Bilinear Pooling For Event Camera Pose Estimation

Ahmed Tabia, F. Bonardi, S Bouchafa

Estimation de la pose d'une caméra événementielle par apprentissage profond

A. Tabia[1] F. Bonardi [1] S. Bouchafa [1]

[1] IBISC, Universite Paris-Saclay

ahmed.tabia@universite-paris-saclay.fr

Résumé

Les caméras événementielles sont des capteurs de vision bio-inspirés qui enregistrent la dynamique d'une scène tout en filtrant les données inutiles. De nombreuses méthodes classiques d'estimation de la pose ont été remplacées par des approches de relocalisation de caméra basées sur des réseaux de neurones convolutifs (CNN) et des réseaux de neurones à mémoire à court terme (LSTM) dans l'étude des systèmes de localisation simultanée et de cartographie. Cependant, et en raison de l'utilisation de la couche LSTM, ces méthodes sont faciles à sur-ajuster et prennent généralement beaucoup de temps pour converger. Dans cet article, nous présentons une nouvelle méthode pour estimer la pose 6DOF d'une caméra événementielle à l'aide d'un apprentissage en profondeur.
Notre approche commence par le traitement des événements et la génération d'un ensemble d'images. Elle utilise ensuite deux CNN pour extraire les caractéristiques pertinentes des images générées. Ces caractéristiques sont multipliées à chaque emplacement de l'image à l'aide d'un produit extérieur et regroupées sur les emplacements. Le modèle se termine par une couche de régression qui renvoie la position et l'orientation estimées de la caméra événementielle.

Mots Clef

6-DOF, Estimation De La Pose, Apprentissage Profond, Caméra Événementielle.

Abstract

Event cameras are a type of vision sensor that can capture the dynamics of a scene while filtering out irrelevant data. In recent years, traditional pose estimation methods have been replaced by camera relocalization approaches that rely on deep learning techniques such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) in simultaneous localization and mapping systems. However, these approaches often suffer from overfitting and slow convergence due to the use of LSTM layers. To address this issue, we propose a novel method for estimating the 6 Degrees of Freedom (6DOF) pose of an event camera using a deep learning approach that does not rely on LSTM layers. Our method involves processing the event data to generate a set of images, followed by feature extraction using two CNNs. The extracted features are then combined using bilinear pooling, which calculates the outer product of the feature vectors at each location of the image and pools them across locations. Finally, a regression layer is used to predict the position and orientation of the event camera.

Keywords

6-DOF, Pose Estimation, Deep Learning, Event-based Camera.

1 Introduction

La relocalisation de la pose de la caméra, qui vise à inférer la position et l'orientation de la caméra à partir d'une scène observée [17], est un problème fondamental dans de nombreuses applications de vision par ordinateur, telles que la conduite de véhicules autonomes, la robotique, la réalité augmentée et la localisation visuelle des piétons.
Les méthodes conventionnelles de relocalisation de la vision par ordinateur peuvent être classées en deux catégories : (1) les approches géométriques et (2) les approches basées sur l'apprentissage. Les approches géométriques [14] sont principalement basées sur la correspondance de caractéristiques locales. Leur processus standard consiste à extraire un ensemble de caractéristiques locales d'une image donnée et à effectuer une correspondance 2D-3D avec des points 3D correspondants, puis à calculer la pose de la caméra de six degrés de liberté en utilisant généralement des algorithmes de perspective-n-point [9]. Cette catégorie d'approches repose fortement sur le processus d'extraction et de correspondance de caractéristiques précises, ce qui n'est pas toujours satisfait, en particulier dans le cas de variations d'éclairage [10].
Plus récemment, avec la résurgence de l'apprentissage profond, notamment des réseaux de neurones convolutionnels (CNN), de nombreuses applications de vision par ordinateur ont été revisitées avec des approches axées sur les données. Les nouvelles méthodes ont atteint des performances élevées dans différentes tâches telles que la reconnaissance d'objets [4], la classification d'images [12] et la segmenta-

tion [1]. Les approches basées sur l'apprentissage profond ont montré une grande capacité à extraire des caractéristiques robustes [7], cependant, elles nécessitent une grande quantité de données d'entraînement (généralement des milliers d'images) et de nombreuses ressources informatiques (GPU puissants et coûteux). C'est pourquoi les approches qui ne nécessitent pas une recomputation à chaque fois sur de telles données étendues pourraient être plus intéressantes. Par exemple, le transfert d'apprentissage et l'intégration pourraient être des alternatives intéressantes pour résoudre ce problème.

