

TP 2 - Estimation de mouvement

© Renaud Péteri

Année universitaire 2025-2026

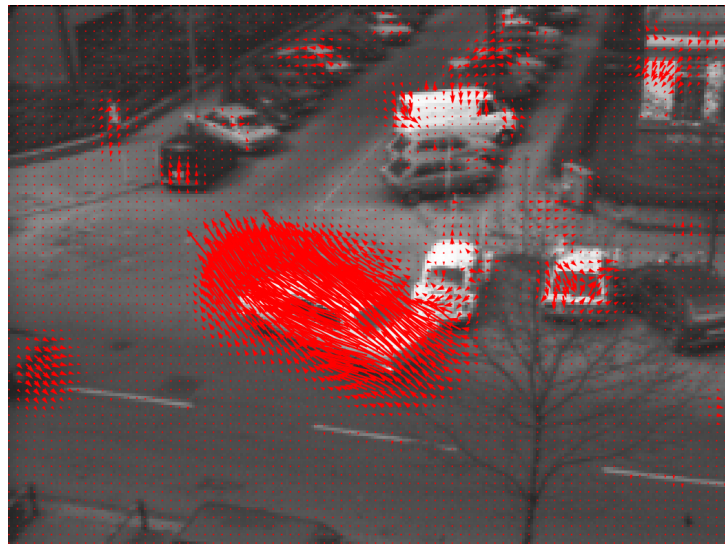


FIGURE 1 – Hamburg Taxi Sequence

1 Estimation du flot optique à partir de deux images

Preliminaires

Vous utiliserez de préférence votre ordinateur si vous avez bien OpenCV installé, sinon sur les VM Linux ou Windows (`pip install opencv-python`).

Dans un premier temps, et pour nous faciliter le développement, soit vous utilisez une webcam (celle de votre portable par exemple) soit vous effectuez les manipulations qui suivent pour utiliser la JeVois comme webcam.

Pour utiliser la jevois comme webcam, vous allez procéder de la façon suivante :

- lancez `jevois-inventor`, puis selectionnez l'onglet `videomappings.cfg`

Avant toute modification, **vous effectuerez une sauvegarde** de ce fichier en copiant son contenu dans un fichier `videomappings.cfg` sur votre ordinateur personnel.

Quand cela est fait, remplacez, sur la jevois, le contenu de ce fichier par l'unique ligne :

```
YUYV 640 480 30.0 YUYV 640 480 30.0 JeVois PassThrough *
```

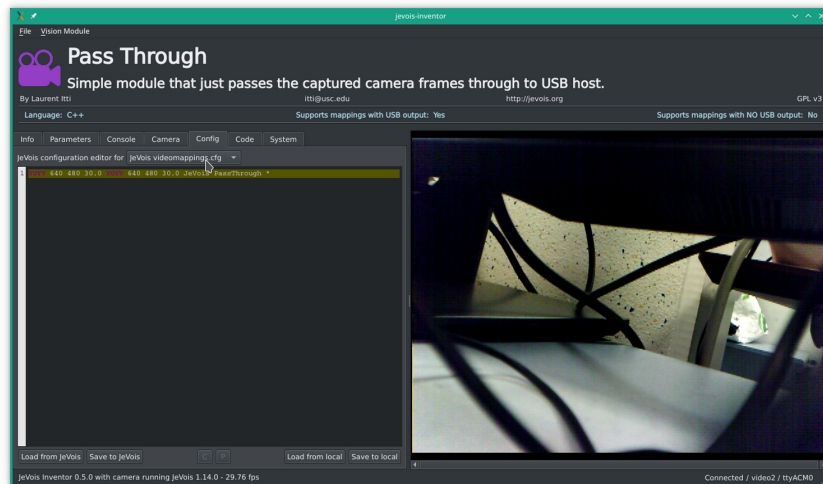


FIGURE 2 – JeVois en *PassThrough*

Sauvegardez par Ctrl+S puis redémarrez la caméra. Elle doit se comporter comme une simple caméra USB. Arrêtez jevois-inventor avant de passer à la suite.

Une première approche d'estimation de mouvement

1. Regarder le tutoriel concernant l'estimation de mouvement avec OpenCV, et plus particulièrement la section intitulée *Dense Optical Flow in OpenCV* :
https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html
2. Faites tourner l'exemple *Dense Optical Flow in OpenCV* dont le code python, que vous devez comprendre, est donné (à quoi correspondant le code couleur sur l'image du champs de déplacement?). On utilisera la séquence *Escalator.avi* disponible sous Moodle.
 Remarque : essayer aussi l'exemple avec le flux de la JeVois ou de votre webcam :
`cap = cv.VideoCapture(0)`
3. Adaptez votre code pour traiter 2 images consécutives de la séquence *taxi* (disponible aussi sous Moodle).
4. Estimez ensuite la norme du champ de mouvement obtenu et affichez-le.
 ATTENTION : pour la suite du TP, ne normalisez pas la norme du vecteur mouvement (`cv.normalize`)!
Appeler l'enseignant pour valider
5. Afin d'éviter de considérer le flot optique des pixels dont la norme du vecteur mouvement est trop "petite", on utilisera la fonction `np.where` pour ne conserver que les pixels dont la **norme du champ de mouvement** est plus grande qu'un seuil=0.75. Affichez le résultat. Essayer aussi avec un seuil=2.
6. Enfin, estimez une carte binaire des pixels en déplacement (toujours pour ceux dont la norme du champ est plus grande que le seuil=0.75)
Appeler l'enseignant pour valider



