

Géométrie des sections avec Python

Soit une surface plane S définie dans le plan $(O, \vec{y}, \vec{z})$.

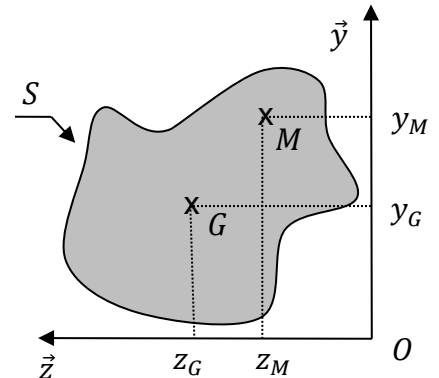
L'objectif de ce TD est de déterminer par calcul numérique les caractéristiques d'une section plane :

- l'aire S
- la position du barycentre y_G, z_G
- les moments quadratiques $I_{(G, \vec{y})}$ et $I_{(G, \vec{z})}$

Mais pour une section complexe on ne peut pas réaliser un calcul analytique d'une intégrale, on va donc réaliser un calcul numérique.

2 méthodes sont testées, il y aura donc 2 fichiers distincts :

- `section_decomposition.py`
- `section_image.py`



1 DÉCOMPOSITION EN SECTIONS ÉLÉMENTAIRES : `section_decomposition.py`

Si la section est simple on peut la décomposer en sections élémentaires.

On définit une liste pour chaque section : $S_i = [\xi_i, A_i, y_{Gi}, z_{Gi}, I_{Giy}, I_{Giz}]$

avec : ξ_i +1 ou -1 selon qu'on ajoute ou enlève la section élémentaire

A_i l'aire de la section élémentaire

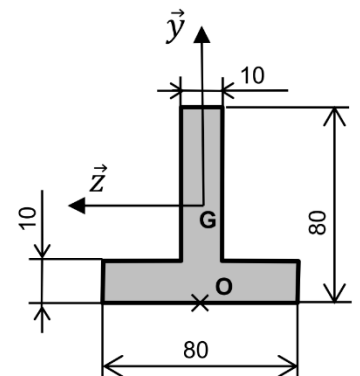
y_{Gi}, z_{Gi} la position du barycentre de la section élémentaire

I_{Giy}, I_{Giz} les moments quadratiques de la section élémentaire

Par exemple pour la section T 80×80×10 vue en TD :

$S_1 = [1, 80 \times 10, 10, 0, 10 \times 80^3 / 12, 80 \times 10^3 / 12]$

$S_2 = [1, 10 \times 70, 10 + 70/2, 0, 70 \times 10^3 / 12, 10 \times 70^3 / 12]$



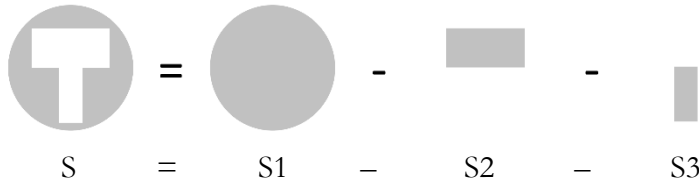
Pour calculer les caractéristiques de la section S on utilise le script suivant :

```
L_S = [S1, S2]      # liste des listes des caractéristiques des sections Si
A = 0               # initialisation de l'aire A
for Si in L_S :
    A += *** à compléter ***      # calcul de S par somme des ksi_i*A_i
yG = 0              # initialisation de la position yG
zG = 0              # initialisation de la position zG
for Si in L_S :
    yG += *** à compléter ***      # calcul de yG et de zG
    zG += *** à compléter ***      # par moyenne barycentrique
I_Gy = 0            # initialisation du moment quadratique I_Gy
I_Gz = 0            # initialisation du moment quadratique I_Gz
for Si in L_S :
    I_Gy += *** à compléter ***    # calcul de I_Gy et I_Gz
    I_Gz += *** à compléter ***    # par somme et théorème de Huyghens
print(A, yG, zG, I_Gy, I_Gz)
```

