Génération d'images omnidirectionnelles à partir d'un environnement virtuel

Ahmed Rida Sekkat, Yohan Dupuis, Pascal Vasseur, Paul Honeine

Génération d'images omnidirectionnelles à partir d'un environnement virtuel

Ahmed Rida SEKKAT[1], Yohan DUPUIS[2], Pascal VASSEUR[1], Paul HONEINE[1]

[1] Normandie Univ, UNIROUEN, LITIS, 76000 Rouen, France

[2] Normandie Univ, UNIROUEN, ESIGELEC, IRSEEM, 76000 Rouen, France

ahmed-rida.sekkat@univ-rouen.fr, yohan.dupuis@esigelec.fr
pascal.vasseur@univ-rouen.fr, paul.honeine@univ-rouen.fr

Résumé – Dans cet article, nous décrivons une méthode pour générer des images omnidirectionnelles en utilisant des images *cubemap* et les cartes de profondeur correspondantes, à partir d'un environnement virtuel. Pour l'acquisition, on utilise le jeu vidéo Grand Theft Auto V (GTA V). GTA V a été utilisé comme source de données dans plusieurs travaux de recherche, puisque c'est un jeu à monde ouvert, hyperréaliste, simulant une vraie ville. L'avancée réalisée dans l'ingénierie inverse de ce jeu nous offre la possibilité d'extraire des images et les cartes de profondeur correspondantes avec des caméras virtuelles à six degrés de liberté. A partir de ces données et d'un modèle de caméra omnidirectionnelle, on propose de générer des images *Fisheye* destinées par exemple à l'entraînement de méthodes par apprentissage.

Abstract – This paper describes a method for generating omnidirectional images using cubemap images and corresponding depth maps that can be acquired from a virtual environment. For this purpose, we use the video game Grand Theft Auto V (GTA V). GTA V has been used as a data source in many research projects, due to the fact that it is a hyperrealist open-world game that simulates a real city. We take advantage of developments made in reverse engineering this game, in order to extract realistic images and corresponding depth maps using virtual cameras with 6DoF. By combining the extracted information with an omnidirectional camera model, we generate Fish-eye images intended for instance to machine learning based applications.

1 Introduction

Percevoir et comprendre l'environnement est une tâche essentielle pour un véhicule autonome. Une des problématiques principales pour le développement de ces véhicules est l'existence de jeux de données. Parmi les jeux de données des images perspectives dédiées à l'étude et le développement des véhicules autonomes, on peut citer Kitti [1], Cityscape [2] et Berkeley DeepDrive [3]. Les caméras omnidirectionnelles permettent de percevoir l'environnement autour avec un champ de vision qui peut atteindre 360°, elles sont de plus en plus utilisées dans le domaine des véhicules intelligents , notamment les caméras *Fisheye* pour leur compacité. Il existe plusieurs jeux de données qui contiennent des images *Fisheye* comme par exemple CVRG [4], LMS [5], LaFiDa [6], SVMIS [7], et GM-ATCI [8]. Cependant on remarque qu'il existe un manque au niveau des jeux de données d'images *Fisheye* embarquées dans un véhicule pour des scènes routières pour des applications de vision par ordinateur. Les récents travaux sur la segmentation sémantique d'images *Fisheye* de scènes routières ont été effectués sur des images perspectives auxquelles une déformation simulant l'effet *Fisheye* est appliquée [9, 10, 11]. Une telle déformation induit des artefacts dans les images obtenues. Un besoin grandissant se fait ressentir pour générer un jeu de données d'images *Fisheye* plus fiable, sans la nécessité de passer par une simple déformation d'images perspectives.

