

DIASI/SIR/24-049/Rév. 0

27/08/2024

Etudes des représentations neurales pour des objets articulés

Rapport de Stage de fin d'études

DE VORA Ariel

TITRE : RAPPORT DE STAGE DE FIN D'ÉTUDES	DIASI/SIR/24-049
AUTEUR : DE VORA ARIEL	
UNITÉ : DIASI/SIR/LSI	Page 2
PROJET : ETUDES DES REPRÉSENTATIONS NEURALES POUR DES OBJETS ARTICULÉS	

NOMBRE DE PAGES : 53**RÉSUMÉ :**

Ce rapport de stage se concentre sur l'exploration et l'extension des techniques de rendu volumique, en particulier le 3D Gaussian Splatting (3DGS), une méthode récente dérivée des Neural Radiance Fields (NeRF). Introduite en 2019 et popularisée récemment, cette technique permet de reconstruire des scènes 3D photoréalistes à partir de vidéos ou de séries d'images, ouvrant ainsi des perspectives prometteuses pour des applications dans l'industrie 4.0.

Le contexte de ce travail s'inscrit dans la nécessité croissante de créer des jumeaux numériques complexes de manière rapide et efficace, sans recourir à des méthodes de modélisation manuelles fastidieuses. À travers l'utilisation de captations de la réalité, combinées aux avancées en représentations 3D telles que le 3D Gaussian Splatting, nous visons à automatiser la modélisation de systèmes industriels complexes, en particulier pour des environnements où un rendu photoréaliste est crucial, comme des ateliers techniques ou des halls d'usine.

Le rapport débute par une revue des technologies existantes, en examinant les approches actuelles, les données d'entraînement disponibles, et les résultats obtenus dans diverses applications. Ces éléments établissent les fondations théoriques et pratiques pour les développements qui suivent.

La deuxième partie du rapport est dédiée à l'extension du 3D Gaussian Splatting vers des applications dynamiques, en intégrant la dimension angulaire pour modéliser le mouvement. Trois approches distinctes sont proposées, chacune visant à améliorer la gestion des scènes en mouvement tout en maintenant une haute précision de rendu. Ce travail apporte ainsi des contributions significatives à la modélisation dynamique, en posant les bases pour la création de jumeaux numériques interactifs et photoréalistes, adaptés aux besoins de l'industrie 4.0. Les résultats obtenus pourront servir comme points de départ pour de futurs développements dans la visualisation et l'interaction en réalité virtuelle avec des objets complexes.

MOTS-CLEFS : Champ de radiance Neuraux (NeRF), algorithmie, géométrie 3D, 3D Gaussian Splatting, rendu volumique

1		<i>Validation électronique</i>	<i>Validation électronique</i>	<i>Validation électronique</i>	<i>Validation électronique</i>	
0	27/08/2024	DE VORA Ariel	Chef de labo	Chef de projet	Chef de service	
Rév.	Date	Rédacteur	Vérificateur	Émetteur	Approbateur	Pages modifiées

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0Page 4

LISTE DE DIFFUSION

DESTINATAIRE	SOCIÉTÉ	Adresse Email



CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 5

TABLE DES MATIÈRES

1 Contexte	7
1.1 Présentation du CEA LIST	7
1.2 Présentation du Projet	9
2 Travaux antérieurs / État de l'art	9
2.1 Rendu Volumique Différentiable	9
2.2 Neural Radiance Fields (NeRF)	11
2.3 3D Gaussian Splatting (3DGS)	12
2.3.1 Présentation des Datasets attendus / d'entraînement	13
2.3.2 Explication Mathématique du 3DGS	14
2.3.3 Optimisation des gaussiennes	16
2.3.4 Analyse des Résultats	18
3 Génération des Dataset	19
3.1 Génération du Dataset Réel	19
3.1.1 Prise de vue de l'Universal Robot (UR)	19
3.1.2 Acquisition des poses des caméras grâce à ParticleSfM	19
3.2 Génération du Dataset Synthétique	23
3.2.1 Acquisition directe des poses avec Unity	23
3.2.2 Acquisition des poses avec COLMAP	24
4 4D Gaussian Splatting (4DGS)	25
4.1 Intégration du temps dans les gaussiennes	25
4.2 Extension des Harmoniques Sphériques	26
4.3 Optimisations	26
4.3.1 Représentation des Rotations en 4D	27
4.3.2 Spécification de la fonction de coût pour le cas 4D	27
4.3.3 Paramétrisation du mouvement dans la scène	27
4.4 Analyse des Résultats	29
5 3D Gaussian Splatting Interactif	30
5.1 Génération du Dataset d'entraînement	31
5.2 Suivi des Groupes	32
5.3 Première Méthode Directe : Modifications des Résultats Issus du 3DGS	34
5.3.1 Translation des gaussiennes	34
5.3.2 Rotation des gaussiennes	36
5.3.3 Gestion des couleurs / Harmoniques Sphériques	37
5.4 2nde Méthode : Boite Noire (MLP)	38
5.4.1 Présentation du MLP	39
5.5 3ème Méthode : Utilisation de points d'ancrage	40
5.5.1 Présentation de Scaffold-GS	40
5.5.2 Adaptation des Translations	43
5.5.3 Augmentation de la dimension d'entrée	43

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 6

5.5.4

Ajout d'une Loss VGG

45

6

Annexes

45

6.1

Automatisation des Procédés

45

6.2

Définitions

47

6.2.1

Alpha Blending

47

6.2.2

Recalage d'Images

48

6.3

Méthodes Mathématiques

49

6.3.1

RAFT

49

6.3.2

OANet

49

6.4

Images

51

7

Bibliographie

52

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 7

1 Contexte

1.1 Présentation du CEA LIST

Le stage que j'ai effectué s'est déroulé au sein du Commissariat à l'énergie atomique et aux énergies alternatives (CEA), un acteur majeur de la recherche scientifique, technique et industrielle en France. Créé en 1945 par décret du Général De Gaulle, le CEA se consacre à des domaines stratégiques tels que la défense et la sécurité, les énergies bas carbone (nucléaire et renouvelables), la recherche fondamentale, et l'innovation technologique pour l'industrie. Avec plus de 21 000 collaborateurs répartis sur neuf centres en France (figure 1), cet organisme public se distingue par sa capacité à transférer ses innovations technologiques vers le secteur industriel.

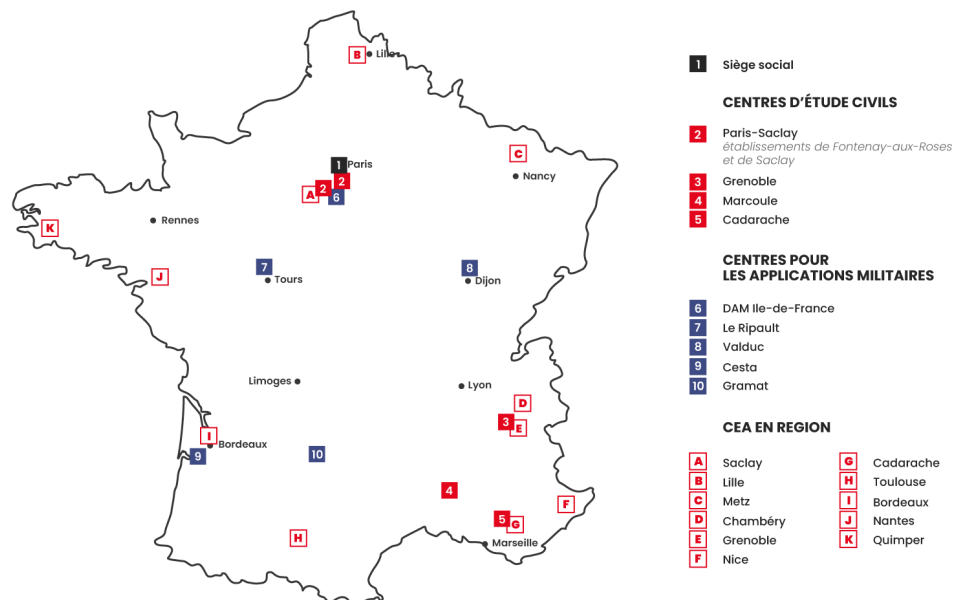


FIGURE 1 – Carte des implantations du CEA

Mon stage s'est déroulé plus spécifiquement au Laboratoire d'Intégration de Systèmes et de Technologies (LIST), un institut de la Direction de la Recherche Technologique (DRT) du CEA, situé sur le Plateau de Saclay en Île-de-France. Le LIST se concentre sur les systèmes numériques intelligents, et ses missions incluent le développement technologique de pointe pour des partenaires industriels, nationaux et internationaux. Chaque année, le LIST collabore avec plus de 750 partenaires, réalisant plus de 200 projets de recherche appliquée dans des domaines tels que la fabrication avancée, les systèmes embarqués, et les radiations ionisantes pour la santé.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 8

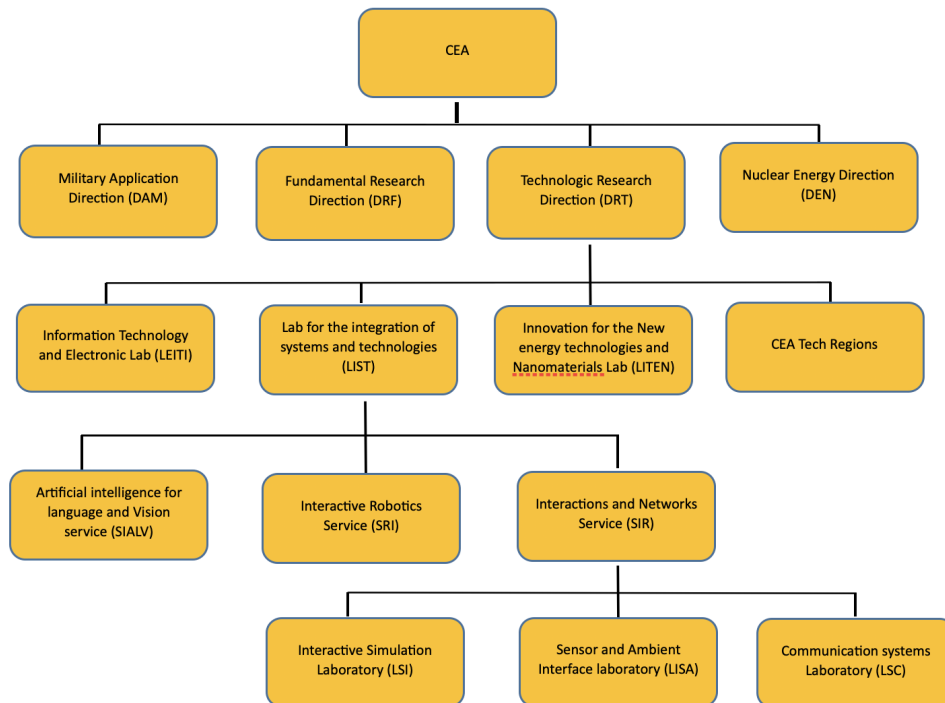


FIGURE 2 – Structure interne du CEA

Au sein du LIST, j'ai intégré le Laboratoire de Simulation Interactive (LSI), une équipe spécialisée dans la réalité mixte et la réalité virtuelle. Le LSI se distingue par sa plateforme XDE Physics, un outil de simulation multi-physique interactif en développement depuis plus d'une décennie, qui permet de manipuler et d'interagir avec des systèmes complexes, qu'ils soient rigides, articulés ou déformables. Cette plateforme est utilisée pour des études d'accessibilité, de validation ergonomique, de formation, et de simulation robotique/cobotique.

Le LSI divise ses projets en trois catégories : industriels, collaboratifs, et internes. Cette structure permet une grande flexibilité dans la gestion des projets, bien que le laboratoire ne suive pas une méthodologie agile stricte. L'organisation horizontale du LSI, dirigé par Martin Courchesne, favorise les échanges et la diffusion des connaissances, créant un cadre d'apprentissage stimulant. Le laboratoire est composé d'environ 17 chercheurs permanents, ainsi que de contractuels, de doctorants et de stagiaires, permettant une intégration rapide et une participation active aux projets innovants en cours de développement.

Travailler dans ce cadre m'a offert l'opportunité de m'impliquer dans des recherches de pointe tout en développant des compétences essentielles, avec un impact direct sur les technologies interactives appliquées au monde industriel.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 9

1.2 Présentation du Projet

Au cours de ce stage, j'ai eu l'opportunité de contribuer à des projets innovants visant à simplifier et accélérer la création de jumeaux numériques complexes, potentiellement interactifs, pour répondre aux défis de l'industrie 4.0. Dans ce contexte, il est essentiel de pouvoir reconstruire numériquement des objets ou des ensembles d'objets complexes de manière rapide et réaliste. Plutôt que de recourir à des modélisations manuelles via des logiciels de CAO, nous avons opté pour des méthodes de captation de la réalité, qui se déclinent en deux principales approches au LIST : l'utilisation de dispositifs LIDAR ou l'exploitation de vidéos et de séries d'images.

Dans le cadre de ce stage, nous avons choisi d'explorer la seconde approche, en particulier la méthode du 3D Gaussian Splatting, une technique récemment présentée à la conférence Siggraph 2023 par l'INRIA. À l'instar des Neural Radiance Fields (NeRF) et des méthodes traditionnelles de photogrammétrie, le 3D Gaussian Splatting permet de reconstruire une scène 3D photoréaliste à partir d'un ensemble d'images ou de vidéos. Cette méthode révolutionne actuellement le domaine de la réalité virtuelle grâce à sa capacité à produire des rendus interactifs compatibles avec les fréquences requises pour les casques VR, en offrant des temps d'entraînement de plus en plus rapides et une qualité de rendu toujours plus élevée.

Le 3D Gaussian Splatting se distingue par son approche d'optimisation basée sur la vision à partir d'images 2D recalées (voir 6.2.2), générant une représentation géométrique non structurée sous la forme d'un nuage de gaussiennes. Ce travail s'inscrit dans la continuité des avancées réalisées avec les NeRFs, qui ont déjà démontré leur potentiel en reconstruisant l'apparence, la géométrie et les liaisons d'outils articulés ou rigides à partir de simples images RGB et des poses de caméra. Notre objectif est d'étendre et d'améliorer cette technologie, en examinant notamment la segmentation, la position des axes de liaisons, et la gestion des liaisons de rotation, tout en développant de nouvelles applications adaptées aux besoins industriels.

Après avoir exploré les bases et les avancées du rendu volumique différentiable et des méthodes associées, la section 2 sera consacrée à une revue des techniques existantes pour les scènes statiques en 3D, en particulier le 3D Gaussian Splatting (3DGS) et son contexte scientifique. Ensuite, dans la section 3, nous présenterons les datasets statiques et dynamiques que nous avons utilisés, qu'ils soient réels ou synthétiques, en soulignant leur utilité pour les différentes applications étudiées. La section 4 proposera une étude comparative approfondie des méthodes de rendu 4D Gaussian Splatting temporelles, ce qui nous conduira, dans la section 5, à présenter les développements réalisés au cours de ce stage pour la visualisation interactive et photoréaliste de modèles dynamiques articulés.

2 Travaux antérieurs / État de l'art

2.1 Rendu Volumique Différentiable

Le rendu volumique différentiable repose sur les principes du rendu volumique, mais avec une approche qui permet l'optimisation de la représentation d'une scène via des gradients différentiables. Cette méthode est cruciale dans des applications comme les champs de radiance neuraux (NeRF [14]), où l'objectif est de générer des vues photoréalistes d'une scène 3D à partir d'un ensemble d'images 2D.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 10

Dans ce contexte, chaque rayon traversant la scène est associé à une couleur $C(r)$ obtenue en intégrant la contribution des différentes densités volumiques σ le long du rayon :

$$C(r) = \int_{t_n}^{t_f} T(t) \sigma(r(t)) c(r(t), \mathbf{d}) dt \quad (1)$$

La transmittance accumulée $T(t)$ représente la probabilité que le rayon traverse la scène sans être interrompu par des particules, et elle est définie par :

$$T(t) = \exp \left(- \int_{t_n}^t \sigma(r(s)) ds \right) \quad (2)$$

La couleur peut être approximée en utilisant une somme pondérée des contributions de chaque point d'échantillonnage i le long du rayon :

$$\hat{C}(r) = \sum_{i=1}^N \alpha_i T_i c_i, \quad \text{ou} \quad T_i = \exp \left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right) \quad (3)$$

De plus, la profondeur du rayon $d(r)$ peut être calculée à partir de l'intégrale suivante :

$$d(r) = \int_{t_1}^{t_2} T(t) \cdot \sigma(r(t)) \cdot t dt \quad (4)$$

ou approximée par :

$$\hat{d}(r) = \sum_{i=1}^N \alpha_i T_i t_i \quad (5)$$

Ces formules permettent au rendu volumique différentiable de capturer avec précision les détails fins de la scène, en tenant compte des variations de densité et de couleur à chaque point dans l'espace. En s'appuyant sur ces fondements du rendu volumique, les champs de radiance neuraux (NeRF) ont considérablement élargi les possibilités de représentation tridimensionnelle en utilisant des réseaux de neurones pour synthétiser des vues photoréalistes à partir d'un ensemble d'images.

