

Dorian

24/11/2024

GROUTEAU

21904277

M2 VMI

COMPTE RENDU

TP 2 Séquence Vidéo :

Optical Flow

<https://github.com/dorianGT/Optical-Flow-Analysis-Lucas-Kanade-and-Farneback.git>



SOMMAIRE

Introduction	3
Vidéos utilisées	3
Méthode Lucas-Kanade	3
Explication de la méthode	3
Implémentation	3
Étapes de l'implémentation	4
Chargement de la vidéo	4
Détection initiale des points d'intérêt	4
Boucle principale	4
Gestion des masques récents	4
Rafraîchissement des points	4
Sauvegarde et libération	4
Choix et Explication des Paramètres	5
cv.goodFeaturesToTrack	5
cv.calcOpticalFlowPyrLK	6
Résultats	7
Méthode Farneback	8
Explication de la méthode	8
Étapes de l'implémentation	8
Chargement de la vidéo	8
Initialisation	8
Calcul du flux optique	8
Annotation et sauvegarde	8
Choix et Explication des Paramètres de cv.calcOpticalFlowFarneback	9
Résultats	10
Utilisation de ces méthodes	12
Lucas-Kanade	12
Farneback	12
Utilisation de Lucas-Kanade pour la détection de Keyframe	13
Processus	13
Résultats	13
Conclusion	14
Sources	14

Introduction

L'optical flow est une technique fondamentale en vision par ordinateur, permettant de détecter et de suivre les mouvements d'objets dans une séquence vidéo. Elle repose sur l'analyse des variations d'intensité lumineuse entre des images successives, ce qui permet de calculer les déplacements apparents des pixels. Dans le cadre de ce projet, nous utiliserons la bibliothèque OpenCV en Python pour explorer et appliquer les fonctionnalités d'optical flow proposées par la documentation officielle d'OpenCV (voir [ici](#)).

Nous allons tester plusieurs méthodes d'optical flow, notamment l'algorithme basé sur les méthodes de Lucas-Kanade et l'algorithme Farneback. Un aspect clé de ce travail sera d'expliquer les choix des paramètres utilisés.

Le compte-rendu détaillera les tests effectués, les paramètres choisis et les résultats obtenus, en mettant l'accent sur l'impact de chaque choix sur les performances et la précision du suivi du mouvement dans les vidéos. Nous présenterons également un cas concret pour l'utilisation de ces méthodes : la détection de changement de scène.

Vidéos utilisées

Pour ce projet, nous avons utilisé trois vidéos :

- cat1.mp4
- herisson.mp4
- oiseau.mp4

Vidéos assez courtes avec du mouvement.

Méthode Lucas-Kanade

Explication de la méthode

La méthode Lucas-Kanade repose sur l'hypothèse que les mouvements locaux sont approximativement constants dans une petite région autour de chaque pixel. Elle calcule le déplacement en résolvant un système d'équations linéaires pour chaque point détecté.

Implémentation

La fonction `process_video_with_optical_flow` applique l'algorithme de flux optique de Lucas-Kanade pour suivre les points dans une vidéo, visualisant leurs trajectoires à l'aide de lignes colorées. Le résultat est sauvegardé sous forme d'une vidéo annotée. Ce traitement permet d'analyser les mouvements des objets ou des points d'intérêt dans une séquence vidéo.

Étapes de l'implémentation

Chargement de la vidéo

La vidéo est ouverte avec `cv.VideoCapture` et ses propriétés (largeur, hauteur, et fps) sont extraites. Ces propriétés servent à configurer un enregistreur vidéo (`cv.VideoWriter`) qui génère la sortie annotée.

Détection initiale des points d'intérêt

La détection de points d'intérêt dans la première image est réalisée avec `cv.goodFeaturesToTrack`. Cette méthode détecte des points (features) significatifs dans l'image à l'aide de l'algorithme de Shi-Tomasi.

Boucle principale

Le flux optique est calculé à chaque frame avec `cv.calcOpticalFlowPyrLK`, permettant de suivre les positions des points détectés. Les trajectoires sont visualisées en dessinant des lignes entre les positions des points dans les frames consécutives.

