

[HAI927I] Projet Image - Compte Rendu n°6

Projet #12.3 : Débruitage d'images

1. Analyse des résultats

Nous avons réussi à générer et afficher tous les résultats que nous voulions. Toutes les données sont présentes dans le dossier root/code/DB/CID22_pgm_M. Toutes les courbes sont présentes dans le dossier compressé root/code/DB/Plot.zip.

A partir de ces courbes, on souhaite étudier la différence des mesures selon le type de bruit appliqué, la méthode de débruitage utilisée et les paramètres employés. Dans le cadre de ce projet, nous étudions seulement le PSNR car il nous reste peu de temps et c'est une mesure que nous avons beaucoup manipulée durant ce master.

Dans le dossier root/code/DB, vous pouvez trouver les notes prises lors de l'analyse des résultats. Ces notes seront mises aux propres afin de les présenter lors de la soutenance. Voici quelques phénomènes que nous avons pu observer :

- Concernant les filtres, il est préférable de prendre une fenêtre 3x3 la plupart du temps.
- L'algorithme Non Local Means offre le meilleur PSNR quand le bruit est très faible.
- La meilleure valeur du paramètre puissance dans le filtre moyenné pondéré dépend de la puissance du bruit présent dans l'image. Avec un bruit faible, on privilégie une forte puissance (dans notre cas 2). Avec un bruit fort, on privilégie une faible puissance (dans notre cas 1.5).
- Concernant le filtre gradient, le filtre moyenné offre un meilleur PSNR que le filtre gaussien dans tous les cas étudiés (environ 5 dB de différence).
- Le meilleur filtre pour retirer du bruit poivre et sel est le filtre médian (voisins = 1, si l'intensité du bruit est inférieure ou égale à 20% (PSNR = [31, 26]) et voisins = 2, sinon (PSNR = [24.5, 23.25]).

- Le meilleur filtre pour retirer du bruit poisson est le filtre moyennneur pondéré (voisins = 5, puissance = 2) (PSNR = [37.5, 18.75]).
- Pour le bruit Speckle, la meilleure méthode pour le retirer est l'algorithme Non Local Means si l'intensité est inférieure à 10 (voisins = 1) (PSNR = [45, 35.5]), le filtre pondéré (voisins = 5) ((puissance, intensité, PSNR), (2, 10, 32.75), (1.25, 15, 30)) sinon si l'intensité du bruit est inférieure à 20 et le filtre gaussien (voisins = 1, moyenne = 0, écart-type = 2.5) (PSNR = [29, 27.4]) sinon.
- Pour le bruit gaussien, la meilleure méthode à utiliser est l'algorithme Non Local Means si l'écart-type est inférieur à 10 (voisins = 1) (PSNR = [46, 30]), le filtre médian sinon si l'écart-type est inférieure à 25 (voisins = 1) (PSNR = [29.25, 25.25]) et le filtre médian sinon (voisins = 2) (PSNR = 24.2 pour un écart-type de 35).
- Pour le bruit impulsif, le meilleur filtre est le filtre pondéré si l'intensité du bruit est inférieure ou égale à 2.5 (voisins = 1, puissance = 2) (PSNR = 34.5), le filtre médian sinon si l'intensité est inférieure ou égale à 15 (voisins = 1) (PSNR = [31, 27.75]) et le filtre médian sinon (voisins = 2) (PSNR = [25.5, 21.5]).

Maintenant l'analyse terminée, nous aimerions comparer les PSNR trouvés avec le PSNR donné par le réseau de neurones. Cependant, une image est débruitée en 30/60 secondes et chaque mesure de PSNR précédemment obtenue est une moyenne sur 100 images. On ne peut donc pas utiliser la même méthode de comparaison utilisée jusqu'à maintenant. De plus, il faudrait créer d'autres scripts générant les images alors qu'il nous reste peu de temps.

Nous avons pensé à prendre une image en particulier (l'lena par exemple) et comparer le PSNR de l'image débruitée avec le réseau de neurones et le PSNR de la même image débruitée avec une méthode traditionnelle avec les meilleurs paramètres trouvés.

2. Mise à jour du poster

Nous avons développé les parties écrites du poster, fait un peu de mise en page et changé la police. Cependant, nous pensons qu'il y a un peu trop de texte. Nous souhaitons donc demander à M.DIBOT des conseils pour trouver une solution à ce problème et pour peut-être améliorer notre poster sur d'autres aspects.

Projet de débruitage d'images

HAI9271 : Projet Image




LIRZIN Léo – SERRANO Léa

Dans le contexte du traitement d'images, le bruit numérique est une dégradation que subit l'image de l'instant de son acquisition jusqu'à son enregistrement et potentiellement lors de sa transmission dans un réseau. Il peut être produit, par exemple, un manque de luminosité dans la scène prise en photo, une haute température des capteurs photographiques, une mauvaise qualité des capteurs photographiques ou bien le bruit présent sur un réseau.

