Vision, réalité virtuelle et augmentée

Compte-rendu de TP

Poursuite de cible

Louis Jean

Master 2 IMAGINE
Université de Montpellier
N° étudiant : 21914083

https://github.com/louis-jean0/HAI935I-arvr

5 novembre 2024

Table des matières

1 Introduction

Dans ce projet, j'ai exploré des techniques de traitement d'images pour analyser des séquences d'images. L'objectif principal était d'implémenter et de tester différentes méthodes pour détecter et suivre des motifs dans des images séquentielles.

2 Outils utilisés

Pour mener à bien ce travail, j'ai utilisé `CImg`, une bibliothèque header-only en `C++` assez légère qui s'est avérée très pratique, ainsi que la bibliothèque standard de `C++`. Les séquences d'images utilisées, au nombre de six, ont été fournies dans le cadre de ce TP par notre enseignant M. Strauss. Le code source qui sert de support à ce compte-rendu est disponible sur le github (lien sur la page de garde).

3 Fonctionnement de l'application

Pour lancer l'application, il suffit d'exécuter un des programmes (chacun correspond à une méthode détaillée plus bas) avec en argument le nom du dossier contenant les images de la séquence.

Dans un premier temps, j'ai dû trouver un moyen de permettre la création d'un rectangle dans l'image, pour pouvoir sélectionner un motif. Pour ce faire, j'ai lu la documentation de CImg, et j'ai réussi à faire en sorte de récupérer les deux premiers clics de l'utilisateur sur la fenêtre pour créer un rectangle à partir de cette paire de points.

Figure 1: Exemple de sélection d'un motif

Une fois le motif sélectionné, la poursuite de cible commence. On aperçoit en direct les images s'enchaîner et le calcul de la nouvelle position du motif. Les images, contenant la position de la cible, sont ensuites stockées dans un dossier propre à la méthode considérée. Un fichier PDF est aussi généré, contenant la suite de toutes les images marquées.

4 Méthodes implémentées

Afin de pouvoir calculer des critères entre les zones de l'image, j'avais besoin de savoir comment découper les dites images. J'ai alors découpé l'image en zones de la même taille que le motif. La question était maintenant de savoir comment parcourir ces zones, avant même de pouvoir calculer une métrique pour évaluer la détection.

4.1 Recherche exhaustive de sous-images

Dans un premier temps, assez naïvement - *et sans lire en profondeur le sujet, je l'avoue* -, j'ai procédé par recherche exhaustive, c'est-à-dire que j'ai comparé toutes les sous-images possibles avec l'image cible.

- **Principe** : analyse de chaque région de l'image en recherchant la sous-image de référence
- **Avantages** : simplicité d'implémentation et garantit une couverture complète
- **Inconvénients** : coût de calcul élevé, trop élevé

4.2 Recherche optimisée dans les zones voisines

Pour remédier à ce problème, j'ai eu l'idée de rechercher le motif uniquement dans les zones voisines du dernier motif détecté, *avant de voir que c'était déjà proposé dans le sujet.* De cette manière, on réduit énormément le nombre de calculs à faire.

- **Principe** : limiter la recherche en limitant sur les régions proches du dernier motif détecté
- **Avantages** : plus rapide que la recherche exhaustive
- **Inconvénients** : risque de manquer des motifs

Prenons une image I prise à l'instant t, et sa successeuse I$'$ prise à l'instant $t+1$. Si le motif recherché a subi une translation entre I et I$'$ plus grande que la zone de recherche du motif, alors on risque très fortement de rater la détection du motif sur l'image I'. Il faut que les images soient fortement temporellement dépendantes, ou que le motif ne se déplace pas vite. Un choix éclairé doit alors être fait sur la taille de la zone de recherche. Dans mon programme, j'ai commencé par faire une recherche définie par la taille du motif. Intuitivement, j'ai pensé à rechercher une zone au dessus, une zone en dessous, une zone à gauche et une zone à droite du motif détecté sur l'image précédente. Cette méthode fonctionnait correctement, mais j'ai expérimenté un peu plus et me suis rendu compte qu'en descendant jusqu'à une zone de 10x10 pixels, j'obtenais de meilleurs résultats et un meilleur temps de calcul. J'ai donc décidé de partir sur cette valeur, de manière totalement empirique.

