Coloriage d'un nuage de points 3D à l'aide d'une caméra pour de la cartographie

Abdallah Samir
Telecom Nancy - Promotion 2025
samir.abdallah@telecomnancy.eu

Boualit Yamine
Telecom Nancy - Promotion 2025
yamine.boualit@telecomnancy.eu

Encadrants :
Sylvain Contassot-Vivier
et Amaury Saint-Jore
LORIA
sylvain.contassotvivier@loria.fr
amaury.saint-jore@loria.fr

***Résumé*—Cet article présente le travail effectué sur la coloration d'un nuage de points en trois dimensions à l'aide d'une caméra et d'images en deux dimensions.
L'objectif initial était de générer en temps réel des nuages de points à partir d'un LIDAR 3D et du middleware ROS 2, puis de les colorer. Ce document explique en détail les différentes méthodes employées, telles que l'analyse des paramètres d'une caméra, l'utilisation d'un z-buffer et les divers algorithmes utilisés pour la coloration et la visualisation des nuages de points. En particulier, l'article décrit le processus de calibration de la caméra, essentiel pour obtenir une correspondance précise entre les points du nuage et les pixels des images 2D. Il explore également les défis techniques liés à l'alignement spatial des données provenant du LIDAR et de la caméra, et propose des solutions pour surmonter ces obstacles.**

I. Introduction

Les robots mobiles, tels que Spot de Boston Dynamics, se révèlent capables d'accomplir des missions d'exploration dans divers environnements. Ces missions visent principalement à cartographier des espaces, notamment ceux difficilement accessibles par l'Homme. Doté de multiples capteurs, le robot est capable de naviguer de manière autonome. En combinant ces capteurs à d'autres systèmes, comme le système NAPS pour une géolocalisation locale, le robot peut évoluer dans des environnements sans GPS. Le système NAPS est combiné à un LIDAR 3D, qui capture un nuage de points représentant l'environnement.

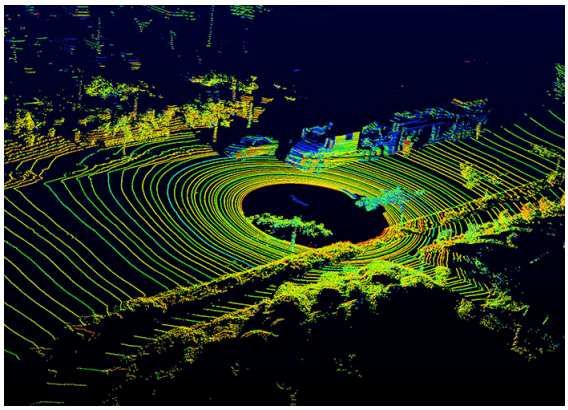

Figure 1. Exemple de nuage de points capturé avec un LIDAR 3D

C'est à ce stade que notre recherche intervient. En ajoutant un système de capture d'images, tel qu'une caméra, il devient possible de colorer ce nuage de points. L'objectif est que le robot, équipé à la fois d'un LIDAR et d'une caméra, puisse fournir en temps réel une représentation en trois dimensions et en couleur de son environnement. Cette approche permettrait une cartographie précise de certaines zones, offrant des informations détaillées sur l'environnement.

Notre travail s'est concentré sur le développement d'algorithmes permettant, à partir d'un ensemble d'images, de coloriser une vidéo. Nous avons principalement testé nos algorithmes sur des jeux de données d'essai, plutôt que sur des captures en temps réel.

Cet article est structuré en trois parties principales. La première revient sur l'état de l'art des techniques existantes et des écrits théoriques sur le sujet. Ensuite, la partie méthodologie explique les différentes techniques, algorithmes et technologies utilisés pour atteindre notre objectif. Enfin, la dernière partie présente les résultats obtenus sur divers jeux de données, ainsi que nos conclusions tirées de ces résultats.