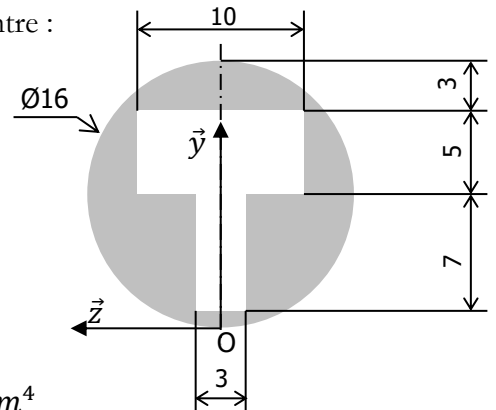
Vous devez retrouver le résultat : 1500 23.666... 0.0 432500.0 889833.333...

Modifier le script pour déterminer les caractéristiques de la section ci-contre :

La section peut être décomposée en trois sections élémentaires :



Résultats: $S = 130,0619 \text{ mm}^2$
 $y_G = 7,6040 \text{ mm}$ $z_G = 0,0 \text{ mm}$
 $I_{(G,\vec{y})} = 2784,5742 \text{ mm}^4$ $I_{(G,\vec{z})} = 2436,9320 \text{ mm}^4$



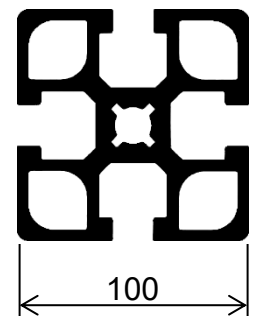
2 TRAITEMENT D'IMAGE : section_image.py

On souhaite déterminer les caractéristiques d'une section très complexe, la décomposition en sections élémentaires n'est pas envisageable.

On utilise alors une image comme celle donnée ci-contre.

On donne les fichiers image de différentes sections, ci-contre : **section3.png**

Créer un dossier, ce dossier doit contenir tous les fichiers *.png et le script **section_image_v0.py**



On utilise la méthode `imread` du module `matplotlib.pyplot` :

```
# importation des modules
import matplotlib.pyplot as plt
# importation du fichier image
image=plt.imread("section1.png")
```

Dans la console demander le type de la variable **image**.

Chaque pixel noir de l'image est utilisé comme élément de surface élémentaire dS .

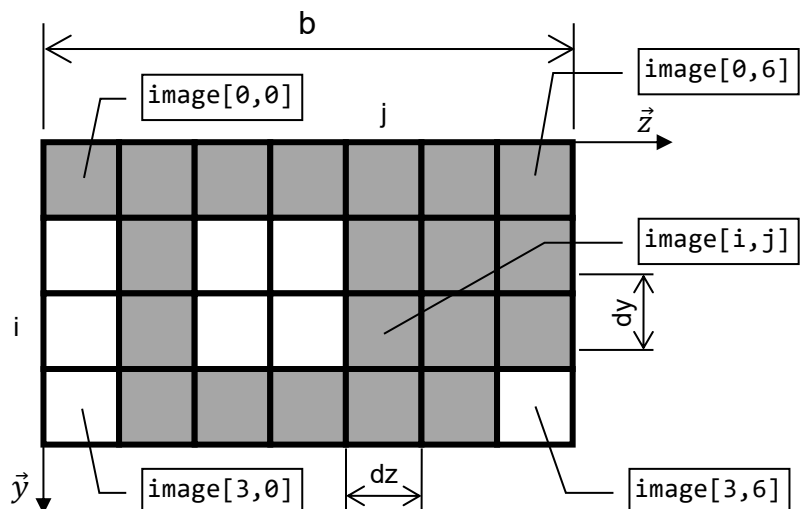
Exemple pour une image de 7×4 pixels :

La largeur en pixels de l'image correspond à la taille de la ligne **image[0]**, sa mesure en mm est **b**.

