RAPPORT DE PROJET DE FIN D'ÉTUDES

Post-Traitements de Segmentations d'Images de Fond d'Œil

ANDRADE Kalvin, MARCINIAK Thibault, SOPÉNA Alexis

2022-2023

Client : DULAU Idris
Encadrant : DESBARATS Pascal

Contents

1 Introduction

L'analyse des images de fond d'œil pourrait permettre d'évaluer l'état vasculaire cérébral d'une personne de manière non-envahissante, rapide et peu coûteuse. Le point clé de cette évaluation réside dans la caractérisation du réseau rétinien, celui-ci doit être segmenté automatiquement de façon fiable et reproductible pour permettre l'extraction de mesures servant à un corps médical pour la réalisation d'un diagnostic.

La **segmentation** d'image est, dans le domaine discret, une opération qui rassemble des pixels entre eux selon certains critères. Les techniques de segmentation sont multiples et peuvent se baser aussi bien sur l'analyse individuelle de pixels que sur l'analyse d'un voisinage à des degrés plus ou moins élevés. L'introduction de l'apprentissage profond pour réaliser cette tâche a pour but de tenter de modéliser les données d'études avec un haut niveau d'abstraction pour combiner les différentes façons d'analyser l'image.

L'apprentissage supervisé est un moyen permettant d'obtenir ces segmentations, et dans une qualité supérieure aux méthodes traditionnelles (niveau de détail, robustesse aux variations de contrastes, ...).L'analyse du fond d'œil se fait sur des images en 2D. Le réseau vasculaire est composé d'artères et de veines qui sont des structures de forme arborescente dont la racine est le disque optique. Les segmentations permettant l'extraction de mesures doivent inclure une distinction entre le réseau d'artères et le réseau de veines. Cette distinction se traduit par une couleur différente dans la segmentation générée, bleu pour les veines, rouge pour les artères.Les artères et les veines sont deux structures distinctes, mais la 2D fait apparaître des superpositions entre les artères et les veines. La couleur verte est utilisée pour les cas de superpositions dans la segmentation. Toutes les images en contiennent.

La segmentation finale est composée de quatre classes (quatre couleurs), le rouge pour les artères, le bleu pour les veines, le vert pour les superpositions et le noir pour le reste de l'images. Ces segmentations ainsi générées par apprentissage supervisé ont cependant un défaut majeur, la discontinuité des classes au sein d'une même branche. Il devient alors impossible d'extraire des mesures sur des artères ou des veines si celles-ci ne sont pas déterminables comme telles.

Notre projet vise à résoudre cette problématique. A partir de segmentation générées par apprentissage supervisé, nous proposons une méthode de post-traitement pour traiter la discontinuité des classes au sein d'une même branche.

2 Analyse de l'Existant

2.1 Apprentissage supervisé pour la segmentation artères-veines

Il existe plusieurs méthodes afin de segmenter une image de fond d'oeil par apprentissage supervisé. Dans cette section nous allons présenter une première méthode qui fait une segmentation de toutes les classes en même temps. Puis, une autre méthode qui fait une segmentation binaire pour chaque classe (artère, veine) pour obtenir une segmentation finale en fusionnant chaque segmentation binaire.

2.1.1 Segmentation multi-classes en une étape

Un premier article [1], nous présente un réseau de neurones convolutif permettant de prédire toutes les classes en une seule fois. Le principe de leur méthode est de classifier les pixels d'une image de fond d'oeil en artère (rouge) ou veine (bleu). Les auteurs de cet article ont utilisé l'ensemble des données RITE [7] qui contient une vérité terrain artère/veine d'image rétiniennes provenant du jeu de données publique DRIVE [8].

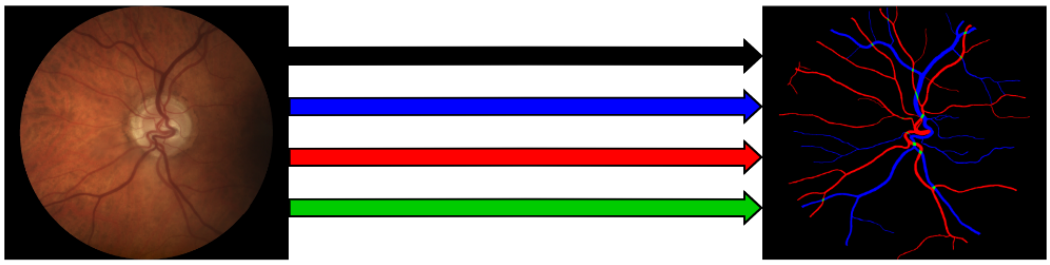

Figure 1: Schéma de segmentation multi-classes en une étape. Bleu : veines, rouge : artères, vert : superposition artère-veine, noir : fond de l'image.

Le schéma de la figure 1 représente une classification 4 classes d'une image de fond d'oeil. Les résultats présentés avec le réseau de neurones convolutif de l'article montrent que tous les pixels ne sont pas prédit de la bonne classe. De plus, leur méthode ne prend pas en compte les superpositions entre artère et veine, les pixels qui représente une superposition artère-veine sont de la même couleur que les pixels du fond de l'image.

2.1.2 Segmentation multi-classes en plusieurs étapes

Un deuxième article [2], propose une méthode d'apprentissage profond permettant la mesure automatisée des vaisseaux rétiniens. Leur système se nomme RMHAS (Retina-bases Microvascular Health Assessment System), il se compose de plusieurs étape pour arriver à une segmentation des vaisseaux rétiniens de qualité. Premièrement leur système évalue la qualité de l'image de fond d'oeil avant de procéder à une segmentation. Il utilise un réseau en U à branches multiples pour la segmentation binaire . Puis, il génère une carte intermédiaire des caractéristiques des vaisseaux rétiniens, qui est fusionné avec l'image d'entrée et divisée en trois branches distinctes pour la segmentation des artères, des veines et du disque optique de la rétine. Enfin ils font deux évaluations différentes sur la qualité des segmentations issues de leur méthode pour ne garder que les segmentations de qualité.

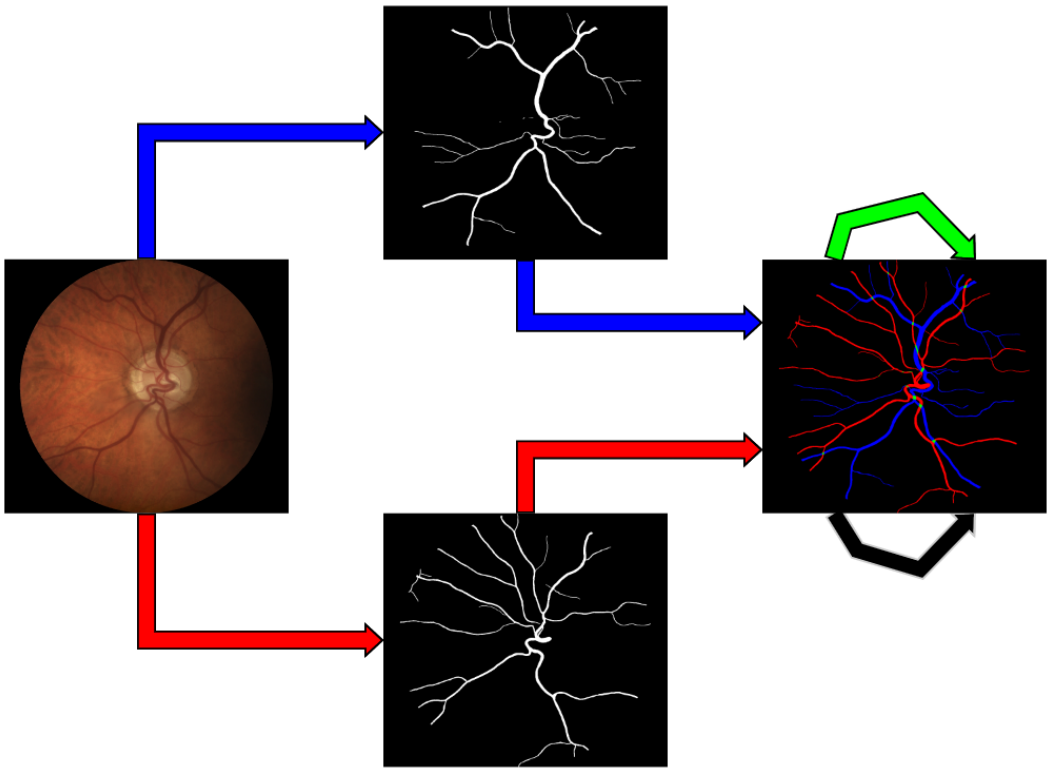

Figure 2: Schéma de segmentation multi-classes en plusieurs étapes. Bleu : veines, rouge : artères, vert : superpositions artère-veine, noir : fond de l'image.

