



DOSSIER PEDAGOGIQUE

| | |
|--|----------|
| 1 OBJECTIF DU TP | 1 |
| 1.1 DRONE DE PRISE DE VUE AERIEENNE EQUIPE D'UNE NACELLE | 1 |
| 1.2 ENVIRONNEMENT RECREE | 1 |
| 1.3 OBJECTIF | 2 |
| 1.3.1 Seuillage | 2 |
| 1.3.2 Détection du centre | 2 |
| 2 PRISE EN MAIN DES IMAGES A TESTER ET DE QUELQUES FONCTIONS UTILES | 2 |
| 2.1 IMPORT DES BIBLIOTHEQUES | 2 |
| 2.2 LECTURE DES IMAGES TESTS | 2 |
| 2.3 ANALYSE DE LA STRUCTURE DE L'IMAGE SEUILLEE | 3 |
| 2.4 AJOUT D'UN POINT AU CENTRE L'IMAGE | 3 |
| 2.5 MESURE D'UN TEMPS | 3 |
| 3 PREMIER ALGORITHME : RECHERCHE DU BARYCENTRE DE L'IMAGE | 4 |
| 3.1 ALGORITHME CHERCHE_CENTRE1 | 4 |
| 3.2 ALGORITHME CHERCHE_CENTRE2 | 4 |
| 4 SECOND ALGORITHME : RECHERCHE DES LIMITES DE LA FORME | 5 |
| 5 ANALYSE D'UN PROGRAMME AVEC DES FONCTIONS NUMPY | 6 |
| 6 COMPLEMENT D'INFORMATION : CODAGE RGB | 6 |

Compétences Visées :

- ☐ concevoir un algorithme répondant à un problème précisément posé....
- ☐ concevoir l'en-tête (ou la spécification) d'une fonction, puis la fonction elle-même,
- ☐ traduire un algorithme dans un langage de programmation
- ☐ étudier l'effet d'une variation des paramètres sur le temps de calcul, sur la précision des résultats, sur la forme des solutions pour des programmes d'ingénierie numérique choisis
- ☐ documenter une fonction, un programme plus complexe



DOSSIER PÉDAGOGIQUE

**Traitement d'image : algorithmes
de reconnaissance du centre d'un
objet cible**

Sujet

CPGE

1 OBJECTIF DU TP

1.1 Drone de prise de vue aérienne équipé d'une nacelle

L'utilisation de drones pour filmer une scène (événements sportifs, publicité, actions militaires...) est de plus en plus répandue car elle permet d'avoir des angles de vues plus intéressants et au plus proche de l'action, en s'affranchissant d'éventuels obstacles (foule, obstacles naturels,...).

La fonctionnalité « **suivi de cible** » permet de placer systématiquement et automatiquement le sujet à filmer (par exemple le véhicule en tête d'une course automobile) au centre de l'image.

Le drone, mobile, est en général piloté à l'aide d'une télécommande par une personne au sol. La position et l'orientation du drone est variable, d'autant plus qu'il est soumis à des actions extérieures aléatoires (vent, trous d'air ...).

Un système appelé nacelle, permet de déplacer la caméra en l'orientant autour de 2 ou 3 axes.

La nacelle est actionnée par des moteurs, dont les positions sont commandées à partir des données de vols issues d'une centrale inertielle qui mesure la position du drone par rapport à l'horizontale ainsi que son accélération, sa vitesse. Ceci permet par exemple d'imposer à la caméra de garder une position horizontale quelle que soit la position du drone.

Dans le cas du suivi de cible, le retour d'image fourni par la caméra permet de connaître les mouvements de la cible dans l'image et d'en déduire les déplacements à imposer à la nacelle afin de recentrer automatiquement la cible sur l'image.



1.2 Environnement recréé

Le support d'étude dans cette activité est la nacelle de Drone associée à sa caméra dans un environnement recréé.

Dans la configuration du TP, la nacelle ne peut corriger que la position verticale de la cible dans l'image. La position horizontale de la cible dans l'image ne pourra être corrigée qu'en pilotant le drone lui-même. Le cadre de l'étude est limité au suivi de cible réalisé par la nacelle : le drone sera donc considéré fixe et on s'intéressera uniquement au suivi sur un axe vertical Y.

