Compte- Rendu TP1 TSI

Sommaire

1. Filtrage à partir d’un histogramme : seuillage par maximisation de la variance interclasses
2. Filtrage d’ordre : filtre médian
3. Filtrage passe-haut dans l’espace de Fourier
4. Filtrage passe-haut dans l’espace direct : détection de contours
5. Conclusion
6. Filtrage à partir d’un histogramme : seuillage par maximisation de la variance interclasses

Problématique :

Lors de cet exercice, nous utiliserons l’image « cellules.jpg ». La base de cette méthode est un outil utilisé par les statisticiens pour connaître le degré de séparation de 2 ou plusieurs sous populations obtenues à partir d'un même ensemble de témoins : la variance interclasse. Dans cette première partie, le filtrage/seuillage est effectué à partir de la seule connaissance de l’histogramme de l’image.

Méthode :

Dans un premier temps, nous allons télécharger l’image puis nous allons convertir l’image en niveau de gris afin de calculer l’histogramme et l’afficher à l’aide des fonctions « *imread*», « *imshow »*, *« rgb2gray* » et « *imhist* ». Cet histogramme va nous permettre de créer une fonction prenant en paramètre l’histogramme qui affiche d’une part la courbe de variance interclasses en fonction du seuil et retourne, d’autre part, la valeur de seuil qui maximise celle-ci. La méthode est la suivante, on choisit un seuil s entre 0 et 255, on calcule la variance interclasses V(s) avec la formule fournie dans l’énoncé puis on détermine ensuite le maximum de V(s). On rappelle la formule :



* P**0** (respectivement P**1**) est la probabilité d’appartenir à la classe fond (resp. Objet)
* m**0** (respectivement m**1**) est la valeur moyenne de la classe fond (resp. Objet)
* m est la valeur moyenne de l’image.

Pour déterminer ces valeurs, on a tout d’abord déterminé le nombre de pixels en sommant l’histogramme (fonction « *sum* »). On ne fixe pas le seuil, (il varie entre 0 et 255 = -1,) on stockera les valeurs dans des tableaux. On utilisera des boucles for. Pour calculer m**0** (respectivement m**1**), il faudra déterminer N**0** (respectivement N**1**).

On détermine le maximum de V(s) avec la fonction « *find* » et « *max* ».

Résultats :

Ainsi, on peut clairement observer que les niveaux de gris les plus présents dans l’image se situent …... ce qui correspond à la présence des objets dans l’image.

1. Filtrage d’ordre : filtre médian

Problématique :

Le filtre médian est un filtre numérique non linéaire, souvent utilisé pour la réduction de bruit. La réduction de bruit est une étape de prétraitement classique visant à améliorer les résultats de traitements futurs (détection de bords par exemple). La technique de filtre médian est largement utilisée en [traitement d'images](https://fr.wikipedia.org/wiki/Traitement_d%27images) numériques car il permet sous certaines conditions de réduire le bruit tout en conservant les contours de l'image.

Généralités :

L’idée de fonctionnement du filtre médian est la suivante. Prenons une matrice M comme suit avec

Le filtre médian doit trier les valeurs des 9 pixels par ordre croissant et prendre la valeur médiane c’est-à-dire la cinquième valeur. Ce filtre permet de modifier une valeur aberrante sans faire de calcul de moyenne. Il respecte les contours. Pour réaliser ce filtre, on utilisera la fonction « *ordfilt2* ».

Le bruit poivre et sel est une dégradation de l’image sous la forme de pixels noir et blanc répartis au hasard.

Résultats :

de

1. Filtrage passe-haut dans l’espace de Fourier
2. Filtrage passe-haut dans l’espace direct : détection de contours

Problématique :

En traitement d’image il nous ait parfois demande de réaliser un filtrage passe haut pour ne récupérer que les hautes fréquences. En image les hautes fréquences sont représentées par la répétition de même motif comme des lignes. Ce type de filtrage permet aussi de détecter les contours d’une image. Pour ce faire un des moyens les plus simples qui renvoi des résultats satisfaisants est le filtrage de Sobel. Comment fonctionne-t-il et quels sont ses améliorations.