La figure 2 illustre les étapes pour obtenir une segmentation 4 classes en faisant deux prédictions de segmentations binaire, une pour le réseau d'artères et une pour le réseau de veines. L'image résultante de la fusion entre les deux segmentations binaires est ensuite segmentée en 4 classes où les pixels classifiés en bleu sont des veines, les pixels classifiés en rouge sont des artères. Les pixels classifiés en vert sont les superpositions entre artères et veines.

2.2 Post-traitements pour la segmentation artères-veines

Cet article [3], présente un algorithme de post-traitements d'images 3 classes générées par leur modèle.

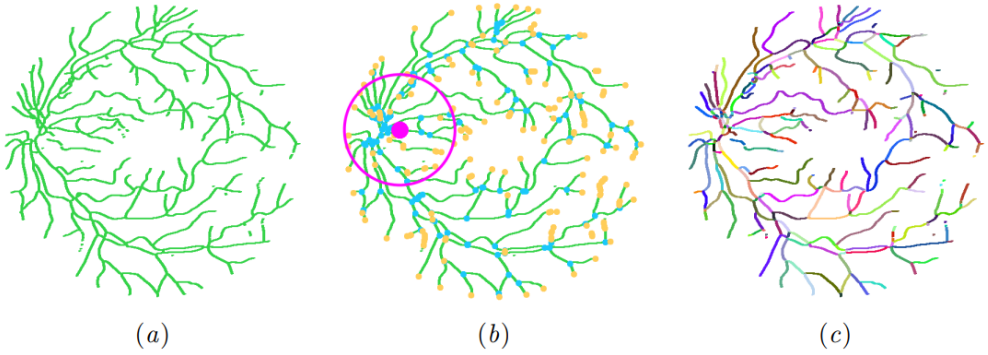

Figure 3: Exemple d'unification des étiquettes à l'intérieur d'un segment.

Leur méthode propose dans un premier temps d'étiqueter tous les segments d'une segmentation 2 classes. Comme on peut le voir dans la figure 3 sur l'image (a), elle détecte un point clé qui définit un point central sur l'image. Puis, elle détecte toutes les bifurcations et les bouts des branches qui sont représentés par des points bleus et jaunes sur l'image (b). Cela crée des segments qui peuvent ensuite être étiquetés, qui sont représentés par une couleur unique sur l'image (c).

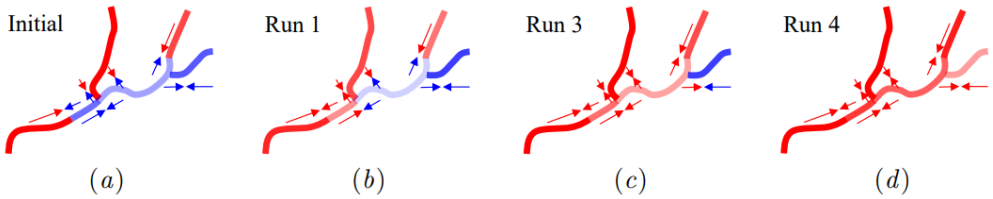

Figure 4: Exemple de propagation de la prédiction.

Grâce aux segments créés précédemment et aux points-clés, des vecteurs sont attribués à chaque segment avec un poids indiquant si le segment est bien prédit ou non. Le poids est calculé en fonction de le direction de son vecteur par rapport à la position du point-clé. La figure 4 est une illustration de plusieurs itérations de leur algorithme. Une image présentant une discontinuité de classe (a) devient une image ne possédant une seule classe (d).

2.3 Méthode d'apprentissage supervisé du client

L'objectif de notre projet est le post-traitements des images générées par la méthode d'apprentissage supervisé de notre client. A la différence des images générées par les méthodes de l'état de l'art 2.1, ces images présentent une segmentation 5 classes (blanc pour vaisseaux indéterminés). La figure 5 illustre la méthode proposée. En plus de la segmentation binaire artère et de la segmentation binaire veine, une troisième segmentation (vaisseaux indéterminés) est générée. L'ajout d'une segmentation binaire des vaisseaux permet de réduire une perte d'informations si des vaisseaux sur l'image de fond d'oeil n'ont pas pu être prédit en artère ou en veine. Comme on peut le voir sur la figure 5, l'image finale générée par la méthode de notre client est une segmentation 5 classes. Si en post-traitements de cette segmentation on fait une propagation des couleurs sur les pixels blancs alors on passe d'une segmentation 5 classes à 4 classes, on obtient un gain d'information.

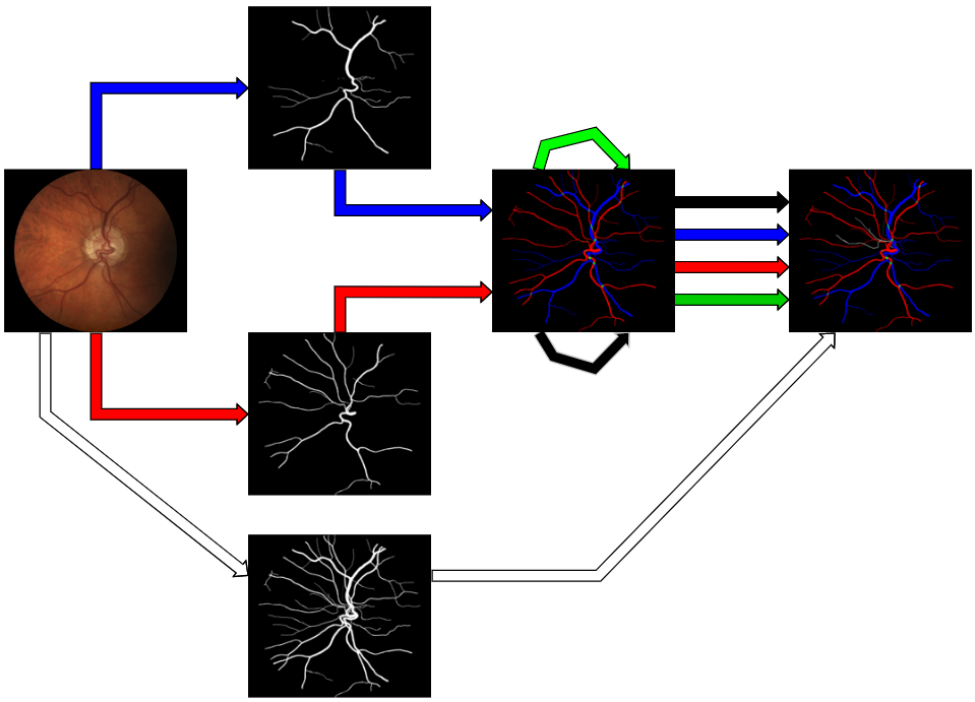

Figure 5: Schéma de segmentation multi-classes du client. Bleu : veines, rouge : artères, vert : superpositions artère-veine, blanc : vaisseaux indéterminés (soit artère soit veine), noir : fond de l'image.

2.4 Algorithme de remplissage par diffusion

L'algorithme de remplissage par diffusion est un algorithme très utilisé notamment dans les jeux-vidéos ou dans les logiciels de traitement d'image afin de remplir une composante connexe à l'aide d'une couleur ou un motif voulu. Bien que son implémentation à l'aide d'une pile est assez simple, les résultats obtenus par cet algorithme sont très corrects.
Voici son fonctionnement à l'aide d'une pile dans le cadre du traitement d'image.

- Un pixel est choisi comme pixel de départ.
- On colore ce pixel de la couleur souhaitée.
- On regarde ses pixels voisins en fonction du type de connexité. On colore les voisins avant de les rajouter sur la pile afin que leurs voisins soient traités à leurs tours.
- L'étape précédente est répétée jusqu'à ce que la pile soit vide.

