Investigation of feature recognition in clinical CT images as a step towards adaptive radiotherapy : Progress Report

Louis Jaugey, Alix Moawad

4 décembre 2019

1 Aims and objectives of the project

Radiotherapy treatment is based on the destruction of cancerous cells by ionizing radiation. During this process, sane surrounding cells could also be affected by ionizing beam and it is therefore paramount to minimize its effect on those cells. Recent technologies have been devised to modulate the beam intensity in space, allowing a treatment highly adapted to a specific tumour. However, in order to use these techniques to their full potential, the shape and position of the tumour must be precisely known. The process of contouring the cancerous region can be long. The aim of this report is therefore to automate this process on 3-dimensional CT-scans.

2 Progress of the project

Data visualization A bank of 2D images representing slices of a 3D scan were at our disposal. The first step was to familiarize ourselves with the Python programming language and the specific libraries used for image treatment. We begun by displaying the axial, sagittal and coronal views shown on Fig. 1.

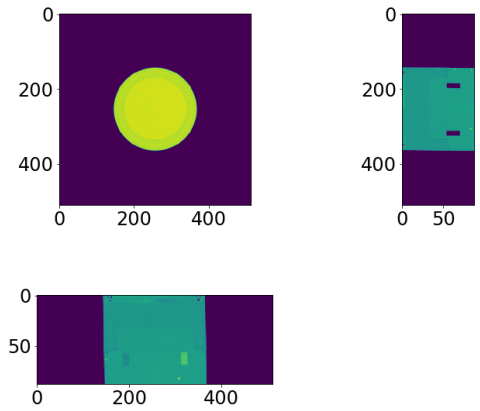


Figure 1 – Axial, sagittal and coronal views of a 3D scan

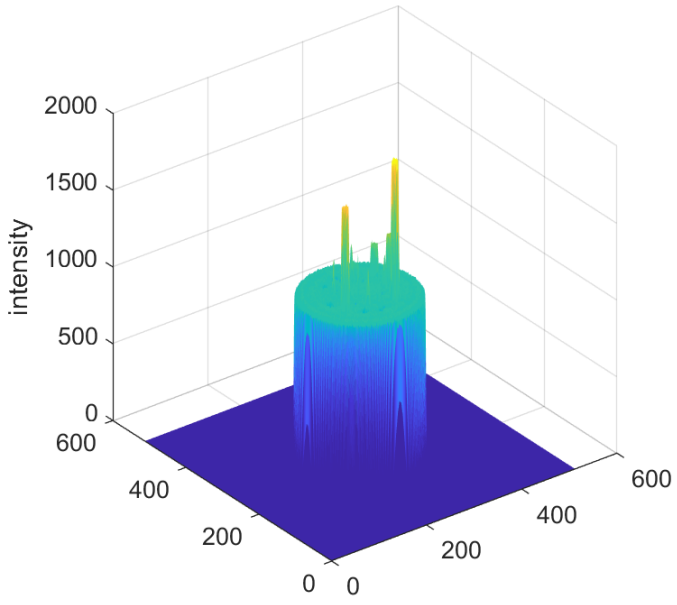
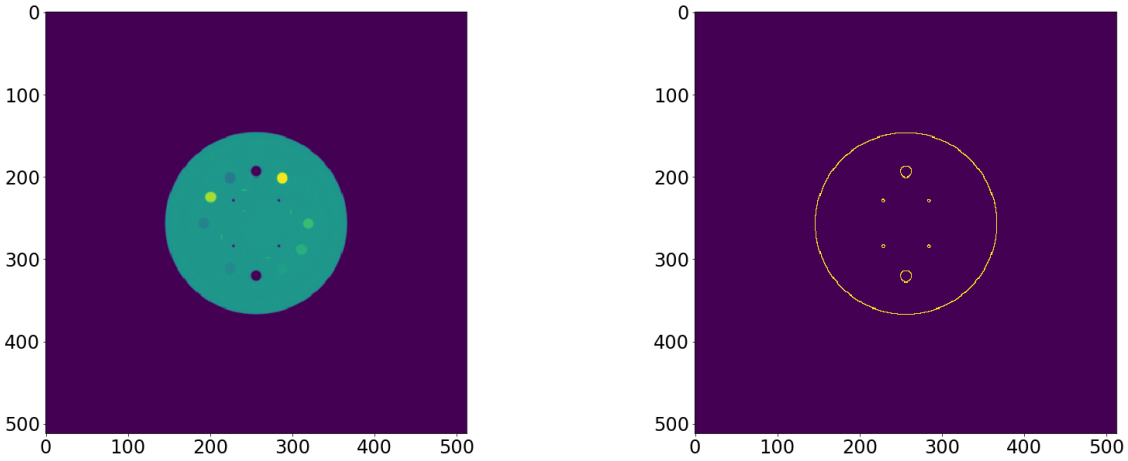