Généralités

Le filtre de Sobel est défini par ce masque matriciel :

Pour les contours horizontaux.

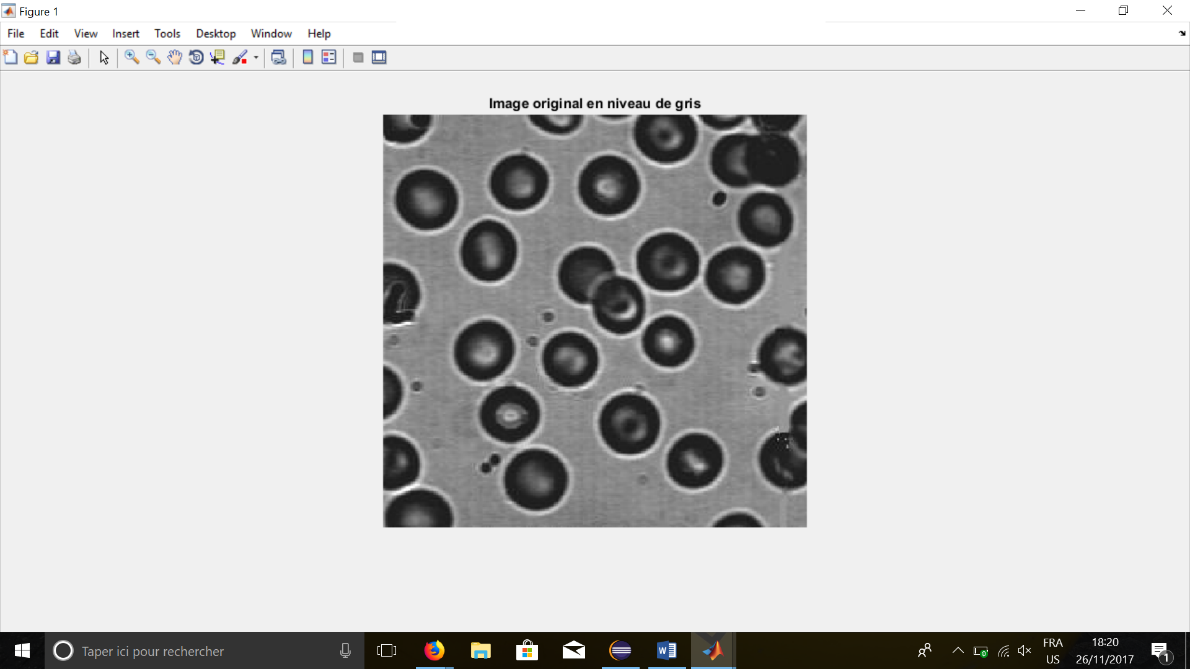
Pour les contours verticaux.