En outre, les deux catégories de méthodes conventionnelles de relocalisation de caméras souffrent encore des changements d'éclairage, du flou et des images plates dans lesquelles il est difficile d'extraire des caractéristiques. Ces problèmes sont principalement liés à la nature des images d'entrée capturées par les caméras conventionnelles et alimentées dans ces méthodes.

Les caméras événementielles, également appelées caméras neuromorphiques, sont des capteurs d'image qui réagissent aux changements locaux de luminosité. Contrairement aux caméras conventionnelles, la sortie brute de ces caméras est une séquence d'événements asynchrones (changements discrets de luminosité par pixel) correspondant à des changements dans l'éclairage de la scène. Ces caméras présentent plusieurs avantages par rapport aux caméras conventionnelles. Elles ont une résolution temporelle élevée, une gamme dynamique étendue et aucun flou de mouvement. Ces avantages rendent l'utilisation des caméras événementielles idéale dans le contexte des applications robotiques, en particulier l'estimation de la pose.

S'appuyant sur ces avantages, Rebecq et al. [18] ont présenté une méthode qui combine les informations de l'IMU et de l'événement pour estimer la pose de la caméra 6DOF. Plus récemment, une méthode appelée (SP-LSTM) a été proposée pour estimer le 6DOF d'une caméra événementielle par [15]. La méthode utilise une achitecture VGG16 [20] formée à partir de zéro avec l'algorithme de descente de gradient stochastique et deux couches LSTM spatiales empilées. La méthode donne des résultats prometteurs dans l'estimation de la pose de la caméra, mais nécessite un temps d'apprentissage coûteux en raison du recyclage du modèle de réseau complet avec la couche LSTM.

Dans cet article, nous présentons une nouvelle méthode qui résout les problèmes des couches LSTM. Nous proposons un modèle d'apprentissage en profondeur composé de deux CNN conçus pour extraire les caractéristiques pertinentes des images événementielles. Les caractéristiques extraites sont ensuite agrégées en utilisant le produit extérieur à chaque emplacement de l'image et sont ensuite regroupées à l'aide d'une opération de regroupement bilinéaire [11]. Nous tirons également parti des nouveaux développements en matière d'apprentissage en profondeur et utilisons l'optimiseur ADAM [8] ainsi que la fonction d'activation ELU [2]. L'optimiseur ADAM et la fonction d'activation ELU contribuent à une meilleure convergence et à une amélioration des performances du modèle d'apprentissage profond. Nous avons mené des expériences sur différents ensembles de données et rapporté des résultats supérieurs par rapport aux méthodes de l'etat de l'art.

2 Méthode Proposée

Nous avons proposé un nouveau réseau neuronal convolutif entièrement basé sur l'attention pour l'estimation de la pose. Le mécanisme d'attention proposé aide le modèle à se concentrer sur les régions d'images pertinentes pour le mouvement. L'utilisation du pooling bilinéaire dans le contexte de l'estimation de la pose d'une caméra événementielle peut contribuer à améliorer les performances de l'algorithme en combinant les caractéristiques d'une manière qui accroît leur pouvoir de discrimination et leur robustesse au bruit.
Étant donné une séquence d'événements capturée à l'aide d'une caméra événementielle, nous traitons d'abord l'événement brut et créons un ensemble d'images d'événements selon [15]. La prétraitement des images est présenté dans la section 2.1. Une fois que les événements sont convertis en images, comme illustré dans la figure 1, ils sont entrés dans un réseau de neurones convolutifs. Les caractéristiques extraites sont ensuite agrégées en utilisant le regroupement bilinéaire [11] pour l'estimation de la pose. Les détails sur l'architecture du modèle sont présentés dans la section 2.2.