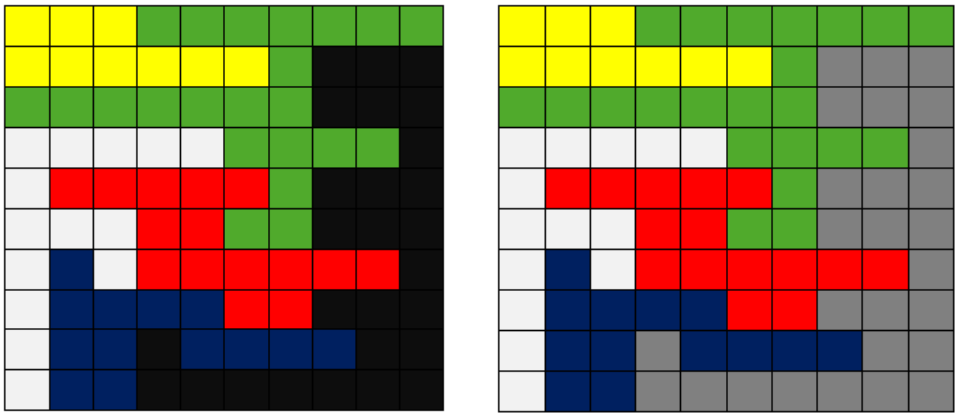

Figure 6: Exemple de remplissage par diffusion à partir d'une case noire, où on propage du gris dans toute la partie connexe

2.5 Métriques dévaluation

Les métriques d'évaluation sont des mesures quantitatives utilisées pour évaluer les performances d'un modèle statistique pour une tâche spécifique. Elles comprennent généralement quatre composantes de base qui définissent la matrice de confusion (CM) : Vrai Positif (TP), Faux Positif (FP), Vrai Négatif (TN) et Faux Négatif (FN). La MC est obtenue par une comparaison pixel à pixel entre une segmentation prédite et sa vérité terrain associée. Pour les vaisseaux rétiniens, la vérité terrain est une segmentation annotée manuellement réalisée par un spécialiste médical. Dans la CM, les pixels positifs font référence aux vaisseaux rétiniens en blanc et les pixels négatifs font référence aux pixels d'arrière-plan en noir. Vrai(True) fait référence aux pixels bien classés et Faux(False) fait référence aux pixels mal classés. Ainsi, TP est le nombre de pixels blancs bien classés.

Plusieurs métriques d'évaluation peuvent être utilisées pour résumer les performances globales d'un modèle, telles que : Precision (P), Recall (R), le F1-score et le coefficient de similarité de Dice (Dice) (le F1-score et le coefficient de similarité de Dice sont numériquement égaux dans le cas binaire).

Chacune de ces mesures a en commun de ne pas utiliser les TN dans le cadre de l'évaluation de la qualité. Cette particularité conduit à une évaluation plus robuste des qualités des segmentations dans notre cas de déséquilibre de classe où les vaisseaux rétiniens ne représentent que 10 F1-score et le coefficient de similarité de Dice sont particulièrement intéressants car ils équilibrent Precision et Recall.

Precision	Recall	Dice
$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{2*TP}{2*TP+FP+FN}$

Table 1: Precision, Recall & Dice

D'autres mesures d'évaluation des performances de segmentation des vaisseaux rétiniens sont également utilisées, telles que : Accuracy, Balanced Accuracy , Specificity et Aire sous la courbe (AUC). Ces mesures, à l'inverse de la précision, du rappel et du coefficient de similarité de Dice, calculent les TN dans le cadre de leur évaluation, ce que nous considérons comme une erreur.

2.6 Données utilisées

Pour réaliser et évaluer nos post-traitements, nous avons à disposition 97 segmentations générées par la méthode du client. A chaque segmentation automatique est associée une segmentation manuelle d'un ophtalmologue. Les 97 segmentations ont été générées à partir de 3 jeux de données de fond d'oeils 7 : DUALMODAL2019 [4](30 images), HRF [5](45 images) et LES-AV [6] (22images).

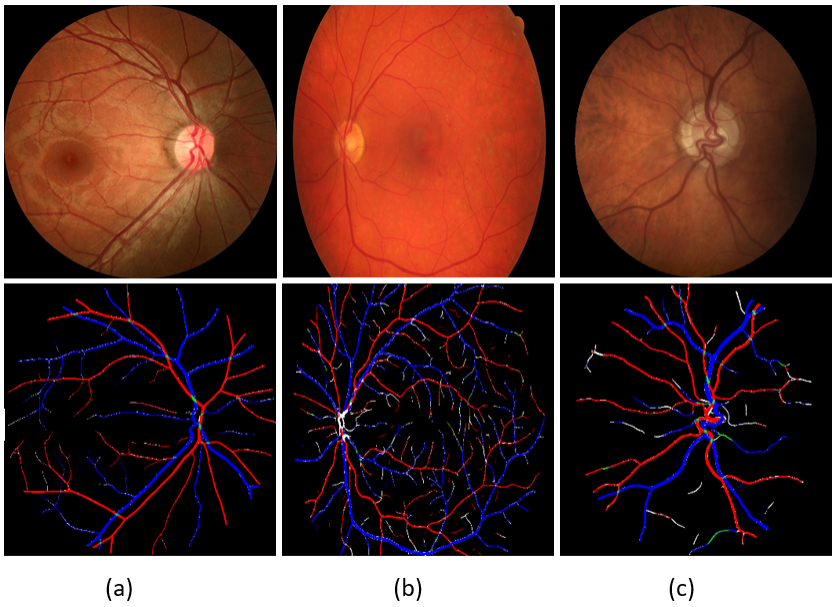

Figure 7: Exemples de données sources avec leurs segmentations générées par la méthode du client pour chaque jeu de données. (a) : DUALMODAL2019, (b) : HRF, (c) : LES-AV.

3 Besoins logiciel

3.1 User stories

- **US1 :** En tant qu'utilisateur, je souhaite améliorer la segmentation en ajoutant des pixels verts aux zones de croisement entre artère et veine.
- **US2 :** En tant qu'utilisateur, je souhaite améliorer la segmentation en remplaçant les pixels blancs par des pixels rouge, bleu ou vert selon ses pixels voisins.
- **US3 :** En tant qu'utilisateur, je souhaite améliorer la segmentation en corrigeant les problèmes de discontinuités de classes.
- **US4 :** En tant qu'utilisateur, je souhaite pouvoir appliquer l'algorithme de post-traitements sur un jeu de données.

3.2 Besoins

Grâce à la rédaction de nos user stories, nous pouvons plus facilement déduire et mettre en place une liste de besoins fonctionnels et non fonctionnels qui nous servira tout au long du projet.

3.2.1 Besoins Fonctionnels

- **BF1:** Propager les pixels rouges et bleus afin qu'il n'y est plus aucuns pixels blancs sur l'image et donc de passer d'une segmentation 5 classes à une segmentation 4 classes.
- **BF2 :** Garder les pixels verts correctement prédits.
- **BF3 :** Remplacer les pixels verts qui ne sont pas sur un croisement artère/veine par des pixels rouges ou bleus.
- **BF4 :** Détecter les zones de croisement artère/veine et remplacer la couleur des pixels en vert.
- **BF5 :** Corriger la discontinuité de classes générée par la méthode du client.

3.2.2 Besoins Non Fonctionnels

- **BNF1 :** Lancer un script sur un jeu de données de segmentation généré par la méthode du client.
- **BNF2 :** Générer une image en moins de 2 secondes.

4 Organisation

4.1 Déroulement du projet

Pour la réalisation de ce projet, nous avions pensé à développer un algorithme servant à résoudre tous les besoins énoncés précédemment. Cependant cette approche n'est pas optimale, surtout dans le cadre d'un projet de groupe, car nous étions confrontés à de nombreux obstacles qui sont les différents cas et problèmes à gérer d'une traite. Nous avons donc abordé le problème d'une autre approche.