Pour effectuer des mesures, on dispose : de la nacelle équipée d'une caméra de prise de vue, d'un logiciel de commande et de visualisation des grandeurs physiques mesurées, et d'un écran de projection permettant de simuler le déplacement de la cible.



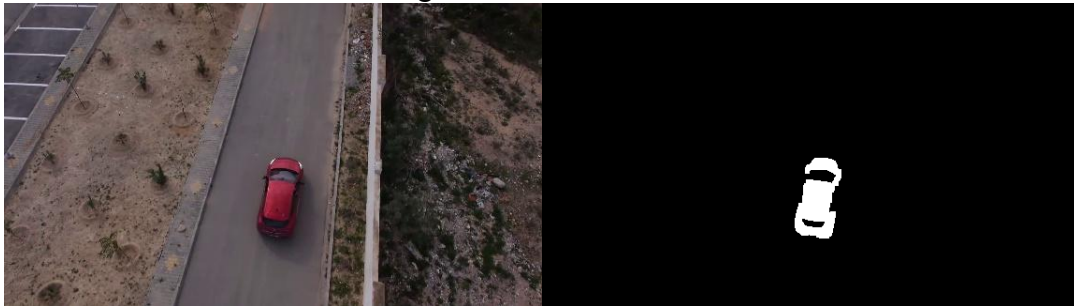
1.3 Objectif

Divers algorithmes de tracking sont utilisés pour arriver à détecter les mouvements de la cible à partir de l'image filmée (algorithme KLT[Kanade Lucas 1992], RMRm [Obodez 1995],...).

On s'intéressera à un algorithme simplifié basé sur une technique de traitement d'image. Pour l'utiliser, il faut que l'objet à reconnaître ait un caractère particulier (forme, couleur,...) de façon à le différencier du fond : dans l'exemple ci-dessous l'objet à détecter est le véhicule, caractérisé par sa couleur rouge.

1.3.1 Seuillage

Dans un premier temps l'image est seuillée : les pixels dont la couleur s'approche de la couleur rouge sont différenciés des autres comme le montre l'image ci-dessous :



1.3.2 Détection du centre

Il suffit alors de détecter le centre de l'objet à partir de l'image seuillée.

On propose de mettre en place dans ce TP différents algorithmes permettant d'obtenir le centre de l'objet à partir de l'image seuillée, et de comparer leurs performances en termes d'efficacité et de temps de calcul.

Le temps de calcul est un élément fondamental car il va conditionner la qualité du suivi ; un temps de calcul trop long pouvant même rendre le système instable.

2 PRISE EN MAIN DES IMAGES À TESTER ET DE QUELQUES FONCTIONS UTILES

2.1 Import des bibliothèques

Utiliser les lignes suivantes pour importer les bibliothèques nécessaires :

```
python
import matplotlib.pyplot as plt    #tracé de courbes
import matplotlib.image as mpimg  #visualisation des images
import time as t                  #mesure des temps
```

2.2 Lecture des images tests

On propose de travailler avec 2 images test et leur version seuillée.

Prendre connaissance de ces 4 images qui sont stockées dans les fichiers suivants : *im1.png* ; *im1_s.png* ; *im2.png* ; *im2_s.png*.

Vous pouvez utiliser les instructions suivantes pour lire une image à partir d'un fichier :

```
python
#pour pyzo, changement du répertoire de travail
import os
os.chdir(r'D:/etc...') #taper ici le chemin où sont stockées les images

#lecture des images
img1=mpimg.imread('im1.png')
imgls=mpimg.imread('im1_s.png')
# affichage des images
plt.imshow(img1)
plt.figure()
plt.imshow(imgls,cmap='gray') #affichage particulier pour une image en niveau
de gris
plt.show()
```

Vous remarquerez le sens des axes.

2.3 Analyse de la structure de l'image seuillée

On travaille dans ce TP à partir des images seuillées.
Ces images sont stockées sous la forme d'une matrice, contenant :

- 0 pour un pixel noir
- 1 pour un pixel blanc

Vérifier cette structure, et tester la méthode *shape* :

```
python
imgls[0,0]
imgls[200,300]

imgls.shape
```

Pour information, l'annexe en fin de sujet présente la façon dont est codée l'image en couleur.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Structure de la matrice image seuillée

