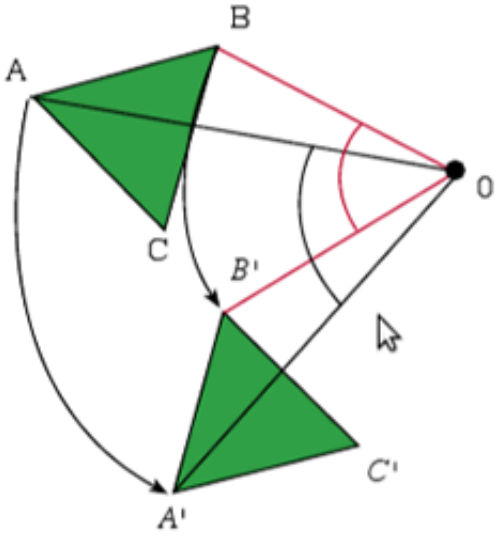
GEOMETRÍA COMPUTACIONAL

TRANSFORMACIÓN ISOMÉTRICA AFÍN



LUCAS DE TORRE

Índice

1. Introducción

Dado cualquier sistema $S = \{a^j, \{x^j, y^j, ...\}\}_{j=1}^{n}$ queremos aplicar una transformación isométrica afín correspondiente a una rotación aplicada en torno al centroide y una traslación.

Partiendo primero de una figura tridimensional, buscamos modificarla y, posteriormente, realizar una animación de una familia paramétrica continua hasta la trasformación simultánea de una rotación de $\theta = 3\pi$ y una traslación con $v = (d, d, 0)$, donde d es el diámetro mayor de S.

Después, dado el sistema representado por "arbol.png", consideraremos el subsistema σ dado por el primer color cuando $rojo < 240$ y hallaremos el centroide del subsitema. Por último, aplicaremos la misma transformación isométrica afín que antes al subsistema σ.

2. Material

Primero, modificamos la figura dada cambiando su color. Después, hallamos su diámetro $d1$.

Después, creamos la función *transf3D*, la cual aplica a un conjunto de puntos en el espacio la transformación dada por una matriz junto con el desplazamiento dado por un vector. También declaramos la función *animate3D*, la cual contiene la matriz M de rotación y el vector $v1 = (d1, d1, 0)$ de desplazamiento y aplica la transformación dada por M y $v1$ a un sistema. Además, hallamos el centroide de la figura para que la rotación se realice en torno al centroide. Definimos la matriz M como:

$$M = \begin{pmatrix} \cos(3\pi t) & -\sin(3\pi t) & 0 \\ \sin(3\pi t) & \cos(3\pi t) & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Finalmente, mediante la función *animation.FuncAnimation* (que hace uso de la función animate3D) creamos la animación de nuestra figura.

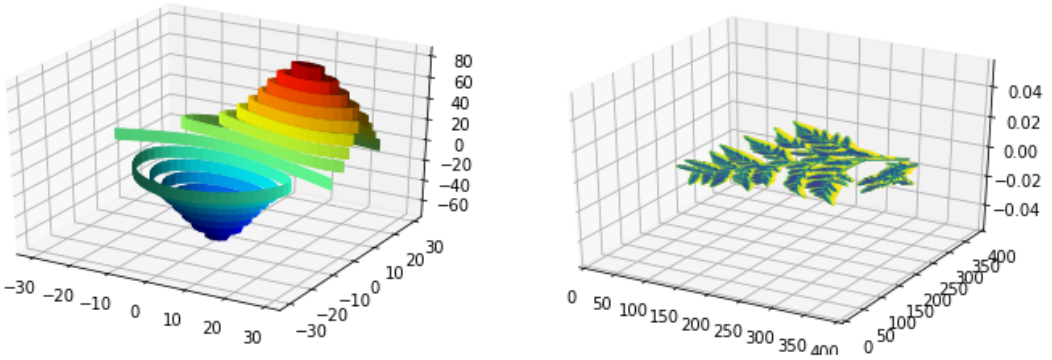
Por otro lado, dado el sistema dado por "arbol.png", consideramos el subsistema σ. Hallamos su centroide como la media de todos sus puntos y también hallamos su diámetro $d2$.