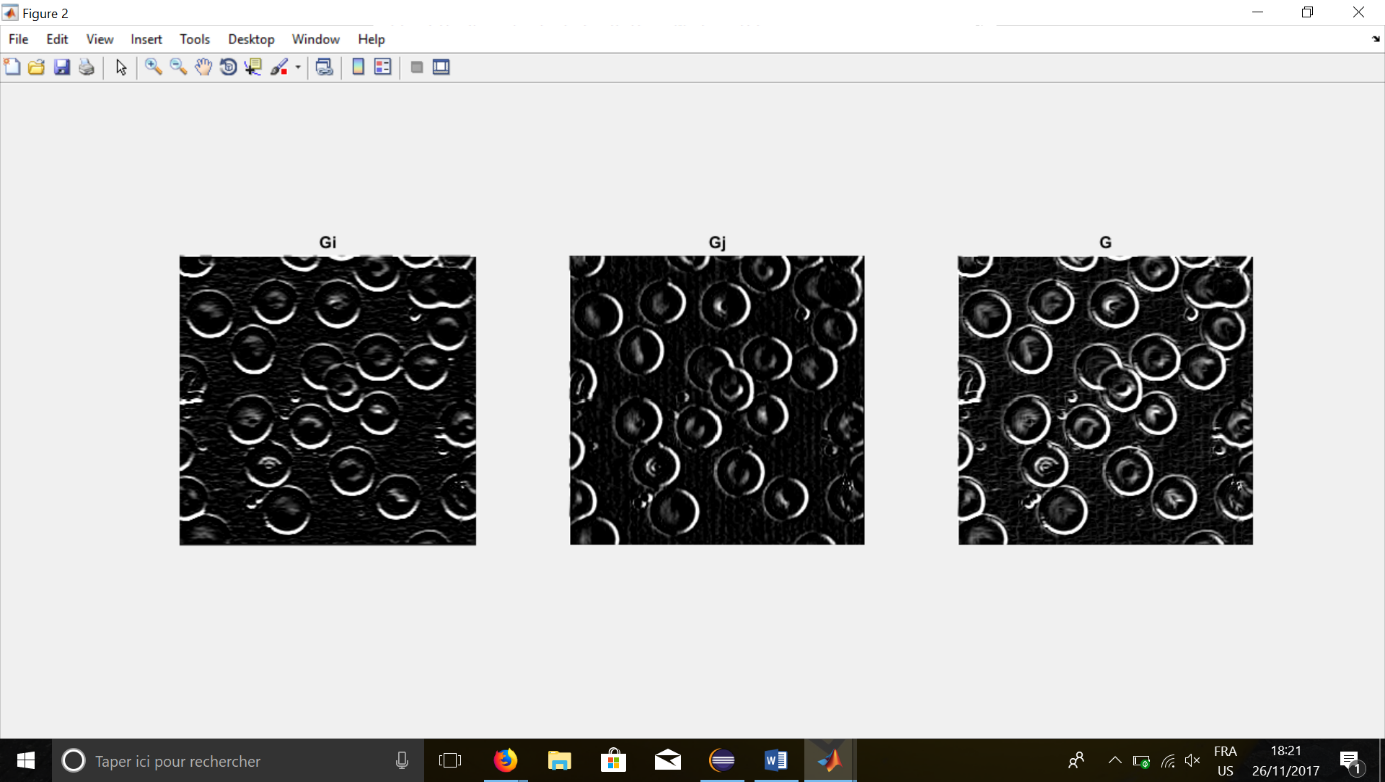
Le masque d’un filtre de Sobel contient un filtre de lissage. Ce lissage est utilisé pour atténuer le bruit de l’image (notamment avec un passe-haut) qui risque de corrompre l’information matricielle.

Applications

Pour appliquer ce filtre à notre image (ici cellules.jpg), nous devons avant tout passer notre image en niveau de gris pour avoir des matrices 2D. Puis nous réalisons un filtrage de Sobel vertical puis horizontal. Nous récupérons les gradients dans une matrice puis nous réalisons la norme des composantes pour restituer notre image filtrée.

Voici les résultats après filtrage :





Gi correspond à l’image après filtrage de Sobel horizontal.

