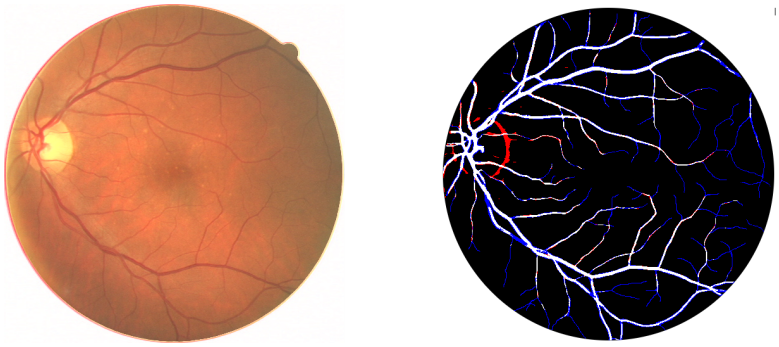
GBM8770
Traitement numérique d'images médicales

Projet: Multi-Scale Line Detector

Automne 2020

Professeur: Benjamin De Leener
Chargés de laboratoire : Vicente Enguix Chiral, Gabriel Lepetit-Aimon



Objectifs :

Dans un premier temps, ce projet vous amènera à implémenter un algorithme de segmentation et des outils d'analyse de cet algorithme. À cette occasion vous allez évaluer la pertinence des métriques de segmentation proposées par l'article.

Dans un second temps, vous serez conduit à concevoir des expériences pour vérifier les hypothèses établies par les auteurs. À partir de ces expériences vous devrez rédiger le résultat de vos expériences ainsi qu'une discussion.

Remise du travail :

Ce travail est à réaliser **en binome** et à remettre sur moodle au plus tard **le 7 décembre**.

L'implémentation du projet doit être en Python. Le fichier de rendu doit être une archive zip contenant tous les fichiers nécessaires pour exécuter le code du projet.

Que ce soit en Matlab ou en Python, le squelette du code de la classe MSLD et des fonctions utilitaires (notamment pour le chargement des données) vous sont fournis. Ces fichiers contiennent des commentaires `TODO` indiquant quelles parties du code sont à compléter et à quelles questions ils correspondent. Un certain nombre d'informations propres à l'implémentation Matlab ou Python sont décrites en commentaire du code, prenez soin de les lire!

Plus que pour les précédents laboratoires, portez votre attention sur la qualité du rapport (rédaction, pertinence de l'argumentation...), en particulier pour les parties III et IV.

Partie I: Implémentation de la MSLD

Préparatifs

Dans un soucis d'optimisation de l'implémentation, nous allons utiliser des convolutions dès que cela est possible. Particulièrement pour le calcul des moyennes d'intensités sur une fenêtre de taille W et le long des lignes de longueur L. De plus ces masques ne seront calculés qu'une seule fois, lors de l'instanciation de la classe MSLD.

Q1. Donnez la valeur du masque permettant le calcul de la valeur moyenne d'une fenêtre de taille `W`. Complétez le constructeur de la classe MSLD en définissant `avg_mask`.

Q2. Expliquez comment construire les masques permettant le calcul de la valeur moyenne le long de lignes de taille `L` pour `n_orientation`. Puis complétez le constructeur de la classe MSLD avec la définition de `line_detectors_masks`.

Vous pouvez dès à présent instancier l'objet: `msld=MSLD(W,L,n_orientation)` en remplaçant les hyper-paramètres `W`, `L` et `n_orientation` par leurs valeurs conseillées par l'article.

Pour entraîner et valider l'algorithme nous allons utiliser la base de données DRIVE.

Q3. Complétez la fonction `load_dataset()` pour qu'elle charge et renvoie la base de données d'entraînement et de test (en incluant pour chaque échantillon: l'image, le label et le masque). Puis affichez l'image, le label et le masque de la première image de l'ensemble d'entraînement.

On notera cette image `I1` dans la suite de l'énoncé.

Basic Line Detector

Q4. À l'aide de la section 3.1 de l'article, implémentez `basicLineDetector(grey_lvl, L)`.

Note: Assurez vous que le padding de la convolution soit pertinent pour éviter les aberrations sur les bords.

Q5. Affichez la réponse du filtre appliqué à l'image `I1` pour les longueurs `L=1` et `L=15`. Comparez les deux et commentez.

Note: Attention le filtre BLD ne s'applique pas à tous les canaux de l'image. Relisez la section 3.1 de l'article pour plus d'informations...

Multi-Scale Line Detector

Q6. À l'aide des section 3.2 et 3.3 de l'article, implémentez `multiScaleLineDetector(image)`.

Q7. Affichez le résultat de l'algorithme appliqué à l'image I1. Comparez avec les réponses du filtre obtenues à la question Q5, et commentez.

Apprentissage du seuil

Pour calculer le seuil donnant la meilleure précision sur l'ensemble d'entraînement efficacement, nous allons utiliser la courbe ROC (Receiver Operating Characteristic). Cette courbe sera étudiée en détail dans la partie II. Pour le moment, il vous faut juste savoir qu'elle associe à chaque seuil possible, le taux de faux positifs et le taux de vrais positifs si ce seuil était choisi.

Q8. Donnez la formule de la précision en fonction du taux de faux positifs `FPR`, du taux de vrais positifs `TPR`, ainsi que du nombre de valeurs positives `P`, de valeurs négatives `N` et du nombre total de pixels S dans l'image de label.

Q9. Implémentez la méthode `roc(dataset)` qui calcule le tpr et fpr associé à chaque seuil pour un dataset donné. Vous pouvez utilisez les fonctions `roc` de Matlab ou sklearn, attention cependant à ne sélectionner que les pixels appartenant au masque.

