TP Imagerie 3D no 3 (3 heures)

- **Le TP est noté : le compte-rendu doit être déposé** sous **forme électronique à sur le moodle** avant **le mercredi 16 octobre 2019** (minuit)
- Tout compte-rendu envoyé sous une mauvaise ou hors-délai sera sanctionné par un 0.
- Le compte-rendu doit inclure **la date**, **vos noms** et être composé d'1 à 2 pages de texte **décrivant l'algorithme** ainsi que les commandes lancées **AVEC quelques captures d'écran** pour évaluer le résultat **ET la source** de votre programme ou la liste des fonctions macros utilisées.
- Le tout doit être sous la forme d'un **unique fichier pdf**.
- Le TP peut se faire seul ou en binôme
- La participation active pendant le TP pourra aussi être prise en compte.
- **Tout plagiat sera lourdement sanctionné**.

  - **Master 2 Informatique** : obligatoirement C/C++ avec la bibliothèque CImg (http://cimg.sourceforge.net/). Voir Annexe 2.
  - **Autres Masters** : au choix soit en C/C++ avec la bibliothèque CImg, soit en utilisant des macros du logiciel Fiji (http://fiji.sc/). Voir Annexe 3.

1. Etudier l'image liver. Essayer de trouver empiriquement le seuil qui permet de segmenter au mieux le(s) réseau(x) vasculaire(s). Visualiser le résultat. Que se passe-t-il si on prend un seuil supérieur ou inférieur ?

2. En vous inspirant du chapitre III.a. de l'article[1] qui est associé à ce TP, décrivez un algorithme de segmentation du réseau vasculaire qui n'est fondé que sur une interaction minimale avec le praticien (un unique clic dans l'image).

3. Programmer tout ou partie de cet algorithme en C/C++ en utilisant CImg (obligatoire pour les Master 2 Informatique) ou à l'aide un autre langage ou des macros Fiji (au choix pour les autres Masters 2). On trouvera en Annexe 2 quelques fonctions Cimg qui pourront être utiles. Il est indispensable de construire la courbe Number of segmented voxels = f (Threshold) et de la visualiser à l'écran. Mais, on pourra ne pas programmer la sélection automatique du seuil à partir de la courbe.

4. Tester le programme sur les 2 images (*liver_07* et *BREBIX*) en cliquant dans la veine porte à l'intérieur du foie (voir 9 dans la planche anatomique de l'annexe 1). Déterminer graphiquement le seuil et visualiser le résultat en rendu surfacique avec Fiji (voir annexe 3).

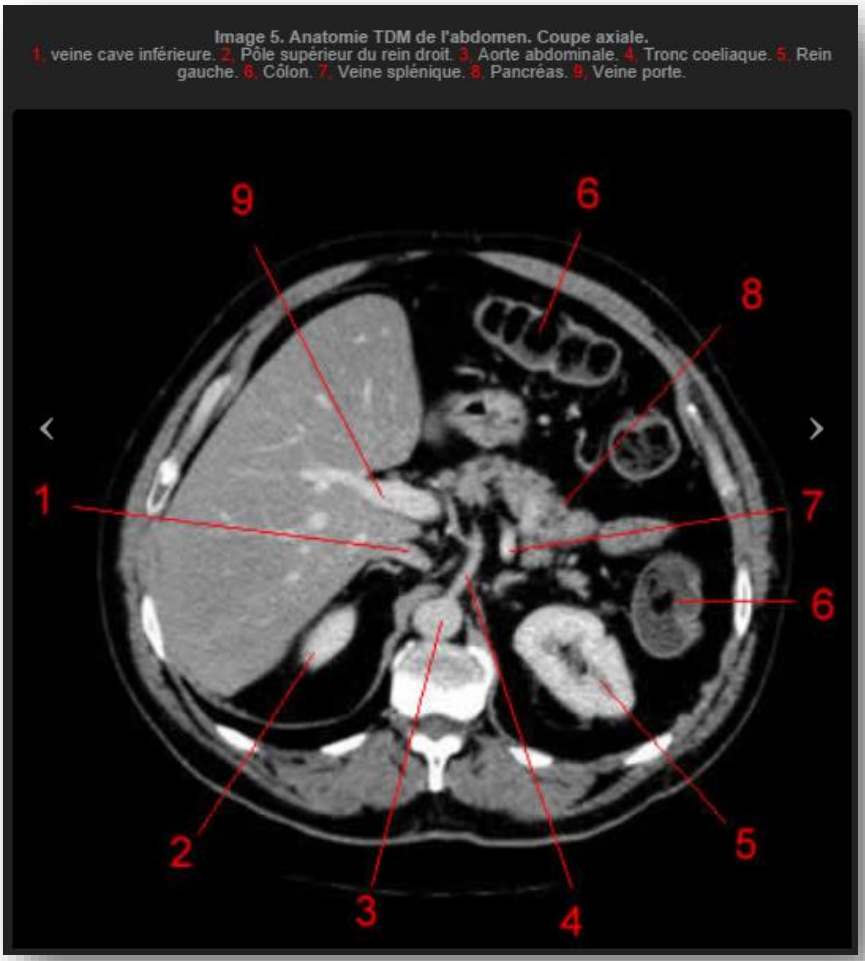
5.1. Pour les Master 2 Informatique : comment pourrait-on programmer une fonction plus optimisée que draw_fill pour construire la courbe ? Décrivez en quelques lignes votre choix algorithmique et donnez la complexité.

