks one is considered the concept of visualization operation. The user is previewed the low resolution view in order to preview the low resolution

version and select his region of interest, as the user select his region, a call back operation is applied to the blocks version in order to determine the related blocks to this region, load it into memory, reconstruct a new high resolution volume from these blocks and preview it to the user in the high resolution view, figure (17).

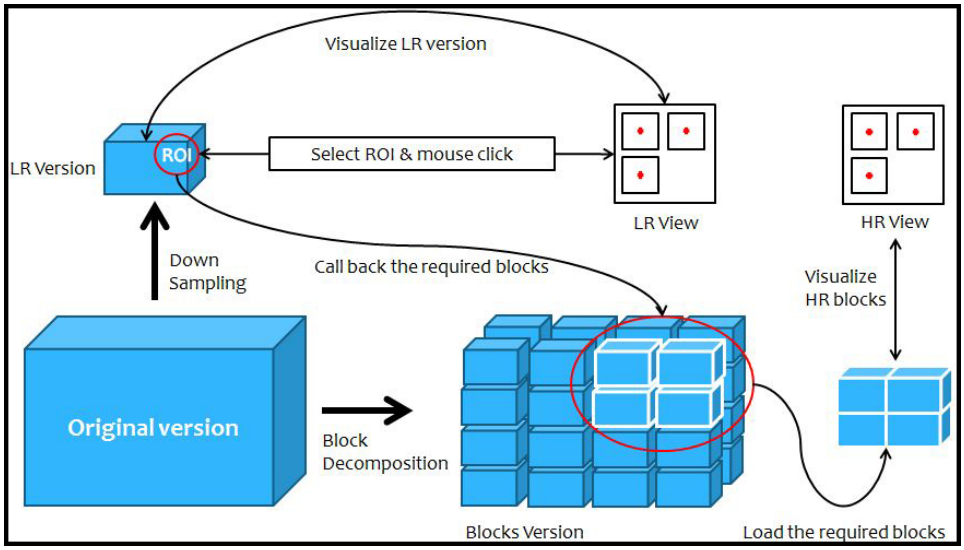

Figure 17: Visualization operation

This figure illustrates the visualization operation, where there is a low resolution view that preview the low resolution version. This low resolution image is generated from the original high image by applying down-sampling. The original image also decomposed into blocks to create the block's version by a factor of (4,4,2). Of course, by user interaction that provides the decomposition parameters. Once the user selects his region on the low resolution view and click the mouse button, as we mentioned previously. The application determines the related blocks, these blocks are loaded into memory, then a new image is reconstructed from these blocks and finally, preview this last image in the high resolution view.

3.2 Processing Large 3D Images

Actually, this processing operation is different from normal image processing operators because we are going to apply these processing operators on separated blocks, so we need to study matching results between these blocks according to the applied operator. In this section, we will preview the avail-

able processing operators then, the processing steps and finally the problems of applying some operators on separated blocks, the proposed solutions and which one is the optimal.

As the user preview the low resolution view and select processing the image, a list of the available processing operators is shown to him that contains: Erosion, dilation and thresholding. The user has to select the desired process among these operators then provide its sufficient parameters. In case of erosion and dilations, he has to provide four parameters that are: width, height and depth of the structuring element in addition to the number of time for this operator to be applied. In case of thresholding, he has to provide the threshold value. When the user provide the required parameters, he can go forward with the processing procedure that is consisted of:

1. Preview the process operators list.
2. Select the required operator and provide its parameters.
3. Apply this process on the low resolution view and preview result.
4. Give the user the control to select his ROI to apply this process on it
5. Provide the sufficient parameters to apply this process operator on the high resolution part and preview the result.
6. Ask the user to add this process to the process list or no.
7. Finally, ask the user to apply the stored process list on the high resolution blocks or no.

Regarding the process list, in order to give the user an advance option and more interaction with the image we innovate the process list that gives the ability to apply multiple processing operators stored with its parameters in text file, so the use can apply the segmentation for example that consists of several processing operators by loading the process list file and run the process without need to apply the process procedure one by one and provide the parameters in each step, in addition to the ability to change the parameters of this process list by modifying the text file directly.

Finally, after building the process list or even loading an existing one from a saved text file, the next stage is to apply these sequenced processes on the separated stored high resolution blocks. For the thresholding operator, we could go along each block and apply thresholding normally, but in case of erosion or dilation there is a difference. Because of existing some objects in the image that may become smaller in erosion or larger in dilation.