2.1 Prétraitement d'image

Contrairement aux caméras classiques basées sur des images, qui capturent une image entière à un intervalle de temps prédéterminé, les caméras événementielles ne capturent qu'un seul événement à un instant donné en fonction des changements de luminosité au niveau d'un pixel local. La première étape de ce projet consiste à résoudre le problème de la relocalisation de pose, inspiré par [15][7][6]. Pour cela, nous avons transformé le flux d'événements en une image événementielle $I \in \mathbb{R}^{h*w}$, où h et w sont les dimensions de l'image événementielle. Formellement, un événement e est un tuple représenté par

$$e =< e_t, (e_x, e_y), e_p >$$

où e_t est le moment de l'événement, (e_x, e_y) sont les coordonnées du pixel et $e_p = \pm 1$ est la polarité qui indique le changement de luminosité au niveau du pixel courant. L'image événementielle est calculée à partir du flux d'événements comme suit :

$$I(e_x, e_y) = \begin{cases} 0 \;\; si \; e_p = -1 \\ 1 \;\; si \; e_p = \;\; 1 \end{cases} \tag{1}$$

La deuxième étape consiste à agrandir l'image pour qu'elle ait une taille de 224×224 pixels, en respectant le rapport d'aspect de l'image d'origine, et à la donner en entrée aux CNN. La figure 1 montre un exemple d'images d'événements obtenues après le prétraitement du flux d'événements [5]. L'étape de prétraitement joue un rôle important

car elle affecte la qualité des images d'événements, qui sont utilisées pour entraîner les CNN et estimer la relocalisation de la caméra.

2.2 Le modèle de réseau

Dans notre méthode, nous proposons d'extraire différents ensembles de caractéristiques de l'image d'événements. Nous utilisons deux réseaux de neurones convolutifs, notés respectivement A et B (voir la Figure 2). Deux cartes de caractéristiques sont extraites à partir des réseaux A et B, qui appliquent plusieurs transformations de pooling et de non-linéarité à l'image d'événements originale. L'intuition est que A et B apprennent différentes caractéristiques de l'image d'entrée. Ensuite, les sorties de A et B sont combinées par une couche de pooling bilinéaire. Cette couche fournit une représentation puissante qui fusionne les deux ensembles de caractéristiques en exploitant les informations d'ordre supérieur capturées sous la forme de corrélations par paires entre les caractéristiques extraites. Dans nos expériences, nous utilisons le modèle MobileNetV2 pré-entraîné [19] comme premier extracteur de caractéristiques A et VGG16 [20] comme second extracteur de caractéristiques B. Les MobileNetV2 et VGG16 utilisés ont déjà été entraînés sur une très grande collection d'images provenant d'ImageNet [3] et ont obtenu d'excellents résultats en relocalisation et en classification d'images.

Nous notons la pose de la caméra événementielle par $y = [p,q]$, où $p \in \mathbb{R}^3$ représente la position de la caméra en trois dimensions et le quaternion $q \in \mathbb{R}^4$ code l'orientation de la caméra. Réseaux de neurones en parallèle peut offrir des avantages tels que l'extraction de caractéristiques complémentaires, la fusion des informations multi-échelles, l'augmentation de la capacité du modèle et, éventuellement, la réduction du coût computationnel. L'utilisation de deux réseaux de neurones peut offrir un bon compromis entre la complexité du modèle et la performance obtenue. Ajouter davantage de réseaux de neurones en parallèle augmenterait la complexité du modèle, ce qui pourrait entraîner une augmentation du coût computationnel et un risque accru de surapprentissage. De plus, il se peut que les gains de performance ne soient pas significatifs avec l'ajout de réseaux supplémentaires.

