Segmentation d'instance dans des images fisheye

Rémi Dufour, Cyril Meurie, Olivier Lézoray, Ankur Mahtani

▶ **To cite this version:**

Rémi Dufour, Cyril Meurie, Olivier Lézoray, Ankur Mahtani. Segmentation d'instance dans des images fisheye. ORASIS 2021, Centre National de la Recherche Scientifique [CNRS], Sep 2021, Saint Ferréol, France. hal-03339664

HAL Id: hal-03339664
https://hal.science/hal-03339664v1

Submitted on 9 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Segmentation d'instance dans des images fisheye

Rémi Dufour[1] Cyril Meurie[2,1] Ankur Mahtani[1] Olivier Lézoray[3,1]

[1] FCS Railenium, F-59300 Famars, France
[2] COSYS-LEOST, Université Gustave Eiffel, IFSTTAR, Univ. Lille, Villeneuve d'Ascq, France
[3] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, Caen, France

remi.dufour@railenium.eu ; cyril.meurie@univ-eiffel.fr ;
ankur.mahtani@railenium.eu ; olivier.lezoray@unicaen.fr

Résumé

Le défi de la segmentation d'instance a été exploré principalement avec des images rectilinéaires. Cependant, la segmentation d'instance dans des images fisheye implique des difficultés supplémentaires et n'a pas encore été complètement explorée. Un modèle CNN capable de fonctionner aussi bien sur des données rectilinéaires que sur des données fisheye sans modification permet d'envisager de multiples applications, et possède des avantages en terme de puissance de calcul requise, ce qui se révèle être un aspect crucial pour les systèmes temps réels, comme ceux embarqués en environnement transport. Dans cet article, nous démontrons qu'un ensemble simple d'augmentations fisheye permet d'améliorer les performances sur des images fisheye tout en conservant celles obtenues sur des images rectilinéaires.

Mots Clef

Segmentation d'instance, Deep Learning, CNN, Fisheye, Distorsions

Abstract

The challenge of instance segmentation has been explored mainly with rectilinear images. But in fisheye images, this challenge has not yet been fully explored. A CNN model that can perform well in the rectilinear or fisheye domain without specific modifications has advantages in terms of computing resources requirements, of paramount importance for real-time systems e.g., in transportation systems. In this paper, we show that a simple set of fisheye augmentations can improve fisheye performances while preserving rectilinear performances.

Keywords

Instance segmentation, Deep Learning, CNN, Fisheye, Distortions

1 Introduction

Dans les environnements de transports, et plus particulièrement dans le ferroviaire avec l'arrivée du futur train autonome, la détection et le suivi de personnes est une brique cruciale pour des objectifs de surveillance et de sécurité. Parmi les usages potentiels, nous pouvons par exemple citer la détection d'intrusion, de foule, d'agitation, de violence ou d'évanouissement. La segmentation d'instance de personnes est une première étape importante pour ces différentes tâches, et constitue un domaine de recherche ou de grands progrès ont été réalisés depuis une dizaine d'années. Les caméras fisheye ou rectilinéaires sont toutes deux utiles pour la surveillance d'environnement transport, en fonction du champs de vision désiré. Néanmoins, elles ont des caractéristiques visuelles très différentes, il est donc naturel de développer des algorithmes dédiés à chacun de ces types d'image. Cependant, pour des raisons de puissance de calcul limitée, il peut être plus intéressant de disposer d'un algorithme de détection capable de traiter les deux types d'images, sans modification spécifique. Pour accomplir cet objectif, nous proposons une technique d'augmentation de données qui transforme pendant l'entraînement une partie des images rectilinéaires en images à effet fisheye (FE). Ceci constitue avec les différentes expérimentations la principale contribution de ce papier pour la tâche de segmentation d'instance.

Ce papier est organisé comme suit : les travaux pertinents déjà existants dans le domaine visé sont présentés en Section 2. L'approche proposée est détaillée en Section 3. Les résultats expérimentaux sont présentés et discutés en Section 4. Finalement, la Section 5 aborde les conclusions et les perspectives de ces travaux.

