

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2019-2020

Analyse d'images échographiques des poumons

Rapport

**POLYTECH[®]**
TOURS

Tuteurs académiques

Makris PASCAL

Ragot NICOLAS

Hidane MONCEF

Étudiant

Ponceau NATHANAËL (DI5)

12 avril 2020



Liste des intervenants

Nom	Email	Qualité
Ponceau NATHANAËL	nathanael.ponceau@etu.univ-tours.fr	Étudiant DI5
Makris PASCAL	pascal.makris@univ-tours.fr	Tuteur académique, Département Informatique
Ragot NICOLAS	nicolas.ragot@univ-tours.fr	Tuteur académique, Département Informatique
Hidane MONCEF	moncef.hidane@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Ponceau Nathanaël susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Makris Pascal, Ragot Nicolas et Hidane Moncef susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Ponceau Nathanaël, *Analyse d'images échographiques des poumons: Rapport*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2019-2020.

```
@mastersthesis{
  author={Nathanaël, Ponceau},
  title={Analyse d'images échographiques des poumons: Rapport},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2019-2020}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
I Introduction	1
1 Contexte	2
2 Objectifs	2
3 Hypothèses	3
4 Bases méthodologiques	3
II Description générale	5
5 Environnement du projet	5
6 Caractéristiques des utilisateurs	6
7 Fonctionnalités du système	6
8 Structure générale du système	8
III État de l’art	9
9 Automated B-Line Scoring on Thoracic Sonography	9
9.1 Approche.....	9
9.2 Algorithme	10
9.3 Résultat	11

10	Novel Automatic Detection of Pleura and B-lines (Comet-Tail Artifacts) on In-Vivo Lung Ultrasound Scans.....	11
10.1	Approche.....	11
10.2	Algorithme	11
10.3	Résultat	13
11	Line Detection as an Inverse Problem : Application to Lung Ultrasound Imaging..	13
11.1	Approche.....	13
11.2	Algorithme	13
11.3	Résultat	15
IV	Analyse et Conception	16
12	Marche aléatoire	16
12.1	Marche aléatoire à une dimension	17
12.2	Marche aléatoire en deux dimensions.....	17
12.3	Segmentation d'une image échographique avec une marche aléatoire en deux dimensions.....	18
12.4	Implementation en Python.....	18
13	Détection des lignes B	18
13.1	Calcul du gradient de l'image	19
13.2	Implémentation en Python.....	19
13.3	Transformée de Hilbert.....	19
13.4	Implementation en Python.....	19
13.5	Transformée logarithmique	20
13.6	Implementation en Python.....	20
13.7	Calcul du masque binaire	20
13.8	Implementation en Python.....	20
14	Filtrage séquentiel alternatif	20
14.1	Erosion	21
14.2	Dilatation	21
14.3	Ouverture morphologique.....	21
14.4	Fermeture morphologique.....	22
14.5	Algorithme de filtrage séquentiel alternatif	22
14.6	Implementation en Python.....	23
15	Filtre Top-hat	23
V	Mise en oeuvre	24
16	Les outils et bibliothèques.....	24
16.1	Environnement de développement intégré.....	24
16.2	OpenCv	24

16.3	Pydicom	25
16.4	QT	25
16.5	PyInstaller.....	25
16.6	Inno Setup.....	25
16.7	Autres bibliothèques.....	25
17	Implémentation	26
17.1	Algorithme de détection des lignes B.....	26
17.1.1	Détection de la zone de la ligne pleural : Modification.....	26
17.2	Détection des lignes B.....	28
17.2.1	Détection des lignes B : modifications	29
17.2.2	Détection des lignes B : conclusion	31
17.3	GUI.....	31
17.3.1	Classe QLabelSelectable	32
17.4	Calcul du pourcentage de pixel blanc	33
17.5	Modularité du programme et ré-utilisation.....	33
18	Risques.....	34
VI	Bilan et conclusion	35
19	Fait au S9.....	35
20	Reste à faire au S9.....	35
21	Fait au S10.....	35
22	Reste à faire au S10.....	36
23	Planning.....	36
24	Bilan sur la qualité	36
25	Bilan auto-critique sur la gestion du projet	36
	Annexes	37
A	Spécifications fonctionnelles Détaillé	38
1	Définition de la fonction OpenDicomFile	40
1.1	Rôle	40
1.2	Description.....	40
2	Définition de la fonction OpenPatientsFolderPath.....	40
2.1	Rôle	40
2.2	Description.....	41
3	Définition de la fonction Pleural-line detection.....	41
3.1	Rôle	41
3.2	Description.....	42
4	Définition de la fonction B-line detection	42

4.1	Rôle	42
4.2	Description.....	43
5	Définition de la fonction Alternate sequential filtering.....	43
5.1	Rôle	43
5.2	Description.....	44
6	Définition de la fonction Top-hat filtering	44
6.1	Rôle	44
6.2	Description.....	45
7	Définition de la fonction Show B Lines	45
7.1	Rôle	45
7.2	Description.....	46
8	Définition de la fonction Save Result.....	46
8.1	Rôle	46
8.2	Description.....	47
B	Spécifications non fonctionnelles	48
1	Contraintes de développement et conception	48
2	Contraintes de fonctionnement et d'exploitation	48
2.1	Performances.....	48
2.2	Capacités.....	48
2.3	Contrôlabilité.....	49
2.4	Sécurités	49
C	Description des interfaces externes du logiciel	50
1	Interfaces matériel/logiciel	50
2	Interfaces homme/machine	50
3	Interface logiciel/logiciel.....	51
D	Gestion de projet	52
1	Diagramme de Gant du S9	53
2	Diagramme de Gant du S10	55
E	Github, installeur et exécutable	56
1	Création de l'exécutable à partir des sources.....	56
2	Création de l'installeur à partir de l'exécutable	56
F	Doc d'installation utilisateur	58
G	Doc d'installation développeur	62
H	Doc d'utilisation	63

I	Webographie	66
J	Bibliographie	67

Première partie

Introduction

Ce document établit les spécifications système du Projet Recherche et Développement (PRD) : “Analyse d’images échographiques des poumons”.

Il présente en détails, dans un premier temps, le contexte de la réalisation, sa description générale ainsi que les interfaces externes du logiciel puis, dans un second temps, l’architecture générale du système, une description des fonctionnalités et les différentes conditions de fonctionnement.

Ce PRD est proposé par l’équipe de Reconnaissance des Formes et Analyse d’images de l’Ecole d’Ingénieur Universitaire Polytechnique de Tours en collaboration avec le CHRU Bretonneau.

Il est supervisé par messieurs Nicolas Ragot, Hidane Honcef et Pascal Makris qui représentent la MOA. La réalisation du projet est confiée à moi-même, Nathanaël Ponceau, qui représente la MOE.

Ce document a été relu et approuvé par Nicolas Ragot, Hidane Honcef et Pascal Makris.

1 Contexte

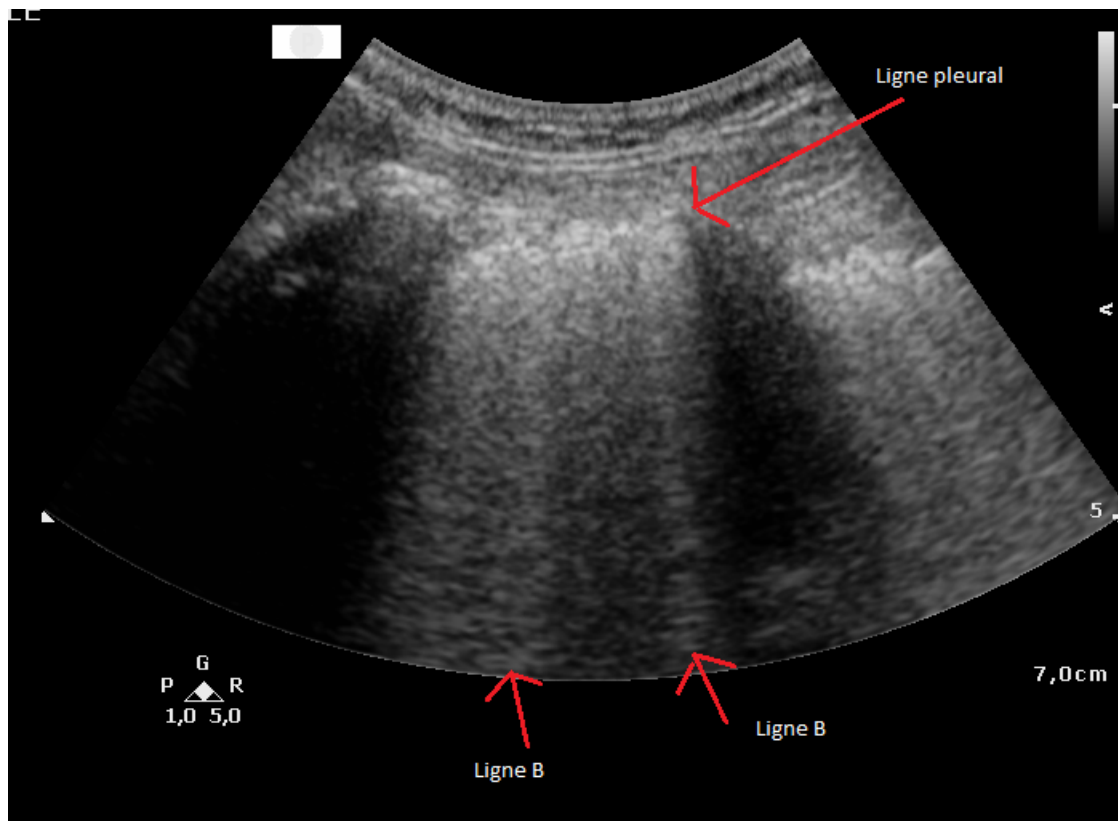
En effet son utilisation pour les poumons était jusqu'à présent considéré comme peu pertinente étant donné que le poumon est constitué d'un tissu mou rempli d'air donc peu échogène. L'échographie donnant par conséquent des résultats difficilement exploitables pour le diagnostic médical.

Cependant un cas particulier s'y prête bien. Il concerne les pathologies respiratoires où les poumons se remplissent d'eau, facilitant la propagation des ultrasons permettant de ce fait une meilleure détection des parois.

L'utilisation de l'échographe plutôt que d'autres modalités peut être préférée dans un premier temps car il est non invasif et facile à réaliser.

Il est en effet possible de détecter sur l'échographie la ligne pleurale ainsi que des lignes B qui selon leur nombre indique présence voire la gravité de la pathologie.

De plus, la forme/régularité de la ligne pleurale peut aussi être un symptôme de la maladie.



Jusqu'à présent, la détection de la ligne pleurale et des lignes B sont réalisées visuellement par les médecins.

Ce type de détection peut être informatisée. Il pourrait permettre à du personnel non formé de réaliser des examens préliminaires sans l'aide d'un médecin. La solution recherchée est de pouvoir intégrer l'opération de détection dans l'échographe même.

2 Objectifs

Ce projet consiste à mettre en place une méthode de détection de la ligne pleurale et des lignes B afin de pouvoir livrer un logiciel d'aide au diagnostic des maladies pulmonaires.

Le principe retenu s'appuiera sur le fait que l'expert utilise 14 points d'observation obtenus par déplacement de la sonde échographique. Ces différents points donnant lieu à autant de vidéos, il faudra donc pouvoir détecter pour chacune-elle, la ligne pleural et les ligne B présentes. Il est demandé pour le rendu visuel que chaque détection d'une ligne pleural se traduise par la superposition d'une ligne B détectée par l'algorithme. En complément à cela, seront affichées la moyenne de toutes les ligne B détectées selon les différents axes d'observation ainsi que l'épaisseur de la ligne pleural.

Dans le cas particulier où le nombre de lignes B présentes est très important, scénario qui ne permet pas le décomptage par l'expert car exercice difficile pour ce dernier, il a été convenu que l'algorithme calculera un pourcentage entre la quantité de pixels blancs et noirs présents au sein de la zone étudiée.

3 Hypothèses

Les recherches effectuées lors de l'étude de l'état de l'art ont permis de découvrir et de lister différentes méthodes utilisées aujourd'hui pour détecter la ligne pleural ainsi que les lignes B. Parmi celles-ci, la méthode décrite dans l'article intitulé "Novel Automatic Detection of Pleura and B-lines (Comet-Tail Artifacts) on In-Vivo Lung" publié en 2016 par Moshavegh Ramin, Hansen Kristoffer Lindskov, Møller-Sørensen Hasse, Hemmsen Martin Christian, Ewertsen Caroline, Nielsen Michael Bachmann et Jensen Jørgen Arendta été choisie.

