

UNIVERSIDAD DE GUADALAJARA



CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Sem. Algoritmia

Reporte de práctica

Nombre del alumno:	Ricardo David Lopez Arellano
Profesor:	Erasmó Gabriel Martínez Soltero
Título de la práctica:	"Tarea 10. PRIM"
Fecha:	21 abril 2023

Introducción

En esta practica se realizo el algoritmo de PRIM.

Metodología

```
C:\Users\david_000\Desktop\7mo semestre\SEM.ALGORITMIA\Nueva carpeta (2)\Tarea_9.py

Tarea_9.py x
1  import cv2
2  import numpy as np
3  from collections import defaultdict
4
5  # con esta funcion reviso si un punto ya esta en alguna de mis listas
6  def isInTheList(elemento, arreglo):
7      for i in arreglo:
8          if np.array_equal(i, elemento):
9              return True
10         return False
11
12 # para cargar el mapa
13 mapa = cv2.imread('mapa_3.png')
14 # pasamos la imagen a escala de grises
15 gray = cv2.cvtColor(mapa, cv2.COLOR_BGR2GRAY)
16 # muestro la imagen en escala de grises
17 cv2.imshow('mapa', gray)
18 cv2.waitKey()
19 # obtengo una binarizacion en blanco todos lo pixeles cuyo valor en sea entre 254 y 255
20 ret, th1 = cv2.threshold(gray, 254, 255, cv2.THRESH_BINARY)
21 # hago un kernel de 11x11 de unos. Los Kernels se acostumbra hacerse de tamaño no par y cuadrados
22 # para que se den una idea algo así:
23 """
24 1 1 1 1 1 1 1 1 1 1 1
25 1 1 1 1 1 1 1 1 1 1 1
26 1 1 1 1 1 1 1 1 1 1 1
27 1 1 1 1 1 1 1 1 1 1 1
28 1 1 1 1 1 1 1 1 1 1 1
29 1 1 1 1 1 1 1 1 1 1 1
30 1 1 1 1 1 1 1 1 1 1 1
31 1 1 1 1 1 1 1 1 1 1 1
32 1 1 1 1 1 1 1 1 1 1 1
33 1 1 1 1 1 1 1 1 1 1 1
34 1 1 1 1 1 1 1 1 1 1 1
35 """
36 kernel = np.ones((11, 11), np.uint8)
37 # aplico un filtro de dilatacion. Este filtro hace que los puntos blancos se expandan
38 # probocando que algunos puntitos negros desaparezcan #le pueden hacer un cv.imshow para que vean el resultado
39 th1 = cv2.dilate(th1, kernel, 1)
40 kernel = np.ones((11, 11), np.uint8)
41 # Despues aplico uno de erosion que hace lo opuesto al de dilatacion
42 th1 = cv2.erode(th1, kernel, 1)
43 # aplico un flitro gausiando de 5x5 para suavisar los bordes
44 th1 = cv2.GaussianBlur(th1, (5, 5), cv2.BORDER_DEFAULT)
```