2.4 Ajout d'un point au centre de l'image

On souhaite afficher au cours de ce TP un symbole au centre de la forme reconnue.
Tester les fonctions suivantes en affichant un symbole au centre de l'image (choisir les valeurs pour x et y).

```
python
plt.plot([x],[y], 'g^')
plt.plot([x],[y], 'ro')
plt.plot([x],[y], 'bs')
plt.show()
```

2.5 Mesure d'un temps

On rappelle la fonction **clock** de la bibliothèque **time** pour mesurer un temps d'exécution:

```
python
t1=time.clock()
.....
t2=time.clock()
t=t2-t1
```

3 PREMIER ALGORITHME : RECHERCHE DU BARYCENTRE DE L'IMAGE

3.1 Algorithme cherche_centre1

On donne ci-dessous l'expression des coordonnées x_g et y_g moyenne pondérée associée aux pixels de l'image affectés des poids p_{ij} :

$$x_g = \frac{1}{s} \sum_{i=0}^{h-1} \left(\sum_{j=0}^{l-1} p_{ij} * j \right) \quad y_g = \frac{1}{s} \sum_{i=0}^{h-1} \left(\sum_{j=0}^{l-1} p_{ij} * i \right) \quad s = \sum_{i=0}^{h-1} \left(\sum_{j=0}^{l-1} p_{ij} \right)$$

Avec :

- p_{ij} : valeur du pixel de coordonnées (i,j) : p_{ij} vaut 1 (pixel blanc) ou 0 (pixel noir)
- h : nombre de lignes
- l : nombre de colonnes

Q1. Implémenter une fonction **somme** qui prend pour argument une matrice image seuillée, et qui retourne la somme des pixels de la matrice image (quantité s définie ci-dessus).

Q2. Mettre en place la fonction **cherche_centre1** qui prend pour argument une matrice image seuillée, et qui retourne les coordonnées x_g et y_g .

Q3. Tester cette fonction sur les 2 images tests fournies. Vous placerez un carré bleu au centre de la forme détecté.

Q4. Mesurer le temps d'exécution de cette fonction.

3.2 Algorithme cherche_centre2

On cherche à améliorer le temps de calcul. L'opération $p_{ij} * i$, gourmande en temps de calcul, n'est pas forcément nécessaire, puisque la quantité p_{ij} est soit égale à 1, soit à 0.

Q5. Proposer une fonction **cherche_centre2**, sur le même principe que **cherche_centre1**, mais qui ne réalise pas de multiplication.

Q6. Tester cette fonction sur les 2 images tests fournies.

Q7. Mesurer le temps d'exécution de cette fonction. Comparer avec celui obtenu pour l'algorithme précédent.

5 ANALYSE D'UN PROGRAMME AVEC DES FONCTIONS NUMPY

- Q11. Prendre connaissance de la fonction **cherche_centre4** fournie dans le fichier `algo_a_commenter.py` présent dans le dossier *RESSOURCES*.
- Q12. Copier la fonction dans votre script et documenter la en ajoutant des lignes de commentaires pour chaque ligne de programme. Indiquer aussi quel est le principe retenu et en quoi cet algorithme diffère des précédents.
- Q13. Tester cette fonction (placer un triangle vert au centre de la forme détectée) et mesurer le temps de calcul.

6 COMPLEMENT D'INFORMATION : CODAGE RGB

Les images avec l'extension png « portable network graphics » sont enregistrées sous forme matricielle avec différents formats.

Le format des images utilisées pour le TP est **RGB float32** qui code sur 32 bits chaque couleur.

```
array([[[ 0.29411766,  0.27058825,  0.27058825], [ 0.32156864,  0.29411766,  0.30588236], ..., [
0.22352941,  0.16862746,  0.16862746]]], dtype=float32)
```

Les 3 valeurs de chaque liste correspondent aux trois couleurs RGB (Red, Green, Blue) de chaque pixel de l'image. Les valeurs des trois couleurs sont comprises entre 0 et 1.

Exemple

| | R | V | B | Couleur obtenue |
|-----------------|-----|-----|-----|-----------------|
| [0, 0, 0] | 0 | 0 | 0 | Noir |
| [1, 1, 1] | 1 | 1 | 1 | Blanc |
| [1, 0, 0] | 1 | 0 | 0 | Rouge |
| [0.3, 0.3, 0.3] | 0.3 | 0.3 | 0.3 | Gris |

On trouve aussi le codage RGBA qui ajoute une quatrième composante associée à la transparence.