Rapport de TP analyse vidéo

Auteurs : Abderahim LAGRAOUI & Hamza BAAKILI

Table des matières

1 Introduction

Pour l'analyse vidéo, différentes approches peuvent être envisagées. Historiquement, une méthode « classique » telle que l'analyse en composantes principales (PCA) dans OpenCV a été utilisée pour extraire des caractéristiques à partir de séquences d'images, en se basant sur des transformations linéaires simples et rapides.

Aujourd'hui, grâce au deep learning, des méthodes plus récentes comme VideoFlow offrent une modélisation plus précise des dynamiques temporelles des vidéos. En utilisant des réseaux de neurones profonds pour apprendre des flux optiques complexes, VideoFlow permet d'améliorer la prédiction de frames futures et l'extraction d'informations spatiales et temporelles, dépassant souvent les performances des approches classiques comme la PCA.

2 Méthode classique

3 Méthode avec Deep Learning

3.1 VideoFlow

VidéoFlow est une méthode avancée pour le calcul du flux optique entre plusieurs images, exploitant des indices temporels pour estimer des flux optiques plus précis. Cette approche repose sur deux modes principaux : **MOF (Multi-frame Optical Flow)** et **BOF (Bi-frame Optical Flow)**. Le choix entre ces deux modes dépend du contexte des données d'entrée : MOF est adapté pour des séquences contenant plusieurs images, tandis que BOF est conçu pour des séquences limitées à trois images.

La méthode d'inférence avec VidéoFlow se décompose en plusieurs étapes clés, comme illustré dans le fichier `inference.py`.

L'inférence est effectuée dans les modes suivants :

— En mode **MOF** (Multi-frame Optical Flow), les séquences d'images sont traitées par lots pour optimiser l'utilisation de la mémoire et améliorer la vitesse d'exécution.

— En mode **BOF** (Bi-frame Optical Flow), l'inférence est simplifiée et optimisée pour des séquences contenant seulement trois images.

3.2 Commandes pour l'Inférence

Les étapes décrites ci-dessus sont orchestrées par les commandes suivantes :

```
python -u inference.py --mode MOF --seq_dir <chemin_images>
--vis_dir <chemin_sortie>
```

Pour des séquences de trois images, il est recommandé d'utiliser le mode BOF :

```
python -u inference.py --mode BOF --seq_dir <chemin_images>
--vis_dir <chemin_sortie>
```

3.3 Exemples de prédictions

Voila un exemple d'une inférence sur un jeu de données sur la figure 1.

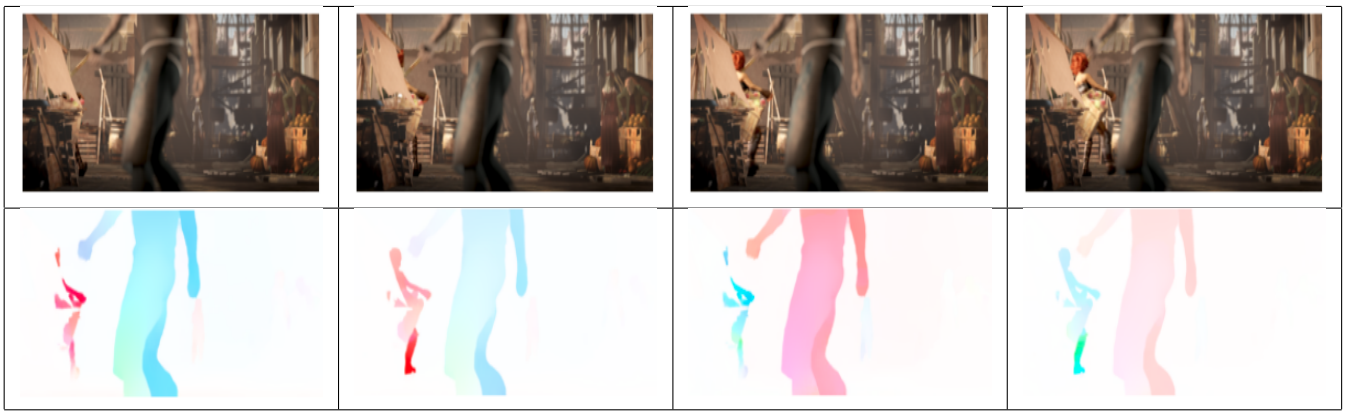

FIGURE 1 – Exemple de prédictions

4 Modules ajoutés au dépôt

4.1 Conversion du flow vers une image

Ce module Python permet de lire, convertir et visualiser les données de flux optique stockées dans des fichiers .flo. La fonction read_flo_file lit un fichier .flo et extrait les données de flux optique, les retournant sous forme de tableau NumPy avec une forme (hauteur, largeur, 2), où la troisième dimension contient les composantes du flux horizontal et vertical. La fonction flow2img convertit les données de flux optique en image en utilisant l'espace colorimétrique HSV, où la magnitude et la direction du flux sont encodées respectivement dans les canaux de teinte et de saturation. La fonction convert_flo_to_png parcourt tous les fichiers .flo d'un répertoire d'entrée, les convertit en images avec flow2img, puis enregistre les résultats au format .png dans un répertoire de sortie. Le module utilise matplotlib pour afficher les images converties et gère les exceptions lors du processus de conversion. Le script peut être exécuté en ligne de commande en prenant les chemins des répertoires d'entrée et de sortie en arguments.