Gestion des masques récents

Un système de masques cumulés stocke les trajectoires sur une durée définie, simulant un effet de "traînée".

Rafraîchissement des points

De nouveaux points sont périodiquement ajoutés pour compenser la perte de points due à des mouvements brusques ou à une durée excessive.

Sauvegarde et libération

Les frames annotées sont enregistrées dans une vidéo. Les ressources associées à la capture et à l'enregistrement sont libérées à la fin.

Choix et Explication des Paramètres

cv.goodFeaturesToTrack

Cette méthode identifie les points d'intérêt dans une image à l'aide de l'algorithme de Shi-Tomasi. Les paramètres définissent les critères de sélection des points les plus significatifs.

Paramètres principaux :

- maxCorners :
 - Nombre maximum de coins à détecter.
 - Augmenter cette valeur permet de suivre plus de points, mais augmente la complexité.
 - Valeur choisie : 200.
 - Raison : Suivre un nombre suffisant de points pour une analyse fiable sans alourdir les calculs.
- qualityLevel :
 - Fraction du meilleur coin utilisé comme seuil pour exclure les coins de qualité inférieure.
 - Valeur choisie : 0.2
 - Raison : Plage équilibrée, valeurs basses pour plus de coins, valeurs élevées pour plus de précision.
- minDistance :
 - Distance minimale entre deux coins détectés.
 - Permet d'éviter la détection de coins trop proches.
 - Valeur choisie : 7.
 - Raison : Suffisant pour éviter les coins trop proches pour réduire le bruit et éviter les redondances.
- blockSize :
 - Taille du voisinage carré utilisé pour calculer le gradient pour chaque coin.
 - Une valeur plus élevée réduit la sensibilité au bruit, mais peut manquer de petits détails.
 - Valeur choisie : 7.
 - Raison : Compromis entre sensibilité et robustesse au bruit.

cv.calcOpticalFlowPyrLK

Cette méthode calcule le déplacement des points entre deux frames successives en utilisant le flux optique pyramidal de Lucas-Kanade.

Paramètres principaux :

- winSize :
 - Taille de la fenêtre de recherche autour de chaque point.
 - Une petite fenêtre est plus sensible aux petits mouvements, tandis qu'une grande fenêtre est utile pour des mouvements plus larges.
 - Valeur choisie : (21, 21).
 - Raison : Taille équilibrée pour capter à la fois les petits et moyens mouvements, adaptée à la plupart des vidéos.
- maxLevel :
 - Nombre de niveaux dans la pyramide d'images utilisée pour le calcul.
 - Une pyramide d'images réduit la résolution à chaque niveau, ce qui aide à détecter les mouvements importants.
 - Valeur choisie : 3.
 - Raison : Permet de détecter les mouvements plus larges en analysant plusieurs résolutions d'image, augmenter est peu nécessaire pour nos vidéos.
- criteria :
 - Critères d'arrêt de l'algorithme : soit un seuil d'erreur (`cv.TERM_CRITERIA_EPS`), soit un nombre maximum d'itérations (`cv.TERM_CRITERIA_COUNT`), soit une combinaison des deux.
 - Valeur choisie : (`cv.TERM_CRITERIA_EPS | cv.TERM_CRITERIA_COUNT`, 10, 0.03) avec 10 le nombre d'itérations maximum et 0.03 le seuil de convergence pour l'erreur.
 - Raison : Arrêt après 10 itérations ou une précision suffisante (erreur < 0.03), garantissant un bon compromis entre vitesse et précision. Nous avons fait varier ces valeurs, il faut rester à nombre d'itération suffisant si nous voulons obtenir de bons résultats. Un seuil d'erreur trop élevé finit prématurément le processus.

Résultats



Exemple 1



Exemple 2



Exemple 3

Les résultats obtenus montrent une visualisation claire et exploitable des mouvements des points d'intérêt dans les trois vidéos analysées. Les trajectoires tracées permettent d'observer et de comprendre les dynamiques de déplacement de manière détaillée.

Exemple 1 : Les lignes tracées illustrent précisément la trajectoire descendante des rouleaux. La méthode met en évidence le flux global vers le bas.

