**Rapport**

Segmentation des images médicales et suivi de l’évolution

**Membres du groupe :**

Axel De Oliveira Lopes

Guillaume Orset-Prelet

Charlotte Ying

**SOMMAIRE**

[**Introduction**](#_ks91nhu8s3hn) **3**

[**Segmentation par seuillage de l’histogramme**](#_c5gpwa6pso2m) **3**

[**Segmentation par K-means**](#_v8v4t24zce64) **3**

[**Labellisation des images**](#_bhyskhv8loph) **4**

[**Localisation des zones à suivre**](#_bw20v0keve32) **6**

[**Conclusion**](#_eiytc7appniy) **7**

# 

# 

# Introduction

Le secteur médical est un des domaines d’application de l’IA parmi les plus prometteurs.

L’IA est très utile dans le domaine de l’imagerie en particulier sur 2 points :

* la classification d’images et la segmentation de zones spécifiques. Les algorithmes de classification d’images peuvent grandement faciliter le diagnostic en classifiant une image dans une catégorie prédéfinie de pathologie.
* Les algorithmes de segmentation des images sont fréquemment utilisés sur une très grande majorité de types d’imagerie.

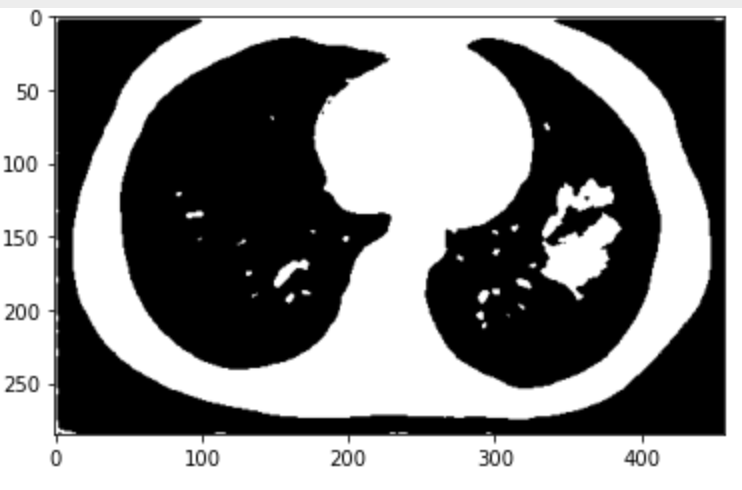
Nous allons nous intéresser plus particulièrement à la segmentation et à la labellisation des images médicales de poumons affectés par la COVID-19.

# Segmentation par seuillage de l’histogramme

La segmentation par seuillage de l’histogramme permet de transformer une image en couleur ou en niveaux de gris en une image binaire. C’est-à-dire que l’on modifie les pixels pour obtenir une image en noir et blanc afin de mettre en avant les parties qui nous intéressent. Ici, nous souhaitons utilisé cette technique pour représenter en noir les poumons et en blanc la partie impactée par le covid.

Dans notre cas, nous avons utilisé au préalable différentes techniques d’optimisation de l’image afin d’obtenir une histogramme le plus pertinent possible. En effet, nous transformons tout d’abord l’image couleur en niveaux de gris. Puis, nous utilisons un filtre gaussien pour “débruiter” l’image. Finalement, nous recherchons automatiquement le treshold optimal sur l’histogramme de l’image après “grayscale” et débruitage.

Voici l’image correspondant aux poumons impactés par le covid. La zone en blanc où agit le covid est plus nettement identifiable.



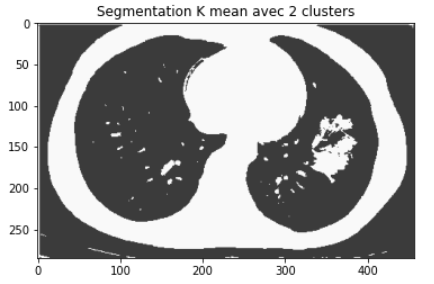
# Segmentation par K-means

La méthode de clustering permet de former des groupes homogènes au sein d’un ensemble de données, en se basant sur leurs caractéristiques communes. L’algorithme des K-means est une méthode de clustering non supervisée.

Dans le cadre de l’imagerie, la méthode des k-means permet de rassembler des pixels en K groupes ou “clusters”, choisis arbitrairement, en fonction de leur couleur par exemple. Chaque pixel est alors affecté au centroïde le plus proche en termes de couleur pour ensuite re-déterminer les centroïdes.

La difficulté avec cette méthode est qu’il faut connaître le nombre de clusters au préalable, or il est souvent difficile de prédire ce paramètre de telle sorte qu’il optimise une analyse.