3.2.1 Problems of mathematical morphological operators

To have a detailed and more closer view on this matter let's have a look on the figure (18):

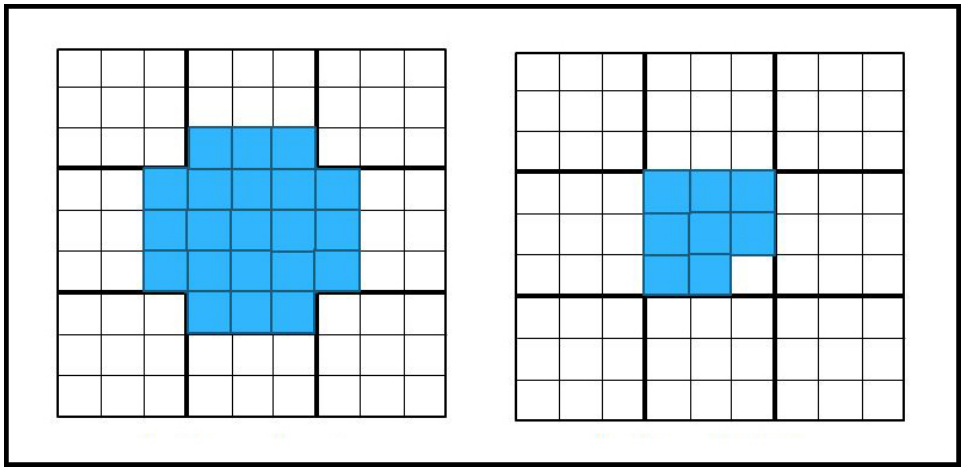

Figure 18: Problems of morphological operators

We have an example of 2D two images by size of 9*9 pixels with these objects in each one, these images is decomposed into nine blocks by applying a decomposition factor of (3,3), we are going to apply erosion on the left image in other words, we want to apply erosion on the separated blocks using this structure element (3*3), by applying the normal erosion function on the separated 2D small images, we will get a result as shown in the figure (19):

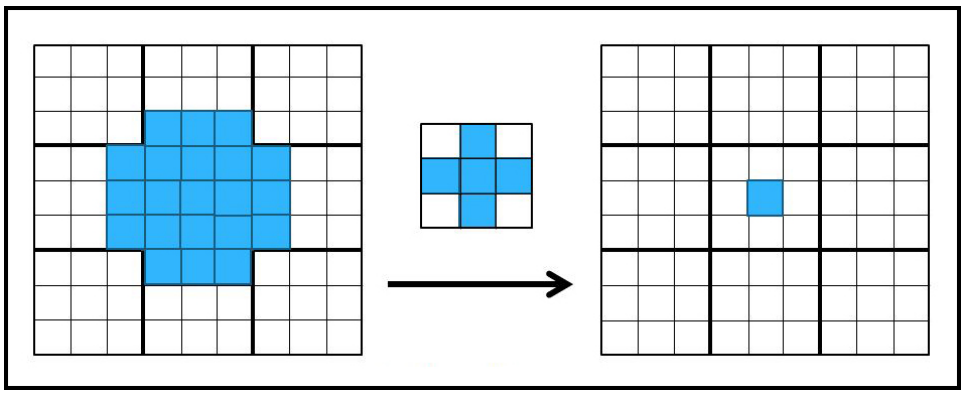

Figure 19: Problem of erosion

By the way, we want to apply dilation on the right image using the same

structure element (3*3), also by applying the normal dilation function on the separated 2D small images, we will get a result as shown in the figure (20):

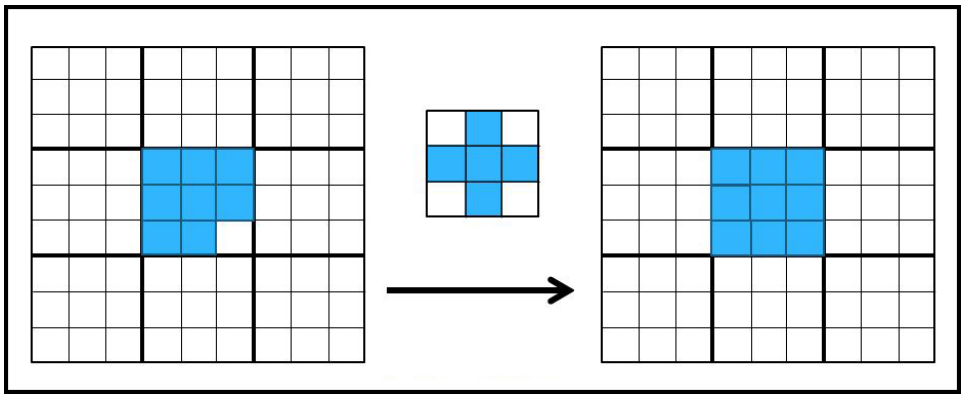

Figure 20: Problem of dilation

Of course, the two previous results of applying erosion and dilation are wrong, where the correct results should be like the figure (21):