II. État de l'art

Les nuages de points constituent un ensemble de points en trois dimensions, chaque point étant défini par ses coordonnées X, Y et Z dans un repère spatial. La coloration d'un nuage de points repose sur la capacité à identifier précisément ces points et à leur attribuer une couleur correspondante. Lors de cette attribution, il est crucial de vérifier l'exactitude de la coloration afin d'éviter toute erreur.
Le processus de coloration peut être réalisé de différentes manières, chacune présentant ses propres avantages, inconvénients et degrés de facilité d'implémentation. La précision de l'attribution des couleurs est un critère déterminant pour choisir la méthode la plus appropriée. Toutefois, il est également essentiel de considérer l'usage prévu de la coloration. Dans notre cas spécifique, où l'objectif est de réaliser une coloration en temps réel, il est impératif de prendre en compte ce besoin particulier lors de la sélection de la méthode de coloration. Cela signifie que la méthode choisie doit non seulement minimiser les erreurs d'attribution, mais

aussi être suffisamment rapide et efficace pour fonctionner en temps réel.

Nous allons voir ici quatre techniques existantes pour réaliser la coloration en se basant sur les articles [1], [2], [3] et [4].

A. Coloration par label

Dans un premier temps, nous allons aborder la technique de labélisation des objets contenus dans un nuage de points. Cette technique repose sur la reconnaissance et l'identification d'objets spécifiques dans le nuage de points, leur attribuant ainsi une couleur distinctive. Pour cela, nous nous appuyons sur une base de données contenant de nombreux objets et leurs couleurs respectives. Dans le cas d'un nuage de points unique, la reconnaissance peut être réalisée manuellement par un opérateur humain, qui attribuera une couleur à chaque objet identifié. Cependant, ce processus est souvent fastidieux et inefficace, surtout lorsque le nombre d'objets est important. Pour pallier ces limitations, il est nécessaire d'adopter une approche basée sur l'intelligence artificielle. L'une des méthodes les plus prometteuses est l'utilisation d'un perceptron multicouches, un type de réseau de neurones artificiels. Ce réseau reçoit en entrée un objet non coloré extrait du nuage de points. À travers plusieurs couches de neurones et des calculs pondérés, le réseau de neurones transforme cet objet en une version colorée.

Le succès de cette méthode dépend en grande partie de

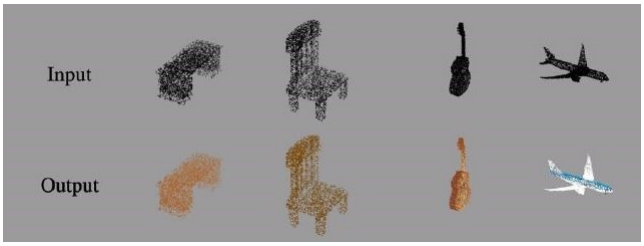

Figure 2. Résulat du perceptron multicouches sur plusieurs objets (source [1])

la performance du modèle de réseau de neurones. Pour garantir des résultats optimaux, il est crucial de disposer d'une base de données d'entraînement suffisamment vaste et diversifiée. Cette base de données doit contenir un large éventail d'objets et leurs couleurs associées, permettant au modèle de généraliser efficacement à de nouveaux objets.

Il est également important de prendre en compte les défis inhérents à l'apprentissage supervisé. Par exemple, le modèle peut être sujet au surapprentissage, où il mémorise les détails spécifiques des objets d'entraînement sans apprendre à généraliser à de nouveaux objets. De plus, des erreurs d'interprétation peuvent survenir, notamment lorsque les objets dans le nuage de points présentent des variations subtiles que le modèle a du mal à reconnaître correctement.

En résumé, la technique de labélisation des objets dans un nuage de points repose sur une combinaison de reconnaissance manuelle et de techniques d'intelligence artificielle. L'utilisation de réseaux de neurones, comme le perceptron multicouches, permet d'automatiser et d'accélérer ce processus, bien que cela nécessite une base de données robuste et une gestion attentive des défis liés à l'apprentissage supervisé. [1]

B. Photogrammétrie

La méthode de photogrammétrie offre une approche différente pour résoudre le problème. Ici, le processus s'inverse : plutôt que de partir d'un nuage de points pour attribuer des couleurs, on utilise des images en deux dimensions pour générer le nuage de points. Ce dernier est synthétisé à partir des informations contenues dans les images, sans recourir à d'autres données.

Ce nuage de points nouvellement créé peut être directement coloré puisqu'il est issu d'une image couleur. Les reliefs et les profondeurs de cette image sont convertis en points tridimensionnels. Le principe s'inspire de la vision binoculaire humaine, qui permet de percevoir la profondeur à partir de deux points dont la position relative est connue. Ainsi, en superposant des séquences d'images 2D, il devient possible d'estimer des modèles 3D.