Cette méthode a été choisie afin de garantir la remise d'un livrable fonctionnelle aux médecins du CHU à l'issue de la période du PRD. Cependant lors de la réalisation de l'état de l'art il est apparu l'existence d'autres méthodes sans doute plus efficace mais leurs complexités mathématiques nécessitaient un temps de compréhension relativement long au regard des contraintes et du périmètre du projet.

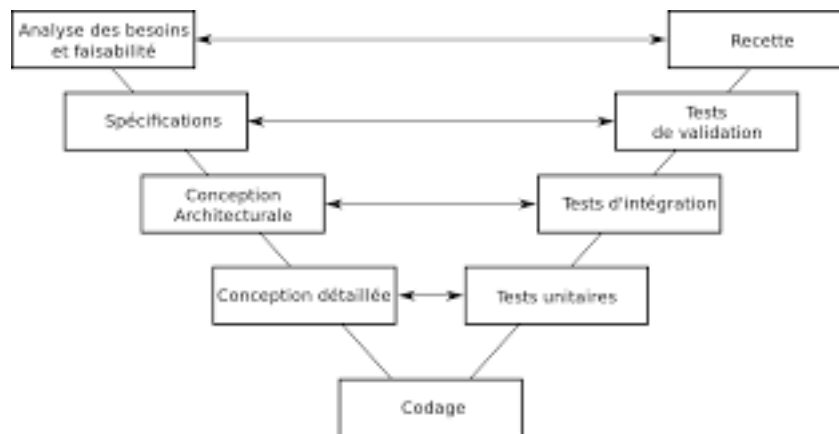
Néanmoins, il a été convenu que si le déroulé du projet permettait de finir plus tôt, une autre approche serait tentée. Dans cette perspective, la réalisation de l'interface devra prévoir l'ajout d'algorithmes différents devant être testé sur le long terme.

4 Bases méthodologiques

Pour le développement du projet, le choix du langage python a été retenu avec prise en compte de la norme PEP8 (Python Enhancement Proposal) qui aura pour obligation un développement normalisé du code.

Concernant l'outil de gestion du projet, l'auteur réalisera un découpage en tâches de ce dernier qui sera présenté sous la forme d'un diagramme de Gantt. En complément à cela, le suivi proprement dit et la gestion sera mis en place à partir de l'outil Trello.

En termes de méthodologie de gestion de projet, le modèle du cycle en V a été choisi.



Pour modéliser le système, on utilisera le langage UML (Unified Modeling Language) afin de décrire le plus clairement possible à partir des diagrammes l'architecture de l'application.

La rédaction du document sera réalisée à partir du compilateur Latex.

Deuxième partie

Description générale

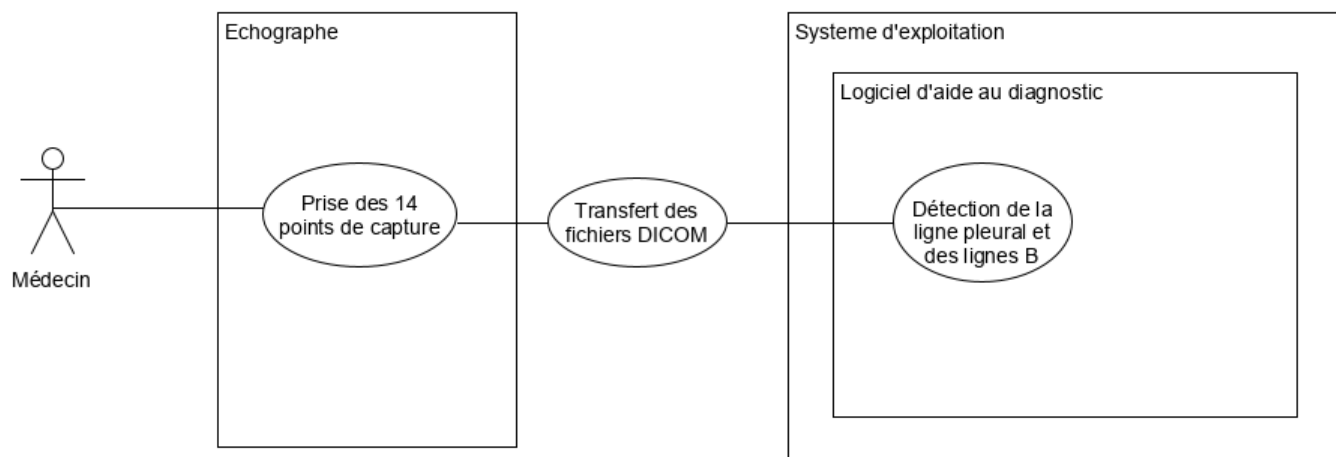
5 Environnement du projet

Pour ce projet, aucun existant n'est présent.

L'environnement de développement est le suivant :

1. - Système d'exploitation : Windows
2. - Language de développement : Python 3.7
3. - IDE : Pycharm

L'intégration du logiciel dans son environnement peut être résumé comme suit :



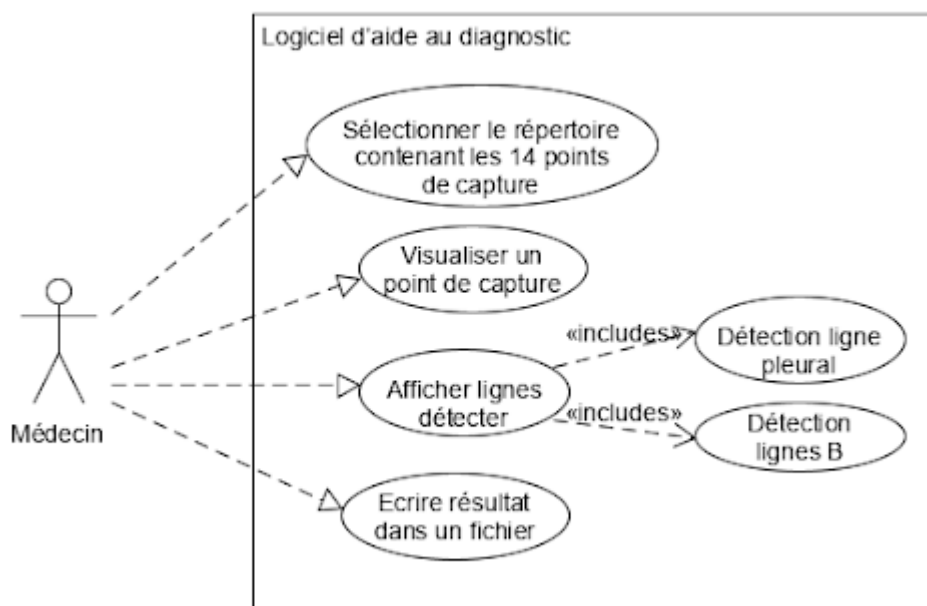
6 Caractéristiques des utilisateurs

Pour cette application, il n'y a qu'un seul type d'utilisateur : le médecin/praticien :

Caractéristiques	Connaissance de l'informatique	Expérience de l'application	Type d'utilisateur	Droit d'accès
Réponses	Non	Non	Régulier	Tous les droits

Il n'est pas nécessaire d'avoir des connaissances en informatique pour que les médecins utilisent l'application étant donné qu'elle nécessite uniquement l'importation de séquences vidéos.

7 Fonctionnalités du système



L'utilisateur grâce à l'application pourra choisir un répertoire contenant les 14 points de captures respectant l'arborescence déjà utilisée par les médecins. Celle-ci sera présentée ci dessous.




















Ensuite, en choisissant un point de capture, présent dans le répertoire, il pourra visualiser la vidéo en question et choisir d'afficher les lignes détectées par l'algorithme (image par image).

Dans le cas de la ligne pleurale, le médecin pourra afficher l'épaisseur de la ligne pleurale. Pour finir, l'utilisateur pourra sauvegarder ces résultats dans un fichier csv mais aussi les images affichant en surimpression les lignes.






Dernière action possible, si l'utilisateur le souhaite, il sera également possible de sauvegarder la moyenne du nombre de lignes détectées et la moyenne de l'épaisseur de la ligne pleurale sur les 14 points de capture.

L'arborescence peut être divisée en 3 parties :















Un dossier correspondant à un seul et unique patient :

 1Patient 1CP01	25/09/2019 12:49	Dossier de fichiers
 2Patient 1BR02	25/09/2019 12:50	Dossier de fichiers
 3Patient 1DR03	25/09/2019 12:51	Dossier de fichiers
 4Patient 1HM04	25/09/2019 12:53	Dossier de fichiers
 5Patient 1PF05	25/09/2019 12:54	Dossier de fichiers
 6Patient 1AR06	25/09/2019 12:55	Dossier de fichiers
 7Patient 1MD07	25/09/2019 12:37	Dossier de fichiers
 8Patient 1GM08	25/09/2019 12:38	Dossier de fichiers
 09Patient 1TM09	25/09/2019 12:39	Dossier de fichiers
 10Patient 1PG10	25/09/2019 12:40	Dossier de fichiers
 11Patient 1HF11	25/09/2019 12:42	Dossier de fichiers
 12Patient 1FS12	25/09/2019 12:43	Dossier de fichiers
 13patient 1HJ13	25/09/2019 12:44	Dossier de fichiers
 14Patient 1BJ14	25/09/2019 12:45	Dossier de fichiers
 15Patient 1HN15	25/09/2019 12:46	Dossier de fichiers
 16Patient 1AJ16	25/09/2019 12:47	Dossier de fichiers
 17Patient 1SG17	25/09/2019 12:48	Dossier de fichiers
 18Patient 1SJ18	25/09/2019 12:57	Dossier de fichiers
 19Patient 1MJ19	25/09/2019 12:58	Dossier de fichiers

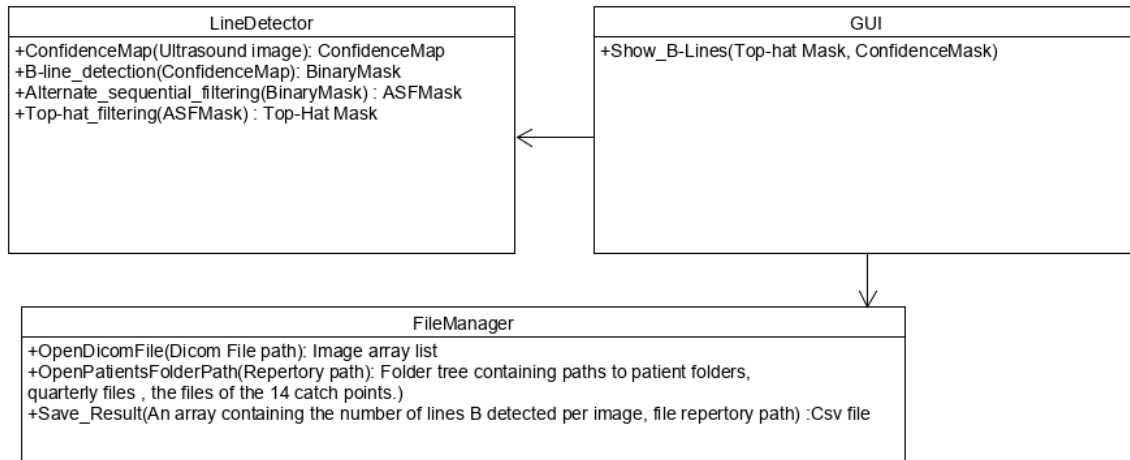
Ce dossier patient est lui même divisé en trimestre de traitement :

 101M0	25/09/2019 12:16	Dossier de fichiers
 101M3	18/05/2019 11:40	Dossier de fichiers
 101M6	18/05/2019 11:40	Dossier de fichiers
 101M9	18/05/2019 11:40	Dossier de fichiers
 101M12	18/05/2019 11:40	Dossier de fichiers

Dans chaque dossier trimestriel de traitement, on retrouve les 14 vidéos des points de capture de l'échographe :

 2019010A	11/06/2019 11:59	Fichier	41 038 Ko
 2019010C	11/06/2019 11:59	Fichier	38 712 Ko
 2019010E	11/06/2019 11:59	Fichier	41 509 Ko
 2019010G	11/06/2019 11:59	Fichier	45 351 Ko
 2019010I	11/06/2019 12:00	Fichier	43 907 Ko
 2019010K	11/06/2019 12:00	Fichier	41 897 Ko
 2019010M	11/06/2019 12:00	Fichier	46 240 Ko
 2019010O	11/06/2019 12:00	Fichier	40 604 Ko
 2019010Q	11/06/2019 12:00	Fichier	45 020 Ko
 20190100	11/06/2019 11:59	Fichier	47 324 Ko
 20190102	11/06/2019 11:59	Fichier	49 462 Ko
 20190104	11/06/2019 11:59	Fichier	45 884 Ko
 20190106	11/06/2019 11:59	Fichier	42 091 Ko
 20190108	11/06/2019 11:59	Fichier	42 050 Ko

8 Structure générale du système



La structure générale du logiciel peut être résumée comme ci-dessus. Le projet étant réalisé en Qt, la notion de controller est directement intégrée à la vue donc à notre classe GUI.

Une instance de la classe LineDetector sera utilisée par le GUI pour toutes les images visualisées afin d'afficher en surimpression les lignes détectées (ligne pleurale comprise) et leur nombre.

Une instance de la classe FileManager permettra au GUI de naviguer dans l'arborescence choisie par l'utilisateur et pouvoir ouvrir les fichiers DICOM s'y trouvant pour en récupérer les images.

Cette même classe sera ensuite utilisée pour sauvegarder les résultats dans un fichier Csv.

Troisième partie

État de l'art

Dans cette partie, on décrit les différentes méthodes existantes de détection de la ligne pleurale et des lignes B extraites d'articles de différentes revues scientifiques.

9 Automated B-Line Scoring on Thoracic Sonography

9.1 Approche

Cette méthode est extraite de l'article "Automated B-Line Scoring on Thoracic Sonography" du Massachusetts Institute of Technology Lincoln Laboratory, Lexington Massachusetts en collaboration avec le Department of Emergency Medicine, Division of Emergency Ultrasound, Massachusetts General Hospital, Boston. Il a été publié en 2013.

Les échographes utilisés sont de marque MicroMaxx 2-dimensional réglés sur une fréquence variant entre 2 et 5 MHz.

Pour tester les résultats de leur algorithme de détection, 50 vidéos d'échographies de poumons sont tirées d'une autre étude réalisée au Massachusetts General Hospital.

9.2 Algorithme

Etape 1 : Polar reformatting

La forme d'une image échographique, comme on peut le voir dans la figure 1 partie B, est en forme d'éventail, ce qui n'est pas pratique car une image est normalement représentée sous forme rectangulaire.

Pour obtenir une image rectangulaire, les images extraites d'une vidéo sont découpées selon 40 coupes angulaires et 60 coupes radiales, ce qui permet de recréer des images rectangulaires de 60 lignes et 40 colonnes.

Chaque coupe angulaire représente une colonne.

Etape 2 : Pleural line estimation

La ligne pleurale est enlevée avant de détecter les lignes B. Un algorithme de détection de ligne et de seuillage d'image est utilisé pour estimer la position de la ligne pleurale et l'exclure de l'image.

Etape 3 : Feature extraction

On sélectionne parmi les coupes angulaires celles qui possèdent une intensité lumineuse uniforme du début de la ligne pleurale jusqu'en bas de l'image.

Etape 4 : Classification

Dans cette étape, 5 caractéristiques des lignes B sont utilisées pour les détecter dans les coupes angulaires choisies dans l'étape 3.

L'unité de mesure de ces caractéristiques est l'intensité lumineuse des pixels de l'image.

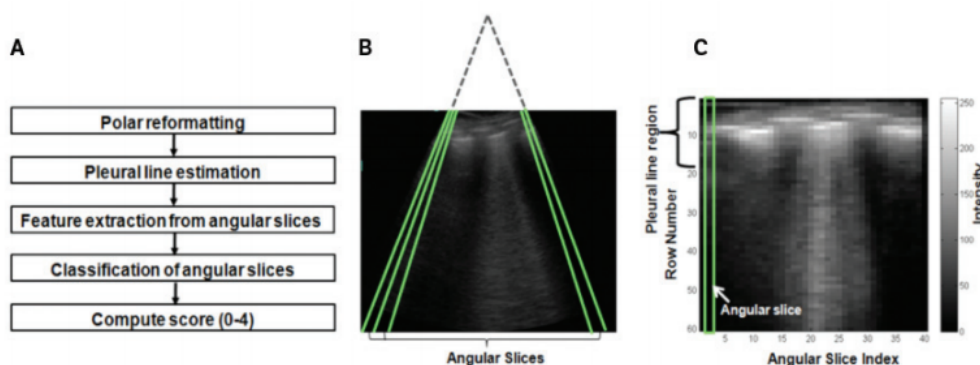
Ces caractéristiques sont la moyenne, la médiane, l'épaisseur de la colonne précédente, la valeur de la dernière ligne de la colonne, le ratio de la valeur de la dernière colonne par rapport à la valeur maximum de cette colonne et le ratio de la valeur de la moitié d'une colonne par rapport à la valeur maximum de cette colonne.

On considère une colonne comme possédant une ligne B si les 5 caractéristiques dépassent une valeur prédéfinie.

Etape 5 : Compute score

Pour calculer la note d'une image de la vidéo, on fait la somme des lignes détectées sur cette frame puis, finalement pour attribuer une note à la video, on choisit la note maximum obtenue sur toutes les images de la vidéo.

Figure 2. Automated B-line scoring algorithm. **A.** Algorithm flow. **B.** Polar reformatting that shows the angular slices. **C.** Normalized image of Figure 1A after polar reformatting. Note the pleural line regions at the top of the view.



