Python Files\images.py

```python
import matplotlib.pyplot as plt
import numpy as np
from math import pi, cos, sin
from PIL import Image


# cf main
def droite(r: float, theta: float, precision: int = 1) -> tuple:
    theta = theta / precision
    if theta == 0 or theta == 180:
        return (0,r)
    else :
        theta_rad = (pi/180)*theta
        a = -(cos(theta_rad)/sin(theta_rad))
        b = r/sin(theta_rad)

    return(a,b)


# enregistre l'image [image] avec l'ensemble des intersections détectées, présentes dans la liste [liste_intersections]
def save_intersection(liste_intersections: list, image: list) -> None:
    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    plt.clf()
    liste_pts = []
    for i in range(len(liste_intersections)):
        _,(x,y) = liste_intersections[i]
        liste_pts.append((x,y))
    for i in range(len(liste_pts)):
        x,y = liste_pts[i]
        plt.plot(x,y,marker="+",color="red")
    plt.axis("off")
    plt.imshow(image)
    plt.savefig("07. intersections.png", dpi = 199)
    plt.close()


```

```python
# enregeistre l'espace de Hough dans un fichier
def save_hough(espace_hough: list) -> None:
    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    ax.axis('off') #enlève les axes
    ax.imshow(espace_hough, cmap="gray")
    plt.savefig("00. Hough.png", dpi=1600)
    plt.close()


# permet d'obtenir les intersections, mais a partir de la matrice d'adjacence, qui a été modifié par la fonction cluster
def save_cluster(matrice_adj: list, image: list) -> None:
    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    for i in range(len(matrice_adj)):
        x, y = matrice_adj[i][i]
        plt.plot(x,y,marker="+",color="red")
    ax.axis('off')                                                  #enlève les axes
    ax.imshow(image,  extent=[0,len(image[0])-1, len(image)-1, 0])  # permet d'avoir l'image dans le bon sens et à la bonne taille
    plt.savefig("08. cluster.png")
    plt.close()


# Enregistre un chemin renvoyé par un algorithme
# de plus court chemin tel que Dijkstra ou A*
# chemin_liste est une liste de points qui constituent le chemin
def save_chemin(chemin_liste: list, mat_adj: list, image: list, texte: str, color: str) -> None:
    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    c = -1
    for k in range(len(mat_adj)):                                   # affichage des points
        i, j = mat_adj[k][k]
        c +=1
        plt.plot(i, j,'bo')
        if k in chemin_liste :
            plt.text(i+5, j-15, f"n°{c}", c="red", fontsize=16)     # affichage des numero de sommet si il appartient au sommet
    liste_sommet = []
    for i in range(len(mat_adj)):
        liste_sommet.append(mat_adj[i][i])
    for i in range(len(chemin_liste)-1):
        x1,y1 = liste_sommet[chemin_liste[i]]
```

```
        x2,y2 = liste_sommet[chemin_liste[i+1]]
        plt.plot([x1,x2], [y1,y2], color)
    plt.imshow(image)
    ax.imshow(image, cmap = "gray",  extent=[0,len(image[0])-1, len(image)-1, 0])
    ax.axis('off') #enlève les axes
    plt.savefig(f"10. chemin - {texte}.png")
    plt.close()


# enregistre les images avec les differents segments, mis dans la liste [liste_intervalles]
def save_liste_inter(liste_intervalles: list, image: list, nom: str) -> None:
    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    for i in liste_intervalles :
        if i != [] :
            for k in i:
                x_values = [k[0][0], k[1][0]]
                y_values = [k[0][1], k[1][1]]
                plt.plot(x_values, y_values, linestyle="--")
    plt.axis('off')
    plt.imshow(image)
    plt.savefig(f"{nom}.png", dpi = 199)
    plt.close()


# enregistre l'image de la ville avec les intersections nommées
# add_str est une chaine de caractere ajoutée au nom du fichier
def save_graphe(mat_adj: list, img: list, add_str: str = "") -> None:
    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    c = -1
    for k in range(len(mat_adj)): # affichage des points
        i, j = mat_adj[k][k]
        c +=1
        plt.plot(i, j,'bo')
        plt.text(i+5, j-15, f"n°{c}", c="red", fontsize=16)
    plt.axis('off')
    plt.imshow(img)
    plt.savefig(f"09. graphe{add_str}.png", dpi = 199)


```

```python
def save_full_graphe(mat_adj: list, img: list, add_str: str = "") -> None:
    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    c = -1
    for k in range(len(mat_adj)): # affichage des points
        i, j = mat_adj[k][k]
        c +=1
        plt.plot(i, j,'bo')
        plt.text(i+5, j-15, f"n°{c}", c="red", fontsize=16)

    for k in range(len(mat_adj)): # affichage des points
        for l in range(k+1, len(mat_adj)):
            if mat_adj[k][l] != -1 and mat_adj[k][k] != (-1, -1) and mat_adj[l][l] != (-1, -1):
                x1, y1 = mat_adj[k][k]
                x2, y2 = mat_adj[l][l]
                plt.plot([x1,x2], [y1,y2])

    plt.axis('off')
    plt.imshow(img)
    plt.savefig(f"09. graphe{add_str}.png", dpi = 199)


# affiche l'image d'origine avec les droites
def save_with_droites(image: list, liste_r_theta: list, precision_hough: int = 1, name: str = "show_with_droite"):
    fig, ax = plt.subplots()
    fig.set_size_inches(18.5, 10.5)
    x = np.array(range(len(image[0])))

    for i in range(len(liste_r_theta)): #plot chaque droite
        a, b = droite(liste_r_theta[i][0], liste_r_theta[i][1], precision_hough)
        y = a*x+b
        ax.axis('off') #enlève les axes
        ax.plot(x, y, '--', linewidth=1, color="red")

    img = ax.imshow(image,cmap='gray', extent=[0,len(image[0])-1, len(image)-1, 0])
    plt.savefig(name + ".png", dpi=199)
    plt.close()


def save_custom_graphe(mat_adj: list, img: list, pts: list, add_str: str = "") -> None:
    fig, ax = plt.subplots()
```

```
    fig.set_size_inches(18.5, 10.5)
    c = 0

    for k in pts: # affichage des points
        for l in pts:
            if mat_adj[k][l] != -1 and mat_adj[k][k] != (-1, -1) and mat_adj[l][l] != (-1, -1) :
                x1, y1 = mat_adj[k][k]
                x2, y2 = mat_adj[l][l]
                plt.plot([x1,x2], [y1,y2], "b-")


    for k in pts: # affichage des points
        i, j = mat_adj[k][k]
        c +=1
        plt.plot(i, j,'ro')
        #plt.text(i+5, j-15, f"n°{c}", c="red", fontsize=16)



    plt.axis('off')
    plt.imshow(img)
    plt.savefig(f"11. custom graphe{add_str}.png", dpi = 199)
```