Un aspect essentiel de cette méthode réside dans la projection des points 2D d'une image en 3D, en supposant que les paramètres de la caméra sont connus. Le principe est que pour chaque point, on essaie de le retrouver dans différents points de vue. Ainsi, d'une image à l'autre, on va repérer le même point et en connaissant la position des points de vue, on en déduit la profondeur.[2]

La photogrammétrie est largement utilisée pour cartographier de vastes espaces extérieurs, en particulier à partir d'images satellites. Bien que ces nuages de points puissent manquer de précision, ils permettent toutefois de repérer les formes essentielles du relief. Un exemple emblématique de l'utilisation de la photogrammétrie est l'application Google Earth, qui offre une visualisation 3D des espaces à partir d'images satellites 2D.

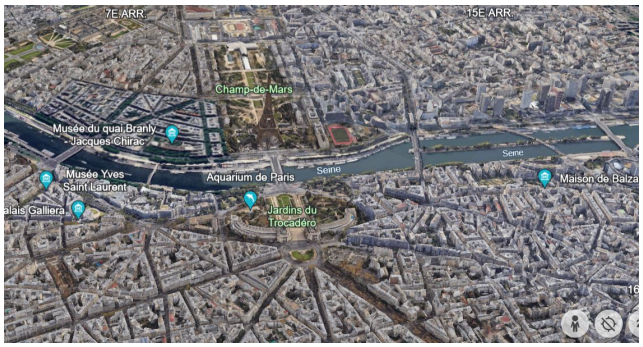

Figure 3. Utilisation de photogrammétrie dans Google Earth

Malgré son intérêt et sa fonctionnalité, cette méthode ne répond pas parfaitement à notre objectif de colorer un nuage

de points en temps réel avec une précision suffisante. Par conséquent, elle ne correspond pas pleinement à nos besoins.

C. Réseau antagoniste génératif conditionnel (cGAN)

L'approche par réseau antagoniste génératif conditionnel, ou cGAN, présente des similitudes avec la méthode de labélisation précédemment évoquée. Tout comme dans la méthode de labélisation, l'objectif est d'utiliser un réseau de neurones pour prédire la couleur d'un objet en 3D. Cependant, cette fois, le réseau de neurones adopte une structure cGAN.

Le fonctionnement d'un réseau GAN (Generative Adversarial Network) est le suivant : deux systèmes sont mis en compétition. D'abord, le générateur crée de fausses images, visant à les rendre les plus réalistes possibles pour tromper le discriminateur, qui, lui, évalue la vraisemblance des images qu'il reçoit, provenant à la fois du générateur et d'une base de données réelle. Son rôle est de distinguer les images réelles des images générées par le générateur. Une fois le modèle entraîné, il est capable de labéliser de nouvelles images comme étant vraies ou fausses.

Cependant, dans notre contexte, deux sorties ne sont pas suffisantes, d'où la nécessité d'utiliser un cGAN (Conditional Generative Adversarial Network). Avec un cGAN, des informations plus précises, appelées labels de classe, sont transmises au générateur et au discriminateur pour orienter leur production de données. Ces labels permettent de préciser les données générées par le générateur et évaluées par le discriminateur, accélérant ainsi le processus pour obtenir le résultat désiré. Les labels guident la production du générateur pour qu'il génère des informations plus précises. Ainsi, des points colorés, vrais ou faux, sont générés et le discriminateur apprend à partir de ces données.

Le système calcule des erreurs pour s'améliorer, notamment des erreurs de prédiction de couleur tel que :
$L^{point} = ||C_{fake} - C_{real}||$
mais aussi des erreurs d'image en comparant les valeurs entre l'image générée et l'image réelle :
$L^{image} = ||I_{fake} - I_{real}||$
Ces métriques permettent d'évaluer la qualité des prédictions et d'ajuster les paramètres du réseau pour obtenir des résultats plus précis.
Ci-dessous on peut voir un exemple d'un jeu d'images colorées que le discriminateur va devoir labéliser et ainsi mieux prédire les sorties pour chaque point 3D.[3]

