



HAI918I Image, sécurité et deep learning

Projet-CR6: Création de la base d'images et test du modèle pré-entraîné

Luna BOSSU
Jean Louis DEURVEILHER

Master IMAGINE
Faculté des Sciences
Université de Montpellier

November 25, 2024

1 Introduction

Cette semaine, nous nous sommes concentrés sur l'exploration du modèle FFDNET (Fast and Flexible Denoising Convolutional Neural Network). Dans un premier temps, nous avons souhaité créer une base de données d'images conséquente et pertinente en lien avec le sujet que nous voulions traiter : le débruitage d'images générées par ray tracing.

Pour cela, nous avons utilisé le moteur de rendu Tungsten, qui permet de générer rapidement des images par lancer de rayons et qui est efficacement paramétrable pour être scripté. Ensuite, nous avons évalué les résultats de la méthode FFDNET pré-entraînée sur cette base de données.

Enfin, nous avons effectué des recherches et tenté de maîtriser le processus d'entraînement du modèle afin de l'adapter à une partie de nos données. Cependant, nous avons rencontré certaines difficultés liées à l'environnement.

2 Création d'une base de données d'images

Dans un premier temps, l'objectif a été de créer une base de données robuste et pertinente. Pour cela, nous avons utilisé le moteur de rendu Tungsten, qui permet de générer différentes scènes au format JSON avec divers paramètres, de manière efficace.

2.1 Moteur de rendu Tungsten

Le moteur de rendu Tungsten est un moteur de rendu physique open-source spécialisé dans le ray tracing (lancer de rayons). Il est conçu pour produire des images de haute qualité en simulant de manière réaliste les interactions de la lumière avec les objets et les matériaux. Tungsten se distingue par sa rapidité, sa flexibilité et son efficacité, grâce à une architecture optimisée pour tirer parti des processeurs modernes.

Il prend en charge une variété de fonctionnalités avancées, telles que les matériaux complexes, les éclairages globaux, les réflexions et les réfractions précises, ainsi que le rendu volumétrique. De plus, Tungsten permet un contrôle précis des scènes grâce à des fichiers de configuration au format JSON, ce qui facilite l'automatisation et l'intégration dans des flux de travail programmatisques.

2.2 Scripting et création de la base d'images

Grâce au moteur tungsten nous avons pu effectuer divers rendu avec différents samples. Nous avons sélectionné 17 scènes 3D différentes que nous avons rendu par raytracing avec des paliers de samples par pixels allant de 64 à 1024 avec un pas de 64.

Cela nous donne une base d'images de 272 images de 1024x1024 rendues par raytracing, en plus d'une image très haute qualité qui nous servira de référence pour le modèle et les calculs de performances.

Voici quelques images obtenues:

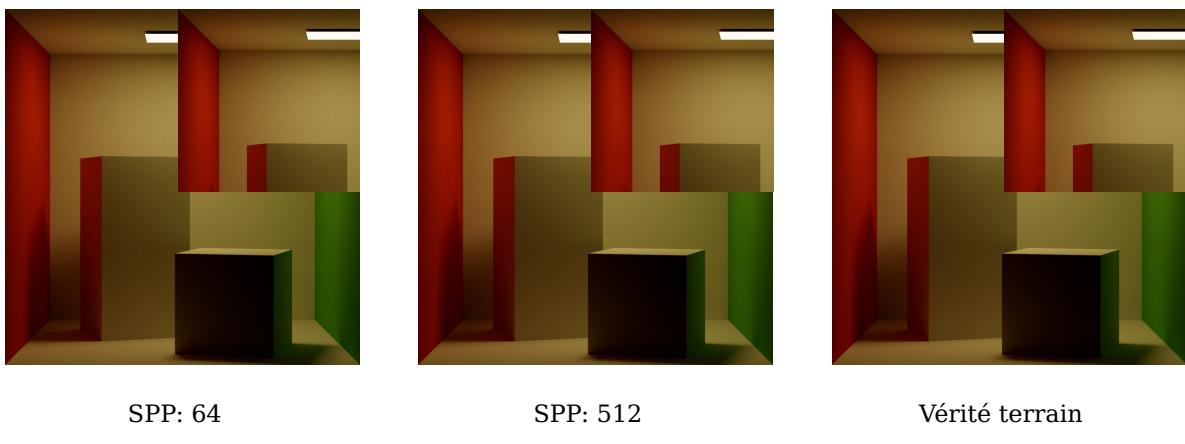
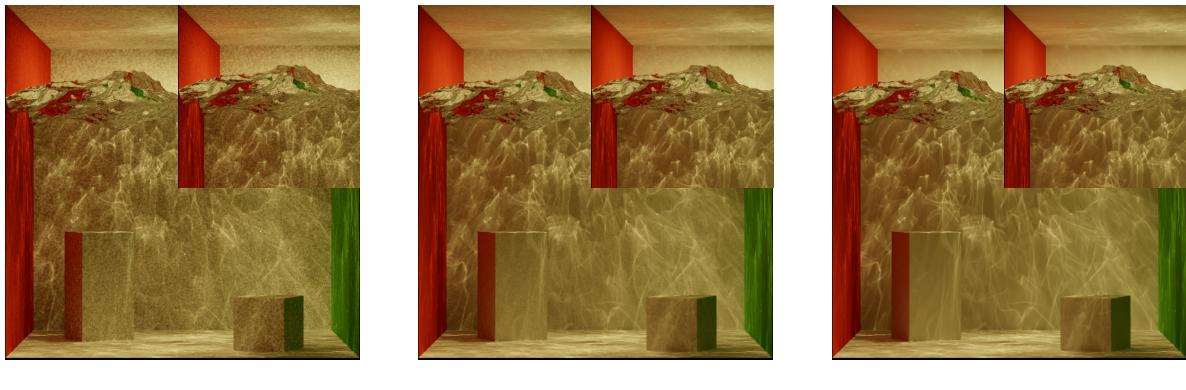