9.3 Résultat

Pour valider les résultats de l'algorithme, ces prédictions sont comparées aux notes de deux experts. Lors d'un premier essai utilisant 13 vidéos extraites des 50 vidéos de l'étude, l'algorithme obtient des notes similaires à 90 pourcent aux experts. Dans un deuxième essai utilisant d'autres vidéos de l'étude mais aussi des vidéos ne contenant aucune ligne B, l'algorithme obtient 100 pourcent de notes similaires aux notes des deux experts.

10 Novel Automatic Detection of Pleura and B-lines (Comet-Tail Artifacts) on In-Vivo Lung Ultrasound Scans

10.1 Approche

Cette méthode est extraite de l'article "Novel Automatic Detection of Pleura and B-lines (Comet-Tail Artifacts) on In-Vivo Lung Ultrasound Scans" du Dept. of Elec. Eng., Technical University of Denmark, DK-2800 Lyngby, Denmark en collaboration avec le Dept. of Radiology, Dept. of Thoracic anesthesiology, Copenhagen University Hospital, DK-2100 Copenhagen, Denmark. Il a été publié en 2016.

Les vidéos utilisées pour l'étude sont obtenues avec un échographe BK3000 ultrasound scanner (BK Ultrasound, Denmark) sur une fréquence de 5.5 MHz.

Pour réaliser ces vidéos, 6 patients dont 3 ayant subi de grosses opérations chirurgicales et 3 patients "normaux" sont choisis pour réaliser 6 vidéos de 50 images.

L'algorithme a été testé sur les 300 images générées.

10.2 Algorithme

L'algorithme est composé de 4 étapes :

Etape 1 : Calcul de la map de confiance et détection de la ligne pleurale

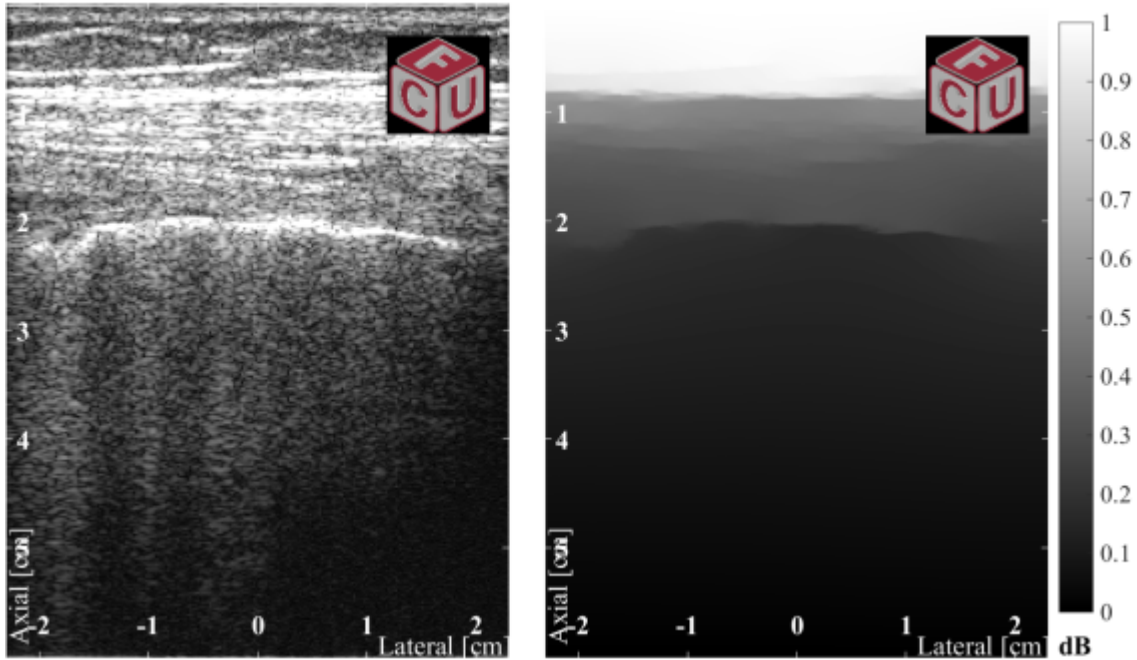
Le but de cette première étape est de déterminer la position de la ligne pleurale en utilisant un algorithme de segmentation d'image appelé Random Walker.

Cet algorithme est modifié en utilisant la loi de Beer-Lambert qui exprime la profondeur du signal en fonction de l'atténuation. Le signal atténué peut être représenté par

$$I = I_0 \exp(-\alpha d)$$

où I_0 est l'intensité du pixel original, α est le coefficient d'atténuation et d la distance depuis la source de l'image.

De cette manière, plus le Random Walker est loin de la source de l'image, plus il colorie le pixel en noir, ce qui permet de générer une carte de confiance.

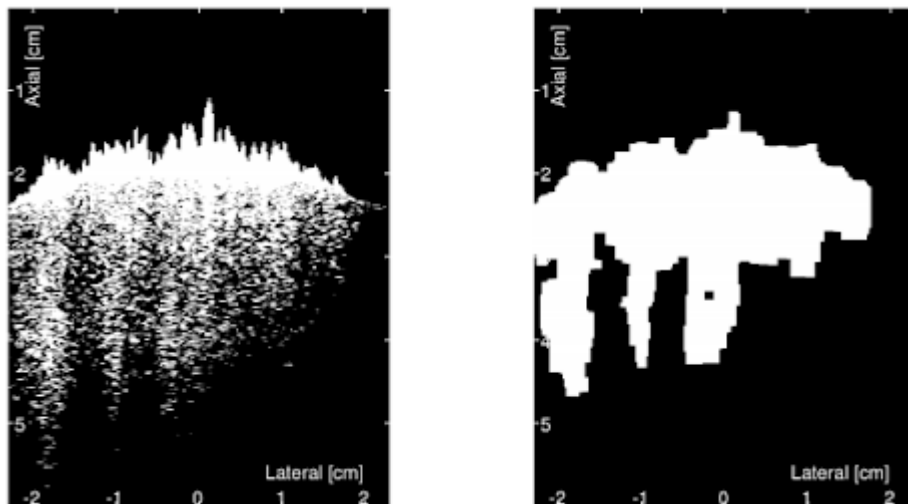


La ligne pleurale est détectée en seuillant la carte de confiance par rapport à la valeur moyenne des pixels de toute la carte. De cette manière, on peut enlever la partie de l'image supérieure à la ligne pleurale.

Etape 2. Détection des lignes B

Pour détecter les lignes B, le gradient de l'image est calculé puis utilisé pour calculer la transformée de Hilbert.

On utilise la valeur absolue de ces résultats pour calculer la transformée logarithmique, ce qui permet de générer un masque binaire soulignant les lignes B.



Etape 3. Filtre Alternate sequential

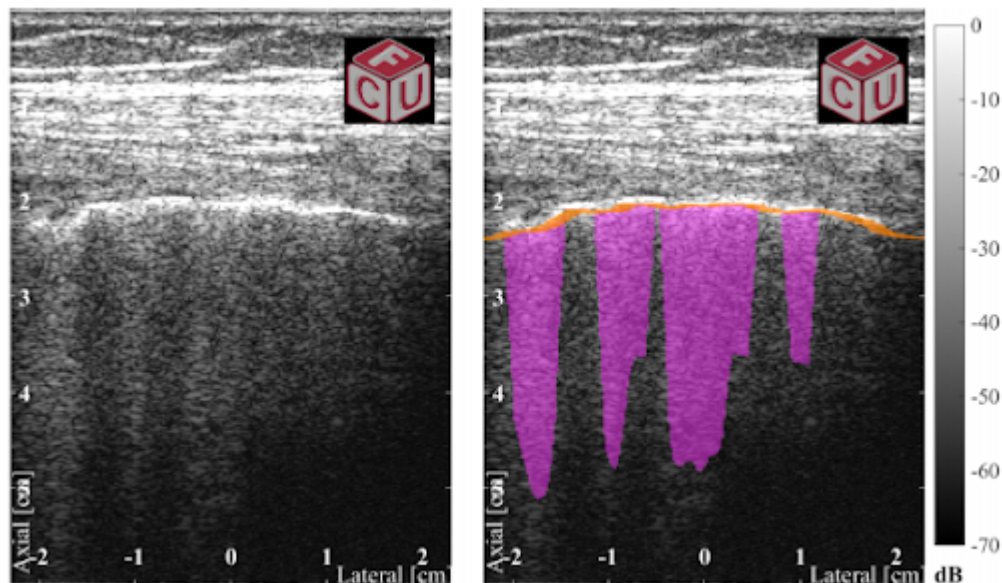
Pour affiner le résultat de ce masque binaire, on utilise un Alternate sequential filtering qui va combiner plusieurs ouvertures et fermetures morphologiques pour générer un nouveau masque. Ce masque ne va conserver que les formes allongées faisant partie des lignes B.

Cependant certaines lignes B peuvent encore être collées les unes aux autres.

Etape 4. Filtre Top-hat

Pour séparer les lignes B collées latéralement, on va utiliser la transformée Top-hat qui utilise

des ouvertures morphologiques pour extraire des formes brillantes d'un fond noir irréguliers. Ce traitement permet de générer une courbe représentant les lignes B et permettant de les afficher en surimpression sur l'image.



(a) The pleural line outlined and B-lines are overlaid on the lung scan of a patient after surgery.

10.3 Résultat

La conclusion de cette étude est que l'épaisseur moyenne de la ligne pleurale des patients atteints d'œdème des poumons est supérieur à 20 pourcent à celle des patients "normaux". L'algorithme n'a pas été testé sur d'autres vidéos en dehors de celle utilisée pour l'étude.

11 Line Detection as an Inverse Problem : Application to Lung Ultrasound Imaging

11.1 Approche

Cette méthode est extraite de l'article "Line Detection as an Inverse Problem : Application to Lung Ultrasound Imaging" du Elizabeth Blackwell Institute, University of Bristol. Il a été publié le 29 juin 2017.

Les vidéos utilisées pour l'étude sont obtenues avec un échographe SonoSite S-ICU C60 sur une fréquence de 6 à 13 MHz.

Ses vidéos sont extraites de deux études réalisées précédemment sur 23 enfants âgés de 8 à 18 ans dont 8 atteints de lésion rénale aiguë (AKI) et 15 patients atteints d'insuffisance rénale terminale (ESRD). Au début de l'étude, des images simulées sont utilisées.

11.2 Algorithme

Cette algorithme peut être divisé en 3 parties :

Etape 1 : Détection de la ligne B par un problème inverse

Dans les images tirées d'échographie, une ligne peut être représentée par l'équation $y = HCx + n$ où H représente la fonction d'étalement du point, C la transformée inverse de Radon, y représente l'image avec $y(i,j)$, n le bruit présent dans l'image et x les lignes dans le domaine de la transformée de Radon.

Le problème inverse consiste donc à trouver x ce qui peut se diviser en 2 sous-problèmes :

Problème 1) : Restaurer l'image w à partir de l'image échographique $y = Hw$ en résolvant :

$$\hat{w} = \arg \min_w \{ \|y - \mathcal{H}w\|_2^2 + \alpha \|w\|_p^p \}$$

ou $p = 1$ ou $p = 2$ si l'on se place le domaine de Laplace ou gaussien.

Problème 2) : Déterminer les lignes x appartenant au domaine de Radon en résolvant l'équation

$$\hat{x} = \arg \min_x \{ \|\hat{w} - Cx\|_2^2 + \beta \|x\|_q^q \}.$$

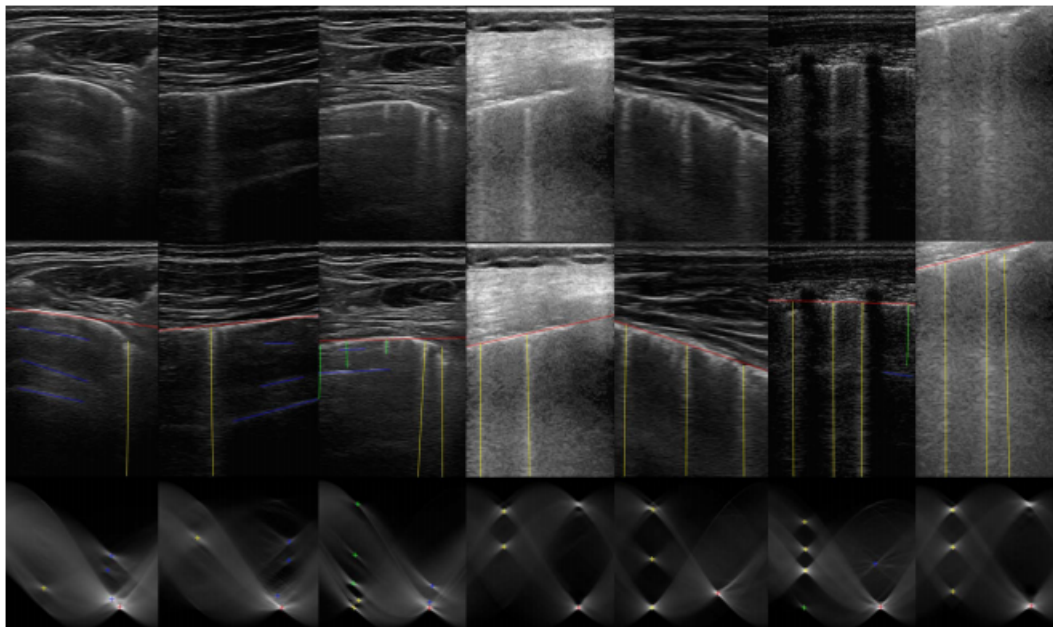
Ces deux sous-problèmes sont mis sous la forme d'une seule équation afin de réduire la complexité du calcul et éviter d'avoir à résoudre les deux problèmes l'un après l'autre.

$$\hat{x} = \arg \min_x \{ \|y - \mathcal{H}Cx\|_2^2 + \alpha \|Cx\|_p^p + \beta \|x\|_q^q \}.$$

Finalement, cette équation est résolue par l'utilisation d'un algorithme des directions alternées (alternating direction method of multipliers) qui permet d'extraire toutes les lignes présentes dans l'image échographique.

Etape 2 : Restauration des lignes obtenues

Les lignes obtenues dans l'étape 1 peuvent être floues : pour les affiner, pour les rendre plus claires, on ajoute à notre équation représentant les lignes dans une image échographique un noyau de convolution D . Ce qui change notre équation en $y = HCDx + n$ et on applique la même procédure qu'à l'étape 1.



Dans cette image, on peut observer dans chaque colonne l'image originale, les lignes détectées et l'image originale dans la transformée de Radon.

Etape 3 : Déterminer l'identité des lignes trouvées

Pour déterminer la ligne pleurale, on utilise les caractéristiques tel que l'horizontalité par rapport à l'image et son intensité lumineuse, qui est toujours supérieure à 0.75.

Pour déterminer les lignes B, on utilise leurs caractéristiques comme la verticalité par rapport à la ligne pleurale, leur longueur et le fait qu'elles ne soient pas coupées par d'autres lignes.

11.3 Résultat

La méthode utilisée a été trouvée plus performante à 50 pourcent que les autres méthodes existantes.

De plus, le temps de calcul pour trouver les lignes est de 0.45 à 0.75 secondes par image, ce qui est plus rapide que le diagnostic d'un expert qui prend en général 3 à 10 minutes selon l'image à analyser.

Quatrième partie

Analyse et Conception

Dans cette partie, on détaille l'implémentation des méthodes extraite de l'état de l'art de l'article : Novel Automatic Detection of Pleura and B-lines (Comet-Tail Artifacts) on In-Vivo Lung Ultrasound Scans.

On présente l'implémentation possible en Python et si il existe déjà des bibliothèques qui réalise ses opérations.

12 Marche aléatoire

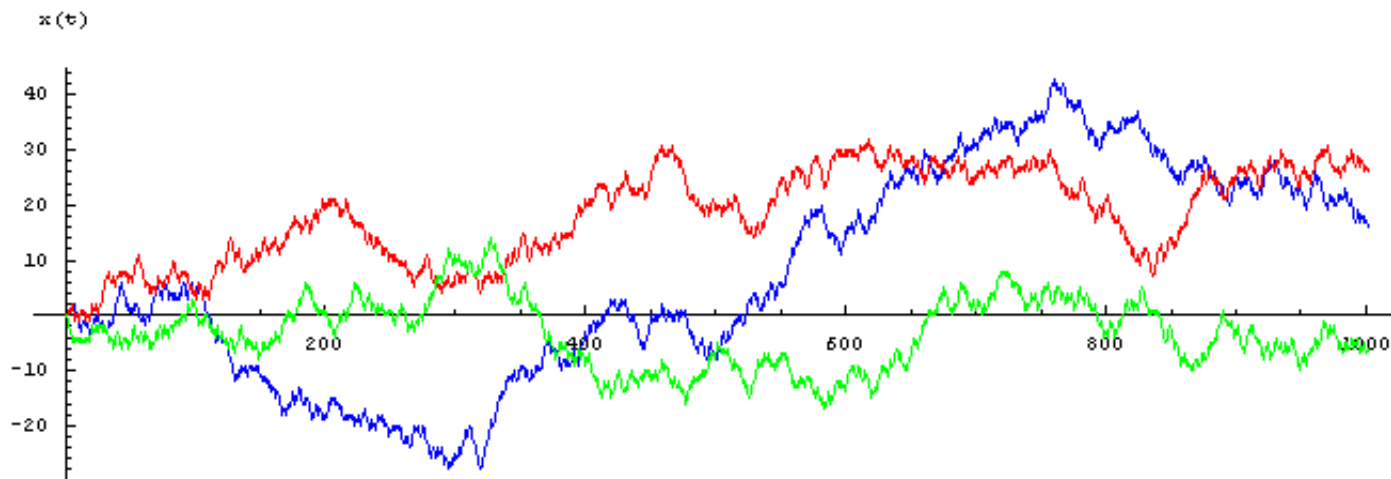
Pour détecter la ligne pleurale, on utilise une carte de confiance générée à partir d'un algorithme de segmentation appelée Marche aléatoire (Random Walker).

Une marche aléatoire est un processus aléatoire ou stochastique qui consiste à décrire un chemin dans une ou plusieurs dimensions par des étapes (ou marches) aléatoires.

12.1 Marche aléatoire à une dimension

Un exemple basique de marcheur aléatoire en 1 dimension est l'utilisation de la dimension entière en partant de zéro. Dans ce cas, le marcheur peut soit réaliser une marche vers l'avant (incrément de 1) ou une marche vers l'arrière (décrément de 1). Chaque marche est réalisée avec une probabilité aléatoire.

Un exemple plus concret est une personne sur un escalier qui tire à pile ou face pour savoir si elle monte d'une marche ou descend d'une marche. Pour chaque marche, il n'y a que deux choix.



Exemple tirée de wikipedia de 3 marches aléatoires à une dimension où la probabilité de monter ou descendre est de 0.5.

12.2 Marche aléatoire en deux dimensions

Dans le cas, d'une marche aléatoire en deux dimensions (par exemple une image) le marcheur possède 4 choix : monter, descendre, se déplacer à gauche et se déplacer à droite.

Voici une implémentation basique de cette marche en Python :