2 État de l'art

L'algorithme le plus souvent utilisé comme référence dans la tâche de segmentation d'instance est Mask R-CNN [1]. Il est constitué d'une architecture de type Faster R-CNN [2], auquel Mask R-CNN rajoute une branche qui effectue la segmentation d'instance. Une partie centrale de ce type d'algorithme est le backbone qui extrait des caractéristiques de l'image. Les progrès réalisés dans la tâche de segmentation d'instance ont aussi été permis grâce à

la publication de base de données conséquentes et adaptées pour l'entraînement. Parmi ces datasets, nous pouvons citer MS COCO [3], CityScape [4] et WoodScape [5]. Cependant, les principaux datasets de segmentation d'instance ne contiennent pas d'images fisheye, à l'exception de WoodScape [5] qui contient 10000 images en environnement urbain, mais qui malheureusement s'avère inadapté pour la segmentation de personnes en environnement ferroviaire (ie. à l'intérieur de voitures de train). Les CNN qui ont été entraînés sur des images rectilinéaires apprennent à reconnaître des objets visuellement rectilinéaires et en conséquence, offrent des performances plus faibles sur des images fisheye, qui présentent des distorsions en barillet, particulièrement intenses aux bords de l'image. Les approches du problème susmentionné peuvent être classifiées en deux catégories : premièrement, les approches qui corrigent les distorsions fisheye pour permettre l'usage d'algorithmes classiques. Un défaut est que la correction (fisheye dewarping) peut causer des artefacts au bord de l'image, ce qui implique la nécessité de trouver un compromis entre le champ de vision, les artefacts, et le temps de calcul. Deuxièmement, les approches qui adaptent l'algorithme de détection afin de le rendre capable de traiter directement les images fisheye. Cela peut être réalisé en modifiant l'algorithme lui-même ou bien les données d'entraînement.

Plusieurs algorithmes de segmentation sémantiques adaptés aux images fisheye ont été développés. Une architecture CNN modifiée, combinée avec une stratégie d'augmentation de données FE (Fisheye Effect) est utilisée dans les travaux de Saez et al. [6], dans le contexte des voitures autonomes. Yang et al. utilise des distorsions en barillet et coussinet au sein d'une stratégie d'augmentation de données pour améliorer les performances sur des images panoramiques [7]. Blott et al. [8] présente une technique d'augmentation de données FE qui améliore les performances de segmentation d'instance sur des images fisheye artificielles et réelles.

Néanmoins, la segmentation d'instance en image fisheye est un sujet de recherche moins exploré. Saito et al. [9] a utilisé en 2010 une technique de soustraction d'arrière-plan, en acceptant comme contraintes une caméra statique et des personnes droites. Ces suppositions ne sont pas valides dans le cadre de notre application qui est la surveillance de personnes à l'intérieur d'un train. En 2019 Wang et al. [10] a entraîné Mask R-CNN en ajoutant une classe "Fisheye-Person" et une augmentation de données par rotation aléatoire comprise entre 0 et 360 degrés.

Dans ce papier, nous utilisons une stratégie d'augmentation de données similaire à Blott et al. [8] afin d'obtenir une performance de segmentation d'instance satisfaisante sur des images rectilinéaires et fisheye acquises dans des environnements de transports. Cette stratégie est détaillée dans la section suivante.

