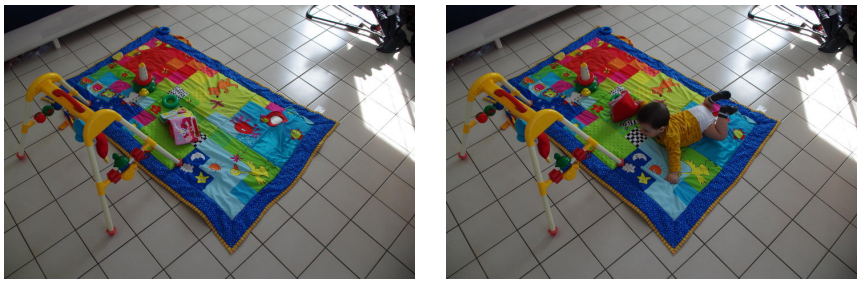
SY32 – TD Vision 02 : Flot optique

Exercice 1 : Détection par soustraction du fond

Comme tous parents suivant les dernières tendances, nous utilisons les technologies connectées 4.0 pour surveiller notre enfant de quelques mois. Nous avons placé une caméra dans le salon de sorte à pouvoir contrôler ses faits et gestes à distance, pendant que nous sommes au travail ou que nous rencontrons nos amis sur les réseaux sociaux.
Pour cela, nous pouvons détecter bébé et son tumulte par une méthode simple : la soustraction du fond.



(a) Fond (fond.jpg). (b) Image à traiter (bb1.jpg).

FIGURE 1 – Images à utiliser pour pratiquer la détection d'éléments par soustraction du fond.

Nous allons appliquer la détection de personnes par soustraction du fond avec les images représentées en Figure 1.

1. Ouvrir les fichiers images et convertir les images en niveaux de gris (veiller à garder des données codées en entiers entre 0 et 255, mais permettant des opérations arrivant à des valeurs négatives).

```
import numpy as np
from skimage import io
from skimage import color
from skimage import util
import matplotlib.pyplot as plt

%matplotlib auto

B = io.imread("fond.jpg")
I = io.imread("bb1.jpg")
B = util.img_as_ubyte(color.rgb2gray(B)).astype('int')
I = util.img_as_ubyte(color.rgb2gray(I)).astype('int')
```

2. Implémenter une détection en calculant une différence simple entre l'image à traiter et l'image du fond, et en y appliquant un seuil. Ceci permet de générer un masque de détection. Afficher la différence, le masque, et commenter.

```
D = np.abs(I-B).astype('uint8')

plt.figure()
plt.imshow(D)
plt.colorbar()
plt.show()
```

3. Que pourrait-on faire pour débruiter le masque de détection ?
4. À présent, appliquer le principe de soustraction du fond par blocs 16×16 : chaque bloc doit valoir la moyenne des différences calculées entre les points des images le constituant. Les blocs doivent partitionner l'image sans recouvrement. Seuiller et comparer avec les résultats précédents.
5. Afficher une segmentation de l'élément détecté dans l'image traitée. M est le masque créé à la question précédente. Un résultat possible est montré en Figure 2.

```
S = np.zeros(I.shape, 'uint8')
S[M > 0] = I[M > 0]

plt.figure()
plt.imshow(S, cmap='gray')
plt.show()
```

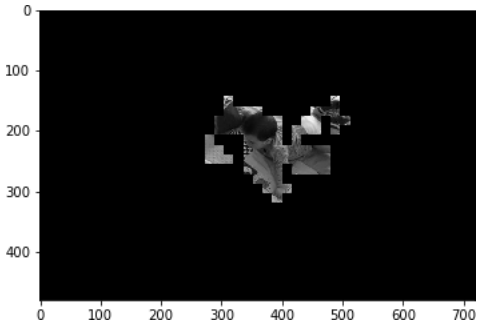


FIGURE 2 – Résultat de la segmentation par soustraction du fond par bloc.

Exercice 2 : Segmentation par flot optique

(a) I_1 à l'instant t (taxi15.png). (b) I_2 à l'instant $t + dt$ (taxi16.png).

FIGURE 3 – Images à utiliser pour estimer le champ de mouvement entre elles.

Calculer la direction et l'amplitude des mouvements permet une caractérisation plus fine des éléments dynamiques de la scène. Il faut ici estimer le champ de déplacement de chaque pixel. Nous allons calculer le flot optique entre les images représentées en Figure 3.
Les hypothèses sont : la luminance des objets est constante entre les deux images, et leur déplacement dans l'image est très faible.
La première hypothèse se traduit par l'expression :

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

La deuxième hypothèse nous permet d'appliquer le développement de Taylor du premier ordre du terme de droite sans commettre une erreur notable :

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}.dx + \frac{\partial I}{\partial y}.dy + \frac{\partial I}{\partial t}.dt$$

Le vecteur (dx, dy) représente le déplacement du pixel (x, y) de l'image 1 à l'image 2 et dt l'écart temporel entre les deux images.