L'aspect différentiable du rendu volumique joue un rôle crucial dans l'optimisation des scènes 3D en permettant d'ajuster précisément les paramètres en fonction des erreurs observées dans les images de référence. Lorsque les poses de caméras sont connues, et qu'une erreur est identifiée pixel par pixel dans le plan image, il devient possible, via rétropropagation, de calculer un gradient pour chaque paramètre du rendu volumique le long des rayons. Cette différentiabilité permet d'optimiser les densités et couleurs volumiques pour qu'elles correspondent exactement aux images de référence, ouvrant ainsi la voie à des méthodes comme le NeRF (section 2.2), où l'apprentissage direct des atténuations et des couleurs le long des rayons est effectué par un réseau de type MLP. De manière similaire, le 3D Gaussian Splatting (3DGS) exploite cette différentiabilité pour optimiser les paramètres des primitives spécifiques, les gaussiennes 3D, qui constituent une représentation explicite et efficace de la scène. Ces deux approches illustrent l'efficacité du rendu volumique différentiable dans la génération de représentations tridimensionnelles photoréalistes et l'optimisation de scènes complexes.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 11

2.2 Neural Radiance Fields (NeRF)

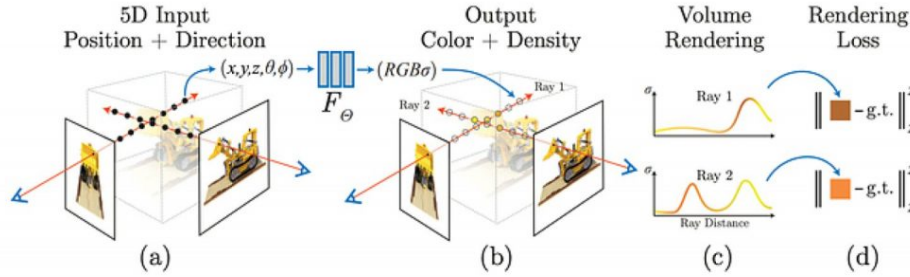


FIGURE 3 – Aperçu de la représentation de scène par un NeRF et de la procédure de rendu différentiable.

Les champs de radiance neuraux (NeRF [14]) sont une avancée majeure dans la représentation et le rendu de scènes 3D avec une fidélité remarquable. Ces modèles utilisent un réseau de neurones, généralement un perceptron multicouche (MLP) dénoté par : $\mathbf{F}_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$ et l'optimisent afin que l'évaluation pour un jeu arbitraire (position, direction) renvoie une couleur directionnelle ainsi qu'une densité volumique la plus proche de la ground-truth possible.

Contrairement aux méthodes traditionnelles qui se basent uniquement sur la géométrie 3D, les NeRF combinent cette géométrie avec une représentation neuronale qui permet de capturer des détails complexes, tels que les effets de lumière et les variations de couleur.

Les NeRF représentent une scène continue sous forme d'une fonction vectorielle 5D, où l'entrée est une position 3D combinée avec une direction de vue 2D (définie par deux angles, θ et ϕ). La sortie de cette fonction est constituée d'une couleur (r, g, b) et d'une densité volumique σ , ces dernières étant cruciales pour synthétiser des images photoréalistes à partir de points de vue arbitraires. Le processus de rendu se base sur les principes du rendu volumique classique, où la densité volumique $\sigma(\mathbf{x})$ est interprétée comme la probabilité qu'un rayon traversant la scène soit interrompu par une particule infinitésimale à une position donnée $\mathbf{x}$.

Pour générer une image, on estime l'intégrale continue 1 en traçant des rayons depuis la caméra à travers chaque pixel de l'image souhaitée.

Ceci donne la possibilité de réécrire l'équation 3 comme :

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) c_i, \quad \text{ou} \quad T_i = \exp \left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right) \quad (6)$$

Deux MLP sont optimisés en simultané, un fin et un grossier. Réécrire l'équation 3 comme :

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i \quad \text{ou} \quad w_i = T_i (1 - \exp(-\sigma_i \delta_i)) \quad (7)$$

permet d'évaluer les poids $\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$, et de tirer N_f points de cette distribution afin de les ajouter aux

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 12

N_c déjà existants. Ceci augmente le nombre d'intégrales évaluées le long des trajectoires pertinentes, et aussi d'éviter des calculs redondants dans les zones présentant peu de détails (ou non-visibles).

Finalement, l'entraînement est supervisé par la fonction de coût suivante, somme des erreurs quadratiques moyennes entre la ground-truth ($C(r)$) et respectivement les deux rendus (fin ($\hat{C}_f(r)$) et grossier ($\hat{C}_c(r)$)) :

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right] \quad (8)$$

Où $\mathcal{R}$ est l'ensemble des rayons d'un échantillon. Ces techniques permettent ainsi aux NeRF de produire des rendus visuellement cohérents, même lorsqu'ils sont observés sous différents angles, offrant une solution puissante pour la synthèse d'images de scènes 3D complexes.

Afin de réduire les temps d'entraînement et d'augmenter la qualité de rendu, deux stratégies principales ont été mises en place. Plutôt que de se limiter à un ensemble discret de positions dans l'espace, les NeRF utilisent un échantillonnage stratifié qui permet d'évaluer le MLP à des positions continues, améliorant ainsi la précision du rendu. Ceci est effectué en subdivisant $[t_n, t_f]$ en N sous-intervalles, puis en tirant uniformément des échantillons depuis chacun de ces intervalles.

$$t_i \sim \mathcal{U} \left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n) \right] \quad (9)$$

Une seconde approche permettant de surmonter cette limitation consiste à effectuer une transformation des coordonnées d'entrée vers un espace de dimension supérieure, facilitant l'apprentissage des fonctions à haute fréquence nécessaires pour un rendu précis des détails fins. Ceci a été introduit par Rahaman et al. et intégré au NeRF par le fait de mapper les entrées dans un espace de dimension supérieure $\mathbb{R} \rightarrow \mathbb{R}^{2L}$ en utilisant des fonctions à hautes fréquences avant de les passer au réseau. γ est ainsi introduit comme fonction d'encodage, appliquées aux 5 entrées avant de les passer au MLP en reformulant le réseau de neurones comme la composition des deux fonctions $\mathbf{F}' = \mathbf{F} \circ \gamma$.

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)) \quad (10)$$

où $\mathbf{p}$ correspond à l'une des entrées x, y, z, θ, ϕ .

Bien que NeRF permette de produire des rendus d'une grande fidélité, cette méthode souffre de limitations importantes en termes de temps d'apprentissage et de rendu. En effet, elle nécessite un grand nombre d'inférences du MLP, avec plusieurs dizaines, voire centaines d'échantillons par rayon, ce qui peut conduire à des temps de rendu de l'ordre de 48 heures.

2.3 3D Gaussian Splatting (3DGS)

Le Neural Radiance Fields (NeRF) a révolutionné la représentation des scènes 3D en utilisant une fonction volumétrique continue pour encoder la couleur et la densité de chaque point de l'espace. Cependant, malgré sa capacité à produire des rendus réalistes, le NeRF reste gourmand en ressources computationnelles. C'est dans ce contexte que le 3D Gaussian Splatting (3DGS [10]) se positionne comme une méthode alternative

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 13

plus efficace, tout en maintenant une qualité de rendu élevée.

Le 3DGS commence avec un ensemble d'images d'une scène statique, accompagné des caméras calibrées via Structure from Motion (SfM), ce qui génère également un nuage de points épars. À partir de ces points, des gaussiennes 3D sont créées, définies par une position (moyenne), une matrice de covariance et une opacité α , offrant ainsi une flexibilité considérable pour l'optimisation. Cette représentation compacte permet de modéliser efficacement des structures fines grâce à l'utilisation de splats volumiques hautement anisotropes. Le composant directionnel de l'apparence (couleur) du champ de radiance est représenté via des harmoniques sphériques (SH), suivant les pratiques standards.

L'algorithme 3D Gaussian Splatting suit un processus d'optimisation itérative des paramètres des gaussiennes (illustré dans la Figure 4), tels que la position $x \in \mathbb{R}^3$, l'opacité $\alpha \in \mathbb{R}$, leur orientation $q \in \mathbb{R}^4$ (un quaternion) et échelle $s \in \mathbb{R}^3$ selon chaque axe, et la couleur c (RGB) calculées grâce aux coefficients des harmoniques sphériques (SH). Partant des points générés par Structure from Motion (SfM), les gaussiennes 3D sont initialisées et projetées dans l'espace image en fonction des paramètres des caméras. Un contrôle adaptatif de la densité est ensuite appliqué pour affiner la représentation des gaussiennes, en s'assurant que les zones nécessitant plus de détails reçoivent une plus grande densité de splats. L'efficacité du rendu repose sur un rasteriseur différentiable basé sur des tuiles, qui permet un alpha-blending précis des splats anisotropes, tout en maintenant l'ordre de visibilité grâce à un tri rapide. Le flux de gradients, représenté par les flèches bleues, permet l'optimisation continue des paramètres des gaussiennes lors du passage arrière, sans limite sur le nombre de gaussiennes pouvant recevoir des gradients, garantissant ainsi une convergence rapide vers une représentation fidèle de la scène.

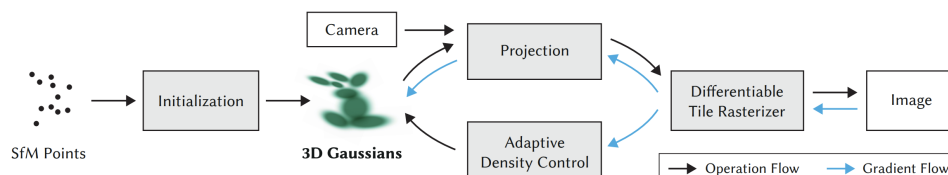


FIGURE 4 – Résumé / Graphique de la méthode 3DGS

Dans les sous-sections suivantes, nous présenterons les datasets utilisés pour l'entraînement et la validation du modèle, puis nous détaillerons les aspects mathématiques du 3DGS. Nous aborderons ensuite l'optimisation des gaussiennes, en discutant de l'ajustement des positions et des hyperparamètres.

2.3.1 Présentation des Datasets attendus / d'entraînement

Une structure de données spécifique est attendue : le format de sortie de COLMAP.

```
<location>
|---input
|   |---<image 0>
|   |---<image 1>
|   |---...
```

Commissariat à l'énergie atomique et aux énergies alternatives
Institut List | Département Intelligence Ambiante et Systèmes Interactifs
Service Interactions et Réseaux | Laboratoire de Simulation Interactive
CEA Saclay - Nanno-Innov - Bâtiment 861 - Point Courrier 173
91191 Gif-sur-Yvette Cedex
T. +33 (0)1 69 08 00 99
www-list.cea.fr

Établissement public à caractère industriel et commercial | RCS PARIS B 775 685 019



CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 14

```
|---distorted
|---database.db
|---sparse
|---0
|---...
```

L'obtention de ces données, les logiciels ainsi que les méthodes mathématiques utilisées sont détaillés dans la section 3.

2.3.2 Explication Mathématique du 3DGS

Le 3D Gaussian Splatting (3DGS) repose sur la représentation d'une scène 3D à l'aide de gaussiennes tridimensionnelles. Chaque gaussienne est paramétrée par une moyenne $\mu \in \mathbb{R}^3$, une matrice de covariance $\Sigma \in \mathbb{R}^{3 \times 3}$, une couleur $c \in \mathbb{R}^3$, et une opacité $\alpha \in \mathbb{R}$. L'objectif est de projeter ces gaussiennes dans l'espace 2D d'une caméra pour obtenir une vue de la scène. Ce processus implique plusieurs étapes de transformation, que nous détaillons ci-dessous.

Projection des gaussiennes Pour projeter une gaussienne 3D dans l'espace image, il est nécessaire de transformer ses paramètres à travers la chaîne de transformations définie par la caméra. La caméra est décrite par sa matrice d'extrinsèques T_{cw} , qui transforme les points de l'espace des coordonnées monde vers l'espace des coordonnées caméra, et sa matrice d'intrinsèques P , qui définit la projection de l'espace caméra vers l'espace image.

La matrice d'extrinsèques est définie comme suit :

$$T_{cw} = \begin{bmatrix} R_{cw} & t_{cw} \\ 0 & 1 \end{bmatrix} \in \mathbf{SE}(3), \quad (11)$$

où R_{cw} est la matrice de rotation et t_{cw} est le vecteur de translation. La matrice de projection P est définie par :

$$P = \begin{bmatrix} \frac{2f_x}{w} & 0 & 0 & 0 \\ 0 & \frac{2f_y}{h} & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad (12)$$

où f_x et f_y sont les longueurs focales de la caméra, (w, h) sont la largeur et la hauteur de l'image, et (n, f) sont les plans de clipping proches et lointains.

La projection d'une gaussienne 3D est réalisée en transformant d'abord sa moyenne μ en coordonnées caméra $t \in \mathbb{R}^4$, puis en coordonnées normalisées $t' \in \mathbb{R}^4$, et finalement en coordonnées pixel $\mu' \in \mathbb{R}^2$:

$$t = T_{cw} \begin{bmatrix} \mu \\ 1 \end{bmatrix}, \quad t' = \mathbf{P}t, \quad \mu' = \begin{bmatrix} \frac{1}{2} \left(\frac{wt'_x}{t'_w} + 1 \right) + c_x \\ \frac{1}{2} \left(\frac{ht'_y}{t'_w} + 1 \right) + c_y \end{bmatrix}, \quad (13)$$

où (c_x, c_y) sont les coordonnées du point principal de l'image.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 15

Transformation de la Covariance La projection perspective d'une gaussienne 3D ne résulte pas en une gaussienne 2D. Pour obtenir la matrice de covariance projetée Σ' , une approximation par une expansion de Taylor du premier ordre est effectuée autour de la moyenne t dans le cadre de la caméra. Cette transformation affine est donnée par la matrice $J \in \mathbb{R}^{2 \times 3}$:

$$J = \begin{bmatrix} \frac{f_x}{t_z} & 0 & -\frac{f_x t_x}{t_z^2} \\ 0 & \frac{f_y}{t_z} & -\frac{f_y t_y}{t_z^2} \end{bmatrix}, \quad (14)$$

et la matrice de covariance projetée $\Sigma' \in \mathbb{R}^{2 \times 2}$ est alors :

$$\Sigma' = J R_{cw} \Sigma R_{cw}^\top J^\top. \quad (15)$$

Paramétrisation de la Covariance 3D La covariance 3D Σ est paramétrée à partir d'une amplitude $s \in \mathbb{R}^3$ et d'un quaternion de rotation $q \in \mathbb{R}^4$. Le quaternion $q = (x, y, z, w)$ est converti en une matrice de rotation R :

$$R = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{bmatrix}, \quad (16)$$

La covariance 3D Σ est alors définie par :

$$\Sigma = R S S^\top R^\top, \quad (17)$$

où S est une matrice diagonale définissant l'échelle des axes principaux de la gaussienne.

Ces transformations permettent de représenter efficacement la scène en projetant et en adaptant dynamiquement les gaussiennes 3D pour correspondre aux vues de la caméra, tout en préservant la fidélité géométrique et visuelle de la scène.

Finalement, la couleur est calculée de manière similaire à celle utilisée dans les champs de radiance neuraux, et est écrite sous une forme équivalente à l'équation 3 :

$$C = \mathcal{I}(u, v) = \sum_{i \in \mathcal{N}} \alpha_i T_i c_i, \quad \text{ou} \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \quad \text{ou} \quad \alpha_i = (1 - \exp(-\sigma_i \delta_i)) \quad (18)$$

où (u, v) représente la position du pixel dont on veut calculer la couleur C , et $\mathcal{N}$ est l'ensemble de points triés qui composent le pixel.

Les harmoniques sphériques sont essentielles pour modéliser l'aspect directionnel des couleurs dans le 3D Gaussian Splatting (3DGS). Comme les séries de Fourier décomposent une fonction en sinusoïdes, les harmoniques sphériques décomposent une fonction définie sur une sphère en une somme de fonctions orthogonales, chacune correspondant à une certaine fréquence angulaire. Les fonctions harmoniques sphériques $Y_\ell^m(\theta, \varphi)$ sont définies par :

$$Y_\ell^m(\theta, \varphi) = \sqrt{\frac{(2\ell + 1)(\ell - m)!}{4\pi(\ell + m)!}} P_\ell^m(\cos \theta) e^{im\varphi} \quad (19)$$

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 16

$P_\ell^m(\cos \theta)$ étant le polynôme de Legendre associé. Cette base orthonormale permet de capturer de manière compacte les variations de couleur en fonction de l'angle de vue, en stockant les informations directionnelles sous forme de coefficients. Ces coefficients sont beaucoup plus économes en mémoire que des alternatives comme les cube maps, qui nécessiteraient de stocker une carte entière pour chaque point.

2.3.3 Optimisation des gaussiennes

Contrairement aux NeRF, les 3DGS n'impliquent pas l'utilisation et l'entraînement de MLP, mais procèdent uniquement par optimisation directe des attributs des gaussiennes. L'optimisation des Gaussiennes 3D dans le cadre du Gaussian Splatting s'effectue par des itérations successives de rendu d'images, comparées aux vues d'entraînement du dataset capturé. Cette méthode corrige la géométrie mal placée, un problème courant dû aux ambiguïtés de la projection 3D en 2D. L'optimisation se divise en deux phases distinctes : d'une part, une descente de gradient stochastique, tirant pleinement parti des frameworks GPU accélérés et des noyaux CUDA personnalisés pour certaines opérations spécifiques sur les paramètres des gaussiennes (position, opacité, échelle, coefficients SH), et d'autre part, la modification du nombre de splats par densification, fusion ou suppression. Cette dernière permet l'adaptation à la complexité géométrique de la scène au fil des itérations, augmentant progressivement le nombre de gaussiennes, ce qui entraîne un passage d'un nuage de points épars à un nuage dense.

Ajustement des Positions Le processus commence avec un ensemble de points épars issus de SfM. L'optimisation densifie progressivement les gaussiennes et contrôle leur densité par unité de volume pour représenter plus fidèlement la scène. Après un échauffement de l'optimisation, la densification se produit toutes les 100 itérations, et les gaussiennes avec une opacité α inférieure à un seuil ϵ_α sont supprimées.

