Estimation de profondeur monoculaire par fine-tuning LoRA

Application à l'inspection de pneus industriels (données Zivid)

Abdelali Chikhi - Ayman Zejli - Mouad Azennag - Loic Magnan | IMDS 5A

12 janvier 2026

Résumé

Ce projet vise à estimer une carte de profondeur à partir d'une image RGB dans un contexte industriel d'empilement de pneus. Nous utilisons Depth Anything (Vision Transformer) comme modèle de base, puis nous l'adaptons à notre domaine via un fine-tuning LoRA (Low-Rank Adaptation). La vérité terrain provient de fichiers `.npy` contenant des nuages de points (XYZ) issus d'un capteur 3D Zivid, dont on extrait le canal Z. Malgré un jeu de données très réduit (58 paires), nous obtenons une amélioration nette des métriques (RMSE, $\delta_1/\delta_2/\delta_3$) et des résultats qualitatifs cohérents (formes et contours des pneus correctement reconstruits). Le rapport détaille les choix techniques, les différentes approches testées, ainsi que les difficultés rencontrées et leurs corrections.

Table des matières

1 Introduction

L'estimation de profondeur monoculaire consiste à prédire, pour chaque pixel d'une image, une distance relative (ou métrique) entre la caméra et la scène. Ce problème est fondamental en vision (robotique, perception, inspection), mais il est intrinsèquement ambigu : plusieurs scènes 3D peuvent produire une image 2D similaire.

Les modèles récents comme Depth Anything exploitent de larges pré-entraînements sur des données variées. Néanmoins, un modèle générique s'adapte difficilement à un environnement industriel spécifique : objets répétitifs, surfaces brillantes, occlusions, et bruit capteur dans certaines zones. Dans ce projet, nous adaptons Depth Anything au domaine *pneus industriels* avec une technique de fine-tuning légère (LoRA), adaptée aux petits jeux de données.

2 Objectifs et cahier des charges

2.1 Objectif principal

Prédire une carte de profondeur à partir d'images RGB de pneus empilés, afin de se rapprocher au mieux de la profondeur Z mesurée par le capteur 3D (Zivid).

2.2 Contraintes

— **Dataset très petit** : seulement 58 paires (risque d'overfitting).

— **Scène industrielle difficile** : surfaces brillantes, zones sombres, occlusions.

— **Profondeur partiellement invalide** : pixels NaN / vides dans les fichiers `.npy`.

— **Ressources limitées** : besoin d'une solution efficace en paramètres (LoRA) et en mémoire (accumulation de gradients).

3 Jeu de données (Zivid) et split

3.1 Format des données

Le dataset contient :

— des images RGB (format `.png`)

— des fichiers `.npy` contenant le nuage de points de taille $(H, W, 3)$: canaux (X, Y, Z)

La vérité terrain de profondeur est obtenue par extraction du canal Z :

$$Z_{\mathrm{GT}}(u, v) = \texttt{npy}[u, v, 2]$$

Lorsque le fichier est déjà en 2D, on considère que c'est directement une profondeur.

3.2 Split entraînement / validation

Nous utilisons un split 80/20 :

— 46 échantillons pour l'entraînement

— 12 échantillons pour la validation

Le split est reproductible via un `random_state` fixe.

4 Difficultés du problème (pneus industriels)

Les pneus empilés posent plusieurs difficultés importantes pour la profondeur monoculaire :

— **Surfaces brillantes / reflets** : les reflets spéculaires perturbent la relation apparence → profondeur.

— **Occlusions** : parties cachées entre pneus, zones de contact, trous et cavités.

— **Répétitivité** : textures et formes similaires à plusieurs profondeurs différentes (ambiguïtés).

— **Bruit capteur** : dans certaines zones (ombre, réflexion), le nuage de points contient des valeurs invalides.

Ces points expliquent la nécessité d'un masque de validité et d'une perte qui favorise les contours.

5 Approches testées et choix final

5.1 Approche A : profondeur linéaire (métrique) vs inverse-depth

Nous avons testé deux représentations de la profondeur :

— **Profondeur linéaire** : normalisation directe de Z entre un minimum et un maximum.

— **Inverse depth** : $d = 1/Z$, puis normalisation.

Constat Depth Anything est pré-entraîné avec une représentation proche de l'inverse depth. En pratique, l'entraînement en profondeur linéaire pouvait réduire certains termes du loss mais menait parfois à une instabilité des métriques (problèmes d'échelle). L'inverse depth s'est révélée plus stable et plus compatible avec les poids pré-entraînés.