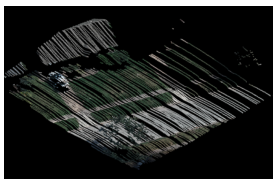

Figure 4. Image réelle

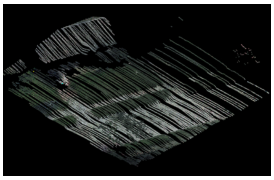

Figure 5. Image fausse générée (source [3])

D. Coloration à partir de données photographiques

La dernière solution que nous avons répertoriée est la technique de coloration à partir d'une caméra. Le principe consiste à utiliser un ensemble de données photographiques conjointement avec une capture d'un LIDAR. Cette méthode permet d'identifier, à partir de l'image 2D, la couleur correspondante des différents points du nuage 3D sans calculs importants.

Plusieurs défis se posent pour appliquer cette technique. Tout d'abord, il est essentiel de synchroniser les données photographiques avec le nuage de points. Le LIDAR et la caméra utilisée doivent avoir des horloges synchronisées, surtout si la coloration s'effectue sur une séquence continue, telle qu'une vidéo. Sans cette synchronisation, les correspondances entre les points du nuage et les pixels de l'image risquent d'être incorrectes, compromettant ainsi la précision de la coloration.

Ensuite, la coordination des points de vue est cruciale. Étant donné que le LIDAR et la caméra ne sont pas nécessairement positionnés au même endroit, il est nécessaire de s'assurer que le point à colorer dans le nuage de points est visible dans l'image. Inversement, il faut éviter de colorer des points qui ne sont pas capturés par l'image. Cela nécessite une compréhension précise des angles de vue et des positions relatives du LIDAR et de la caméra.

Pour réaliser cette coordination, il est indispensable de connaître les différents paramètres de la caméra, appelés paramètres intrinsèques et extrinsèques. Les paramètres intrinsèques comprennent les caractéristiques internes de la caméra, telles que la distance focale et le point principal, tandis que les paramètres extrinsèques décrivent la position et l'orientation de la caméra dans l'espace. Ces informations permettent de définir le repère de la caméra et de réaliser les conversions nécessaires pour localiser correctement les points du nuage dans l'image. On retrouve par exemple dans une matrice instrinsèque, les informations liées à la focale f_i et au positionnement du centre C_i.[4]

$$[K] = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 6. Matrice intrinsèque d'une caméra

En maîtrisant ces paramètres, il devient possible de superposer les données du LIDAR et les images de la caméra de manière précise, assurant ainsi une coloration fidèle des points du nuage 3D. Cette technique offre une solution prometteuse pour obtenir des nuages de points colorés plus particulièrement en temps réel. C'est pourquoi c'est cette méthode que nous avons choisi d'implémenter et d'approfondir.

III. Méthodologie

A. Modélisation du problème

Dans le cadre de la coloration par données photographiques, le problème posé peut se formuler ainsi : pour chaque point du nuage de points, déterminer le point/pixel de l'image sur lequel il est projeté. En d'autres termes, l'objectif est de définir les caractéristiques de la projection des points du repère lidar vers le repère de l'image. Pour résoudre ce problème de projection des points du nuage de points vers les pixels de l'image, le modèle de caméra à sténopé (*pinhole camera model*) est couramment utilisé. Ce modèle simplifie le processus de projection en supposant une caméra où la lumière passe par un unique trou, le diaphragme, pour former une image inversée sur un plan image. Dans ce modèle, les coordonnées 3D du point P (dans le repère du lidar) sont d'abord converties en coordonnées de la caméra, puis projetées sur le plan image en utilisant la formule de projection perspective.[7]

Nous nous intéressons d'abord au calcul de la projection d'un point $P = [x, y, z]^T$ dans le repère (à 3 dimensions) de la caméra vers le repère image de la manière suivante :

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{f \cdot x}{z} \\ \frac{f \cdot y}{z} \end{bmatrix}$$

