INF8725 - Traitement de signaux et d'images

TP3 - Restauration, segmentation et image couleur

Automne 2016

Chargé de laboratoire : Hamza Bendaoudi

Objectifs :
Ce laboratoire a pour objectif de manipuler et analyser les différentes techniques de restauration et de segmentation d'image en niveaux de gris et en couleur.

Remise du travail :
La date de remise est le **30 novembre et le 07 décembre à 23h55 pour le groupe B2 et B1, respectivement.** Une pénalité de 3 points par jour sera appliquée lors d'un retard.

Documents à remettre :
Les exercices doivent être codés dans un fichier TP.m. Les réponses aux questions doivent être incluses dans le code. Les exercices doivent être séparés par des cellules (*Insert cell divider* ou %%). Vous devez bien identifier chaque exercice et sous-question, et bien commenter le code.

Créer un fichier html à l'aide de Publish to html de Matlab pour avoir un fichier html de votre code et de vos graphiques. Veuillez remettre tous vos fichiers (.m et dossier html) dans un seul fichier zip et nommez ce fichier selon vos matricules (Mat1_Mat2.zip).

Pour inclure les fonctions dans le html, ajouter 'type fonction.m' dans votre .m principal.

Vérifier également que les graphiques et les figures sont lisibles dans le html.

Une pénalité de 3 points sera appliquée si ces consignes ne sont pas respectées.

Première partie :

Exercice 1 (5 points) : Restauration d'image

Une voiture de course conduit par un individu se croyant sur un grand prix de formule ford a été photographiée par un radar automatique. La vitesse d'obturation de la caméra était 30 pixels/secondes trop lente : l'image est floue. Bien gêné de sa faute, il vous demande de l'aider à restaurer l'image pour en extraire des informations pertinentes à l'enquête.

1. (0.5 point) Chargez l'image *Formula_Ford.png* et affichez-là.
2. (1.5 point) Construisez la PSF nécessaire pour déconvoluer votre image. Attention, vous devez calculer les bons paramètres pour la fonction fspecial.
3. (1 point) Déconvoluez l'image avec le filtre inverse. Utilisez la fonction deconvwnr avec le paramètre K à 0. Affichez l'image restaurée et commentez sur le résultat.
4. (1 point) Sachant que l'écart type du bruit gaussien dans l'image est d'environ 1e-3, trouvez une meilleure estimation du paramètre K, sachant que K représente le ratio du bruit et du signal (*NSR*).
5. (1 point) Déconvoluez l'image bruitée avec votre PSF et le filtre inverse avec votre nouveau paramètre K. Affichez l'image restaurée. Que pouvez-vous transmettre à l'agent ?

Exercice 2 (6 points) : Segmentation d'image

Vous devez implémenter le filtre de Canny, puis tester ce filtre sur une image qui a déjà été segmentée au préalable.

1. (0.5 point) Chargez l'image *escaliers.jpg*. Affichez cette image.
2. (2 points) Implémentez la fonction *Filtre_Canny*. Cette fonction prend en paramètre l'image à segmenter, un masque gaussien (*3x3, sigma=0.5*), et le seuil *Th* pour l'étape de binarisation. Elle retourne une image binaire représentant les contours segmentés. Référez-vous aux notes de cours pour l'algorithme de Canny.
3. (1 points) Implémentez la fonction *Calculer_Precision* qui prend en paramètre l'image segmentée et une image de référence. Cette fonction retourne la

performance, le taux de faux positifs et le taux de faux négatifs. Voici comment calculer ces valeurs :

$$Performance = \frac{ContoursCorrects}{ContoursCorrects + FauxPositifs + FauxNegatifs}$$

$$TPF = \frac{FauxPositifs}{ContoursCorrects + FauxPositifs + FauxNegatifs} \quad (1)$$

$$TFN = \frac{FauxNegatifs}{ContoursCorrects + FauxPositifs + FauxNegatifs}$$

ContoursDetectes = Nombre de pixels contour dans l'image segmentée
ContoursReferences = Nombre de pixels dans l'image de référence
ContoursCorrects = Nombre de pixels correctement détectés (ImageSegmentée ∩ ImageRéférence)
FauxPositifs = Nombre de pixels détectés comme contour dans l'image segmentée, mais pas dans l'image de référence (ContoursDetectes - ContoursCorrect)
FauxNegatifs = Nombre de pixels non détectés dans l'image segmentée, mais détectée dans l'image de référence (ContoursReferences - ContoursCorrects)

4. (1.5 points) Chargez l'image de référence *escaliers_TrueSeg.jpg*. Attention, à cause du bruit JPEG, il faut binariser cette image. Utilisez un seuil de 128. Segmentez escaliers.jpg avec votre filtre de Canny en utilisant un masque gaussien de taille 3x3 et de sigma égale à 0.5. Variez le seuil pour le filtre de Canny et regardez les résultats de la fonction *Calculer_Precision*. Que remarquez-vous ?
5. (1 points) Avec la fonction que vous venez d'implémenter, utilisez la transformée de Hough pour détecter seulement les lignes verticales (avec une marge angulaire de +- 1 degrées). Superposez ces lignes sur l'image originale et affichez le résultat.