4.3 Critères basés sur la distance

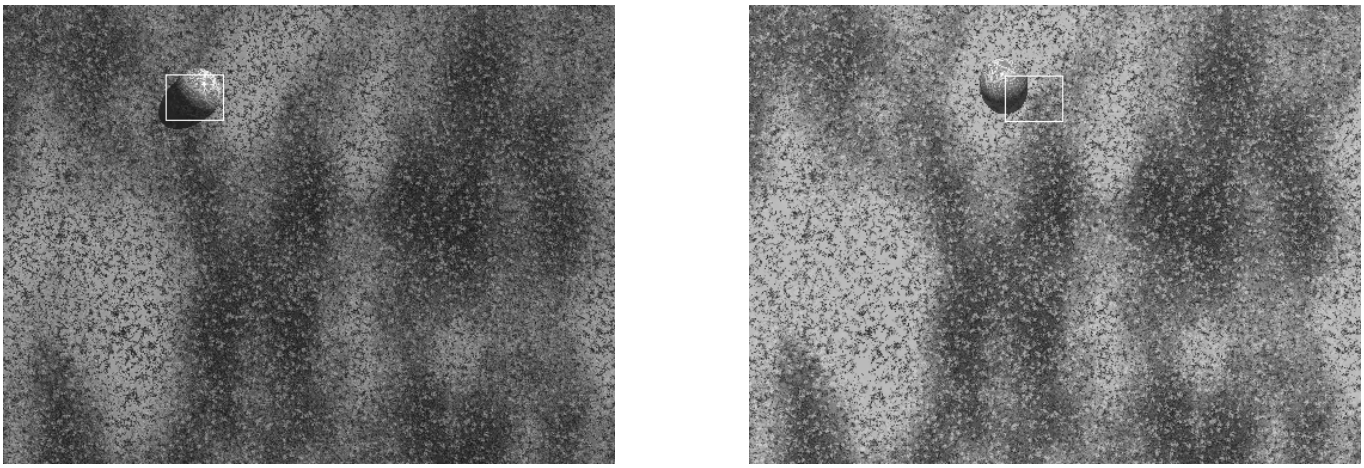
Une approche proposée pour les critères de ressemblance était de calculer des distances. C'est la première sur laquelle je me suis penché.

4.3.1 Somme des différences absolues (SAD)

La SAD calcule la somme des différences absolues pixel par pixel entre le motif et la sous-image courante.

- **Principe** : calculer SAD entre les pixels du motif et ceux de la sous-image courante, stocker la distance minimale parmi toutes les sous-images
- **Avantages** : rapide, simple à comprendre et à implémenter
- **Inconvénients** : très sensible aux variations de luminosité

La plupart du temps pour SAD, le tracking était correct. Cependant, j'ai remarqué des petits décalages de la zone détectée lors de changement d'intensité lumineuse, qui mettent à mal le suivi.



(a) 1ère itération (b) 188ème itération

Figure 2: Résultats pour SAD

Aussi parfois selon la taille de la zone, j'avais un décrochage où je perdais carrément tout le motif, et ma détection était complètement fausse.

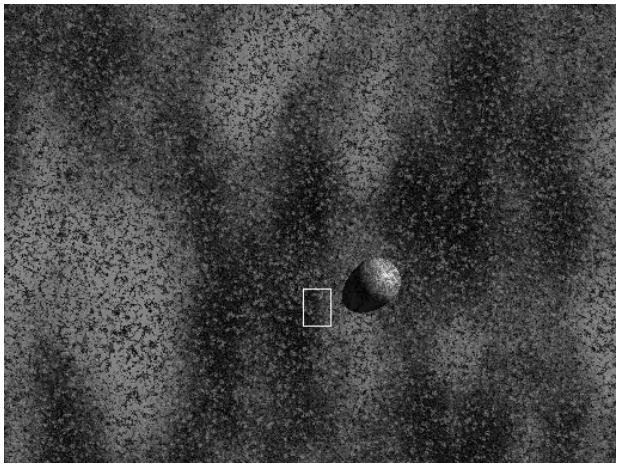


Figure 3: Décrochage pour SAD à la 100ème itération

4.3.2 Somme des carrés des différences (SSD)

La SSD utilise les carrés des différences au lieu des valeurs absolues, ce qui accorde plus d'importance aux grandes différences. Je n'ai pas remarqué de différence significative entre les deux méthodes, qui présentaient les mêmes avantages / défauts dans mon implémentation.

4.3.3 ZSAD et ZSSD

Les versions normalisées ZSAD et ZSSD consistent à soustraire la moyenne des pixels avant de calculer les différences. Cela permet de compenser les changements de luminosité dans l'image. Ces méthodes m'ont donné des résultats plus satisfaisants que SAD et SSD classiques. Avec ces méthodes effectivement j'ai réussi à supprimer l'effet négatif de la variation de luminosité, par contre j'ai toujours le problème du décrochage lorsque je prends une zone trop petite.

4.4 Critères basés sur la corrélation

La corrélation permet de comparer directement le motif avec les zones de l'image en mesurant leur similarité.

4.4.1 Corrélation

Cette méthode de corrélation classique calcule la ressemblance entre le motif et la sous-image en utilisant la somme des produits de leurs valeurs.

- **Principe** : calculer la corrélation entre les pixels du motif et ceux de la sous-image courante, stocker la corrélation maximale
- **Avantages** : permet **en temps normal** une comparaison plus robuste
- **Inconvénients** : plus complexe à calculer que ZSAD et ZSSD

Je précise **en temps normal** car dans mon cas, cette méthode n'a pas du tout fonctionné. En effet, la détection décrochait systématiquement pour terminer sa "course" dans le coin supérieur gauche.