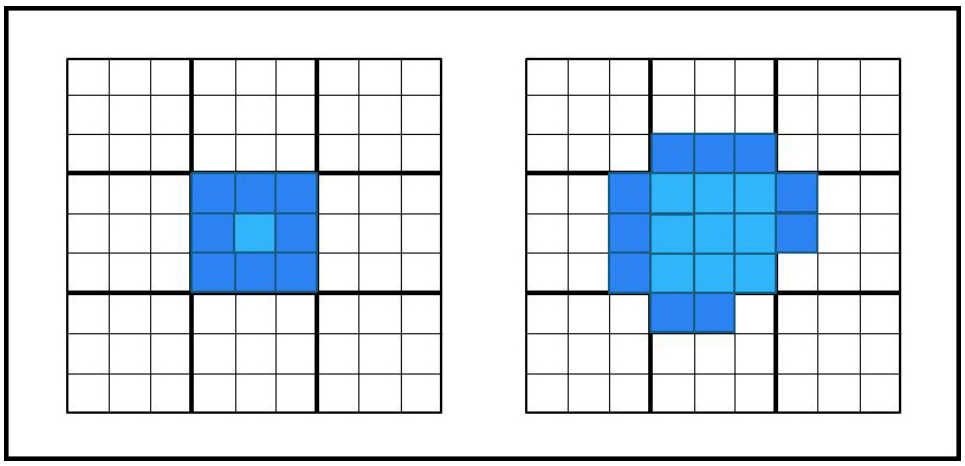

Figure 21: Correct erosion and correct dilation

To get deal with such kind of problem that we called an overlapping problems,three solutions are proposed that differ in the required memory to be applied, complexity and the require time, it's remarkable that all the coming section we will consider a structural element by size 3*3:

1. For each block: increase the block size in all dimensions according the structure element size, then synchronize the increased area after ap-

plying dilation, with the result of applying this operator on all of the block's neighbors one by one, like the figure (22):

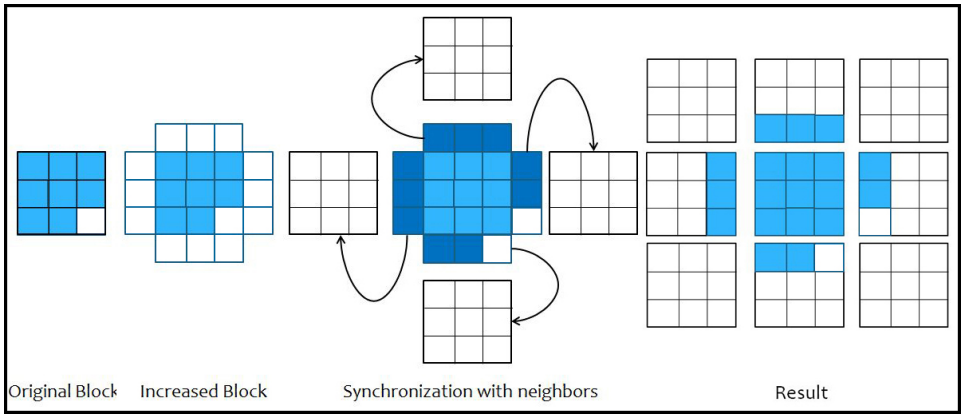

Figure 22: Problem of dilation: First solution

2. For each block: reconstruct a new block from the original block and its neighbors, the new size will be computed from the size of the original block and the size of the structure element, then apply the dilation operator and finally store the result for the considered block and ignore the increasing area. To clarify this issue we take another example, the considered block doesn't have any object, while all of its neighbors have an object in light blue, as shown in the figure below, after applying the dilation, the object area will increased to the considered block that is quite correct as shown in dark blue. Finally, we save the result of considered block only, figure (23):

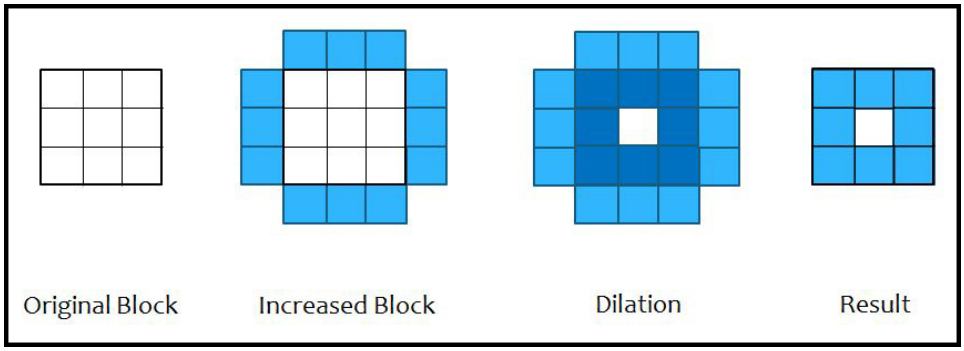

Figure 23: Problem of dilation: Second solution

3. For each block: load the block and all of its neighbors according to

the structure element, then apply the dilation, finally, store result for the considered block only and ignore the rest blocks. To clarify this issue, we take the same example of previous case as shown in figure, when the considered block doesn't contain any object, while all of its neighbors contain an object in light blue, after applying the dilation, an object's area will be also increased in dark blue. Finally, we save the result of considered block only, figure(24):