Choix final Nous retenons **inverse depth + normalisation**, car c'est la configuration la plus stable en entraînement et la plus cohérente avec Depth Anything.

5.2 Approche B : data augmentation

Avec seulement 58 images, l'augmentation est un levier important contre l'overfitting. Nous avons appliqué uniquement des augmentations **géométriquement cohérentes** entre RGB et profondeur :

— flips horizontaux / verticaux (appliqués à l'image et au depth)

— *optionnel* : brightness/contrast uniquement sur RGB (pas sur depth)

Attention Les rotations/crops agressifs peuvent casser la correspondance géométrique et n'ont pas été retenus.

5.3 Approche C : gestion des pixels invalides (valid mask)

Les fichiers `.npy` contiennent parfois des NaN ou des valeurs invalides. Nous construisons un masque :

$$M(u,v) = \begin{cases} 1, & \text{si } 0 < Z(u,v) < 10000 \text{ et } Z(u,v) \text{ n'est pas NaN} \\ 0, & \text{sinon} \end{cases}$$

Ce masque est essentiel : sans lui, le modèle apprend sur du bruit, ce qui dégrade la convergence.

5.4 Approche D : résolution et alignement des dimensions

Nous avons rencontré des erreurs de dimensions entre :

— la profondeur GT (résolution d'origine)

— la prédiction du modèle (résolution interne du réseau)

— le masque (parfois dans une autre résolution)

Correction Nous imposons une taille cible unique pour **RGB, GT depth, mask** : 756×1260 (multiple de 14, adapté au ViT). Ensuite, pendant l'évaluation, nous **redimensionnons** la prédiction à la résolution GT via interpolation bicubique.

6 Prétraitement (pipeline final)

6.1 Extraction du canal Z

On extrait Z depuis le fichier `.npy` :

$$Z(u,v) = \begin{cases} \texttt{npy}[u,v,2] & \text{si } \texttt{npy} \in \mathbb{R}^{H\times W\times 3} \\ \texttt{npy}[u,v] & \text{si } \texttt{npy} \in \mathbb{R}^{H\times W} \end{cases}$$

6.2 Redimensionnement

Images, profondeur et masque sont redimensionnés à 756×1260 pour assurer l'alignement.

6.3 Normalisation en inverse depth

On convertit :

$$d = \frac{1}{Z+\epsilon}$$

Puis on normalise entre 0 et 1 :

$$d_{\text{norm}} = \frac{d - d_{\min}}{d_{\max} - d_{\min} + \epsilon}$$

avec $d_{\min} = 1/Z_{\max}$ et $d_{\max} = 1/Z_{\min}$.

6.4 Masque de validité

Le masque M est utilisé à l'entraînement et à l'évaluation pour ignorer les pixels invalides.

7 Modèle : Depth Anything + LoRA

7.1 Depth Anything

Depth Anything est un modèle basé sur un Vision Transformer (ViT) pré-entraîné sur des données massives, capable de prédire une profondeur dense.

7.2 Pourquoi LoRA ?

Un fine-tuning complet du modèle risquerait fortement l'overfitting avec 58 images. LoRA ajoute des matrices de faible rang dans certaines couches (souvent attention) :

$$W \leftarrow W + \Delta W, \quad \Delta W = BA$$

où A et B sont de rang faible. Cela réduit drastiquement le nombre de paramètres entraînables tout en permettant l'adaptation au domaine.

7.3 Avantages LoRA pour ce projet

— Adaptation efficace avec peu de données
— Moins de mémoire GPU/CPU et entraînement plus stable
— Conservation des capacités générales du modèle pré-entraîné

8 Fonction de perte (loss) et justification

Notre loss combine :

— **L1 loss** : précision globale en profondeur
— **Gradient loss** : netteté des contours (bords de pneus)

8.1 Perte L1

$$L_1 = \frac{1}{|M|} \sum_{(u,v)\in M} \left| \hat{d}(u,v) - d(u,v) \right|$$

8.2 Perte sur gradients

On calcule des gradients discrets :

$$\partial_x d(u,v) = d(u,v+1) - d(u,v), \quad \partial_y d(u,v) = d(u+1,v) - d(u,v)$$

et on pénalise l'écart entre gradients prédits et GT. Cette composante améliore la reconstruction des bords.

8.3 Perte totale

$$\mathcal{L} = L_1 + 3 \cdot L_{\text{grad}}$$

Le coefficient 3 a été choisi pour donner plus de poids aux contours, importants pour les pneus.

