Synthèse de vues intermédiaires pour la navigation urbaine

Rapport de recherche basé sur le prototype Street View

Maxim Quénel Pierre Louis Brun Mame Alè Seye Denis Bereziuc

GitHub : https://github.com/Maxim-Quenel/Intermediate-View-Synthesis-for-Urban-Navigation
YouTube : Résultat final et comparaison avec Street View : https://youtu.be/juUFj2ofpQY
Démo d'utilisation : https://youtu.be/8IzJGnT3ex0

Janvier 2026

Abstract

La navigation immersive de type Street View repose encore majoritairement sur des transitions entre panoramas discrets, ce qui produit un "effet tunnel" et des déformations de parallaxe. Ce rapport présente un système complet de synthèse de vues intermédiaires, construit autour d'une estimation de profondeur monoculaire, d'un warping 3D différentiable (softmax splatting), d'une fusion bidirectionnelle et d'un inpainting pour les disocclusions. Nous analysons le projet en profondeur, détaillons l'acquisition des données Street View, les prétraitements, l'architecture des modèles (Depth Anything V2 et UNet entraîné sur Cityscapes Depth and Segmentation, scènes extérieures), les paramètres de génération et les métriques utilisées. Les résultats incluent des analyses quantitatives (fichiers `metrics_forward.json` et `metrics_ground_truth.json`) et qualitatives (captures d'écran, cartes de profondeur, galerie d'animation). Le rapport discute les limites, les leçons apprises et les pistes d'extension vers des modèles plus robustes aux baselines larges.

1 Introduction

La navigation visuelle urbaine a fortement progressé ces dernières années, mais les systèmes grand public conservent un paradigme de navigation par "sauts" entre panoramas. Cette discontinuité produit des artefacts perceptuels (distorsions, déformation des objets proches, manque de parallaxe) qui limitent la sensation de déplacement réel. L'objectif de ce projet est de proposer une synthèse de vues intermédiaires permettant un mouvement avant continu entre deux points géographiques, tout en conservant une complexité computationnelle compatible avec une exécution interactive.

Le prototype est construit en tant qu'application Flask, capable de : (i) collecter des images Street View alignées sur l'axe de la route via l'API Google, (ii) estimer la profondeur par modèles monoculaires, (iii) reprojeter la scène dans un nouvel espace caméra, (iv) fusionner des warps forward/backward, (v) combler les trous par inpainting, et (vi) exporter une séquence vidéo. Ce rapport détaille l'ensemble du pipeline, ainsi que ses performances et limites.

2 Problème et formulation

Soit deux images sources I_A et I_B associées à deux positions de caméra le long d'une route, séparées par une distance d estimée à partir des métadonnées GPS. L'objectif est de générer une suite d'images intermédiaires I_t pour $t \in [0, 1]$ qui simulent un mouvement de caméra avant continu. Contrairement à une interpolation 2D classique, la synthèse doit respecter la parallaxe 3D : les objets proches doivent se déplacer plus vite que l'arrière-plan.

Le problème peut être formulé comme une reprojection : estimer une profondeur D_A et D_B, convertir les pixels en points 3D, appliquer un déplacement virtuel de caméra, puis reprojeter sur le plan image. Le défi principal vient de la qualité de la profondeur (souvent relative) et des zones disoccluses, qui ne sont visibles dans aucune image source.

3 Travaux connexes

Le projet s'inscrit dans un écosystème dense de recherche mêlant estimation de profondeur, correspondance de caractéristiques et synthèse de nouvelles vues.

3.1 Estimation de profondeur et géométrie

L'estimation de profondeur monoculaire a évolué des architectures CNN classiques vers des modèles de fondation robustes. Si des approches supervisées restent pertinentes pour des domaines spécifiques, les modèles récents comme Depth Anything V2[1] offrent une généralisation impressionnante en zéro-shot. En parallèle, la récupération de la géométrie 3D s'appuie traditionnellement sur des descripteurs épars (SIFT[10]) ou appris (SuperPoint[12], XFeat[14]), filtrés par RANSAC[11] ou des matchers modernes comme LightGlue[13]. Plus récemment, des méthodes denses comme LoFTR[3], RoMa[4] ou l'approche 3D explicite MASt3R[2] permettent une reconstruction plus cohérente, même en présence de changements de perspective importants.