Pour faciliter le développement ainsi que l'avancement de notre projet, nous avons décidé d'utiliser un algorithme qui servira de base à ce projet. Pour cela nous avions besoin pour la première étape d'un algorithme de coloration assez naïf améliorant grandement la segmentation initiale. La seconde étape qui caractérise cette approche est donc d'appliquer cet algorithme sur nos images, pour ensuite lister tous les différents cas, les avantages ainsi que les inconvénients et les problèmes rencontrés dans les résultats obtenus. La prochaine étape est d'améliorer cette base en gérant un ou plusieurs cas considérés comme inconvénients pour pouvoir obtenir une version un peu plus avancée de l'algorithme de base. En répétant les étapes deux et trois, nous obtiendront un algorithme avancé capable de résoudre le besoin principal de notre projet qui est de traiter les défauts de la segmentation initiale.

L'algorithme que nous avons choisi pour la base de ce projet est l'algorithme de remplissage par diffusion.

4.2 Choix techniques

Pour réaliser ce projet nous devions utiliser le langage Python. Le client nous a également fourni différents jeux de données qu'il a lui même segmenté après une prédiction faite par un modèle de Deep Learning. Comme nous travaillons sur des images il est très intéressant d'utiliser les bibliothèques NumPy et OpenCV, en effet les images peuvent être traitées comme des tableaux de pixels. Nous pouvons donc effectuer de nombreuses opérations sur ces tableaux pour réaliser efficacement nos éventuels traitements nécessaires. Nous avons également utilisé le coefficient de Dice pour évaluer la qualité de nos algorithmes.

5 Réalistation

5.1 Version 1

Afin d'avoir une base solide sur laquelle travailler, nous avons implémenté un algorithme de remplissage par diffusion qui va permettre une propagation des couleurs sur les pixels blancs des segmentations à cinq classes que le client nous a fournies. L'implémentation suit ce déroulement.

- On parcourt toute l'image en entrée, puis on récupère les coordonnées de chaque pixel blanc trouvé.
- Parcours de la liste des coordonnées des pixels blancs stockés et on stock la couleur de chacun de ses voisins.
- On ignore les pixels noirs et blancs qui ne nous intéressent pas
- On colore le pixel courant de la couleur dominante de ses voisins. Si le nombre de pixels bleus est égal au nombre de pixels rouges, alors on laisse le pixel courant en blanc.
- On répète l'opération plusieurs fois.

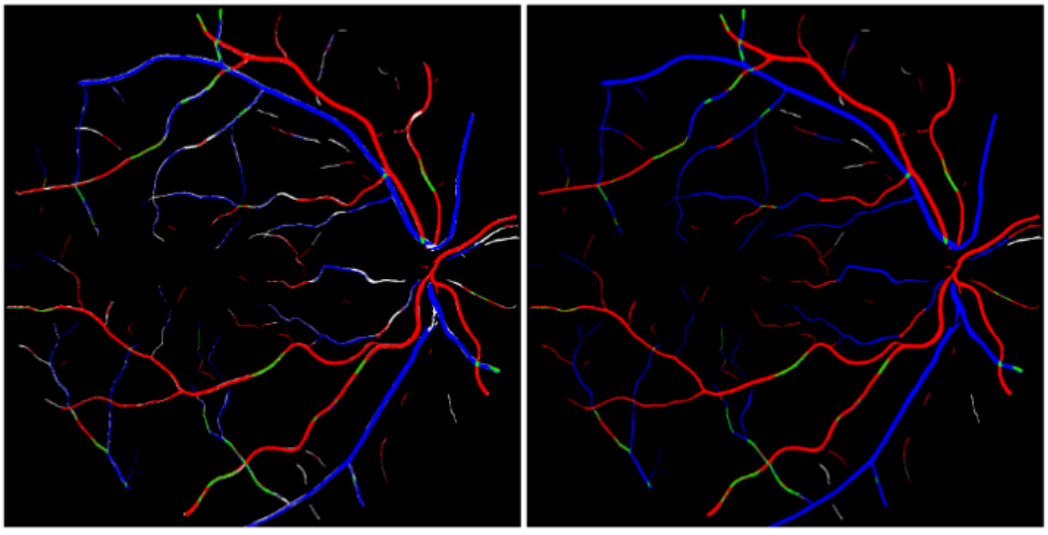

Figure 8: Algorithme de remplissage par diffusion appliqué sur une segmentation issue de DUALMODAL2019.

5.1.1 Avantages

- Nous pouvons constater une propagation des couleurs. Les vaisseaux blancs initialement connexes à des segments colorés ont reçu la couleur de ces segments voisins. La propagation des couleurs a bien été effectuée.
- L'algorithme est très rapide. Nous présenterons les résultats ultérieurement.

5.1.2 Limitations

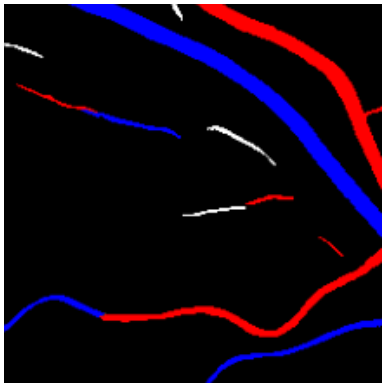

Figure 9: Exemple de segments blancs non connectés au reste du réseau.

Comme le montre la figure 9, certains vaisseaux isolés restent blancs. L'algorithme va propager les couleurs en fonction de la couleur des pixels voisins tel un flux, les segments blancs qui ne sont pas reliés au reste du réseau ne peuvent pas recevoir cette propagation des couleurs.

Une première approche pourrait être d'attribuer aux pixels de ce segment, la couleur majoritaire des segments à proximité de celui-ci. Une discontinuité des classes peut-être observée à l'intérieur d'un même vaisseau.

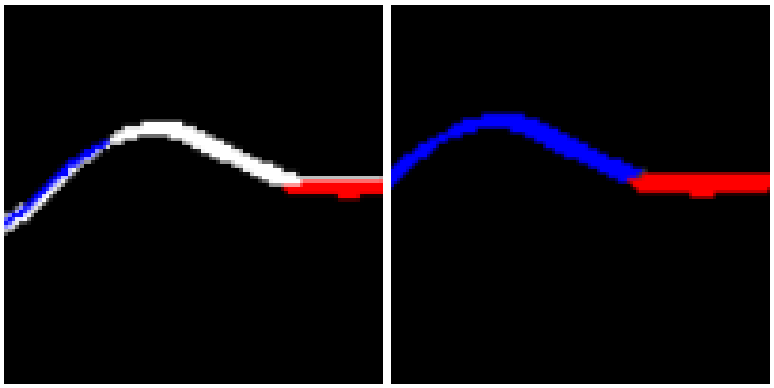

Figure 10: Propagation d'une couleur par rapport à l'autre dans un même segment.

Dans l'exemple de la figure 10, toujours issu des données de DUALMODAL2019, nous avions un segment à la fois bleu et rouge, les deux couleurs étant séparées par des pixels blancs. Après avoir appliqué l'algorithme de remplissage par diffusion, le segment a été propagé en bleu sur la partie blanche. Cette propagation s'explique car on effectue un balayage de l'image de haut en bas et de gauche à droite afin de récupérer la liste des pixels blancs. Par conséquent, les premiers pixels à être propagés seront les pixels blancs les plus à gauche et les plus hauts sur l'image. C'est pour cette raison que la partie blanche de ce segment a été colorée en bleu et non en rouge malgré sa présence.

Une éventuelle solution pour ce problème de discontinuité serait de modifier le sens du balayage des pixels de l'image. Pour cela il faudrait que l'algorithme s'adapte pour les différents cas de propagation. En effet dans la même image, le sens peut changer en fonction du vaisseau propagé. Nous pouvons supposer d'effectuer un balayage des pixels depuis le disque optique pour obtenir la meilleure propagation.

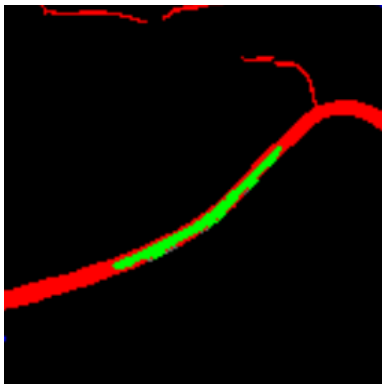

Figure 11: Présence de vert indésirable.