Dans nos expériences, les CNN utilisés ont été pré-entraînés sur l'ensemble de données ImageNet [3] avec des dimensions d'entrée de $224 \times 224 \times 3$. Des caractéristiques profondes sont apprises à partir des images d'événements d'entrée obtenues à partir de l'étape de prétraitement. Les cartes de caractéristiques de sortie des réseaux A et B sont respectivement représentées par la matrice V de dimensionnalité $n \times d$ et la matrice U de taille $m \times d$. Ici, n et m sont le nombre de noyaux dans les couches de sortie des réseaux A et B, respectivement. La dimensionnalité de chaque filtre est d ; elle est obtenue en aplatissant la carte de caractéristiques 2D, c'est-à-dire l'image de sortie qui a subi plusieurs convolutions de noyaux et des transformations de pooling. L'opération de pooling bilinéaire est ensuite définie comme suit :

$$X = UV^T, U \in \mathbb{R}^{m*d}, V \in \mathbb{R}^{n*d}, X \in \mathbb{R}^{m*n} \tag{2}$$

La connexion entre les sorties du CNN et la bilinéaire pooling est précédée d'une fonction d'activation ELU [2] $F(x)$. Elle est définie comme suit :

$$F(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x <= 0 \end{cases} \tag{3}$$

Pour régresser le vecteur de pose à sept dimensions, une couche de régression linéaire est ajoutée à la fin du modèle (voir Figure 2, la couche de sortie).

Paramètres d'entraînement et fonction de perte

Dans notre méthode, nous entraînons notre modèle en utilisant l'optimiseur Adam [8] (avec les paramètres $\beta_1 = 0,9$ et $\beta_2 = 0,999$) pour obtenir de hautes performances en calculant le taux d'apprentissage adaptatif de chaque hyperparamètre, et pour éviter la redondance et obtenir une mise à jour de gradient plus rapide.

Nous choisissons la perte $l1$ lisse au lieu de la fonction de perte des moindres carrés dans notre implémentation $l1$ lisse est moins sensible aux valeurs aberrantes que la perte $l2$. Cela est dû au fait que la perte $l2$ élève au carré les erreurs, ce qui amplifie l'impact des valeurs aberrantes sur la fonction de coût. La perte $l1$ lisse, en revanche, utilise une combinaison de la perte $l1$ (erreurs absolues) et de la perte $l2$ (erreurs quadratiques), ce qui lui permet de réduire l'impact des valeurs aberrantes sur l'optimisation du modèle.. La perte $l1$ lisse sur n échantillons est définie comme suit :

$$smoothl1(x,y) = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{4}$$

ou z_i est donné par :

$$z_i = \begin{cases} 0.5(x_i - y_i)^2/\beta, & \text{if } |x_i - y_i| < \beta \\ |x_i - y_i| - 0.5 * \beta, & \text{autrement} \end{cases} \tag{5}$$

où x et y sont les vecteurs de pose de la vérité terrain et de la caméra cible, respectivement. β est un paramètre optionnel qui spécifie le seuil auquel changer entre la perte l1 et la perte l2. Lorsque β varie, la partie de la perte l1 a une pente constante de 1. Dans notre implémentation, nous avons fixé β à 1.

Suivant le travail de Kendall et al. [7], lors de la phase de test, nous normalisons le quaternion pour qu'il ait une longueur unitaire et utilisons la distance euclidienne pour évaluer la différence entre deux quaternions. En théorie, la distance devrait être mesurée dans l'espace sphérique, mais en réalité, le réseau profond produit un quaternion prédit $\widehat{q}$ qui est suffisamment proche du quaternion de vérité terrain q. Cela rend la différence entre la distance sphérique et la distance euclidienne insignifiante.