1. Déduire des formules de l'énoncé l'équation de la contrainte de mouvement :

$$I(x, y, t + dt) - I(x, y, t) + \frac{\partial I}{\partial x}.dx + \frac{\partial I}{\partial y}.dy = 0$$

2. Calculer l'image des différences entre les deux trames $\Delta I = I_2 - I_1$ (travailler en type flottant pour ne pas tronquer les résultats des opérations). Elle correspond à $I(x, y, t + dt) - I(x, y, t)$.
3. Calculer le gradient selon X et selon Y de l'image I_1, en utilisant les noyaux de convolution suivants :

$$G_x = \frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix} \quad G_y = \frac{1}{2}\begin{bmatrix}1\\0\\-1\end{bmatrix}$$

   Ainsi nous avons des matrices contenant les valeurs $\frac{\partial I}{\partial x}$ et $\frac{\partial I}{\partial y}$ en chaque point.
4. Afin d'obtenir une estimation « robuste » du flot optique, il est nécessaire de faire une hypothèse supplémentaire : le vecteur déplacement est supposé constant sur une fenêtre d'observation Ω,

illustrée en Figure 4. On va considérer que Ω est de dimensions $N \times N$ pixels, N impair et > 1. Pour chaque point $(u, v) \in \Omega$, l'équation de contrainte de mouvement s'écrit :

$$\underbrace{I(u, v, t + dt) - I(u, v, t)}_{\Delta I(u,v)} + \frac{\partial I}{\partial u}.dx + \frac{\partial I}{\partial v}.dy = 0$$

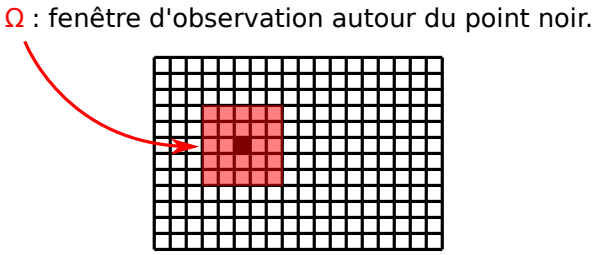


FIGURE 4 – Le flot optique est supposé constant dans le voisinage au point considéré (en noir). Ce voisinage est défini par la fenêtre Ω centrée sur ce point. Chaque point de Ω contribue de manière égale et donne une équation pour résoudre le flot optique du point étudié.

Détailler le sytème d'équation résultant de l'écriture matricielle de ces équations pour tous les points de Ω, sous la forme $A.d = B$ avec $d = \begin{pmatrix} dx \\ dy \end{pmatrix}$ (Décrire A et B).

Aide : A est de dimensions $(N^2, 2)$, et B de dimensions $(N^2, 1)$.

5. Ce système d'équations linéaires a plus d'équations (N^2 le nombre de points de Ω) que d'inconnues (2 : dx et dy), ce qui permet de le résoudre par la méthode des moindres carrés.
   Dans ces conditions, on admet que l'on peut écrire :