```
# Code Python pour une marche aléatoire en 2 dimensions.
```

```
import numpy
import pylab
import random
```

```
# Nombre n de marche aléatoire
n = 100000
```

```
#Création de 2 tableaux représentant les coordonnées x et y
x = numpy.zeros(n)
y = numpy.zeros(n)
```

```
# On remplit les tableaux de coordonnées avec les résultats de la marche
for i in range(1, n):
    val = random.randint(1, 4)
    if val == 1:
        x[i] = x[i - 1] + 1
        y[i] = y[i - 1]
```

```

elif val == 2:
    x[i] = x[i - 1] - 1
    y[i] = y[i - 1]
elif val == 3:
    x[i] = x[i - 1]
    y[i] = y[i - 1] + 1
else:
    x[i] = x[i - 1]
    y[i] = y[i - 1] - 1

```

Implementation tirée du site <https://www.geeksforgeeks.org/>

Dans cette implémentation, on détermine un nombre de marches puis à chaque étape on choisit aléatoirement un déplacement parmi les 4 possibilités. Chaque déplacement choisie correspond à une coordonnée dans un espace à deux dimensions qui est stocké dans un tableau d'entier 2d.

12.3 Segmentation d'une image échographique avec une marche aléatoire en deux dimensions

Pour segmenter une image échographique avec notre marcheur en deux dimensions, on introduit la notion de labels ou de marqueurs qui représentent des caractéristiques associées à notre image. Une caractéristique fondamentale d'une image échographique est l'atténuation du signal en fonction de sa profondeur. Cette caractéristique est représentée par la loi de Beer-Lambert exprimée par

$$I = I_0 \exp(-\alpha d)$$

où I_0 est l'intensité du pixel original, α est le coefficient d'atténuation et d la distance depuis la source de l'image.

On génère donc plusieurs labels dépendant de l'atténuation du signal en fonction de sa profondeur de cette manière plus notre marche aléatoire commence loin de nos marqueurs plus la probabilité qu'il l'atteigne est faible.

Cette procédure permet donc de séparer notre image en plusieurs parties en fonction de leur intensité que l'on appelle carte de confiance. Finalement, on détecte la ligne pleurale en effectuant un seuillage de la carte de confiance par rapport à la valeur moyenne de tous les pixels de la carte de confiance.

Une fois la ligne pleurale détectée, on enlève de l'image la partie supérieure à celle-ci afin de ne conserver que les lignes B.

12.4 Implementation en Python

Une bibliothèque Python appelée scikit-image implémente déjà un algorithme de marche aléatoire. On pourra donc l'utiliser pour faciliter le développement.

13 Détection des lignes B

Pour détecter les lignes B, il nous faut d'abord calculer le gradient de notre image puis utiliser ce résultat avec la transformée de Hilbert.

Finalement, on effectue une transformée logarithmique sur la transformée de Hilbert pour calculer un filtre binaire représentant les lignes B.

13.1 Calcul du gradient de l'image

Il existe plusieurs manières de calculer le gradient d'une image comme l'algorithme utilisée n'est pas précisé dans l'article, le gradient de Sobel sera utilisé. Il permet de recréer une image mettant en avant ces contours.

Le calcul du gradient de Sobel peut être résumé par l'utilisation d'un noyau (kernel) en x et en y :

1. $G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
2. $G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

On applique ensuite un produit de convolution pixel par pixel à l'image en utilisant ces noyaux pour obtenir le gradient de l'image en x ou en y.

13.2 Implémentation en Python

La bibliothèque scipy implémente déjà l'opération de convolution.

Un exemple du calcul du gradient de Sobel avec scipy :

```
#image
img= ...

# noyau en x
kx = np.array([[1,0,-1],[2,0,-2],[1,0,-1]])
# noyau en y
ky = np.array([[1,2,1],[0,0,0],[-1,-2,-1]])
# Calcul du gradient en x
x=ndimage.convolve(img,kx)
# Calcul du gradient en y
y=ndimage.convolve(img,ky)
```

13.3 Transformée de Hilbert

La transformée de Hilbert permet de transformer un signal réel dans le domaine complexe, on peut l'exprimer par la fonction suivante :

$$\hat{s}(t) = \mathcal{H}\{s\}(t) = \text{vp} \{(h * s)(t)\} = \text{vp} \left\{ \int_{-\infty}^{\infty} s(\tau) h(t - \tau) d\tau \right\} = \frac{1}{\pi} \text{vp} \left\{ \int_{-\infty}^{\infty} \frac{s(\tau)}{t - \tau} d\tau \right\}$$

Pour notre méthode de détection de ligne, on applique la transformée de Hilbert au gradient axial de notre image.

13.4 Implémentation en Python

La bibliothèque scipy implémente déjà la transformation de Hilbert.

Exemple d'utilisation de la transformée de Hilbert :

```
from scipy.signal import hilbert

hilbert_signal = hilbert(gradient_axial)
```

13.5 Transformée logarithmique

Les valeurs absolues de la transformée de Hilbert sont utilisées pour calculer la transformée logarithmique. Il s'agit simplement d'appliquer la fonction log sur ces valeurs. Ce procédé permet de compresser les données obtenues.

13.6 Implementation en Python

Pour effectuer une transformée logarithmique mais aussi calculer les valeurs absolues, on utilise la bibliothèque numpy.

Exemple d'utilisation en Python :

```
Import numpy as np
```

```
Absolute_signal = np. absolute(hilbert_signal)  
Log_transform = np.log(Absolute_signal)
```

13.7 Calcul du masque binaire

A partir de la transformée logarithmique obtenue, on crée un masque binaire enveloppant ces données. Un masque binaire définit une région de l'image et l'appliquer à l'image original permet de séparer cette région de cette image.

13.8 Implementation en Python

Une bibliothèque Python appelée opencv permet d'appliquer un masque à une image.

Exemple d'utilisation en Python :

```
img_original = cv2.imread('image_echo.jpg')  
mask = Log_transform  
result = cv2.bitwise_and(img,img,mask = mask)
```

14 Filtrage séquentiel alternatif

On utilise un filtre séquentiel alternatif (Alternate sequential filtering) sur le masque binaire afin de ne conserver que les formes allongées sur l'axe des y qui correspondent aux lignes B. Ce filtre est composé d'ouverture et de fermeture morphologique.

Une opération morphologique consiste à comparer une image avec une autre image appelé élément structurant (ou noyau) ici nos lignes B (représentées par des formes allongées sur l'axe des y). On regarde si cet élément est inclus dans l'image ou si il la touche.

Pour comprendre l'ouverture et la fermeture morphologique, il faut d'abord comprendre le principe d'érosion et de dilatation.

14.1 Erosion



L'érosion consiste à gommer les contours de l'objet au premier plan en utilisant un noyau. Par exemple, s'il l'on prend un noyau de taille 5*5 ne contenant que des chiffres 1 correspondant à la couleur blanche. On parcourt cette image, pixel par pixel avec ce noyau et on ne conserve la couleur blanche du pixel courant que si tous les pixels compris dans ce noyau possèdent la valeur 1.

14.2 Dilatation



La dilatation consiste en l'inverse de l'érosion, un pixel prend la couleur blanche si au moins un des pixels du noyau possède la valeur 1.

14.3 Ouverture morphologique



L'opération d'ouverture morphologique consiste en la succession d'une érosion par une dilatation. Elle est principalement utilisée pour enlever du bruit dans une image.

14.4 Fermeture morphologique



L'opération de fermeture morphologique est l'inverse d'une ouverture morphologique. Elle consiste en la succession d'une dilatation par une érosion. Elle est principalement utilisée pour enlever du bruit dans un objet au premier plan.

14.5 Algorithme de filtrage séquentiel alternatif

L'algorithme de filtrage proposé dans l'article :

Input: Binary mask (B_{in}) outlining the most prominent edges computed in Section 2.2 indicating the maximum size of the structuring element to be used in ASF.

Output: Binary mask (B_{ASF}) in which the strong axial B-lines are highlighted.