3.2 Flux optique et interpolation vidéo (VFI)

La synthèse de vues intermédiaires peut être abordée comme un problème d'interpolation de frames vidéo (VFI). Des modèles comme RIFE[19], IFRNet[20] ou FILM[21] excellent pour fluidifier des vidéos à faible mouvement. De même, l'estimation de flux optique dense via RAFT[15] ou SEA-RAFT[16] permet de modéliser le mouvement 2D. Cependant, ces méthodes purement 2D peinent souvent à maintenir la cohérence structurelle sur les larges baselines typiques de Street View (sauts de plusieurs mètres), justifiant l'usage de guidage par profondeur explicite.

3.3 Synthèse de vues et inpainting

Pour la reprojection, le softmax splatting[5] s'est imposé comme une méthode efficace pour gérer les conflits de profondeur de manière différentiable, surpassant les z-buffers rigides. Si les approches volumétriques comme NeRF (Zip-NeRF[6], S-NeRF[18]) ou le 3D Gaussian Splatting[7] offrent un réalisme photométrique supérieur, elles nécessitent souvent un entraînement par scène coûteux. Des travaux spécifiques comme *Infinite Nature*[22] ou le *Cross Attention Renderer*[17] proposent des alternatives hybrides pour la génération de trajectoires infinies. Enfin, la gestion des zones disoccluses (les "trous" créés par le mouvement) est critique. L'inpainting moderne à large masque, illustré par LaMa[8] ou MAT[9], est essentiel pour halluciner des textures plausibles dans ces zones invisibles des images sources.

Le projet choisit volontairement une approche géométrique légère, combinant profondeur monoculaire et splatting, plutôt qu'un rendu volumétrique lourd. Cette décision est cohérente avec la contrainte "Street View" où l'on dispose de peu d'images par segment (baselines larges, déplacements d'environ 10 m).

4 Méthodologie

4.1 Acquisition Street View

La collecte d'images est réalisée avec l'API Google Street View. La fonction `fetch_source_images` calcule l'orientation principale de la route, interroge la métadata pour obtenir l'ID du panorama le plus proche, puis avance mètre par mètre dans la direction du déplacement jusqu'à détecter un nouveau panorama. Les images sont téléchargées avec une taille 640.0000 px × 640.0000 px, un champ de vue de 90.0000°, un pitch nul et un heading aligné sur la route. Les métadonnées enregistrées incluent latitude, longitude, ID du panorama et distance réelle entre panoramas successifs (formule de Haversine).

Cette acquisition garantit un flux unidirectionnel, similaire à une caméra frontale, afin de réduire la complexité géométrique des panoramas sphériques. Les captures d'écran de l'interface (Figure 1 et Figure 2) illustrent la sélection d'adresse et la présentation des images sources.

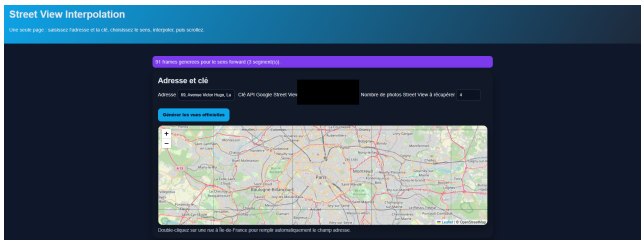


Figure 1: Interface de sélection d'adresse et carte interactive.

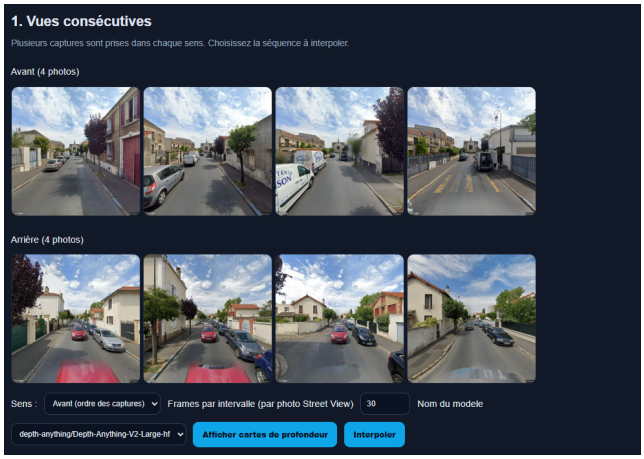


Figure 2: Vues consécutives Street View (sens avant et arrière).