C:\Users\david_000\Desktop\7mo semestre\SEM.ALGORITMIA\Nueva carpeta (2)\Tarea_9.py

```
Tarea_9.py x
43 # aplico un filtro gausiano de 5x5 para suavizar los bordes
44 th1 = cv2.GaussianBlur(th1, (5, 5), cv2.BORDER_DEFAULT)
45 # muestro como queda mi mapa
46 cv2.imshow('thres', th1)
47 cv2.waitKey()
48 # Aplico la deteccion de Esquinas de Harris. para mas informacion consulten https://doc
49 dst = cv2.cornerHarris(th1, 2, 3, 0.05)
50 ret, dst = cv2.threshold(dst, 0.04 * dst.max(), 255, 0)
51 dst = np.uint8(dst)
52 ret, th2 = cv2.threshold(th1, 235, 255, cv2.THRESH_BINARY)
53 th2 = cv2.dilate(th2, kernel, 1)
54 # aqui devuelvo la imagen binarizada a tres canales
55 th2 = cv2.cvtColor(th2, cv2.COLOR_GRAY2BGR)
56 # find centroids
57 ret, labels, stats, centroids = cv2.connectedComponentsWithStats(dst, 30, cv2.CV_32S)
58 vertices = np.int0(centroids)
59
60 aux1 = vertices
61 aux2 = vertices
62 verticesConectados = []
63 aristas = []
64 # aqui voy a buscar cuales son las esquinas que estan conectadas
65 for h in range(len(aux1)):
66     i = aux1[h]
67     for k in range(h, len(aux2)):
68         j = aux2[k]
69         if not (i == j).all():
70             print(i, end='')
71             print(j)
72             p1 = (i+j)/2
73             p2 = (p1+j)/2
74             p3 = (p1+i)/2
75             p4 = (p2+j)/2
76             p5 = (p2+i)/2
77             p6 = (p2+p1)/2
78             p7 = (p1+p3)/2
79
80             if(th2[int(p1[1])][int(p1[0])] == [255, 255, 255]).all() and \
81                (th2[int(p2[1])][int(p2[0])] == [255, 255, 255]).all() and \
82                (th2[int(p3[1])][int(p3[0])] == [255, 255, 255]).all() and \
83                (th2[int(p4[1])][int(p4[0])] == [255, 255, 255]).all() and \
84                (th2[int(p5[1])][int(p5[0])] == [255, 255, 255]).all() and \
85                (th2[int(p6[1])][int(p6[0])] == [255, 255, 255]).all() and \
86                (th2[int(p7[1])][int(p7[0])] == [255, 255, 255]).all():
```

C:\Users\dauid_000\Desktop\7mo semestre\SEM.ALGORITMIA\Nueva carpeta (2)\Tarea_9.py

```
Tarea_9.py x
86         (th2[int(p7[1])][int(p7[0])] == [255, 255, 255]).all():
87             costA = np.linalg.norm(i-j)
88             aristas.append([i,j,costA])
89             if not isInTheList(i,verticesConectados):
90                 verticesConectados.append(i)
91             if not isInTheList(j,verticesConectados):
92                 verticesConectados.append(j)
93
94     graph = {'nodos': verticesConectados, 'aristas': aristas}
95
96     k = defaultdict(list)
97
98     for n1,n2,c in graph['aristas']:
99         k[n1[0],n1[1]].append((n2,c))
100        k[n2[0], n2[1]].append((n1, c))
101
102
103     #A continuación esta el algoritmo de PRIM, el algoritmo lo saque de la siguiente pagina
104     # https://gist.github.com/davcastroruiz/f7e041ac68970d339176c0be4631d07b
105
106     listaVisitados = []
107     grafoResultante = {}
108     listaOrdenada = []
109     resultado = []
110
111     #1. Elegir el nodo de origen
112     origen = verticesConectados[0]
113     #2. Agregarlo a la lista de visitados
114     listaVisitados.append(origen)
115     #3. Agregar sus aydacentes a la lista ordenada
116     for destino, peso in k[origen[0],origen[1]]:
117         listaOrdenada.append((origen, destino, peso))
118     '''ORDENAMIENTO INSERT PARA LA LISTA'''
119     while listaOrdenada:
120         pos=0
121         act=0
122         listAux=[]
123         for i in range(len(listaOrdenada)):
124             listAux=listaOrdenada[i]
125             act=listaOrdenada[i][2]
126             pos=i
127             while pos> 0 and listaOrdenada[pos-1][2] > act:
128                 listaOrdenada[pos] = listaOrdenada[pos-1]
129                 pos=pos-1
```