Dans la littérature, on trouve plusieurs travaux effectués sur des environnements virtuels pour le développement ou la validation des systèmes de conduite autonomes. Les environnements virtuels ont plusieurs avantages, à savoir le coût ainsi que la variété de la nature des données qui peuvent être générées, comme les cartes de profondeur ou encore la segmentation sémantique. Ces environnements virtuels rendent possible la simulation de différents capteurs. Il existe actuellement des environnements virtuels *open source* pour générer des données, comme Carla simulator [12], SYNTHIA [13], et VEIS [14]. Toutefois, à cause du peu de ressources allouées, ces environnements résultent en un graphisme moins réaliste que les jeux de vidéo de très haute qualité, dits jeux AAA comme Grand Theft Auto V (GTA V). Dans GTA V, il existe un environnement similaire à la vie réelle (météo, saison, trafic routier régulé, feux de circulation, signalisation, piétons, différents types de véhicule, ...). Pour ces raisons, ce jeu a été utilisé à plusieurs reprises comme source de données pour générer des images très réalistes de trafic routier.

Dans cet article, nous proposons la génération d'images *Fi-*

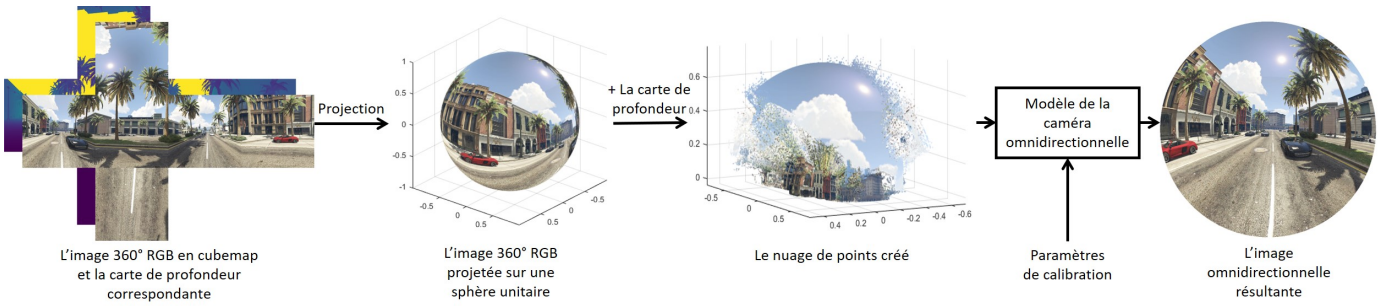

FIGURE 1 – Étapes de génération d'images omnidirectionnelles.

sheye à partir du jeu vidéo GTA V. Pour cela, nous profitons de l'ingénierie inverse de ce jeu, qui nous accorde ainsi l'accès à des informations telles que les images selon différentes vues et leurs cartes de profondeur. Nous proposons d'utiliser ces informations pour générer des images *Fisheye*, en utilisant le modèle proposé par Scaramuzza et al. [15].

2 Travaux connexes

Grâce aux avantages qu'offrent GTA V et les outils de *modding* associés, plusieurs travaux récents ont été effectués sur la génération de données à partir de ce jeu. On peut citer Doan et al. [16] qui proposent une méthode pour générer des images perspectives en utilisant une caméra virtuelle à six degrés de liberté. Richter et al. [17] ont utilisé GTA V pour obtenir des captures avec la segmentation sémantique pixel par pixel en utilisant d'autres outils comme un *middleware* libre appelé renderdoc entre le jeu et le GPU. Angus et al. [18] ont aussi extrait des images de segmentation sémantique en changeant les textures du jeu dans les fichiers de ce dernier. Richter et al. [19] ont généré un *benchmark* de plusieurs types de données à partir de GTA V, toutes annotées avec des données de vérité terrain pour les tâches de vision de bas niveau et de haut niveau, y compris le flux optique, la segmentation sémantique, la détection et suivi d'objets ainsi que l'odométrie visuelle. Johnson-Roberson et al. [20] ont utilisé des données extraites de GTA V, pour montrer que les algorithmes de l'état de l'art entraînés uniquement à l'aide de données synthétiques, fonctionnent mieux que s'ils sont entraînés sur des données du monde réel annotées manuellement, lorsqu'ils sont testés sur le jeu de données KITTI [1] pour la détection de véhicules. D'où l'intérêt et l'apport que peut nous apporter ce genre de données synthétiques générées à partir d'un environnement virtuel.