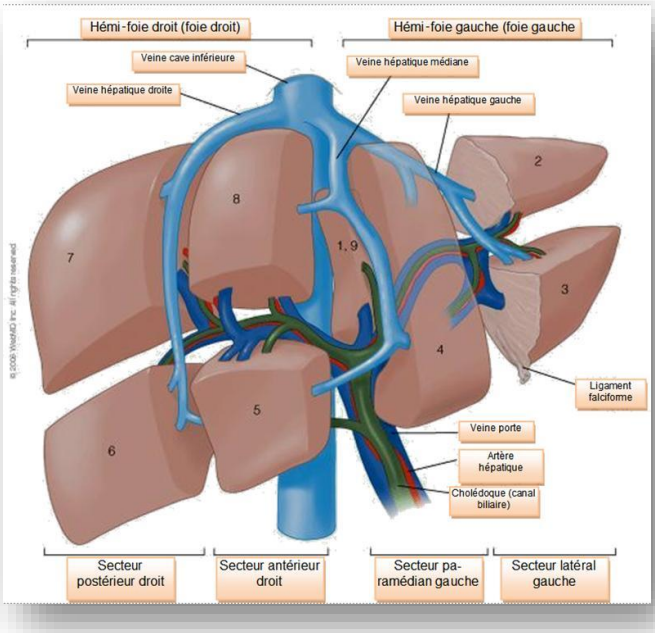
5.2. Pour les autres Master, utiliser les fonctions de squelettisation 3D (skeletonization 3D) de Fiji pour modéliser et analyser le réseau vasculaire. Décrivez et justifiez vos manipulations et les résultats obtenus.

---

[1] *D. Selle, B. Preim, A. Schenk and H. O. Peitgen, "Analysis of vasculature for liver surgical planning," in IEEE Transactions on Medical Imaging, vol. 21, no. 11, pp. 1344-1357, Nov. 2002. doi: 10.1109/TMI.2002.801166*

Annexe 1 : un peu d'anatomie…





Annexe 2 : quelques fonctions Cimg utiles

http://cimg.eu/reference/structcimg__library_1_1CImg.html

CImg<T>& **draw_fill** (const int x, const int y, const int z, const tc *const color, const float opacity, CImg< t > & region, const float sigma = 0)
Draw filled 3d region with the flood fill algorithm.

Parameters

x	X-coordinate of the starting point of the region to fill.
y	Y-coordinate of the starting point of the region to fill.
z	Z-coordinate of the starting point of the region to fill.
color	Pointer to **spectrum()** consecutive values, defining the drawing color.
[out] **region**	Image that will contain the mask of the filled region mask, as an output.
sigma	Tolerance concerning neighborhood values.
opacity	Opacity of the drawing.
is_high_connexity	Tells if 8-connexity must be used (only for 2d images).

Returns region is initialized with the binary mask of the filled region.