Figure 2 – Surface plot of the intensity of slice 60

Threshold method The goal of the first edge detector algorithm was to be able to determine the shape of the phantom's outer boundary. The method used for this detection is the simplest developed in this project. As discussed below, it does have a lot of limitations.

Fig.2 shows the base image with the intensity represented on the z axis. It gives a surface in a 3D space. Using this plot, the height (or the intensity) of the plateau (see Fig.2) can be determined and its value defines a threshold. The algorithm then looks for the pixels whose intensity go from an intensity below this threshold to an intensity above it. The results of this method are shown in Fig.3.

This algorithm can detect the outer boundary of the phantom, which was the primary goal of this algorithm. Holes inside the phantom are also detected. These holes are indeed filled with air, the intensity therefore comparable with the one outside the phantom.



(a) Initial image of slice 60

(b) Threshold method applied to slice 60

Figure 3 – Results of the threshold method

As one can see with the results presented in Fig. 3, this method does not allow the detection of the different region in the phantom if their intensity is not lower than the threshold. A lot of holes are not displayed on Fig. 3. This is therefore not applicable for further use.

Derivatives peaks in 1D This method is based on the variation (or the derivative) of the intensity. Unlike the previous algorithm, this should allow the detection of all the shapes inside the phantom. This method was first applied to a single row of the matrix representing the image. The intensity of the 200th row is shown on Fig.4.

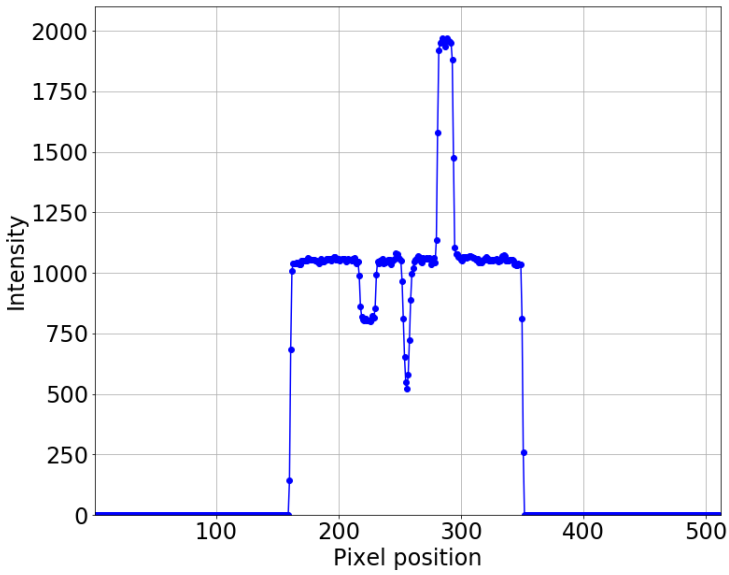


Figure 4 – Intensity of 200th row (slice 60)