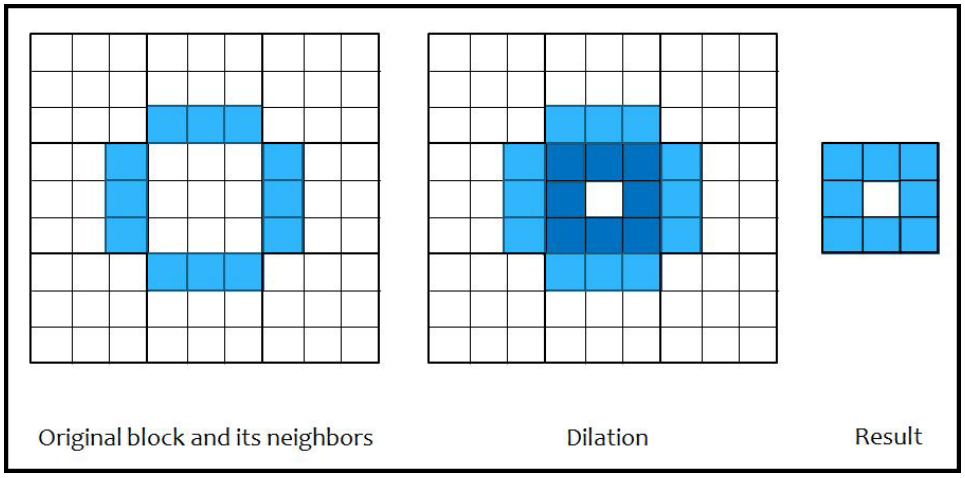

Figure 24: Problem of dilation: Third solution

Here is a table of comparative among these three solution:

	First Case	**Second Case**	**Third Case**
Complexity	Go through a specific part of all block's neighbors to synchronize the results that will increase the complexity. Loading 33 blocks and storing 33 temporary results values	Go through a specific part of all block's neighbors to reconstruct the new block by new size that will increase the complexity. Loading 33 blocks. Loading 33 blocks and storing 9 final results	No complexity increasing. Loading 33 blocks and storing 9 final results
Time	More time to synchronize results of the increased area with the neighbors	More time to reconstruct the new block from the original one and the specific area from the neighbors	Normal time and shorter than two previous solutions
Memory	Require buffer to store the temporary results	Require buffer to store the temporary results	Require memory to load the block and its neighbors

From the previous table it's quite clear that the last solution is better than the others, so we can take it into consideration and go forward.

Regarding the erosion operator, after solving the problem of dilation we can do these steps to apply the erosion:

- Find the image's inverter.
- Apply the dilation operator on the inverted image.
- Go back from the inverted image to the original one.

4 Results

Three tools have built for our purpose, the first one is the Down-sampling application, second is the block decomposition application and finally, the application of visualization and processing.

- Down-sampling application
  This application basically is about preprocessing section, it applies a down-sampling on an existing high resolution image to generate a low resolution one by the user interaction who provides the down-sampling factor.

- Block-decomposition application
  The user has to provide both the source and destination directories, then the application check the content of source directory and provide dimensions size of the high resolution, then ask the user to provide the decomposition parameters that are the decomposition factor along each dimension, then the application creates a number of blocks directories depending on decomposition factors, then go along the high resolution slices and decomposes it into blocks slices and save each part in its place among the blocks directories, then, the application goes along each block directory and generate a (.hdr/.img) file from the blocks slices that are in (.tiff) format and saves it in the same directory. Finally, the application creates a configuration file that contains information about the decomposed image that are decomposition factor and the block size.

- Application of visualization and processing
  The basic application used by the user to preview the very large 3D medical image, once the user run this application, it asks the user to provide the directories of both low resolution version and block's one, then it loads the low resolution version of the image in the low resolution view, read the configuration file of the block's version and preview it to user in addition to the list of available options to interact with the image, figure (25).

  This list consists of the following options: (1) Click : Visualize your ROI. (2) Space : Apply new process. (3) C : Change the intensity. (4) L : Load a process list. (5) F : Finish processing. (6) E : Exit. We are going to review them in more details:

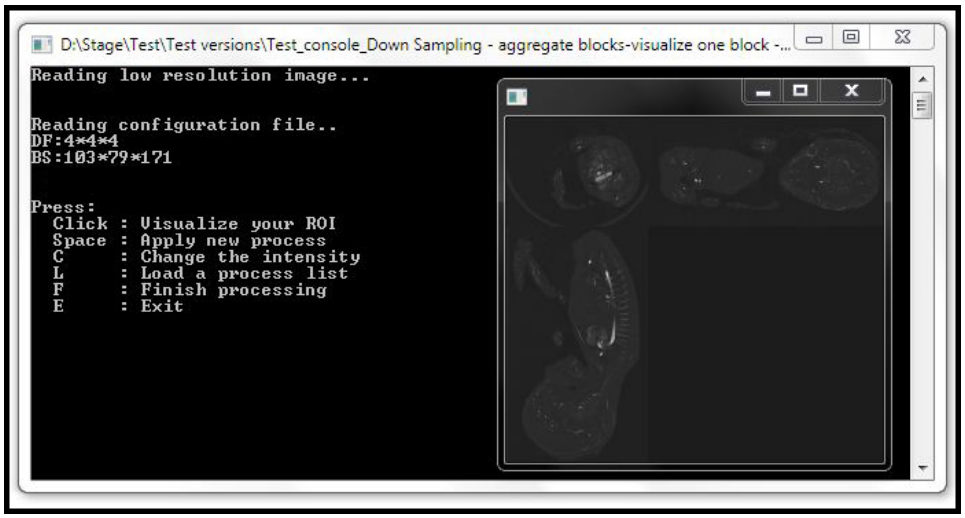

Figure 25: Low resolution version and the list of available options

1. Click : Visualize your ROI
   The user continues previewing the 3D low resolution version in the low resolution view, as he determine his region of interest and he want to visualize it in high resolution he has to click the left mouse button, the application in this case will execute the following steps:
   - Take the coordinates of this region in the low resolution image.
   - Compute the coordinates in the high resolution.
   - Determine the required blocks only and load them into memory.
   - Reconstruct a new 3D high resolution part from these blocks
   - Finally, preview this last 3D high resolution part to the user in the high resolution view, figure (26).

   When the user closes the high resolution view, the application asks him whether he wants to save this high resolution part or no. This option gives users the possibility to save his high resolution region of interest and preview it later easily.

2. Space : Apply new process
   This option gives users the ability to apply process operators on both low resolution image and high resolution part, as the user select this option by press keyboard's space button, the applica-

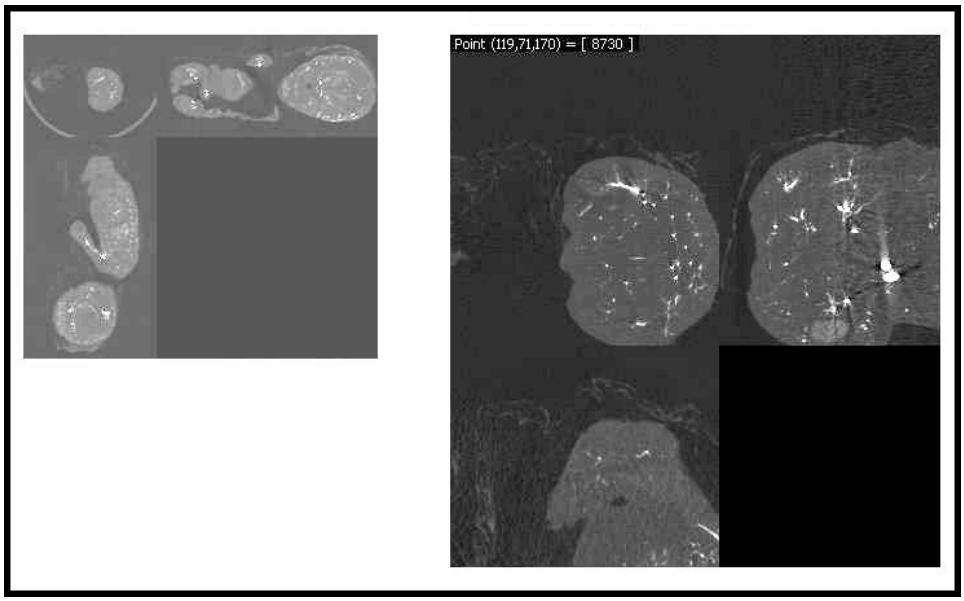

Figure 26: Low resolution version and it's related high resolution part

tion will list the available processes that are: Erosion, dilation and thresholding. Actually, applying any one of these operators is divided into two stages, the first one is the applying on the low resolution version and second one is the applying on a selected high resolution part. Regarding the first stage, as the user select a process, the application will ask him to provide its parameters as mentioned in the previous section then, it will apply this process on the low resolution version and previous the result, like the figure (27) that show the result of applying dilation by this parameters (2,2,2,2).

When the user close this result, the application will ask him whether he wants to keep this result of the low resolution version or no, and it will ask the user also whether he wants to apply this process for a selected high resolution part or no. Actually, these two options allow the user to apply all of his process's operators on the low resolution version and preview the results without reaching any high resolution block. When the user selects to keep changing on the low resolution version and select to apply it on a high resolution part, we will move to the second stage of applying the process on a high resolution part, where the application asks the user to select his region of interest and provide the new parameters for this process's operator to be applied because