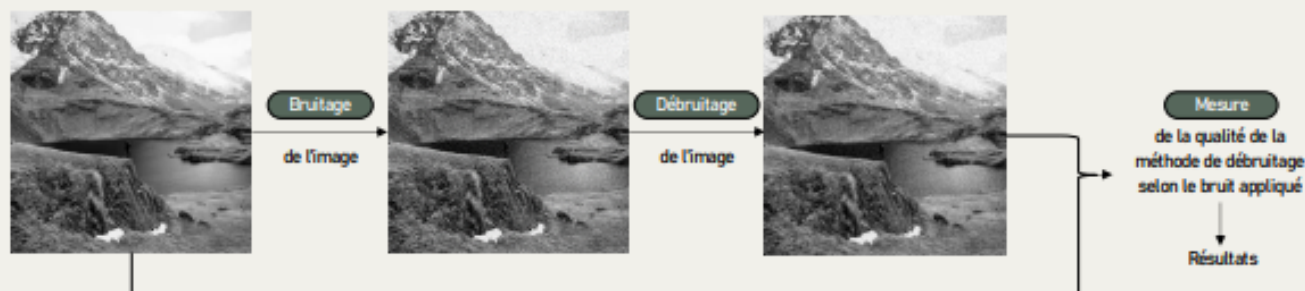
Le bruit génère donc des défauts dans l'image en modifiant à tort la couleur de certains pixels. Le but du débruitage est d'enlever ces défauts afin d'obtenir une image ressemblant le plus à l'image "en clair" (c'est-à-dire, l'image n'ayant pas de bruit). Cependant, le débruitage a tendance à flouter l'image bruitée, ce qui réduit sa qualité. On définit donc les caractéristiques d'une bonne méthode de débruitage : la protection des bordures, la préservation des textures, le lissage des zones uniformes et la non-génération d'avantages de défauts.








Notre projet s'intéresse aux méthodes de débruitage et leur efficacité selon le type de bruit traité.

Dans un premier temps, nous générons notre propre bruit afin de contrôler l'intensité et le type de bruit appliqué.

Ensuite, nous appliquons plusieurs méthodes de débruitage que nous avons implémentées aux images bruitées. Ces méthodes sont divisées en trois groupes : des  filtres, un  algorithme et un  réseau de neurones.

Finalement, nous mesurons la qualité du débruitage à partir de l'image de base et l'image bruitée débruitée selon plusieurs métriques. De ces mesures, nous traçons des courbes et déduisons les meilleures méthodes de débruitage selon chaque type de bruit.



Bruitage	Débruitage	Mesure
<p>Poisson et sel : simule le mal fonctionnement des capteurs ou l'inefficacité d'un programme de traitement d'image</p> <p>Gaussien : simule les défauts présents dans des images prises avec peu de lumière ou transmises sur un réseau bruité</p> <p>Poisson : simule le manque ou l'irrégularité de l'arrivée de photons sur des capteurs</p> <p>Speckle : simule le bruit communément trouvé dans les images médicales et les images acquises avec des radars</p> <p>Impulsif : simule une corruption de la mémoire de stockage, une instabilité du signal électrique ou des erreurs lors de la transmission</p>	<ul style="list-style-type: none"> Moyenneur : moyenne des voisins d'un pixel Moyenneur pondéré : moyenne des voisins d'un pixel pondérée par leur similarité Gaussien : moyenne des voisins d'un pixel pondérée par leur proximité Gradient : moyenne des voisins d'un pixel en respectant les contours devinés Médian : remplacement du pixel par la valeur qui sépare au milieu l'ensemble des valeurs des voisins Non-Local Means : calcul d'une valeur d'un pixel avec moyenne pondérée de patches extraits de l'image Transformer : réseau de neurones entraîné pour débruite des images	<p>PSNR : évalue la ressemblance entre deux images</p> <p>SNR : estime la cohérence d'une image</p> <p>SSIM : calcule la similarité de structure entre deux images</p> <p>RMSE : calcule la distance entre deux images</p>

Un réseau de neurones est un ensemble d'algorithmes inspirés par le cerveau humain. Le but d'un réseau de neurones est de simuler l'activité du cerveau humain, et plus spécifiquement la reconnaissance de motifs et la transmission d'informations entre les différentes couches de connexions neuronales.

Le réseau de neurones implémenté pour le débruitage s'appelle Restormer et est un réseau de neurones récurrent qui utilise la technique du restorming. Il se spécialise dans la restauration d'images, telles que le débruitage, le défloutage, le dépluie et le débrouillage.