The first step in the development of this algorithm was to determine the derivative of the intensity. Since a single row was studied, only the horizontal derivative was computed using the centred finite differences :

$$\left.\frac{\partial I}{\partial x}\right|_i \approx \frac{I_{i+1} - I_{i-1}}{2h} \propto (I_{i+1} - I_{i-1}) \quad (1)$$

where I_i is the intensity of the ith pixel in a chosen row and h is the distance between two pixels. Since h is constant, we only consider the intensity difference in (1). Using this property, the directional derivative of I is computed on a single row of the image. The peaks of the derivative gives the points where the rate of change of the intensity is maximum. This is the case when we leave one region of the

phantom to another one. So applying the function *findpeaks* (from *scipy.signal* library) on the derivative should give the boundaries in the phantom. Before using *findpeaks*, the image needed to be smoothed since the original one was noisy. A running mean was applied on a horizontal subset of size 5, just before *findpeaks*. The result obtained with this method is presented in Fig.5.

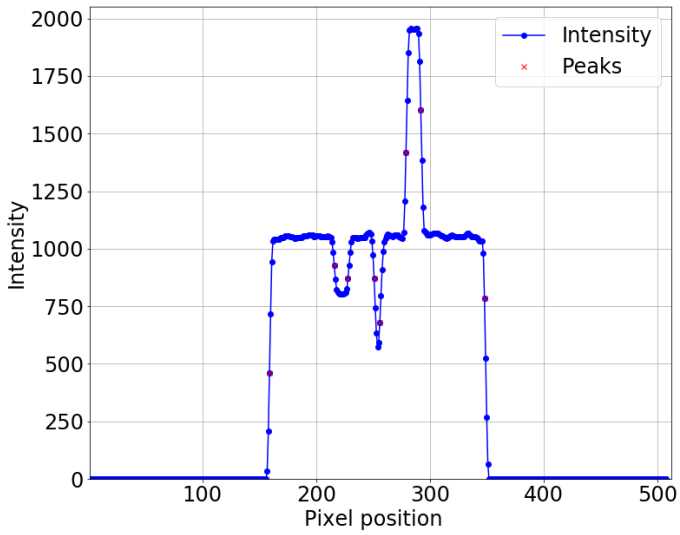


Figure 5 – Caption

Comparing this figure with Fig.4, one can see the smoother aspect of the second image. The obtained peaks indeed give the borders of the different regions. However, there seems to be a bias with the *findpeaks* function. On Fig.5 the peaks are selected sooner (one pixel sooner) when the derivatives is negative than when it is positive. Though a single pixel of difference is not such an issue. Moreover, *findpeaks* requires a threshold that defines the minimal height of the peaks and this choice is arbitrary. For this reason a different method, based on statistical values, was explored and presented in the following section.

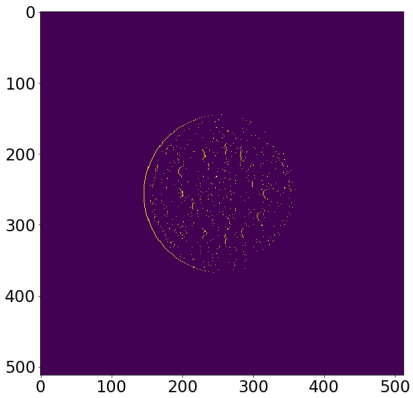
Statistical method on the derivatives

In the same way as the previous method, we are interested in the directional derivatives of the intensity. Here, we use a subset of size n on which the mean μ_1 and variance σ_1 of the intensity is computed. The mean intensity μ_2 of the next subset is compared with μ_1 and σ_1 in the following way :

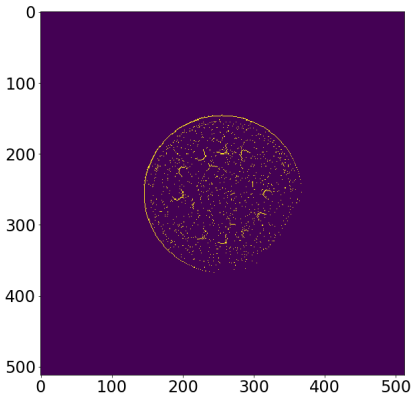
$$\mu_2 > \mu_1 + C\sigma_1 \tag{2}$$

where C is a given coefficient.