4.2 Estimation de profondeur

Deux modèles sont utilisés :

- **Depth Anything V2** (modèle pré-entraîné) : il génère une profondeur relative en zéro-shot. La carte est normalisée par min/max et un seuil ciel est appliqué (valeurs < 0.03 fixées à 0) pour éviter le bruit distant.
- **UNet personnalisé** (entraîné sur Cityscapes Depth and Segmentation, scènes urbaines extérieures) : un réseau U-Net 2D avec blocs `Conv+BN+ReLU`, entrée RGB, sortie profondeur. Le notebook `train_unet.ipynb` détaille le téléchargement via KaggleHub, la classe `CityscapesKaggleDataset` (support `.npy`/`.png`) et l'entraînement (batch size 8, learning rate 10^{-4}, 5 epochs, perte MSE, resize 128.0000 px × 256.0000 px). Ce choix aligne l'entraînement sur des scènes extérieures proches de Street View.

4.3 Reprojection 3D et softmax splatting

Chaque image RGB et sa profondeur associée sont converties en nuage 3D par back-projection, avec une caméra pinhole (FOV 90.0000°, $f_x = f_y = (w/2)/\tan(\text{FOV}/2)$). La profondeur normalisée est transformée en une pseudo-profondeur métrique par inversion ($z = 1/(d + \epsilon)$), avec un ancrage "ciel" fixé à une grande distance (10 000.0000 m). Un shift Δz simule le déplacement

de la caméra. Le warping est réalisé par un softmax splatting différentiable :

$$\hat{I}(p) = \frac{\sum_q I(q) \exp(m(q)) 1[q \to p]}{\sum_q \exp(m(q)) 1[q \to p] + \epsilon}, \tag{1}$$

avec un métrique m basé sur la profondeur (occlusions gérées par pondération).

La projection utilise les formules classiques : pour un pixel (u, v) avec profondeur z,

$$x = (u - c_x)\, z/f_x, \quad y = (v - c_y)\, z/f_y, \tag{2}$$

et après déplacement $z' = z - \Delta z$,

$$u' = x\, f_x/z' + c_x, \quad v' = y\, f_y/z' + c_y. \tag{3}$$

Les pixels avec $z' \leq 0.1$ sont invalidés afin d'éviter les divisions instables, puis interpolés via le splatting.

4.4 Fusion bidirectionnelle et inpainting

Le système génère un warp *forward* (A→B) et un warp *backward* (B→A). Une fusion adaptative combine les deux selon : (i) un fondu temporel linéaire t ; (ii) une logique d'occlusion par profondeur ; (iii) une détection de changement d'état (différence RGB > 30) pour accélérer la transition sur les objets dynamiques. Les trous restants sont comblés par LaMa si disponible, sinon par inpainting OpenCV (Navier-Stokes).

4.5 Planification de mouvement

Le déplacement est guidé par un *distance hint* issu des métadonnées GPS. La distance réelle d est convertie en distance virtuelle $s = 0.50 \times d$ (fallback $s = 0.35$). Le freinage dynamique applique un facteur

$$t_{geo} = t\,(1 - \alpha t), \quad \alpha = 0.36, \tag{4}$$

permettant un ralentissement progressif en fin de segment. Le code force également l'insertion des images officielles aux bornes des segments pour limiter les dérives cumulées.

4.6 Interface et export vidéo

L'application Flask expose : (i) la génération des sources, (ii) la prévisualisation de profondeur, (iii) l'évaluation des modèles, (iv) la galerie d'animation. La cadence vidéo est calculée par :

$$\text{fps} = \text{clamp}\left(\frac{v}{d_{frame}}, 8, 60\right) \tag{5}$$

avec d_{frame} la distance réelle par frame et v une vitesse cible (8.0000 m/s par défaut).

4.7 Pipeline résumé

Le pipeline complet peut être résumé par les étapes suivantes :

1. Acquisition de N images Street View successives et des distances GPS.
2. Estimation des profondeurs D_A, D_B (Depth Anything ou UNet).
3. Warping 3D de I_A et I_B vers une vue intermédiaire via softmax splatting.
4. Fusion bidirectionnelle avec heuristiques d'occlusion et de changement d'état.
5. Inpainting des disocclusions (LaMa ou OpenCV).
6. Insertion des images officielles aux bornes des segments, export en frames puis vidéo.

5 Détails d'implémentation

Le code est organisé en modules clairs : `generate_frames.py` pour l'acquisition Street View, `depth_models.py` pour l'inférence profondeur, `interpolate.py` pour le warping et l'inpainting, `depth_metrics.py` pour l'évaluation, et `app.py` pour l'interface Flask. Les modèles sont chargés avec un cache LRU afin de limiter les surcoûts d'inférence lors de la prévisualisation et de l'évaluation.