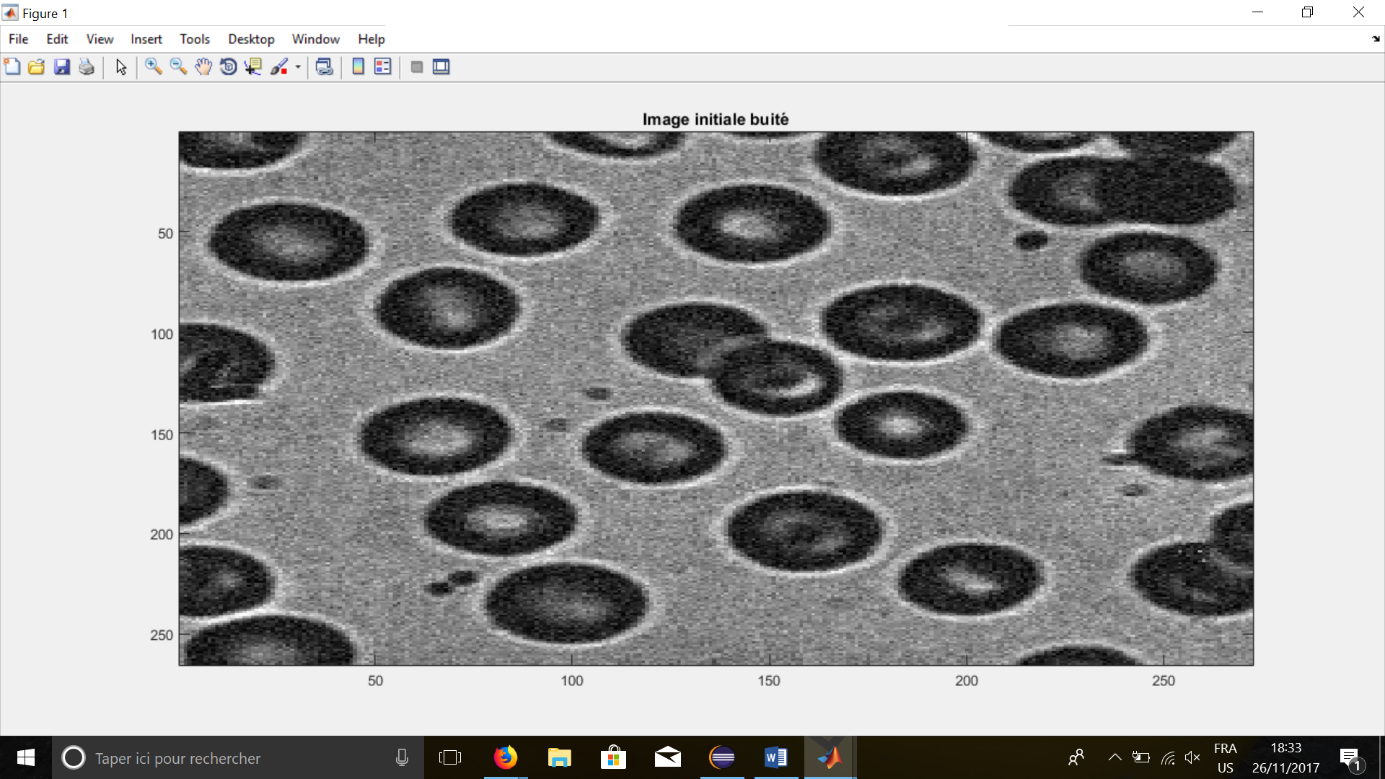
Gj correspond à l’image après filtrage de Sobel vertical.