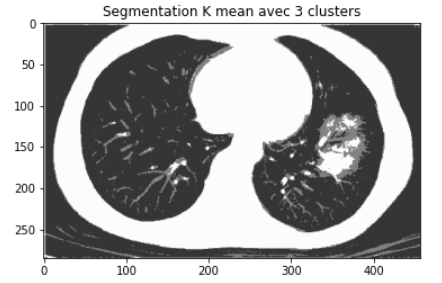
Dans le cas des images médicales de poumons, affectés ou non par le virus de la COVID-19, nous avons d’abord fait tourner l’algorithme avec 2 clusters. Voici le résultat :



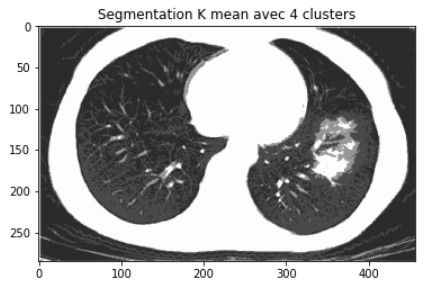
Nous avons obtenu ce résultat en :

1. convertissant l’image d’origine en une matrice de pixels avec 3 couleurs RGB, donc matrice MxNx3. Chaque ligne est un vecteur dans l'espace 3D des RGB,
2. on définit ensuite le critère d’arrêt de l’algorithme : nous avons choisis 100 itérations et epsilon, la précision, égale à 0.2,
3. nous lançons ensuite l’algorithme des k-means de la librairie cv2,
4. on traduit la valeur des centroïdes en uint8,
5. on remplace ensuite la valeur des pixels avec la valeur de leur centroïde
6. et on affiche le résultat.

Voici le résultat en segmentant l’image en 3 clusters :



Voici le résultat d’une segmentation en 4 clusters :



On observe une perte d’information relativement importante avec une segmentation en 2 clusters tandis que nous observons la présence d’information inutile avec une segmentation en 4 clusters.

# Labellisation des images

Tout d’abord, nous inversons les nuances de gris de notre image car nous avons remarqué que la labellisation se faisait sur les éléments de fond et non sur les éléments qui nous intéressaient.

Nous avons commencé par convertir notre image dans un colorspace différent : hed. C’est un color space largement utilisé pour faire de l’analyse d’images médicales.

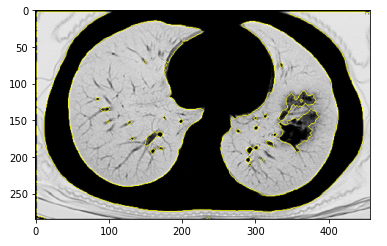
Nous avons décidé d’utiliser la méthode de seuillage automatique d’Otsu car cette dernière nous donnait de meilleurs résultats.

Nous avons donc créé un masque qui va déterminer les contours de chacun de nos labels en passant par la méthode de seuillage automatique d’Otsu. Ce seuillage permet de transformer une image en niveau de gris en image binaire, c’est-à-dire en image noir et blanc. Cela suppose de trouver un seuil permettant de déterminer l’affectation d’un pixel à une classe ou à une autre. La méthode d’otsu permet de trouver le meilleur seuil à prendre en minimisant la variance intra classe. L’algorithme compare donc le résultat de chacun des seuillages possibles et récupère le meilleur.

Le masque que nous récupérons délimite nos labels, et nous appliquons 2 méthodes qui permettent de rendre notre labellisation plus précise :

1. La première méthode ‘opening’ supprime les labels les plus petits qui ne nous intéressent pas.
2. La seconde méthode ‘closing’ va supprimer les trous qu’il pourrait y avoir dans nos labels dû à l’analyse de l’image ou l’image elle-même.

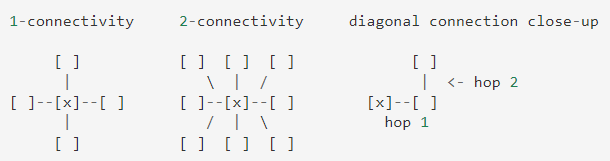
On obtient le masque ci-dessous, que nous superposons ensuite sur notre image. On s’aperçoit que le masque est cohérent et nous pouvons donc continuer la labellisation.



Nous transformons les zones que notre masque a récupérées en labels avec la méthode ‘label’ de skimage.measure. On obtient une image avec nos labels colorés d’une manière différente pour chaque label pour bien constater quels sont les labels qui ont été pris en compte.

La méthode de labellisation consiste à étiqueter les régions d’une matrice d’entiers qui sont connectées entre elles.

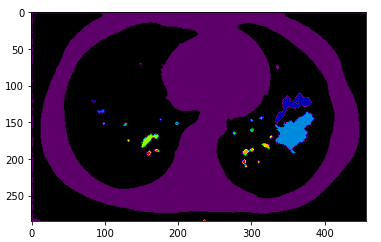
Deux pixels sont connectés lorsqu’ils sont voisins et qu’ils ont la même valeur. Sur un plan, ils peuvent être voisins soit par 1-connectivité, soit par 2-connectivité.