La hauteur de l'image correspond à la taille du tableau **image**

image[i,j] est un tableau de 3 valeurs comprises entre 0 et 1 correspondant au code RVB de la couleur du pixel *i,j*

image[0,0] contient [0,0,0] (noir)
image[3,6] contient [1,1,1] (blanc)



La méthode de calcul est de traduire l'écriture d'une intégrale en une somme finie :

$$I_{exact} = \int_{M \in S} f(M) \cdot dS \quad \rightarrow \quad I_{num} = \sum_{M \in S} f(M) \cdot \Delta S$$

$f(M)$ peut être n'importe quelle fonction scalaire dépendant du point M appartenant à la section S .

	Résistance des Matériaux	TP info
	GÉOMÉTRIE SECTIONS PYTHON	Page 3/6

On a par définition :

$$\text{Aire : } S = \int_{M \in S} f_1(M) \cdot dS \quad \text{avec } f_1(M) = 1$$

$$\text{Barycentre : } S \cdot \overrightarrow{OG} = \int_{M \in S} \overrightarrow{f_2}(M) \cdot dS \quad \text{avec } \overrightarrow{f_2}(M) = \overrightarrow{OM} \Leftrightarrow \begin{cases} S \cdot y_G = \int_{M \in S} f_{2a}(M) \cdot dS & \text{avec } f_{2a}(M) = y_M \\ S \cdot z_G = \int_{M \in S} f_{2b}(M) \cdot dS & \text{avec } f_{2b}(M) = z_M \end{cases}$$

$$\text{Moments quadratiques : } \begin{cases} I_{O,\vec{y}} = \int_{M \in S} f_3(M) \cdot dS & \text{avec } f_3(M) = z_M^2 \\ I_{O,\vec{z}} = \int_{M \in S} f_4(M) \cdot dS & \text{avec } f_4(M) = y_M^2 \end{cases}$$

La somme numérique va être réalisée en parcourant tous les pixels de l'image, la somme ne devra être effectuée que si le pixel est sombre ce qui correspond à l'assertion $M \in S$.

Un pixel sombre à partir d'un gris 50%, soit quand la somme des 3 valeurs RVB du pixel est inférieure à 1,5.

Écrire la fonction `pixel_noir(i,j)` qui retourne **True** si le pixel est sombre et **False** si le pixel est clair.

Tester la fonction dans la console : `pixel_noir(0,0)` doit retourner **True**
`pixel_noir(0,3)` doit retourner **True**
`pixel_noir(3,0)` doit retourner **False**

À la suite de votre script saisir et compléter les lignes suivantes :

```
b = 70 # largeur de la section en mm
nz = len(** à compléter **) # largeur de l'image en pixel
ny = len(** à compléter **) # hauteur de l'image en pixel
dz = ** à compléter ** # largeur d'un pixel en mm
dy = dz # hauteur d'un pixel en mm
dS = dy*dz # élément de surface en mm2
S,SyG,SzG,I0y,I0z = 0,0,0,0,0 # initialisation des sommes
for i in range(ny): # parcours de chaque ligne
    for j in range(nz): # parcours de chaque colonne
        if pixel_noir(i,j): # si le pixel est sombre
            S += dS # calcul de l'intégrale S
print("S=",round(S,1),"mm^2")
```

L'exécution du script doit donner : **S= 2000.0 mm^2**

Si cela fonctionne pour le calcul de l'intégrale de S cela fonctionnera pour toutes les intégrales à condition de pouvoir déterminer les coordonnées y_M et z_M du pixel de position i,j .

À l'aide de la figure de la page 2, écrire juste à la suite de la ligne d'initialisation des sommes les 2 fonctions `yM(i,j)` et `zM(i,j)` qui retournent respectivement la valeur de y_M et z_M du centre du pixel de position i,j . Les variables `dy` et `dz` étant des variables globales peuvent être utilisées.