9 Protocole d'entraînement

9.1 Paramètres principaux

— 15 époques
— batch size = 1 (images haute résolution)
— accumulation de gradients = 8 (batch effectif ≈ 8)
— précision mixte (fp16) lorsque disponible

9.2 Stabilité et problèmes rencontrés

Durant le développement, plusieurs erreurs ont été rencontrées puis corrigées :

— **Incohérence de clés** (`depth_gt` vs `labels`) dans le data collator ⇒ correction des clés.
— **Mismatch de résolutions** (mask/labels/pred) ⇒ uniformisation des tailles et interpolation bicubique.

10 Métriques d'évaluation

Nous utilisons des métriques classiques en depth estimation :

— **RMSE**

— **AbsRel** : erreur relative moyenne

— $\delta_1, \delta_2, \delta_3$: proportion de pixels avec un ratio d'erreur inférieur à 1.25^k

Remarque importante Dans notre implémentation, les métriques ont été calculées sur une profondeur normalisée (inverse depth normalisée). Cela reste pertinent pour comparer les époques (tendance, convergence). Pour une interprétation métrique stricte (mm/m), une dé-normalisation en Z peut être utilisée lors de la visualisation.

11 Résultats quantitatifs

Nous rapportons les résultats sur le jeu de validation (12 images). Les performances s'améliorent régulièrement et atteignent un plateau stable.

Pourquoi la loss de validation baisse peu ? La loss de validation reste relativement stable pour deux raisons principales. Premièrement, la fonction de perte utilisée n'est pas une simple perte L_1 : elle inclut un terme sur les gradients (L_{grad}) destiné à préserver les discontinuités (bords des pneus). Ce terme est plus sensible au bruit local et génère naturellement une courbe moins "lisse", surtout sur un petit ensemble de validation. Deuxièmement, la validation ne contient que 12 images : la variance statistique est donc élevée, et de petites différences de scènes (reflets, occlusions) peuvent faire fluctuer la loss d'une époque à l'autre.

Pourquoi AbsRel peut sembler "élevée" ? La métrique AbsRel est ici calculée sur une profondeur *normalisée* (inverse depth normalisée), et non directement sur la profondeur métrique Z (en mm/m). Dans ce cadre, la valeur absolue d'AbsRel n'est pas directement comparable aux seuils classiques rapportés dans la littérature (souvent calculés sur Z). Ce qui est significatif, en revanche, est la tendance : AbsRel chute fortement entre l'époque 1 et 5 (de 3.12 à ≈ 1.08) puis se stabilise, tandis que RMSE diminue et que les métriques $\delta_1, \delta_2, \delta_3$ augmentent régulièrement. Cela indique une amélioration réelle de la cohérence géométrique malgré la normalisation.

Epoch	ValLoss	RMSE	LogRMSE	AbsRel	δ_1	δ_2	δ_3
1	1.8637	0.5916	0.4243	3.1284	0.2107	0.3914	0.5011
5	1.8407	0.4614	0.4223	1.0782	0.4053	0.5971	0.6713
10	1.8363	0.4408	0.4257	1.0821	0.4266	0.6233	0.6826
15	**1.8342**	**0.4211**	0.4300	1.1112	**0.4518**	**0.6334**	**0.6872**

TABLE 1 – Évolution des performances sur le jeu de validation.

11.1 Analyse

Le RMSE diminue de façon significative (environ 29% entre l'époque 1 et 15). Les métriques δ augmentent progressivement, montrant que de plus en plus de pixels sont prédits dans des marges d'erreur acceptables. La loss de validation reste relativement stable car elle combine une composante sur gradients (plus bruitée) et l'échantillon de validation est petit.

12 Résultats qualitatifs (visualisations)

Cette partie présente des exemples visuels : RGB, GT depth, profondeur prédite, et carte d'erreur. Les figures permettent de confirmer que le modèle reconstruit correctement la géométrie globale des pneus, avec des erreurs concentrées dans les zones brillantes ou occluses.