3 Méthode générale

L'idée de la méthode est de pouvoir générer des images omnidirectionnelles à partir d'un environnement virtuel. Cette méthode est possible si on arrive à extraire une image 360° avec la carte de profondeur correspondante. Pour pouvoir récupérer des images 360°, on extrait six images dans six différentes directions, qu'on projette ensuite sur une sphère unitaire. On remplace alors la valeur du rayon de chaque pixel par la valeur de la carte de profondeur correspondante pour obtenir un nuage de points 360° représentant ainsi un repère monde. Les cartes de profondeur nous permettent essentiellement de découper les six images pour composer l'image *cubemap*. Une fois à ce stade, on projette ces points 3D en utilisant le modèle de modélisation de caméra omnidirectionnelle approprié. Pour cela, les paramètres du modèle sont calculés à partir d'une caméra calibrée. On pourra par exemple utiliser le modèle proposé par Geyer et Daniilidis [21] ou Barreto et Araujo [22], celui de Mei et Rives [23] ou encore le modèle présenté par Scaramuzza et al. [15] pour générer des images omnidirectionnelles. Le schéma fonctionnel de génération de ces images est présenté dans la FIGURE 1.

La méthode proposée s'applique sur tous les modèles susmentionnés. Par souci de clarté, nous détaillons dans la suite uniquement le modèle proposé par Scaramuzza et al. [15] pour générer des images *Fisheye*. Ce modèle de calibration de caméras omnidirectionnelles permet de calculer les paramètres intrinsèques de la caméra. Ce qui signifie qu'il nous permet de trouver la relation entre un pixel 2D donné et le vecteur 3D correspondant, partant du point de vue du miroir, comme simplifié dans la FIGURE 2. Soient (u, v) les coordonnées métriques d'un pixel p par rapport au centre de l'image omnidirectionnelle, et (x, y, z) celles du vecteur 3D P correspondant. La fonction de calibration à estimer est la fonction qui associe un point p de l'image à son vecteur 3D correspondant P, selon

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ f(w) \end{bmatrix}, \text{ avec } w = \sqrt{u^2 + v^2}. \quad (1)$$

Ce modèle considère la fonction $f(w)$ comme une fonction polynomiale de la forme

$$f(w) = a_0 + a_1 w + a_2 w^2 + a_3 w^3 + a_4 w^4 + ... \quad (2)$$

Les paramètres de calibration a_i sont estimés par moindres carrées sur des données acquises par une caméra réelle [24].

4 Application

Comme précisé auparavant, on a choisi d'appliquer cette méthode sur le jeu GTA V afin de créer des images *Fisheye*. Pour cela, on a besoin d'une vue 360° et de la carte de profondeur correspondante. À cet égard, nous avons utilisé des outils de *modding* pour pouvoir contrôler la caméra dans le jeu et on a utilisé le code GTAVisionExport [20] afin d'extraire les images RGB et les cartes de profondeur associées. Une fois les captures acquises, on procède à la création des images *Fisheye*.

4.1 Outils de *modding* et de capture

La communauté de *modding* sur GTA V est très large, plusieurs scripts ont été développés pour changer des paramètres dans le jeu. Cela grâce à une bibliothèque libre appelée Script Hook V [25] en C++ ou encore ScriptHookDotNet2 [26] qui est une couche au-dessus de Script Hook V, rendant possible l'écriture de scripts de modification en C#. Ces bibliothèques permettent l'exécution d'appels à des fonctions natives de GTA V, et donc la mise en place de scripts de modification appelés *mods*. On utilise cet outil dans ce travail pour changer la caméra par défaut de GTA V et la remplacer par une autre caméra à laquelle on change les paramètres extrinsèques de rotation par rapport au repère monde pour prendre six captures à partir du même point de vue. On change aussi le champ de vision vertical pour être égal à 90°, afin d'avoir une vue complète à 360°. Grâce à l'outil *open source* GTAVisionExport [20] via des fonctions DirectX, on extrait les images RGB et les cartes de profondeur correspondantes. Moyennant ce même code, on obtient également une segmentation sémantique en 5 classes, à savoir véhicules, bâtiment et arrière-plan, piétons et conducteurs, verdure et ciel.