```

1: procedure ASF
2:    $B_{ASF} = B_{in}$ 
3:   for each  $i \in N$  do
4:     Let  $S_{lin}$  be a line structuring element of length  $N$ .
5:      $B_{ASF} = (B_{ASF} \ominus S_{lin}) \oplus S_{lin}$  # performing the morphological opening.
6:      $B_{ASF} = (B_{ASF} \oplus S_{lin}) \ominus S_{lin}$  # performing the morphological closing.
7:   end for
8: end procedure

```

Note: \oplus and \ominus denote morphological dilation and erosion respectively.

Cette algorithme prend en entrée le masque binaire calculé précédemment et la forme du noyau à utiliser pour réaliser les opérations morphologiques.

On parcourt ensuite le masque binaire et pour chaque élément du masque correspondant au noyau. On réalise une opération d'ouverture et fermeture morphologique.

Dans l'article, il n'est pas précisé la dimension du noyau ni ses valeurs. Il faudra donc dans la partie développement réaliser des tests pour déterminer le noyau obtenant les meilleurs résultats.

14.6 Implementation en Python

La bibliothèque opencv implémente les opérations morphologique.

Exemple :

```
import cv2 as cv
import numpy as np
img = cv.imread('j.png',0)
kernel = np.ones((5,5),np.uint8)
opening = cv.morphologyEx(img, cv.MORPH_OPEN, kernel)
closing = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)
tophat = cv.morphologyEx(img, cv.MORPH_TOPHAT, kernel)
```

15 Filtre Top-hat

L'utilisation du filtrage séquentiel alternatif a permis de ne conserver que les lignes B mais certaines lignes B peuvent encore être collés latéralement. Pour séparer ces lignes B, on utilise un filtre Top-hat.

Le filtre Top-hat consiste encore une fois en la succession d'opérations morphologique. Il s'agit de la différence entre l'image original et l'image obtenue par une ouverture morphologique.



La partie critique de ce filtrage est la détermination du noyau comme dans la partie filtrage séquentiel alternatif.

Cinquième partie

Mise en oeuvre

Cette partie va détailler la mise en oeuvre du projet. Notamment, les outils et librairies utilisées, l'implémentation et l'analyse des résultats.

16 Les outils et librairies utilisées

Cette section couvre les différents outils et bibliothèques utilisés pour le développement du PRD.

16.1 Environnement de développement intégré

Pour le développement de notre outil, j'ai utilisé PyCharm de JetBrains qui est l'un des IDE les plus utilisés pour développer en Python.

Il permet de gérer facilement les environnements virtuels Python afin de ne pas utiliser que les librairies nécessaires au projet. On peut en effet changer en un clic de version de Python ainsi que d'environnements virtuels ce qui a été pratique car certaines des bibliothèques dont j'avais besoin n'étaient pas compatibles avec certaines versions de Python (préciser en annexe).

De même, il intègre dans son interface graphique la gestion du versionning avec git et permet de le synchroniser avec Github ou Gitlab.

16.2 OpenCv

OpenCV est une bibliothèque de traitement d'images développée par Intel. Elle est à la base du développement en C++ et la version Python n'est qu'un wrapper. Ce qui permet de conserver la puissance du C++ pour le traitement d'images tout en utilisant Python pour sa rapidité pour le développement.

Pour notre projet, elle est utilisée pour réaliser des sauvegardes des images au format TIFF sans compression, pour le calcul du gradient axial, les opérations morphologiques ainsi que la création de masques.

16.3 Pydicom

Pydicom est une bibliothèque Python open-source permettant de gérer le format de fichier DICOM (Digital Imaging and Communications in Medicine). Ce format de fichier est utilisé par tous les appareils médicaux que ce soit en radiologie, radiothérapie, cardiologie et ophtalmologie.

Dans notre cas, elle est utilisée pour lire le fichier DICOM extrait de l'échographe contenant les images à analyser.

16.4 QT

QT est une bibliothèque de création d'interface graphique. Comme OpenCV, elle est à la base développée en C++ et la version Python n'est qu'un wrapper.

Il s'agit d'une bibliothèque multiplate-forme qui s'intègre parfaitement à Python et régulièrement mise à jour.

Dans notre projet, je l'utilise pour réaliser la partie vue de l'interface graphique. De plus elle possède la particularité de fonctionner avec un pattern design Modèle-Vue, le contrôleur étant intégré dans la vue.

16.5 PyInstaller

Pyinstaller est un outil Python permettant de créer un exécutable Windows à partir du code source d'un projet. Il analyse l'environnement virtuel utilisé pour déterminer les bibliothèques à inclure ainsi que toutes les dépendances externes (ex : icône ou fichier de ressource). Il embarque également un interpréteur Python ce qui évite à l'utilisateur d'installer Python sur sa machine.

16.6 Inno Setup

Inno Setup est un outil créé pour réaliser des installateurs Windows.

Dans notre projet, il est utilisé pour fournir un installateur afin d'éviter à nos utilisateurs de manipuler l'exécutable directement.

En effet, il installe le programme à l'emplacement choisi par l'utilisateur et crée même un raccourci bureau et raccourci startup.

16.7 Autres bibliothèques

1. Scipy est une bibliothèque Python contenant de nombreux outils mathématiques. Elle est utilisée pour réaliser la transformée de Hilbert.
2. PIL est une bibliothèque de traitement d'image. Elle est utilisée en complément d'OpenCv pour réaliser des conversions d'images.

17 Implémentation

17.1 Algorithme de détection des lignes B

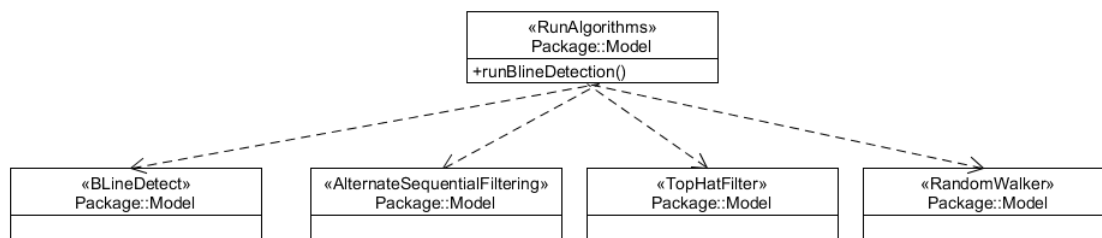
Dans la partie analyse et conception, on détaille l'algorithme en 4 sous-parties :

- Détection de la zone de la ligne pleurale
- Détection des lignes B
- Filtre alternatif séquentiel
- Filtre top-hat

J'ai donc décidé de diviser ces 4 parties en 4 classes contenues dans le package Model.

De plus, j'utilise une cinquième classe RunAlgorithms permettant de lancer les 4 classes dans l'ordre correct d'exécution.

On peut donc résumer la partie Algorithme de détection des lignes B par ce diagramme de classes :

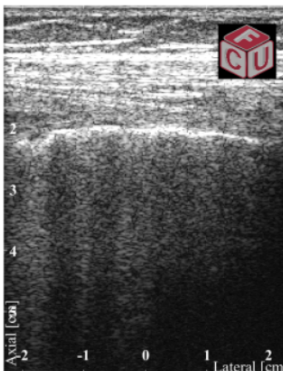


17.1.1 Détection de la zone de la ligne pleural : Modification

Dans la première partie de l'algorithme, il était prévu d'utiliser un random walker pour détecter la zone de la ligne pleurale et l'enlever de l'image afin de ne conserver que la partie inférieure à cette ligne.

Cependant dans l'étude utilisée pour concevoir l'algorithme les images sont pré-découpées en largeur à la main pour ne contenir que la partie de l'image avec la ligne pleurale et les lignes B.

Exemple d'image utiliser dans l'étude :



Exemple d'image obtenue par l'hôpital de tours :

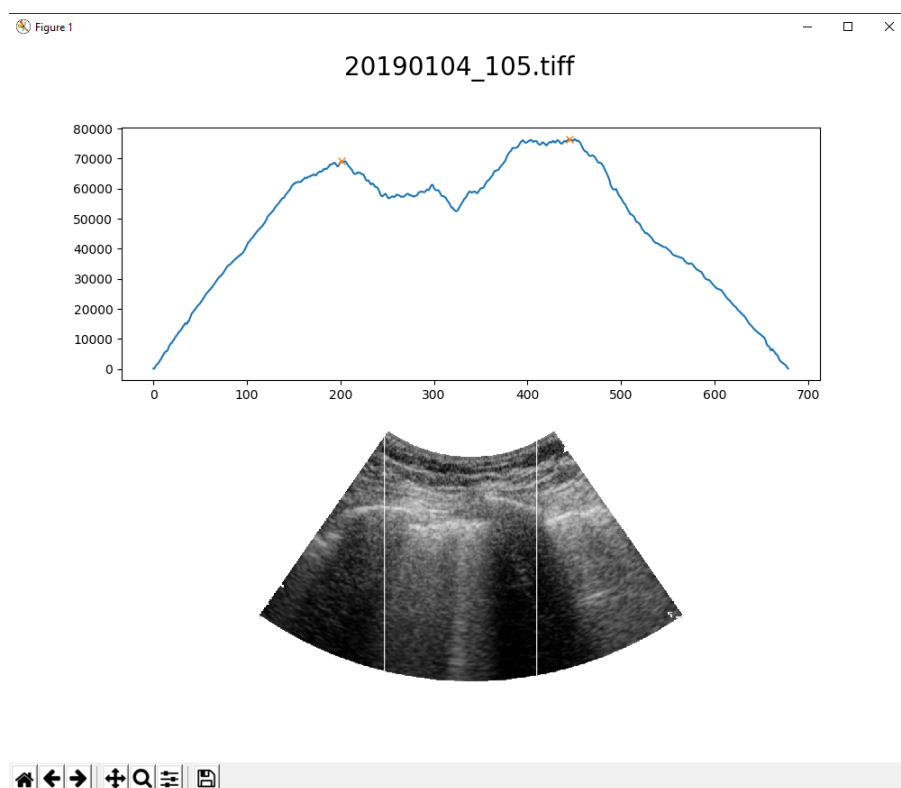


Dans notre cas, on peut observer que nos images ne sont pas pré-découpées, il faut donc tenter de les découper automatiquement nous-même.

Il faut donc enlever les parties sombres correspondant aux côtes dans l'image. Pour cela, on réalise un histogramme axial de l'intensité des pixels de l'image afin de ne conserver que les parties correspondantes aux lignes B et à la ligne pleurale.

Ensuite, on recherche les deux maximums locaux de l'image qui nous permettent de choisir les points de découpe optimaux.

Exemple d'histogramme générer :



Les deux croix orange représentent les deux maximums locaux trouvés dans l'histogramme et les deux lignes blanches représentent la découpe qui sera effectuée.

Mais comme on peut le remarquer cette découpe est loin d'être parfaite et n'est pas utilisable pour toutes les images.

Finalement, j'ai totalement changé de méthode pour la détection de la ligne pleurale et j'ai réalisé un outil de sélection dans l'image pour les médecins.

Il n'y a donc plus d'utilité à utiliser le random walker puisque cette opération de détection est réalisée manuellement par les médecins.

L'implémentation de cette méthode est détaillée dans la partie GUI.

17.2 Détection des lignes B

La détection des lignes B se fait en plusieurs étapes :

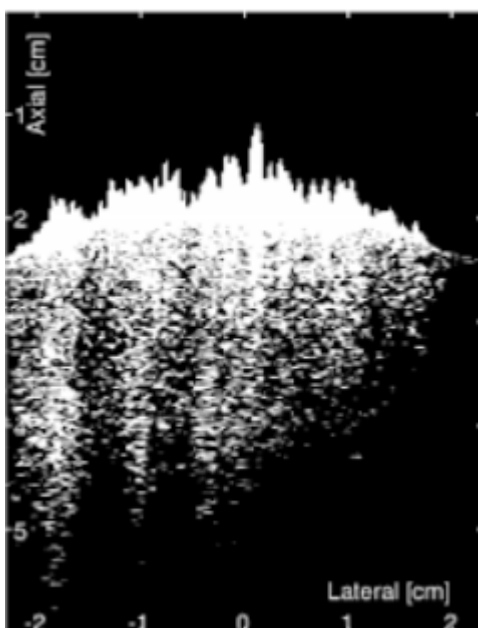
- Calcul du gradient axial de l'image
- Transformée de Hilbert du gradient
- Transformée logarithmique de la transformée de Hilbert
- Création d'un masque binaire avec la transformée logarithmique

Ces étapes sont divisées en fonction dans la classe BLineDetect du package model.

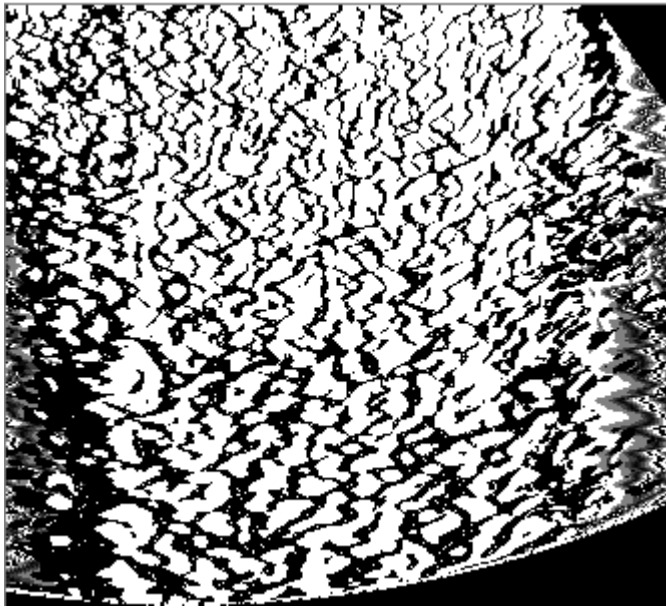
L'implémentation de ces méthodes n'a pas posé de problèmes cependant les résultats de cette étape de filtrage ne sont pas concluants.

En effet dans l'étude on obtient un masque qui contient les lignes B.

Exemple de résultat de l'étude :



Exemple de résultat avec notre filtrage :



Le masque obtenu n'est pas tout identique à celui de l'étude. Ceci est sûrement dû à la différence du bruit présent dans les images, aux différentes méthodes d'acquisition utiliser et aux méthodes de pré-filtrage de l'image.

17.2.1 Détection des lignes B : modifications

Pour tenter de corriger les résultats obtenues, j'ai essayé différentes méthodes.

Tout d'abord en changeant les valeurs du noyaux de convolution lors du calcul du gradient.

Les différentes valeurs du noyaux utiliser :

- $(1, 2, 1), (0, 0, 0), (-1, -2, -1)$
- $(-1, 0, 1), (-2, 0, 2), (-1, 0, 1)$
- $(1, 1, 1), (0, 0, 0), (-1, -1, -1)$

Malheureusement, aucune de ces valeur de noyaux n'a donner de meilleurs résultats.

J'ai ensuite essayer d'enlever du bruit avec des fonction de débruitage présente dans OpenCV.

Il y a deux fonction de débruitage dans OpenCv :

La première `fastNlMeansDenoising` prend l'image à débruiter, la taille de fenêtre de recherche et un entier de 0 à 10 déterminant la puissance du filtre.

La deuxième `fastNlMeansDenoisingMulti` utilise plusieurs images de la même séquence pour enlever le bruit avec les même paramètres.

Image original :

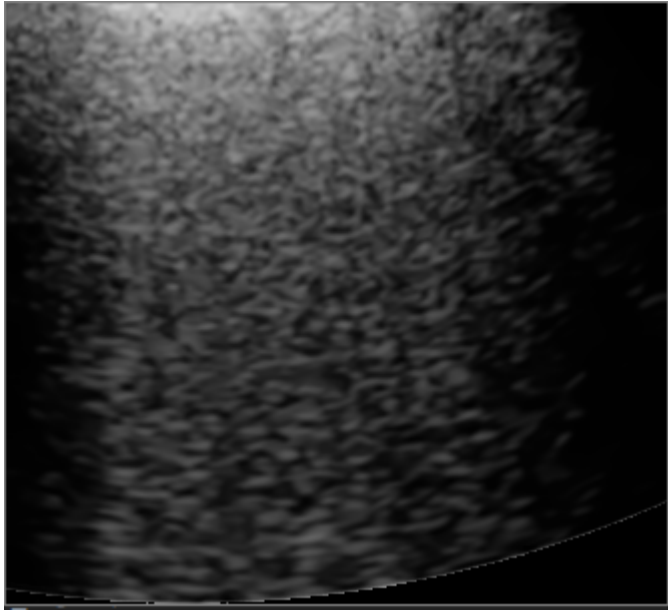


Image débruiteur avec fastNlMeansDenoising :

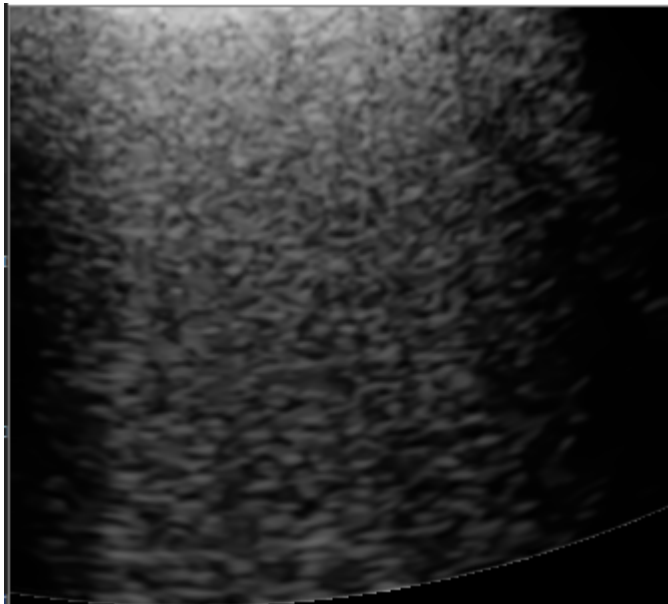
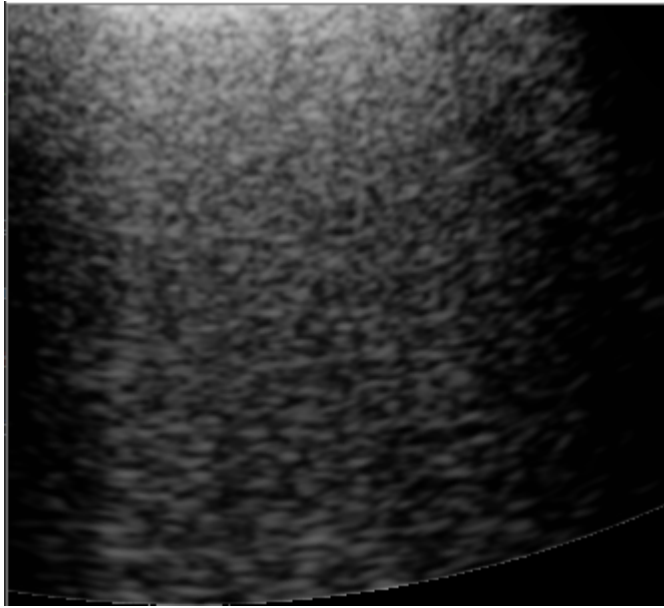


Image débruiteur avec fastNlMeansDenoising :



On observe que la différence est assez dure à voir mais que les lignes B sont plus effacées avec ces techniques de débruitage car en effet les lignes B sont par définitions du bruits.

Il n'est donc pas pertinent de tenter d'enlever du bruits dans les images car ce bruits contient les informations les plus importantes.

17.2.2 Détection des lignes B : conclusion

Malheureusement, il semble que la méthode choisie dans l'étude ne soit pas adapter pour nos images échographique. Malgré l'utilisation de différentes méthodes pour corriger ces résultats, il n'a pas était possible de trouver de meilleur solution.

Ces mauvais résultats empêche de vérifier si l'implémentation des deux classes contenant le Alternate sequential filtering et le filtre Top-hat sont fonctionnelles.

Ils empêchent également d'obtenir un résultat sur le nombres de lignes B détecter.

17.3 GUI

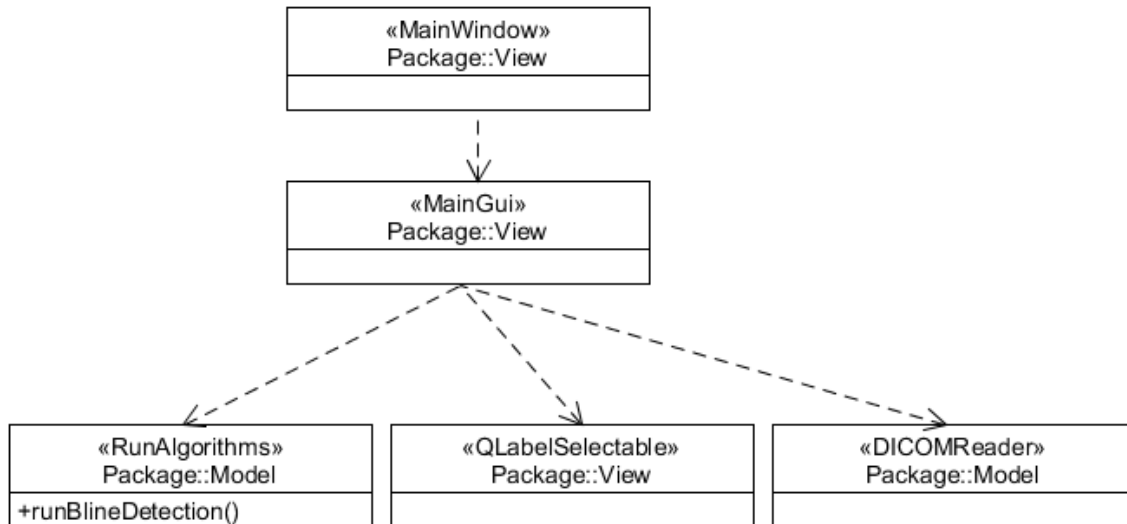
Le GUI est réaliser à partir de la bibliothèque QT5. Il est composé principalement de la classe MainWindow, MainGui et de la classe QLabelSelectable.

La classe MainWindow contient les différents menus de la fenêtre principale ainsi que des paramètres d'affichage.

Elle encapsule la classe MainGui qui contient les différents boutons, affichage des images échographiques, l'instance de la classe permettant de lire les fichiers DICOM ainsi que l'instance de la classe RunAlgorithms.

Dans le paradigme de QT, la vue agit comme un contrôleur la classe MainGui gère donc le fonction du programme.

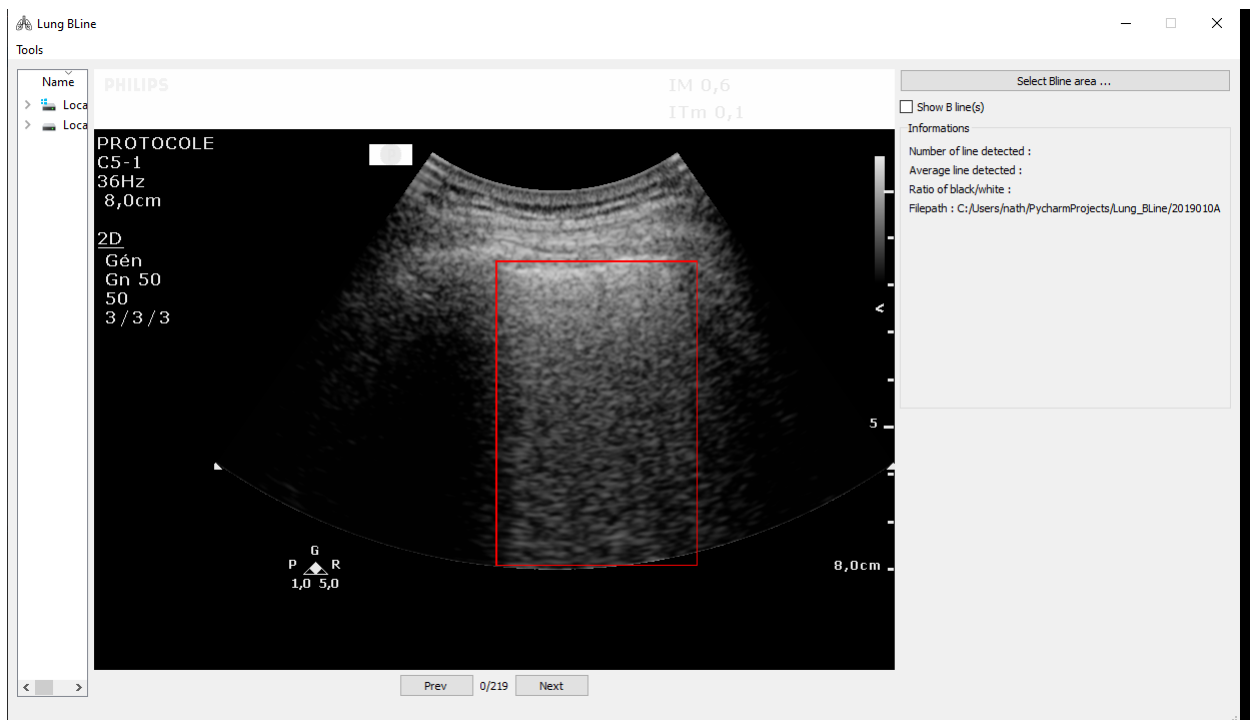
On peut résumer son fonctionnement avec ce diagramme de classe :



17.3.1 Classe QLabelSelectable

La classe QLabelSelectable implémente la possibilité de sélectionner une partie de l'image pour déterminer la zone d'intérêt contenant les lignes B.

On commence une sélection par un clic sur le bouton "Select BLine area".



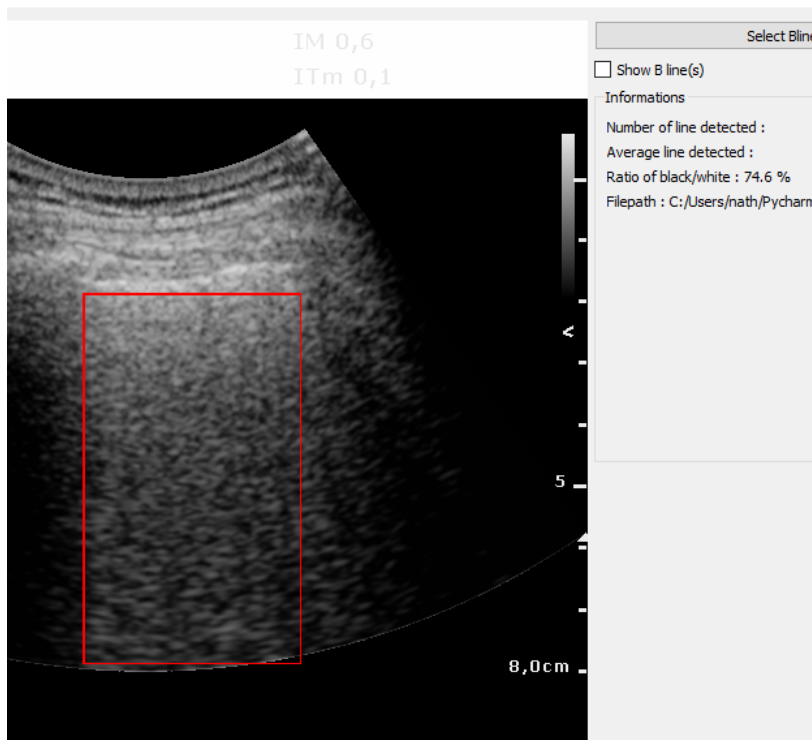
La zone sélectionnée est ensuite transmise par le MainGui à la classe RunAlgorithms pour effectuer les opérations de filtrage permettant de détecter les lignes B et de calculer le pourcentage de pixel blanc.

17.4 Calcul du pourcentage de pixel blanc

Pour déterminer le ratio de pixels blanc par rapport aux pixels noir, on compte le nombre de pixel ayant une valeur supérieur à un seuil prédéfinie (dans cette exemple, on choisit de considérer que tous les pixel ayant une valeur supérieur à 40 sont considérés comme blanc).

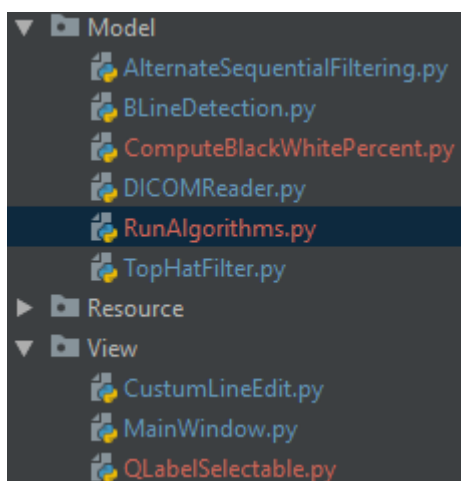
Puis on divise ce nombre de pixels par rapport aux nombres total de pixels dans l'image sélectionner.

Exemple de résultat pour le calcul du ratio de pixels blanc :



17.5 Modularité du programme et ré-utilisation

Lors du développement du programme, j'ai particulièrement fait attention à ce qu'il soit le plus modulaire possible afin d'assurer une possible reprise par des étudiants des années suivantes.



En effet le programme est divisé en module séparant la vue-contrôleur et le modèle de cette manière, il est possible d'ajouter de nouveaux algorithmes de détection.

De plus l'algorithme de détection des lignes B développé durant ce PRD est lancé par la classe RunAlgorithms.

Il est possible d'ajouter une nouvelle fonction run à RunAlgorithms pour ajouter de nouveaux algorithmes et de créer de nouvelles classes de filtrage dans le package model.