Le contrôle adaptatif des gaussiennes permet de remplir les zones vides, notamment dans les régions sous-représentées géométriquement ("sous-reconstruction") ou celles où les gaussiennes couvrent de grandes zones ("sur-reconstruction"). Les gradients de position dans l'espace des vues sont utilisés pour identifier les régions nécessitant une densification.

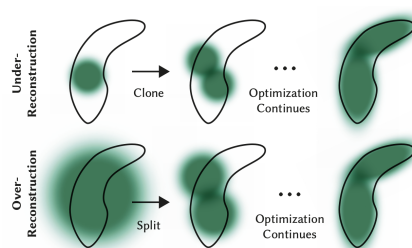


FIGURE 5 – Contrôle Adaptatif des gaussiennes

Les petites gaussiennes dans les régions sous-représentées sont clonées en créant une copie de même taille,

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 17

déplacée dans la direction du gradient de position. Les grandes gaussiennes dans les régions à forte variance sont scindées en deux, leur échelle étant divisée par un facteur $\phi = 1.6$. Pour modérer l'augmentation du nombre de gaussiennes, la valeur de α est réduite à près de zéro toutes les 3000 itérations, permettant ainsi à l'optimiseur de décider quelles gaussiennes doivent être conservées ou supprimées. Cette optimisation s'effectue à l'aide d'un rasteriseur basé sur des tuiles, des blocs de 16x16 pixels. Les gaussiennes sont pré-triées selon la profondeur et l'ID de la tuile, ce qui permet un α -blending rapide.

Le rasteriseur rapide permet également une rétropropagation efficace, sans limite sur le nombre de gaussiennes pouvant recevoir des mises à jour de gradient. Pendant le passage arrière, la séquence complète des points mélangés est récupérée pour chaque pixel, permettant ainsi un apprentissage précis des scènes avec une complexité de profondeur variable.

La descente de gradient stochastique dans le cadre du 3D Gaussian Splatting est au cœur de l'optimisation des splats, en particulier pour les paramètres de position, de covariance, d'opacité et de coefficients SH. Pour rendre cette optimisation efficace, il est essentiel de calculer les gradients avec précision et rapidité. Dans ce contexte, la matrice de covariance Σ est décomposée en une rotation R , une matrice diagonale S , et une matrice intermédiaire M , telle que $\Sigma = RSS^T R^T = MM^T$. Les dérivées partielles par rapport à l'échelle s et à la rotation q sont obtenues en utilisant la règle de la chaîne :

$$\frac{d\Sigma'}{ds} = \frac{d\Sigma'}{d\Sigma} \frac{d\Sigma}{ds}, \quad \frac{d\Sigma'}{dq} = \frac{d\Sigma'}{d\Sigma} \frac{d\Sigma}{dq}$$

où $\frac{\partial M_{i,j}}{\partial s_k} = \begin{cases} R_{i,k} & \text{si } j = k \\ 0 & \text{sinon} \end{cases}$. Pour dériver les gradients de la rotation, les quaternions sont utilisés pour représenter q , et la conversion de q en une matrice de rotation $R(q)$ s'exprime ainsi :

$$R(q) = \begin{pmatrix} \frac{1}{2} - (q_j^2 + q_k^2) & q_i q_j - q_r q_k & q_i q_k + q_r q_j \\ q_i q_j + q_r q_k & \frac{1}{2} - (q_i^2 + q_k^2) & q_j q_k - q_r q_i \\ q_i q_k - q_r q_j & q_j q_k + q_r q_i & \frac{1}{2} - (q_i^2 + q_j^2) \end{pmatrix}$$

Les gradients des composantes de q sont alors donnés par :

$$\frac{\partial M}{\partial q_r} = 2 \begin{pmatrix} 0 & -s_y q_k & s_z q_j \\ s_x q_k & 0 & -s_z q_i \\ -s_x q_j & s_y q_i & 0 \end{pmatrix}, \quad \frac{\partial M}{\partial q_i} = 2 \begin{pmatrix} 0 & s_y q_k & -s_z q_r \\ s_x q_k & 0 & -s_z q_j \\ -s_x q_j & s_y q_r & 0 \end{pmatrix}$$

Ceci est reproduit pour $\frac{\partial M}{\partial q_j}$ et $\frac{\partial M}{\partial q_k}$. Ces calculs permettent d'effectuer une rétropropagation efficace, assurant une optimisation précise des paramètres des gaussiennes en fonction de la structure de la scène et des erreurs observées dans le rendu final.

Estimation des erreurs L'optimisation des tenseurs contenant les attributs des gaussiennes est faite par le biais d'une rétropropagation de l'erreur respectivement à une fonction de coût, combinaison d'une perte L_1 et d'un terme D-SSIM :

$$L = (1 - \lambda)L_1 + \lambda L_{D-SSIM} \quad (20)$$

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 18

où SSIM représente une mesure de l'indice de similarité structurelle (Structural Similarity Index Measure) :

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (21)$$

L'évaluation de la qualité se fait grâce à deux méthodes/fonctions :

$$PSNR(I) = 10 \cdot \log_{10} \left(\frac{MAX(I)^2}{MSE(I)} \right) \quad (22)$$

$$LPIPS(x, y) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|w_l(\zeta)(x_{hw}^l - y_{hw}^l)\|_2^2 \quad (23)$$

2.3.4 Analyse des Résultats

Vous verrez dans les figures 6 et 7 deux vues d'un rendu effectué grâce au code 3DGS présenté par l'INRIA :

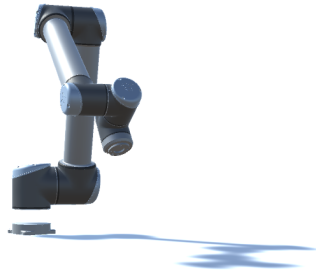


FIGURE 6 – Rendu 1 sur données synthétiques : 3DGS

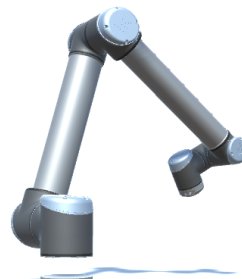


FIGURE 7 – Rendu 2 sur données synthétiques : 3DGS

Contrairement au NeRF, le 3D Gaussian Splatting (3DGS) propose une alternative plus efficace, avec des temps de rendu réduits à moins d'une heure, tout en offrant des scores SSIM et LPIPS supérieurs d'environ 10% (respectivement inférieurs) à ceux de NeRF. Cependant, cette amélioration de l'efficacité et de la qualité du rendu se fait au prix d'une augmentation significative de la taille en mémoire, avec des besoins allant de 200MB à 1GB pour 3DGS, contre moins de 10MB pour NeRF (voir tableau 8). Ces différences illustrent les compromis entre qualité de rendu, temps de calcul, et utilisation de la mémoire, rendant 3DGS particulièrement adapté pour des applications où la rapidité et la qualité du rendu sont prioritaires, ce qui est notamment crucial dans notre contexte, où l'objectif est le contrôle d'un modèle articulé.

Dataset Method/Metric	Mip-NeRF360						Tanks&Temples						Deep Blending					
	SSIM [↑]	PSNR [↑]	LPIPS [↓]	Train	FPS	Mem	SSIM [↑]	PSNR [↑]	LPIPS [↓]	Train	FPS	Mem	SSIM [↑]	PSNR [↑]	LPIPS [↓]	Train	FPS	Mem
Plenoxels	0.626	23.08	0.463	25m49s	6.79	2.1GB	0.719	21.08	0.379	25m5s	13.0	2.3GB	0.795	23.06	0.510	27m49s	11.2	2.7GB
INGP-Base	0.671	25.30	0.371	5m37s	11.7	13MB	0.723	21.72	0.330	5m26s	17.1	13MB	0.797	23.62	0.423	6m31s	3.26	13MB
INGP-Big	0.699	25.59	0.331	7m30s	9.43	48MB	0.745	21.92	0.305	6m59s	14.4	48MB	0.817	24.96	0.390	8m	2.79	48MB
M-NeRF360	0.792 [†]	27.69 [†]	0.237 [†]	48h	0.06	8.6MB	0.759	22.22	0.257	48h	0.14	8.6MB	0.901	29.40	0.245	48h	0.09	8.6MB
Ours-7K	0.770	25.60	0.279	6m25s	160	523MB	0.767	21.20	0.280	6m55s	197	270MB	0.875	27.78	0.317	4m35s	172	386MB
Ours-30K	0.815	27.21	0.214	41m33s	134	734MB	0.841	23.14	0.183	26m54s	154	411MB	0.903	29.41	0.243	36m2s	137	676MB

FIGURE 8 – Résultats qualitatifs (comparaison entre 3DGS et Mip-NeRF)

3 Génération des Dataset

Dans ce projet, les datasets utilisés présentent une spécificité notable : ils capturent un ou plusieurs objets en mouvement au cours du temps, avec des prises de vue effectuées par une ou plusieurs caméras simultanément.

3.1 Génération du Dataset Réel

Afin d'entraîner les différents MLP ou d'optimiser les tenseurs d'attributs des gaussiennes, une grande quantité d'informations est nécessaire. Dans un premier temps il faudra une série d'au moins 80 à 100 images, placées autour de l'objet, dépendant de la nature de la scène ainsi que du niveau de détail voulu.

3.1.1 Prise de vue de l'Universal Robot (UR)

3.1.2 Acquisition des poses des caméras grâce à ParticleSfM

Structure from Motion (SfM) est une technique en vision par ordinateur qui permet de reconstruire une scène 3D à partir d'une série d'images 2D prises sous différents angles. Cette méthode repose sur l'extraction des points d'intérêt communs à travers les images, ce qui permet de déterminer la position relative de la caméra ainsi que la structure 3D de la scène observée. Ces informations, les positions exactes ainsi que les caractéristiques internes et externes des caméras sont ordonnées dans deux jeux de matrices différentes : les matrices intrinsèques et extrinsèques.

Les matrices extrinsèques correspondent aux matrices dites "World to Camera", matrices de transformations permettant le passage du référentiel de la caméra au référentiel global de la scène. Elles se découpent en deux parties distinctes :

$$\begin{pmatrix} R_{3 \times 3} & T_{3 \times 1} \\ 0_{1 \times 3} & 1 \end{pmatrix} \quad (24)$$

où le vecteur $T_{3 \times 1}$ correspond à la translation, et la matrice $R_{3 \times 3}$ à la rotation, représentant respectivement la position de l'origine du système de coordonnées du monde exprimée selon le système de coordonnées centré sur la caméra, ainsi que l'orientation de la caméra.

Les matrices intrinsèques contiennent quant-à-elles les informations relatives aux caméras, telles que les focales, les formats des senseurs / capteurs d'images et les points principaux des caméras.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 20

$$\begin{pmatrix} fl_x & \gamma & c_x \\ 0 & fl_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (25)$$

f_x et f_y étant les distances focales en terme de pixels, γ le coefficient d'asymétrie et c_x et c_y les points principaux (de façon idéale le centre de l'image).

Ces données essentielles ne sont néanmoins pas capturées en même temps que les images, nous forçant à utiliser des logiciels tiers afin de les estimer (une étude sur la fiabilité de ce processus sera faite). C'est ici que l'utilisation de deux logiciels tiers a été cruciale : COLMAP (directement intégré au sein des repos de 3DGS) ainsi que ParticleSfM [25]. COLMAP ([17] et [18]) a pour rôle de générer l'ensemble des matrices dont nous avons besoin, ParticleSfM, d'un autre côté, servira à retrouver les matrices extrinsèques lors de la capture de scènes dynamiques. ParticleSfM est particulièrement utile dans ce contexte, car il permet de distinguer les parties immobiles des parties en mouvement dans une scène dynamique (voir figure 9), facilitant ainsi une reconstruction classique par SfM à partir des éléments immobiles. Ce dernier sera utilisé lors de l'exploitation de notre prise de vue de l'UR (réelle) effectuée grâce à un iPhone 15 pro, générant automatiquement les données nécessaires pour la reconstruction des matrices intrinsèques.



FIGURE 9 – Segmentation du mouvement (vert : pixels en mouvement, bleu : pixels immobiles)

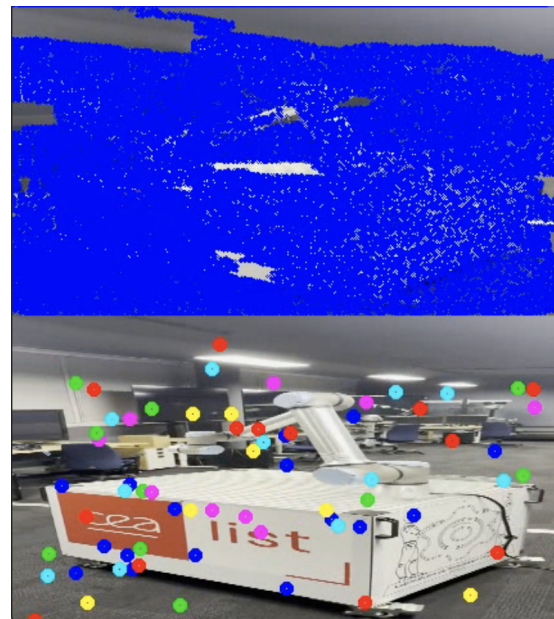


FIGURE 10 – Visualisation des points de référence utilisés

La méthode ParticleSfM commence par l'estimation d'un champ de déplacement dense qui mappe chaque pixel d'une image à ses coordonnées correspondantes dans l'image suivante. Ce processus se déroule en trois étapes principales : extraction des trajectoires, labellisation des trajectoires, et optimisation itérative.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 21

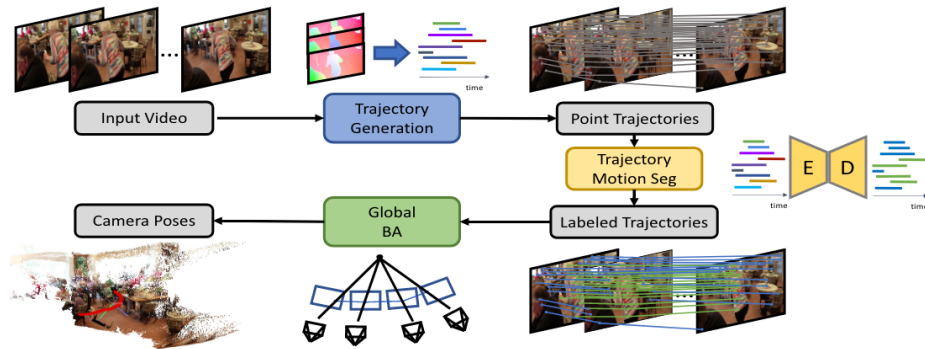


FIGURE 11 – Aperçu de la méthode utilisée afin de retrouver les positions des caméras.

Acquisition des trajectoires de points denses Dans cette étape, le réseau RAFT (Recurrent All-Pairs Field Transforms for Optical Flow 6.3.1) est utilisé comme prédictor de flow optique. Pour chaque point visible sur la première image, les trajectoires de ces points sont suivies de manière récurrente jusqu'à ce qu'elles soient occultées, ce qui est vérifié grâce à un contrôle de la cohérence du flux optique. Durant ce processus, de nouvelles trajectoires sont générées dans les zones qui n'en présentaient pas auparavant, correspondant ainsi à des points nouvellement visibles. Afin de lutter contre la dérive, où les trajectoires des points commencent à s'éloigner des trajectoires réelles, la méthode exploite la cohérence des chemins en tirant parti du flux optique entre des paires d'images non adjacentes. Cette approche est formalisée par les équations suivantes :

$$p'_1 = p_0 + F_{0 \rightarrow 1}(p_0), \quad p'_2 = p'_1 + F_{1 \rightarrow 2}(p'_1) \quad (26)$$

$$L = (p_1 - p'_1)^2 + (p_2 - (p_0 + F_{0 \rightarrow 2}(p_0)))^2 + (p_2 - (p_1 + F_{1 \rightarrow 2}))^2 \quad (27)$$

Une contrainte de stride 2 (équation 27) est généralement suffisante pour améliorer la précision des trajectoires, mais cette méthode peut être étendue pour exploiter un flux optique sur une plus longue distance.

Segmentation du mouvement basée sur les trajectoires L'idée initiale est d'exploiter les trajectoires denses précédemment calculées pour estimer les étiquettes de mouvement, permettant ainsi de localiser les différents objets ou points de référence en mouvement. Le processus commence par un encodeur qui prend en entrée les données de trajectoires irrégulières, caractérisées par des débuts, des fins et des durées différentes pour chaque trajectoire, et les intègre dans un espace de caractéristiques de haute dimension. Toutes les caractéristiques encodées sont ensuite transmises à un décodeur, qui effectue une agrégation des caractéristiques en tenant compte du contexte local et global parmi les trajectoires, afin de fusionner ces informations et de permettre la régression finale de l'étiquette de mouvement.