Les sorties sont organisées dans `street_view_project_output/run-YYYYMMDD-HHMMSS`, avec des sous-dossiers pour les sources, les frames, les vidéos et les métriques. Chaque comparaison de modèles est stockée dans un répertoire `model_metrics/forward` ou `model_metrics/ground_truth` avec un `metrics.json` et un résumé `metrics.txt`.

5.1 Gestion de la profondeur

Le choix du modèle (UNet ou Depth Anything) est résolu dynamiquement selon les fichiers disponibles. Pour UNet, le code supporte des checkpoints préparés avec ou sans BatchNorm, et adapte automatiquement l'architecture en fonction du `state_dict`. Pour Depth Anything, les entrées sont traitées via `AutoImageProcessor` et le résultat est interpolé à la résolution originale.

5.2 Complexité et performance

L'évaluation Street View (4 images) est exécutée en 1.4900 s sur la machine courante (voir `duration_s` dans `metrics_forward.json`), ce qui donne un ordre de grandeur pour l'inférence multi-image. Le pipeline exploite CUDA si disponible, mais fournit un fallback CPU. L'export vidéo utilise OpenCV et un FPS adaptatif afin de garder une vitesse de déplacement stable quelle que soit la distance entre panoramas.

6 Expériences et protocole

6.1 Jeux de données

Street View. Les expériences principales utilisent des séquences de 4 panoramas (`source_00` à `source_03`), récupérées automatiquement par l'API. Les images sont rectilignes, alignées sur la route et de taille 640.0000 px × 640.0000 px.

Cityscapes Depth and Segmentation. Le modèle UNet est entraîné sur ce dataset urbain extérieur (Kaggle) afin de mieux correspondre aux scènes Street View. Les images RGB et profondeurs sont redimensionnées pour l'entraînement rapide, et chargées via `CityscapesKaggleDataset`.

Batch ground truth (Cityscapes). Un lot de 100 paires est utilisé pour l'évaluation (`data_train_test/photos` et `data_train_test/depth`). Les métriques reportées sont des moyennes sur le batch, et les visuels montrent un sample représentatif.

6.2 Prétraitement

Les cartes de profondeur sont normalisées par min/max. La version Depth Anything applique un seuil ciel $d < 0.03$. Les sorties UNet sont redimensionnées vers la résolution originale et normalisées en $[0, 1]$. Ces choix favorisent la stabilité du warping mais limitent l'interprétation métrique absolue.

6.3 Paramètres des modèles

UNet : U-Net à 4 niveaux (64-128-256-512), BN, MSE loss, Adam (10^{-4}), batch size 8, 5 epochs, images 128.0000 px × 256.0000 px (Cityscapes). **Depth Anything V2** : modèle pré-entraîné `depth-anything/Depth-Anything-V2-Large-hf`.

6.4 Courbe d'entraînement

La Figure 3 résume l'évolution de la perte MSE moyenne par epoch lors de l'entraînement du UNet.

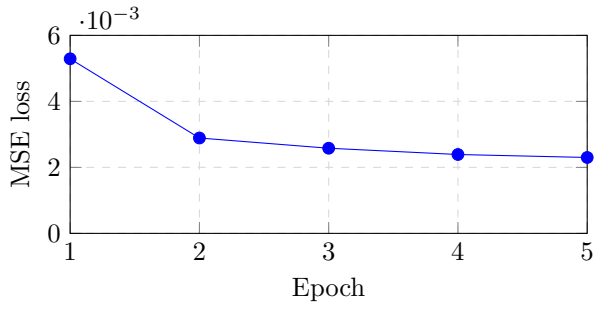


Figure 3: Évolution de la loss MSE moyenne par epoch lors de l'entraînement UNet (Cityscapes Depth and Segmentation).

6.5 Métriques

L'évaluation utilise les fonctions de `depth_metrics.py` :

- MAE : $\frac{1}{N}\sum |d_a - d_b|$
- RMSE : $\sqrt{\frac{1}{N}\sum (d_a - d_b)^2}$
- Pearson r : corrélation linéaire des profondeurs valides
- Edge F1 : F1-score sur cartes d'arêtes obtenues par Sobel (percentile 90)
- Valid ratio : proportion de pixels $d > 0.05$

6.6 Sensibilité aux paramètres

Plusieurs paramètres influencent directement la qualité visuelle : le ratio distance réelle/virtuelle (fixé à 0.50), la force de freinage (0.36), le seuil ciel (0.03), le seuil d'occlusion (0.5) et le seuil de changement visuel (30). Une valeur de ratio trop faible fige la scène, tandis qu'un ratio trop fort génère des déchirures sur les bords. Le freinage réduit les artefacts en fin de segment mais peut produire un mouvement non linéaire perceptible si la séquence est courte. L'utilisation de LaMa apporte une meilleure continuité spatiale que l'inpainting OpenCV, au prix d'un temps d'inférence plus long.