Il est cependant important de noter que cette transformation se situe uniquement dans le plan image et non dans le repère de l'image numérique, d'où la nécessité d'ajouter d'autres paramètres. Il faut d'abord effectuer une translation pour tenir compte du fait que le plan image a pour origine le centre de celui-ci, tandis que le plan de l'image numérique a généralement pour origine le pixel situé en haut à gauche de l'image. On introduit donc les paramètres $c_x = \frac{\text{largeur de l'image}}{2}$ et $c_y = \frac{\text{hauteur de l'image}}{2}$. Il est également nécessaire de prendre en compte la conversion des coordonnées de mètres en pixels en introduisant les paramètres k et l qui correspondent à la conversion des deux axes du plan image.
Ainsi, nous obtenons :

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{f \cdot l \cdot x}{z} + c_x \\ \frac{f \cdot k \cdot y}{z} + c_y \end{bmatrix} = \begin{bmatrix} \frac{\alpha \cdot x}{z} + c_x \\ \frac{\beta \cdot y}{z} + c_y \end{bmatrix}$$

Il serait cependant préférable de mettre ces calculs sous forme linéaire afin de disposer dans la suite d'outils propres à l'algèbre linéaire. Pour cela, nous pouvons introduire les coordonnées homogènes en ajoutant une coordonnée supplémentaire à chaque repère :

$$P'_h = [x'_w, y'_w, w]^T \quad \text{et} \quad P_h = [x, y, z, 1]^T$$

La transformation devient donc :

$$P'_h = \begin{bmatrix} \alpha x + c_x z \\ \beta y + c_y z \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_x & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P_h$$

Nous pouvons décomposer cette transformation de la manière suivante :

$$P'_h = K \begin{bmatrix} I & 0 \end{bmatrix} P$$

Avec :

$$K = \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_x \\ 0 & 0 & 1 \end{bmatrix}$$

K correspond à ce que l'on appelle les paramètres intrinsèques de la caméra.

Il est maintenant nécessaire d'effectuer un changement de coordonnées, du repère du lidar vers celui de la caméra, modélisé par la matrice suivante :

$$\begin{bmatrix} R & T \\ 1 & 0 \end{bmatrix}$$

où R est une matrice de rotation et T un vecteur de translation, simplifié en $[R\ T]$. La transformation complète s'exprime donc de la manière suivante :

$$P'_h = K \begin{bmatrix} R & T \end{bmatrix} P_h = M P_h$$

Nous nommerons M la matrice de projection.

B. Calcul des paramètres

Nous cherchons à présent à calculer les coefficients de M. Pour cela, nous avons besoin de n points de référence dont nous connaissons les coordonnées $P^{(i)}$ dans le monde réel et les coordonnées (homogènes) $p^{(i)}$ dans l'image numérique. Sachant que M possède 12 coefficients, il nous faut donc au minimum 6 points de référence. Nous disposons des égalités suivantes pour $i \in [\![1 : n]\!]$:

$$p_w^{(i)} = \begin{bmatrix} x_w^{(i)} \\ y_w^{(i)} \\ w \end{bmatrix} = M P^{(i)} = \begin{bmatrix} m_1 \cdot P^{(i)} \\ m_2 \cdot P^{(i)} \\ m_3 \cdot P^{(i)} \end{bmatrix}$$

avec m_k la k-ième ligne de M, et donc :

$$p^{(i)} = \begin{bmatrix} \frac{x_w^{(i)}}{w^{(i)}} \\ \frac{y_w^{(i)}}{w^{(i)}} \end{bmatrix} = \begin{bmatrix} \frac{m_1 P^{(i)}}{m_3 P^{(i)}} \\ \frac{m_2 P^{(i)}}{m_3 P^{(i)}} \end{bmatrix}$$

Nous obtenons ainsi le système d'équations linéaires suivant :

$$x^{(i)}(m_3 P^{(i)}) - m_1 P^{(i)} = 0$$

$$y^{(i)}(m_3 P^{(i)}) - m_2 P^{(i)} = 0$$

ce qui correspond à l'équation matricielle suivante :

$$\begin{bmatrix} (P^{(1)})^T & 0 & -x^{(1)}(P^{(1)})^T \\ 0 & (P^{(1)})^T & -y^{(1)}(P^{(1)})^T \\ & \vdots & \\ (P^{(n)})^T & 0 & -x^{(n)}(P^{(n)})^T \\ 0 & (P^{(n)})^T & -y^{(n)}(P^{(n)})^T \end{bmatrix} \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} = Pm = 0$$

Que l'on peut résoudre par la méthode des moindres carrés, en minimisant $\|Pm\|^2$ avec la contrainte $\|m\|^2 = 1$.