C:\Users\david_000\Desktop\-\7mo semestre\SEM.ALGORITMIA\Nueva carpeta (2)\Tarea_9.py

```
Tarea_9.py x
129         pos=pos-1
130         listaOrdenada[pos]=listAux
131
132     #4. Tomar vertice de la lista ordenada y eliminarlo
133     vertice = listaOrdenada.pop(0)
134     d = vertice[1]
135
136     #5. Si el destino no esta en la lista de visitados
137     if not isInTheList(d,listaVisitados):
138     #6. Agregar a la lista el nodo destino
139         listaVisitados.append(d)
140     #7. Agregar a la lista los aydacentes del nodo destino
141         for key, lista in k[d[0],d[1]]:
142             if not isInTheList(key,listaVisitados):
143                 listaOrdenada.append((d, key, lista))
144
145         origen = vertice[0]
146         destino = vertice[1]
147         peso = vertice[2]
148
149     #8. Agregar vertice al nodo resultante
150     if isInTheList(origen, grafoResultante):
151         if isInTheList(destino, grafoResultante):
152             lista = grafoResultante[origen[0], origen[1]]
153             grafoResultante[origen[0], destino[1]] = lista + [origen, destino, peso]
154             lista = grafoResultante[destino[0], destino[1]]
155             lista.append([origen, destino, peso])
156             grafoResultante[destino[0], destino[1]] = lista
157         else:
158             grafoResultante[destino[0],destino[1]] = [origen, destino, peso]
159             lista = grafoResultante[origen[0],origen[1]]
160             lista.append([origen, destino, peso])
161             grafoResultante[origen[0],origen[1]] = lista
162     elif isInTheList(destino, grafoResultante):
163         grafoResultante[origen] = [origen, destino, peso]
164         lista = grafoResultante [destino]
165         lista.append([origen, destino, peso])
166         grafoResultante[destino] = lista
167     else:
168         grafoResultante[destino[0],destino[1]] = [origen, destino, peso]
169         grafoResultante[origen[0],origen[1]] = [origen, destino, peso]
170
171     resultado = []
172
```