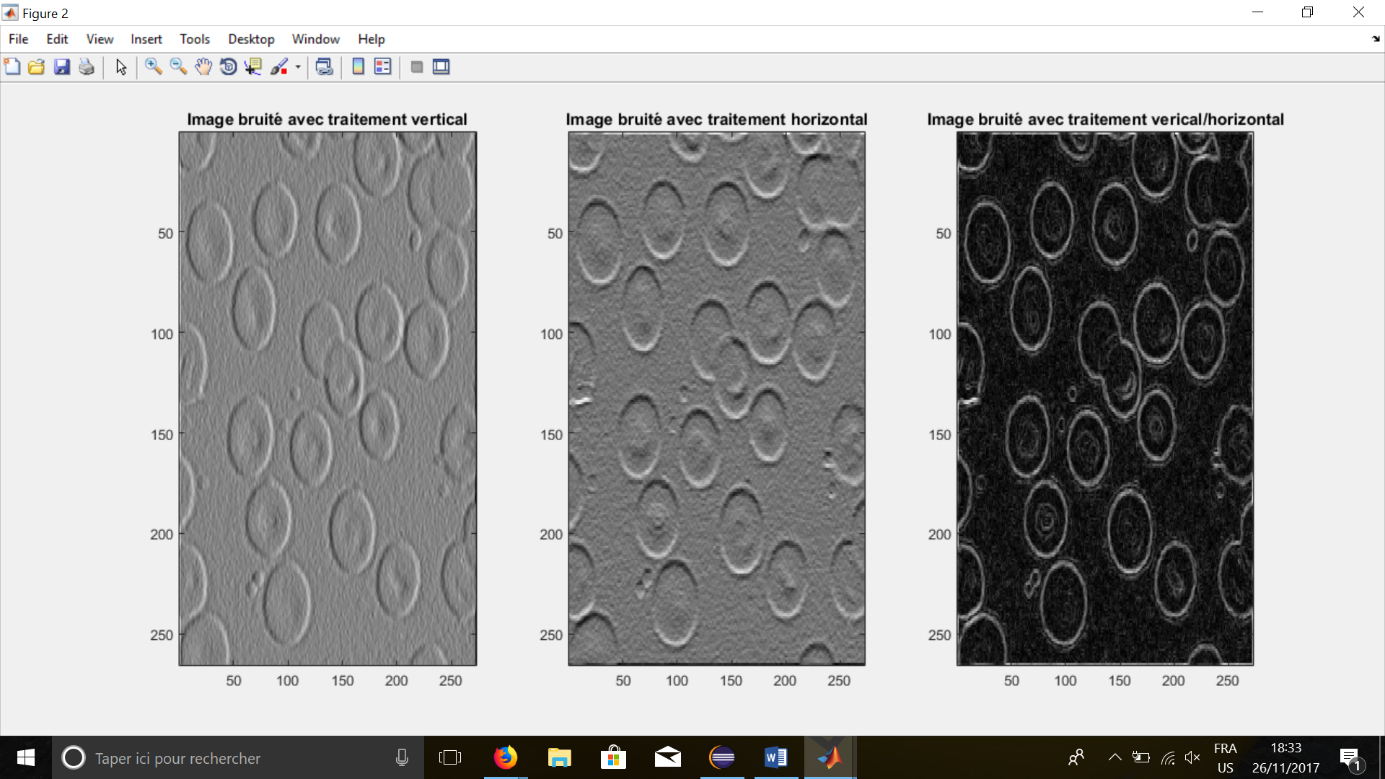
G correspond à l’image après la réunion des deux filtrages précédant.