4.2 Génération d'images *Fisheye*

Pour obtenir une image 360°, on capture six images dans les six directions (haut, bas, avant, arrière, gauche, droite), en procédant à la rotation de la caméra autour d'un seul point. Ces images sont projetées sur une sphère pour former une image 360°. Avec ces images on peut aussi créer une image panoramique. Une fois obtenues les six captures et les cartes de profondeur correspondantes, les images sont rectifiées sur la largeur de façon à conserver un champ de vision horizontal correspondant à 90°. Les images RGB sont ensuite projetées sur une sphère unitaire de telle sorte que le rayon soit égal aux valeurs associées dans les cartes de profondeur. Le repère de la sphère est donc équivalant au repère monde. On s'assure qu'il y a assez de points 3D pour que chaque pixel dans l'image omnidirectionnelle soit associé au moins à un point 3D. On applique alors la projection de ce nuage de points en une image *Fisheye* à l'aide de la *toolbox* de Scaramuzza et al. [15], en utilisant des paramètres de calibration d'une vraie caméra *Fisheye* calibrée calculés auparavant. On applique cela à chaque fois sur la demi-sphère pour générer deux images *Fisheye* pour chaque prise 360°, comme présenté sur le schéma fonctionnel FIGURE 1.

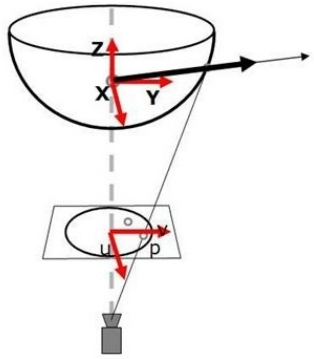

FIGURE 2 – Le modèle de caméra omnidirectionnelle proposé par Scaramuzza [15].

Les données générées peuvent être utilisées comme vérité-terrain pour des applications très variées, comme la localisation et cartographie simultanées (SLAM), l'odométrie visuelle, ou encore l'estimation de cartes de profondeur. Elles peuvent être aussi utilisées pour la reconnaissance et la classification d'objets, également la détection et le suivi. On pourra aussi les utiliser pour évaluer les algorithmes de segmentation sémantique développés pour les images *Fisheye*, ou pour l'entraînement. A titre d'exemple, la FIGURE 3 présente des images *Fisheye* générées avec un champ de vision de 180° en RGB, en carte de profondeur et en segmentation sémantique. Il est à noter que le jeu GTA V contient essentiellement des environnements semblables à ceux des États-Unis, donc il y a un risque d'erreur ou biais à prévoir pour une application d'apprentissage en France.

5 Conclusion

Ce document présente une méthode qui peut être utilisée pour générer des jeux de données d'images omnidirectionnelles à partir d'environnements virtuels, ainsi qu'une application sur GTA V, générant des images *Fisheye* avec la carte de profondeur et une segmentation sémantique de base. La prochaine étape consiste maintenant à développer une méthode d'évaluation de la qualité des données générées. Il existe de nombreuses extensions possibles à cette application, y-compris la génération de jeux de données, utilisant différents types de modèles de caméras omnidirectionnelles. Ces jeux de données peuvent être utilisés comme références d'évaluation pour différentes applications de vision et d'apprentissage profond, dont les algorithmes appliqués aux images perspectives présentent des performances limitées sur les images omnidirectionnelles.

Références

[1] J. Fritsch, T. Kuehnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

(a) Image *Fisheye* RGB. (b) Carte de profondeur *Fisheye*. (c) Segmentation sémantique *Fisheye*.