```
class RunAlgorithms:
    """
    This class is used to run BLine detection algorithms.
    """

    def runBlineDetection(self, img):
        """
        This function takes the selected image from the user and run the bline detection algorithm.
        It return the bline detected.
        """
        bline_detect = BLineDetect(img)
        binary_mask = bline_detect.get_binary_mask()
        asf = ASF(binary_mask)
        top_hat_filter = TopHatFilter(asf)

        return top_hat_filter

    def runAnotherBlineDetection(self, img):
        pass

    def runAnotherAnotherBlineDetection(self, img):
        pass
```

Il suffit de modifier une ligne de code de la fonction startFiltering dans le MainGui pour lancer une nouvelle fonction run de la classe RunAlgorithms.

```
def startFiltering(self):
    """
    This function is used to call filtering function when the bline area has been selected.
    The variable to use is self.selectedPixmap
    """
    if self.qcheckbox_show_filter.isChecked():
        # start bline detection and show bline
        run = RunAlgorithms()
        run.runBlineDetection(self.selectedPixmap)
```

18 Risques

Il existe des risques sur les choix que j'ai pu faire lors de l'implémentation.

Notamment la sélection manuelle de la zone d'intérêt contenant les lignes B peut mener à des erreurs humaines d'appréciation de cette zone.

Mais ce risque est contrebalancé par le fait que des médecins vont utiliser le logiciel et que leurs niveaux d'expertise minimisent ce risque.

Le deuxième risque présent est lors du calcul du ratio de pixels blancs par rapport aux pixels. En effet, le choix d'une valeur de seuil arbitraire pour déterminer si un pixel est blanc ou non ne permet d'assurer une valeur de ratio exacte.

Sixième partie

Bilan et conclusion

19 Fait au S9

Les objectifs réalisés au S9 :

1. Rédaction du cahier de spécification
2. Rédaction de l'état de l'art
3. Choix d'une méthode pour détecter la ligne pleurale et les lignes B
4. Prototype pour ouvrir les fichiers Dicom et en extraire les images
5. Prototype pour déterminer la zone d'intérêt à exploiter par la méthode de détection de ligne
6. Modifier le cahier de spécification pour y inclure les ajouts des encadrants
7. Finir de rédiger la partie Analyse et Conception

20 Reste à faire au S9

Reste à faire au S9 :

1. Ajouter une table des figures et résoudre le bug empêchant d'utiliser des figures
2. Au niveau des spécifications, la fonction save-result doit être modifiée pour sauvegarder également les images contenant les lignes détectées

21 Fait au S10

Les objectifs réalisés au S10 :

1. Rédaction de la partie mise en œuvre
2. Rédaction de la doc d'utilisation
3. Rédaction de la doc d'installation
4. Documentation de toutes les classes
5. Développement de l'algorithme de détection des lignes B
6. Développement de l'algorithme de calcul du ratio de pixels blancs par rapport aux pixels noirs
7. Développement du GUI pour utiliser l'algorithme
8. Mise en place d'un git
9. Modularité du programme pour assurer une future reprise du projet

22 Reste à faire au S10

Reste à faire au S10 :

1. Développer la fonction save-result
2. Aller plus loin : Trouver d'autres méthodes de détection adapter aux images échographiques

23 Planning

La plupart des objectifs du planning ont été réalisés dans les temps. Sauf pour la partie Analyse et Conception ce qui à entraîner un retard sur la création du planning du S10.

Pour le S10, tous les objectifs en termes de développement ont était atteint malgré un manque de résultats satisfaisant.

Ce manque de résultats a empêché d'atteindre l'objectif d'afficher le nombre de lignes B présentes dans la vidéos.

Malheureusement, il à était impossible de comparer les résultats obtenus avec l'autre PRD ainsi que de se rendre à l'hôpital pour montrer les résultats aux médecins à cause de l'épidémie de coronavirus.

24 Bilan sur la qualité

Pour la qualité de l'algorithme de détection des lignes B, ce n'est pas très bon car il n'est pas adapté pour nos images échographiques. Malgré de nombreuses tentatives avec différentes méthodes pour tenter de résoudre les problèmes, il n'a pas été possible d'atteindre l'objectif final d'afficher les lignes B détecter et leurs nombre.

Pour la qualité du programme, je pense qu'elle est très bonne car j'avais bien préparé le cahier de spécification ainsi que la modélisation.

J'ai également assuré la possible reprise du projet grâce à une modularité des classes du projet, veiller à écrire du code lisible et toujours documenter.

De plus l'auto installer que j'ai généré permet une utilisation très facile du programme.

25 Bilan auto-critique sur la gestion du projet

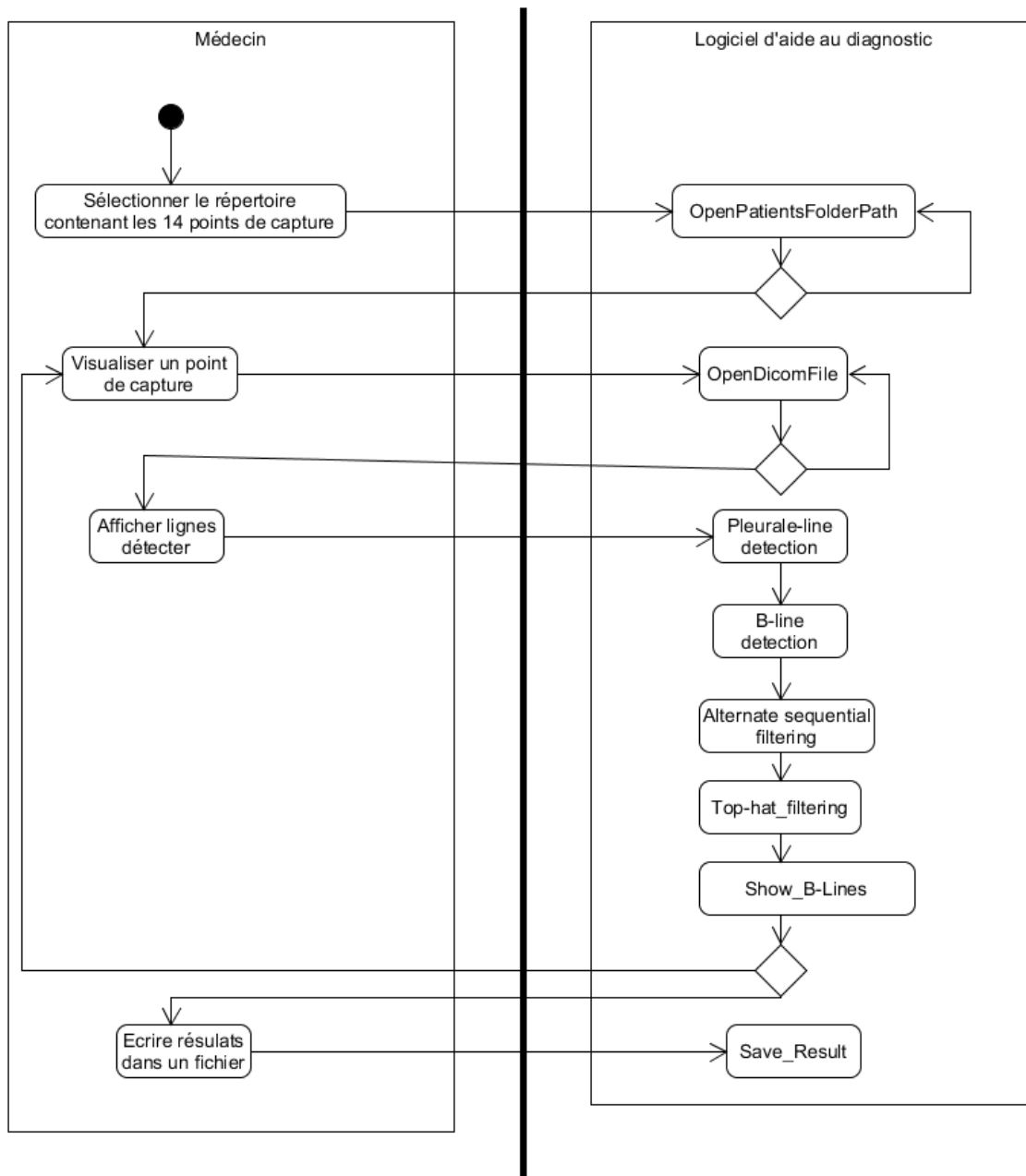
Pendant cette année, je pense avoir bien avancé dans mon projet et bien suivie le planning que je m'étais fixé. Même si je suis un peu déçu de ne pas avoir obtenu de résultats fiables concernant la détection des lignes B. Ce manque de résultats est peut-être due au fait que j'ai insisté trop longtemps pour continuer avec la méthode choisie et essayée de la faire fonctionner.

Ainsi qu'à un manque de communication de ma part avec mon encadrant à la moitié du S10, plus communiquer avec lui m'aurait certainement permis de me rendre compte plus tôt qu'un changement de méthode de détection des lignes était nécessaire.

Finalement lors de ce PRD, j'ai beaucoup appris sur la gestion d'un projet à long terme, découvert de nouvelles technologies et de nouvelles manières d'appréhender la résolution d'un problème.

Annexes

A

Spécifications fonctionnelles
Détailé

Ce diagramme d'activité résume les interactions possible de l'utilisateur avec le système. Le médecin peut choisir le répertoire contenant les différents fichiers échographique. Il peut ensuite choisir de visualiser un point de capture ce qui entraîne l'appel de la fonction OpenDicomFile. Si il choisit d'afficher les lignes détecter cela déclenche la procédure de détection de ligne. Finalement, il peut choisir de sauvegarder les résultats obtenus dans un fichier.

1 Définition de la fonction OpenDicomFile

1.1 Rôle

Cette fonction a pour but d'ouvrir un fichier Dicom contenant les images d'un point de capture et de renvoyer une liste d'images sous format TIFF non compressées.

1.2 Description


Entrée	Chemin vers le fichier Dicom
Sortie	Liste d'images contenue dans le fichier Dicom
Priorité	Primordial
Pré-condition	Chemin valide vers le fichier Dicom
Post-condition	N/a
Gestion des erreurs	Une exception est levée si le fichier n'existe pas, ne respecte pas le format ou est corrompue.
Interaction avec des composants	Utilisé par la classe FileManager et correspond au cas d'utilisation "Visualiser un point de capture" .

2 Définition de la fonction OpenPatientsFolderPath

2.1 Rôle

Cette fonction permet de déterminer l'emplacement du dossier où est contenu l'arborescence comme présenté dans la partie Fonctionnalité du système afin de pouvoir à terme l'afficher dans l'interface utilisateur.

2.2 Description

Entrée	Chemin vers le répertoire
Sortie	Arborescence du dossier contenant les chemins vers les dossiers patients, leurs sous dossiers trimestriels ainsi que les fichiers des 14 points de captures. 
Priorité	Primordial
Pré-condition	Chemin valide vers le répertoire
Post-condition	Il faut s'assurer que les chemins des fichiers soit bien aux format Dicom
Gestion des erreurs	Une exception est levée si aucun répertoire n'est choisie ou ne contient pas de fichier dicom.
Interaction avec des composants	Utilisé par la classe FileManager et correspond au cas d'utilisation "Sélectionner le répertoire contenant les 14 points de capture".

On remarque ici que nos post conditions ne vérifient pas que l'arborescence est respectée car il est possible que la capture de tous les points ne soit pas réalisée ou que l'utilisateur souhaite seulement consulter un unique fichier.

3 Définition de la fonction Pleural-line detection

3.1 Rôle

Le but de cette fonction est de réaliser une carte de confiance des images extraites d'un point de capture.

Cette carte de confiance a pour but de mettre en évidence la ligne pleurale et permettre de la tracer. L'image échographique est découpé à partir de la ligne pleural détecter par la fonction Pleural-line detection.

L'algorithme utilisé est un Random Walker modifié prenant en compte l'atténuation du signal de l'échographe en fonction de la profondeur (Détailé dans le document Analyse et Conception).

3.2 Description

Entrée	Image échographique
Sortie	Map de confiance sous forme d'image, Image original découpé à partir de la ligne pleurale
Priorité	Primordial
Pré-condition	N/a
Post-condition	N/a
Gestion des erreurs	Une exception est levée si une erreur apparaît lors du calcul de la carte de confiance permettant d'identifier quelle étape de la méthode est en erreur.
Interaction avec des composants	Utilisé par la classe LineDetector et correspond au cas d'utilisation "Détection ligne pleurale".

4 Définition de la fonction B-line detection

4.1 Rôle

Cette fonction a pour but de détecter et localiser les lignes B en utilisant la carte de confiance déterminée par la fonction ConfidenceMap.

Comme les lignes B sont caractérisées par leur départ de la ligne pleurale vers la fin (le bas) de l'image échographique, on peut exclure les données de l'image supérieure à la ligne pleurale.

La détection des lignes B est réalisée par l'utilisation de la transformée de Hilbert puis, à partir des résultats fournis, un masque binaire est calculé (Détailé dans le document Analyse et Conception).

Ce masque binaire permet d'afficher les lignes B les plus visibles mais des artéfacts non désirés peuvent apparaître parfois selon les cas.

4.2 Description

Entrée	Image échographique
Sortie	Un masque binaire sous forme d'image
Priorité	Primordial
Pré-condition	N/a
Post-condition	N/a
Gestion des erreurs	Une exception est levée si une erreur apparaît lors du calcul du masque binaire permettant d'identifier quelle étape de la méthode est en erreur.
Interaction avec des composants	Utilisé par la classe LineDetector et correspond au cas d'utilisation "Détection lignes B".

5 Définition de la fonction Alternate sequential filtering

5.1 Rôle

Cette fonction va utiliser des ouvertures et des fermetures morphologiques appliquées au masque binaire afin de ne conserver que les éléments correspondant aux caractéristiques d'une ligne B et éliminer les artéfacts (Algorithme détaillé dans le document Analyse et Conception).

5.2 Description

Entrée	Le masque binaire généré par la fonction B-line detection
Sortie	Un masque ASF sous forme d'image
Priorité	Primordial
Pré-condition	N/a
Post-condition	N/a
Gestion des erreurs	Une exception est levée si une erreur apparaît lors du calcul du masque binaire ASF permettant d'identifier quelle étape de la méthode est en erreur.
Interaction avec des composants	Utilisé par la classe LineDetector et correspond au cas d'utilisation "Détection lignes B".

6 Définition de la fonction Top-hat filtering

6.1 Rôle

La fonction Top-hat filtering va utiliser le masque ASF généré par la fonction Alternate sequential filtering afin de séparer les lignes B les unes des autres et les identifier clairement.

Cette fonction va combiner des transformations White Top-hat et Black Top-hat qui consistent en des opérations d'ouvertures et de fermetures morphologiques (Algorithme détaillé dans le document Analyse et Conception).

6.2 Description

Entrée	Le masque binaire généré par la fonction B-line_detection
Sortie	Un masque Top-hat sous forme d'image, Nombre de ligne détecter, Pourcentage de blanc par rapport au noir sous la zone de ligne pleural
Priorité	Primordial
Pré-condition	N/a
Post-condition	N/a
Gestion des erreurs	Une exception est levée si une erreur apparaît lors du calcul du masque Top-hat permettant d'identifier quelle étape de la méthode est en erreur.
Interaction avec des composants	Utilisé par la classe LineDetector et correspond au cas d'utilisation "Détection lignes B".

7 Définition de la fonction Show B Lines

7.1 Rôle

Cette fonction va utiliser le masque généré par la fonction Top-hat filtering pour afficher en surimpression les lignes B détectées sur l'image ainsi que la ligne pleurale.

7.2 Description

Entrée	Le masque Top-hat et la carte de confiance
Sortie	Affichage en surimpressions des lignes b et de la ligne pleural sur une image.
Priorité	Secondaire
Pré-condition	N/a
Post-condition	N/a
Gestion des erreurs	N/a
Interaction avec des composants	Utilisé par la classe GUI et correspond au cas d'utilisation "Afficher lignes détecter".

8 Définition de la fonction Save Result

8.1 Rôle

Cette fonction sauvegardera dans un fichier au format csv les résultats obtenus sur un point de capture ainsi que la moyenne des lignes détectées par image.

8.2 Description

Entrée	Un tableau contenant le nombre de ligne B détectés par image sauvegarder dans la classe FileManager et le répertoire choisie pour la sauvegarde.
Sortie	Un fichier csv
Priorité	Secondaire
Pré-condition	N/a
Post-condition	Le répertoire choisie pour la sauvegarde doit être valide.
Gestion des erreurs	Une exception est levée si une erreur apparaît lors de la génération du fichier.
Interaction avec des composants	Utilisé par la classe FileManager et correspond au cas d'utilisation "Ecrire résultat dans un fichier".

B

Spécifications non fonctionnelles

1 Contraintes de développement et conception

Le langage choisi pour ce projet est Python 3.7 pour les nombreuses bibliothèques proposées dans le domaine du traitement d'images.

Liste exhaustive des bibliothèques à utiliser :

1. QT5 : pour créer l'interface graphique
2. Pydicom : pour lire les fichiers DICOM de l'échographe
3. OpenCV et Scipy : pour effectuer les traitements d'images de notre méthode
4. Numpy : pour gérer facilement des matrices de pixels

2 Contraintes de fonctionnement et d'exploitation

Les contraintes de fonctionnement et d'exploitation se divisent en 4 parties, les Performances, Capacités, Contrôlabilité et Sécurité.

2.1 Performances

Pour l'utilisateur, le traitement d'une vidéo échographique contenant environ 300 images, la méthode choisie peut être coûteuse en temps de calcul ; il faut donc s'assurer que le temps d'attente ne dépasse pas les 2s par image afin de conserver une certaine fluidité.

La rapidité du traitement dépendra du CPU de l'ordinateur utilisé.

Pour vérifier que le temps de traitement est bien respecté, on effectuera des mesures de temps. Dans le cas, où ce temps serait trop élevé, il faudra penser à optimiser le code peut-être par du multi-threading.

2.2 Capacités

Le système devra être en capacité de traiter chaque image de la vidéo avec la méthode de détection de ligne afin de pouvoir afficher le nombre moyen de lignes détectées ou le pourcentage moyen de pixels blancs par rapport aux pixels noirs.

2.3 Contrôlabilité

Pour contrôler le bon déroulement de l'algorithme, un fichier de log sera créé afin de lister les différentes étapes de l'algorithme et aussi le nombre de ligne comptées dans chaque image ou le pourcentage de pixels blancs/noirs.

Afin de faciliter la prise en main de l'application par l'utilisateur, des fenêtres sous forme de pop-up seront utilisées pour prévenir l'utilisateur d'une erreur. Par ailleurs le nombre de lignes détectées sera intégrées dans la vue principale du logiciel.

2.4 Sécurités

Étant donné le fait qu'il n'est prévu qu'un seul utilisateur par poste, il n'est pas prévu de mettre en place une procédure de login.

Du point de vue des fichiers médicaux, leur confidentialité n'est pas remise en cause car l'application n'est amenée à aucun moment à faire des modifications sur le contenu (entête non extrait) et encore moins d'exportation en dehors de la machine qui héberge les dites images.

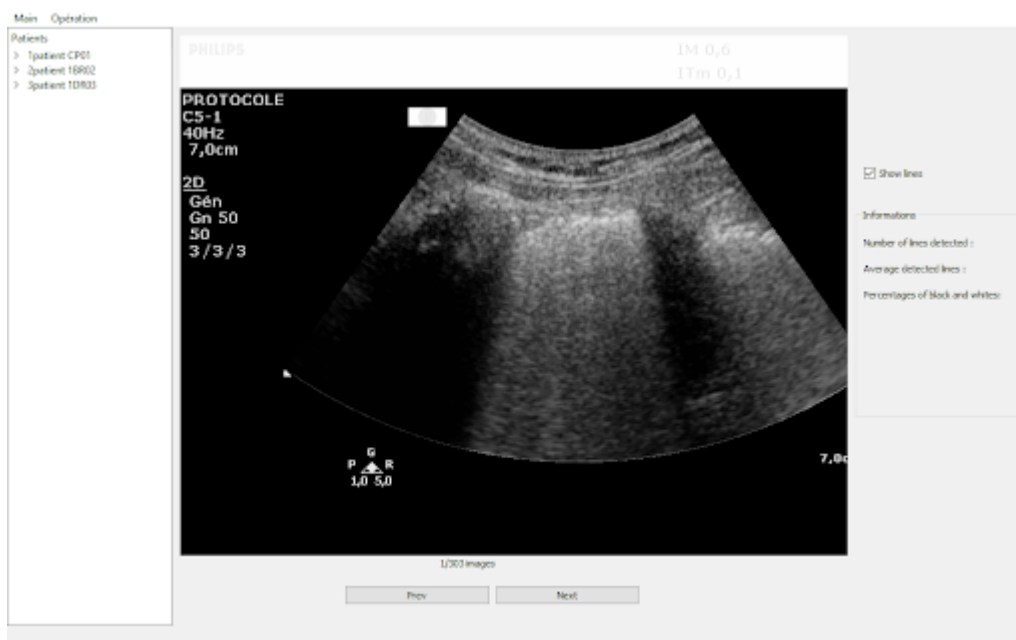
C

Description des interfaces externes du logiciel

1 Interfaces matériel/logiciel

L'interfaçage de l'échographe avec notre application est réalisé par l'intermédiaire des fichiers DICOM généré par les points d'observations lors du déplacements de la sonde échographique.

2 Interfaces homme/machine



L'utilisation de l'interface homme/machine se veut la plus simple possible afin de faciliter son appropriation par les médecins qui n'ont pas ou peu de connaissances en informatique.

Comme décrit dans les fonctionnalités du système, l'utilisateur peut sélectionner le répertoire contenant les points de captures.

Celui-ci s'affiche dans un bandeau à part, un double-clic sur l'un des points de captures provoque l'affichage dans la fenêtre principale de la vidéo.

Par ailleurs, grâce à un simple clic d'un bouton, l'utilisateur peut afficher les lignes détectées par notre méthode de détection de lignes développée.

Par ailleurs, un champ texte affiche la moyenne du nombre de lignes détectées sur toute la vidéo mais aussi la moyenne du nombre de lignes sur l'ensemble des points de captures.

La réalisation du calcul de la moyenne du nombre de lignes détectées sur toute la vidéo est effectué en tâche de fond car elle peut prendre un certain temps.

Un bouton permet de changer l'affichage pour indiquer le pourcentage de pixels blancs par rapport au nombre de pixels noirs.

3 Interface logiciel/logiciel

Afin d'assurer un déploiement relativement aisé de l'application, nous utiliserons l'application PyInstaller afin de générer un exécutable qui sera remis au client (le médecin).



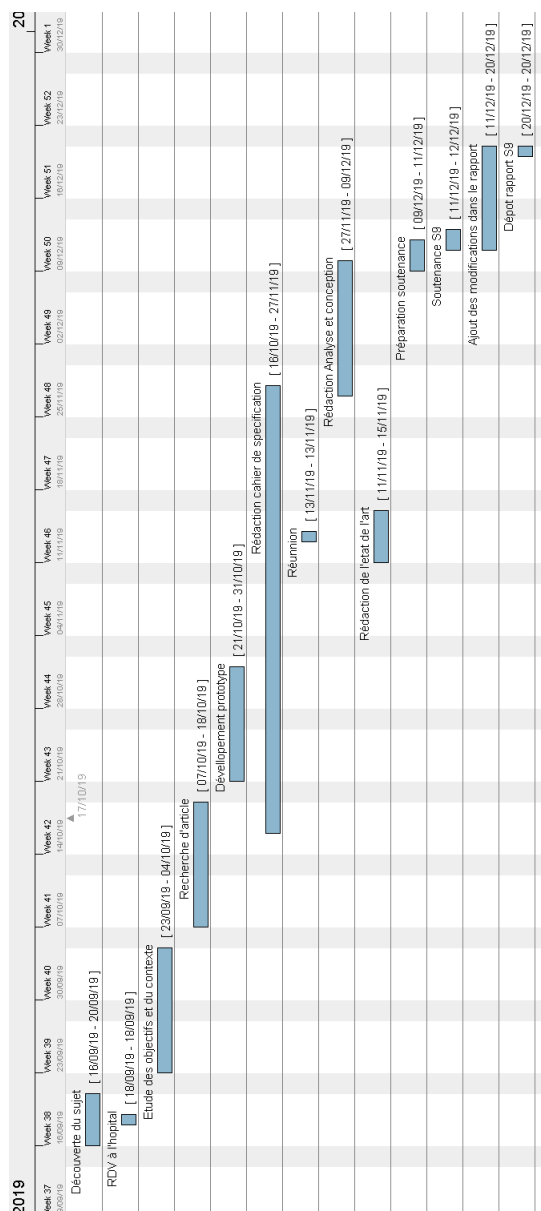
Ce choix est justifié par le fait que PyInstaller permet d'embarquer dans un exécutable Windows à la fois un interpréteur Python ainsi que toutes les bibliothèques et leur dépendance inhérentes au projet.

On notera que pour utiliser un exécutable généré par PyInstaller, il faut que windows soit mis à jour.

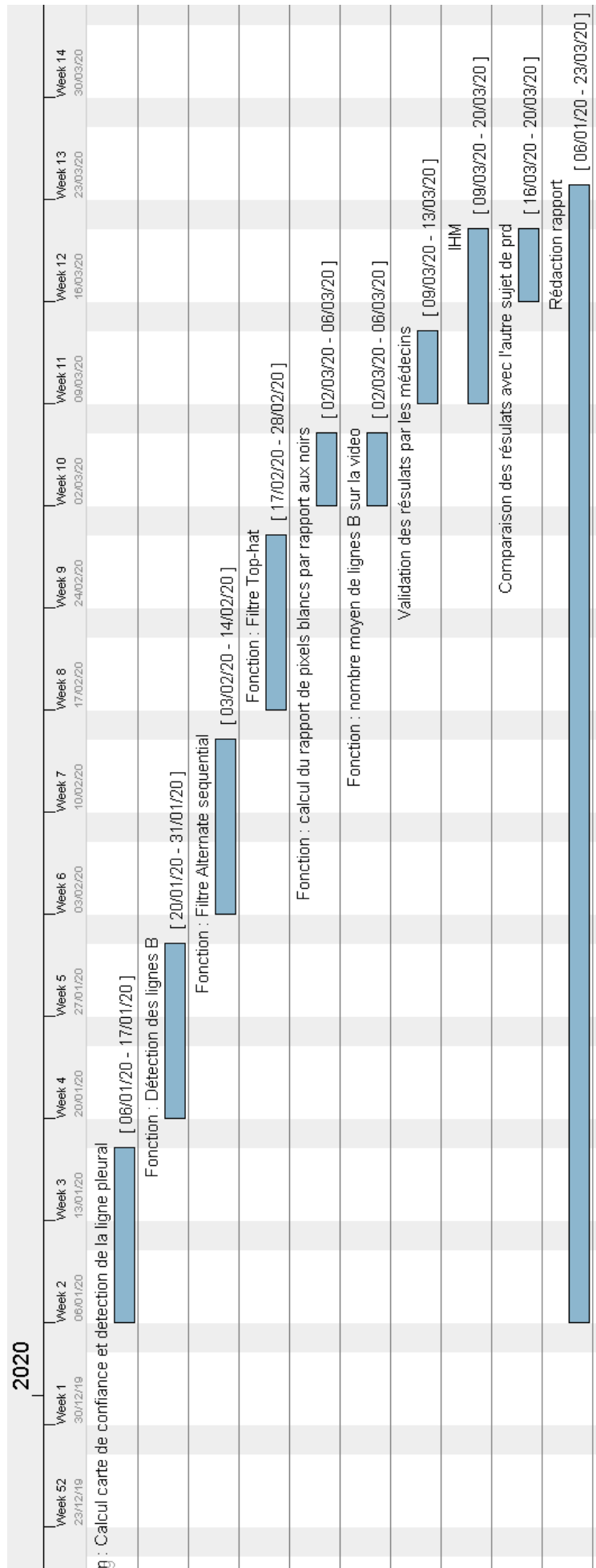
D

Gestion de projet

1 Diagramme de Gant du S9



2 Diagramme de Gant du S10



E

Github, installeur et exécutable

Le code du projet est disponible sur github avec également l'installateur windows ou directement l'exécutable.

- Github : https://github.com/nathtest/Lung_BLine
- Installateur windows : https://github.com/nathtest/Lung_BLine/blob/master/Innosetup/LungBLine_setup.exe
- Exécutable : https://github.com/nathtest/Lung_BLine/blob/master/dist/Lung%20BLine.exe

1 Création de l'exécutable à partir des sources

Pour générer l'exécutable à partir des sources, il suffit de se mettre à la racine du projet. Puis de lancer dans une ligne de commande :