Annexe 3 : quelques macros Fiji utiles

http://rsb.info.nih.gov/ij/developer/macro/functions.html

- **close()**
  Closes the active image. This function has the advantage of not closing the "Log" or "Results" window when you meant to close the active image. Use *run("Close")* to close non-image windows.
- **setSlice(n)**
  Displays the nth slice of the active stack. Does nothing if the active image is not a stack.
- **makePoint(x, y)**
  Creates a point selection at the specified location.
- **setZCoordinate(z)**
  Sets the Z coordinate used by *getPixel()*, *setPixel()* and *changeValues()*. The argument must be in the range 0 to n-1, where n is the number of images in the stack.
- **getPixel(x, y)**
  Returns the value of the pixel at *(x,y)*.
- **getVoxelSize(width, height, depth, unit)**
  Returns the voxel size and unit of length ("pixel", "mm", etc.) of the current image or stack.
- **getResult("Column", row)**
  Returns a measurement from the ImageJ results table or NaN if the specified column is not found. The first argument specifies a column in the table. It must be a "Results" window column label, such as "Area", "Mean" or "Circ.". The second argument specifies the row, where 0<=*row*<nResults. *nResults* is a predefined variable that contains the current measurement count. (Actually, it's a built-in function with the "()" optional.) Omit the second argument and the row defaults to nResults-1 (the last row in the results table).
- **getNumber("prompt", defaultValue)**
  Displays a dialog box and returns the number entered by the user. The first argument is the prompting message and the second is the value initially displayed in the dialog. Exits the macro if the user clicks on "Cancel" in the dialog. Returns defaultValue if the user enters an invalid number.
- **setBatchMode(arg)**
  Controls whether images are visible or hidden during macro execution. If *arg* is 'true', the interpreter enters batch mode and newly opened images are not displayed. If *arg* is 'false', exits batch mode and

disposes of all hidden images except for the active image, which is displayed in a window. The interpreter also exits batch mode when the macro terminates, disposing of all hidden images.

- **Plot.create("Title", "X-axis Label", "Y-axis Label", xValues, yValues)**
  Generates a plot using the specified title, axis labels and X and Y coordinate arrays. If only one array is specified it is assumed to contain the Y values and a 0..n-1 sequence is used as the X values. It is also permissible to specify no arrays and use *Plot.setLimits()* and *Plot.add()* to generate the plot. Use *Plot.show()* to display the plot in a window, or it will be displayed automatically when the macro exits.
- **getCursorLoc(x, y, z, modifiers)**
  Returns the cursor location in pixels and the mouse event modifier flags. The coordinate is zero for 2D images. For stacks, it is one less than the slice number.
  Modifiers = rightButton=4
  Modifiers = leftButton=16

Annexe 4 : surface rendering

Fiji: Plugins 3DViewer

Edit/Display as/Surface. Ne pas oublier que resampling factor=paramètre de rééchantillonnage.
1 : résolution originale
2 : l'image est divisée par 2 dans toutes les directions

Le maillage apparaît noir ! Edit/Change color permettra de mieux le voir.

File/Export Surfaces (dans un format standard .obj ou .stl) permet de le visualiser avec MeshLab par exemple.

Attention, les normales peuvent être inversées ce qui fait que le maillage apparaît sombre ! Dans MeshLab, on peut utiliser Filters/Normals, Curvatures & Orientation/Invert Faces Orientation.

Fig. 1. 2-D slice of a CT volume dataset. The applied contrast agent provides a high vessel-to-tissue contrast and reveals the highlighted portal vein and some branches of the hepatic vein. The two dark spots inside the liver represent liver metastasis. (Dataset provided by Prof. Galanski, Medical School Hannover.)

II. Medical Background

Because of the complex vascular anatomy of the liver, surgical interventions are challenging. Four different vessel systems supply and drain the liver: the portal vein, hepatic vein, hepatic artery, and bilary ducts. A successful operation requires enough remaining liver tissue supplied by all four vessel systems. Since the portal vein, the hepatic artery, and the bile ducts parallel, the portal vein is regarded as the leading structure for these three vessel systems.

For a surgeon, it is difficult to mentally construct the 3-D structure of vessel systems based on planar slices of radiological data (cf. Fig. 1) and to estimate which part of a vessel system would be damaged as a consequence of a surgical intervention [22].