Por último, teniendo definidas las funciones *transf2D* (que aplica a un conjunto de puntos en un plano la transformación dada por una matriz junto con el desplazamiento dado por un vector) y *animate2D* (que contiene la matriz M de rotación, que es la misma que antes, y el vector $v2 = (d2, d2, 0)$ de desplazamiento y aplica la

transformación dada por M y $v2$ a un sistema), haciendo uso de la función *animation.FuncAnimation* (que hace uso de la función animate2D) creamos la animación de nuestra figura.

3. Resultados

Podemos ver la primera figura modificada a la izquierda. La animación de la figura de la izquierda está en el archivo *p7a.gif* y la animación de la figura de la derecha está en el archivo *p7b.gif*.



El centroide de la figura de la derecha se sitúa en el punto (173.482, 204.156, 0).

4. Conclusiones

Llama la atención lo mucho que se acorta el tiempo de ejecución si, al hallar el diámetro de una figura, en lugar de hacer las distancias entre todos los puntos calculamos el cuadrado de esas distancias. Esto refleja claramente lo costoso que es realizar raíces cuadradas.

5. Anexo A: Código

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 10 18:58:33 2020
@author: Jorge Sainero y Lucas de Torre con la plantilla de Robert
"""

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from matplotlib import animation
from skimage import io, color


"""
Ejemplo para el apartado 1.
Modifica la figura 3D y/o cambia el color
https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html
"""

######################      APARTADO 1
    ############################
def animate3D(t):
    M = np.array([[np.cos(3*np.pi*t),-np.sin(3*np.pi*t),0],[np.sin
    (3*np.pi*t),np.cos(3*np.pi*t),0],[0,0,1]])

    v=np.array([d,d,0])*t

    ax = plt.axes(xlim=(0,200), ylim=(0,200), projection='3d')
    #ax.view_init(60, 30)

    x,y,z = transf3D(X-bx, Y-by, Z-bz, M=M, v=v)
    ax.contour(x, y, z, 16, extend3d=True,cmap = plt.cm.get_cmap('
    jet'))
    return ax,

def init2():
    return animate3D(0),

def transf3D(x,y,z,M, v=np.array([0,0,0])):
    xt = np.zeros(shape=(len(x),len(x)))
    yt = np.zeros(shape=(len(x),len(x)))
    zt = np.zeros(shape=(len(x),len(x)))
    for i in range(len(x)):
        for j in range(len(x)):
            q = np.array([x[i][j],y[i][j],z[i][j]])
            xt[i][j], yt[i][j], zt[i][j] = np.matmul(M, q) + v
    return xt, yt, zt
```

```
45
46
47 def diametro3D(x,y,z):
48     d=0
49     xaux=x.reshape(-1)
50     yaux=y.reshape(-1)
51     zaux=z.reshape(-1)
52     for i in range(len(xaux)):
53         for j in range(i+1,len(xaux)):
54             aux=(xaux[i]-xaux[j])*(xaux[i]-xaux[j]) +(yaux[i]-yaux[
    j])*(yaux[i]-yaux[j]) +(zaux[i]-zaux[j])*(zaux[i]-zaux[j])
55             if (aux>d):
56                 d=aux
57     return np.sqrt(d)
58
59
60 fig = plt.figure()
61 ax = plt.axes(projection='3d')
62 X, Y, Z = axes3d.get_test_data(0.05)
63 cset = ax.contour(X, Y, Z, 16, extend3d=True,cmap = plt.cm.get_cmap
    ('jet'))
64 ax.clabel(cset, fontsize=9, inline=1)
65 plt.show()
66
67 d = diametro3D(X,Y,Z)
68 bx=np.mean(X)
69 by=np.mean(Y)
70 bz=np.mean(Z)
71
72 animate3D(np.arange(0.1, 1,0.1)[5])
73 plt.show()
74
75
76 fig = plt.figure(figsize=(6,6))
77 ani = animation.FuncAnimation(fig, animate3D, frames=np.arange
    (0,1,0.025), init_func=init2,
78                               interval=20)
79
80 ani.save("p7a.gif", fps = 10)
81
82
83
84
85 ######################      APARTADO 2
    ############################
86
87 def animate2D(t):
88     M = np.array([[np.cos(3*np.pi*t),-np.sin(3*np.pi*t),0],[np.sin
    (3*np.pi*t),np.cos(3*np.pi*t),0],[0,0,1]])
```