1



2

2 Avec la caméra JeVois

Nous allons désormais porter notre programme précédent sur la caméra JeVois pour créer un détecteur d'intrusions.

1. Créez un module `OpticalFlow` (voir TP1) dans lequel vous mettrez votre code.
Remarque : si vous souhaitez sauvegarder votre module, ou bien l'éditer depuis votre ordinateur, vous pouvez monter la caméra comme un périphérique USB.
2. Tester le portage du tutorial de la section 1 sur la JeVois en se référant à :
<http://jevois.org/tutorials/ProgrammerInvFlowLK.html>
3. En vous inspirant de l'exemple précédent, adapter votre programme pour le faire tourner sur la JeVois

Appeler l'enseignant pour valider



3

4. Enfin, fixer votre caméra, et écrivez un programme pour que la caméra détecte les intrusions apparaissant dans son champs de vision (c'est à dire lorsqu'il y a un mouvement important dans le flux vidéo- par exemple un seuil sur le pourcentage de pixels en mouvement significatif). On pourra écrire 'Detected !' (attention : ne pas mettre d'accents) en rouge **sur la frame de sortie** lorsqu'une intrusion est détectée.

Appeler l'enseignant pour valider



4

Pour aller plus loin : utilisation du flot normal

Dans le but d'accélérer l'estimation du champs de mouvement avant de déployer le programme sur la JeVois, on va remplacer la méthode `cv.calcOpticalFlowFarneback(...)` par le calcul de la norme du flot normal.

On rappelle que l'expression du flot normal au pixel \mathbf{p} à l'instant t est :

$$\vec{v}_N(\mathbf{p}, t) = -\frac{I_t(\mathbf{p}, t)}{\|\vec{\nabla} I(\mathbf{p}, t)\|} \vec{n} \quad (1)$$

avec $I_t(\mathbf{p}, t)$ la dérivé temporelle au pixel \mathbf{p} à l'instant t , $\|\vec{\nabla} I(\mathbf{p})\|$ la norme de son gradient spatial, et \vec{n} un vecteur unitaire normal aux contours.

Nous nous intéresserons uniquement à la norme du flot normal :

$$\|\vec{v}_N(\mathbf{p}, t)\| = \frac{|I_t(\mathbf{p}, t)|}{\|\vec{\nabla} I(\mathbf{p}, t)\|} = \frac{|I_t(\mathbf{p}, t)|}{\sqrt{I_x^2(\mathbf{p}, t) + I_y^2(\mathbf{p}, t)}}$$

Son calcul est très simple car il s'agit d'estimer une différence finie suivant x , y , et t :

$$\begin{aligned} I_t(\mathbf{p}, t) &= I(\mathbf{p}, t+1) - I(\mathbf{p}, t) \\ I_x(\mathbf{p}, t) &= I(\mathbf{p}[x+1, y], t) - I(\mathbf{p}[x, y], t) \\ I_y(\mathbf{p}, t) &= I(\mathbf{p}[x, y+1], t) - I(\mathbf{p}[x, y], t) \end{aligned}$$

1. Créez la fonction `ComputeDerivatives(im1, im2)` qui calcule et renvoie les gradients spatiaux I_x et I_y de l'image `im1` suivant x et y ainsi que le gradient temporel I_t entre `im1` et `im2`.

On convertira le plus tot possible (après `imread`) les images en double (fonction `double()`) pour pouvoir stocker les valeurs négatives du gradient.

Faites un test en affichant les gradients spatiaux et temporels pour la première

image de la séquence.

2. Créez la fonction `[NFnorm] = NormalFlowNom(im1, im2)` qui permet d'estimer la norme du champs de vecteurs normaux (u, v) entre deux images *im1* et *im2*. Ici aussi, on utilisera la fonction `where` pour éviter d'estimer le flot optique sur les pixels dont la norme est trop "petite".
3. L'estimation du flot normal est sensible au bruit : améliorer votre algorithme en intégrant un filtrage Gaussien préalable sur les images de paramètre σ (à mettre en paramètre de votre fonction). Tester en affichant la norme des vecteurs du flot normal pour la première image de la séquence.



Appeler l'enseignant pour valider