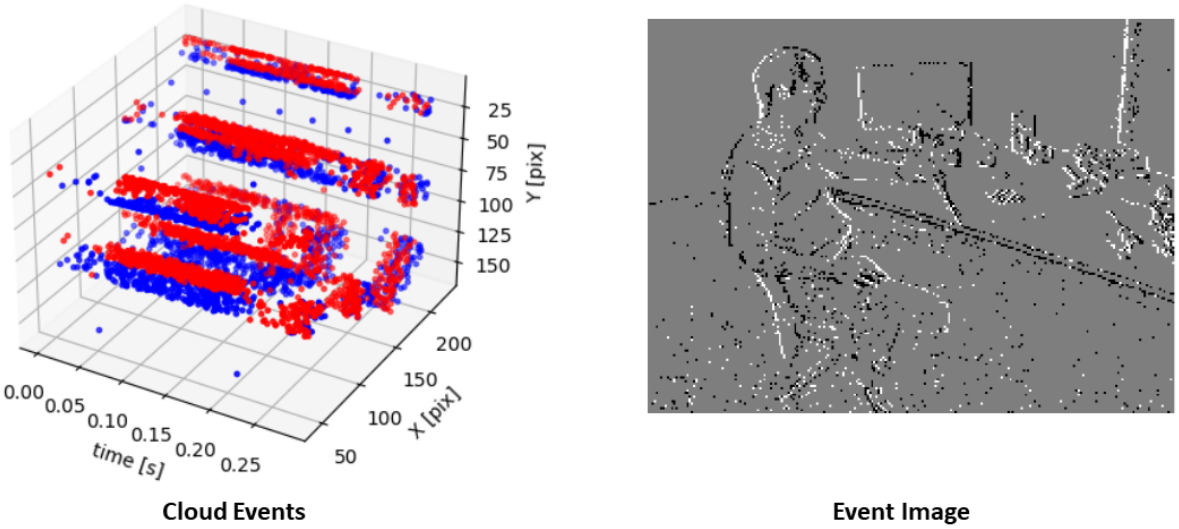

FIGURE 1 – Prétraitement d'images à partir de nuages de points vers des images d'événements.

3 Résultats expérimentaux

La méthode proposée a été évaluée sur une collection de six jeux de données réels d'événements. Dans cette section, nous présentons les ensembles de données utilisés pour l'évaluation de la méthode, l'environnement d'entraînement et les résultats expérimentaux.

3.1 Base de données

Nous avons mené des expériences sur l'ensemble de données d'une caméra événementielle collectées par [13]. L'ensemble de données comprend une collection de scènes capturées par une DAVIS240C dans des mini-laboratoires. Il contient le nuage d'événements, des images, des mesures IMU et l'étalonnage de la caméra de la DAVIS. Les poses de caméra réelles sont collectées à partir d'un système de capture de mouvement avec une précision submillimétrique à 200Hz. Nous adoptons la marque temporelle du système de capture de mouvement pour construire les images d'événements. Tous les événements avec des marques temporelles entre t et $t+1$ du système de capture de mouvement sont regroupés en une seule image d'événements. Sans perte de généralité, nous considérons la pose de caméra réelle de cette image d'événements comme la pose de caméra prise par le système de capture de mouvement à l'instant $t+1$. Cette méthode limite techniquement la vitesse de la caméra événementielle à la vitesse du système de capture de mouvement. Nous suivons le même protocole d'évaluation que dans [15]. Le protocole comprend deux types de séparations :

— **Split Aléatoire** où nous sélectionnons 70% des images d'événements pour l'entraînement et les 30% restants pour les tests.

— **Nouveau Split** dans laquelle nous sélectionnons les 70% premiers de chaque événement pour l'entraînement, puis le reste 30% de l'événement pour le test. De cette manière, nous avons deux séquences indépendantes sur la même scène. En suivant Nguyen et al. [15], la séquence d'entraînement est sélectionnée à partir de la marque temporelle t_0 jusqu'à t_{70}, et la séquence de test est de la marque temporelle t_{71} jusqu'à t_{100}).

3.2 Environnement d'entraînement

Une fois la phase de prétraitement terminée, des patchs de taille 224×224 pixels sont extraits de chaque image et introduits dans les réseaux de neurones convolutifs sous forme de jeu de données de patchs. Pour évaluer l'efficacité de notre méthode proposée dans cet article, nous avons mené plusieurs expériences et comparé nos résultats à ceux d'architectures d'apprentissage profond utilisant des modèles à la pointe de la technologie utilisant des LSTM. Nous avons utilisé Pytorch [16] pour implémenter la méthode proposée. Toutes nos expériences ont été réalisées sur une plate-forme composée d'un processeur Intel(R) Xeon(R) CPU @ 2,00GHz, d'une mémoire RAM de 24 Go et d'un seul GPU Tesla T4. Les réseaux ont été entraînés avec 350 époques avec un taux d'apprentissage fixé à $2\exp{-3}$ avec une décroissance de momentum de $4\exp{-3}$ et un poids de pénalisation fixé à 0.