La figure 11 nous montre du vert conservé dans certaines régions où il parait être indésirable. Sur la même image, on peut remarquer du vert au milieu d'un vaisseau dont le voisinage ne comporte que du rouge. Il n'y a qu'une dizaine de pixels bleus autour de ce segment donc on peut supposer que le vert devrait être remplacé par du rouge qui est quasiment totalitaire sur cette région du vaisseau.

Pour résoudre ce cas-là, nous pourrions peut-être comme pour le premier point, vérifier la couleur dominante dans un rayon autour de ces pixels verts. Établir un certain seuil, un ratio de dominance de couleur, pourrait potentiellement limiter une mauvaise attribution de couleur à ces pixels verts. Par exemple dans un rayon de vingt pixels, si le ratio de pixels rouges sur le bleu est supérieur à 80%, alors on colore le vert en rouge.

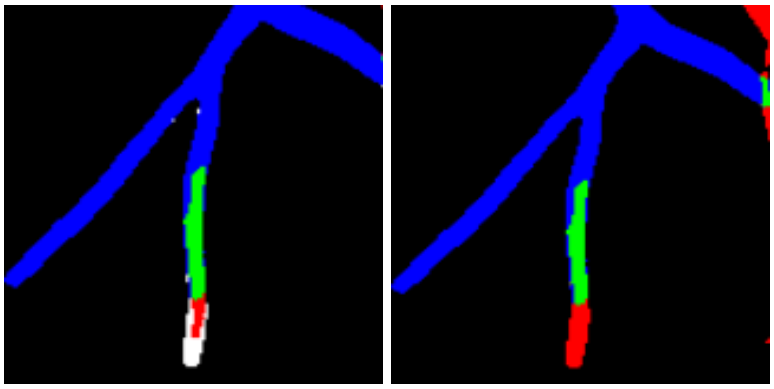

Figure 12: Exemple de propagation d'une mauvaise prédiction.

Avec comme exemple la figure 12, une image tirée du dataset LESAV, on peut observer une propagation un peu étrange du rouge. En effet, il n'y a que très peu de rouge sur la prédiction et le segment est parfaitement rattaché à un vaisseau bleu. Le remplissage par diffusion propage donc ce rouge, cependant il est clair que du bleu est attendu ici.

Il est possible de gérer ce cas en utilisant encore le même principe des voisins les plus proches en prenant un rayon suffisamment grand pour capturer la dominance de bleu. Vérifier la connexité du segment complet avec les segments aux alentours peut aussi être une piste de recherche afin d'effectuer une bonne propagation des couleurs.

Dans notre implémentation, nous avons choisi d'effectuer 50 tours de boucles afin de conclure la propagation. Ce nombre est le résultat de plusieurs tests réalisés afin de déterminer quel est le nombre de tours de boucles nécessaires à la propagation complète des couleurs sur la totalité des segmentations que le client nous a fournies.

Nous pouvons améliorer notre code en ajoutant une condition d'arrêt. En effet si l'algorithme propageait les couleurs tant que l'image d'entrée contient des pixels blancs, alors nous n'aurions plus besoin d'utiliser une constante comme nombre de tours de boucles. Afin de s'assurer que le programme termine, il faut néanmoins corriger le cas des segments blancs non-connexes afin d'éviter une boucle infinie.

5.1.3 Résultats

Une fois cet algorithme appliqué sur toutes les segmentations initiales, nous pouvons calculer le coefficient Dice moyen sur chacun des datasets entiers. Nous pouvons également calculer le temps d'exécution moyen sur les images afin de vérifier que nous ne dépassons pas les deux secondes par image.

	DUALMODAL2019	HRF	LESAV	TEMPS D'EXECUTION
SEGMENTATION INITIALE	A : 0.9748 V : 0.9533 S : 0.1943	A : 0.9727 V : 0.9592 S : 0.0055	A : 0.9585 V : 0.9591 S : 0.2387	
POST-TRAITEMENT VERSION 1	A : 0.9927 V : 0.9891 S : 0.6510	A : 0.9950 V : 0.9020 S : 0.0051	A : 0.9812 V : 0.9807 S : 0.5167	662ms
VÉRITÉS-TERRAIN	A : 1 V : 1 S : 1	A : 1 V : 1 S : 1	A : 1 V : 1 S : 1	

Table 2: Dice moyen par jeu de données pour la segmentation des artères (A) des veines (V) et des superpositions (S). La segmentation initiale correspond à la segmentation générée par le client.

Ainsi avec l'algorithme de remplissage par diffusion nous servant de base, nous obtenons pour chaque image une nouvelle segmentation en 0.7 secondes en moyenne. La contrainte du temps d'exécution est par conséquent bien respectée. En ce qui concerne la qualité de la segmentation représentée par les coefficients Dice calculés, nous remarquons que nous avons de meilleures valeurs pour les artères et les veines mais que nous perdons en efficacité sur les superpositions des deux.

5.2 Version 2

Pour améliorer nos traitements et ainsi obtenir une meilleure segmentation que celle obtenue avec l'algorithme Flood-Fill, nous devons limiter et corriger les défauts présentés plus haut.

5.2.1 Remplacement du vert

Avec la propagation, nous avons pu observer que des zones vertes se retrouvaient à l'intérieur d'un vaisseau de couleur rouge ou bleu.

Pour corriger ce défaut, nous avons finalement essayé une autre méthode que celle énoncée précédemment qui nous semblait trop naïve. Nous avons plutôt décidé de rajouter une étape au début de l'algorithme. Maintenant, tous les pixels verts de l'image seront changés en blanc avant les autres traitements. Pour cela à l'aide de NumPy et d'OpenCV nous avons remplacés tous les pixels ayant comme valeur (0,255,000) par la valeur (255, 255, 255). Comme chaque classe de la segmentation est représentée par une couleur unique, nous pouvons facilement

repérer et changer la couleur des pixels.
Désormais lorsque les couleurs des vaisseaux bleus ou rouges seront propagées à l'endroit où se trouvait le vert après avoir utilisé l'algorithme de remplissage par diffusion.
Nous aurions pu simplement appliquer Flood-Fill sur les pixels verts cependant l'utilisation d'OpenCV et NumPy qui sont en partie développés en C, nous donne un meilleur temps d'exécution qu'en utilisant Python et ses boucles.

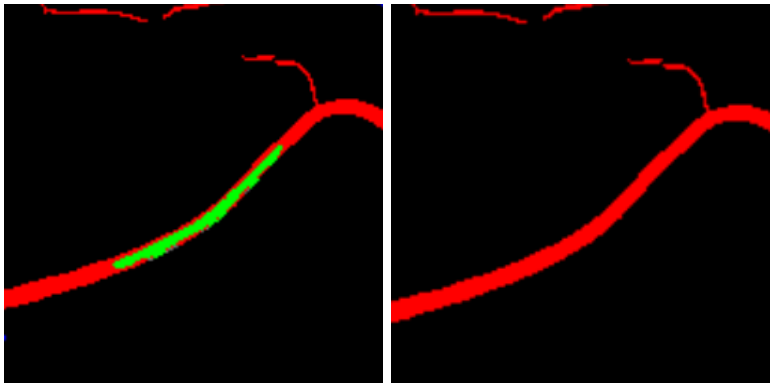

Figure 13: Suppression du vert initialement segmenté et propagation de couleur.

5.2.2 Segments blancs non-connexes

Bien que les pixels blancs à l'intérieur des vaisseaux colorés ont bien été propagés, les segments blancs non-connexes au reste du réseau ne peuvent pas être propagés car ils sont seulement entourés de pixels blancs ou noirs.

Comme énoncé précédemment, une solution serait de trouver quelle est la couleur majoritaire sur les segments à proximité dans un rayon donné à partir de ces pixels blancs. En effet on peut supposer que la couleur d'un segment entouré majoritairement de segments bleus est également bleue.