7 Résultats

7.1 Comparaison sans ground truth (Street View)

Le fichier `metrics_forward.json` compare UNet et Depth Anything sur 4 images source (sens forward). La Table 1 résume les résultats. Le Pearson r positif indique une cohérence structurelle modérée, avec un écart d'échelle attendu (UNet Cityscapes normalisé vs profondeur relative de Depth Anything).

Table 1: Comparaison UNet vs Depth Anything (Street View, forward).

Métrique	MAE	RMSE	r	Edge F1	Valid
Valeur	0.1560	0.2241	0.5583	0.2146	0.9995

7.2 Comparaison avec ground truth (batch Cityscapes, 100 images)

Le fichier `metrics_ground_truth.json` évalue les deux modèles sur un batch de 100 paires (photos/profondeurs). Depth Anything obtient des MAE/RMSE plus faibles, un Pearson r plus élevé et un Edge F1 supérieur (Table 2). Les valeurs sont des moyennes sur le batch, et les visuels ci-dessous illustrent un sample du lot.

Table 2: Comparaison avec ground truth (batch Cityscapes, moyennes sur 100 images).

Modèle	MAE	RMSE	r	EdgeF1	Valid
UNet	0.1493	0.1874	0.7419	0.1701	0.7950
Depth Anything	0.0669	0.0805	0.9784	0.2416	0.7917

7.3 Visualisations quantitatives

Figure 4 montre une comparaison graphique sur le batch Cityscapes (moyennes). Figure 5 affiche la variation de MAE par image pour la séquence Street View.

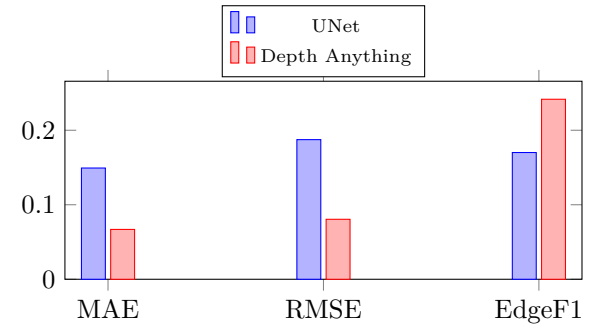


Figure 4: Comparaison graphique sur la GT (batch Cityscapes, plus bas est meilleur pour MAE/RMSE, plus haut pour Edge F1).

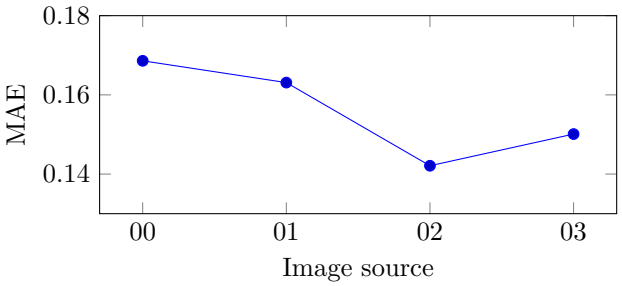


Figure 5: MAE par image (Street View forward, UNet vs Depth Anything).

7.4 Analyses qualitatives

Les figures suivantes rassemblent des sorties qualitatives et des captures d'écran du système.

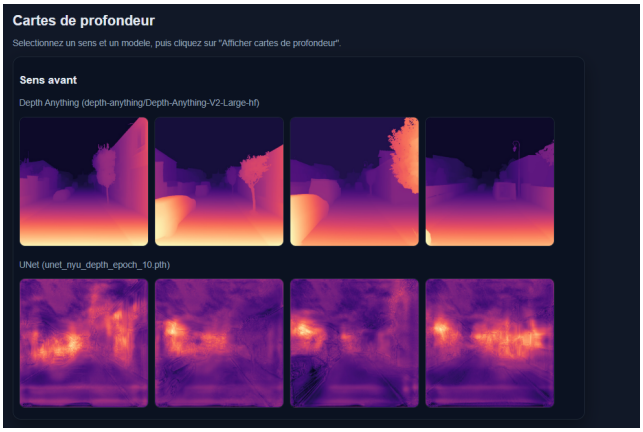


Figure 6: Prévisualisation des cartes de profondeur dans l'interface.