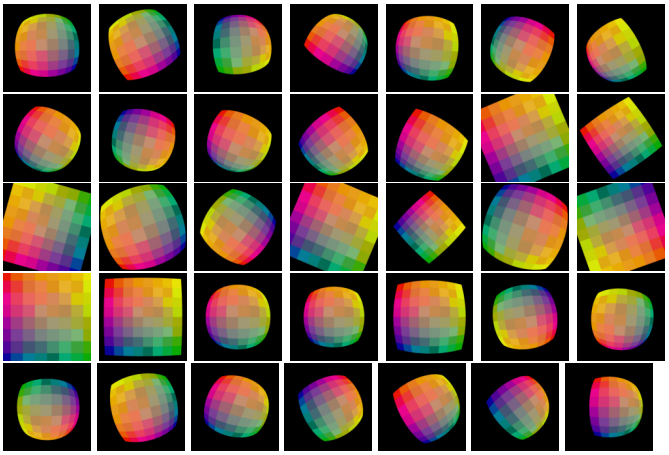
3 Méthode proposée

L'approche proposée utilise comme baseline un algorithme de segmentation d'instance existant, Mask R-CNN [1]. Nous cherchons à l'améliorer afin de le rendre capable de traiter des images fisheye en conservant de bonnes performances sur des images rectilinéaires. L'approche d'adaptation au domaine fisheye consiste à ré-entraîner Mask R-CNN sur sa base de données (COCO2017), avec une augmentation de données FE. Cette augmentation utilise le même modèle de transformation que [8], originellement décrit dans [11]. Ce modèle projette l'image sur la sphère unitaire, déplace le référentiel, puis projette les points sur le plan image normalisé. Ensuite, des distorsions sont opérées selon le modèle introduit par Brown [12], et enfin la projection est effectuée avec les paramètres de la caméra.
Ce modèle de projection est utilisé pour créer un ensemble de transformations qui peut être appliqué efficacement pendant l'entraînement. Dans ce papier, nous comparons deux ensembles de transformations FE. Le premier contient 35 transformations FE avancées, illustrées par la Figure 1a). Le deuxième contient 8 transformations FE plus simples, différenciées uniquement par leur rotation. Le premier ensemble a les propriétés suivantes : translations dans l'intervalle [0;0.5] en échelle normalisée, rotations quelconques autour de l'axe z, mise à l'échelle dans l'intervalle [50%;120%], distorsions radiales dans l'intervalle [-0.5,-1.0], et paramètre de projection ξ dans l'intervalle [0;1]. Le deuxième ensemble a les propriétés suivantes : pas de translation, 8 rotations différentes, selon un pas de $\frac{\pi}{4}$, pas de mise à l'échelle, pas de distorsion, paramètre ξ de 1.
Un ratio d'augmentation est défini afin d'appliquer l'augmentation FE à une portion des exemples d'entraînement. Quand une image est choisie pour l'augmentation FE, une transformation spécifique est choisie au hasard parmi les transformations pré-calculées afin d'optimiser le temps de calcul.
L'entraînement de Mask R-CNN est effectué avec les paramètres par défaut du framework Detectron [13], avec quelques ajustements comme une plus grande taille de batch par GPU, et moins de pas d'entraînement (traitement d'un batch), afin d'obtenir de bonnes performances sur notre matériel (GPU Nvidia Tesla V100). Deux différents programmes d'entraînement sont utilisés : un programme long pour l'architecture resnet101, et un court pour l'architecture resnet50. Le programme long prévoit 90k pas d'entraînement et un learning rate de 0.02 qui est divisé par 10 aux pas 50k et 70k. Le programme court prévoit 40k pas d'entraînement et un learning rate de 0.02 qui est divisé par 10 aux pas 20k et 30k.

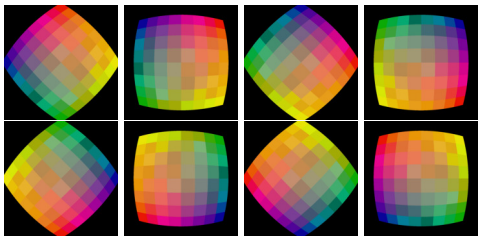
4 Résultats expérimentaux

4.1 Datasets d'évaluation

Les résultats de détection sont évalués sur le subset de validation de COCO2017 (ie. COCO2017val) ainsi que