L'extraction des caractéristiques est réalisée en utilisant un modèle de transformateur, basé sur des "multi-head attention models" issus du traitement du langage et étendus aux tâches de vision. Les trajectoires sont rognées et remplies (padded) pour correspondre à la taille de la fenêtre temporelle de taille $(L, 2)$, puis augmentées avec les informations de mouvement consécutif $\Delta u_i, \Delta v_i = (u_{i+1} - u_i, v_{i+1} - v_i)$. Les objets en mouvement étant plus faciles à classifier dans l'espace 3D, l'intégration des informations de profondeur relative normalisée

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 22

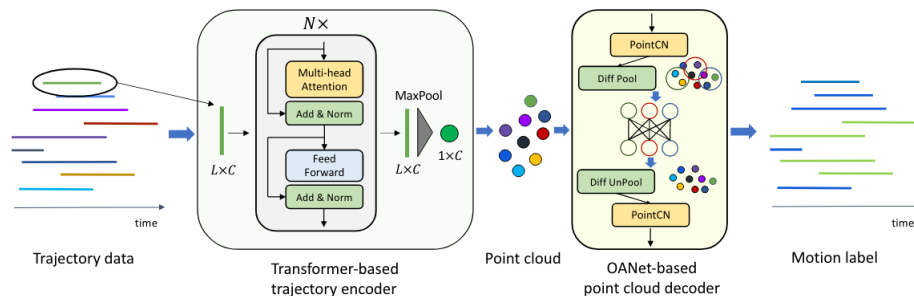


FIGURE 12 – Trajectory-Based Motion Segmentation (ParticleSfM)

provenant de MiDaS [16] permet de dissiper les ambiguïtés de la segmentation des mouvements basée uniquement sur les déplacements en 2D. Les données de mouvement 3D $(\Delta x_i, \Delta y_i, \Delta z_i)$ sont également utilisées comme entrée pour le modèle de transformateur.

Finalement, l'ensemble de caractéristiques de taille $(L, 10)$ est transmis à deux perceptrons multi-couches (MLP). Le résultat de taille (L, C) est ensuite introduit dans le module de transformateur, constitué de quatre blocs, chacun comportant un mécanisme d'attention multi-têtes et des couches de propagation avant (feed-forward layers), le tout conservant la même forme de sortie. Un max-pooling est effectué sur la dimension temporelle, aboutissant à un vecteur de forme $(1, C)$ pour chaque trajectoire. Le nuage de caractéristiques de forme (N, C) de l'espace de haute dimension est finalement introduit dans un décodeur spécifique basé sur OANet (Order-Aware Network 6.3.2).

Ajustement global par lot L'ajustement global par lot (Global Bundle Adjustment [21]) est la procédure consistant à affiner une reconstruction visuelle pour produire une structure 3D et des paramètres de prise de vue qui soient conjointement optimaux.

Ce processus repose sur une pipeline de Structure-from-Motion (SfM) utilisant des trajectoires de points denses. La première étape consiste à résoudre les poses relatives des caméras pour des vues voisines, en utilisant des correspondances échantillonnées entre des paires de pixels statiques extraits des trajectoires de points, en se basant sur les étiquettes de mouvement. Ensuite, les poses initiales des caméras sont estimées à l'aide de moyennes de rotations et de translations. Une fois ces poses déterminées, un ajustement global par lot est appliqué sur les trajectoires de points construites. Ce processus permet d'affiner la pose finale en se basant sur les pixels statiques le long des trajectoires, tout en ignorant les pixels qui ont été classés comme appartenant à des objets en mouvement.

Quelques résultats qualitatifs sur un jeu de données de vérification synthétique sont présentés figure 13, démontrant la précision de la méthode présentée.

Cependant, cette qualité de rendu s'accompagne d'un coût en termes de temps de calcul : l'extraction de l'intégralité des poses associées à une vidéo de 30 secondes à une fréquence de 20 images par seconde

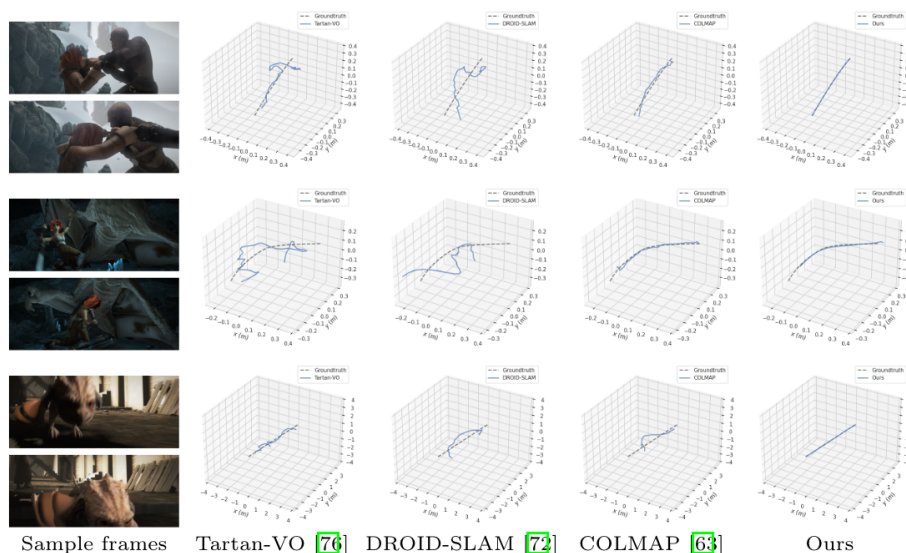


FIGURE 13 – Qualitative results of moving camera localization on MPI Sintel dataset

(résolution de 608 x 1080 pixels) nécessite environ 7h30 de traitement sur un GPU NVIDIA A100 équipé de 80GB de mémoire vidéo.

3.2 Génération du Dataset Synthétique

Afin de simplifier les entraînements, d'effectuer des prises de vue adaptées, et de vérifier la robustesse des modèles, la génération de jeux de données synthétiques s'est avérée nécessaire. Deux jeux de données ont été créés : l'un en 3D et deux autres en 4D. Le premier jeu de données 4D a été conçu pour la méthode de 4D Gaussian Splatting (4DGS) décrite dans la section 4, où une simulation temporelle multi-caméras a été réalisée. Cette simulation (reconstitution complète de la scène comprenant chaque point de vue de l'UR), impossible à reproduire avec le dataset réel en raison de l'utilisation d'un seul iPhone dû à des défis de synchronisation, a permis de capturer les dynamiques temporelles avec une grande précision. Le second jeu de données 4D concerne la simulation d'un bras robotique dans M positions différentes, avec une capture multi-caméras pour chaque configuration, afin de fournir un ensemble complet de vues (180 points de vue différents) pour optimiser les modèles liés au bras articulé, comme détaillé dans la section 5. Ces jeux de données synthétiques offrent de plus une flexibilité et une richesse d'informations essentielles pour le développement et la validation des modèles.

3.2.1 Acquisition directe des poses avec Unity

Les travaux se concentrent principalement sur le robot Universal Robot 10 (UR10), qui est fidèlement reproduit dans l'environnement Unity, permettant une simulation complète de la physique du bras robotique ainsi que l'intégration du logiciel de contrôle du robot. Nous avons donc choisi d'utiliser ce logiciel pour générer le dataset synthétique.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 24

Pour capturer les données, les extensions natives de Unity se sont avérées insuffisantes en raison du nombre élevé de caméras présentes dans la scène. Pour pallier cette limitation, j'ai développé des scripts en C# (détaillés dans le document C# ci-joint) qui permettent de capturer la scène en 3D et en 4D, ainsi que d'extraire les matrices intrinsèques et extrinsèques des caméras. Ces matrices sont ensuite converties au format **.npy** pour être facilement intégrées dans nos programmes.

Ces deux méthodes (Unity et COLMAP) ont été entièrement automatisées en ligne de commande, facilitant ainsi leur utilisation dans divers scénarios. Les données générées avec Unity servent de base ou de ground truth pour mesurer l'erreur entre les poses réelles (provenant d'Unity) et les poses estimées par des logiciels tiers tels que ParticleSfM et COLMAP. En plus de vérifier les estimations de poses, ces données sont également utilisées pour entraîner les modèles.

3.2.2 Acquisition des poses avec COLMAP

Dans le cadre d'une extension Les informations nécessaires aux différents entraînements peuvent également être générées avec COLMAP, que ce soit en ligne de commande ou via l'interface graphique (GUI). Ces poses estimées sont bien moins précises que les poses synthétiques, mais nous permette d'estimer l'ordre de grandeur des erreurs que nous aurons lors de l'utilisation sur les données réelles, le but étant de modéliser l'UR à partir de captations réelles. De nombreux tutoriels sont disponibles en ligne pour guider l'utilisateur dans ce processus. La méthode utilisée pour obtenir ces données est similaire à celle de ParticleSfM, ne nécessitant qu'une série d'images comme point de départ. Voici un exemple de résultat obtenu pour la scène comportant le robot UR10 :

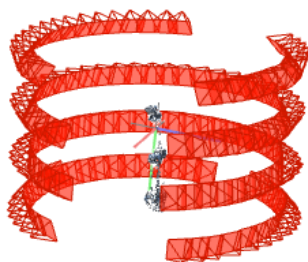


FIGURE 14 – Processus de reconstruction (COLMAP)

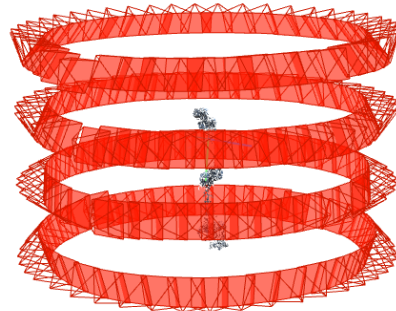


FIGURE 15 – Scène reconstruite (COLMAP)

Cette méthode propose une reconstruction précise de la scène, atteignant une erreur moyenne d'écart de placement par caméra de 0.007 (distance normalisée en divisant par le diamètre de la scène).

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 25

4 4D Gaussian Splatting (4DGS)

Compte tenu des résultats prometteurs obtenus avec le 3D Gaussian Splatting, diverses extensions et améliorations ont été développées pour enrichir cette méthode. L'objectif principal de ce stage était d'introduire de nouveaux degrés de liberté lors de l'entraînement, en particulier en ce qui concerne les notions d'angle et de mouvement. En étendant la méthode vers la 4D, l'incorporation de la dimension temporelle permet de représenter des scènes dynamiques, où l'animation volumique devient possible. Cela signifie qu'il est désormais possible de jouer une animation et de déplacer librement le point de vue de l'observateur à travers la scène, offrant ainsi une flexibilité accrue pour l'exploration et l'analyse de scénarios en mouvement. Le travail [23] ouvre des perspectives inédites pour la modélisation et le rendu d'animations tridimensionnelles complexes, où les objets ne sont plus figés dans le temps, mais évoluent de manière fluide et contrôlable.

4.1 Intégration du temps dans les gaussiennes

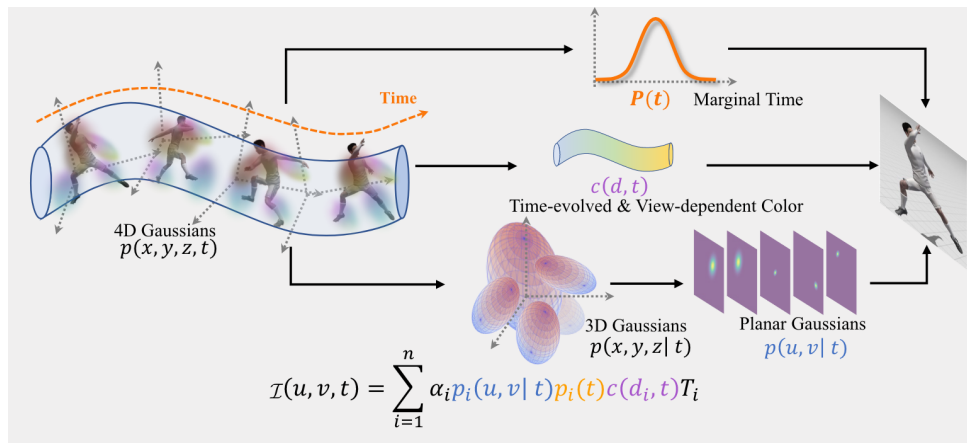


FIGURE 16 – Représentation schématique de la méthode 4DGS

L'intégration du temps dans les gaussiennes implique une approche où le temps et l'espace sont traités différemment, avec la couleur d'un pixel $\mathcal{I}(u, v, t)$ déterminée par ses coordonnées spatiales (u, v) ainsi que sa position temporelle t . L'extension de l'équation 18 à la 4D résulte en :

$$\mathcal{I}(u, v, t) = \sum_{i \in \mathcal{N}} p_i(u, v, t) \alpha_i c_i(d) \prod_{j=1}^{i-1} (1 - p_j(u, v, t) \alpha_j) \quad (28)$$

$$= \sum_{i \in \mathcal{N}} p_i(t) p_i(u, v | t) \alpha_i c_i(d) \prod_{j=1}^{i-1} (1 - p_j(t) p_j(u, v | t) \alpha_j) \quad (29)$$

Pour représenter une gaussienne en 4D, il est essentiel de considérer que les variables t et (x, y, z) sont indépendantes, ce qui permet de réécrire la probabilité conditionnelle d'une gaussienne comme $p_i(x, y, z | t) = p_i(x, y, z)$. L'ajout de $p_i(t)$ dans l'équation 28 introduit une dimension temporelle, modifiant l'évolution des

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 26

attributs de la gaussienne au fil du temps.

Pour créer un modèle cohérent de gaussienne 4D, les dimensions spatiales et temporelles sont traitées de manière identique, et la représentation de la matrice de covariance Σ reste similaire à celle utilisée pour le 3DGS (voir 17). Cependant, les matrices de covariance en 4D ont une dimension supplémentaire : $S = \text{diag}(s_x, s_y, s_z, s_t) \in \mathbb{R}^{4 \times 4}$ et $\mathbb{R}$ est une matrice de rotation de taille 4×4 .

Cette représentation permet de retrouver les propriétés des gaussiennes 3D à un instant t fixé :

$$\mu_{xyz|t} = \mu_{1:3} + \Sigma_{1:3,4} \Sigma_{4,4}^{-1} (t - \mu_t) \quad (30)$$

$$\Sigma_{xyz|t} = \Sigma_{1:3,1:3} - \Sigma_{1:3,4} \Sigma_{4,4}^{-1} \Sigma_{4,1:3} \quad (31)$$

4.2 Extension des Harmoniques Sphériques

L'aspect directionnel des splats varie dans le temps en fonction de leur déplacement et de leur rotation, ce qui nécessite une représentation dynamique des couleurs. Dans la version originale du 3D Gaussian Splatting, les couleurs sont représentées à l'aide d'harmoniques sphériques (SH). Plutôt que de dupliquer les gaussiennes à chaque instant avec des variations de couleur, la notion d'harmoniques sphériques a été étendue pour inclure une composante temporelle, augmentant la dimension de cette base de fonctions.

Pour cette extension temporelle, le choix de la base de fonctions 1D s'est porté sur les séries de Fourier, en raison de leur optimisation efficace et des temps de calcul réduits :

$$Z_{nl}^m(t, \theta, \phi) = \cos\left(\frac{2\pi n}{T}t\right) Y_l^m(\theta, \phi), \quad \text{où } l \geq 0, -l \leq m \leq l \quad (32)$$

où Y_l^m représente les harmoniques sphériques 3D et n le degré de la série de Fourier. Cette combinaison forme une base orthonormée dans le système de coordonnées donné.

4.3 Optimisations

Plusieurs articles récents ont proposé des optimisations basées sur des modèles similaires, en ajustant certaines composantes pour surmonter les défis de convergence dans des scènes complexes. Ces optimisations se divisent en deux grandes catégories : l'optimisation du mouvement des gaussiennes dans le temps par l'ajustement du couplage spatio-temporel dans la représentation des rotations 4D, et l'ajout de contraintes sur le flux des gaussiennes [12, 7]. Dans la première méthode, tous les paramètres d'une gaussienne sont temporellement dépendants, tandis que dans la seconde, seuls certains attributs (couleur, orientation, position) varient dans le temps, en conservant un nombre fixe de gaussiennes 3D en mouvement. Les optimisations présentées s'inspirent du travail de [4]. Une étude comparative approfondie serait nécessaire pour évaluer les avantages de chaque méthode et envisager leur intégration dans la modélisation du bras robotique articulé. Cependant, cette analyse est rendue difficile par l'absence de code disponible, à l'exception de celui associé à [23].

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 27

4.3.1 Représentation des Rotations en 4D

En tirant parti des avancées en algèbre géométrique, les rotations en haute dimension, précédemment représentées par deux quaternions, ont été remplacées par des rotors 4D ($\mathbf{r}$), composés de 8 coefficients $(s, b_{01}, b_{02}, b_{03}, b_{12}, b_{13}, b_{23}, p)$, tels que :

$$\mathbf{r} = s + b_{01}\mathbf{e}_{01} + b_{02}\mathbf{e}_{02} + b_{03}\mathbf{e}_{03} + b_{12}\mathbf{e}_{12} + b_{13}\mathbf{e}_{13} + b_{23}\mathbf{e}_{23} + p\mathbf{e}_{123}$$

où $\mathbf{e}_{123} = \mathbf{e}_0 \wedge \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3$ et $\mathbf{e}_{ij} = \mathbf{e}_i \wedge \mathbf{e}_j$ avec $i \in \{0, 1, 2, 3\}$. Ici, $\wedge$ représente le produit extérieur et les $\mathbf{e}_i$ sont les vecteurs formant la base orthonormée de l'espace euclidien de dimension 4.

4.3.2 Spécification de la fonction de coût pour le cas 4D

Afin de stabiliser l'entraînement, deux termes de régularisation ont été ajoutés à la fonction de coût présentée dans l'équation 20 :

Une perte d'entropie $\mathcal{L}_{entropy}$, qui encourage les gaussiennes à avoir une opacité α proche de 1.

$$\mathcal{L}_{entropy} = \frac{1}{N} \sum_{i=1}^N -\alpha_i \log(\alpha_i) \quad (33)$$

Une perte de cohérence qui favorise la proximité des gaussiennes dans l'espace :

$$\mathcal{L}_{consistent4D} = \frac{1}{N} \sum_{i=1}^N \left\| \mathbf{s}_i - \frac{1}{K} \sum_{j \in \Omega_i} \mathbf{s}_j \right\| \quad (34)$$

où Ω_i représente les K plus proches voisins de la i -ème gaussienne (obtenus avec l'algorithme KNN).