4.2 Visualisation

Ce module permet de visualiser et de naviguer à travers trois ensembles d'images (directes, inversées et réelles) en utilisant Matplotlib. La classe FrameNavigator permet de charger, afficher et naviguer entre les frames via des clics de souris : le clic gauche fait avancer les images et le clic droit les recule. Les images sont chargées à partir de répertoires spécifiés et affichées côte à côte. Ce module est expliqué en details sur la section de l'interface graphique.

4.3 Évaluation des méthodes

Ce module permet de calculer et de comparer différentes métriques (aEPE, aAE, et MSE) entre des flux optiques (représentés par des fichiers .flo ou des images). Il inclut des fonctions pour lire les fichiers .flo, calculer les différences entre deux flux optiques, et évaluer les erreurs moyennes (aEPE, aAE) et l'erreur quadratique moyenne (MSE). Il prend en charge l'analyse de trois jeux de données différents, en utilisant des répertoires spécifiés pour chaque flux optique, et génère un histogramme des résultats pour une visualisation facile. Enfin, le script permet de traiter les données depuis la

ligne de commande en spécifiant les répertoires des fichiers et la méthode à utiliser ('dl' ou 'classique').

4.4 Redimentionnement des images

Les images sur le dataset `GITW` sont en une résolution très grande 1920x1080. Et le GPU ne peut pas supporter les images de cette taille. Ainsi le besoin de redimensionner ces images. Ce module permet de réduire la résolution de toutes les images contenues dans un répertoire source et de les sauvegarder dans un répertoire de destination avec une taille spécifiée. L'utilisateur peut spécifier la largeur et la hauteur souhaitées pour les images redimensionnées. Le script lit chaque fichier image du répertoire source, redimensionne l'image à la taille souhaitée et la sauvegarde dans le répertoire de destination. Si le répertoire de destination n'existe pas, il est créé automatiquement. Le processus est lancé via la ligne de commande avec les chemins des répertoires d'entrée et de sortie ainsi que les dimensions cibles de redimensionnement.

5 Comparaison des résultats

5.1 Avec le jeu de données MPI-Sintel

Les deux figures 3 et 2 montrent les résultats de comparaison des deux méthodes VideoFlow et PCAFlow sur MPI-Sintel. ON remarque que la méthode avec Deep Learning est plus performante sur les trois ensembles de frames. Pourtant, les deux méthodes font leurs plus mauvaises prédictions sur l'ensemble `temple_3`. Cela peut être du au fait que cet ensemble contient beaucoup de transitions majeurs en termes de temps entre ces frames.

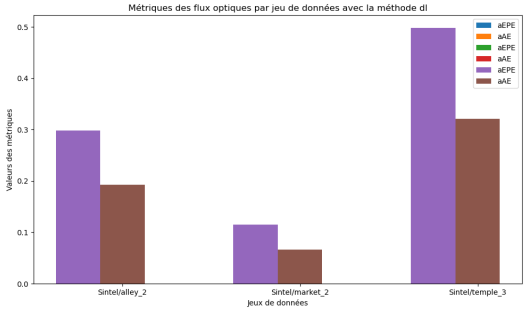

FIGURE 2 – Avec VideoFlow

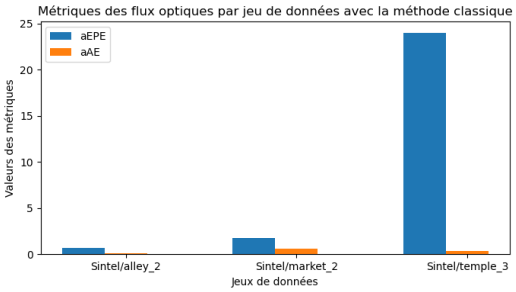

FIGURE 3 – Avec PCAFlow

5.2 Avec le jeu de données GITW

Les deux figures 5 et 2 montrent les résultats de comparaison des deux méthodes VideoFlow et PCAFlow sur GITW. On remarque que les dux méthodes performent assez bien avec VideoFlow ayant toujours la meilleure performance.