Cette passe succède la propagation de Flood-Fill. Après avoir appliqué l'algorithme sur une segmentation, nous récupérons les indices des pixels blancs restants sur l'image grâce à l'utilisation très pratique des méthodes NumPy. Parcourir l'image de nouveau pour récupérer les pixels blancs restants n'est certainement pas la méthode la plus efficace, cependant nous l'améliorerons par la suite.
Nous allons maintenant à partir de chaque pixel blanc restant, faire une coupe du tableau de pixels représentant l'image grâce au slicing de NumPy. Nous effectuons une coupe de ce tableau à partir du pixel blanc courant pour récupérer

un tableau correspondant au voisinage non-connexe du pixel blanc courant dans ce rayon. Comme pour Flood-Fill, les pixels noirs et blancs sont ignorés et la médiane du voisinage nous donnant la couleur dominante dans ce rayon peut maintenant être calculée. Il nous reste plus qu'à colorer le pixel blanc courant de la couleur que l'on vient de calculer.
La valeur du rayon a été paramétrée sur 20 pixels à la suite de tests sur les images pour savoir quel rayon fonctionnait le mieux en moyenne.

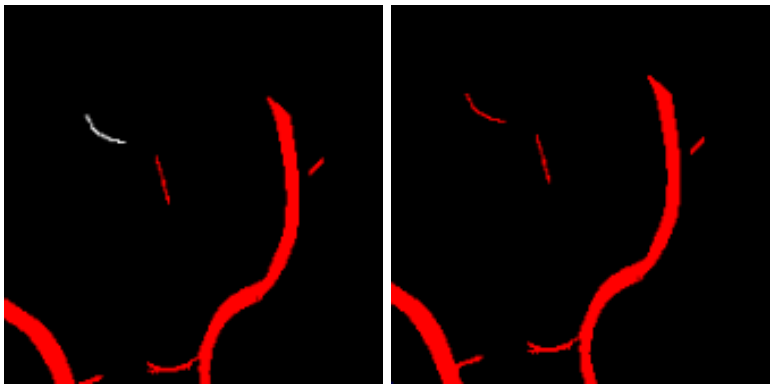

Figure 14: Propagation de la couleur sur les segments proches non-connectés.

5.2.3 Avantages

- Le remplacement du vert pour permettre la propagation à sa place est très efficace dans les zones simples et évidentes comme sur l'exemple montré.
- Les segments blancs sont bien colorés lorsque le voisinage est clairement unicolore
- L'algorithme est rapide.

5.2.4 Limitations

Du point de vue algorithmique, notre amélioration proposée n'est pas très efficace. En effet, nous effectuons de nouveau un parcours de tous les pixels de l'image pour récupérer les pixels blancs. Bien que nous utilisons NumPy, ce n'est pas du tout efficace et nous devons réfléchir à une nouvelle solution pour améliorer la complexité.

Lorsque nous faisons une recherche de la couleur dominante des pixels voisins dans un rayon, nous effectuons seulement une supposition qui peut devenir aléatoire

dans des réseaux très complexes où de nombreux vaisseaux de couleurs différentes sont proches. Le rayon de 20 pixels a été choisi de manière arbitraire de telle sorte à obtenir un résultat qui semble correct pour l'ensemble des images.

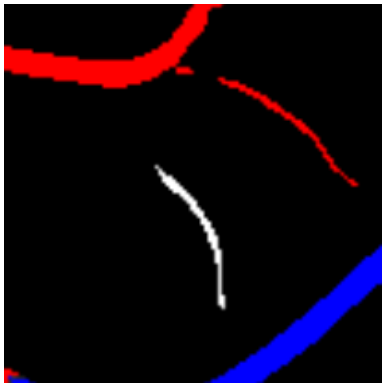

Figure 15: Exemple montrant un segment dont la détermination de la couleur est difficile.

Nous devons impérativement trouver une nouvelle et meilleure solution afin de ne pas fausser les résultats. Avec cette méthode actuelle, si le rayon est trop petit alors des segments resteront blancs car ils n'auront aucune couleur dans leurs voisinages. De plus, si le rayon est trop grand alors l'algorithme va capturer trop de couleurs, et le résultat sera certainement aléatoire et non utilisable par les utilisateurs.

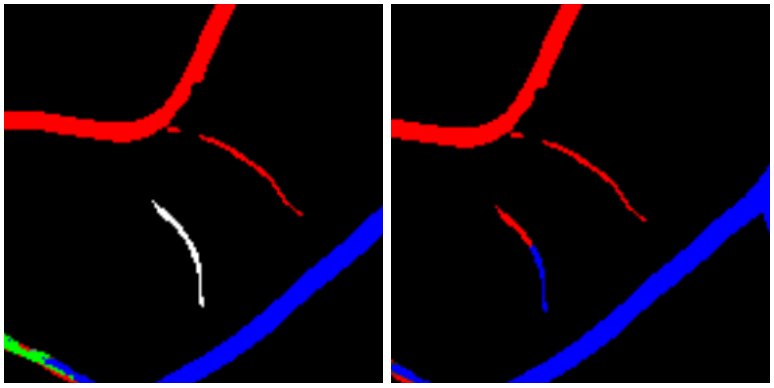

Figure 16: Discontinuité des classes provoquée par la recherche sur un segment initialement blanc.

La coloration des segments blancs indépendants produit des discontinuités dans certains segments. Ce problème est dû à l'ordre de traitement des pixels blancs ainsi qu'à un rayon de recherche trop élevé. Dans cette composante connexe blanche, les pixels les plus hauts sont traités en premier. Ils capturent du rouge car ils sont très proches du segment rouge situé au dessus. Cependant après plusieurs pixels, le rayon de recherche capture les pixels bleus du vaisseau inférieur. Une discontinuité est donc provoquée par notre algorithme.
Afin de régler ce problème, nous allons changer la méthode de propagation cette fois-ci en traitant toute la composante connexe ensemble et non pixel par pixel.

Comme nous supprimons tout le vert pour la propagation, nous n'avons plus d'information sur les croisements des veines et des artères. Nous devons rajouter le vert aux intersections dans les versions précédentes.

5.2.5 Résultats

	DUALMODAL2019	HRF	LESAV	TEMPS D'EXÉCUTION
SEGMENTATION INITIALE	A : 0.9748 V : 0.9533 S : 0.1943	A : 0.9727 V : 0.9592 S : 0.0055	A : 0.9585 V : 0.9591 S : 0.2387	
POST-TRAITEMENT VERSION 2	A : 0.9840 V : 0.9785 S : 0.0498	A : 0.9944 V : 0.9912 S : 0.0144	A : 0.9590 V : 0.9604 S : 0.0419	1819ms
VÉRITÉS-TERRAIN	A : 1 V : 1 S : 1	A : 1 V : 1 S : 1	A : 1 V : 1 S : 1	

Table 3: Dice moyen par jeu de données pour la segmentation des artères (A) des veines (V) et des superpositions (S). La segmentation initiale correspond à la segmentation générée par le client.

5.3 Version 3

Bien que certains inconvénients ont été retirés, nous avons pu voir que notre méthode est encore grandement améliorable et que de nouvelles limites sont apparues. Une nouvelle version est donc implémentée pour améliorer la précédente.

5.3.1 Amélioration des segments blancs

Pour cette nouvelle version de l'algorithme, nous avons essayé une nouvelle approche concernant la coloration des segments blancs non-connectés au reste du réseau. Dans la version précédente, nous traitions les pixels un par un, mais cette fois-ci nous allons traiter toute la composante connexe.

Grâce à la méthode proposée par OpenCV ***connectedComponentsWithStats*** nous pouvons extraire toutes les composantes connexes d'une image d'entrée. Pour cela nous effectuons un seuillage de façon à obtenir une image binaire seuillée avec seulement les pixels blancs, tous les autres pixels sont noirs. La méthode proposée par OpenCV nous permet d'obtenir toutes les composantes connexes qui composent l'image, et également d'autres informations nécessaires. Parmi ces informations, nous récupérons le centre de masse de chaque composante connexe. C'est maintenant à partir de ce centre de masse que nous allons appliquer notre recherche de couleur à proximité. Une fois la couleur dominante trouvée, nous colorons tous les pixels ayant le même label, c'est à dire les pixels appartenant à la même composante, de la couleur que nous venons de trouver. Ainsi, tous les pixels d'une même composante connexe auront la même couleur et cela nous évite de provoquer de nouvelles discontinuités des classes dans la segmentation du réseau vasculaire.
Pour la recherche de couleur, à l'aide d'une fonction récursive nous pouvons maintenant augmenter progressivement le rayon de recherche. Le rayon est initialisé à une valeur faible qui augmentera si aucune couleur n'est trouvée. De cette façon nous diminuons les risques de mauvaises attributions de couleurs, et les segments blancs les plus éloignés du reste du réseau seront quand même colorés.