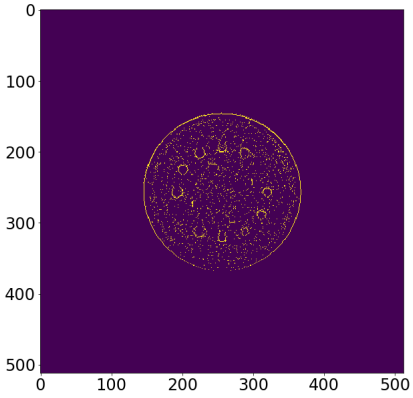
If this condition is fulfilled then the pixel in the middle of the second subset considered as a contour. The condition (2) can only be true for a positive derivative (from a low to a high intensity), the condition must therefore be tested in all four directions (i.e. left to right, top to bottom and so on). This method was applied on the entire image and the different steps are presented in Fig.6.



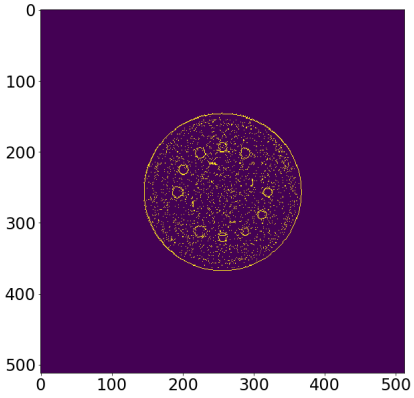
(a) Left ⟶ Right



(b) Top ⟶ Bottom



(c) Left ⟶ Right



(d) Bottom ⟶ Top

Figure 6 – The different steps of the statistical method.

This technique has a clear issue with noise. A simple algorithm that removes the isolated pixel was developed and the final result is shown on Fig.7.

This method is able to detect the different region inside the phantom. However, the boundaries are not smooth and above all, some of them are not closed. Furthermore, although most of the noise was filtered, there still remain some artefacts.

Since the results obtained with this method were not satisfying, we decided to further develop the previous algorithm.

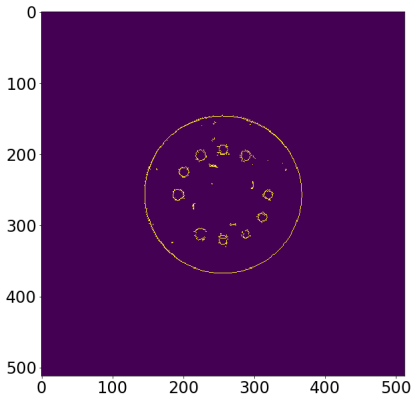
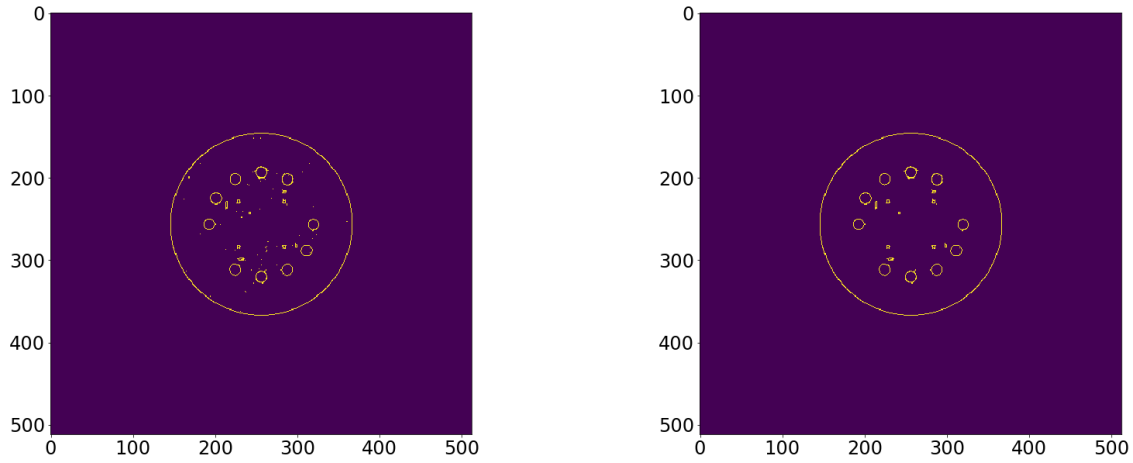