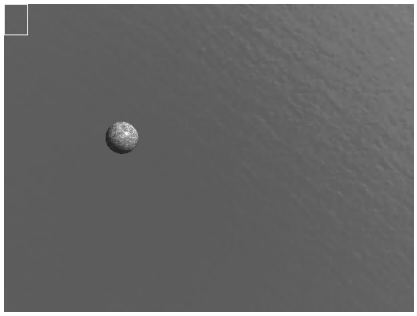


Figure 4: Décrochage pour la corrélation classique à la 19ème itération

4.4.2 Corrélation de Pearson

La corrélation de Pearson est une version améliorée de la corrélation classique, en le sens qu'elle est centrée et normalisée, en faisant intervenir les moyennes et les écarts-types des échantillons. C'est avec cette méthode que j'ai obtenu les résultats les plus concluants. Même sur des séquences où le motif lui-même change (par exemple lorsque le motif suit la tête d'un personnage qui effectue une rotation), la détection reste stable. Je n'ai plus non plus le problème de décrochage lorsque je choisis une zone trop petite.

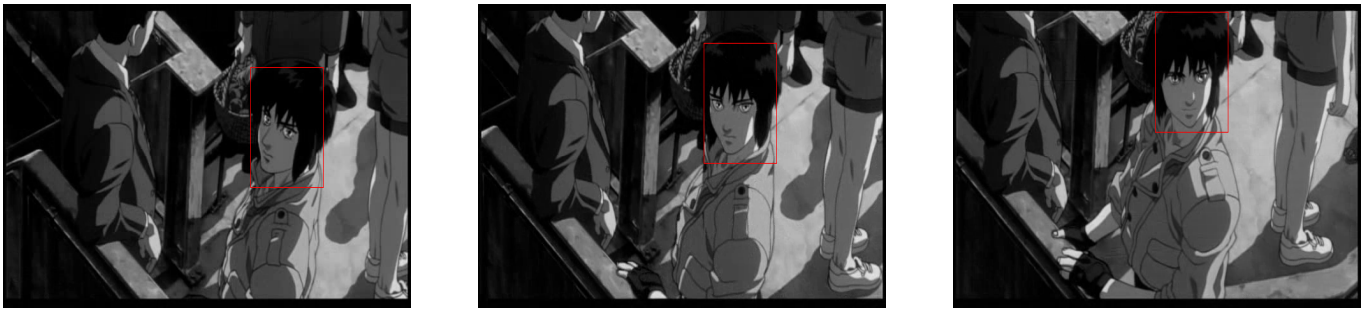


Figure 5: Début - milieu - fin de la séquence *Ghost4* avec la méthode de Pearson

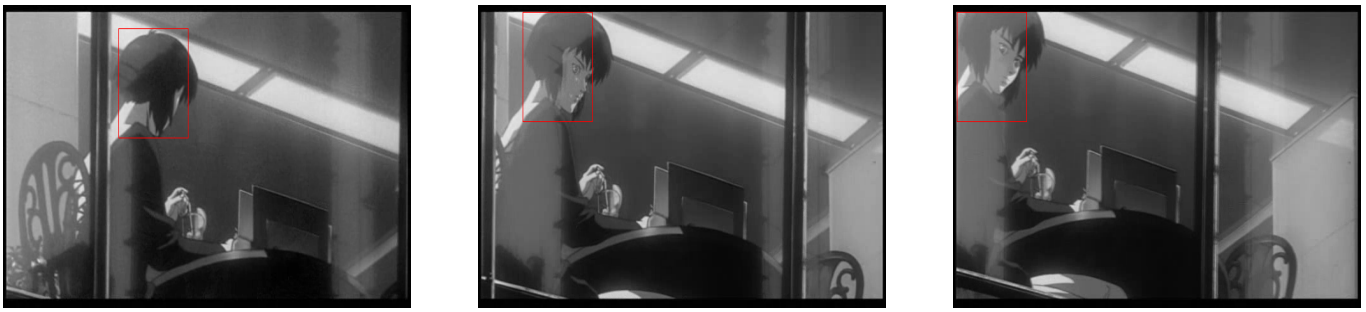


Figure 6: Début - milieu - fin de la séquence *Ghost3* avec la méthode de Pearson

En plus, comme vu en cours, on peut se permettre de seuiller la corrélation de Pearson, en faisant l'hypothèse qu'une valeur de corrélation au-dessus de 0,7 est considérée comme très bonne, évitant alors des calculs inutiles.

5 Conclusion

Au terme de ce projet, j'ai exploré diverses méthodes de poursuite de cible dans des séquences d'images en utilisant des critères de distance et de corrélation. J'ai vraiment pris du plaisir à réaliser ce petit projet. À vrai dire, je ne pensais vraiment pas que l'on pouvait faire de la poursuite de cible aussi convaincante avec des outils mathématiques si accessibles. C'est pourquoi j'aurai aimé prendre plus de temps pour implémenter le flot optique, qui lui a l'air bien plus costaud en terme de maths, et plus efficace pour gérer des déplacements rapides et complexes.

Merci pour le temps et l'attention que vous avez consacrés à la lecture de ce compte-rendu.