Le filtrage a en effet fonctionné, les contours apparaissent mais il manque des informations. En effet ce n’est pas tout le contour de la cellule qui est contraste. De plus nous pouvons voir à l’intérieur des cellules des contrastes de niveaux de gris, nous n’avons pas encor de coupures franches au niveau des contours.

Amélioration :

Nous refaisons le même traitement en ayant ajoute à l’image principale un bruit gaussien centrée de variance 0,01 :



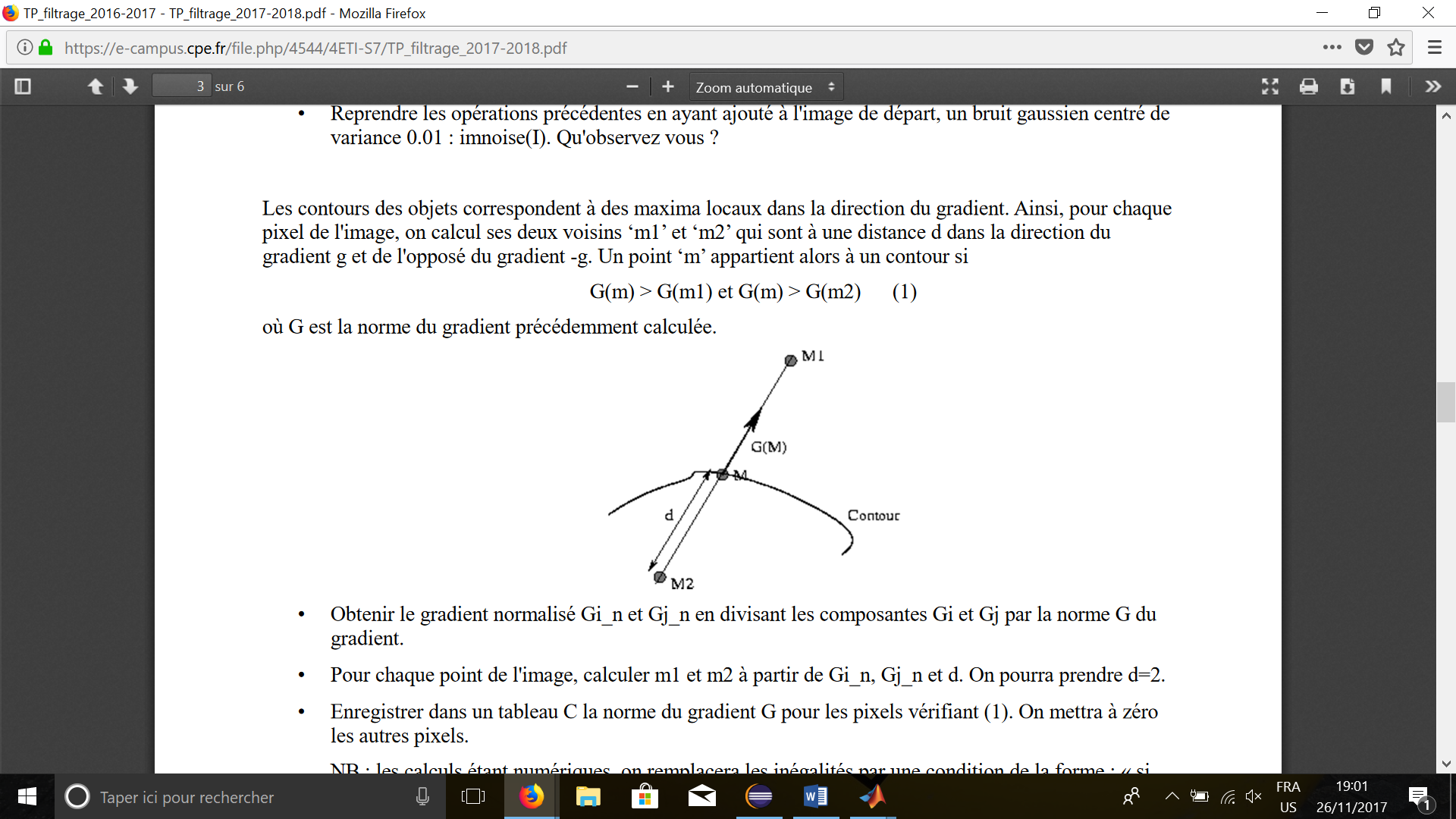


Ici bruiter notre image a permis d’enlever des détails à l’original, ainsi traiter l’image bruitée avec le filtre de Sobel est plus aisé et nous rend un résultat plus satisfaisant. Nous observons les contours de chaque cellule avec précisions et pouvons ainsi les dénombrer.