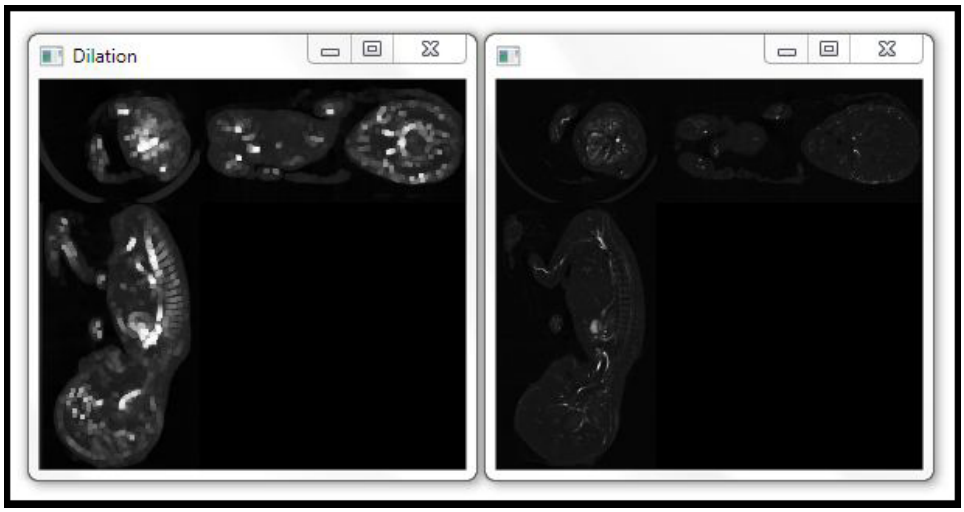

Figure 27: Applying dilation on the low resolution version by parameters (2,2,2,2)

these parameters are changed in case of applying the same process on the low resolution version and the high resolution blocks, for example, if want to apply an erosion by a structure element of size (2,2,2) on a low resolution image that is generated from the high resolution image by a down-sampling factor (4,5,6)then, the erosion on the high resolution blocks should be applied using a structure element of size (4*2,5*2,6*2). Once the user provide the required parameters, the application will execute the following steps:

- Take the coordinates of this region in the low resolution image.
- Compute the coordinates in the high resolution.
- Determine the required blocks only and load them into memory.
- Reconstruct a new 3D high resolution part of these blocks
- apply the selected process's operator by the stored parameters
- Finally, preview the result of applying the selected process's operator on this last 3D high resolution part to the user in the high resolution view.

Here is an example of applying the dilation operator on the low resolution version by parameters (2,2,2,2) and on a selected high

resolution part by parameters (4,4,4,2) in the figure (28).

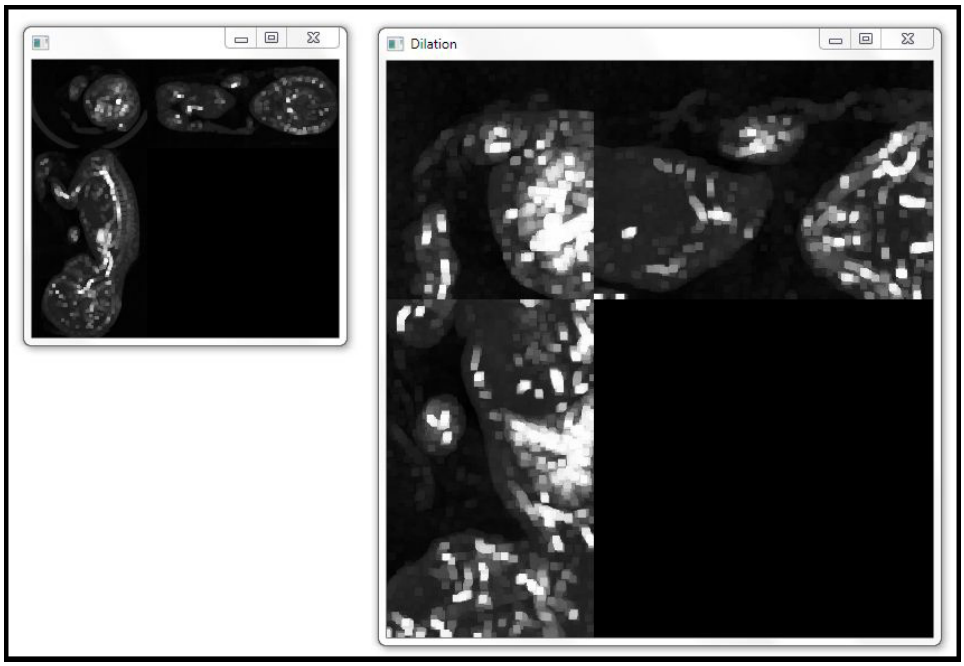

Figure 28: Applying dilation on the high resolution part by parameters (4,4,4,2)

When the user closes the high resolution view, the application will ask him whether he wants to add this process operator to the process list or not, then he will ask the user whether he wants to save this view or no, also this option gives users the possibility to save his high resolution processed region of interest and preview it later easily and finally, the application will go back to the main menu.

3. C : Change the intensity
   The default intensity scale is (0-255), by this option, the user can change this range to have a clearer visualization, as he selects this option, the application asks him to provide both the new minimum and maximum range values. Finally, the application will apply the new intensity range and preview the result, like the figure (29).