Q10. À l'aide de la méthode `roc(dataset)` et de la formule de la question 8, complétez la méthode `learnThreshold(dataset)` qui identifie le seuil pour laquelle la précision est la plus élevée.

Q11. Utilisez cette fonction pour apprendre le seuil sur les images d'entraînement. Pour quelle raison faut il absolument conserver une partie des images (l'ensemble de test) et ne pas les utiliser pendant l'entraînement?

Note: il n'est pas anormal d'obtenir un seuil différent de celui annoncé par les auteurs...

Pour éviter cette phase d'entraînement on veut évaluer la possibilité de remplacer le seuil appris par un seuillage dynamique type Otsu.

Q12. Affichez l'histogramme des valeurs prédites par l'algorithme pour l'image I1 et superposez la ligne verticale correspondant au seuil appris. Un seuillage Otsu est-il utilisable pour identifier un tel seuil? Argumentez.

Affichage et Région d'intérêt

Q13. Implémentez la fonction `segmentVessels(image)` qui applique le seuil à la carte de probabilité calculée sur une image. Affichez le résultat de l'algorithme sur l'image I1.

Q14. Vous devriez voir apparaître des erreurs de segmentations aux bords du fond d'oeil. À quoi est dû ce phénomène?

Q15. Pour palier à ce problème, érodez la région d'intérêt (le masque) de 10 pixels, puis effectuez l'apprentissage du seuil à nouveau. Comparer les valeurs du seuil et de la précision globale.

Q16. Complétez la méthode `showDiff(sample)` qui affiche les faux positifs en rouge, les faux négatifs en bleu, les vrais positifs en blanc et les vrais négatifs en noirs. Avec cette fonction, affichez la différence entre la prédiction et le label sur l'image I1.

Partie II: Métriques de Segmentation

Précision Globale et Locale

Q1. Implémentez la fonction `naiveMetric(dataset)` qui évalue la précision et la matrice de confusion de l'algorithme pour un dataset donné. Puis calculez ces métriques sur l'ensemble de test.

(Encore une fois n'oubliez pas de ne sélectionner que les pixels appartenant à la région d'intérêt!)

Q2. Les auteurs proposent une seconde métrique: la précision locale. Quelles raisons avancent-ils pour motiver cette proposition?

Q3. Grâce à une opération morphologique, créez une copie du dataset `test` tel que son attribut `mask` corresponde à la région d'intérêt "locale" proposée par les auteurs. Puis, sans modifier la méthode `naiveMetric(dataset)`, calculez la précision et la matrice de confusion locale.

Q4. Cette seconde métrique met plus en valeur l'algorithme proposé que la précision globale. Au vue des erreurs de l'algorithme révélées à la question I.Q16, donnez une raison supplémentaire (omise par les auteurs) qui explique ce phénomène.

Indice Dice (Bonus)

De nombreux indices existent pour mieux évaluer les performances de segmentation que la précision globale (Précision balancée, Indice de Jaccard, Kappa de Cohen...). Nous allons ici utiliser l'indice Sørensen-Dice. En notant Y les labels et $\hat{Y}$ les prédictions de l'algorithme, l'indice Dice est défini par: $\text{Dice}(Y, \hat{Y}) = 2\frac{Y \cap \hat{Y}}{Y + \hat{Y}}$

Q5.(Bonus) En quoi cette métrique répond aux limitations de la précision globale?

Q6.(Bonus) Implémentez la fonction `dice(dataset)` et calculez sa valeur sur la région d'intérêt globale et locale sur l'ensemble de test.

Courbe ROC et Aire sous la courbe (AUR)

Le choix d'un seuil de segmentation revient à faire un compromis entre faux-positifs et faux-négatifs (un seuil élevé limitera les faux-positifs mais augmentera les faux-négatifs et inversement). Le choix du bon compromis dépend bien souvent de l'application. Cependant les algorithmes se distinguent par le choix des caractéristiques à seuiller (ici un filtre MSLD) plutôt que par la méthode de sélection du seuil... La courbe ROC permet de représenter les performances de l'algorithme indépendamment du seuil choisi.

Pour construire cette courbe, on calcule le taux de faux-positifs (FPR) et le taux de vrais-positifs (TPR=$1-$FNR) pour chaque valeur de seuil. La courbe obtenue (FPR en abscisse, TPR en ordonnée) caractérise l'efficacité de l'algorithme à distinguer les vaisseaux du fond, indépendamment du seuil.

Enfin, pour simplifier la comparaison entre deux courbes ROC, on extrait leurs aires sous la courbe (AUR).

Q7. Que signifie une AUR de 1, de 0,5 ou de 0 pour les performances du modèle?

Q8. Implémentez la méthode `plotROC(dataset)` qui trace la courbe ROC et calcule son AUR. Puis faites de même pour la région d'intérêt globale et pour la région d'intérêt locale. Comparez les résultats et commentez.

Partie III: Validation de l'hypothèse de recherche de l'article

À l'aide de votre implémentation de l'algorithme et des différents cas du dataset de test, concluez sur la validité de l'hypothèse de recherche de l'article (tel que vous l'aviez formulé dans le).

Partie IV: Discussion

Q1. Discutez du choix des hyper-paramètres `W`, `L` (nombre d'échelles et répartition) et `n_orientation`. Leurs valeurs ont-elles des justifications théoriques? Quels sont leur impact en pratique?

Q2. Les auteurs ont choisi le seuil donnant *la meilleure précision* sur l'ensemble d'entraînement. Connaissant les analyses effectuées dans la partie II, discutez de ce choix.

Q3. En vous appuyant sur vos expérimentations pour les deux dernières parties de cet énoncé, proposez des recommandations pour améliorer l'algorithme.