(a) Ensemble de 35 transformations



(b) Ensemble de 8 transformations

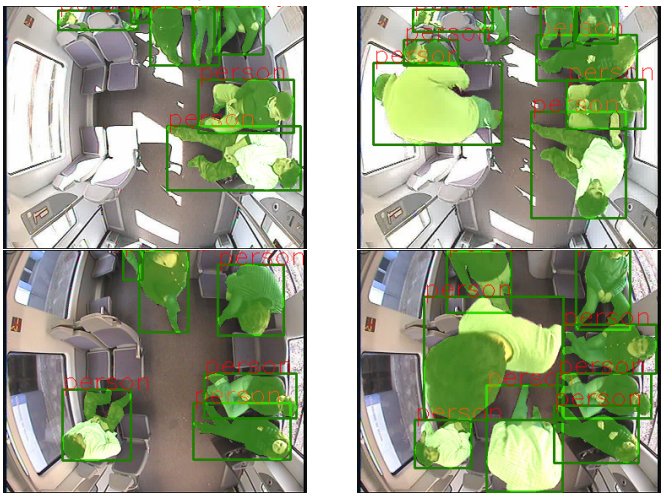
FIGURE 1 – Deux ensembles de transformations FE (Fisheye Effect)

trois datasets créés spécifiquement pour notre application : valBOSS, trainDoor et trainDoorAug, dans lesquels nous avons segmenté les personnes à l'aide de CVAT [14]. valBOSS contient 60 frames tirées de deux séquences du dataset BOSS [15] (filmées dans un train en mouvement avec une caméra grand angle avec distorsions en barillet et un background défilant). trainDoor contient 121 frames tirées d'un dataset conçu pour simuler le passage d'usagers à travers une porte de train, filmé avec une caméra à objectif fisheye. Pour constituer trainDoorAug, nous avons utilisé un renversement vertical (miroir vertical) sur trainDoor pour doubler le nombre d'exemples et obtenir 242 frames. Des exemples de ces datasets sont illustrés en Figure 2. Nous évaluons la performance de notre stratégie avec les outils officiels de MS COCO. Nous considérons les métriques suivantes [3] : APall (la précision moyenne pour différents seuils de confiance), AP50 (la précision pour un seuil de 50%), APL, APM, APS (la précision pour les grands, moyens et petits objets). Pour valBOSS, TrainDoor et trainDoorAug, seul le label "person" est considéré. Les scores sont obtenus sans crossvalidation, avec un seul entraînement.

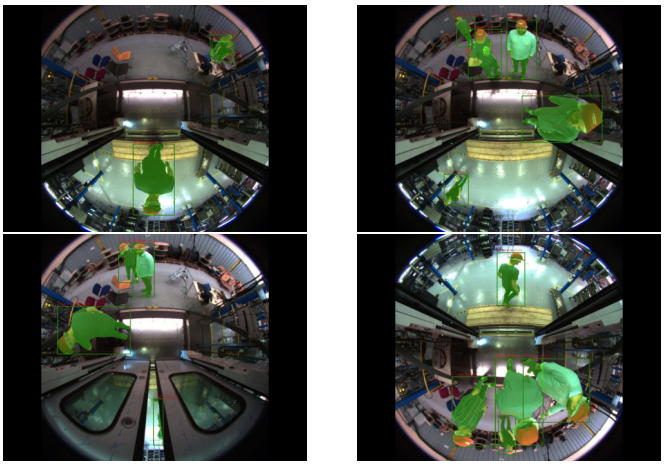
4.2 Pré-entraînement sur image rectilinéaire

Ainsi, nous avons entraîné deux modèles en utilisant l'augmentation FE. Le premier modèle est initialisé uniquement avec un backbone ImageNet [16]. Le deuxième est initialisé à partir de la référence Mask R-CNN pré-

(a) Images annotées du dataset COCO2017



(b) Images annotées du dataset valBOSS



(c) Images annotées du dataset trainDoor

FIGURE 2 – Images annotées des datasets COCO2017, valBOSS et trainDoor

entraînée sur MS COCO. Les performances sont évaluées sur COCO2017val avec augmentation FE car cela suffit à montrer l'intérêt du pré-entraînement. Dans le Tableau 1, nous montrons que le deuxième modèle gagne 4% en APall par rapport au premier sur COCO2017val augmenté avec 35 transformations FE. Nous en concluons que les apprentissages de Mask R-CNN en domaine rectilinéaire sont utiles en transfer learning dans le domaine FE. Dans le reste du papier, tous les modèles seront initialisés ainsi.