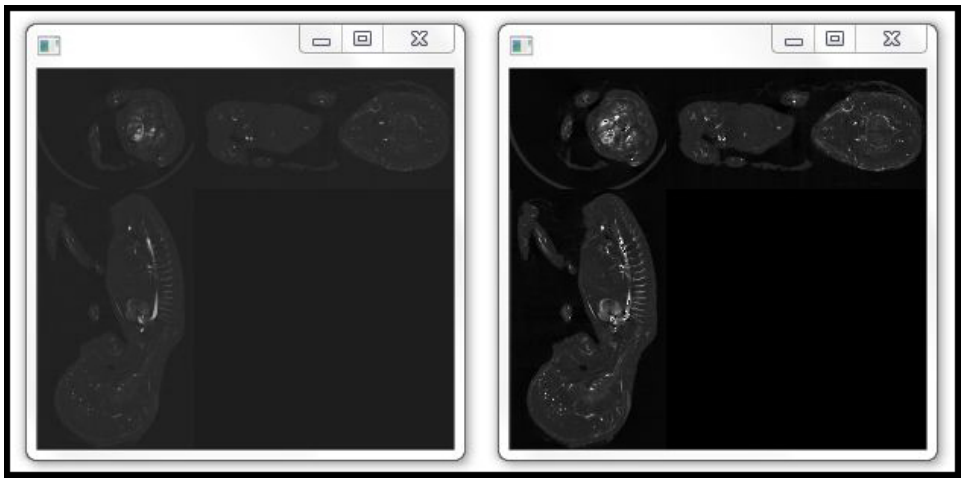

Figure 29: Change the intensity range from 0-255 to 0-511

4. L : Load a process list
   This option allow the user to load an existing process list file from the hard drive to be applied to the high resolution blocks. the application will ask the user to provide both the directory and the name of the process list file, then the application will load this file then, preview the content of this list and the parameters of each process's operator.
5. F : Finish processing
   In case of existing a process operator or more in the process list and the user select this option, the application will go along each process on this list and ask users to provide its parameters and as the user provides all the required parameters, the application will apply this list of the high resolution blocks.
6. E : Exit
   Gives users the possibility to exit the application at any point of time during the visualization or processing sections.

4.1 Discussion

Choosing parameters plays such an essential manner in the visualization and processing operations, especially the parameters of decomposition into blocks. For a better understanding of this point let's take the following example on 2D images.

We have an image with the size of (800*800)pixel, we applied a block's

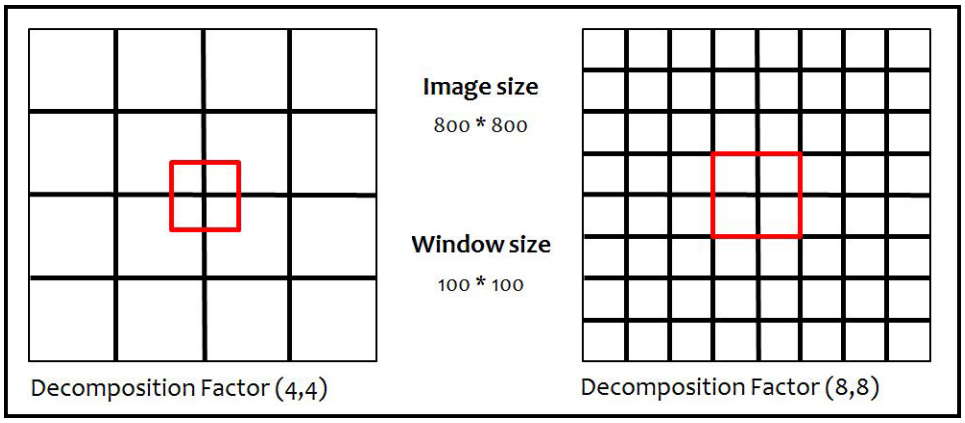

Figure 30: Compare the two decomposition factors

decomposition and visualize our region of interest that is in red color in two situations of different decomposition factors, the figure (30):

1. First situation: decomposition factor (4,4)
   We will get 16 blocks by a block size of 200*200=40000 pixel and for the visualization of our region we need to load four blocks by a size 200*200=40000 pixel of each one so we are going to load 4*(200*200)=160000 pixel into memory.

2. Second situation: decomposition factor (8,8)
   We will get 64 blocks by a block size of 100*100=10000 pixel and for the visualization of our region we need to load four blocks by a size 100*100=10000 pixel of each one so we are going to load 4*(100*100)=40000 pixel only into memory.

It's quite clear that the selected decomposition factor value in the second situation is better than the first one, if we draw the histogram that represent the relation between the required memory and block size in previous two situations we will get the figure (31).