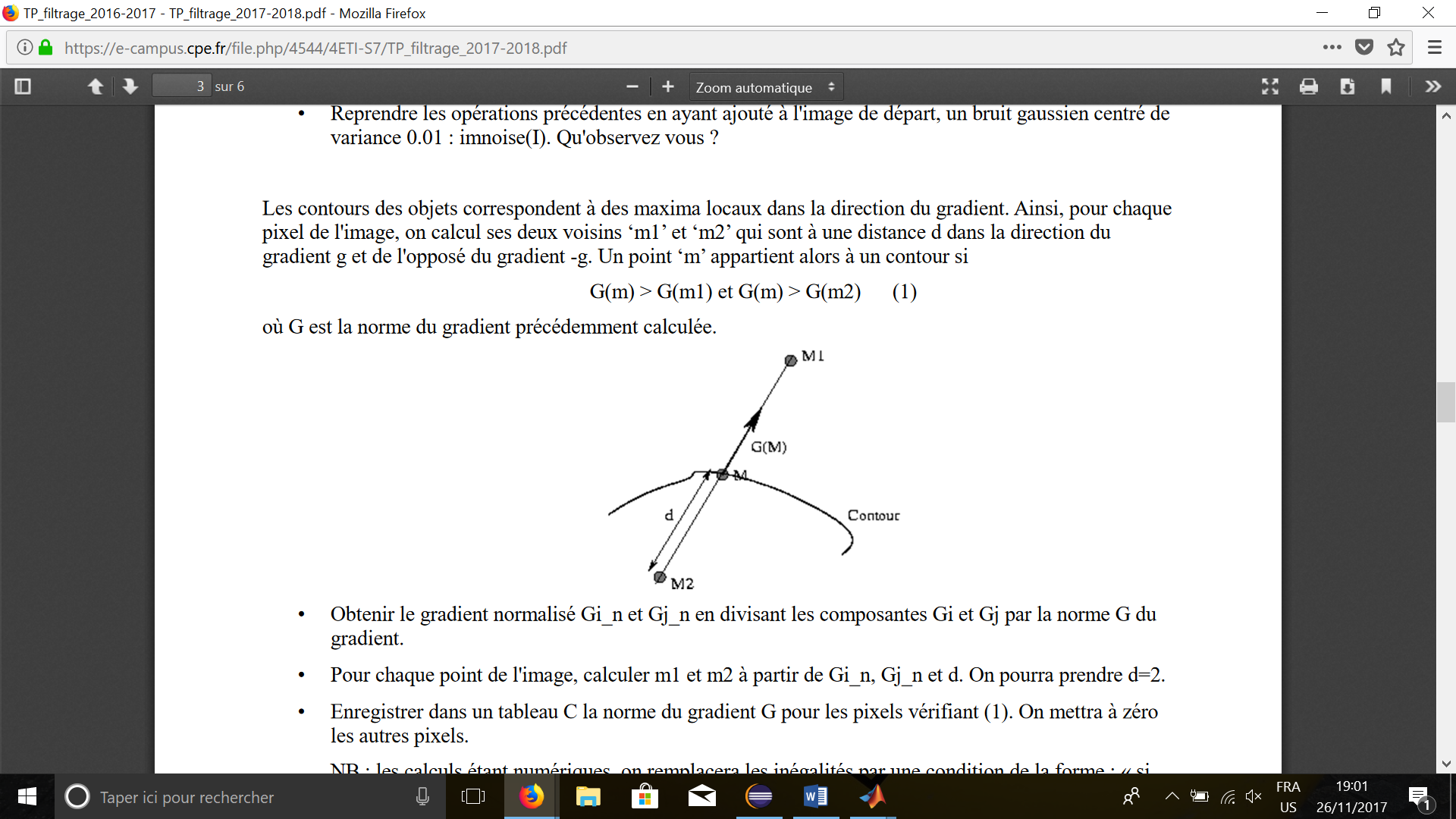
L’amélioration ici faite est sur la totalité du contour affiché. Cependant nous avons toujours une information sur l’intérieur des cellules qui apparait, cela risque de gêner l’analyse faite après le traitement.

Nous choisissons maintenant de « seuiller » notre contour. Pour ce faire, nous voulons pour chaque points M appartenant à l’image savoir si celui-ci fait partie du contour. Nous créons alors deux points temporaires M1 et M2. Ces points sont à une distance d dans la direction du gradient g et de l’opposé du gradient -g. Chaque point M appartient au contour si et seulement si :

G(M)>G(M1) et G(M)>G(M2)