	APall	**AP50**	**AP75**	**APS**	**APM**	**APL**
ImageNet	0.12	0.22	0.12	0.05	0.15	0.20
COCO2017	**0.16**	**0.28**	**0.16**	**0.07**	**0.19**	**0.27**

TABLE 1 – Impact d'une initialisation avec backbone ImageNet vs modèle complètement entraîné sur COCO2017, évalué sur COCO2017val avec augmentation FE.

4.3 Ratio d'augmentation FE

En utilisant l'augmentation FE pour 100% des images, le modèle n'est plus exposé aux images rectilinéaires, ce qui diminue ses performances sur celles-ci. Afin d'utiliser l'augmentation FE tout en conservant de bonnes performances sur des images rectilinéaires, nous avons expérimenté différents ratios d'augmentation et un backbone ResNet50 sur COCO2017val. Les résultats sont illustrés sur la Figure 3. Nous pouvons observer une augmentation des performances sur les images fisheye et une diminution des performances sur les images rectilinéaires lorsque nous faisons varier le ratio d'augmentation FE. Nous considérons qu'un ratio de 50% est un bon compromis car il apporte une amélioration de 9% en domaine FE, et ne diminue les performances rectilinéaires que de 2%. Nous avons confirmé ce choix de ratio en l'utilisant pour entraîner un modèle Mask R-CNN avec backbone de type ResNet101 et en l'évaluant sur COCO2017val.

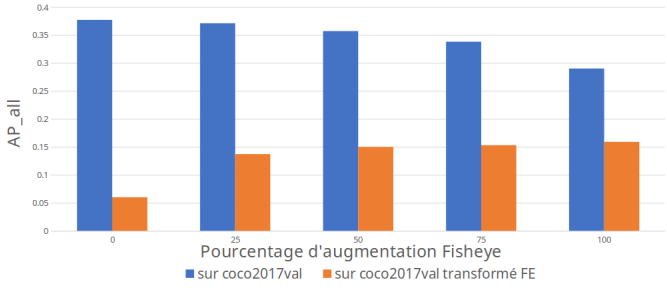


FIGURE 3 – Evolution de AP_all sur COCO2017val rectilinéaire (Bleu) et transformé FE (Orange) en fonction du ratio d'augmentation FE pendant l'entraînement.

Le Tableau 2 montre les performances sur COCO2017val d'un Mask R-CNN avec un backbone ResNet101 et une 50% d'augmentation FE . Ce modèle dont l'entraînement a été augmenté avec un ratio de 50% a obtenu 39.5% en APall contre 40.1% pour la référence. Cette différence est négligeable, ce qui confirme que le modèle conserve de bonnes performances en domaine rectilinéaire.

	APall	**AP50**	**AP75**	**APS**	**APM**	**APL**
Reference	40.1	61.9	44.0	22.6	43.6	52.6
50% FE aug	39.5	60.7	43.2	22.1	43.5	51.7

TABLE 2 – Performance d'un Mask R-CNN **resnet101** avec augmentation FE de 50% vs Reference sur COCO2017val

4.4 Evaluation sur datasets dédiées : valBOSS, trainDoor et trainDoorAug

Après avoir évalué les performances sur les images rectilinéaires de COCO2017, nous avons démarré une série de tests visant à confirmer les bonnes performances des paramètres choisis sur nos datasets dédiées : valBOSS, trainDoor et trainDoorAug. Le Tableau 3 illustre les résultats obtenus.

	APall	**AP50**	**AP75**	**APM**	**APL**
valBOSS					
Reference	18.0	43.9	12.1	**26.6**	10.0
50% FE aug	**18.0**	**46.0**	**12.8**	25.1	**15.1**
trainDoor					
Reference	55.8	78.5	66.3	49.7	59.9
50% FE aug	**70.3**	**90.6**	**83.6**	**59.1**	**77.6**
trainDoorAug					
Reference	43.5	64.7	50.4	31.5	52.2
50% FE aug	**67.0**	**90.7**	**79.0**	**54.7**	**74.5**

TABLE 3 – Performance d'un Mask R-CNN **resnet101** entraîné avec 50% d'augmentation FE vs Reference sur trois datasets FE