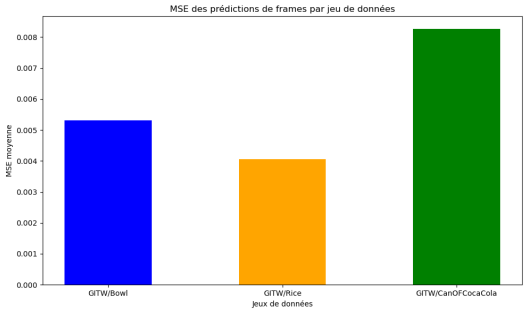

FIGURE 4 – Avec VideoFlow

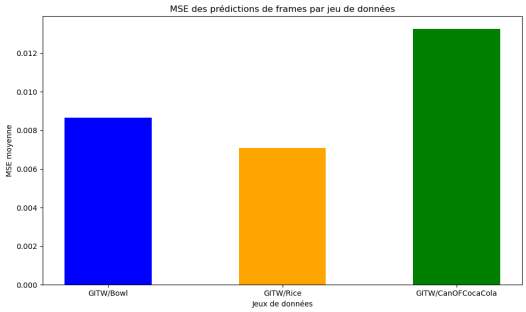

FIGURE 5 – Avec PCAFlow

6 Interface graphique

Dans le cadre de ce projet, une interface graphique interactive a été conçus en utilisant Matplotlib et Pillow pour naviguer au sein de deux séries d'images : les images directes et les images réelles. La classe FrameNavigator est au cœur de cette interface et initialise une figure comportant trois sous-figures, bien que seules deux soient activement utilisées pour l'affichage des images. Chaque axe est configuré pour ne montrer que l'image, sans repères ni axes superflus, avec un titre indiquant la progression dans la séquence. J'ai opté pour le backend WebAgg de Matplotlib, ce qui permet de rendre l'interface accessible via un navigateur web et ainsi d'améliorer l'interactivité et la flexibilité de l'utilisation.

Pour rendre l'expérience utilisateur fluide, j'ai intégré la gestion des événements de clic sur les images. Grâce à la méthode *mpl_ connect*, chaque clic de souris sur les zones dédiées à une séquence d'images déclenche la fonction *on_ click*, qui détermine si l'utilisateur souhaite avancer ou reculer dans la séquence. En détectant un clic gauche ou droit, le programme met à jour l'indice correspondant à l'image à afficher. La méthode *update_frame* se charge ensuite de rafraîchir l'image affichée et de mettre à jour son titre pour refléter la nouvelle position dans la série. Cette approche interactive et réactive permet une exploration intuitive des images, offrant ainsi une interface utilisateur efficace et conviviale pour l'analyse des séquences d'images comme indique dans la figure suivante 6 :

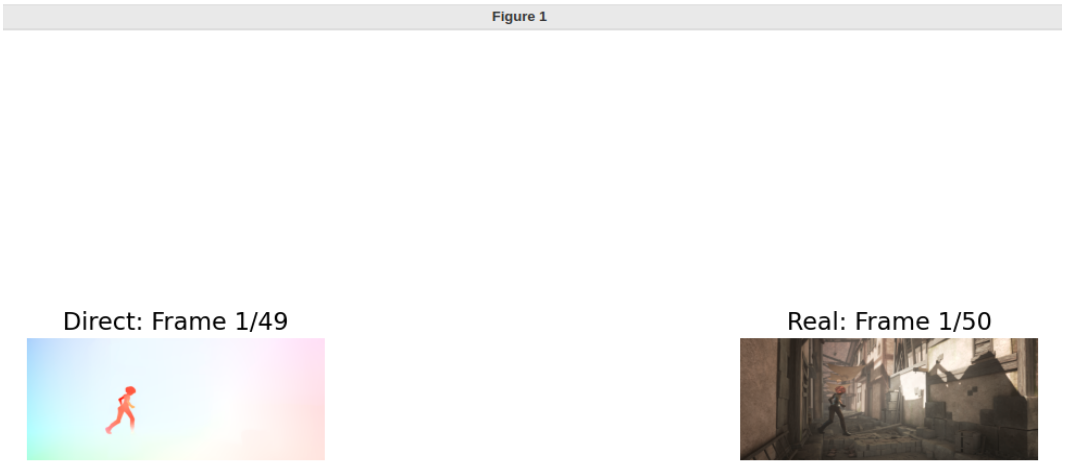

FIGURE 6 – Exemple d'affichage d'interface graphique

7 Conclusion

Dans ce projet, nous avons exploré deux approches principales pour l'analyse vidéo : une méthode classique reposant sur l'analyse en composantes principales (PCAFlow) et une méthode moderne basée sur le deep learning, représentée par VideoFlow. Ces deux approches ont été comparées à travers différentes expérimentations sur des jeux de données variés tels que MPI-Sintel et GITW.

Les résultats obtenus montrent clairement que les méthodes basées sur le deep learning, bien que plus exigeantes en termes de ressources computationnelles, offrent des performances significativement meilleures, particulièrement pour la prédiction de frames et la modélisation des flux optiques complexes. Cependant, les approches classiques conservent un intérêt notable, notamment pour leur rapidité et leur simplicité d'implémentation, ce qui les rend adaptées à des scénarios où les ressources sont limitées.