Figure 1: Cornell box rendue avec le moteur Tungsten avec des SPP respectifs de 64, 512 et l'image de vérité terrain

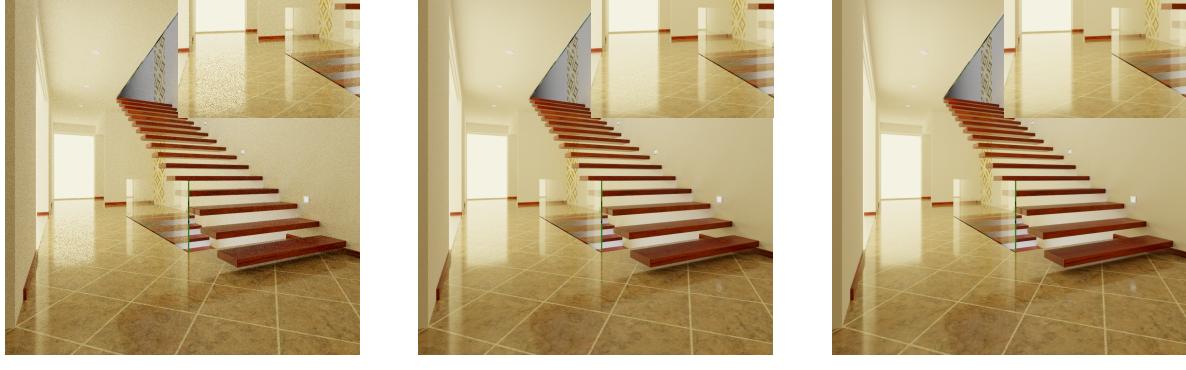


SPP: 64

SPP: 512

Vérité terrain

Figure 2: Water caustic rendue avec le moteur Tungsten avec des SPP respectifs de 64, 512 et l'image de vérité terrain



SPP: 64

SPP: 512

Vérité terrain

Figure 3: Staircase2 rendue avec le moteur Tungsten avec des SPP respectifs de 64, 512 et l'image de vérité terrain



SPP: 64

SPP: 512

Vérité terrain

Figure 4: Bathroom2 rendue avec le moteur Tungsten avec des SPP respectifs de 64, 512 et l'image de vérité terrain

Afin d'automatiser ces rendus, nous avons créer de nombreux scripts permettant de paramétrier efficacement le moteur pour les différentes scènes car en effet, le temps de rendu d'autant d'images de cette dimensions prends énormément de temps.

3 Utilisation d'un modèle FFDNet pré-entraîné

Bien que notre bût ultime soit d'entraîner notre propre modèle spécialisé pour notre type de données, nous avons voulu d'abord tester le modèle. Pour ce faire, nous avons utiliser un modèle pré-entraîné de FFDNet. Nous l'avons ensuite intégré à notre outil ligne de commande.

3.1 FFDNet

FFDNet est un modèle de réseau de neurone convolutif qui a pour but d'effectuer le débruitage d'images.

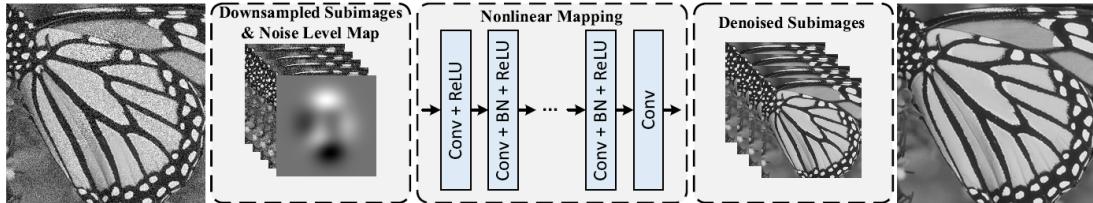


Figure 5: Architecture du modèle FFDNet

L'implémentation du modèle que l'on a utiliser viens avec plusieurs scripts d'utilités tel qu'un script pour traiter une image spécifique, un script pour préparer les données d'entraînement et de validation et un script pour réaliser l'entraînement.