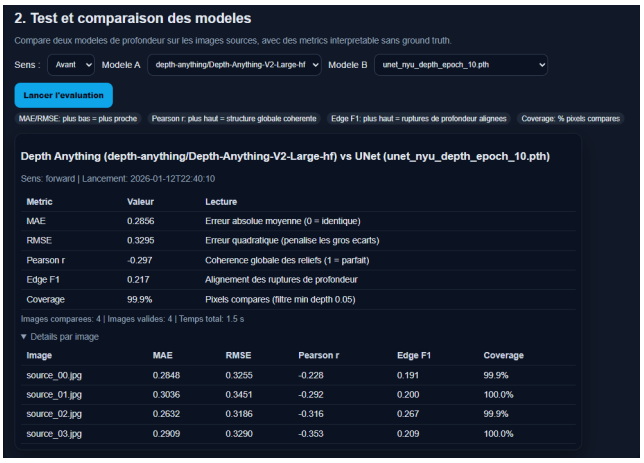


Figure 7: Interface d'évaluation et métriques comparatives.

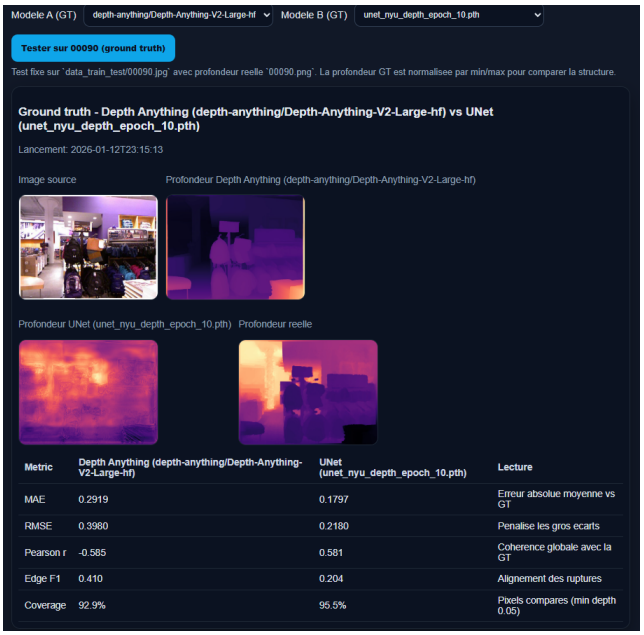


Figure 8: Comparaison des profondeurs avec une GT batch (interface).

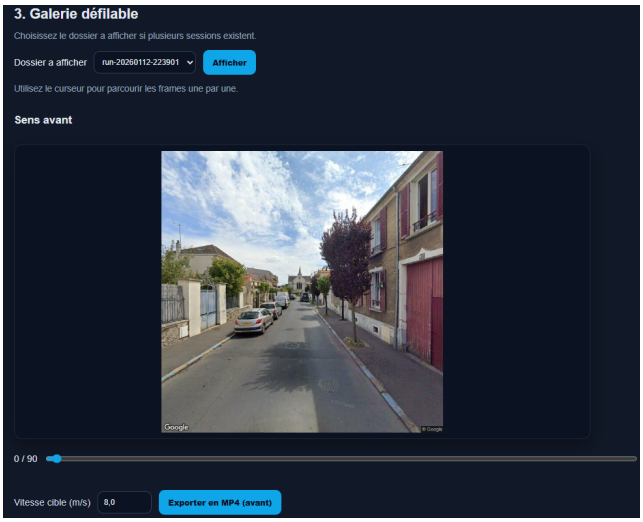
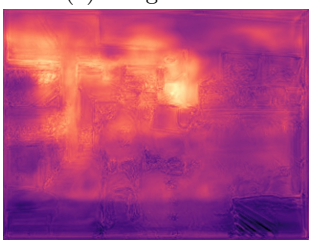


Figure 9: Galerie défilable permettant de parcourir les frames.

(a) Image source

(b) Profondeur GT



(c) UNet

(d) Depth Anything

Figure 10: Exemple de comparaison visuelle.