Tester les fonctions dans la console : `yM(0,0)` doit retourner **5**
`yM(3,6)` doit retourner **35**
`zM(0,0)` doit retourner **5**
`zM(3,6)` doit retourner **65**

		Résistance des Matériaux	TP info
		GÉOMÉTRIE SECTIONS PYTHON	Page 4/6

Ajouter à la suite du calcul de l'aire S, dans le bloc exécuté si la condition pixel sombre est vérifiée :

```
SyG += yM(i,j)*dS          # calcul de l'intégrale S.yG
SzG += zM(i,j)*dS          # calcul de l'intégrale S.zG
IOy += zM(i,j)**2*dS       # calcul de l'intégrale IOy
IOz += yM(i,j)**2*dS       # calcul de l'intégrale IOz
```

Puis ajouter en fin de script

```
yG = SyG/S                  # calcul de yG
zG = SzG/S                  # calcul de zG
IGy = IOy-S*zG**2          # calcul de IGy par le théorème de Huyghens
IGz = IOz-S*yG**2          # calcul de IGz par le théorème de Huyghens
print("yG=",round(yG,2),"mm")
print("zG=",round(zG,2),"mm")
print("IGy=",round(IGy,0),"mm^4")
print("IGz=",round(IGz,0),"mm^4")
```

L'exécution du script doit donner :

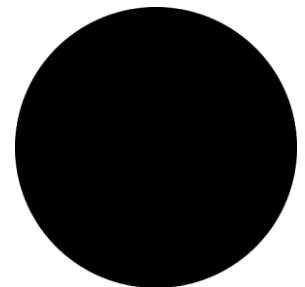
```
S= 2000.0 mm^2          yG= 18.5 mm          zG= 39.0 mm
IGy= 708000.0 mm^4      IGz= 285500.0 mm^4
```

Pour vérifier l'exactitudes des résultats on peut utiliser une section simple mais définie par une image de grande résolution, par exemple un disque.

On donne le fichier image **section2.png**, c'est une image de 1 mégapixel (1MPx)

En début de script, remplacer **section1.png** par **section2.png**

```
# importation du fichier image
Image = plt.imread("section2.png")
```



Exécuter le script et comparer les résultats obtenus avec les relations exactes.

Comment obtenir les caractéristiques d'un disque de diamètre **30mm** ?

Maintenant que le script fonctionne, on peut automatiser le calcul en définissant dans un nouveau script **section_image_V1.py** une fonction reprenant les lignes de code précédentes dans une fonction **calcul(fichier,largeur)** documentée comme cela :

```
def calcul(fichier,largeur) :
    """ Calcul de l'aire, de la position du barycentre et des 2 moments
    quadratiques centraux de la section géométrique définie par une image
    entrées : fichier (str), nom du fichier image en .png de la section
             largeur (float), largeur (cote horizontale) en mm de la section
    sortie : [S, yG, zG, IGy, IGz], (list) , liste contenant S, yG, zG, IGy, IGz
    affichage de la section avec une croix indiquant le barycentre """
```

Faire la saisie des lignes ci-dessus et copier le script précédent sous cette documentation.

Indenter ces lignes en les sélectionnant et en appuyant sur la touche Tab.

Modifier l'affectation des 2 variables **image** et **b** en utilisant les arguments de la fonction.

Terminer la définition de la fonction par

```
    return [S,yG,zG,IGy,IGz]
```

		Résistance des Matériaux	TP info
		GÉOMÉTRIE SECTIONS PYTHON	Page 5/6

Tester la fonction en appelant dans la console : `calcul("section2.png",30)`

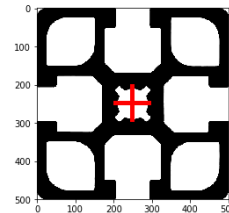
Vérifier le résultat obtenu.