Nous pouvons remarquer une légère amélioration des performances avec la méthode proposée sur la dataset valBOSS, mais celle-ci n'est pas suffisamment significative pour pouvoir tirer des conclusions. A contrario, une réelle amélioration est constatée sur trainDoor et encore davantage trainDoorAug. Notons que ce dernier dataset est aussi plus difficile car l'augmentation de renversement vertical fait que la moitié des personnes présentes sont "à l'envers" (i.e. tête en bas, pieds en haut) et l'autre moitié "à l'endroit" (i.e. tête en haut, pieds en bas). L'orientation a une importance car les CNN ne sont pas invariants par rotation [17]. C'est vraisemblablement la raison pour laquelle la différence de performance entre la référence entraînée sur des personnes droites dans COCO2017 et le modèle entraîné avec augmentation FE (impliquant différentes rotations) est plus élevée sur trainDoorAug (voir Figure 4). Cette évaluation est donc biaisée en faveur de l'augmentation FE, qui contient des rotations.

FIGURE 4 – Première ligne : résultats d'inférence avec resnet101 Mask R-CNN. Deuxième ligne : résultats d'inférences avec resnet101 Mask R-CNN avec augmentation 35FE.

4.5 Augmentation FE vs renversement vertical

Afin de construire une référence plus "pertinente", nous avons ensuite entraîné un Mask R-CNN avec un backbone ResNet50 (et non plus le ResNet101 pour des questions de temps de traitement) et une augmentation de renversement vertical selon un ratio de 50%. Le Tableau 4 illustre les résultats obtenus sous le nom "halfvflip". Les résultats montrent qu'une augmentation de renversement vertical est suffisante pour améliorer les performances sur des images fisheye sur la dataset trainDoorAug. En effet, on observe une augmentation de 17% APall par rapport à la référence. De plus, l'utilisation d'une augmentation 35FE permet d'améliorer les performances par rapport au renversement vertical avec un avantage de 1.9% APall sur valBOSS et de 1.4% sur trainDoorAug.

	APall	**AP50**	**AP75**	**APM**	**APL**
valBOSS					
Reference	14.6	38.1	9.3	21.5	8.8
halfvflip	12.0	35.2	6.6	19.1	8.5
35 transformations	13.9	39.7	7.8	20.6	9.2
8 transformations	**22.8**	**52.8**	**17.8**	**29.6**	**9.5**
trainDoorAug					
Reference	43.6	67.5	50.5	31.5	51.0
halfvflip	60.6	86.4	69.4	**53.9**	65.3
35 transformations	61.0	87.6	71.7	51.2	67.0
8 transformations	**62.0**	**87.7**	**74.2**	51.4	**68.5**

TABLE 4 – Performance d'un Mask R-CNN **resnet50** entraîné avec 50% d'augmentation halfvflip ou FE (avec deux ensembles de transformations différents) vs Référence sur deux datasets FE

4.6 Choix des transformations

Étant donné les résultats précédents, nous nous sommes questionnés sur la nécessité d'utiliser un ensemble de 35 transformations complexes, sachant qu'une augmentation simple permet d'apporter une amélioration substantielle. En particulier, considérant les 35 transformations FE, la rotation semble être le paramètre le plus important. Nous avons donc entraîné un modèle avec une augmentation FE utilisant un nouvel ensemble de transformations contenant seulement 8 rotations différentes avec un effet fisheye minimal. Comme précédemment, nous évaluons ce modèle sur nos deux bases de données dédiées. Les résultats sont illustrés dans le Tableau 4. Nous pouvons observer une nette amélioration sur le dataset valBOSS et une amélioration légère sur le dataset trainDoorAug en comparaison avec les résultats de renversement vertical. Ces résultats montrent qu'une grande partie du défi associé aux images fisheye est dû aux différentes rotations d'objets qu'elles contiennent, et qu'il n'est pas nécessaire d'utiliser différents paramètres de transformation FE dans les augmentations. Ces résultats finaux sont illustrés qualitativement sur la Figure 5.