*Explications de la connectivité par le site skimage*

Nous avons choisi la 2-connectivité car les labels seront capturés de manière plus précise.

La labellisation se différencie de la classification en ce qu’elle consiste à mettre une étiquette sur un ensemble de points qui sont interconnectés, tandis que la classification regroupe un ensemble d’objets partageant certains points communs.



On affiche la taille de ces labels sous forme d’histogramme ce qui nous sera utile pour la partie de localisation. Les plus gros labels sont en fait le fond de l’image dont nous voulons nous débarrasser, et il reste aussi des très petits labels qui n’ont pas d’importance ou qui sont peut être liés à la qualité de l’image.

# Localisation des zones à suivre

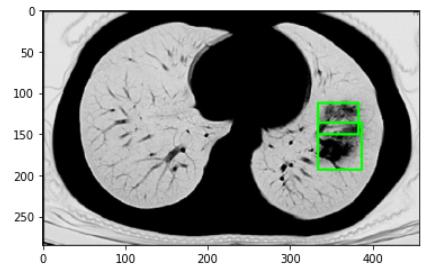
Pour localiser les zones à suivre, nous récupérons les labels trouvés à l’étape de labellisation en fonction de sa surface:

* pour un label i donné,
* nous calculons la surface qu’occupe ce label grâce à la méthode (lab==i).sum(). Cela nous permet de savoir si un label est plus ou moins “imposant”.
* Nous considérons qu’un label est imposant si sa surface estimée est supérieure à 500 mais inférieure à 50 000 : cela nous permet d’ignorer les veines pulmonaires ou le fond par exemple, qui était également vu comme un label. Nous estimons ces labels comme étant “intéressant”.
* Pour ces labels “intéressants”, nous allons les localiser.

Une fois ces labels isolés, nous récupérons leurs coordonnées sur le plan pour ensuite dessiner des rectangles sur l’image d’origine. Nous avons alors récupéré les coordonnées des coins inférieur gauche et supérieur droit de chaque label.

Nous déterminons les coordonnées de la première ligne qui ne contient pas uniquement des 0 (ce qui veut dire que l’un des pixels de la ligne est coloré). On fait la même chose sur les colonnes pour obtenir notre y du ‘starting point label’. On réitère ces 2 étapes mais cette fois-ci en partant du coin de l’image opposé au premier coin.

Le résultat est stocké dans une variable que nous avons nommée coord\_rectangle que nous exploitons pour dessiner des rectangles sur l’image originale. Le résultat est notre image initiale avec autant de rectangles que de labels respectant le critère de surface que nous avons imposé.



# Conclusion

La segmentation est une étape fondamentale en traitement d'image, toutefois, du fait qu’il n’existe pas de méthode unique de segmentation, il est important de définir une segmentation adaptée aux spécificités d’un projet, permettant ainsi d’améliorer une image avant de la labelliser afin d’en extraire les informations pertinentes.

Aujourd’hui, la labellisation est au fondement de l’application de l’intelligence artificielle à l’analyse d’images, elle sert aussi bien dans le domaine du traitement de l’imagerie médicale que dans l’autonomisation des véhicules.

Si l’IA est largement utilisée dans le cadre de la segmentation des images, elle n’a pas encore fait ses preuves sur le diagnostic et le pronostic. L’objectif est donc à terme de pouvoir prédire l’évolution d’une maladie, prévenir une maladie ou encore prédire l’évolution d’un traitement médical pour favoriser la guérison.

**Référence**

* Intelligence artificielle & imagerie médicale. <https://www.ihu-liryc.fr/fr/actualites/b/intelligence-artificielle-imagerie-medicale/>
* “K-means : Focus sur cet algorithme de Clustering & Machine Learning.” *Formation Data Science | DataScientest.com*, 15 May 2020, <https://datascientest.com/algorithme-des-k-means>.
* *Code for How to Use K-Means Clustering for Image Segmentation Using OpenCV in Python - Python Code*. https://www.thepythoncode.com/code/kmeans-for-image-segmentation-opencv-python.
* “Méthode d’Otsu.” *Wikipédia*, 8 Nov. 2021. *Wikipedia*, <https://fr.wikipedia.org/w/index.php?title=M%C3%A9thode_d%27Otsu&oldid=187815418>.
* *K-Means Clustering in OpenCV — OpenCV-Python Tutorials Beta Documentation*. <https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_ml/py_kmeans/py_kmeans_opencv/py_kmeans_opencv.html>.
* *Image Processing (33) | Image Segmentation | Object Labeling & Features*. *www.youtube.com*, <https://www.youtube.com/watch?v=7bOH4E7c8gU>.
* *Thresholding – Image Processing with Python*. <https://datacarpentry.org/image-processing/07-thresholding/>