12.1 Exemples de prédiction

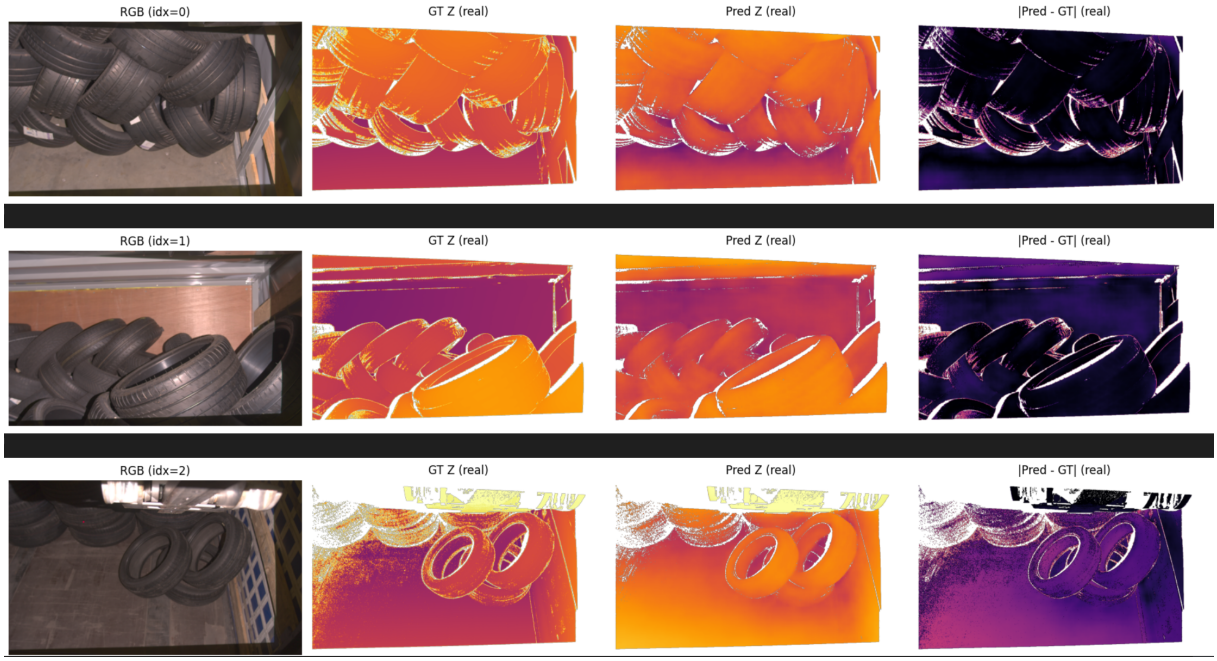

FIGURE 1 – Exemple 1 : (gauche à droite) image RGB, profondeur GT (Z ou inverse depth normalisée), profondeur prédite, erreur absolue.

12.2 Commentaires

Visuellement, la prédiction conserve la structure globale (empilement, formes circulaires) et les bords des pneus sont bien localisés, ce qui confirme l'intérêt de la perte sur gradients. Les erreurs augmentent dans les zones spéculaires et fortement occluses, ce qui correspond aux limitations attendues pour un modèle monoculaire.

13 Discussion : ce qui a marché, ce qui limite

13.1 Pourquoi la méthode fonctionne malgré 58 images ?

La raison principale est l'utilisation d'un modèle pré-entraîné puissant (Depth Anything) et d'une adaptation faible en paramètres (LoRA). Le modèle possède déjà des représentations générales de la géométrie ; LoRA permet d'ajuster ces représentations au domaine des pneus sans réapprendre depuis zéro.

13.2 Limites

— Le dataset est très petit, donc la généralisation reste limitée.

— Les surfaces brillantes et les occlusions créent des ambiguïtés intrinsèques.

— Les métriques sont calculées sur une profondeur normalisée : pour une métrologie stricte, il faudrait dé-normaliser en Z et valider en unités physiques.

13.3 Améliorations possibles

— Augmenter le dataset (plus de scènes, angles, conditions d'éclairage).

— Ajouter une calibration explicite (conversion vers mm) et évaluer en unités réelles.

— Tester une loss multi-échelle ou une pondération plus fine des zones de forte courbure.

— Tester des augmentations photométriques plus robustes (RGB seulement) et régularisations supplémentaires.

14 Conclusion

Ce projet démontre qu'un modèle de profondeur monoculaire pré-entraîné peut être efficacement adapté à un environnement industriel (pneus empilés) via LoRA, malgré un dataset très réduit. Les résultats quantitatifs montrent une amélioration progressive (baisse RMSE, hausse δ), et les résultats qualitatifs confirment la cohérence géométrique des prédictions. L'approche ouvre la voie à des applications d'inspection et de contrôle visuel à faible coût, tout en identifiant clairement les limites liées aux reflets, occlusions et à la disponibilité de données.