

Python Files\images.py

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from math import pi, cos, sin
4 from PIL import Image
5
6
7 # cf main
8 def droite(r: float, theta: float, precision: int = 1) -> tuple:
9     theta = theta / precision
10    if theta == 0 or theta == 180:
11        return (0,r)
12    else :
13        theta_rad = (pi/180)*theta
14        a = -(cos(theta_rad)/sin(theta_rad))
15        b = r/sin(theta_rad)
16
17    return(a,b)
18
19
20 # enregistre l'image [image] avec l'ensemble des intersections détectées, présentes dans la liste [liste_intersections]
21 def save_intersection(liste_intersections: list, image: list) -> None:
22     fig, ax = plt.subplots()
23     fig.set_size_inches(18.5, 10.5)
24     plt.clf()
25     liste_pts = []
26     for i in range(len(liste_intersections)):
27         _,(x,y) = liste_intersections[i]
28         liste_pts.append((x,y))
29     for i in range(len(liste_pts)):
30         x,y = liste_pts[i]
31         plt.plot(x,y,marker="+",color="red")
32     plt.axis("off")
33     plt.imshow(image)
34     plt.savefig("07. intersections.png", dpi = 199)
35     plt.close()
36
37
```

```

38 # enregistre l'espace de Hough dans un fichier
39 def save_hough(espace_hough: list) -> None:
40     fig, ax = plt.subplots()
41     fig.set_size_inches(18.5, 10.5)
42     ax.axis('off') #enlève les axes
43     ax.imshow(espace_hough, cmap="gray")
44     plt.savefig("00. Hough.png", dpi=1600)
45     plt.close()
46
47
48 # permet d'obtenir les intersections, mais a partir de la matrice d'adjacence, qui a été modifié par la fonction cluster
49 def save_cluster(matrice_adj: list, image: list) -> None:
50     fig, ax = plt.subplots()
51     fig.set_size_inches(18.5, 10.5)
52     for i in range(len(matrice_adj)):
53         x, y = matrice_adj[i][i]
54         plt.plot(x,y,marker="+",color="red")
55     ax.axis('off') #enlève les axes
56     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
57     plt.savefig("08. cluster.png")
58     plt.close()
59
60
61 # Enregistre un chemin renvoyé par un algorithme
62 # de plus court chemin tel que Dijkstra ou A*
63 # chemin_liste est une liste de points qui constituent le chemin
64 def save_chemin(chemin_liste: list, mat_adj: list, image: list, texte: str, color: str) -> None:
65     fig, ax = plt.subplots()
66     fig.set_size_inches(18.5, 10.5)
67     c = -1
68     for k in range(len(mat_adj)): # affichage des points
69         i, j = mat_adj[k][k]
70         c +=1
71         plt.plot(i, j, 'bo')
72         if k in chemin_liste :
73             plt.text(i+5, j-15, f"n°{c}", c="red", fontsize=16) # affichage des numero de sommet si il appartient au sommet
74     liste_sommet = []
75     for i in range(len(mat_adj)):
76         liste_sommet.append(mat_adj[i][i])
77     for i in range(len(chemin_liste)-1):
78         x1,y1 = liste_sommet[chemin_liste[i]]

```