Conclusion :

- Meilleures méthodes et paramètres
- Graphe des meilleurs résultats trouvés
- Promotion logiciel



Lien GitHub vers le projet : https://github.com/LIRZIN/HAI927-Projet_IMAGE

Lien GitHub vers le réseau de neurones utilisé : <https://github.com/swz30/Restormer>

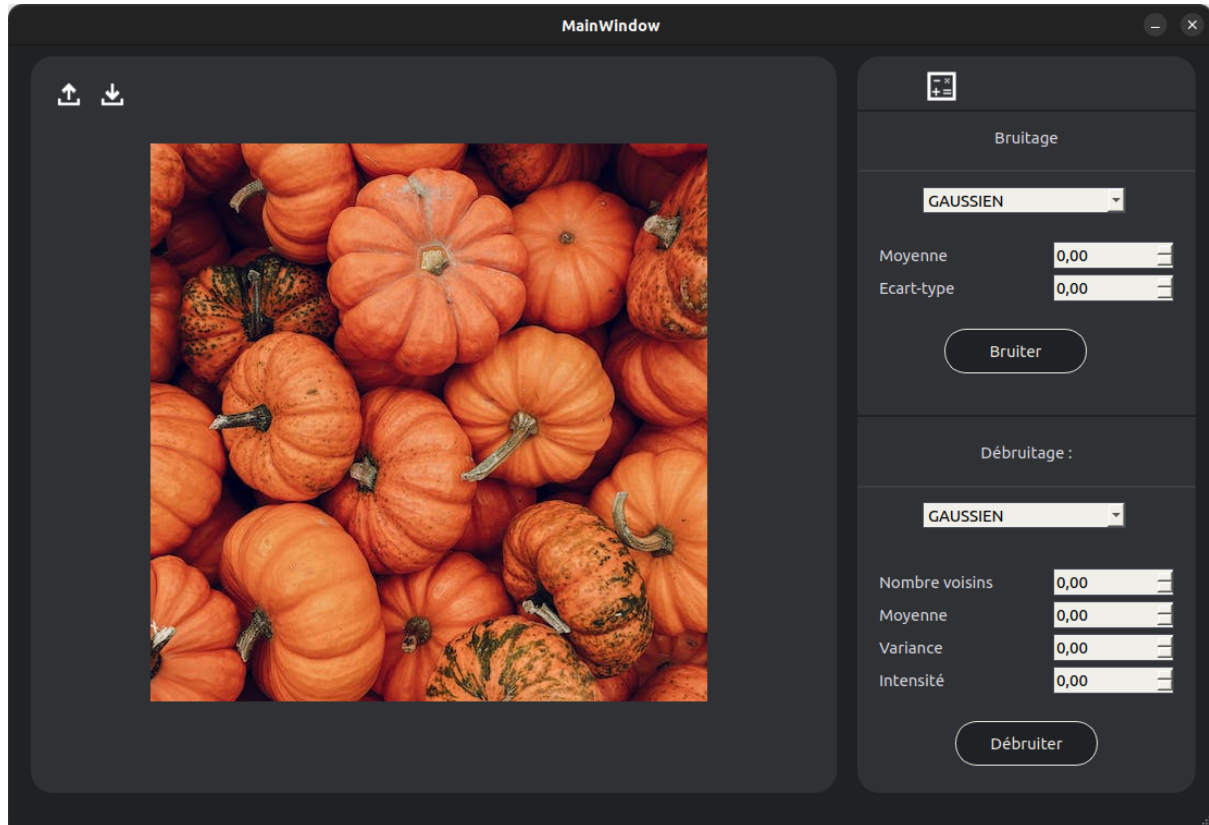
Supervisé par : William PUECH

Nicolas DIBOT

3. Avancement de l'application QT

Nous avons mis en place l'application QT qui nous permet d'importer une image (en pgm ou ppm), de la bruite et débruiter en utilisant nos algorithmes et de la télécharger (en pgm, ppm, png, jpg/jpeg).

Voici à quoi ressemble notre application :



Sur la partie de gauche, on peut importer/exporter notre image et on affiche en grand l'image actuelle (ici, on vient d'importer une image). Si on applique du bruit ou du débruitage dessus, on aura l'image bruitée ou débruitée à la place de l'image actuelle.

Sur la partie de droite, on peut appliquer du bruit ou du débruitage.

Il suffit de sélectionner le mode voulu (en sélectionnant l'algorithme sur le menu déroulant) et d'ajouter des valeurs dans les encadrés. Ensuite il faut appuyer sur le bouton du dessous pour appliquer l'algorithme.

4. Objectifs de la semaine

- Terminer le poster en prenant en compte les retours de M.DIBOT
- Faire les slides de présentation
- Terminer l'application QT