4.3.3 Paramétrisation du mouvement dans la scène

La modélisation directe des dynamiques d'une gaussienne 3D peut être réalisée en ajustant chacun de ses attributs à une courbe dépendante du temps. Parmi les approches couramment utilisées (par exemple [7]), l'ajustement polynômial dans le domaine temporel et l'ajustement par séries de Fourier dans le domaine fréquentiel se démarquent par leur simplicité et leur efficacité. Cependant, chaque méthode présente des avantages et des inconvénients. Par exemple, l'ajustement polynômial est efficace pour les mouvements lisses en utilisant un petit ordre de polynômes, mais il peut facilement sur-ajuster les mouvements violents si un ordre plus élevé est utilisé, entraînant des oscillations non réalistes dans la trajectoire ajustée. D'autre part, les séries de Fourier capturent bien les variations associées aux mouvements violents, mais nécessitent une réduction manuelle de l'ordre lorsqu'il s'agit de mouvements plus doux.

Pour surmonter ces limitations, le papier [12] propose un modèle de déformation à double domaine (Dual-Domain Deformation Model, DDDM) qui intègre les polynômes et les séries de Fourier en un modèle d'ajustement unifié. Il est supposé que seuls la rotation $\mathbf{q}$, la radiance $\mathbf{c}$ et la position μ d'une gaussienne changent au cours du temps, tandis que l'échelle s et l'opacité α restent constantes. Le changement dans chaque

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 28

attribut est modélisé par une déviation $D(t)$, somme d'un polynôme $P_N(t)$ et d'une série de Fourier $F_L(t)$, superposée aux attributs de base S_0 au temps de référence t_0 :

$$S(t) = S_0 + D(t), \quad D(t) = P_N(t) + F_L(t),$$

où

$$P_N(t) = \sum_{n=0}^N a_n t^n,$$

$$F_L(t) = \sum_{l=1}^L (f_{sin}^l \cos(lt) + f_{cos}^l \sin(lt)).$$

L'analyse comparative illustrée de la figure 17 montre que l'approche proposée dépasse les méthodes traditionnelles en capturant efficacement les trajectoires complexes de mouvement, comme le démontrent les points de données échantillonnés. Une extension vers la représentation d'un modèle articulé est envisageable quant aux exigences en terme de mémoire : stockage de $\mathcal{N} \times (4 + 48 + 3) \times (N + 2L)$ coefficients, $\mathcal{N}$ étant le nombre de gaussiennes, N le degré du polynôme et L l'ordre de la série de Fourier, correspondant à environ 1GB pour 100 000 gaussiennes, prenant en compte des changements de la position $\mu \in \mathbb{R}^3$, de la couleur $c \in \mathbb{R}^{48}$ et de l'orientation $q \in \mathbb{R}^4$. La quantité de mémoire nécessaire explose néanmoins quand on ne se restreint plus qu'à un seul degré de liberté (modélisation de plusieurs joints de l'UR).

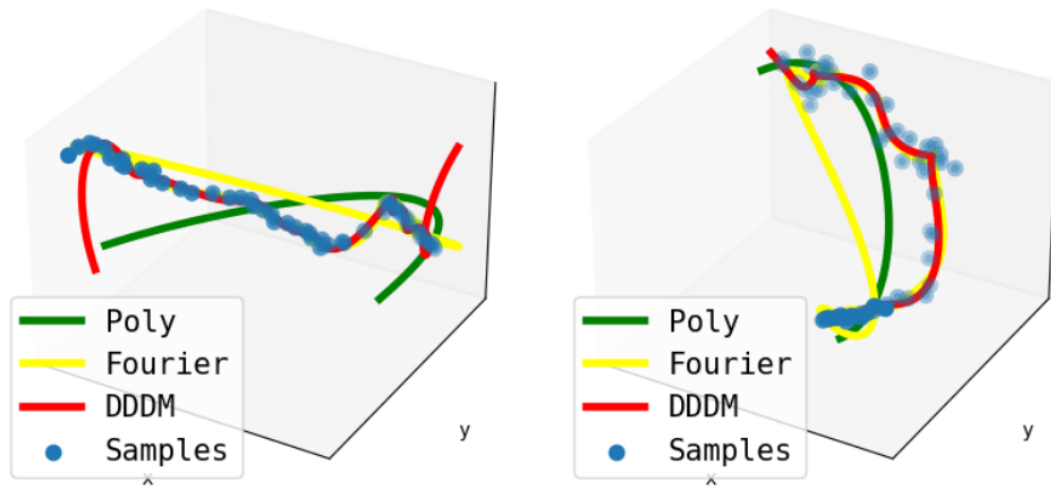


FIGURE 17 – Modélisation de Trajectoires

4.4 Analyse des Résultats

Cette méthode a été appliquée au dataset réel de l'UR, ce qui a abouti à un PSNR d'environ 42 sur les images d'entraînement et de 21 sur les images tests. Les rendus correspondants sont présentés dans les images / figures 18, 19, 20, 21. L'entraînement a pris 7h30, mais permet néanmoins d'atteindre des rendus en temps réel grâce à un temps d'inférence de l'ordre de 0.01s par image, correspondant à environ 97fps. La figure 24 illustre la capacité de cette méthode à représenter des scènes comprenant à la fois des objets statiques et en mouvement, une fonctionnalité essentielle pour le 3D Gaussian Splatting interactif. D'autres papiers ont vu le jour depuis mais leur code n'est pas accessible. Les méthodes mentionnées précédemment ([4] et [12]) permettraient une augmentation par un facteur 6 du nombre d'images par secondes (583 fps sur une RTX 4090 et 277 fps avec une RTX 3090), tout cela en obtenant des rendus plus détaillés, comme présenté dans le tableau 23. Cela est rendu possible grâce à un framework CUDA spécialisé et optimisé. Les résultats quantitatifs sur les jeux de données test sont présentés dans la figure 22.



FIGURE 18 – Rendu de la pose connue (4DGS)



FIGURE 19 – Ground-Truth de la pose connue (4DGS)



FIGURE 20 – Rendu de la pose inconnue(4DGS)

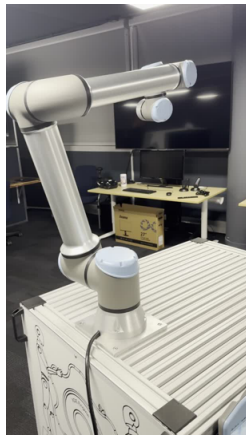


FIGURE 21 – Ground-Truth de la pose inconnue(4DGS)

	Coffee Martini	Spinach	Cut Beef	Flame Salmon	Flame Steak	Sear Steak	Mean
HexPlane	-	32.04	32.55	29.47	32.08	32.39	31.70
K-Planes-explicit	28.74	32.19	31.93	28.71	31.80	31.89	30.88
K-Planes-hybrid	29.99	32.60	31.82	30.44	32.38	32.52	31.63
Mix Voxels	29.36	31.61	31.30	29.92	31.21	31.43	30.80
NeRFPlayer	31.53	30.56	29.353	31.65	31.93	29.12	30.69
HyperReel	28.37	32.30	32.922	28.26	32.20	32.57	31.10
Ours	28.33	32.93	33.85	29.38	34.03	33.51	32.01

FIGURE 22 – Résultats qualitatifs 4DGS sur différentes scènes

ID	Method	PSNR↑	SSIM↑	LPIPS↓	Train↓	FPS↑
<i>a</i>	DyNeRF [33]*†	29.58	-	0.08	1344 h**	0.015
<i>b</i>	StreamRF [32]	28.16	0.85	0.31	79 min	8.50
<i>c</i>	HyperReel [2]	30.36	0.92	0.17	9 h	2.00
<i>d</i>	NeRFPlayer [47]†	30.69	-	0.11	6 h	0.05
<i>e</i>	K-Planes [19]	30.73	0.93	0.07	190 min	0.10
<i>f</i>	MixVoxels [54]	30.85	0.96	0.21	91 min	16.70
<i>g</i>	Deformable4DGS [56]	28.42	0.92	0.17	72 min	39.93
<i>h</i>	RealTime4DGS [61]	29.95	0.92	0.16	8 h	72.80
<i>i</i>	Ours	31.62	0.94	0.14	60 min	277.47

FIGURE 23 – Tableau comparatif de méthodes 4DGS



FIGURE 24 – Synthèse de nouvelles vues dans un environnement urbain

5 3D Gaussian Splatting Interactif

Ce travail a porté sur le développement d'une extension des modèles 3DGS standard [10] et Scaffold-GS [13], visant à modéliser un système articulé capable de simuler des animations avec un contrôle précis de

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 31

l'utilisateur. Dans cette approche, le mouvement est piloté par l'utilisateur, permettant non seulement de régir le déplacement des objets, mais aussi de modifier librement le point de vue au sein de la scène. Au cours de ce stage, nous nous sommes restreints à l'ajout d'un degré de liberté, représenté par un angle θ , introduit dans le but de régir le mouvement d'une partie du robot UR. La création d'un dataset d'entraînement spécifique a été nécessaire pour cette modélisation. Cette méthode pose les bases pour une extension future permettant de modéliser un système articulé complet du robot UR10, ajoutant ainsi jusqu'à 6 degrés de liberté, contre un seul dans le cas du 4DGS.

L'objectif ici n'était pas simplement de se contenter d'un rendu sous Unity, mais de viser un rendu extrêmement photoréaliste. Cette exigence est cruciale pour des applications où la précision visuelle est primordiale, comme dans les simulations d'environnements complexes tels qu'un atelier technique ou un hall d'usine. Un rendu photoréaliste permettrait non seulement de mieux visualiser les interactions entre les objets et leur environnement, mais aussi de garantir que le système puisse être utilisé dans des contextes nécessitant une haute fidélité visuelle.

Pour mener à bien ce travail, une étude bibliographique approfondie a été réalisée, couvrant les méthodes les plus récentes dans ce domaine de recherche très actif (certaines de ces méthodes sont apparues au cours même de ce stage). Les principes de ces méthodes ont été étudiés et testés, lorsque le code source était accessible, afin d'évaluer leur pertinence et leur potentiel pour le développement de modèles articulés dynamiques.

Cette section est dédiée au fitting des coefficients des harmoniques sphériques afin de s'intégrer dans le processus de rotation des gaussiennes, tel qu'illustré dans la figure 26, qui schématise le flux de travail du contrôle des gaussiennes et leur ajustement à travers les différentes étapes du pipeline, en partant de la génération des masques jusqu'à la rotation des centres et orientations à l'aide des matrices D de Wigner. Ce travail est effectué notamment dans le but de représenter les effets directionnels lumineux et d'auto-ombrage lors du mouvement du bras robotique.

5.1 Génération du Dataset d'entraînement

L'intégration d'un degré de liberté supplémentaire entraîne également une augmentation significative de la quantité de données nécessaires. À titre de test, une discrétisation uniforme de l'intervalle $[0, 45]$ (en degrés) a été réalisée en $\mathcal{M}$ valeurs. Cela a conduit à la création de $\mathcal{M}$ datasets 3DGS standards, générant ainsi un total de $\mathcal{M} \times N$ images et matrices. Cette augmentation de la quantité de données a soulevé un nouveau défi, à savoir la gestion de la mémoire, en particulier sur les GPU, dont la capacité est limitée, lors de l'entraînement du modèle. En effet, les images sont chargées à la volée sur le GPU après la phase de rendu, augmentant par un facteur d'environ 2.3 la durée d'entraînement.

De plus, d'autres informations sont requises, et des procédés doivent être développés pour identifier les différentes parties de l'UR afin de classer les gaussiennes par groupe, ainsi que pour déterminer les pivots associés à chaque groupe et chaque axe. Cette segmentation est indispensable car elle permet d'appliquer des transformations géométriques spécifiques sur des groupes de splats, correspondant chacun à un axe du robot. Cette approche est suffisante pour modéliser un système articulé comme le robot UR, voire un modèle souple tel qu'un câble, mais elle reste limitée pour des objets déformables comme des liquides ou des flammes.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 32



FIGURE 25 – Visualisation des degrés de libertés de l'UR10

Pour le cas test, ces opérations ont été réalisées manuellement à l'aide du logiciel MeshLab (voir Figure 27 et 28), avec pour objectif d'étendre cette approche à l'automatisation, comme décrit dans la section 6.1. Ce même logiciel a également été utilisé pour calculer les pivots, représentés en rouge sur la Figure 29, illustrant les trois pivots nécessaires pour les rotations autour de l'axe x .

Les données d'entraînement sont finalement converties en une instance de torch.Dataset afin de pouvoir profiter de l'optimisation et des automatisations du torch.DataLoader suivant le code en annexe ??.

5.2 Suivi des Groupes

Deux approches ont été explorées pour assurer le suivi de l'appartenance des gaussiennes à leurs groupes respectifs : la pré-segmentation puis mise en place d'un suivi individuel des gaussiennes tout au long du processus d'optimisation, et la post-segmentation réalisée après coup.

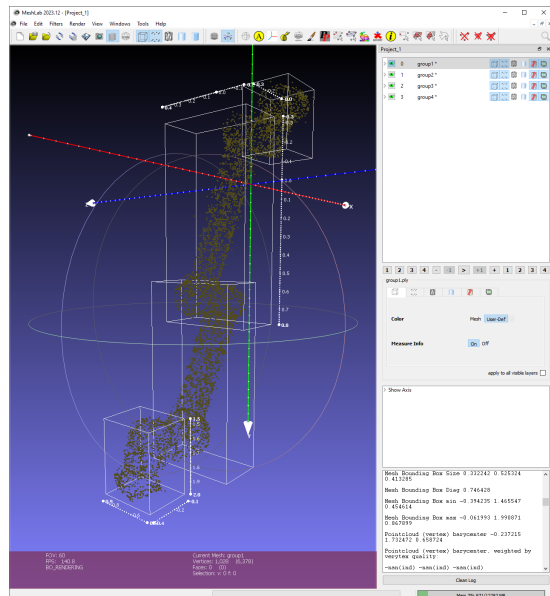


FIGURE 27 – Bounding Boxes

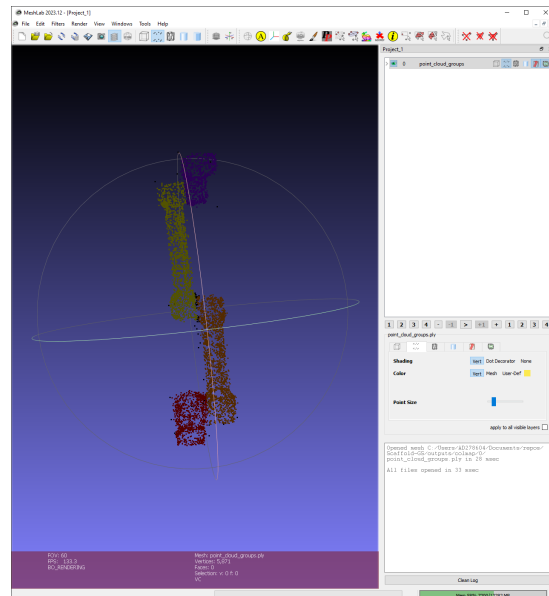


FIGURE 28 – Groupes générés

5.3 Première Méthode Directe : Modifications des Résultats Issus du 3DGS

Les gaussiennes possèdent trois propriétés principales dépendant de θ : leur position, leur orientation, et leur couleur. Ces propriétés peuvent être modifiées directement à l'aide de produits matriciels, ce qui a conduit à une première approche naïve : translation des centres, rotation des orientations, et transformation des harmoniques sphériques.