5 Conclusion

Nous proposons un moyen d'entraîner Mask R-CNN afin qu'il soit performant à la fois sur des images à angle de vue fisheye et rectilinéaires. Nous avons montré qu'une augmentation de données constituée de transformations FE peut accomplir cet objectif. Un petit ensemble de 8 rotations avec transformation FE est suffisant pour obtenir une large amélioration, et un ratio d'environ 50% d'augmentation de données est un bon compromis pour obtenir de bonnes performances à la fois sur les images rectilinéaires et fisheye, avec le même réseau de neurones. Nous avons confirmé cela sur deux datasets acquises en environnement transport et présentant des distorsions en barillet ou des effets fisheye. La méthode proposée peut être appliquée à de nombreux problèmes concrets nécessitant des détecteurs polyvalents capables de traiter des images fisheye. Nous prévoyons d'appliquer cette méthode à des tâches telles que le suivi de pose ou la reconnaissance d'actions d'usagers de transport dans la cadre du programme Train Autonome.

Remerciements

Ce travail de recherche est co-financé par l'union européenne avec le Fonds européen de développement régional (région Hauts-de-France) et le programme français "Investissements d'Avenir" et il s'inclut dans le projet collaboratif français TASV (Train Autonome Service Voyageurs), avec SNCF, Alstom Crespin, Thales, Bosch, et Spirops

Références

[1] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-CNN," in *2017 IEEE International Conference on*

Computer Vision (ICCV). IEEE, 2017, pp. 2980–2988.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn : Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 91–99.

[3] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO : Common objects in context," *ECCV*, 2014.

[4] 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," *CVPR*, 2016.

[5] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, P. Varley, D. O'Dea, M. Uricár, S. Milz, M. Simon, K. Amende *et al.*, "Woodscape : A multi-task, multi-camera fisheye dataset for autonomous driving," *arXiv preprint arXiv :1905.01489*, 2019.

[6] A. Saez, L. Bergasa, E. Lopez-Guillen, E. Romera, M. Tradacete, C. Gomez-Huélamo, and J. del Egido, "Real-time semantic segmentation for fisheye urban driving images based on ERFNet," *Sensors*, vol. 19, no. 3, p. 503, 2019.

[7] K. Yang, X. Hu, L. M. Bergasa, E. Romera, and K. Wang, "Pass : Panoramic annular semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 10, pp. 4171–4185, 2020.

[8] G. Blott, M. Takami, and C. Heipke, "Semantic segmentation of fisheye images," in *ECCV*. Springer, 2018, pp. 181–196.

[9] Mamoru Saito, Katsuhisa Kitaguchi, Gun Kimura, and Masafumi Hashimoto, "Human detection from fish-eye image by bayesian combination of probabilistic appearance models," in *2010 IEEE International Conference on Systems, Man and Cybernetics*, 2010, pp. 243–248.

[10] T. Wang, Y. Hsieh, F. Wong, and Y. Chen, "Mask-rcnn based people detection using a top-view fisheye camera," in *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2019, pp. 1–4.

[11] M. Christopher, "Laser-augmented omnidirectional vision for 3d localisation and mapping," Ph.D. dissertation, École Nationale Supérieure des Mines de Paris, 2009. [Online]. Available : https ://pastel.archives-ouvertes.fr/pastel-00004652

[12] E. Sanz-Ablanedo, J. R. Rodriguez-Pérez, J. Armesto, and M. F. A. Taboada, "Geometric stability and lens decentering in compact digital cameras," *Sensors*, vol. 10, no. 3, pp. 1553–1572, 2010.

[13] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https ://github.com/facebookresearch/detectron, 2018.

[14] N. Manovich and B. Sekachev, "Powerful and efficient computer vision annotation tool (cvat)," https ://github.com/opencv/cvat, 2019.

[15] S. A. Velastin and D. A. Gómez-Lira, "People detection and pose classification inside a moving train using computer vision," in *International Visual Informatics Conference*. Springer, 2017, pp. 319–330.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet : A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[17] D. Marcos, M. Volpi, and D. Tuia, "Learning rotation invariant convolutional filters for texture classification," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2012–2017.