Figure 7 – Caption

Derivatives peaks in 2D The first difference with the method presented before is that a Gaussian filter was applied to the image. This filter does a convolution product between the intensity and a 5×5 matrix whose components are given by a 2D Gaussian function with variance σ. The function *gaussian filter* is implemented in the *scipy.ndimage* library and its arguments simply are the original image and σ.

The same process was then applied as explained above on the whole matrix in both x and y direction. The results are presented in Fig.8.



(a) Original result

(b) With the noise filter applied

Figure 8 – The different steps of the statistical method.

As for the statistical method, the contours are not really smooth. Nonetheless, these contours now define closed regions and the noise was efficiently eliminated by the filter.

Laplacian The two previous codes used directional derivatives of the intensity matrix. In order to find the contours, we had to find the maximums of the derivatives. We could also work on the Laplacian operator instead of the derivatives. Indeed, looking for the maximums of the derivatives is equivalent to searching for the zeros of the Laplacian. We chose to use finite difference approximations of the second derivative of the intensity matrix : central inside the pixel image and backward and forward on the edges of the image.

Canny Filter In this section, we tried to use the Canny edge detector already implemented in the *skimage.feature* library. This algorithm is a standard method used in image treatment. Canny edge detector is able to find every contour in a 2D-image and works as follow : a gaussian filter is first applied to the image to reduce the image's noise. Then the intensity gradient is computed at every point of the image. The contours are defined at local maxima of the gradient. The final step is the edge tracking by hysteresis. This method needs two arguments called low and high thresholds, given to the function. For each point of the intensity gradient, the same process is used : if the pixel intensity is under the low threshold, the point is rejected. If the pixel intensity is over the high threshold, the point is accepted and considered as forming a contour. If the pixel intensity is between the high and the low thre-

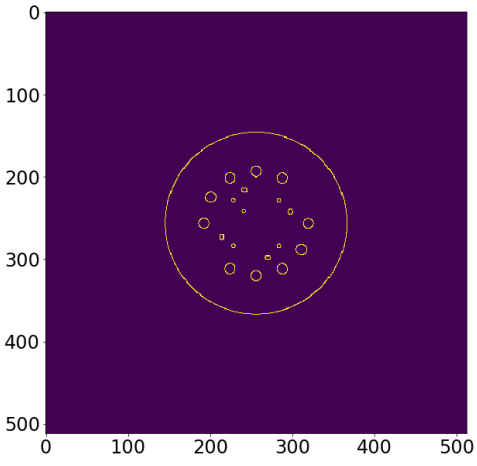


Figure 9 – Contours found by the Canny filter

shold, the point is only accepted if it is linked to a point already accepted. The matrix obtained with the Canny filter is displayed in Fig.10

At this stage of the algorithm, two categories of pixels are defined : the one (a) considered as forming a border and the others. The next step is to differentiate each border inside the image. In order to establish which pixel belongs to which border, we iterate over the matrix given by canny filter. When we arrive at a pixel belonging to the category (a), this pixel is considered as part as border number 1 and we then search recursively around this pixel for the other pixels belonging to the same border. The same process is repeated as many times as the number of different borders found in the image. To see better the different colors used to describe the different borders, we chose to fill each borders with the color of their contour as a first draft. The result is shown on Fig. 10. To be more coherent, we then decided not to use random colors but colors defined as the mean intensity inside each border. As you can see on Fig. 11, we retrieved the initial image but with each border detected.