Exemple 2 : Les trajectoires forment un arc de cercle, capturant fidèlement le mouvement de roulade du hérisson vers la droite. Cette précision montre que l'algorithme est efficace pour suivre des mouvements complexes et courbés.

Exemple 3 : Dans cette séquence, les traits mettent en évidence deux types de mouvements :

- Le déplacement des oiseaux vers l'avant, visible par des trajectoires linéaires.
- Le battement de leurs ailes, représenté par de petites oscillations autour de chaque point suivi.

Les trois vidéos ont été choisies pour inclure une variété de mouvements comme des mouvements lents et continus (Exemple 1), des mouvements rapides et circulaires (Exemple 2), des mouvements complexes et multi-échelles (Exemple 3).

Malgré ces différences, les algorithmes de Lucas-Kanade offrent des résultats robustes et exploitables. Les ajustements des paramètres, tels que la taille des fenêtres de recherche et le nombre d'itérations, ont permis de garantir une précision adaptée à chaque type de mouvement.

Ces résultats montrent que l'algorithme est bien adapté à l'analyse de séquences avec différents types et vitesses de mouvement, tout en conservant une bonne performance globale.

Méthode Farneback

Explication de la méthode

L'algorithme de Farneback génère un flux optique dense, ce qui signifie qu'il calcule le vecteur de mouvement pour chaque pixel de l'image. Contrairement à Lucas-Kanade, qui se concentre sur des points spécifiques, Farneback offre une vue d'ensemble complète des mouvements dans une séquence vidéo.

Étapes de l'implémentation

Chargement de la vidéo

La vidéo est ouverte avec `cv.VideoCapture`. Les propriétés de la vidéo (largeur, hauteur, fps) sont extraites pour configurer un enregistreur vidéo avec `cv.VideoWriter`.

Initialisation

Les paramètres par défaut pour Farneback sont configurés (si non fournis). Un tableau HSV (Teinte, Saturation, Valeur) est créé pour la visualisation des flux. Le premier frame est lu et converti en niveaux de gris.

Calcul du flux optique

Pour chaque paire de frames consécutives :

- Le flux optique est calculé avec `cv.calcOpticalFlowFarneback`.
- Les valeurs de flux sont converties en magnitude et angle à l'aide de `cv.cartToPolar`.
- Les informations de magnitude et d'angle sont converties en couleurs (utilisation de HSV) pour visualiser les mouvements.

Annotation et sauvegarde

Les frames annotées sont enregistrées dans une vidéo. Les ressources associées à la capture et à l'enregistrement sont libérées à la fin.

Choix et Explication des Paramètres de cv.calcOpticalFlowFarneback

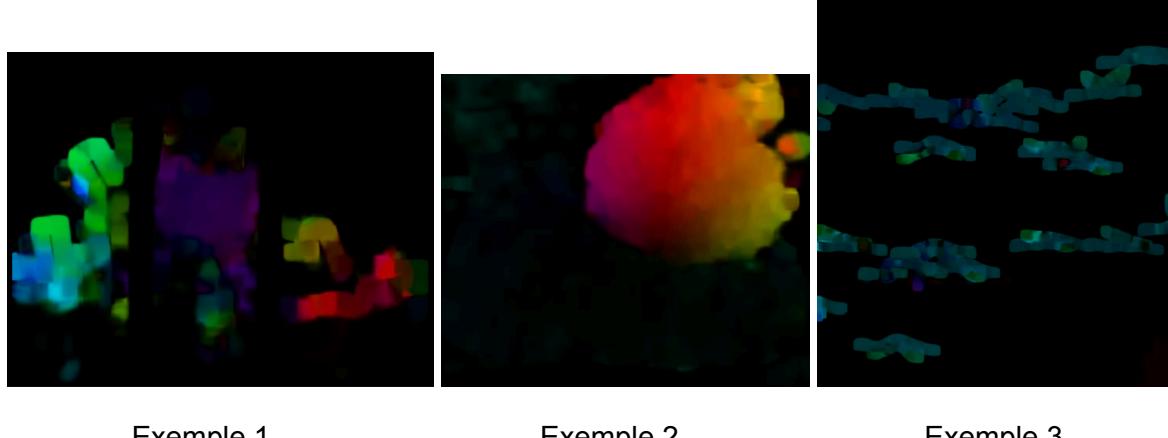
La fonction calcule un flux optique dense en utilisant l'algorithme de Farneback. Les paramètres influencent la précision et la performance de l'analyse des mouvements.