5.3.1 Translation des gaussiennes

La translation des gaussiennes est effectuée sur leurs centres à l'aide des points de pivot P et des coordonnées G'_i (coordonnées de la i -ème gaussienne) selon le produit matriciel suivant :

$$G'_i = R(G_i - P) + P \quad (35)$$

Ce processus est optimisé grâce à la multiplication matricielle par lots (batch-matrix-multiplication) de PyTorch :

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 35

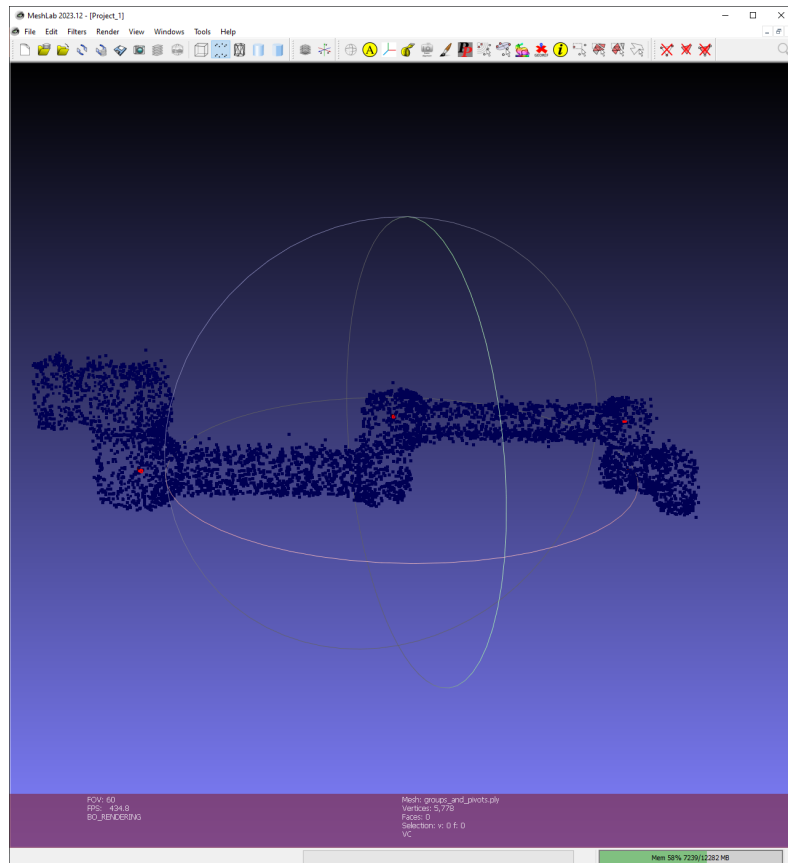


FIGURE 29 – Pivots (rouge)

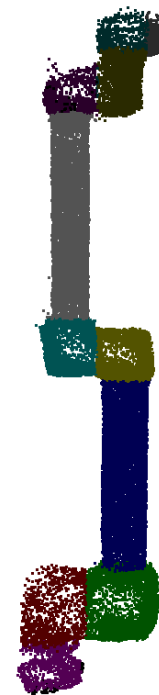


FIGURE 30 – Segmentation en 11 sous-groupes

```

1 def rotate_gaussians(self, group_idx : int, theta : float, axis : str):
2     self regroup_and_prune()
3     self.define_pivots()
4     pivot_point = torch.tensor(self.get_x_pivot(group_idx), dtype=torch.float, device="cuda")
5     group_mask = self._groups >= group_idx
6     N = int(group_mask.sum().item())
7     rotation = self.create_rotation_matrix(theta, axis)
8     rotations = rotation.repeat(N, 1, 1).float()
9     bmm = torch.bmm(rotations, (self._xyz[group_mask] - pivot_point).unsqueeze(-1)).squeeze(-1)
10    repeated_pivot = pivot_point.repeat(N, 1).float()
11    self._xyz[group_mask] = torch.add(bmm, repeated_pivot).float()

```

FIGURE 33 – Calcul des nouvelles coordonnées

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 36

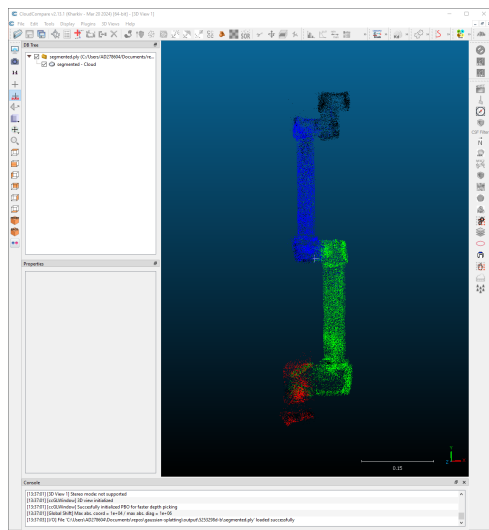


FIGURE 31 – Pré-segmentation des gaussiennes et suivi

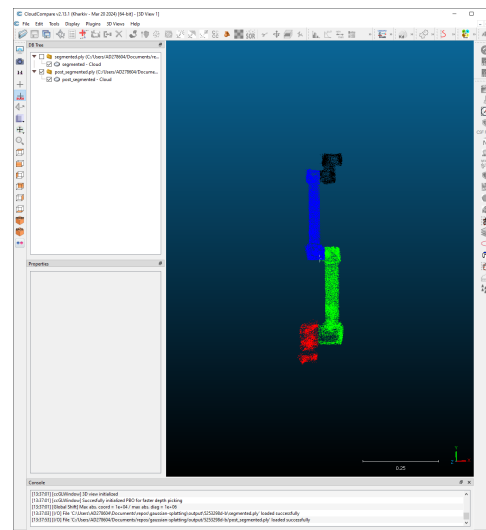


FIGURE 32 – Post-segmentation des gaussiennes

5.3.2 Rotation des gaussiennes

Les gaussiennes n'étant pas des points, l'orientation des axes principaux est importante afin de conserver une structure cohérente. Ne pas procéder à cette manipulation mène à des anomalies comme le montre la figure 34

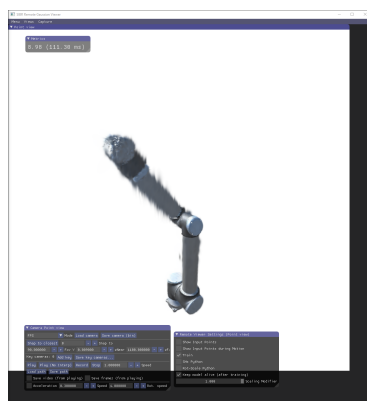


FIGURE 34 – Sans rotation des orientations exemple 1



FIGURE 35 – Sans rotation des orientations exemple 2

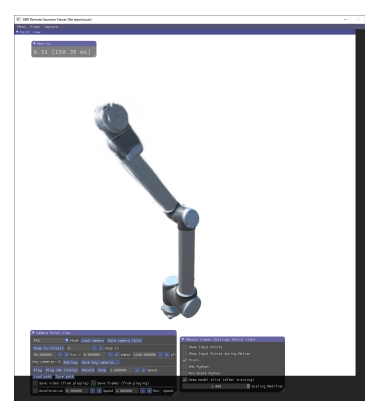


FIGURE 36 – Avec rotation des orientations

Ceci est aussi optimisé sur GPU comme pour le calcul des nouvelles positions des gaussiennes. Néanmoins, l'orientation étant stockée sous forme de quaternion, une étape de conversion a été ajoutée, visible dans le code associé. L'utilisation du module o3 de la librairie e3nn s'est avérée nécessaire afin que les quaternions générés suivent la même convention que dans l'article : (w, x, y, z) où $w \geq 0$.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 37

```

1 def rotate_gaussians(self, group_idx : int, theta : float, axis : str):
2     self.regroup_and_prune()
3     self.define_pivots()
4     pivot_point = torch.tensor(self.get_x_pivot(group_idx), dtype=torch.float, device="cuda")
5     group_mask = self._groups >= group_idx
6     N = int(group_mask.sum().item())
7     rotation = self.create_rotation_matrix(theta, axis)
8     rotations = rotation.repeat(N, 1, 1).float()
9     bmm = torch.bmm(rotations, (self._xyz[group_mask] - pivot_point).unsqueeze(-1)).squeeze(-1)
10    repeated_pivot = pivot_point.repeat(N, 1).float()
11    self._xyz[group_mask] = torch.add(bmm, repeated_pivot).float()
12
13    # same but for the internal rotation of the gaussians
14    rotations = rotation.repeat(N, 1, 1).double()
15    rotated_rotations = build_rotation(self._rotation[group_mask]).double()
16    rotated_rotations = torch.bmm(rotations, rotated_rotations)
17    angles = o3._rotation.matrix_to_quaternion(rotated_rotations.cpu()).float()
18    self._rotation[group_mask] = angles.to(device="cuda")

```

FIGURE 37 – Calcul des nouvelles orientations

5.3.3 Gestion des couleurs / Harmoniques Sphériques

Comme mentionné précédemment, les couleurs, étant dépendantes du point de vue de l'observateur, sont encodées à l'aide d'harmoniques sphériques (SH). Toutefois, étant donné que les positions et les orientations des gaussiennes changent, une transformation des coefficients SH est nécessaire afin d'éviter toute altération des couleurs due aux variations de la direction de vue.

Les matrices $D_{m'm}^{\ell}(\alpha, \beta, \gamma)$ de Wigner ([19] et [6]) sont cruciales pour effectuer des rotations des coefficients d'harmoniques sphériques, ce qui est crucial dans le contexte du 3D Gaussian Splatting Dynamic pour représenter avec précision les propriétés directionnelles (elles sont particulièrement utilisées en mécanique quantique). En tant que représentations irréductibles du groupe $SO(3)$, elles permettent de transformer les harmoniques sphériques sous des rotations définies par les angles d'Euler α , β , et γ , comme illustré dans les figures 38 et 39 (disponibles également en annexe 6.4).

Les éléments $d_{m'm}^{\ell}(\beta)$, composants des matrices $D_{m'm}^{\ell}(\alpha, \beta, \gamma)$, décrivent spécifiquement les rotations autour de l'axe y , essentielles pour ajuster les orientations directionnelles tout en maintenant la cohérence des propriétés spatiales, comme les reflets et les couleurs dans un espace 3D. Les matrices D de Wigner sont définies par :

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 38

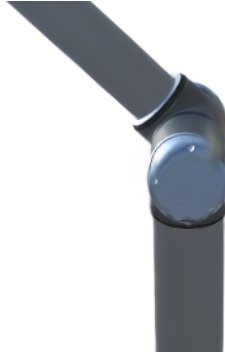


FIGURE 38 – Sans transformations des SH

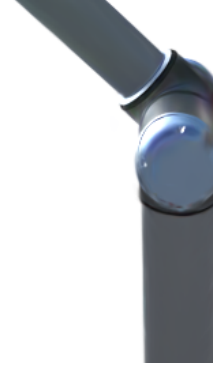


FIGURE 39 – Avec transformation des SH

$$D_{m'm}^l(\alpha, \beta, \gamma) = e^{-im'\alpha} d_{m'm}^j(\beta) e^{-im\gamma}$$

$$d_{m'm}^l(\beta) = [(l+m')!(l-m')!(l+m)!l-m)!]^{\frac{1}{2}}$$

$$\times \sum_{s=s_{\min}}^{s_{\max}} \frac{(-1)^{m'-m+s} \left(\cos \frac{\beta}{2}\right)^{2j+m'-m-2s} \left(\sin \frac{\beta}{2}\right)^{m'-m+2s}}{(l+m-s)!s!(m'-m+s)!(j-m'-s)!}$$

où $s_{\min} = \max(0, m - m')$ et $s_{\max} = \min(j + m, j - m')$.

De plus, ces matrices $D_{m'm}^l(\alpha, \beta, \gamma)$ forment une base orthogonale complète de $L^2(SO(3))$. Cela signifie que toute fonction sur $SO(3)$, représentant des transformations géométriques en trois dimensions, peut être exprimée par une combinaison linéaire de ces matrices. Cette orthogonalité est en lien avec les polynômes de Jacobi, comme l'illustre l'intégrale suivante :

$$\int d\nu D_{mm'}^{j_1} D_{mm'}^{j_2*} = C \int_{-1}^1 dx (1-x)^a (1+x)^b P_{k_1}^{(a,b)}(x) P_{k_2}^{(a,b)}(x) = C \delta_{j_1 j_2}$$

Finalement, la figure 40 donne une représentation des différences entre le rendu et l'image de référence. Celle-ci est générée en prenant en compte la différence absolue pixel par pixel et en la normalisant afin de représenter une carte de pourcentage de différence quant à la couleur RGB. Cette méthode permet d'obtenir des temps de rendus très bas, de l'ordre de 0.089 secondes, même si elle ne permet pas de modéliser tous les détails lors du mouvement d'une partie articulée.

5.4 2nde Méthode : Boite Noire (MLP)

L'utilisation de la méthode naïve permet de capturer la couleur de la partie transformée, mais présente des limitations importantes en ce qui concerne la gestion des changements de position dans l'espace. Cette approche ne parvient pas à reproduire les variations liées aux nouvelles positions des objets, ce qui entraîne un manque de généralisation, en particulier pour les tâches spéculaires et les ombres. Ces propriétés dépendent

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 39

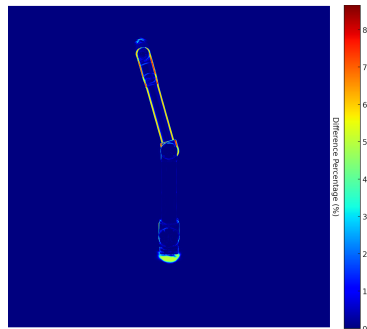


FIGURE 40 – Différence normalisée

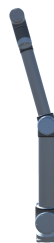


FIGURE 41 – Image de référence

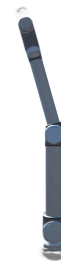


FIGURE 42 – Rendu associé

fortement de l'environnement et des caractéristiques de la source lumineuse, ce que la méthode naïve ne peut pas gérer efficacement. Pour remédier à ces limitations, diverses approches de régression ont été explorées, incluant des modèles non paramétriques (sans hypothèse de forme) et l'ajustement polynômial. Cependant, en raison de la complexité de la structure en entrée et des interactions entre les différentes composantes de l'UR (les groupes de gaussiennes), ces approches, inspirées par les travaux présentés dans [22], n'ont pas abouti aux résultats escomptés. En conséquence, l'implémentation d'un MLP a été privilégiée ([3] a donné beaucoup de bases pour son implémentation).

5.4.1 Présentation du MLP

Pour pouvoir ajuster les emplacements des tâches spéculaires et des ombres, il est nécessaire de mettre en place une méthode de régression non paramétrique permettant d'adapter les coefficients d'harmoniques sphériques en fonction de l'angle du bras de l'UR. Ces coefficients sont représentés par des tenseurs de taille $(\mathcal{N}, 16, 3)$, où $\mathcal{N}$ correspond au nombre de gaussiennes dans la scène, qui excède souvent les centaines de milliers.

L'architecture adoptée repose sur un réseau de neurones profonds défini par $DNN : [\cos(\theta), \sin(\theta)] \times SH_0 = [\mathcal{N}, 16, 3] \rightarrow SH_i = [\mathcal{N}, 16, 3]$, où i représente l'indice de la position associée à θ . Les entrées seront ensuite complétées par l'ajout des positions normalisées des gaussiennes. Des travaux tels que [5], qui explorent l'application du Deep Learning dans le domaine de la génomique, ont été particulièrement utiles pour aborder la gestion de grandes séquences de données unidimensionnelles. En particulier, ces travaux ont fourni des stratégies efficaces pour le partitionnement des données et le partage de l'information à travers les différentes couches du réseau de neurones.

Cela soulève immédiatement la question de la taille des données à traiter en entrée ainsi que du stockage nécessaire pour la rétropropagation du gradient, effectuée par l'outil autograd de la bibliothèque PyTorch. Ce problème met en avant la nécessité de procéder à une réduction de dimension.

Pour résoudre ce problème, l'objectif est d'organiser les gaussiennes en sous-groupes et de leur associer des

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 40

réseaux de neurones profonds (DNN) plus petits. 11 sous-groupes ont été générés au total, représentant les joints et le bras de chaque partie de l'UR10 (Les 11 sous-groupes sont visibles sur la figure 30). Cette approche permet de combiner efficacement les informations extraites par la suite, tout en assurant que les poids des couches denses peuvent être correctement instanciés sur un GPU. Bien que les couches convolutives soient traditionnellement privilégiées pour le traitement d'images ou de données bidimensionnelles, leur utilisation est inapplicable dans notre cas. En effet, les données des gaussiennes ne sont pas ordonnées spatialement, et un mappage de $\mathbb{R}^3 \rightarrow \mathbb{R}^n$ avec $n \in \{1, 2\}$ entraînerait une perte significative des informations de distance (par exemple lors de l'utilisation de courbes de Lebesgue).

L'ajout de connections résiduelles combiné à la fonction d'activation Sigmoid, permettant de restreindre l'intervalle de sortie à $[0, 1]$ a permis une stabilisation de l'entraînement (code disponible en annexe ??). Ceci a produit le rendu de la figure 43, sur laquelle on peut observer une légère adaptation de la tâche spéculaire de référence quant à l'orientation du bras.



FIGURE 43 – Rendu par 3DGS Dynamique (vue 1)

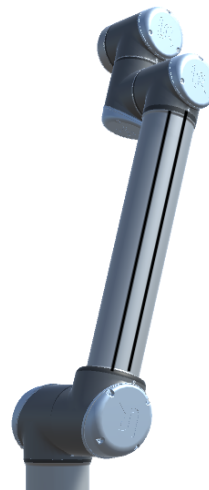


FIGURE 44 – Ground-Truth associée (vue 1)

Néanmoins, cette adaptation ne permet pas une généralisation à l'ensemble des positions que peut prendre le bras de l'UR : des dégradations apparaissent lorsque l'angle est trop éloigné de la valeur de θ utilisée durant l'échauffement.

5.5 3ème Méthode : Utilisation de points d'ancrage

5.5.1 Présentation de Scaffold-GS

Pour réduire la dimension des entrées du MLP, une seconde approche inspirée du travail sur Scaffold-GS [13] a été étudiée. Cette méthode supprime entièrement l'optimisation stochastique directe sur les tenseurs d'attributs des gaussiennes, en confiant à de petits MLP la gestion du rendu des couleurs ($c \in \mathbb{R}^3$), opacités ($\alpha \in \mathbb{R}$), orientations ($q \in \mathbb{R}^4$) et amplitudes ($s \in \mathbb{R}^3$). De plus, la stratégie de clonage et de suppression des

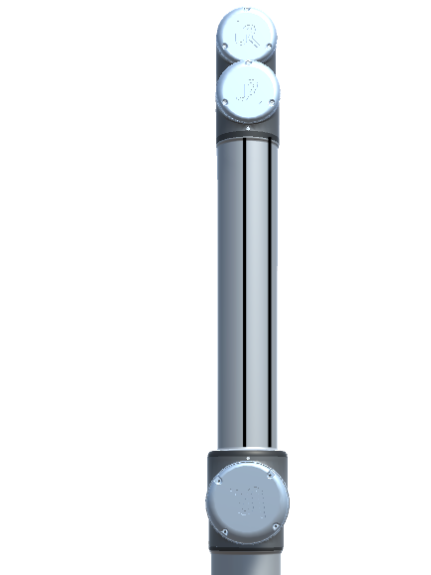


FIGURE 45 – Image de référence



FIGURE 46 – Rendu par 3DGS Dynamique (vue 2)

gaussiennes est remplacée par l'introduction de points d'ancrage, à partir desquels $k = 10$ "gaussiennes neurales" sont générées. À chaque point d'ancrage sont associés des offsets $\mathcal{O}$, des tenseurs optimisables comme dans le 3DGS, ainsi qu'un contexte local $f_v \in \mathbb{R}^{32}$. La scène est d'abord voxelisée à partir du nuage de points dense obtenu via COLMAP, chaque voxel $v \in \mathbf{V}$ servant de point d'ancrage avec sa propre caractéristique contextuelle, facteur de mise à l'échelle $l_v \in \mathbb{R}^3$, et ses offsets $\mathcal{O}_v \in \mathbb{R}^{k \times 3}$. Cette grille de voxels (svo) et les tenseurs associés sont complétés lors de l'observation de nouvelles positions du bras (ces tenseurs seront dupliqués dans un travail futur afin d'obtenir une banque d'informations pour chaque configuration possible du bras).