On peut modifier la valeur du code RVB des pixels pour faire apparaître une croix à la position du barycentre, faire la saisie des lignes suivantes juste avant le `return`.

```
L = min(ny,nz)/10 # longueur de la croix
E = 1/10           # épaisseur de la croix
image[int(yG/dy-1):int(yG/dy+1),int(zG/dz-e):int(zG/dz+e)] = [1,0,0]
image[int(yG/dy-e):int(yG/dy+e),int(zG/dz-1):int(zG/dz+1)] = [1,0,0]
plt.imshow(image)
```

L'exécution du script ne renvoie rien mais la fonction `calcul` étant maintenant définie on peut l'utiliser dans la console, par exemple avec la section du fichier image `section3.png` et une largeur de 100 mm on doit obtenir :

```
[4032.119999996913,
49.849574913487466,
49.97392835534513,
3759471.484436104,
3764762.7635601424]
```

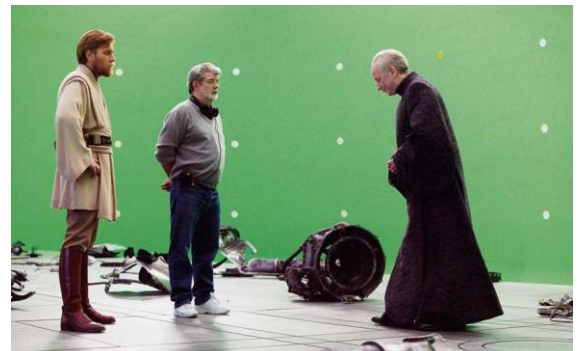


3 BONUS_TRAITEMENT D'IMAGE : fond_vert.py

Une technique d'effets spéciaux du cinéma consiste à filmer un acteur devant un fond vert puis à remplacer ce fond vert par une image d'un décor.

La couleur verte est utilisée car elle est particulièrement distincte de la couleur de la peau. Le fond vert sera facilement identifiable à condition que le motif ne présente pas de vert.

Pour cela il faut ouvrir l'image correspondant au personnage et remplacer, si le pixel est vert, la valeur du pixel de l'image par la valeur du pixel de même position de l'image du décor.



On considère que le pixel est vert si et seulement si :
$$\begin{cases} R < 0,8 \times V \\ V > 0,3 \\ B < 0,8 \times V \end{cases}$$

Pour simplifier les 2 images sont de taille identique.

Visualiser les fichiers images `personnage.png` et `fond1.png` avec un éditeur d'images.

Créer un script réalisant l'incrustation du fichier `fond1.png` dans le fichier `personnage.png`, le résultat sera sauvegardé dans le fichier `resultat1.png`

Recommencer pour les autres fonds pour permettre à Pikachu de choisir son école préférée.

Rappels :

Pour créer un tableau à partir d'un fichier image : `tableau = plt.imread("image.png")`

Pour récupérer les valeurs R, V et B du pixel i, j de l'image : `[R,V,B] = tableau[i,j]`

Pour affecter les valeurs R, V et B au pixel i, j de l'image : `tableau[i,j] = [R,V,B]`

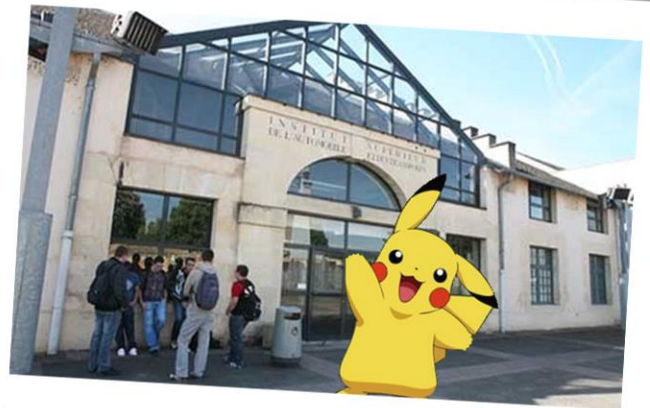
Pour copier le pixel i, j de l'image1 au pixel i, j de l'image2 : `tableau2[i,j] = tableau1[i,j]`

Pour afficher le tableau sous forme d'image dans la console : `plt.imshow(tableau)`

Pour sauvegarder le tableau sous forme d'image dans un fichier : `plt.imsave("image.png",tableau)`

	CPGE	Résistance des Matériaux	TP info
	PT	GÉOMÉTRIE SECTIONS PYTHON	Page 6/6

ENSAIT Roubaix



ISAT Nevers

SIGMA Clermont-Ferrand



ENSAM Lille



Polytech Lyon



Tatooine