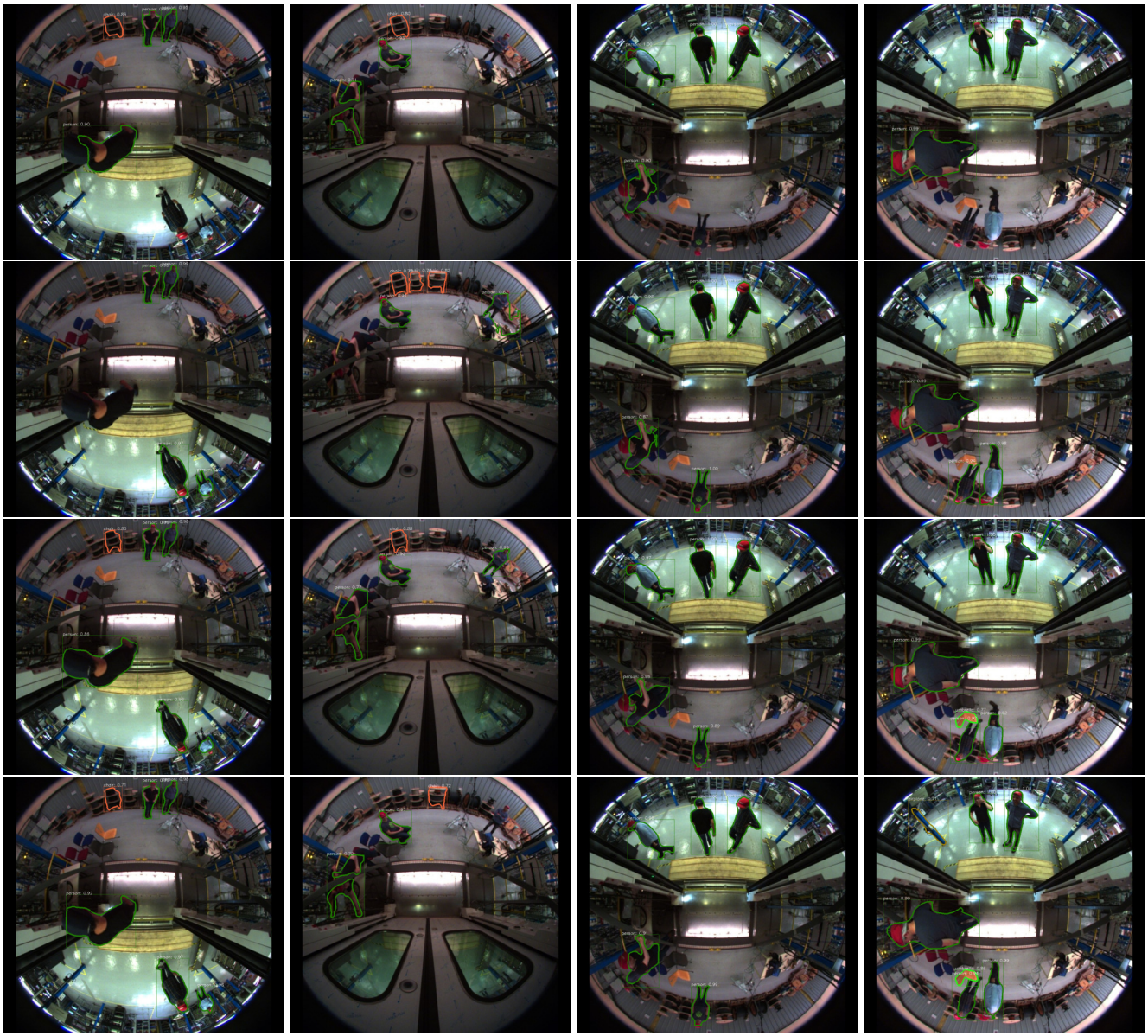


FIGURE 5 – Inférence sur des images du dataset trainDoorAug (Première ligne : résultats avec resnet50 Mask R-CNN avec augmentation de renversement vertical. Deuxième ligne : résultats avec resnet50 Mask R-CNN avec augmentation de 8FE transformations.