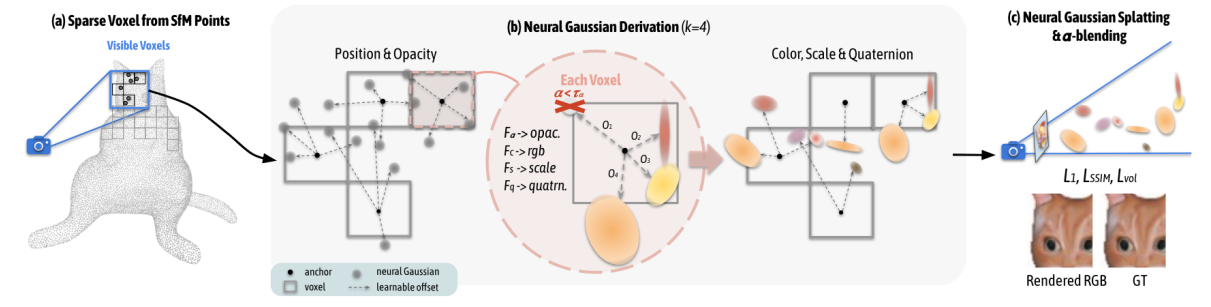


FIGURE 47 – Résumé de Scaffold-GS

Pour améliorer f_v en fonction de la vue et de la résolution, nous construisons une banque de caractéristiques à

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 42

plusieurs résolutions, $\{f_v, f_{v\downarrow 1}, f_{v\downarrow 2}\}$, où $f_{v\downarrow n}$ correspond à un sous-échantillonnage de f_v par un facteur 2^n . Ceci permet une généralisation de la méthode dans le but de capturer des détails à différents niveaux de granularité (figure 48). Ensuite, cette banque est pondérée avec des coefficients de poids prédits par un petit MLP F_w .

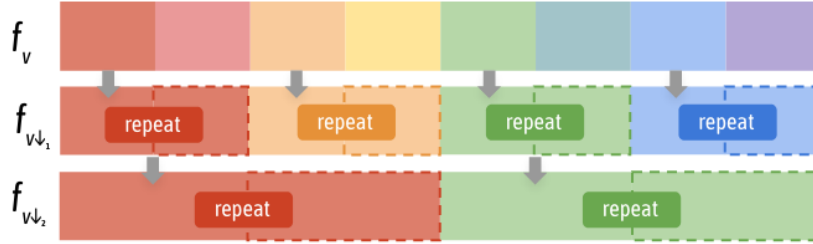


FIGURE 48 – Banque de caractéristiques utilisée

Étant donné une caméra en position $\mathbf{x}_c$ et un point d'ancrage en position $\mathbf{x}_v$, nous calculons leur distance relative δ_{vc} et leur direction de vue $\mathbf{d}_{vc}$ avec :

$$\delta_{vc} = \|\mathbf{x}_v - \mathbf{x}_c\|_2, \quad \mathbf{d}_{vc} = \frac{\mathbf{x}_v - \mathbf{x}_c}{\|\mathbf{x}_v - \mathbf{x}_c\|_2},$$

ensuite, une somme pondérée de la banque de caractéristiques est effectuée avec les poids prédits par le MLP F_w :

$$\{w, w_1, w_2\} = \text{Softmax}(F_w(\delta_{vc}, \mathbf{d}_{vc})),$$

ce qui nous permet d'obtenir la caractéristique d'ancrage intégrée $\hat{f}_v$:

$$\hat{f}_v = w \cdot f_v + w_1 \cdot f_{v\downarrow 1} + w_2 \cdot f_{v\downarrow 2}.$$

Les k gaussiennes neurales associées au point d'ancrage de position $\mathbf{x}_v$ auront pour positions (μ_i) :

$$\{\mu_0, \mu_1, \dots, \mu_{k-1}\} = \mathbf{x}_v + \{\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_{k-1}\} \cdot \mathbf{I}_v \quad (36)$$

$\mathbf{I}_v$ étant un facteur d'échelle. Leurs attributs $a \in \{\alpha, c, q, s\}$ seront directement décodés lors de l'inférence par leur MLP respectifs (voir figure 49). Chaque MLP est doté de fonctions d'activation spécifiques à leur donnée de sortie. Par exemple, la sortie pour l'opacité est activée par une fonction Tanh, pour la couleur par une fonction Sigmoid, et pour l'amplitude par une combinaison de Sigmoid et d'un facteur d'échelle de base. Cette configuration permet de contraindre les sorties dans des intervalles pertinents, garantissant ainsi des valeurs valides pour les différents attributs ($\alpha \in [0, 1)$, $c \in [0, 1]$).

$$a = F_a(\hat{f}_v, \delta_{vc}, \mathbf{d}_{vc}) \quad (37)$$

Une fonction d'activation sigmoïde est appliquée pour contraindre α dans l'intervalle $[0, 1)$ et obtenir des gradients lisses, et une fonction d'activation exponentielle pour l'amplitude de la covariance. La matrice de

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 43

covariance initiale est estimée comme une gaussienne isotrope avec des axes égaux à la moyenne de la distance aux trois points les plus proches.

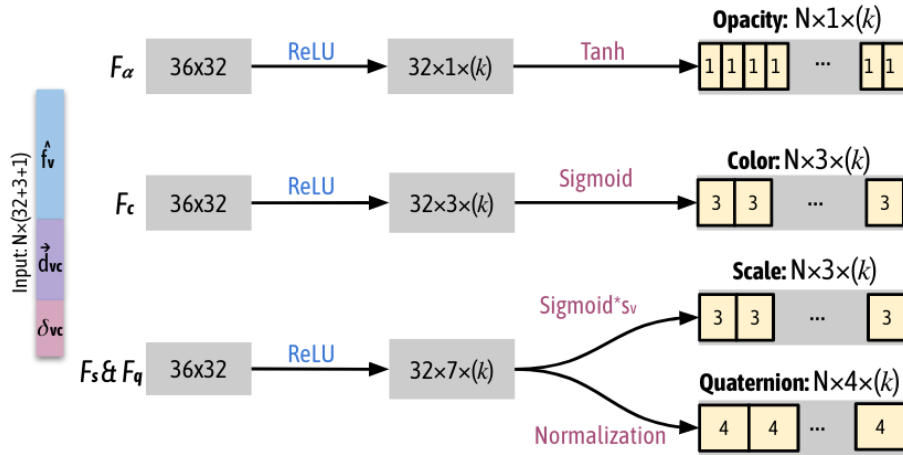


FIGURE 49 – Structure des 3 MLP

L'intégration d'un terme de régularisation volumique dans la fonction de coût favorise la réduction de la taille des gaussiennes tout en minimisant leur chevauchement :

$$\mathcal{L}_{vol} = \sum_{i=1}^{N_{ng}} Prod(s_i) \quad (38)$$

où N_{ng} représente le nombre de gaussiennes neurales dans la scène et $Prod(s_i)$ le produit des valeurs du vecteur contenant les échelles selon les trois axes.

5.5.2 Adaptation des Translations

Le centre des gaussiennes neurales devant être calculés à partir des points d'ancrage $x_v \in \mathbb{R}^3$ et de leurs offsets $\mathcal{O}_i \in \mathbb{R}^3$, une modification de l'algorithme de rotation a dû être mise en place :

$$\{\mathbf{G}'_v\} = \mathbf{R}(\mathbf{x}_v - \mathbf{P}) + \mathbf{R}(\{\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_{k-1}\} - \mathbf{P}) \cdot \mathbf{I}_v \quad (39)$$

où $\mathbf{P} \in \mathbb{R}^3$ représente les coordonnées du pivot et $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ la matrice de rotation associée à θ .

5.5.3 Augmentation de la dimension d'entrée

L'ajout d'un degré de liberté supplémentaire a nécessité la modification des MLP responsables du décodage des attributs des gaussiennes. En particulier, les MLP qui gèrent les paramètres dépendant de l'angle (notamment la couleur) ont vu leur entrée passer à une taille de (38×32) , voire (56×32) .

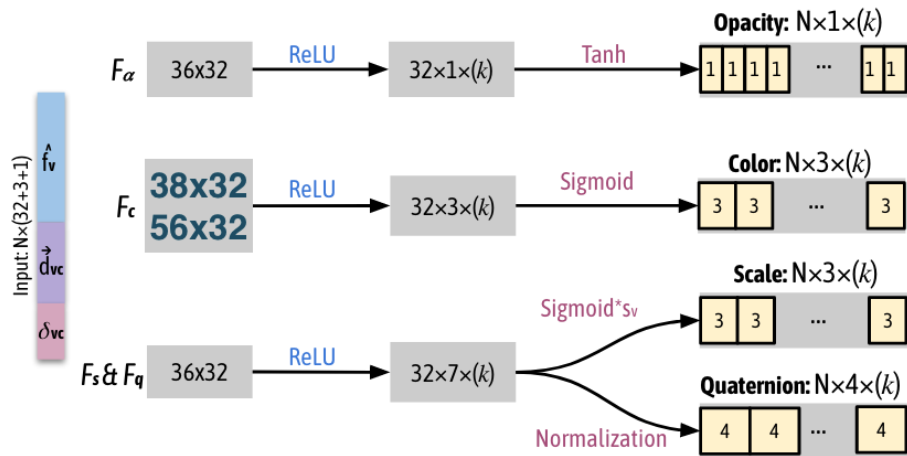


FIGURE 50 – MLP modifié

La donnée angulaire en entrée est enrichie conformément aux conventions standard présentées dans [15], notamment via un encodage positionnel, comme expliqué à la section 10. La figure 50 illustre la structure des MLP avec un mappage avec $L = 1 : \gamma(\theta) = (\sin(\theta), \cos(\theta))$, et $L = 10$.

Cette augmentation de la dimension d’entrée a conduit à un apprentissage plus performant, se traduisant par des scores plus élevés en SSIM et PSNR, bien que des anomalies en bordure de scène soient encore présentes (figures 51 et 52). Le temps moyen d’entraînement est passé d’une demi-heure à 3h.

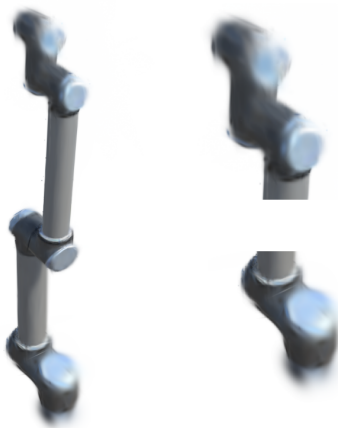


FIGURE 51 – Anomalies en bordure (2.25°)

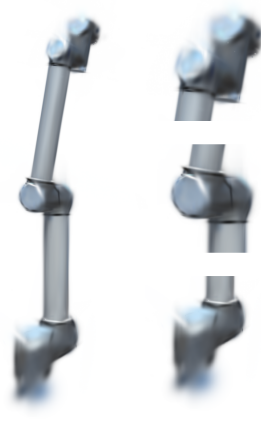


FIGURE 52 – Anomalies en bordure (11.25°)

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 45

5.5.4 Ajout d'une Loss VGG

Pour améliorer la capture des détails fins de la scène, un terme de régularisation basé sur la Loss VGG a été intégré à la fonction de coût. Initialement introduite dans [9], elle vise à maximiser la similarité perceptuelle entre les images générées et les images de référence. Contrairement aux pertes calculées pixel par pixel, la VGG Loss évalue les différences en s'appuyant sur les représentations internes d'un réseau de neurones convolutif pré-entraîné, en particulier le réseau VGG16.

Concrètement, cette perte est calculée en mesurant la distance euclidienne entre les cartes de caractéristiques extraites des images générées et de référence à différents niveaux de convolution du réseau VGG16. Soit $\phi_{i,j}$ la carte de caractéristiques obtenue après la j -ème convolution, et avant la i -ème couche de maxpooling dans le réseau VGG16, nous définissons la VGG Loss comme suit :

$$l_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

où $W_{i,j}$ et $H_{i,j}$ représentent les dimensions de la carte de caractéristiques à la couche i,j , I^{HR} désigne l'image de référence, et $G_{\theta_G}(I^{LR})$ l'image générée par le modèle.

L'intégration de la loss VGG dans notre processus d'apprentissage permet d'améliorer la reconstruction des détails fins, réduisant les artefacts visuels et conduisant à une meilleure représentation des structures complexes dans la scène. cette augmentation de la qualité visuelle est néanmoins quantitativement faible (augmentation de 1.8 du PSNR).

En raison de la nature de la scène synthétique, caractérisée par un fond blanc, les fonctions de coût ont été ajustées pour éviter que le modèle ne privilégie la reconstruction d'une scène entièrement blanche, dû à des différences trop faibles entre les ground-truth et les rendus (96% de chaque image étant composé de blanc). L'introduction de masques binaires segmentant l'objet du fond a permis de mieux contrôler l'apprentissage, en particulier pour les détails fins qui étaient auparavant difficiles à représenter avec précision. Une nouvelle approche, inspirée des techniques présentées dans [2], est en cours de développement. Cette méthode vise à adapter l'entraînement au contexte N-dimensionnel, avec pour objectif principal de faciliter la modification de l'éclairage dans les rendus 3DGS, problème phare présent dans nos implémentations.

6 Annexes

6.1 Automatisation des Procédés

Dans le cadre de ce projet, destiné à être appliqué sur des données réelles, l'acquisition automatique des poses, la segmentation en différentes parties de l'objet, ainsi que la génération des points de pivot sont devenues des étapes indispensables. Ces tâches ne peuvent plus être réalisées manuellement à cause du volume de données (séries d'images ou vidéos) et du besoin d'une précision constante. Cette section détaille les processus automatisés mis en place pour obtenir une segmentation fine, comparable à celle illustrée dans la figure 57.

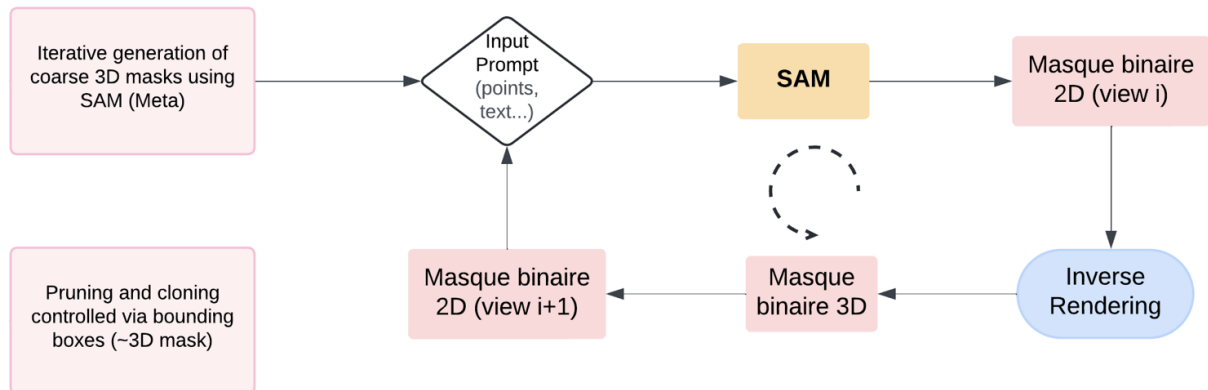


FIGURE 53 – Génération Automatique de Masques 3D

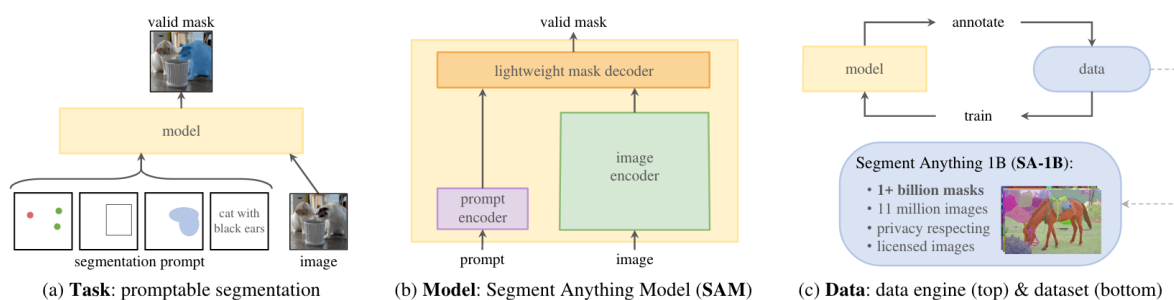


FIGURE 54 – Segment Anything Model (SAM)

Des avancées significatives ont été faites dans ce domaine, particulièrement dans le contexte du retrait et de l'ajout d'objets dans des scènes générées par 3DGS. Des articles récents, tels que Segment Anything in 3D with Radiance Fields [1] ou SAGD [8], proposent des modèles innovants. Cependant, ces modèles sont principalement conçus pour segmenter des objets dans leur totalité, sans se concentrer sur la division d'un objet en ses composants individuels. C'est pourquoi notre attention s'est portée sur le modèle Segment Anything (SAM [11]) développé par META (figure 54), qui sert de fondation à plusieurs de ces méthodes.