3.3 Résultats

Nous utilisons le même protocole de comparaison rapporté dans [15] et utilisé dans PoseNet [7] et Bayesian PoseNet [6].
En tant qu'évaluation quantitative, nous avons choisi de

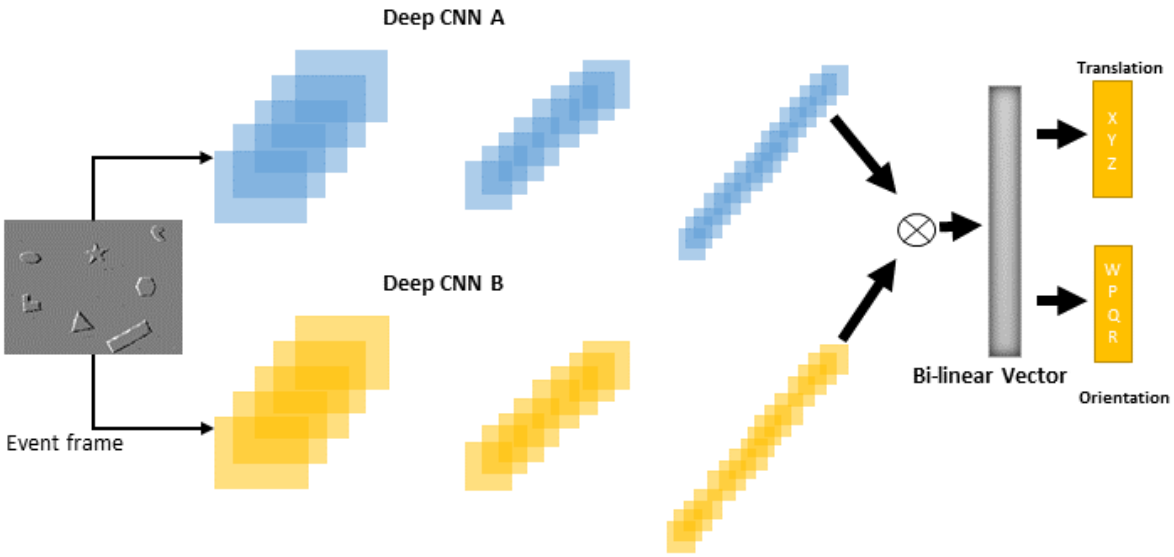

FIGURE 2 – Notre méthode de relocalisation de la pose en 6DOF pour les caméras à événements est présentée ci-dessous. Nous commençons par créer une image événementielle à partir du flux d'événements. Ensuite, nous extrayons des caractéristiques de l'image événementielle créée à l'aide d'un pooling bilinéaire. Le vecteur de caractéristiques est ensuite transmis à une couche entièrement connectée de sept neurones pour régresser le vecteur de pose de la caméra.

	PoseNet [7]		**Bayesian PoseNet** [6]		**SP-LSTM** [15]		**Ours**	
	Median Error	Average Error	Median Error	Average Error	Median Error	Average Error	Median Error	Average Error
shapes rotation	0.109m, 7.388°	0.137m, 8.812°	0.142m, 9.557°	0.164m, 11.312°	0.025m, 2.256°	0.028m, 2.946°	0.018m, 1.753°	0.020m, 2.551°
shapes translation	0.238m, 6.001°	0.252m, 7.519°	0.264m, 6.235°	0.269m, 7.585°	0.035m, 2.117°	0.039m, 2.809°	0.033m, 2.211°	0.036m, 2.717°
box translation	0.193m, 6.977°	0.212m, 8.184°	0.190m, 6.636°	0.213m, 7.995°	0.036m, 2.195°	0.042m, 2.486°	0.029m, 1.507°	0.032m, 1.693°
dynamic 6dof	0.297m, 9.332°	0.298m, 11.242°	0.296m, 8.963°	0.293m, 11.069°	0.031m, 2.047°	0.036m, 2.576°	0.027m, 1.802°	0.029m, 2.394°
hdr poster	0.282m, 8.513°	0.296m, 10.919°	0.290m, 8.710°	0.308m, 11.293°	0.051m, 3.354°	0.060m, 4.220°	0.040m, 2.937°	0.051m, 3.783°
poster translation	0.266m, 6.516°	0.282m, 8.066°	0.264m, 5.459°	0.274m, 7.232°	0.036m, 2.074°	0.041m, 2.564°	0.036m, 2.045°	0.039m, 2.315°
Average	0.231m, 7.455°	0.246m, 9.124°	0.241m, 7.593°	0.254m, 9.414°	0.036m, 2.341°	0.041m, 2.934°	**0,030m, 2.204°**	**0.034m, 2.708°**