*Source : TP1 de TSI de JP. Bruandet*



Gj\_n

Gi\_n

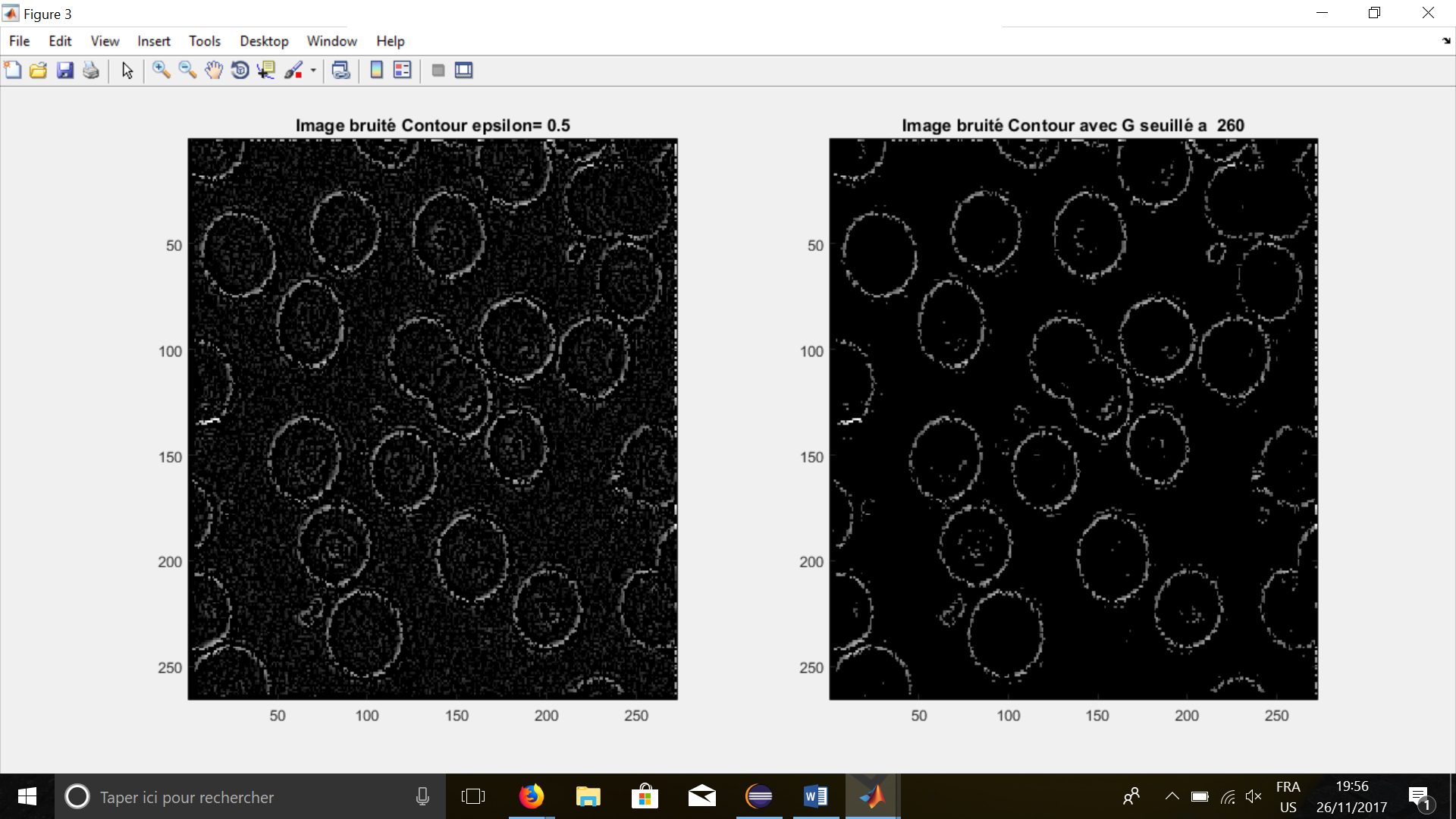
Ө

Les coordonnes de M1 et M2 sont :

M1

M2 avec

De plus nous rajoutons une condition pour qu’un point puisse appartenir à un contour il faut que son gradient soit supérieur à un seuil. Cela permet de supprimer les pixels « entre deux » et homogénéiser l’image :



Cette amélioration permet d’atténuer les composantes à l’intérieur des cellules. Cependant nous perdons la précision sur le contour de l’image. En effet en réduisant la « fenêtre » d’un contour, si celle-ci est mal choisi alors nous supprimons le point appartenant au contour. Il faut donc faire un compromis entre un contour précis sans perte d’information avec possibilité d’affichage de « non-contour » et une présence exclusive de contour mais des pertes de motif sur celui-ci.

En rajoutant la condition de seuil, cela permet de supprimer le bruit afin d’avoir un fond unicolore pour optimiser l’analyse de l’image.

Conclusion

A travers cet exercice nous avons pu réaliser un filtrage passe-haut dans l’espace direct pour réaliser une détection de contours. Nous avons vu la méthode du filtre de Sobel. Celle-ci est simple à mettre en place mais comporte des limites. Elle nécessite des améliorations pour pouvoir n’afficher que les contours et supprimer toutes composantes autres que les contours. Cependant malgré les améliorations misent en place il nous reste toujours à effectuer des compromis entre la qualité générale du traitement de l’image et la qualité du contour. La solution choisie dépendra de l’application.