Paramètres principaux :

- pyr_scale :
 - Facteur d'échelle entre les niveaux successifs de la pyramide.
 - Une valeur proche de 0 donne plus de détails fins, tandis qu'une valeur proche de 1 réduit la résolution.
 - Valeur choisie : 0.5.
 - Raison : Réduire de moitié la résolution entre les niveaux, équilibrant vitesse et détails fins.
- levels :
 - Nombre de niveaux dans la pyramide d'images.
 - Plus il y a de niveaux, plus le calcul prend en compte des mouvements importants.
 - Valeur choisie : 3.
 - Raison : Suffisant pour détecter des mouvements moyens sans trop augmenter le temps de calcul.
- winsize :
 - Taille de la fenêtre de recherche.
 - Une petite fenêtre est sensible aux détails locaux, tandis qu'une grande fenêtre détecte mieux les mouvements globaux.
 - Valeur choisie : 15.
 - Raison : Taille modérée pour capter des mouvements locaux tout en étant robuste face à de légers flous.
- iterations :
 - Nombre d'itérations de raffinement par niveau.
 - Plus d'itérations améliorent la précision, mais augmentent le temps de calcul.
 - Valeur choisie : 3.
 - Raison : Nombre raisonnable pour un bon raffinement des vecteurs sans ralentir excessivement.
- poly_n :
 - Taille du voisinage pixel utilisé pour approximer les dérivées.
 - Une valeur plus petite (ex. 5) permet de détecter des mouvements rapides ou détaillés, tandis qu'une valeur plus grande (ex. 7) offre plus de robustesse au bruit.
 - Valeur choisie : 5.
 - Raison : Détection efficace des mouvements fins tout en restant sensible aux détails.
- poly_sigma :
 - Écart-type pour le filtre gaussien appliqué sur le voisinage.

- Valeur choisie : 1.2.
- Une valeur faible augmente la sensibilité, tandis qu'une valeur plus grande stabilise le flux contre le bruit.
- Raison : Compromis entre sensibilité et robustesse au bruit.
- flags :
 - Indique des options supplémentaires, telles que l'utilisation d'une pyramide d'images initialisée à zéro.
 - Valeur choisie : 0.
 - Raison : Paramètre par défaut, suffisant pour des analyses générales.

Résultats



Les résultats obtenus avec l'algorithme de Farneback offrent une visualisation dense du flux optique, révélant les mouvements globaux et les variations locales dans les séquences vidéo. Contrairement à Lucas-Kanade, qui suit des points d'intérêt spécifiques, Farneback permet d'analyser chaque pixel, offrant une compréhension complète des déplacements.

Exemple 1 : Le chat, représenté en teintes violet-bleu, avance vers la caméra. Les rouleaux, de couleur vert clair, chutent de chaque côté du chat. Cette visualisation met en évidence les directions opposées des mouvements.

Exemple 2 : La méthode capture avec précision le mouvement circulaire du hérisson, représenté par un dégradé de rouge-orange-jaune. L'arrière-plan, quasi immobile, reste dans des teintes uniformes, soulignant la stabilité des zones fixes. Cela montre l'efficacité de Farneback pour isoler et suivre des mouvements prononcés dans un environnement statique.

Exemple 3 : Les flux optiques révèlent les oiseaux en vol, avec des dégradés dynamiques qui capturent à la fois leur déplacement vers l'avant et le battement de leurs ailes. Le ciel en arrière-plan, stable et homogène, est représenté par des couleurs plus neutres, indiquant peu de mouvement.

Comme pour Lucas-Kanade, la méthode Farneback a été testée sur des vidéos présentant différents types de mouvements.

Les résultats montrent que Farneback est particulièrement adapté pour capturer les flux optiques globaux et denses et pour représenter efficacement les variations de direction et de vitesse des mouvements.

Malgré une consommation de ressources (beaucoup) plus élevée comparée à Lucas-Kanade, l'algorithme de Farneback offre une meilleure vue d'ensemble des scènes en mouvement, le rendant idéal pour des applications où une visualisation complète des flux est essentielle.