```
C:\Users\nath\PycharmProjects\Lung_BLine>pyinstaller --onefile --windowed main.spec .\main.py
```

Cette ligne indique à Pyinstaller qu'il faut créer un exécutable en utilisant main.py comme point de départ de l'application.

Le fichier main.spec précise à Pyinstaller où chercher les bibliothèques et ressources nécessaires au fonctionnement du programme.

Cette ligne de commande est disponible dans un fichier create_exe.bat sur le github ainsi que le fichier main.spec également disponible sur le github.

2 Création de l'installateur à partir de l'exécutable

Pour créer l'installateur, il suffit d'installer le programme InnoSetup disponible à l'adresse <https://jrsoftware.org/isinfo.php>.

Puis d'ouvrir le script Lung BLine inno setup.iss disponible sur le github dans le dossier Innosetup avec le programme InnoSetup.

Enfin lancer ce script avec InnoSetup créera automatiquement un installateur avec l'exécutable présent dans le dossier dist de l'application.

Il est possible qu'il soit nécessaire de modifier certains chemins dans le script afin de correspondre à l'architecture de fichier de la machine exécutant le script.

F

Doc d'installation utilisateur

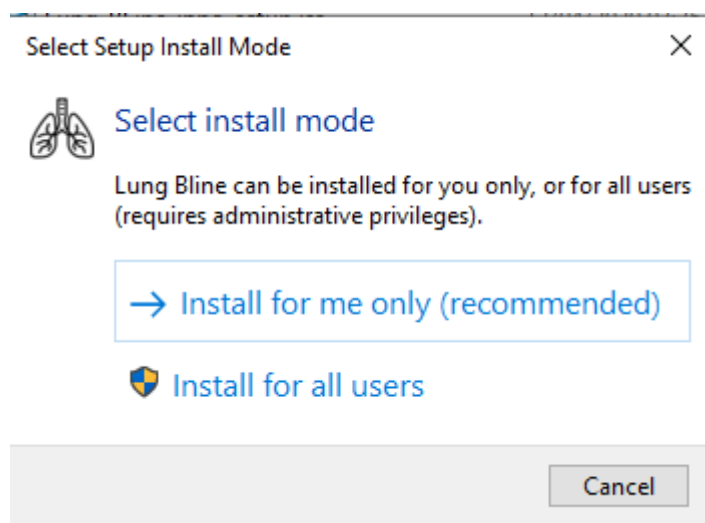
L'installation du programme est très simple, il suffit de télécharger l'installateur du programme sur le github dans le dossier InnoSetup.

Il n'y a pas besoin d'avoir d'interpréteur Python installer sur votre machine puisque l'exécutable en embarque déjà un (ce qui explique sa taille plutôt volumineuse supérieure à 80 Mo).

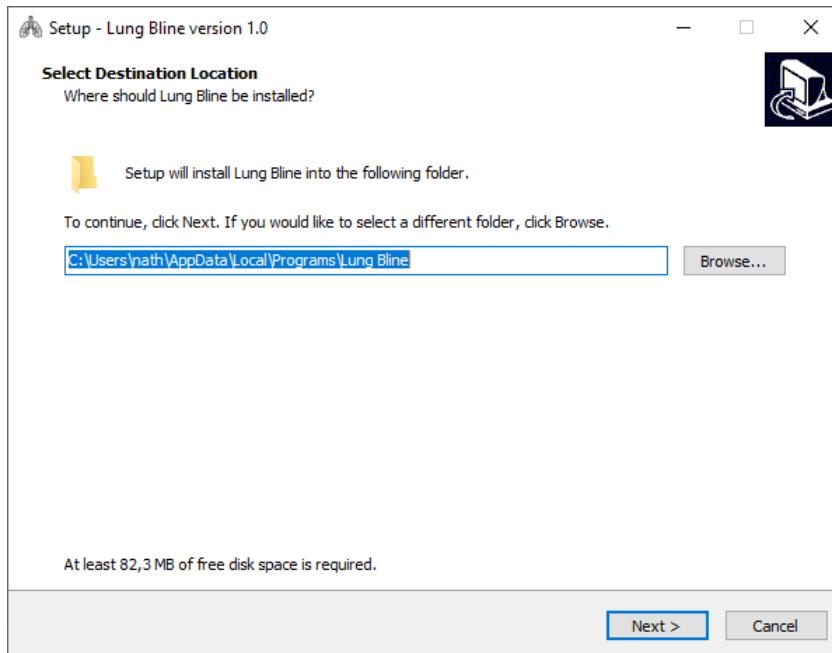
— Installateur windows : https://github.com/nathtest/Lung_BLine/blob/master/Innosetup/LungBline_setup.exe

Puis une fois télécharger, on exécute l'installateur et on suit les instructions.

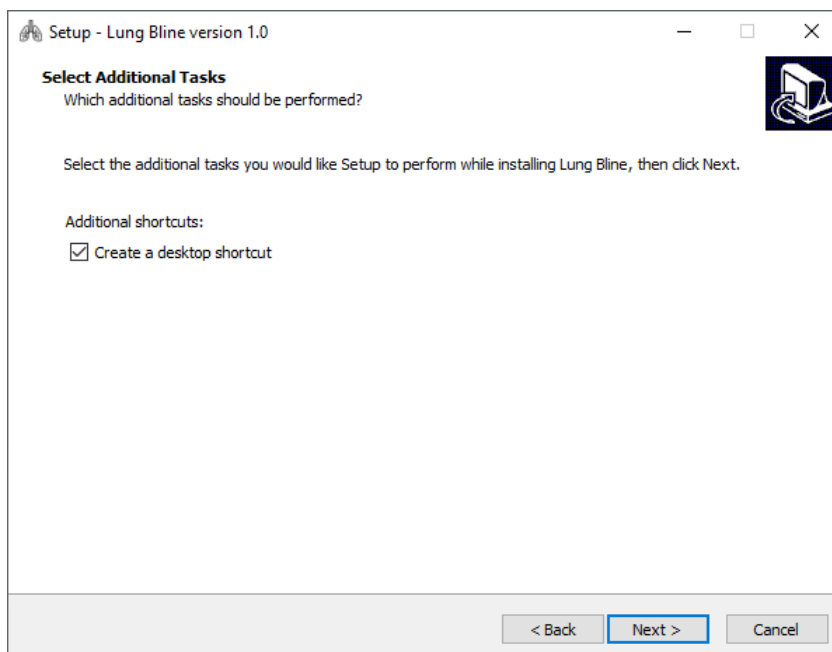
Étape 1 : Choix des privilèges de l'application



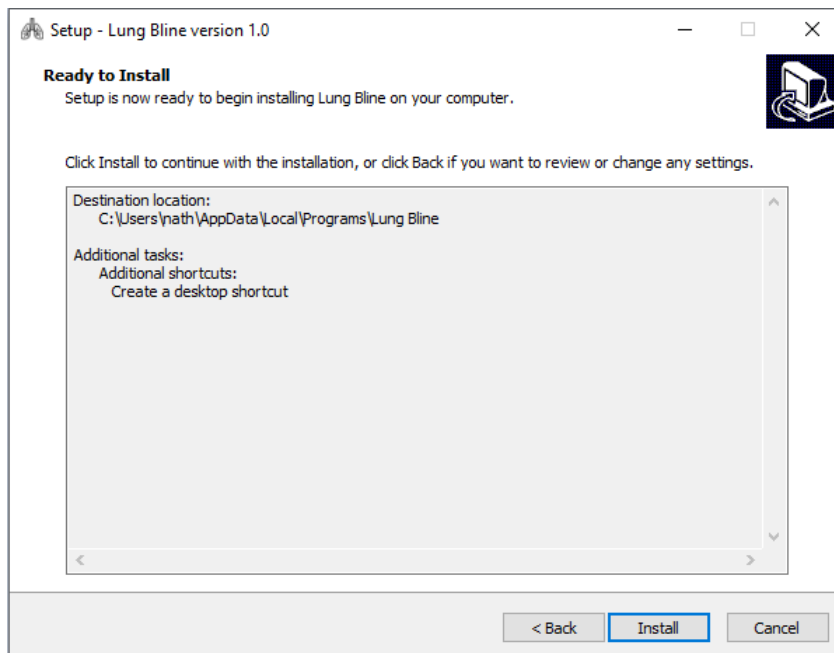
Étape 2 : Choix du chemin d'installation de l'application



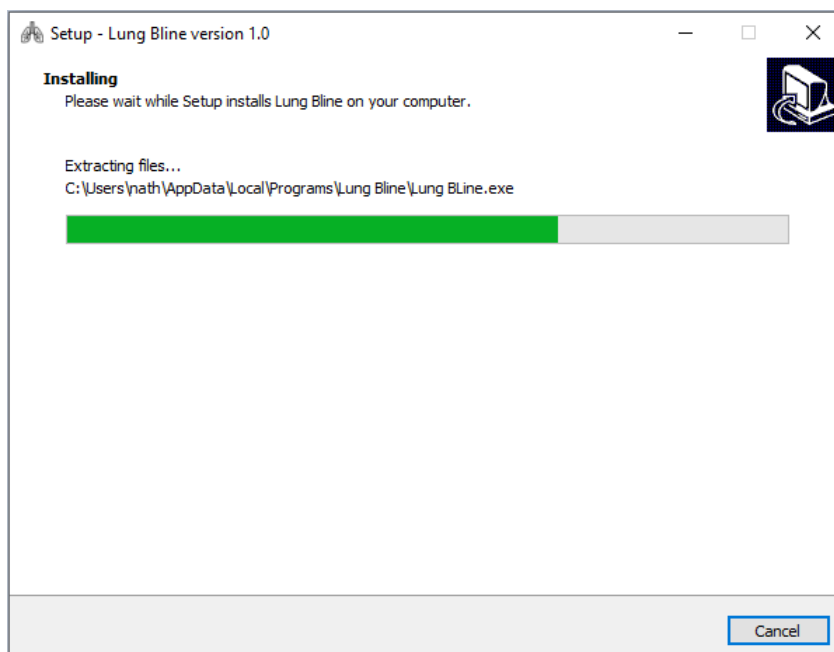
Étape 3 : Choix de création d'un raccourcie sur le bureau



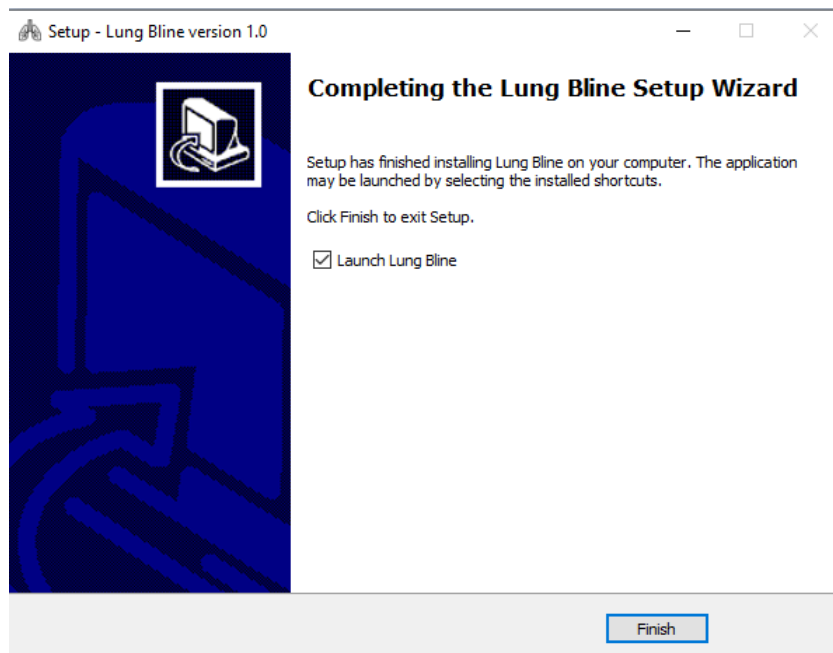
Étape 4 : Lancement de l'installation



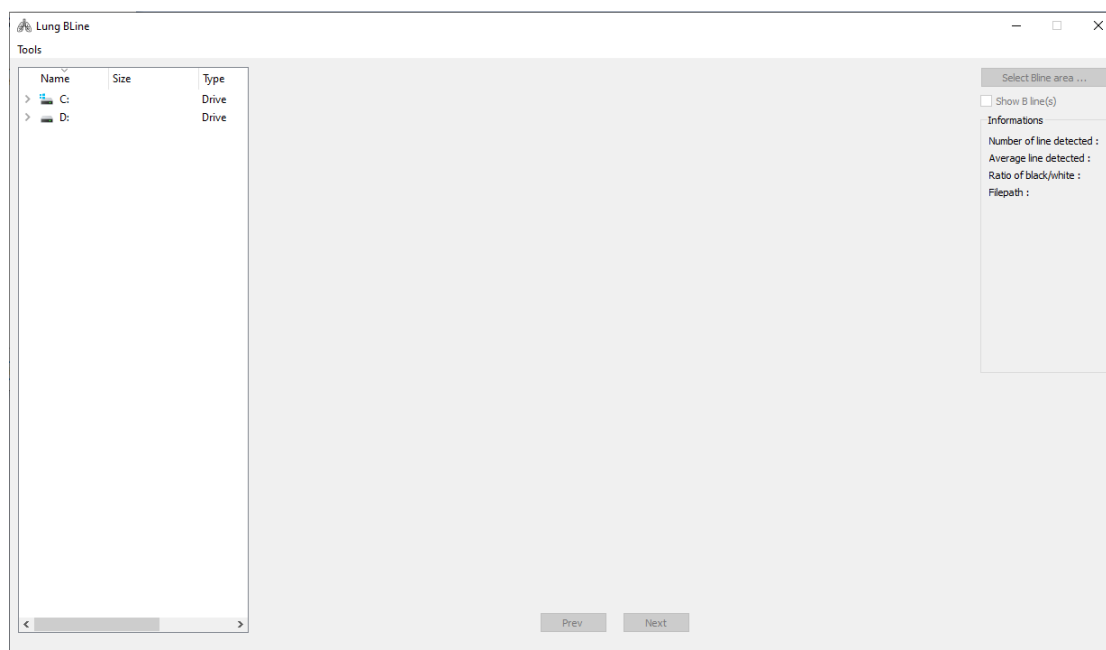
Étape 5 : Installation en cours



Étape 6 : L'installation est finie



Étape 7 : Exécution de l'application





Doc d'installation développeur

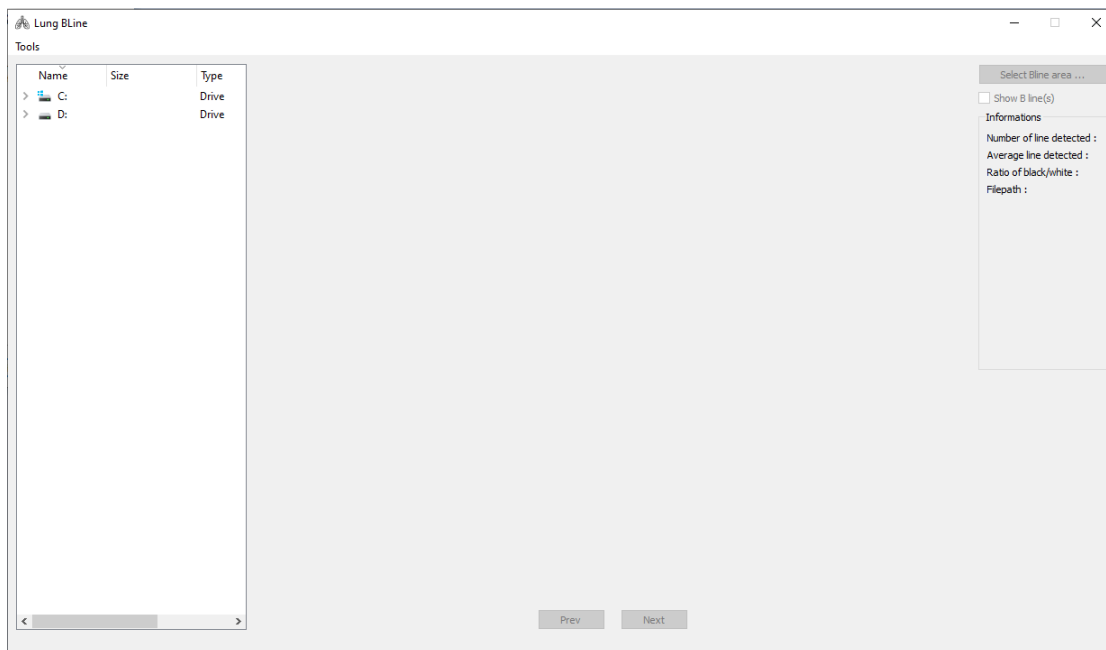
Pour les futurs développeurs de ce projet, l'installation est assez simple.

1. Cloner le projet depuis github https://github.com/nathtest/Lung_BLine
2. S'assurer d'avoir une version de Python supérieur ou égale à 3.7
3. Installer le package numpy 1.18.1
4. Installer le package PyQt5 5.13.1
5. Installer le package matplotlib 3.1.2
6. Installer le package opencv-python 4.2.0.34
7. Installer le package pydicom 1.4.1
8. Installer le package qimage2ndarray 1.8.3
9. Installer le package scikit-image 0.16.2
10. Installer le package scipy 1.4.1
11. L'exécution du programme se fait par le fichier main.py
12. De nombreux modules possède un main indépendant afin de pouvoir les tester

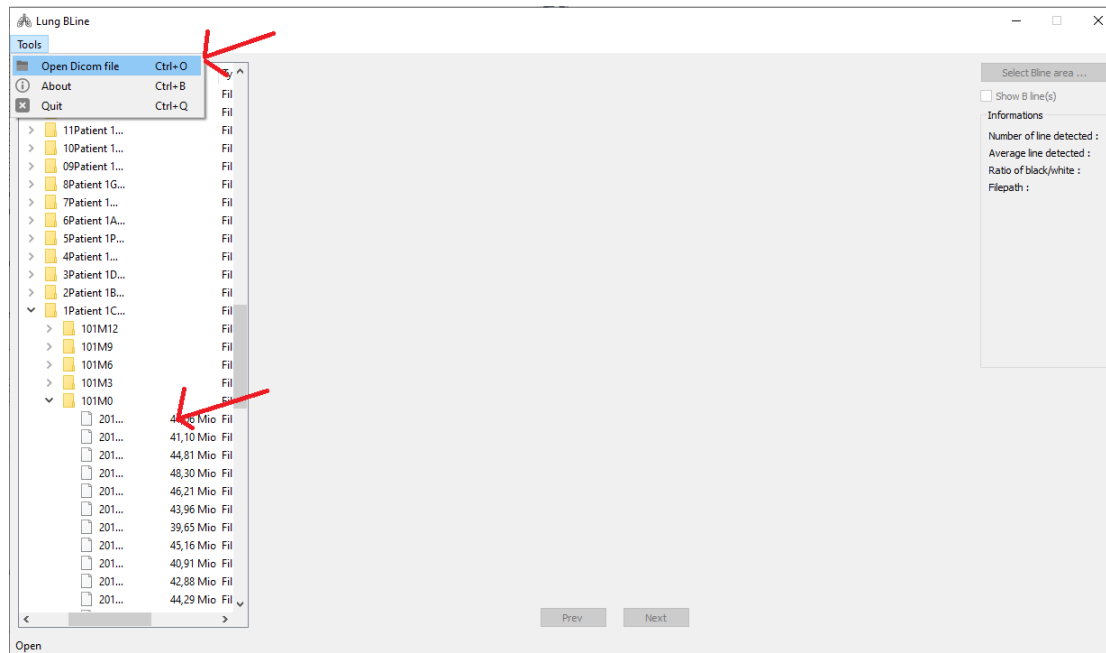
H

Doc d'utilisation

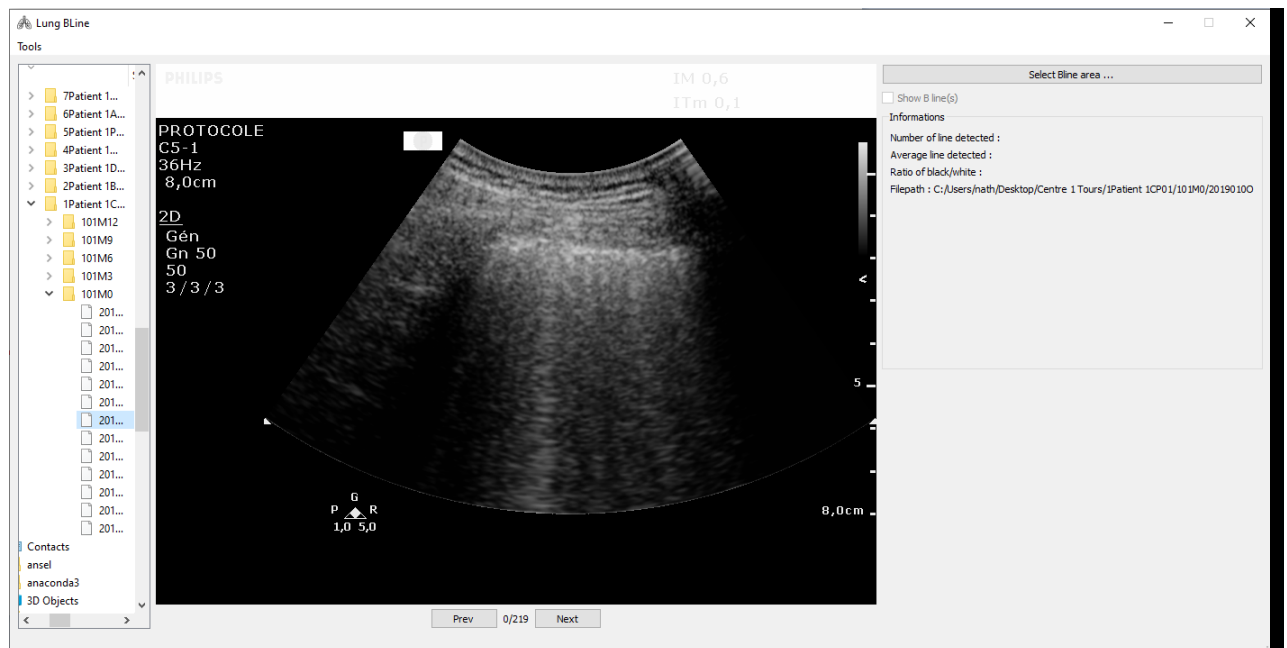
Lorsque le programme est lancé par défaut, on arrive sur cette fenêtre avec une photo barrée qui indique qu'aucun fichier DICOM n'est ouvert.



Pour ouvrir un fichier DICOM, il y a deux possibilités soit passer par le menu Tools -> Open Dicom file ou se déplacer dans l'arborescence disponible à gauche du programme et de double cliquer sur un fichier dicom.



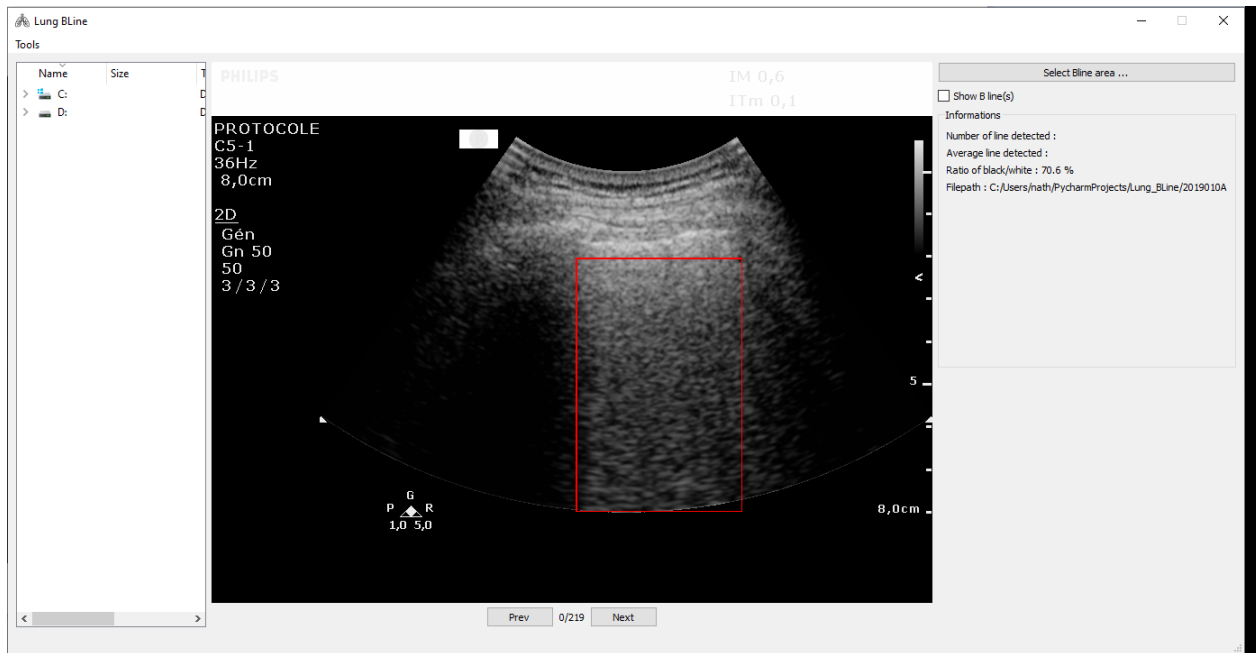
Ensuite, une fois le fichier DICOM ouvert la premier image du fichier devrait apparaître. Il est possible de naviguer dans les images du fichier grâce au bouton Prev (précédent) et au bouton Next (suivant).



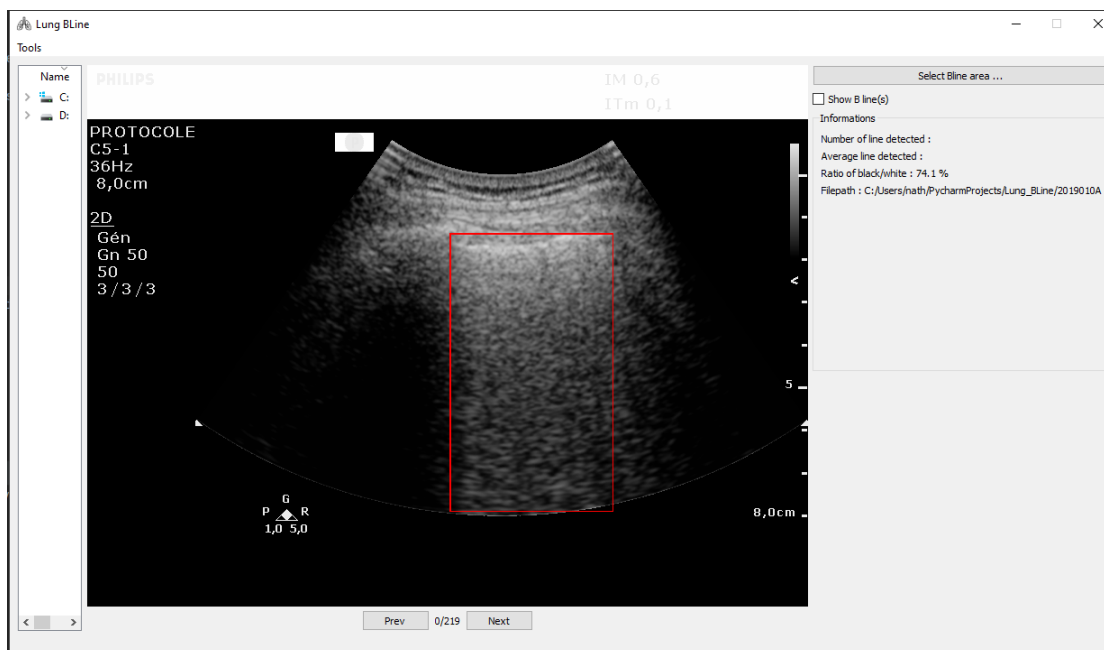
On active la sélection de la zone d'intérêt qui se situe sous la ligne pleural avec le bouton Select Bline area.

Dans cette sélection, il ne faut pas inclure la ligne pleural.

Exemple d'une sélection correcte :



Exemple d'une sélection incorrecte :



A partir du moment où une sélection est réalisée le ratio de pixel blanc par rapport au pixel noir est automatiquement calculé.

En cochant la case Show B-Line, on active l'algorithme de détection des lignes B.

Il est bon de noter que dans cette version l'algorithme n'obtenant pas de résultat concluant, le fait de cocher cette case n'entraînera aucune action.

I

Webographie

1. <https://www.geeksforgeeks.org/random-walk-implementation-python/>
2. https://en.wikipedia.org/wiki/Random_walker_algorithm