```

79     x2,y2 = liste_sommet[chemin_liste[i+1]]
80     plt.plot([x1,x2], [y1,y2], color)
81     plt.imshow(image)
82     ax.imshow(image, cmap = "gray", extent=[0,len(image[0])-1, len(image)-1, 0])
83     ax.axis('off') #enlève les axes
84     plt.savefig(f"10. chemin - {texte}.png")
85     plt.close()
86
87
88 # enregistre les images avec les differents segments, mis dans la liste [liste_intervalles]
89 def save_liste_inter(liste_intervalles: list, image: list, nom: str) -> None:
90     fig, ax = plt.subplots()
91     fig.set_size_inches(18.5, 10.5)
92     for i in liste_intervalles :
93         if i != [] :
94             for k in i:
95                 x_values = [k[0][0], k[1][0]]
96                 y_values = [k[0][1], k[1][1]]
97                 plt.plot(x_values, y_values, linestyle="--")
98     plt.axis('off')
99     plt.imshow(image)
100    plt.savefig(f"{nom}.png", dpi = 199)
101    plt.close()
102
103
104 # enregistre l'image de la ville avec les intersections nommées
105 # add_str est une chaine de caractere ajoutée au nom du fichier
106 def save_graphe(mat_adj: list, img: list, add_str: str = "") -> None:
107     fig, ax = plt.subplots()
108     fig.set_size_inches(18.5, 10.5)
109     c = -1
110     for k in range(len(mat_adj)): # affichage des points
111         i, j = mat_adj[k][k]
112         c +=1
113         plt.plot(i, j, 'bo')
114         plt.text(i+5, j-15, f"n°{c}", c="red", fontsize=16)
115     plt.axis('off')
116     plt.imshow(img)
117     plt.savefig(f"09. graphe{add_str}.png", dpi = 199)
118
119

```

```

120 def save_full_graphe(mat_adj: list, img: list, add_str: str = "") -> None:
121     fig, ax = plt.subplots()
122     fig.set_size_inches(18.5, 10.5)
123     c = -1
124     for k in range(len(mat_adj)): # affichage des points
125         i, j = mat_adj[k][k]
126         c +=1
127         plt.plot(i, j, 'bo')
128         plt.text(i+5, j-15, f"n°{c}", c="red", fontsize=16)
129
130     for k in range(len(mat_adj)): # affichage des points
131         for l in range(k+1, len(mat_adj)):
132             if mat_adj[k][l] != -1 and mat_adj[k][k] != (-1, -1) and mat_adj[l][l] != (-1, -1):
133                 x1, y1 = mat_adj[k][k]
134                 x2, y2 = mat_adj[l][l]
135                 plt.plot([x1,x2], [y1,y2])
136
137     plt.axis('off')
138     plt.imshow(img)
139     plt.savefig(f"09. graphe{add_str}.png", dpi = 199)
140
141
142 # affiche l'image d'origine avec les droites
143 def save_with_droites(image: list, liste_r_theta: list, precision_hough: int = 1, name: str = "show_with_droite"):
144     fig, ax = plt.subplots()
145     fig.set_size_inches(18.5, 10.5)
146     x = np.array(range(len(image[0])))
147
148     for i in range(len(liste_r_theta)): #plot chaque droite
149         a, b = droite(liste_r_theta[i][0], liste_r_theta[i][1], precision_hough)
150         y = a*x+b
151         ax.axis('off') #enlève les axes
152         ax.plot(x, y, '--', linewidth=1, color="red")
153
154     img = ax.imshow(image, cmap='gray', extent=[0, len(image[0])-1, len(image)-1, 0])
155     plt.savefig(name + ".png", dpi=199)
156     plt.close()
157
158
159 def save_custom_graphe(mat_adj: list, img: list, pts: list, add_str: str = "") -> None:
160     fig, ax = plt.subplots()

```

```
161 fig.set_size_inches(18.5, 10.5)
162 c = 0
163
164 for k in pts: # affichage des points
165     for l in pts:
166         if mat_adj[k][l] != -1 and mat_adj[k][k] != (-1, -1) and mat_adj[l][l] != (-1, -1) :
167             x1, y1 = mat_adj[k][k]
168             x2, y2 = mat_adj[l][l]
169             plt.plot([x1,x2], [y1,y2], "b-")
170
171
172 for k in pts: # affichage des points
173     i, j = mat_adj[k][k]
174     c +=1
175     plt.plot(i, j, 'ro')
176     #plt.text(i+5, j-15, f"n°{c}", c="red", fontsize=16)
177
178
179
180 plt.axis('off')
181 plt.imshow(img)
182 plt.savefig(f"11. custom graphe{add_str}.png", dpi = 199)
183
184
185
```