TABLE 1 – Comparaison entre les résultats de notre méthode et les résultats de PoseNet [7], Bayesian PoseNet [6] and SP-LSTM [15]. L'évaluation est réalisée à l'aide du protocole de split aléatoire.

	PoseNet [7]		**Bayesian PoseNet** [6]		**SP-LSTM** [15]		**Ours**	
	Median Error	Average Error	Median Error	Average Error	Median Error	Average Error	Median Error	Average Error
shapes rotation	0.201m, 12.499°	0.214m, 13.993°	0.164m, 12.188°	0.191m, 14.213°	0.045m, 5.017°	0.049m, 11.414°	0.050, 3.681°	0.053m, 6.823°
shapes translation	0.198m, 6.969°	0.222m, 8.866°	0.213m, 7.441°	0.228m, 10.142°	0.072m, 4.496°	0.081m, 5.336°	0.062m, 4.554°	0.068m, 5.854°
shapes 6dof	0.320m, 13.733°	0.330m, 18.801°	0.326m, 13.296°	0.329m, 18.594°	0.078m, 5.524°	0.095m, 9.532°	0.071m, 5.787°	0.091m, 7.550°
Average	0.240m, 11.067°	0.255m, 13.887°	0.234m, 10.975°	0.249m, 14.316°	0.065m, 5.012°	0.075m, 8.761°	**0.061m, 3.448°**	**0.070m, 6.742°**

TABLE 2 – Comparaison entre les résultats de notre méthode et les résultats de PoseNet [7], Bayesian PoseNet [6] et SP-LSTM [15]. L'évaluation est réalisée à l'aide du nouveau protocole de split.

calculer l'erreur médiane et l'erreur moyenne de la pose prédite en position et en orientation. La distance euclidienne est utilisée pour comparer la position prédite avec la position réelle, et l'orientation anticipée est normalisée à une longueur unitaire avant d'être comparée à la vérité terrain. Pour la position et l'orientation, l'erreur médiane et l'erreur moyenne sont enregistrées en m et deg(°), respectivement.

Comparaison avec les méthodes de l'état de l'art. Nous présentons les résultats de comparaison entre notre méthode expliquée dans la section 2.2 et les modèles de l'état de l'art, à savoir PoseNet [7], Bayesian PoseNet [6] et SP-LSTM [15] utilisant CNN et LSTM.

Split aléatoire. Les résultats présentés dans le tableau 1 ont été obtenus en utilisant la stratégie de répartition aléatoire. Nous avons utilisé 6 séquences (**shapes rotation, box translation, shapes translation, dynamic 6dof, hdr poster, poster translation**) pour cette expérience. Dans toutes ces séquences, notre modèle obtient les erreurs médianes et moyennes les plus faibles. Il atteint une erreur médiane moyenne de $0,030$m et $2,204°$ sur l'ensemble des séquences du dataset réel, tandis que la méthode la plus récente, SP-LSTM, donne des résultats de $0,036$m, $2,341°$, les résultats de PoseNet et Bayesian PoseNet étant respectivement de $0,231$m, $7,455°$ et $0,241$m, $7,593°$.