```
89
90      v=np.array([d,d,0])*t
91
92      ax = plt.axes(xlim=(0,400), ylim=(0,400), projection='3d')
93      #ax.view_init(60, 30)
94
95      XYZ = transf2D(x0-bx, y0-by, z0, M=M, v=v)
96      col = plt.get_cmap("viridis")(np.array(0.1+XYZ[2]))
97      ax.scatter(XYZ[0],XYZ[1],c=col,s=0.1,animated=True)
98      return ax,
99
100 def init():
101     return animate2D(0),
102
103 """
104 Transformaci n para el segundo apartado
105 NOTA: Para el primer aparado es necesario adaptar la funci n o
        crear otra similar
106 pero teniendo en cuenta m s dimensiones
107 """
108
109 def transf2D(x,y,z,M, v=np.array([0,0,0])):
110     xt = np.empty(len(x))
111     yt = np.empty(len(x))
112     zt = np.empty(len(x))
113     for i in range(len(x)):
114         q = np.array([x[i],y[i],z[i]])
115         xt[i], yt[i], zt[i] = np.matmul(M, q) + v
116     return xt, yt, zt
117
118 def diametro2D(x,y):
119     d=0
120     for i in range(len(x)):
121         for j in range(i+1,len(x)):
122             aux=(x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j])
123             if (aux>d):
124                 d=aux
125     return np.sqrt(d)
126
127 """
128 Segundo apartado casi regalado
129 Imagen del  rbol
130 """
131
132
133 img = io.imread('arbol.png')
134 dimensions = color.guess_spatial_dimensions(img)
135 print(dimensions)
136 io.show()
```

```
137 #io.imsave('arbol2.png',img)
138
139 #https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html
140 fig = plt.figure(figsize=(5,5))
141 p = plt.contourf(img[:,:,0],cmap = plt.cm.get_cmap('viridis'),
        levels=np.arange(0,240,2))
142 plt.axis('off')
143 #fig.colorbar(p)
144
145 xyz = img.shape
146
147 x = np.arange(0,xyz[0],1)
148 y = np.arange(0,xyz[1],1)
149 xx,yy = np.meshgrid(x, y)
150 xx = np.asarray(xx).reshape(-1)
151 yy = np.asarray(yy).reshape(-1)
152 z  = img[:,:,0]
153 zz = np.asarray(z).reshape(-1)
154
155
156 """
157 Consideraremos s lo los elementos con zz < 240
158 Por curiosidad, comparamos el resultado con contourf y scatter!
159 """
160 #Variables de estado coordenadas
161 x0 = xx[zz<240]
162 y0 = yy[zz<240]
163 z0 = zz[zz<240]/256.
164 #Variable de estado: color
165 col = plt.get_cmap("viridis")(np.array(0.1+z0))
166
167 fig = plt.figure(figsize=(5,5))
168 ax = fig.add_subplot(1, 2, 1)
169 plt.contourf(x,y,z,cmap = plt.cm.get_cmap('viridis'), levels=np.
        arange(0,240,2))
170 ax = fig.add_subplot(1, 2, 2)
171 plt.scatter(x0,y0,c=col,s=0.1)
172 plt.show()
173
174
175
176 d = diametro2D(x0,y0)
177 bx=np.mean(x0)
178 by=np.mean(y0)
179
180 print ("El centroide de la imagen, se situa en las coordenadas:(",
        round(bx,3),round(by,3),")")
181 animate2D(np.arange(0.1, 1,0.1)[5])
182 plt.show()
```

```
183
184
185 fig = plt.figure(figsize=(6,6))
186 ani = animation.FuncAnimation(fig, animate2D, frames=np.arange
        (0,1,0.025), init_func=init,
187                                 interval=20)
188
189 ani.save("p7b.gif", fps = 10)
```