Deuxième partie :

Exercice 3 (5 points) : Segmentation par couleur

La segmentation par couleur consiste à regrouper des pixels qui partagent un même critère de similarité. Pour cet exercice, notre critère de similarité sera l'appartenance à une section du cube RGB. Nous allons donc diviser le cube RGB en plusieurs sections, puis attribuer une couleur unique à ces sections. Alors, tous les pixels se verront attribuer une couleur correspondant à la section dont ils font partie.

1. (0.25 point) Chargez l'image *chateau.jpg* puis affichez-la.
2. (1 points) Implémentez la fonction *ObtenirLUT* qui servira à discrétiser votre cube RGB en I X J X K segments. Cette fonction prend en entrée le nombre de segment voulu, puis retourne la table de conversion (*look-up table (LUT)*). Une table de conversion spécifie une nouvelle valeur pour chaque valeur d'intensité. Dans notre cas, nous allons utiliser la moyenne des segments comme valeur de correspondance. Par exemple, pour une table avec deux segments, les intensités allant de 0 à 127 auront 63 comme valeur de correspondance, tandis que les intensités allant de 128 à 255 auront la valeur 191.
3. (2 points) Implémentez la fonction *Segmenter_Couleur* qui vous permettra de segmenter vos couleurs en fonctions des LUT que vous aurez calculées. Cette fonction prend en entrée une image couleur et trois LUT, une pour chaque composante RGB. Elle retourne ensuite l'image segmentée. Vous allez devoir séparer votre image en ses trois composantes, les traiter individuellement, puis les recombiner. Les images en Matlab sont stockées en matrices tridimensionnelles, selon le format H X L X P (*hauteur x largeur x profondeur*). Il suffit donc de récupérer les valeurs de la troisième dimension pour obtenir la composante voulue. Pour traiter chaque composante, vous pouvez utiliser la fonction *intlut* de Matlab.
4. (0.75 point) Segmentez l'image chateau.jpg en 8 segments pour chaque composante. Affichez et commentez le résultat. Combien allez-vous gagner d'espace entre l'image RGB originale et votre image quantifiée ?
5. (1 point) Déterminez le nombre de segments utilisés pour chaque composante afin d'obtenir l'image ci-dessous :

FIGURE 1 – Image segmentée

Exercise 4 (4 points) : Toon/Paint shading

Le principe du toon/paint shading consiste à transformer une image normale en un rendu de type dessin ou peinture. Ces techniques sont souvent utilisées en infographie pour faire du rendu non-photoréaliste. Nous verrons ici une technique simple basée sur les aplâts de couleurs et la détection de contours.

1. (0.5 point) Chargez l'image *Albert-Einstein.jpg* et affichez-là.
2. (0.5) Utilisez vos fonction *ObtenirLUT* et *Segmenter_Couleur* pour segmenter l'image en quelques tons par composante (de 3 à 10 devrait suffir pour obtenir un effet de peinture/dessin). Affichez l'image résultante.
3. (1) Appliquez un flou gaussien sur votre image segmentée. Utilisez la fonction *fspecial* avec une taille de 7 et une variance de 1. Affichez le résultat. Utilisez la fonction *convn* pour faire une convolution de votre filtre sur une image couleur.
4. (1) Appliquez un filtre de Canny avec la fonction edge sur chacune des composantes (il n'est pas possible d'appliquer edge sur une image couleur) de votre image **originale**. Vous aurez 3 réponses, une pour chaque composante que vous pourrez moyenner. Ensuite, inversez le résultat pour que votre fond d'image soit blanc et que vos contours soient noirs. Afficher le résultat. Note : jouez avec les seuils de Canny pour obtenir des beaux contours.

5. (0.5) Appliquer le même flou gaussien à votre image Canny. Vous pouvez utiliser *conv2* cette fois-ci. Affichez le résultat.
6. (0.5) Multipliez chaque composante de votre image couleur segmentée floue avec les contours flous calculés en 5. Affichez le résultat. N'oubliez pas que vos images devront être au format *double* lors de la multiplication.

Ceci devrait vous donner une image qui ressemble à une peinture/dessin. Prenez notes que ce n'est qu'une façon très simpliste d'y arriver, et qu'il existe des techniques plus poussées [1]. Néanmoins, vous devriez obtenir des résultats satisfaisants :

FIGURE 2 – Image dessin avec 4X3X4 segments par couleur, seuil de Canny à 0.3

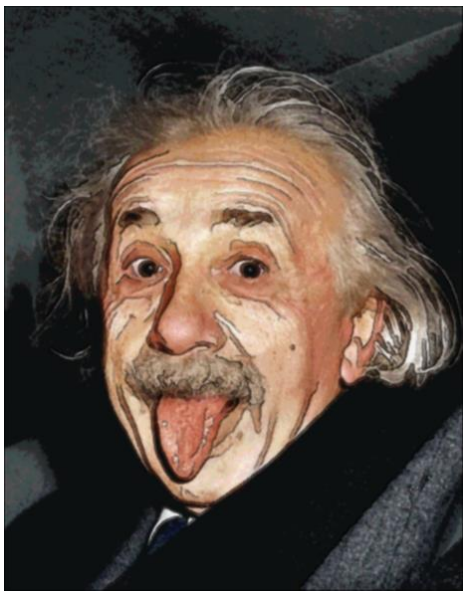

FIGURE 3 – Image peinture avec 10X8X8 segments, seuil de Canny à 0.25

1. http://wwwisg.cs.uni-magdeburg.de/~stefans/npr/overview-summary.html