Nouveau split. Le tableau 2 présente les résultats de comparaison de le nouveau split présentée dans la section 3.1. On peut remarquer que la nouveau split est plus difficile à gérer que la répartition aléatoire car les erreurs de toutes les méthodes sont plus grandes que les erreurs rapportées avec la répartition aléatoire. Nous utilisons trois séquences de la scène de formes(**shapes rotation, shapes translation, shapes 6dof**) dans ce nouveau split. Les résultats de notre méthode sont supérieurs aux résultats obtenus avec les méthodes de pointe. Nous obtenons une erreur médiane moyenne de $0,061$m et $3,448°$ sur l'ensemble des séquences du jeu de données réel, tandis que la méthode la plus récente, SP-LSTM, donne des erreurs de $0,065$m, $5,012°$, et $0,240$m, $11,067°$ et $0,234$m, $10,975°$ pour les erreurs de PoseNet et de Bayesian PoseNet, respectivement.

Nous rappelons que dans la nouvelle répartition, l'ensemble de test est sélectionné à partir des dernières 30% des images événementielles. Cela signifie que nous n'avons pas de relation de "voisinage" entre les images de test et d'entraînement. Dans la répartition aléatoire, les images de test peuvent être très proches des images d'entraînement car nous sélectionnons les images au hasard dans l'ensemble de la séquence pour l'entraînement/test.

En conclusion, les résultats expérimentaux approfondis des répartitions aléatoire et nouvelle montrent que notre méthode relocalise avec succès la pose de la caméra événementielle en utilisant uniquement l'image événementielle provenant du nuage de polarité. La raison critique de l'amélioration est l'utilisation du pooling bilinéaire pour apprendre les caractéristiques de relation spatiale dans l'image événementielle. Les expériences utilisant le nouveau split confirment également que notre approche encode avec succès la géométrie de la scène lors de l'entraînement et généralise bien lors des tests. De plus, notre réseau a également un temps d'inférence rapide et ne nécessite que l'image événementielle en entrée pour relocaliser la pose de la caméra.

4 Conclusion

Dans cet article, nous présentons une nouvelle méthode pour estimer la pose 6DOF d'une caméra événementielle avec un apprentissage en profondeur. Tout d'abord, les événements sont prétraités pour construire des images événementielles. Ensuite, un ensemble de caractéristiques est extrait à l'aide d'un réseau de neurones convolutif en profondeur avec un pooling bilinéaire. Ces caractéristiques extraites sont agrégées et alimentent une couche unique connectée à une couche entièrement connectée pour la régression de la pose. Pour notre apprentissage, nous avons utilisé l'optimiseur Adam au lieu de la descente de gradient stochastique classique. Nous avons également utilisé des fonctions d'activation ELU. De plus, notre méthode a un temps d'inférence rapide et ne nécessite que l'image événementielle pour relocaliser la pose de la caméra. Les résultats sur des ensembles de données disponibles publiquement montrent que notre approche généralise bien et surpasse les travaux récents, y compris les architectures basées sur les LSTM.

Annexe

Merci de votre participation.

Références

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet : A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12) :2481–2495, 2017.

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv :1511.07289*, 2015.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[4] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687. IEEE, 2015.

[5] Guillermo Gallego and Davide Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, 2(2) :632–639, 2017.

[6] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.

[7] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet : A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[8] Diederik P Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

[9] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp : An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2) :155–166, 2009.

[10] Ming Li, Ruizhi Chen, Xuan Liao, Bingxuan Guo, Weilong Zhang, and Ge Guo. A precise indoor visual positioning approach using a built image feature database and single user image from smartphone cameras. *Remote Sensing*, 12(5) :869, 2020.

[11] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015.

[12] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196, 2018.

[13] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator : Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2) :142–149, 2017.

[14] Raul Mur-Artal and Juan D Tardós. Orb-slam2 : An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5) :1255–1262, 2017.

[15] Anh Nguyen, Thanh-Toan Do, Darwin G Caldwell, and Nikos G Tsagarakis. Real-time 6dof pose relocalization for event cameras with stacked spatial lstm networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch : An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[17] Chao Qu, Shreyas S Shivakumar, Ian D Miller, and Camillo J Taylor. Dsol : A fast direct sparse odometry scheme. *arXiv preprint arXiv :2203.08182*, 2022.

[18] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. 2017.

[19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2 : Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.