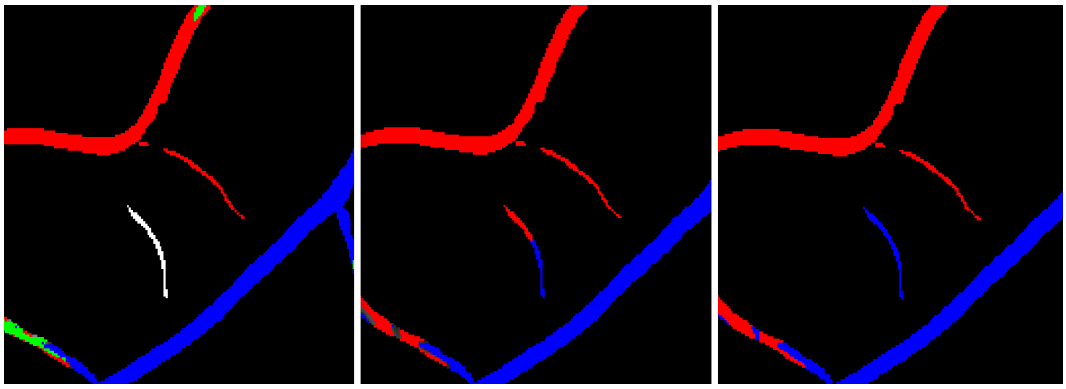

Figure 17: Amélioration de la coloration des segments blancs non-connexes. A gauche la segmentation initiale, au milieu la première version de la coloration produisant de la discontinuité, à droite la nouvelle version sans discontinuité.

5.3.2 Discontinuités simples

Lors de la propagation avec l'algorithme Flood-Fill, nous avons vu que certains pixels de mauvaises couleurs dû à une mauvaise prédiction initiale étaient propagés. Nous proposons une nouveau traitement permettant de gérer ce défaut. Toujours en utilisant la même méthode de OpenCV nous permettant d'extraire toutes les composantes connexes d'une image, nous allons maintenant utiliser l'information de la taille de la composante. Afin de traiter les petites composantes connexes, nous effectuons un seuillage des composantes en fonction de leur surface. Maintenant que nous avons seulement les plus petites composantes, notre méthode est de vérifier tous ses pixels pour s'assurer qu'ils soient bien tous de la même couleur.
Dans le cas où certains pixels sont d'une autre couleur, alors on colore l'entièreté de la composante de la couleur dominante afin de supprimer cette discontinuité.

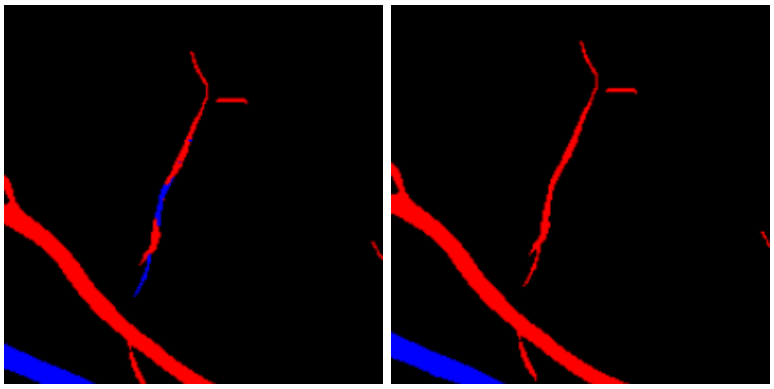

Figure 18: Suppression d'une discontinuité des classes simples.

5.3.3 Avantages

- Il n'y a plus de pixels blancs sur l'image.
- Les segments les plus éloignés des autres reçoivent tous une couleur et la propagation des segments blancs ne provoquent pas de discontinuités des classes.
- Les discontinuités simples, celles présentes dans les petites composantes non-connectées au reste du réseau sont totalement supprimées et nous obtenons bien une propagation d'une seule classe uniquement.

5.3.4 Limitations

- Bien que diminuée, il reste encore une part d'aléatoire dans l'attribution d'une couleur en fonction des pixels à l'intérieur du rayon. Afin de réduire au maximum cette probabilité d'attribuer la mauvaise couleur, nous devons trouver un moyen d'obtenir des informations concernant la structure du réseau. Avec cette information, nous serons en mesure d'effectuer une propagation plus fidèle à la réalité.
- Le choix de la taille de la surface servant au filtrage des composantes, a été décidé après plusieurs tests. Il faudrait trouver une autre méthode afin de choisir seulement les composantes les plus petites.
- Le temps d'exécution de l'algorithme qui gère les discontinuités est assez long. Notre algorithme utilise plusieurs boucles imbriquées en Python pour des parcours de tableaux, ce qui est assez coûteux. Nous devons améliorer

la complexité du programme en utilisant le plus possible les fonctionnalités de NumPy qui proposent des solutions beaucoup plus efficaces.

5.3.5 Résultats

	DUALMODAL2019	HRF	LESAV	TEMPS D'EXÉCUTION
SEGMENTATION INITIALE	A : 0.9748 V : 0.9533 S : 0.1943	A : 0.9727 V : 0.9592 S : 0.0055	A : 0.9585 V : 0.9591 S : 0.2387	
POST-TRAITEMENT VERSION 3	A : 0.9853 V : 0.9800 S : 0.0001	A : 0.9946 V : 0.9912 S : 0.0252	A : 0.9602 V : 0.9623 S : 0.0004	2036ms
DISCONTINUITÉS SIMPLES	A : 0.9852 V : 0.9800 S : 0.0001	A : 0.9946 V : 0.9916 S : 0.0251	A : 0.9602 V : 0.9622 S : 0.0004	6819ms

Table 4: Dice moyen par jeu de données pour la segmentation des artères (A) des veines (V) et des superpositions (S). La segmentation initiale correspond à la segmentation générée par le client.

Nous pouvons voir avec le coefficient Dice que nous obtenons des résultats très similaires avant et après la passe corrigeant le problème de discontinuités simples. Cela nous indique par conséquent que ce cas n'est pas très présent dans nos données ou que le choix de la surface minimale pour le seuillage n'est pas optimal.

6 Tests

Le coefficient de Dice nous a permis d'établir un test fonctionnel en comparant l'efficacité des algorithmes et de faire une moyenne sur toutes les données, où plus le coefficient est proche de 1, plus l'algorithme est efficace. Ces valeurs absolues sont un bon indicateur mais il est aussi important de regarder la différence des résultats entre les algorithmes testés afin de voir s'il y a une progression selon les versions de ces derniers.

Pour cela, nous utilisons le jeu de données contenant les vérités terrain pour chaque dataset. Celles-ci sont composées de deux images binaires : une pour la classe artère et une pour la classe veine.

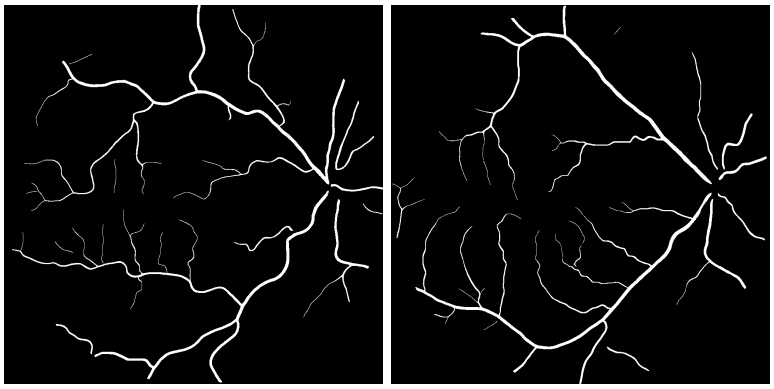

Figure 19: Exemple d'images binaires de vérité terrain (artères à gauche et veines à droite).

En faisant l'intersection des deux images, on obtient l'image binaire qui correspond aux superpositions que l'on est censé avoir.

Figure 20: Superposition attendue en fonction de l'exemple donné.

Concrètement, le script de comparaison utilise le canal rouge de l'image à tester, le compare avec la première image, utilise le canal bleu avec la deuxième image et le canal vert avec la dernière.