L'implémentation viens aussi avec un modèle pré-entraîné, ce modèle est assez générique mais semble assez bien marcher.

3.2 Résultats

Après intégration du modèle dans notre outil ligne de commande, nous avons les résultats suivants



Figure 6: Image bruitée (gauche) et image traitée par le modèle (droite)

Le débruitage semble bien marcher, il a cependant 2 grosses limitations. Premièrement, le modèle a tendance à "lisser" les zones de l'image qui sont texturées, même si il ne s'agit pas de bruit. On perd donc des détails.



Figure 7: Image originale (gauche) et image traitée par le modèle (droite)

Deuxièmement, le filtre a du mal sur les images fortement bruitées et à tendance à créer des artefacts.

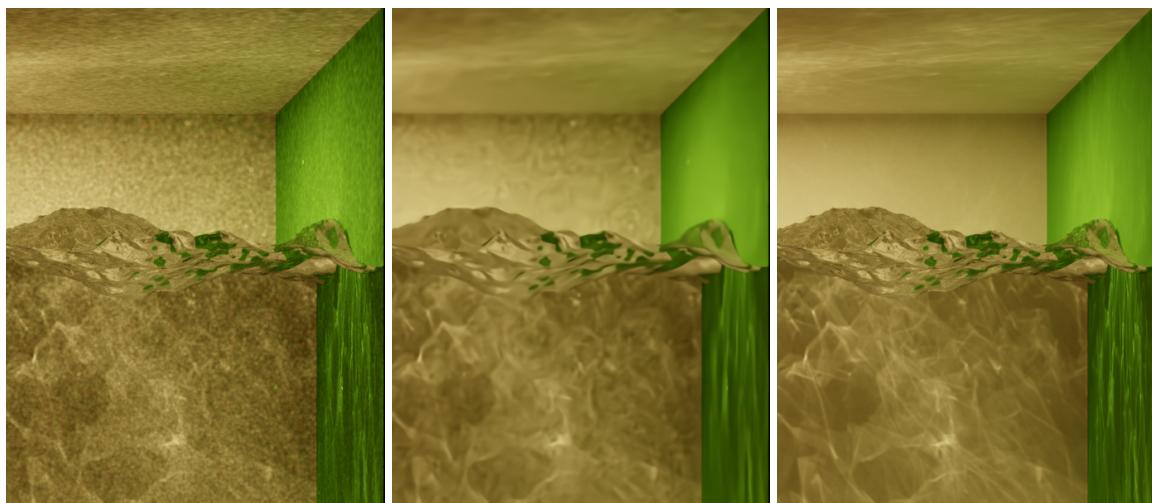


Figure 8: Image bruitée (gauche), image traitée par le modèle (milieu) et image originale (gauche)

3.3 Nouvelle Méthode d'Évaluation sans Référence : NIMA

NIMA est une mesure d'évaluation de la qualité "esthétique" d'une image générée par un réseau de neurones. Contrairement au PSNR et au SSIM, NIMA ne compare pas une image à une référence pour obtenir un score de différence. NIMA attribue un score de 1 à 10 représentant la valeur esthétique d'une image grâce à un réseau de neurones.

Notre implémentation actuelle utilise un réseau pré-entraîné que nous utilisons pour calculer la NIMA de l'image filtrée et de l'image originale.

Malheureusement, il semblerait qu'il y ait un bug dans notre approche, car la valeur de la NIMA semble fluctuer lorsqu'on la calcule plusieurs fois sur la même image. Par exemple, en calculant la valeur NIMA pour une image I une première fois, nous obtenons 4.74. En calculant la même valeur une deuxième fois pour la même image I , nous obtenons 5.68. Cela est étrange, car les données d'entrée sont identiques, et le réseau devrait fournir la même valeur. Nous continuerons à chercher une solution dans les jours à venir.

4 Planning Prévisionnel

Dans les jours à venir, nous aimerais commencer à travailler sur l'approche réseau de neurones. Nous avons déjà identifié un modèle appelé FFDNet, qui semble bien adapté à nos besoins. Une version pré-entraînée de ce modèle est disponible en ligne. Notre objectif initial sera d'utiliser cette version pour une première comparaison avec notre propre version, que nous entraînerons sur nos données.

De plus, nous avons prévu de commencer à créer notre base d'images. Pour cela, nous avons repéré un lanceur de rayon paramétrable open source appelé "Tungsten". Ce lanceur de rayon nous permettra de générer des images avec différents niveaux de bruit en faisant varier le nombre d'échantillons par pixel.