We notice, decreasing the decomposition factor will increase the block size. Increasing the block size will increase the required memory to load the required blocks to visualize the selected area. When the block size is 40000 pixels that is a result of (4,4) decomposition factor, then we need a memory to load 160000 pixels, while the second situation we need a memory to load only 40000 pixels as a block size is 10000 pixels that is a result of (8,8) decomposition factor.
From the other hand, we need to pay attention, because increasing the decomposition factor into high values that lead to very small block size may

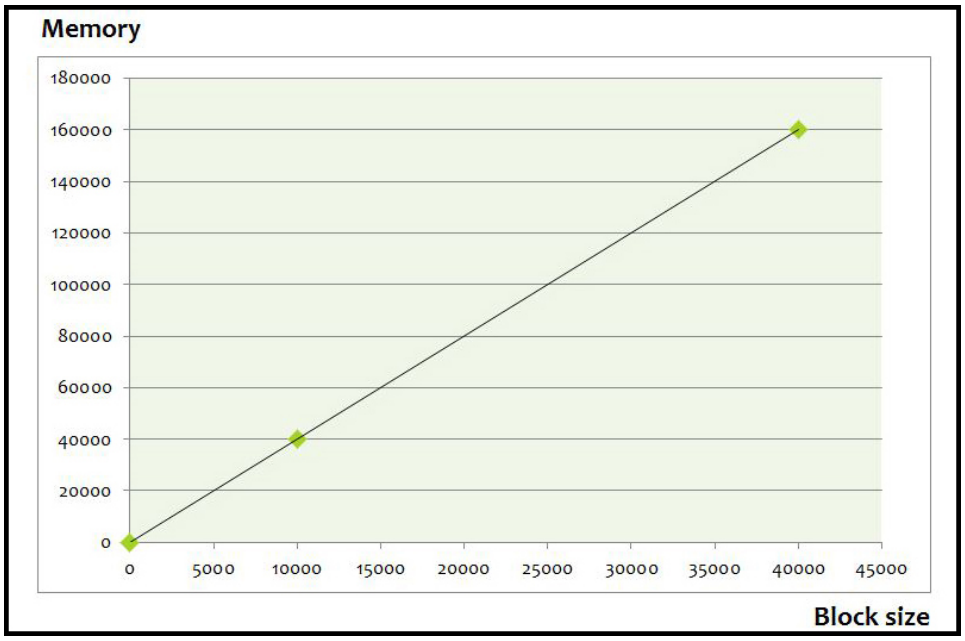

Figure 31: Relation between the required memory and the block size

require so many times to load a lot of small blocks into memory, the time required to load four larger blocks into memory less than the time required to load nine smaller blocks or even sixteen more smaller blocks.

5 Future Work

In the terms of development the applied applications of this topic we have to ask about some points like: Did the visualization and user's interaction are suitable? Did we get a quite satisfactory result? Did the processing operators are enough or we are in need for more filters?

Some filters could be added to the process's operators like Gaussian filter or even blurring and sharpening filters in order to apply more complex processing than just a segmentation or even an advanced segmentation with clearer results.

6 References

[1] Zihong Fan, Antonio Ortega, Optimization of Overlapped Tiling for Efficient 3D Image Retrieval, dcc, pp.494-503, 2010 Data Compression Conference, 2010

[2] Ihm, I., Park, S.: Wavelet-based 3D compression scheme for very large volume data. In: Graphics Interface .98, 107-116, 1998.

[3] S.Wan, E. Ritman,W. Higgins, Multi-generational analysis and visualization of the vascular tree in 3D micro-CT images, Comput. Biol. Med. 32 (2002) 5571.

[4] L.Apostol and F.Peyrin, Connectivity analysis in very large 3D microtomographic images. Nuclear Science Symposium Conference Record, 2004 IEEE

[5] S.Guthe, M.Wand, J.Gonser and W.Straßer, Interactive Rendering of Large Volume Data Sets, WSI/GRIS, University of Tübingen, Visualization 2002 conference proceedings.

[6] H. Peng, Z. Ruan, J. H Simpson and E. W Myers, V3D enables realtime 3D visualization and quantitive analysis of large-scale biological image data sets, volume 28, number 4, April 2010, nature biotechnology.

[7] K. Andriole, J. Wolfe, R. Khorasani, S. Treves, D. Getty, F. Jacobson, M. Steigner, J. Pan, A. Sitek and S. Seltzer, Optimizing analysis, visualization and navigation of large data sets, volume 259, November2 2 May 2011, radiology rsna.

[8] F. Long, J. Zhou and H. Peng, Visualization and analysis of 3D microscopic images, volume 8, issue 6, e1002519, June 2012, plos computational biology.

[9] Zsolt L. Husz, Thomas P. Perry, Bill Hill, and Richard A. Baldock: A tiled 0n-the-fly sectioning server for 3D volumetric atlases, volume 5875/2009, 924-933, advances in visual computing.

[10] G. Captier, G. Subsol, R. Lebrun, F. Meyer, J.M. Gory, F. Canovas. "Dissection ftale virtuelle par micro tomodensitomtrie". 93e Congrs de l'Association des Morphologistes, Rouen (France), March 2011. Abstract published in Morphologie, 95, p. 102103, 2011.