Utilisation de ces méthodes

Les méthodes Lucas-Kanade et Farneback sont toutes deux utilisées pour analyser les mouvements dans des séquences vidéo, mais elles ont des applications et des cas d'utilisation spécifiques en fonction de leurs caractéristiques.

Lucas-Kanade

La méthode Lucas-Kanade est idéale pour le suivi de points d'intérêt dans des vidéos où le mouvement est localisé ou implique des objets spécifiques. Elle est particulièrement adaptée dans les cas suivants :

- Suivi d'objets ou de caractéristiques distinctes : Permet de suivre des points précis comme des coins, des bords ou des objets détectés dans l'image.
- Analyse de mouvements locaux : Utile dans des scénarios où seuls des objets spécifiques ou certaines zones de l'image sont importants.
- Applications en réalité augmentée (AR) : Suivi des points caractéristiques pour aligner des objets virtuels avec des objets réels (peut être une combinaison/extension avec le TP précédent).

Farneback

La méthode de Farneback, produisant un flux optique dense, est plus adaptée pour les applications où une vue d'ensemble des mouvements dans la vidéo est essentielle. Elle est utile dans les cas suivants :

- Analyse de mouvements globaux : Idéal pour estimer des flux de mouvements de fluides, des foules, ou des scènes entières.
- Prétraitement pour segmentation : L'information dense peut être utilisée pour guider d'autres tâches, comme la segmentation basée sur le mouvement.

Utilisation de Lucas-Kanade pour la détection de Keyframe

Je travaille actuellement en entreprise sur la détection de changement de scène dans des vidéos, j'ai eu l'occasion d'utiliser Lucas-Kanade pour cela, j'ai trouvé intéressant de proposer une partie sur cela.

La méthode Lucas-Kanade est particulièrement efficace pour identifier des *keyframes* dans une vidéo, c'est-à-dire des frames où des changements significatifs se produisent. Les keyframes sont essentiels dans des applications telles que la compression vidéo, la segmentation, ou encore l'édition vidéo.

Processus

- Application de l'algorithme Lucas-Kanade précédemment définis.
- Calcul du déplacement moyen : Pour chaque frame, on calcule le déplacement moyen ou la variance des positions des points suivis.
- Détection des changements significatifs : Une frame est marquée comme keyframe lorsque le déplacement moyen ou la variance dépasse un seuil prédéfini, indiquant un changement notable dans la scène.

Résultats

Voici un exemple de résultats obtenus, nous avons fixé un seuil arbitraire et nous avons utilisé la vidéo herisson.mp4 :



Nous observons alors 7 keyframes reflétant les moments importants de mouvement de roulade de l'hérisson.

La méthode Lucas-Kanade, au-delà de la détection et du suivi des mouvements, s'avère être une approche efficace pour l'identification des keyframes dans des vidéos dynamiques. Avec des ajustements précis des seuils et une détection optimale des points d'intérêt, cette méthode peut être adaptée à une large gamme d'applications professionnelles.

Conclusion

Les algorithmes de flux optique Lucas-Kanade et Farneback ont prouvé leur efficacité pour traiter des vidéos avec des mouvements variés, chacun étant mieux adapté à des besoins spécifiques. Lucas-Kanade brille dans l'analyse localisée et précise, tandis que Farneback s'impose pour des visualisations d'ensemble. Leurs applications concrètes, comme la détection de changements de scène ou l'analyse de mouvement global, montrent leur pertinence dans des domaines variés.

Sources

- <https://helios2.mi.parisdescartes.fr/~lomn/Cours/CV/SeqVideo/Videos/>
- <https://helios2.mi.parisdescartes.fr/~lomn/Cours/CV/SeqVideo/>
- https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html
- <https://medium.com/thedeephub/object-tracking-and-path-mapping-using-lucas-kanade-optic-al-flow-in-opencv-2ea018e391d4>
- <https://medium.com/@guptachinmay321/an-insight-into-optical-flow-and-its-application-using-lucas-kanade-algorithm-c12f9f6e3773>
- https://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/tutorials/Lucas-Kanade2.pdf