III. Methods

A. Fast and Robust Vessel Segmentation

The segmentation of the intrahepatic vessels is a prerequisite for a subsequent geometrical and structural analysis. In a preprocessing step, filter functions for noise reduction (Gaussian, median filter) and for background compensation (Laplace-like filters) are applied to the CT data [42]. For background compensation, the size of the filter kernel is chosen such that it is larger than the thickest vessel inside the liver (default is 15×15). The application of this filter is restricted to an interval which is defined such that the lower interval bound roughly corresponds to the gray value of the liver parenchyma and the upper bound corresponds to the brightest values inside the liver.

As a result, intrahepatic vessels can be identified and delineated by using a threshold-based region-growing method. Usually, region-growing segmentation must be repeated with modified thresholds until an appropriate result is found. To accelerate this procedure, we refined the procedure to automatically suggest a threshold.

Initially, a seed voxel of the portal vein close to its entrance into the liver is selected interactively. Starting with this seed voxel, the region-oriented segmentation algorithm iteratively accumulates the 26 adjacent voxels with an intensity equal to or greater than the intensity θ_{beg} of the seed voxel and keeps them in a list $L(\theta_{\text{beg}})$. Using $L(\theta_{\text{beg}})$ as new seed voxels, all adjacent voxels with intensities greater than or equal to $\theta_{\text{beg}} - 1$ are collected in a list $L(\theta_{\text{beg}} - 1)$. The threshold is further decreased until a given threshold θ_{end} is reached which definitively creates only voxels $L(\theta_{\text{end}})$ outside the vessel systems.

The generation of the voxel lists is performed efficiently because voxel lists for the segmentation with threshold $\theta - 1$ have already been constructed when using the threshold θ. In total, some 100 lists are generated which takes approximately 3–5 s on modern PC hardware for high-resolution CT datasets (512×512 matrix with slice distance 3 mm).

The automatic threshold selection is based on the observation that the number of voxels $N(\theta)$ is approximately linear decreasing for $\theta = \theta_{\text{opt}} \cdots \theta_{\text{beg}}$ (Fig. 2). At θ_{opt}, the slope changes considerably because many voxels belonging to the liver tissue are collected for thresholds below θ_{opt}. Thus, a suggestion for θ_{opt} can be found by calculating an optimal fit of two straight lines for $N(\theta)$. For this purpose, the two characteristic parts of the curve are approximated by two regression lines. The points $(\theta_{\text{beg}}, |N(\theta_{\text{beg}})|) \cdots (j, |N(j)|)$, respectively, $(j+1, |N(j+1)|) \cdots (\theta_{\text{end}}, |N(\theta_{\text{end}})|)$ are employed to calculate the correlation coefficients for both lines. j is chosen such that the sum of the two correlation coefficients is maximal.

We found that the position of the crossing of both regression lines yields a good suggestion for θ_{opt} in most cases. If the suggested threshold is not satisfying, it may be changed interactively. On the basis of the generated voxel lists $L(\theta_{\text{beg}}) \cdots L(\theta_{\text{end}})$, the vessel system specified by any threshold θ can be displayed very fast by simply drawing all precalculated voxels from the lists $L(\theta_{\text{beg}}) \cdots L(\theta)$.

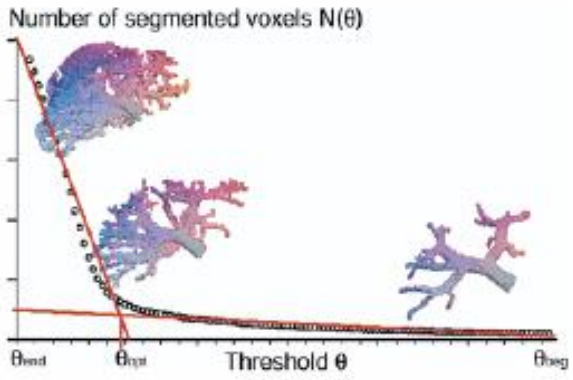


Fig. 2. Estimation of an optimal threshold for vessel segmentation. (©Springer 2000, originally published in [43], reprinted with permission.)