7.5 Analyse qualitative détaillée

Les sorties qualitatives mettent en évidence plusieurs phénomènes récurrents. D'abord, les zones de ciel sont bien stabilisées grâce au seuil de profondeur qui supprime le bruit lointain ; cela réduit les oscillations visuelles mais peut aplatir certaines structures (ex. toits très lointains). Ensuite, les objets fins (poteaux, panneaux) sont sensibles à la qualité des bords de profondeur : une profondeur trop lissée crée des "fantômes" lors du fondu, tandis qu'une profondeur trop bruitée produit des trous irréguliers.

La fusion bidirectionnelle améliore nettement les zones de disocclusion : un objet qui disparaît dans le warp forward est souvent visible dans le warp backward. Le seuil d'occlusion (0.5) et l'accélération de transition pour les zones changeantes (diff RGB > 30) limitent le "ghosting" sur les objets dynamiques (voitures, piétons), mais peuvent introduire des ruptures si le contraste est fort (ex. changement d'ombre).

Enfin, l'inpainting LaMa fournit des textures plausibles pour le ciel et les façades, mais sa cohérence temporelle reste imparfaite.

8 Discussion

Les résultats sur le batch ground truth (100 images) montrent un avantage clair de Depth Anything : MAE/RMSE plus faibles, Pearson r plus élevé et Edge F1 supérieur. Cette différence est cohérente avec une meilleure généralisation zero-shot, tandis que l'UNet Cityscapes reste plus sensible à l'échelle et à la normalisation min/max.

Sur Street View, la comparaison sans GT indique une cohérence structurelle modérée (Pearson r positif), signe que les deux modèles capturent des reliefs comparables, mais avec des divergences d'échelle. Malgré cela, l'approche de fusion bidirectionnelle et le softmax splatting produisent des transitions plausibles, avec des artefacts principalement localisés sur les bords et les objets dynamiques. L'inpainting LaMa limite visuellement les trous, mais la cohérence temporelle n'est pas garantie.

Un point important est l'usage d'un "distance hint" dérivé des métadonnées GPS. Dans les cas où la distance est disponible, l'alignement géométrique est plus stable que le fallback purement heuristique. Cette dépendance aux métadonnées souligne toutefois une fragilité : un GPS bruité ou des panoramas mal espacés peuvent désynchroniser la transition, ce qui amplifie les erreurs de warping.

Le pipeline montre aussi la tension classique entre qualité per-frame et cohérence temporelle. Les cartes de profondeur plus détaillées (Depth Anything) améliorent certaines structures mais produisent des inconsistances d'une frame à l'autre. A l'inverse, les cartes UNet sont plus lisses et peuvent être plus stables, mais perdent des bords fins. Une stratégie hybride (fusion de profondeur, temporal smoothing) pourrait stabiliser le rendu sans perdre en détail.

9 Limites et leçons apprises

- **Baselines larges.** Le warping géométrique souffre lorsque le déplacement entre panoramas est important. Les erreurs de profondeur sont amplifiées et créent des déformations irréversibles.
- **Profondeur relative.** La normalisation min/max et le seuil ciel rendent difficile la comparaison inter-modèle et l'estimation métrique absolue.
- **Données limitées.** L'évaluation ground truth repose sur 100 paires (photos/profondeurs). Cela reste insuffisant pour une conclusion statistique solide.
- **Décalage de domaine.** Même en restant sur des scènes extérieures, un *gap* subsiste entre Cityscapes et Street View (géométrie, dynamique, textures).

- **Essai NYU Depth V2.** Nous avons testé NYU, mais cela ne pouvait pas bien fonctionner en extérieur, le dataset ne contenant pas de photos extérieures.
- **Objets dynamiques.** La détection de changement d'état par différence RGB reste heuristique et peut se tromper sur des changements d'illumination.
- **Reproductibilité.** L'API Street View dépend d'une clé et de données externes (potentiellement non stables dans le temps).

10 Extensions possibles

- Utiliser des modèles de profondeur métrique (Metric3D, MASt3R) pour obtenir un scale absolu plus fiable.
- Exploiter des correspondances denses (RoMa, LoFTR) pour raffiner la fusion et détecter les occlusions.
- Introduire des modèles temporels (VFI ou diffusion conditionnelle) pour stabiliser les transitions.
- Tester sur davantage de séquences et mettre en place une validation quantitative à plus grande èchelle.