SAM permet de générer plusieurs masques valides pour un même objet, chacun associé à un score de confiance. Cette fonctionnalité est cruciale pour notre projet, car elle nous permet de segmenter les différentes parties du bras robotique UR10, facilitant ainsi l'application de transformations spécifiques, comme la rotation d'une partie du bras (figure 56).

En associant des points de référence aux images (comme les étoiles vertes dans la figure 57) et en excluant les éléments qui ne doivent pas être segmentés, nous avons pu obtenir des résultats avec des scores élevés pour les masques générés. Cette précision est indispensable pour garantir que les transformations appliquées aux différentes parties du robot respectent les contraintes géométriques et cinématiques du modèle.

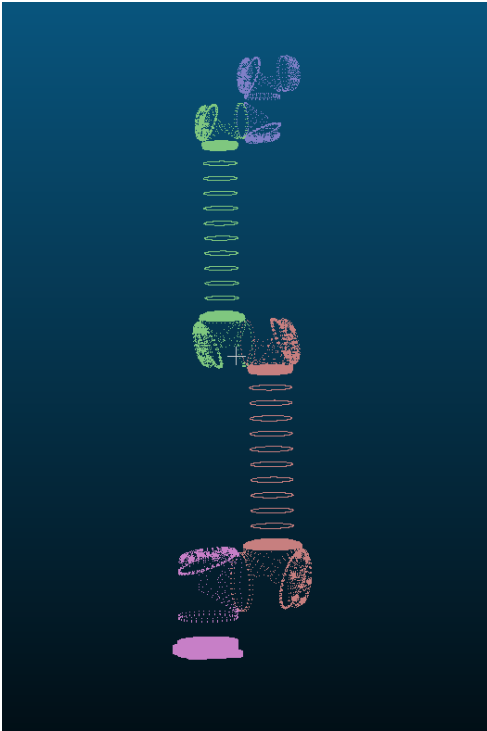


FIGURE 55 – Segmentation de l’UR10



Figure 3: Each column shows 3 valid masks generated by SAM from a single ambiguous point prompt (green circle).

FIGURE 56 – Exemple de Segmentation

Bien que ce travail puisse être étendu à travers l'inverse rendering en tirant parti des techniques de ray tracing et de l'inversabilité du renderer du 3D Gaussian Splatting, cette approche n'a pas été développée ici. Toutefois, l'application de ces méthodes a été explorée dans le cadre du 4D Gaussian Splatting, en utilisant le modèle SAM2, récemment développé par META, qui est capable de segmenter et suivre des parties spécifiques d'un objet, comme l'UR, tout au long d'une vidéo captée en conditions réelles.

6.2 Définitions

6.2.1 Alpha Blending

L'alpha blending est une technique de rendu graphique utilisée pour combiner la couleur d'un pixel d'un objet avec la couleur du pixel de fond, en tenant compte de la transparence (alpha) des pixels. L'équation mathématique caractéristique de l'alpha blending est donnée par :

C_{final} = \alpha C_t + (1 - \alpha)C_b

(40)

où C_{final} est la couleur résultante, C_t est la couleur de l'objet au premier plan, C_b est la couleur du fond, et \alpha est le coefficient de transparence compris entre 0 (transparent) et 1 (opaque) (de l'objet qui est le plus proche

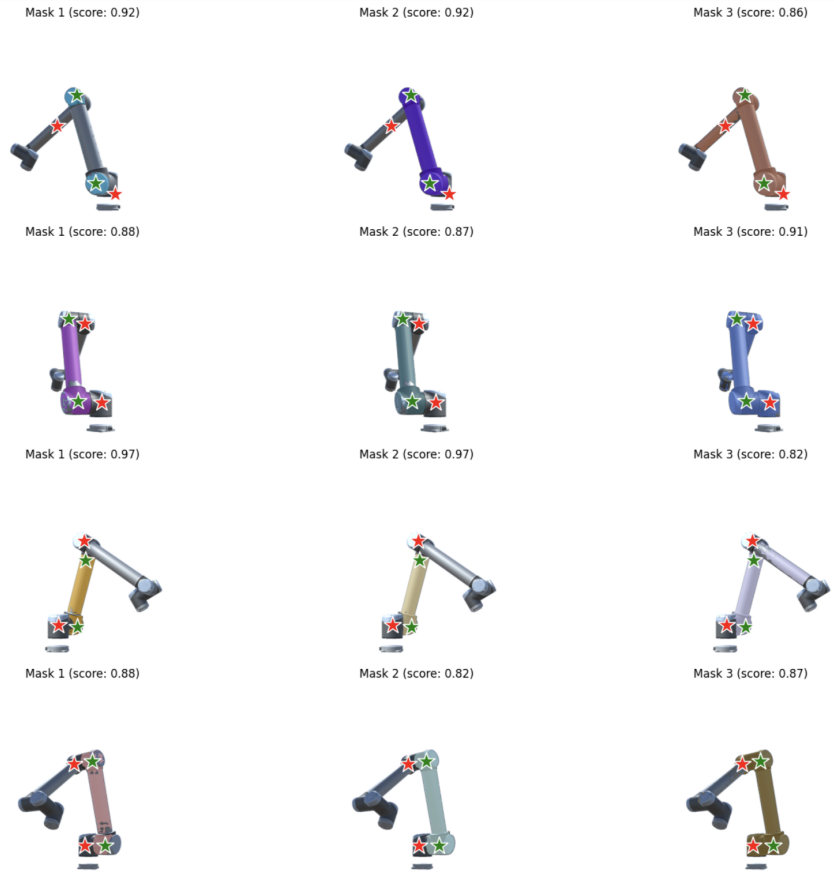


FIGURE 57 – Segmentation de l'UR 10

de la caméra le long de la trajectoire du rayon).

6.2.2 Recalage d'Images

Le recalage d'images est le processus d'alignement de deux ou plusieurs images afin de les superposer de manière cohérente. Cela implique de trouver une transformation géométrique qui minimise les différences entre les images. Mathématiquement, cela peut être formulé comme l'optimisation de la fonction de coût $\mathcal{L}(T)$ où T est la transformation appliquée à une image pour qu'elle corresponde à une autre :

$$\mathcal{L}(T) = \sum_{x \in \Omega} |I_1(T(x)) - I_2(x)| \tag{41}$$

où I_1 et I_2 sont les deux images à aligner, $T(x)$ représente la transformation appliquée aux coordonnées des pixels x , et Ω est le domaine de l'image.

6.3 Méthodes Mathématiques

Extraction des caractéristiques et Similarité Visuelle Les deux images sont encodées, ce qui permet de mapper les images d'entrée vers des cartes de caractéristiques denses, à une résolution réduite (1/8 de la résolution d'origine, voir explication en annexe). L'encodeur de caractéristiques utilise des blocs résiduels pour capturer des détails pertinents. Pour calculer la similarité visuelle entre les images, un volume de corrélation $C(g_\theta(I_1), g_\theta(I_2)) \in \mathbb{R}^{H \times W \times H \times W}$ complet est formé :

$$C_{ijkl} = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh} \quad (42)$$

À partir de ce volume, les trajectoires des points visibles sur la première image sont suivies de manière récurrente jusqu'à ce qu'elles soient occultées, ce qui est vérifié par une vérification de la cohérence du flux optique avant-arrière.

6.3.1 RAFT

Recurrent All-Pairs Field Transforms (RAFT [20]) est un modèle de deep learning conçu pour l'estimation du flux optique. RAFT extrait des caractéristiques à l'échelle des pixels, construit des volumes de corrélation 4D multi-échelle pour toutes les paires de pixels, et met à jour itérativement un champ de flux à travers une unité récurrente. La Figure 58 donne un aperçu de l'architecture de RAFT, qui combine un encodeur de caractéristiques et un bloc de mise à jour pour fournir des prédictions de flux précises à chaque itération.

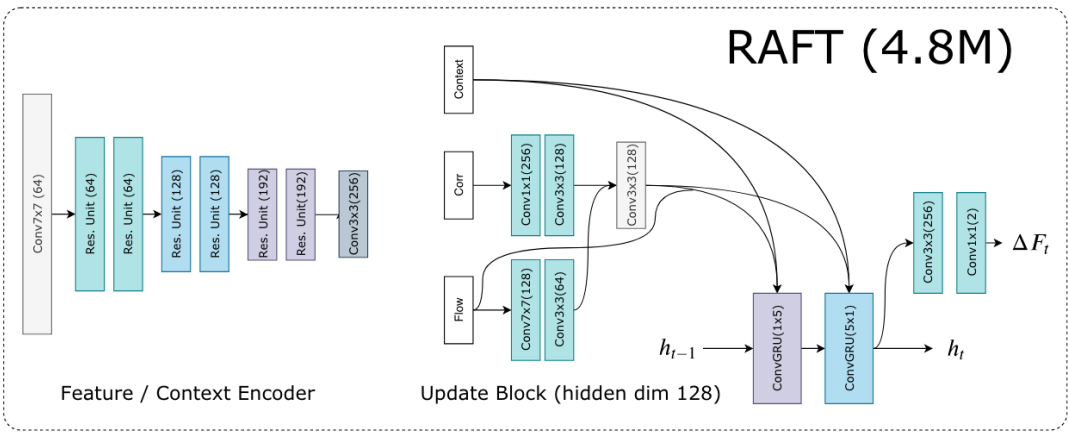


FIGURE 58 – Aperçu du modèle Recurrent All-Pairs Field Transforms

6.3.2 OANet

OANet (Order-Aware Network [24]) est un modèle de deep learning conçu pour établir des correspondances précises entre des points caractéristiques dans deux images, cruciales pour des tâches comme la reconstruction 3D et la pose relative. Le modèle introduit plusieurs opérations innovantes pour capturer à la fois le contexte spatial local et global des correspondances. OANet regroupe les correspondances non ordonnées en clusters hiérarchiques via une couche de pooling différentiable (DiffPool) et applique des blocs de filtrage

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 50

"order-aware" pour mieux modéliser le contexte global. Ces clusters sont ensuite reconstitués à leur taille d'origine par une couche d'unpooling différentiable, permettant d'améliorer la précision des correspondances et des estimations de pose. La figure 59 illustre ces étapes, mettant en avant les blocs de filtrage, le processus de clustering, et l'unpooling différentiable pour capturer efficacement les relations géométriques entre les correspondances.

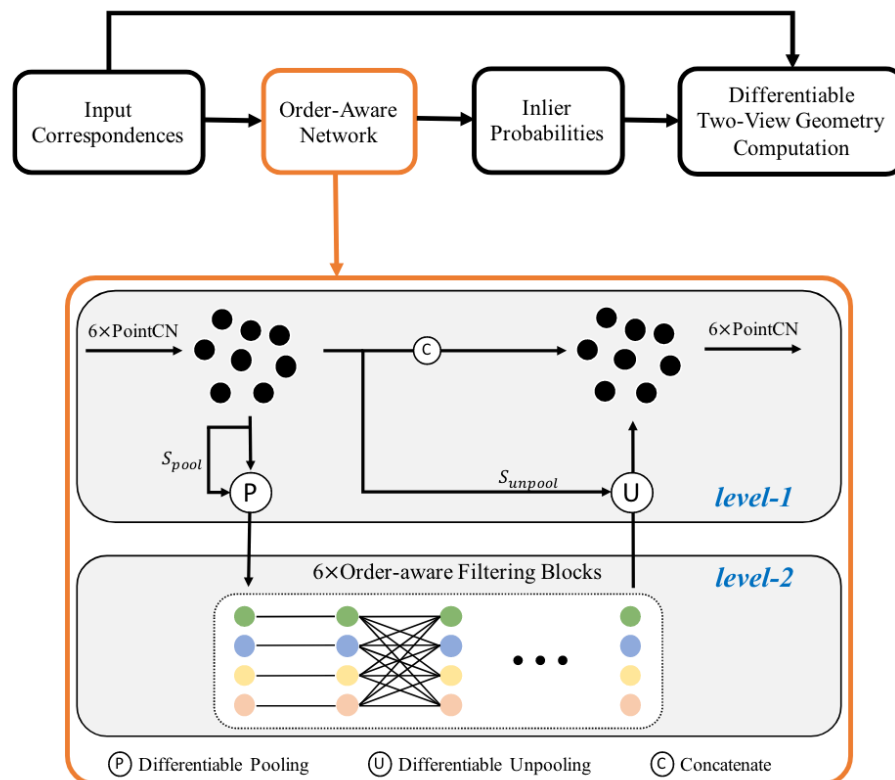


FIGURE 59 – Aperçu du modèle OANet

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 51

6.4 Images

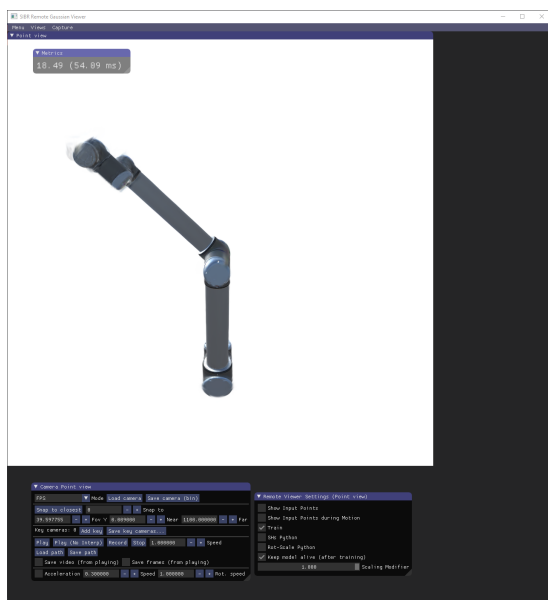


FIGURE 60 – Sans transformations des SH

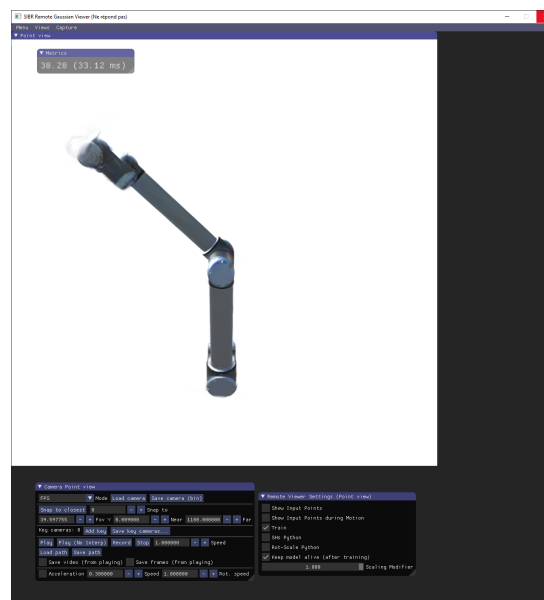


FIGURE 61 – Avec transformation des SH

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 52

7 Bibliographie

Références

- [1] Jiazhong Cen, Jiemin Fang, Zanwei Zhou, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment Anything in 3D with Radiance Fields.
- [2] Stavros Diolatzis, Tobias Zirr, Alexandr Kuznetsov, Georgios Kopanas, and Anton Kaplanyan. N-Dimensional Gaussians for Fitting of High Dimensional Functions.
- [3] Gérard Dreyfus, editor. *Réseaux de neurones : méthodologie et applications ; [prévision, data mining, bio-ingénierie, reconnaissance de formes, robotique et commande de processus]*. Algorithmes. Eyrolles, 2. éd edition.
- [4] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4D Gaussian Splatting : Towards Efficient Novel View Synthesis for Dynamic Scenes.
- [5] Gökçen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J. Theis. Deep learning : New computational modelling techniques for genomics. 20(7) :389–403.
- [6] prefix=van de useprefix=true family=Wiele, given=J. Rotations et Moments Angulaires en Mécanique Quantique - Rotations and angular moments in quantum mechanics. (6).
- [7] Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wenchao Ma, Le Chen, Danhang Tang, and Ulrich Neumann. GaussianFlow : Splatting Gaussian Dynamics for 4D Content Creation.
- [8] Xu Hu, Yuxi Wang, Lue Fan, Junsong Fan, Junran Peng, Zhen Lei, Qing Li, and Zhaoxiang Zhang. SAGD : Boundary-Enhanced Segment Anything in 3D Gaussian via Gaussian Decomposition.
- [9] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution.
- [10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. 42(4) :1–14.
- [11] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything.
- [12] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-Flow : 4D Reconstruction with Dynamic 3D Gaussian Particle.
- [13] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS : Structured 3D Gaussians for View-Adaptive Rendering.
- [14] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF : Representing Scenes as Neural Radiance Fields for View Synthesis.
- [15] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks.
- [16] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation : Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.
- [17] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

CEA - LIST		DIASI/SIR/24-049
DIASI/SIR/LSI		Rév. 0 Page 53

- [18] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.
- [19] Jie Shen, Jie Xu, and Pingwen Zhang. Approximations on $SO(3)$ by Wigner D-matrix and applications.
- [20] Zachary Teed and Jia Deng. RAFT : Recurrent All-Pairs Field Transforms for Optical Flow.
- [21] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms : Theory and Practice*, volume 1883, pages 298–372. Springer Berlin Heidelberg.
- [22] E. Walter and L. Pronzato. *Identification de modèles paramétriques à partir de données expérimentales*. Number ISSN 1246-8193 in MASC Modélisation Analyse Simulation Commande (FRA). Masson.
- [23] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting.
- [24] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning Two-View Correspondences and Geometry Using Order-Aware Network.
- [25] Wang Zhao, Shaohui Liu, Hengkai Guo, Wenping Wang, and Yong-Jin Liu. ParticleSfM : Exploiting Dense Point Trajectories for Localizing Moving Cameras in the Wild.