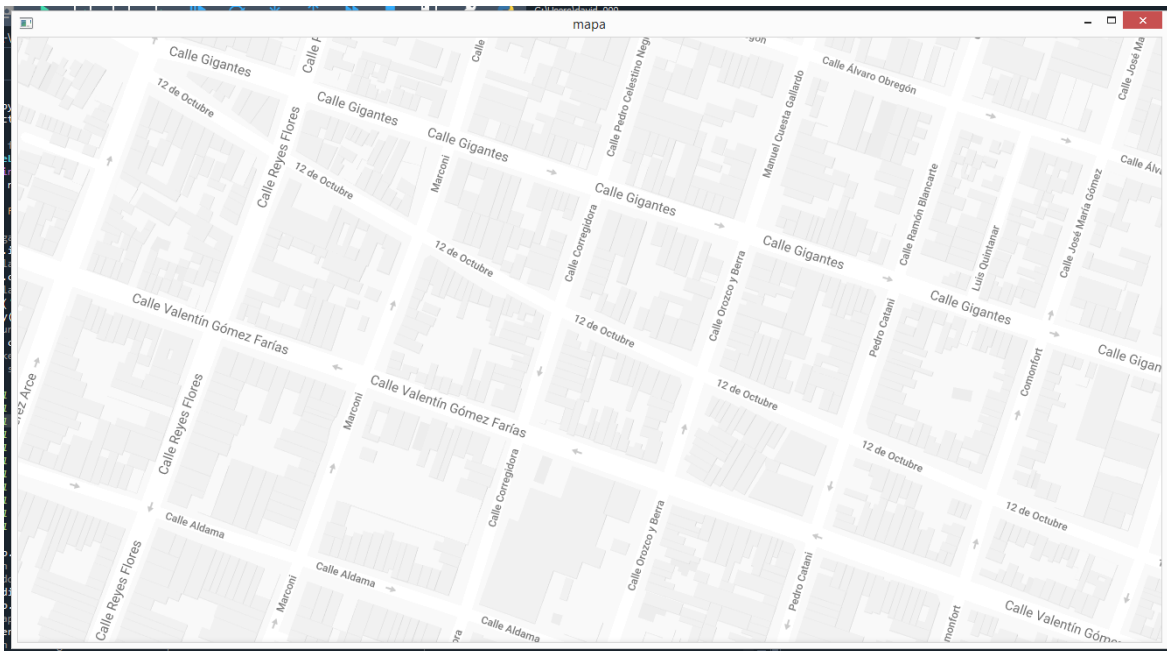
New file

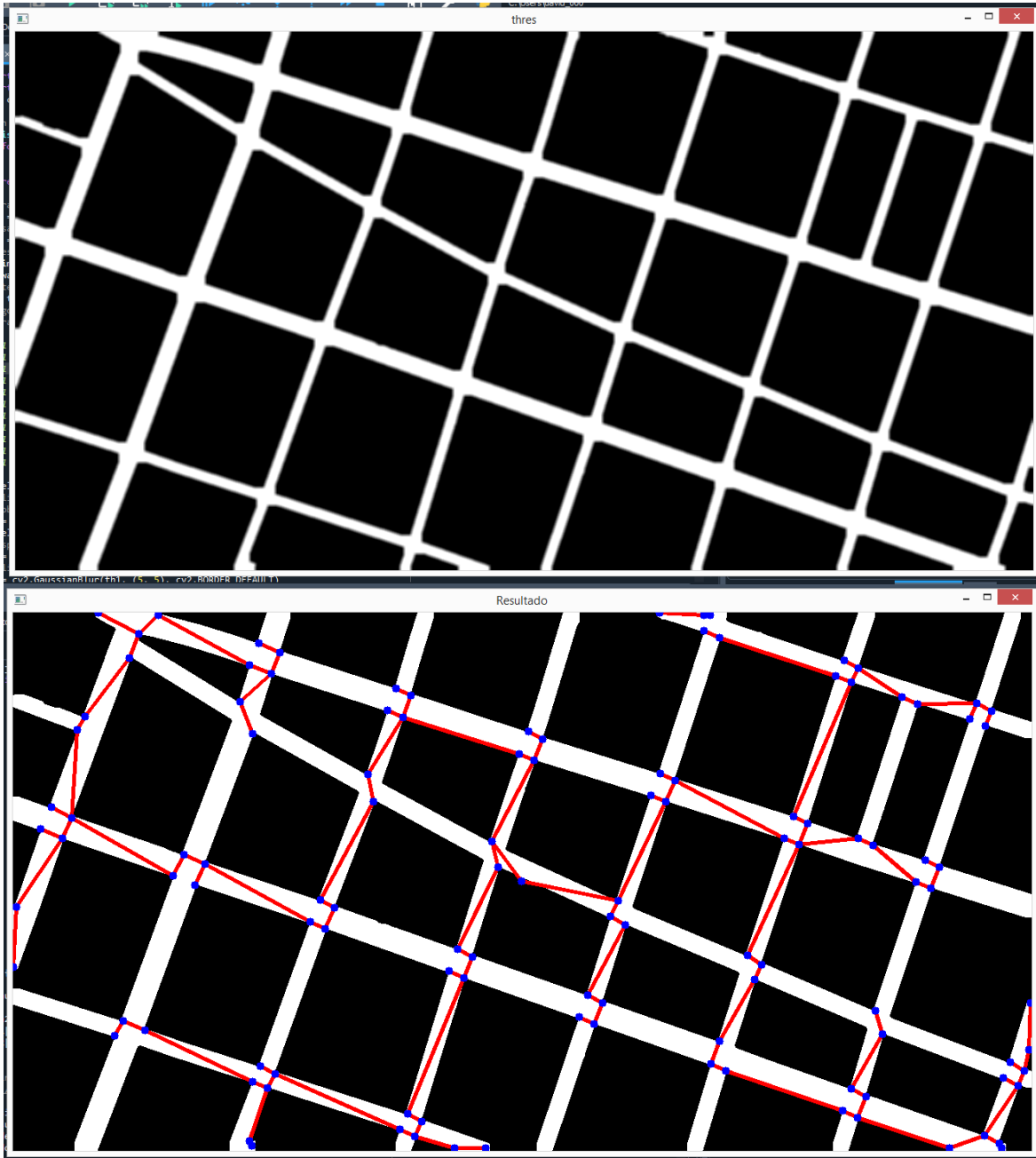


```
172
173     for key, lista in grafoResultante.items():
174         resultado.append(lista)
175
176     for arista in resultado:
177         cv2.line(th2, tuple(arista[0]), tuple(arista[1]), (0, 0, 255), 3)
178
179     for point in vertices:
180         cv2.circle(th2, (point[0], point[1]), 5, (255, 0, 0), -1)
181         cv2.waitKey(1)
182
183     # aqui muestro como quedo de chingon el grafo
184     cv2.imshow('Resultado', th2)
185     cv2.waitKey()
186
```



Resultados





Conclusiones

En esta actividad se logro la practica como se deseaba, se hizo el algoritmo de PRIM y se subio a youtube un video de su funcionamiento, a continuación el link del video:

<https://youtu.be/AAVnM2Bu-Ac>