11 Conclusion

Ce projet fournit un prototype complet de synthèse de vues intermédiaires pour Street View, combinant estimation de profondeur, reprojection 3D et inpainting. Les analyses qualitatives valident la faisabilité d'un mouvement fluide, tandis que les métriques quantitatives mettent en lumière la sensibilité du pipeline à la qualité de la profondeur. La prochaine ètape consiste à renforcer la cohérence géométrique à grande baseline et à étendre l'évaluation statistique.

Reproductibilité (résumé)

- Configuration : renseigner la clé API dans `config.txt` ou via `GOOGLE_API_KEY`.
- Dépendances : `requests`, `opencv-python`, `torch`, `transformers`, `simple-lama-inpainting`, etc. (voir `requirements.txt`).
- Exécution : lancer `app.py` puis générer une session, prévisualiser la profondeur, interpoler et exporter.
- Évaluation : utiliser les boutons "Evaluate models" et "Ground truth" pour générer les fichiers de métriques.

A Détails des métriques par image

La Table 3 liste les métriques par image issues de `metrics_forward.json`. Cette granularité aide à détecter les scènes plus difficiles (ex. changements d'illumination ou objets dynamiques).

Table 3: Métriques par image (Street View, forward).

Image	MAE	RMSE	r	EdgeF1	Valid
source_00.jpg	0.1686	0.2405	0.5039	0.1929	0.9994
source_01.jpg	0.1631	0.2444	0.5775	0.1853	0.9992
source_02.jpg	0.1421	0.2092	0.5949	0.2682	0.9997
source_03.jpg	0.1501	0.2023	0.5570	0.2119	0.9997

References

[1] L. Yang et al., "Depth Anything V2: A more capable foundation model for monocular depth estimation," NeurIPS 2024. https://github.com/DepthAnything/Depth-Anything-V2

[2] E. Vincent et al., "MASt3R: Grounding image matching in 3D," arXiv 2024. https://github.com/naver/mast3r

[3] J. Sun et al., "LoFTR: Detector-free local feature matching with transformers," CVPR 2021. https://github.com/zju3dv/LoFTR

[4] J. Edstedt et al., "RoMa: Robust dense feature matching," CVPR 2024. https://github.com/Parskatt/RoMa

[5] S. Niklaus and F. Liu, "Softmax splatting for video frame interpolation," CVPR 2020. https://github.com/sniklaus/softmax-splatting

[6] J. T. Barron et al., "Zip-NeRF: Anti-aliased grid-based neural radiance fields," ICCV 2023. https://github.com/google-research/multinerf

[7] B. Kerbl et al., "3D Gaussian Splatting for real-time radiance field rendering," SIGGRAPH 2023. https://github.com/graphdeco-inria/gaussian-splatting

[8] R. Suvorov et al., "Resolution-robust large mask inpainting with Fourier convolutions (LaMa)," WACV 2022. https://github.com/advimman/lama

[9] W. Li et al., "MAT: Mask-Aware Transformer for large hole image inpainting," CVPR 2022. https://github.com/fenglinglwb/MAT

[10] SIFT (Scale-Invariant Feature Transform). https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

[11] RANSAC. https://en.wikipedia.org/wiki/Random_sample_consensus

[12] SuperPoint (code repository). https://github.com/magicleap/SuperPointPretrainedNetwork

[13] LightGlue (code repository). https://github.com/cvg/LightGlue

[14] XFeat (code repository). https://github.com/verlab/accelerated_features

[15] RAFT (optical flow, code repository). https://github.com/princeton-vl/RAFT

[16] SEA-RAFT (code repository). https://github.com/princeton-vl/SEA-RAFT

[17] Learning to Render Novel Views from Wide-Baseline Stereo Pairs (code repository). https://github.com/yilundu/cross_attention_renderer

[18] S-NeRF (code repository). https://github.com/fudan-zvg/S-NeRF

[19] RIFE (code repository). https://github.com/megvii-research/ECCV2022-RIFE

[20] IFRNet (code repository). https://github.com/ltkong218/IFRNet

[21] FILM (code repository). https://github.com/google-research/frame-interpolation

[22] Infinite Nature (Google Research). https://github.com/google-research/google-research/tree/master/infinite_nature

[23] Frame Interpolation Transformer and Uncertainty Guidance. https://assets.studios.disneyresearch.com/app/uploads/2023/05/Frame-Interpolation-Transformer-and-Uncertainty-Guidance-1.pdf

[24] Google Maps Immersive View. https://blog.google/intl/fr-fr/nouveautes-produits/explorez-obtenez-des-reponses/google-maps-immersive-view-itineraires-nouvelles-fonctionnalites-ia/