FIGURE 3 – Exemple d'images *Fisheye* générées à partir d'une seule prise.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "BDD100K : A diverse driving video database with scalable annotation tooling," *CoRR*, vol. abs/1805.04687, 2018.

[4] I. Baris and Y. Bastanlar, "Classification and tracking of traffic scene objects with hybrid camera systems," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, Oct 2017.

[5] A. Eichenseer and A. Kaup, "A data set providing synthetic and real-world fisheye video sequences," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1541–1545, Mar 2016.

[6] S. Urban and B. Jutzi, "Lafida - a laserscanner multi-fisheye camera dataset," *J. Imaging*, vol. 3, p. 5, 2017.

[7] G. Caron and F. Morbidi, "Spherical Visual Gyroscope for Autonomous Robots using the Mixture of Photometric Potentials," in *IEEE International Conference on Robotics and Automation*, (Brisbane, Australia), pp. 820–827, May 2018.

[8] D. Levi and S. Silberstein, "Tracking and motion cues for rear-view pedestrian detection," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 664–671, Sep. 2015.

[9] A. M. Sweeney, L. M. Bergasa, E. Romera, M. E. L. Guillén, R. Barea, and R. Sanz, "Cnn-based fisheye image real-time semantic segmentation," *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1039–1044, 2018.

[10] L. Deng, M. Yang, Y. Qian, C. Wang, and B. Wang, "Cnn based semantic segmentation for urban traffic scenes using fisheye camera," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 231–236, June 2017.

[11] L. Deng, M. Yang, H. Li, T. Li, B. Hu, and C. Wang, "Restricted deformable convolution based road scene semantic segmentation using surround view cameras," *CoRR*, vol. abs/1801.00708, 2018.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "CARLA : an open urban driving simulator," in *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, pp. 1–16, 2017.

[13] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset : A large collection of synthetic images for semantic segmentation of urban scenes," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3234–3243, June 2016.

[14] F. Sadat Saleh, M. Sadegh Aliakbarian, M. Salzmann, L. Petersson, and J. M. Alvarez, "Effective use of synthetic data for urban scene semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 84–100, 2018.

[15] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5695–5701, Oct 2006.

[16] A. Doan, A. M. Jawaid, T. Do, and T. Chin, "G2D : from GTA to data," *CoRR*, vol. abs/1806.07381, 2018.

[17] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data : Ground truth from computer games," in *European Conference on Computer Vision (ECCV)* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9906 of *LNCS*, pp. 102–118, Springer International Publishing, 2016.

[18] M. Angus, M. ElBalkini, S. Khan, A. Harakeh, O. Andrienko, C. Reading, S. L. Waslander, and K. Czarnecki, "Unlimited road-scene synthetic annotation (URSA) dataset," *CoRR*, vol. abs/1807.06056, 2018.

[19] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[20] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix : Can virtual worlds replace human-generated annotations for real world tasks ?," in *IEEE International Conference on Robotics and Automation*, pp. 1–8, 2017.

[21] C. Geyer and K. Daniilidis, "A unifying theory for central panoramic systems and practical implications," in *Computer Vision — ECCV 2000* (D. Vernon, ed.), (Berlin, Heidelberg), pp. 445–461, Springer Berlin Heidelberg, 2000.

[22] J. P. Barreto and H. Araujo, "Issues on the geometry of central catadioptric image formation," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 2, pp. II–II, Dec 2001.

[23] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3945–3950, April 2007.

[24] Y. Dupuis, X. Savatier, J. Ertaud, and P. Vasseur, "Robust radial face detection for omnidirectional vision," *IEEE Transactions on Image Processing*, vol. 22, pp. 1808–1821, May 2013.

[25] A. Blade, "Script Hook V." `http://www.dev-c.com/gtav/scripthookv/`, 2015. [Online ; accessed 06-March-2019].

[26] Crosire, "Script Hook V .NET." `https://github.com/crosire/scripthookvdotnet`, 2015. [Online ; accessed 06-March-2019].