C. L'algorithme de coloriage

Après avoir déterminé les coefficients de la matrice de projection, on peut mettre en place un premier algorithme pour colorier un nuage de points :

```
pour i de 0 à n-1 :
 # projection du point dans le repère
 # de l'image en coordonées homogènes
 x, y, w = matrice * nuage[i]
 x /= w
 y /= w
 # le point est-il devant la caméra ?
 si w > 0 alors :
  couleur[i] = image[floor(y)][floor(x)]
 fin si
fin pour
```

D. Z-Buffer

Bien que l'algorithme présenté précédemment permette un coloriage correct, il présente un défaut : des points qui se trouvent derrière d'autres points, et donc qui ne sont pas réellement visibles par la caméra, sont tout de même coloriés. Pour pallier ce problème, on peut introduire la structure suivante : le Z-buffer. Le Z-buffer est une matrice dont la hauteur et la largeur sont égales à celles de l'image, et dont chaque élément correspond la distance minimale pour chaque pixel de l'image aux objets vus par la caméra. Ainsi, un point n'est colorié que s'il peut être inscrit dans le Z-buffer.

On construit le Z-buffer par l'algorithme suivant :

```
# Initialisation du Z-buffer
pour i de 1 à h:
  pour j de 1 à l:
    zbuffer[i][j] = null

# Itération sur les points
pour i de 1 à n:
  x, y, w = matrice * nuage[i]
  x /= w
  y /= w
  k = floor(y)
  l = floor(x)
  prev = zbuffer[k][l]
  si prev == null:
    zbuffer[k][l] = i
  sinon:
    x_prev, y_prev, w_prev
        = matrice * nuage[prev]
    # Comparaison de la distance
    # à la caméra
    si w_prev > w:
      zbuffer[k][l] = i
```

Et l'algorithme de coloriage devient alors :

```
pour i de 1 à h:
  pour j de 1 à l:
    p_index = zbuffer[i][j]
    si p_index != null:
      couleur[p_index] = image[i][j]
```

IV. Résultats

La coloration du nuage s'est donc faite à partir de nos algorithmes et nous avons pu obtenir des résultats sur différents nuages de points.

Pour évaluer l'efficacité de nos méthodes, il était essentiel de disposer de jeux de données appropriés. Cependant, nous ne pouvions pas effectuer de tests avec une capture en temps réel d'un environnement. Par conséquent, nous avons entrepris une recherche minutieuse pour identifier des jeux de données existants qui pouvaient répondre à nos besoins.

Nous avons donc cherché et sélectionné des jeux de données disponibles publiquement, incluant des données 3D issues de LIDAR ainsi que des photographies correspondantes. Les données 3D générées par le LIDAR fournissent des informations précises sur la structure et la topographie de l'environnement scanné. En parallèle, les photographies offrent une représentation visuelle et colorimétrique de l'environnement, nécessaire pour appliquer la coloration aux points du nuage.

L'accès à ces jeux de données a été crucial pour tester et valider nos algorithmes. Cela nous a permis d'effectuer des comparaisons entre les résultats attendus et ceux obtenus, et d'apporter les ajustements nécessaires pour améliorer la précision et l'efficacité de nos méthodes.

Cette partie du document va donc se concentrer sur la présentation des résultats obtenus à partir des deux jeux de données que nous avons sélectionnés. Nous détaillerons les caractéristiques de chaque jeu de données, les défis spécifiques rencontrés lors de leur utilisation, ainsi que les performances de nos algorithmes sur ces jeux de données. Les résultats seront analysés en termes de précision de la coloration, de fidélité aux données originales.

Enfin, nous discuterons des implications de nos trouvailles et des possibilités d'amélioration future de nos méthodes.

A. KITTI dataset

Le KITTI dataset est une base de données de référence dans le domaine de la vision par ordinateur et des véhicules autonomes. Collecté par l'Institut de Technologie de Karlsruhe en Allemagne, ce dataset est conçu pour aider à la recherche et au développement de technologies avancées de perception et de navigation. Ce dataset se distingue par sa diversité et sa richesse, comprenant plusieurs types de données essentiels pour les applications de conduite autonome. Les images, capturées par des caméras montées sur un véhicule, couvrent une variété de scénarios urbains et ruraux. En complément, des scans LiDAR 3D haute résolution fournissent une représentation détaillée de l'environnement. De plus, des données GPS/IMU offrent des informations précises sur la localisation et les mouvements du véhicule.