3. https://fr.wikipedia.org/wiki/Marche_al%C3%A9atoire
4. https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_random_walker_segmentation.html
5. https://docs.opencv.org/master/d5/d0f/tutorial_py_gradients.html
6. https://en.wikipedia.org/wiki/Sobel_operator
7. https://fr.wikipedia.org/wiki/Transformation_de_Hilbert
8. <https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.hilbert.html>
9. <https://docs.scipy.org/doc/numpy/reference/generated/numpy.log.html>
10. https://fr.wikipedia.org/wiki/Morphologie_math%C3%A9matique#Dilatation_et_%C3%A9rosion
11. https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html
12. https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga67493776e3ad1a3df63883829375201f
13. https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#gac2db39b56866583a95a5680313c314ad



Bibliographie

1. <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/9790/1/Novel-automatic-detection-of-pleura-and-B-lines-comet-tail/10.1117/12.2216499.short?S0=1>
2. <https://ieeexplore.ieee.org/abstract/document/7962263>
3. <https://www.ncbi.nlm.nih.gov/pubmed/24277902>

Analyse d'images échographiques des poumons : Rapport

Ponceau Nathanaël

Encadrement : Makris Pascal, Ragot Nicolas et Hidane Moncef

Objectifs

Lors de pathologies respiratoires où les poumons se remplissent d'eau. On utilise un échographe et des lignes particulières appelées **lignes B** apparaissent sur l'échographie. L'objectif de ce projet est de détecter les lignes automatiquement car plus elles sont nombreuses plus la maladie est avancée.

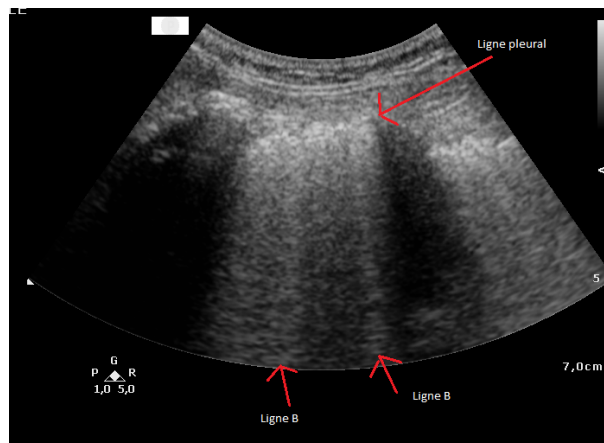
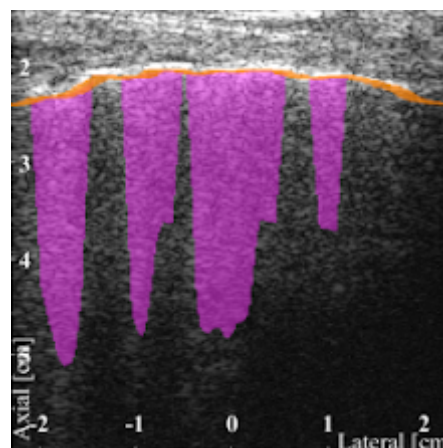


Image échographique d'un poumon malade

Mise en œuvre

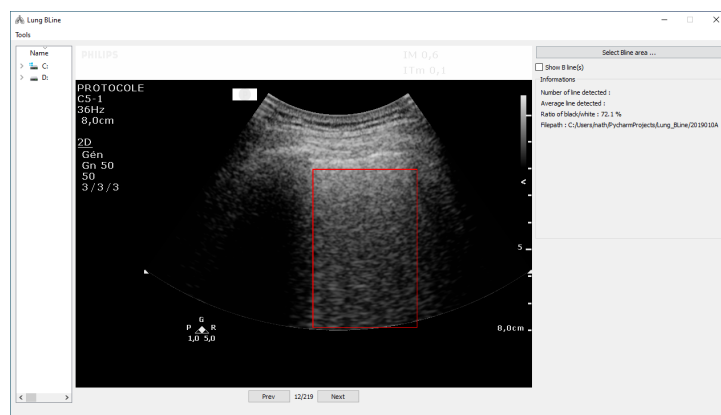
Dans la partie recherche, il faut trouver une **méthode automatique de détection des lignes B** adapter aux images échographiques fournies par le CHU de Tours. Dans la partie développement, il faut intégrer la méthode automatique de détection dans une interface graphique facilitant la lecture et la manipulation des fichiers extraits de l'échographe.



Surimpression des lignes détectées

Résultats attendus

- Détection automatique de la zone contenant les lignes B
- Comptage automatique du nombre de lignes B
- Déterminer le ratio blanc/noir de la zone des lignes B
- Interface graphique intégrant la détection



Logiciel de détection développé

Analyse d'images échographiques des poumons : Rapport

Ponceau Nathanaël

Encadrement : Makris Pascal, Ragot Nicolas et Hidane Moncef

Objectifs

Lors de pathologies respiratoires où les poumons se remplissent d'eau. On utilise un échographe et des lignes particulières appelées **lignes B** apparaissent sur l'échographie. L'objectif de ce projet est de détecter les lignes automatiquement car plus elles sont nombreuses plus la maladie est avancée.

Mise en œuvre

Dans la partie recherche, il faut trouver une **méthode automatique de détection des lignes B** adapter aux images échographiques fournies par le CHU de Tours. Dans la partie développement, il faut intégrer la méthode automatique de détection dans une interface graphique facilitant la lecture et la manipulation des fichiers extraits de l'échographe.

Résultats attendus

- Détection automatique de la zone contenant les lignes B
- Comptage automatique du nombre de lignes B
- Déterminer le ratio blanc/noir de la zone des lignes B
- Interface graphique intégrant la détection

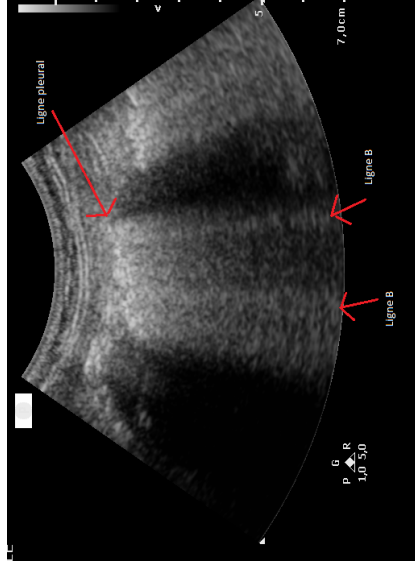
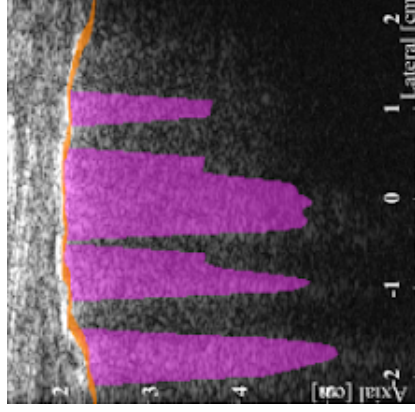
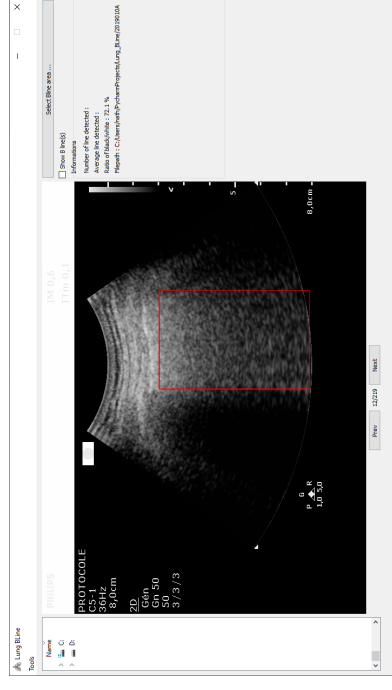


Image échographique d'un poumon malade



Surimpression des lignes détectées



Logiciel de détection développée

Analyse d'images échographiques des poumons

Rapport

Résumé

Ce projet de recherche et développement intitulé Analyse d'images échographiques des poumons en collaboration avec le CHU de Tours va permettre grâce à l'implémentation d'une méthode automatique de détection de lignes d'aider au diagnostic des médecins des maladies pulmonaires.

Mots-clés

Lignes B, Ligne pleurale, Détection de lignes automatiques, Transformée de Hilbert, Masque binaire, Opérations morphologiques, Python

Abstract

This project of research and development named Analysis of ultrasound images of the lungs with the University Hospital of Tours will allow to help the diagnosis of pulmonary diseases with an automatic method of line detection.

Keywords

B lines, Pleural line, Automatic method of line detection, Hilbert transform, Binary mask, Python, Morphological operations

Tuteurs académiques

Makris PASCAL
Ragot NICOLAS
Hidane MONCEF

Étudiant

Ponceau NATHANAËL (DI5)