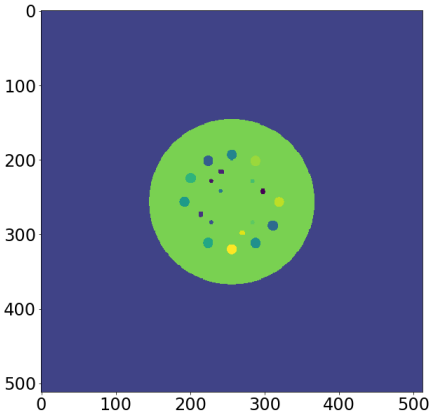
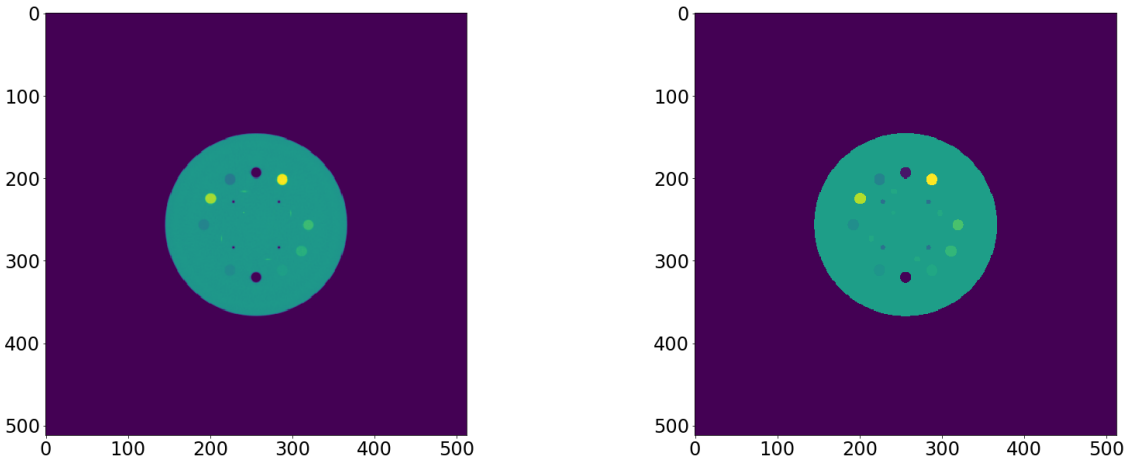


Figure 10 – On verra

However, a few arguments must be given.



(a) (b)

Figure 11

3 Remaining work

The algorithm developed up to this point work in the particular case of a phantom. Real images will probably not have such well defined region and their detection might be far more complicated. There is still some space for improvement to some of the methods used, in particular, the Laplacian and the directional derivatives will be further developed.

Although the different methods developed were applied to 2D images, the final goal of this project is to develop an algorithm for 3D features recognition. A 3D version of the algorithm

was tested where the edges were detected (with the Canny filter) on 2D slices of the scan and then grouped together in 3D. However, there seems to be a stack overflow due to deep recursion with the function grouping boundary together. This problem still has to be overcome.

Finally, the latest algorithm used for pattern recognition often use Machine Learning (ML) methods. This is a path we also would like to explore. We are complete beginners to ML so we have the learn the basics of this field first and then apply them to real images.

-Improve current codes (laplacian and directional derivatives)

-3D

-Machine Learning

4 New timeline and expected results

For the end of this term : first two points above

For second term : Machine learning + report writing