Dans notre cas, nous avons principalement utilisé les données LIDAR 3D, fournies sous forme de fichiers où chaque point

du nuage de points est représenté par un triplet (x, y, z). En complément, nous avons également récupéré les fichiers photos associés à chaque nuage de points. La capture des données du KITTI dataset est effectuée en temps réel ; ainsi, pour chaque instant enregistré, la base de données contient un nuage de points et des photographies. La voiture utilisée pour la collecte de données est équipée de quatre caméras couleur comme l'on peut voir sur la Figure 7, ce qui permet de couvrir à 360 degrés l'ensemble des points du nuage.[5]

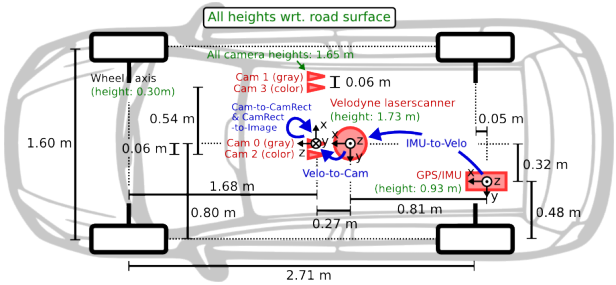

Figure 7. Disposition du matériel sur la voiture KITTI (source [5])

Nous avons appliqué cette coloration à une série de photos consécutives, ce qui nous a permis de créer une vidéo montrant un nuage de points colorisé de manière dynamique.

Figure 8. Prise de vue par la caméra

Figure 9. Nuage de points coloré

On peut observer sur la Figure 9 un léger décalage sur certaines formes de l'image, notamment certains points d'objets à l'avant qui se retrouvent derrière. Cela peut probablement s'expliquer par une synchronisation imparfaite entre la caméra et le LIDAR, entraînant un décalage entre l'image et le nuage de points correspondant. Une solution pourrait être de prendre en compte la position de la voiture pour déterminer plus précisément la correspondance entre les images et les nuages de points. D'autres raisons pourraient expliquer ce décalage, et une étude ainsi que des tests supplémentaires seraient nécessaires pour résoudre ce problème.

B. Backpack Dataset

Pour pouvoir approfondir les résultats et tester nos algorithmes, d'autres données d'essai nous étaient nécessaires. Pour ce faire, notre encadrant nous a fourni deux nuages de points. Dans cet essai, les conditions sont différentes que dans le premier. Ici, chaque nuage de points est accompagné d'une série de photos prises par un appareil en rotation autour de l'objet. Il était donc nécessaire de contrôler quels points étaient observables par quelle caméra. Les résultats obtenus sont les suivants:

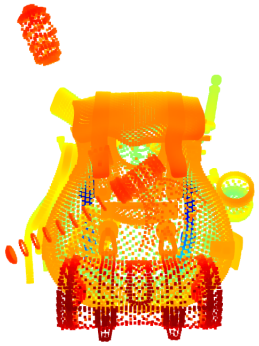

Figure 10. Nuage de points sans couleur d'un sac à dos

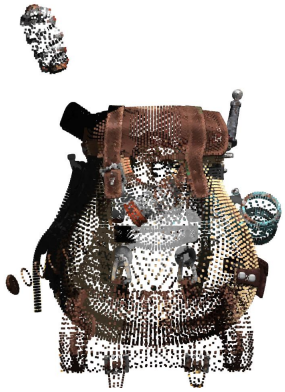

Figure 11. Nuage de points après colorisation

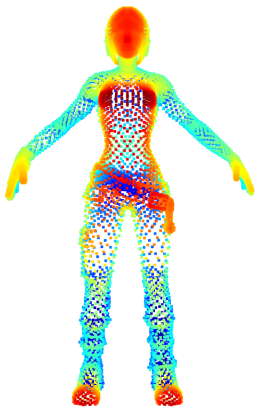

Figure 12. Nuage de points d'un personnage sans couleur

Figure 13. Nuage de points après colorisation

Les résultats obtenus ici en Figure 11 et Figure 13 sont plutôt satisfaisants étant donné que l'on reconnaît les différentes parties des images de base. Cependant, on peut relever un problème, des parties avant de l'image se retrouvent projetées à l'arrière de l'objet comme on peut l'observer Figure 14 où le visage du personnage se retrouve aussi à l'arrière de la tête. Cela s'explique par le choix de la méthode où la caméra choisie pour la coloration est la caméra la plus proche du point. Or, les points de vue ne sont pas à équidistance ainsi, on a cette erreur. Une autre explication est qu'étant donné que le nuage de points n'est pas assez dense, certains point qui devraient être obstrués ne le sont pas, plusieurs pistes ont été évoqués pour résoudre ce problème.

Une solution serait de prendre en compte la normal pour chaque point et ainsi ne considérer pour chaque caméra seulement les points dont la normale est opposée à la caméra donc négative. Cette solution serait facile à exécuter, mais le

Figure 14. Problème de projection

problème est le calcul de la normale. Calculer la normale au point dans un nuage 3D est relativement complexe et nécessite une recherche plus approfondie que nous n'avons pas pu réaliser.
Une autre solution dont nous avons pu discuter, est l'application d'un rayon d'opacité pour chaque point du Z-buffer pour pouvoir déterminer les points qui en cachent d'autres. Pour chaque point nous déterminons une sorte de cône qui permet de déterminer les points cachés par ce cône sans impacter les points voisins qui sont sur la même surface, ce qui serait le cas avec un simple cercle d'opacité.

V. Conclusion

Ce projet nous a donc permis d'explorer diverses méthodes pour la coloration d'un nuage de points 3D. À travers l'étude des paramètres des caméras ainsi que les calculs nécessaires, nous avons obtenu des résultats relativement satisfaisants.

La coloration d'un nuage de points implique d'identifier les points correspondants dans l'image 2D et, via des matrices de projection, de déterminer leur équivalent dans la projection 3D. Les calculs requis pour cette tâche sont assez rapides, avec des temps de traitement de l'ordre de la seconde pour le dataset Backpack et d'une dizaine de secondes pour le dataset KITTI. Ces temps montrent que l'algorithme est efficace et permet de réaliser la coloration de plusieurs nuages de points consécutivement.

Néanmoins, plusieurs aspects doivent encore être améliorés, notamment les problèmes de projection évoqués dans la section IV. Ces problèmes nécessitent encore du temps de recherche et d'approfondissement, que nous n'avons pas pu allouer.

Pour aller plus loin, il est pertinent de mentionner le Gaussian Splatting 3D, une méthode avancée de visualisation des nuages 3D. Comme nous l'avons vu précédemment, les différents nuages de points sur lesquels nous avons travaillé

ne permettent d'obtenir qu'un aperçu de l'environnement avec des écarts entre les points, résultant en un nuage globalement creux. Le Gaussian Splatting associe à chaque point une surface étendue, calculée grâce à une gaussienne. Cela permet de combler les creux du nuage et d'obtenir une représentation colorée plus réaliste de l'espace.[6]

Notre travail peut servir de base pour alimenter un algorithme de calcul de Gaussian Splatting et ainsi produire des scans précis en temps réel.

RÉFÉRENCES

[1] X Cao and K Nagao, *Point cloud colorization based on densely annotated 3d shape dataset*, 2019

[2] T Shinohara, H Xiu and M Matsuoka, *Point2color: 3D Point Cloud Colorization Using a Conditional Generative Network and Differentiable Rendering for Airborne LiDAR*, 2021

[3] Jakob Iglhaut, Carlos Cabo, Stefano Puliti, Livia Piermattei, James O'Connor and Jacqueline Rosette, *Structure from Motion Photogrammetry in Forestry: a Review*, 2019

[4] P. Vechersky, M Cox, P Borges and T Lowe, *Colourising Point Clouds using Independent Cameras*, 2018

[5] A Geiger, P Lenz, C Stiller and R Urtasun, *Vision meets robotics: The KITTI dataset*, 2013

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler and George Drettakis, *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, 2023

[7] K. Hata, S. Savarese, *CS231A Course Notes 1: Camera Models*, 2017