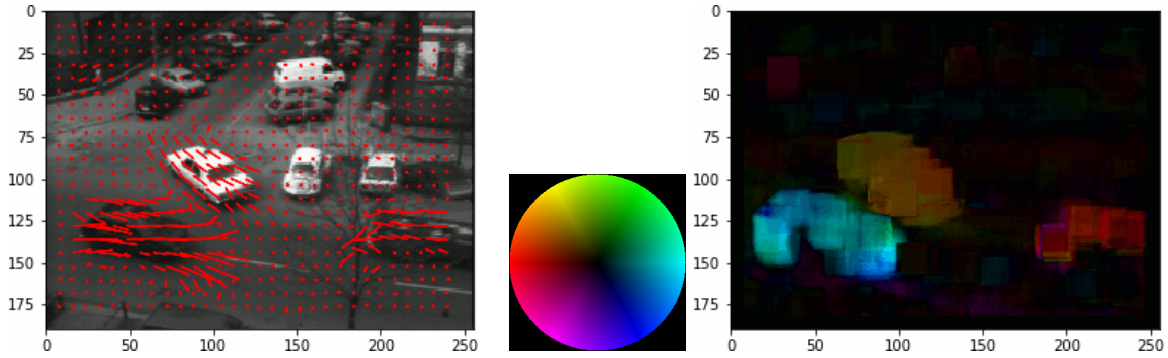
$$d = (A^T A)^{-1} A^T B$$

   ce qui permet de calculer dx et dy.
   Implémenter une fonction permettant de calculer le vecteur de déplacement entre les deux images pour un pixel, en considérant une région centrée sur ce pixel. Le voisinage bidimensionnel d'estimation du flot optique $\Omega(x, y) = \{(x + u - (N/2), y + v - (N/2)) \setminus (u, v) \in [[0; N - 1]]^2\}$ sera paramétré par la variable N, impaire (dans cette formulation, l'opérateur / représente la division entière).
   Choisir les points à traiter à l'aide de la fonction `matplotlib.pyplot.ginput` [1]. Pour l'estimation du flot optique des points sélectionnés, essayer différentes valeurs de $N \in \{5, 11, 21, 31\}$. Visualiser les vecteurs de déplacements associés à ces points, par dessus l'image I_1.
   Avec `numpy`, la division entière se note `//`, le produit matriciel se note `@`, l'inversion de matrice se fait avec la fonction `np.linalg.inv()`. Pour visualiser un vecteur, faire appel à la fonction `plt.arrow(x, y, dx, dy, head_width=1, length_includes_head=True, color='r')`. Lors de l'affichage des vecteurs, il peut être bon de les multiplier par 10 afin de mieux les observer.
6. Calculer le flot optique pour l'image complète. Ignorer les bords de l'image dans un rayon de $N/2$ pour éviter des erreurs de fenêtre d'observation débordant des limites de l'image.
7. Affichage du flot optique :
   (a) Afficher le vecteur de déplacement pour un point sur 8 environ afin de ne pas surcharger l'image. La Figure 5 (a) donne un résultat possible.

   bonus Représenter le flot optique de manière dense en utilisant l'espace colorimétrique TSV (Teinte-Saturation-Valeur = *HSV Hue-Saturation-Value*) : la roue des couleurs de T pour l'angle du déplacement, et V pour l'amplitude. Ces trois composantes doivent être codées en flottants $\in [0; 1]$; il faut donc veiller à mettre à l'échelle l'angle représentant T. S devra prendre une valeur constante, choisir 1 permet des couleurs éclatantes. La Figure 5 (b) donne un résultat possible.
   Suggestion : adoptez un paramètre *amax* pour borner l'amplitude maximale représentée avec V (V=0 pour un vecteur nul, V=1 pour un vecteur d'amplitude $\geq$ *amax*).
8. Trouver un seuil sur l'amplitude des vecteurs du flot optique donnant le masque des objets en mouvement, et afficher le masque ainsi que la segmentation.

bonus Générer l'image $\hat{I}_2$, une estimation de I_2 à partir de I_1 et du flot optique calculé entre I_1 et I_2.

1. En cas de difficultés, une alternative peut être d'écrire des coordonnées de points manuellement dans le code, par exemple `pt = [101, 113]`.



(a) Représentation éparse en vecteurs. (b) Échelle et représentation dense en couleurs.

FIGURE 5 – Flot optique calculé entre les images I_1 et I_2.

Exercice bonus : Calcul du flot optique par correspondance de blocs *block matching*

Reprendre l'exercie 2 en changeant de méthode pour calculer le flot optique :

1. Au lieu de chercher à optimiser le système d'équations vu dans l'exercice précédent, calculer le flot optique par mise en correspondance de blocs. Cette méthode ne considère que les hypothèses de constance de luminosité des objets dans le temps, et de flot localement constant.
   Il faut gérer une fenêtre d'observation autour des points pour tenir compte de leur voisinage (qui constitue un bloc à traiter), ainsi qu'une fenêtre de recherche (à régler en fonction de l'amplitude des mouvements attendus) pour définir la zone de recherche du point correspondant dans l'autre image (il est inutile et couteux de fouiller dans toute l'image).
   La comparaison des points est basée sur les intensités dans leurs fenêtres d'observation, avec une mesure telle que la SSD (*Sum of Squared Difference*, l'erreur quadratique) :

$$SSD(bloc1, bloc2) = \sum_{(u,v)\in N\times N} (bloc2(u,v) - bloc1(u,v))^2$$

   où $bloc1$ est le bloc autour d'un point dans l'image $I1$, et $bloc2$ est le bloc d'un point correspondant candidat dans $I2$. Un point s'étant déplacé entre $I1$ et $I2$ devrait donner une SSD minimale. Lorsqu'il est trouvé, le décalage de position donne le déplacement d.
2. Comparer l'estimation de flot optique entre les deux méthodes.

Liste de fonctions utiles :

- `skimage.io.imread`
- `numpy.zeros` et `numpy.zeros_like`
- `range`
- `scipy.ndimage.filters.convolve`
- `skimage.filters.median` (ou `scipy.ndimage.median_filter`)
- `numpy.array`
- `round`
- `numpy.hstack`
- `numpy.reshape`
- `numpy.sqrt`
- `numpy.sum`
- `numpy.linalg.norm`
- `numpy.copy`
- `numpy.power` (ou opérateur `**`)
- `skimage.color.hsv2rgb`
- `numpy.pad`