Bien que ces résultats donnent beaucoup d'informations, il faut noter que les valeurs de référence sont malgré tout une estimation manuelle et que le but n'est pas forcément d'obtenir un score le plus élevé possible, puisqu'il peut également y avoir des imprécisions dans ces vérités terrain.

Par ailleurs, étant donné que la sortie de notre algorithme est une image, nous avons pu établir un test fonctionnel qui consiste à vérifier manuellement ces images et y chercher des incohérences ou des données inappropriées qui ne pourraient pas se voir simplement avec des comparaisons statistiques.

Par exemple, au début de l'implémentation de l'algorithme Flood-Fill, il y avait un problème qui faisait apparaître des pixels gris en résultat. Ceux-ci n'étaient pas observables en regardant le coefficient de Dice, mais tout de suite visible en regardant les images en sortie du programme.

7 Perspectives

Concernant la segmentation des pixels verts qui représentent la superposition artère-veine, nous avons voulu régler le problème des pixels vert qui n'étaient pas sur une superposition. Dans la deuxième version de notre algorithme nous avons remplacé le vert par des pixels blancs afin d'appliquer notre algorithme de propagation dessus. La prochaine étape était de détecter les superpositions sur l'image afin de colorer ces pixels en vert. Malheureusement nous n'avons pas réussi à mettre en place une méthode pour détecter les superpositions. Une autre méthode nous semble possible pour augmenter le Dice sur les superpositions. En faisant une analyse visuelle sur les segmentations générées par la méthode du client, nous avons remarqué que la plupart des prédictions sur les superpositions étaient correctes. Nous avons donc réfléchi à une solution afin de garder les pixels verts bien prédits et de faire une propagation sur les pixels verts mal prédits, sans succès.

Pour des futures versions de notre algorithme, nous avons pensé à reprendre l'idée de la méthode présenté dans la section 2.2. L'idée est de définir des points-clés dans les segmentations. Ces points représentent les parties importantes nécessaires à une structuration du réseau. En effet, si nous arrivons à définir le début du réseau au niveau du disque optique, les intersections entre les segments ainsi que la fin de chaque branche, alors nous pourrons créer une structure en forme d'arbre qui nous permettra d'effectuer une meilleure propagation. De plus, cette structure nous permettra de plus facilement rajouter le vert correspondant aux intersections sur la segmentation.

Pour définir ces points-clés, nous devons dans un premier temps effectuer une squelettisation binaire de l'image. Squelettiser l'image, c'est réduire la taille de tous les vaisseaux à un pixel d'épaisseur. Avec cette squelettisation, nous pouvons facilement trouver les points terminaux du réseau. En effet, si un pixel n'a qu'un voisin alors il est terminal. S'il a plus de deux voisins alors il s'agit d'un croisement. Nous avions commencé à implémenter un programme permettant de calculer les points terminaux des branches du réseau, cependant sans la squelettisation de l'image nous ne pouvons pas obtenir des bons résultats. Malheureusement, nous n'avons pas réussi à squelettiser l'image malgré plusieurs approches notamment avec des traitements morphologiques comme l'érosion.

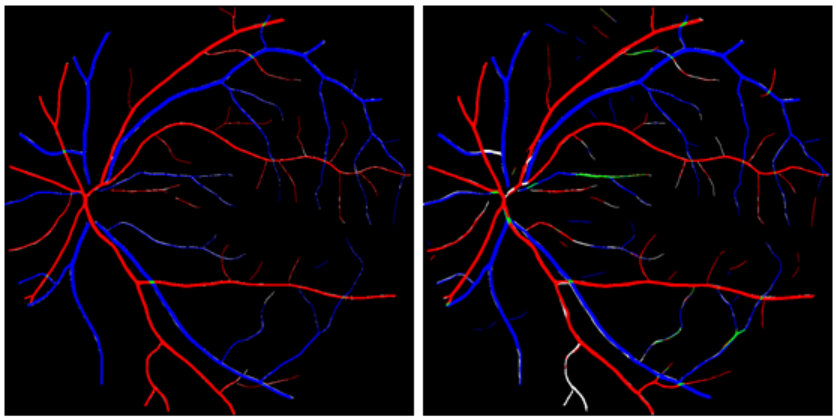

Figure 21: (a) : Exemple de cas simple de DUALMODAL2019, (b) : Exemple de cas difficile de DUALMODAL2019.

Vers la fin de notre projet, notre client nous a fourni des versions plus complexes des segmentations que nous avions utilisées depuis le début. En faisant la comparaison d'un cas simple avec un cas difficile en utilisant la figure 21, on voit que les cas difficiles contiennent un plus grand nombre de discontinuités de classes, ce qui rend notre algorithme moins efficace sur ce genre de cas. En mettant en place les solutions proposées dans cette section, nous pourrions mieux gérer ce genre de cas difficile.

8 Conclusion

Tout le long de ce projet, nous avons essayé d'améliorer notre base algorithmique afin de répondre aux besoins de notre client. Pour cela nous avons recensé les différents avantages ainsi que les limites de l'implémentation. Pour chaque limite, nous essayions de trouver une méthode pour résoudre les problèmes provoqués par les versions précédentes. L'objectif de ce projet paraît simple, cependant le nombre de cas différents sur les erreurs des segmentations initiales ainsi que les déductions à effectuer par le manque d'informations pour certains vaisseaux, rendent le développement de ce projet plus complexe et long que prévu. Cette difficulté, ajoutée à une mauvaise organisation, ne nous a pas permis d'avancer plus loin dans le projet. Nous obtenons des résultats prometteurs sur les données assez simples, des segmentations avec peu de défauts. Cependant, sur des segmentations plus complexes, plus proches de la réalité et avec de nombreux défauts, tous les objectifs et les besoins du clients ne sont pas remplis. Notre algorithme doit en partie être utilisé pour des diagnostics de pathologies, par conséquent il doit proposer des résultats fidèles à la réalité. Bien que notre algorithme ne propose pas des résultats très exploitables, il reste améliorable pour répondre à tous les besoins du client.

Nous souhaitons particulièrement remercier notre client DULAU Idris pour sa disponibilité, son implication et son soutien pendant ces deux mois ainsi que DESBARATS Pascal pour son encadrement et ses précieux conseils.

References

[1] Maria Ines Meyer, Adrian Galdran, Pedro Costa, Ana Maria Mendonça, *Deep Convolutional Artery/Vein Classifiaction of Retinal Vessels*, 2018.

[2] Danli Shi, Zhihong Lin, Wei Wang1, Zachary Tan, Xianwen Shang, Xueli Zhang, Wei Meng, Zongyuan Ge and Mingguang He, *A Deep Learning System for Fully Automated Retinal Vessel Measurement in High Throughput Image Analysis*, 2022.

[3] Liangzhi Li, Manisha Verma, Yuta Nakashima, Ryo Kawasaki, Hajime Nagahara, *Joint Learning of Vessel Segmentation and Artery/Vein Classification with Post-processing*, 2020.

[4] Shulin Zhang, Rui Zheng, Yuhao Luo, Xuewei Wang, Jianbo mao, Cynthia J.Roberts and mingzhai Sun, *Simultaneous Arteriole and Venule Segmentation of Dual-Modal Fundus Images Using a Multi-Task Cascade Network*, 2019.

[5] Jan Odstrcilik, Radim Kolar, Attila Budai, Joachim Hornegger, Jiri Jan, Jiri Gazarek, Tomas Kubena, Pavel Cernosek, Ondrej Svoboda, Elli Angelopoulou, *Retinal vessel segmentation by improved matched filtering: evaluation on a new high-resolution fundus image database*, 2013.

[6] José Ignacio Orlando, Joao Barbosa Breda, Karel van Keer, Matthew B.Blaschko, Pablo J.Blanco, and Carlos A. Bulant, *Towards a glaucoma risk index based on simulated hemodynamics from fundus images*, 2018.

[7] Qiao Hu, Michael D. Abràmoff and Mona K. Garvin, *Automated Separation of Binary Overlapping Trees in Low-Contrast Color Retinal Images*, 2013.

[8] J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever and B. van Ginneken, *Ridge based vessel segmentation in color images of the retina*, 2004.