MEMORIA PRÁCTICA 4 - GCOM: Transformación isométrica afín

Nombre: Gabriel Alba Serrano
Subgrupo: U1

1. Introducción

La transformación isométrica afín es un tipo de aplicación geométrica que preserva la forma y el tamaño de los objetos mientras los mueve en el espacio. Las diferentes transformaciones isométricas afínes se pueden generalizar como la composición de una rotación junto con una traslación.

En una transformación isométrica afín, se mantienen las distancias entre los puntos del objeto, así como los ángulos entre las líneas que lo componen. Además, las transformaciones isométricas y afines son lineales, lo que significa que se pueden representar por medio de matrices y vectores.

La transformación isométrica afín se utiliza en diversos campos, como la gráfica por computadora, la robótica, la ingeniería y la física. En gráfica por computadora, se utiliza para transformar modelos 3D y animaciones, mientras que en robótica se utiliza para controlar los movimientos de los robots. En ingeniería, se utiliza para diseñar estructuras y sistemas, y en física se utiliza para describir la transformación de los sistemas de coordenadas.

2. Material usado

2.1. Método

Utilizando el software de python3 y sus librerías matplotlib y numpy, he calculado la transformación isométrica afín de un sistema de N elementos y además he generado una animación que muestra dicha transformación.

2.2. Datos

La plantilla 'GCOM2023-Practica4-plantilla.py' y la imagen 'arbol.png', de la cuál extraemos el conjunto de píxeles sobre los que vamos a trabajar en el apartado 2.

3. Resultados

1. He tomado como sistema de N elementos un cono utilizando la siguiente parametrización:

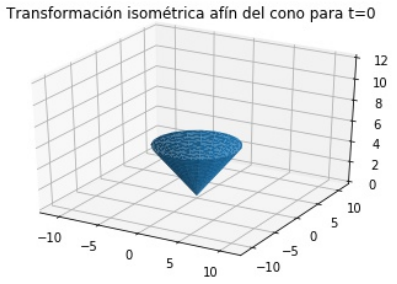
$$x = vcos(u)$$
$$y = vsen(u)$$
$$z = v$$

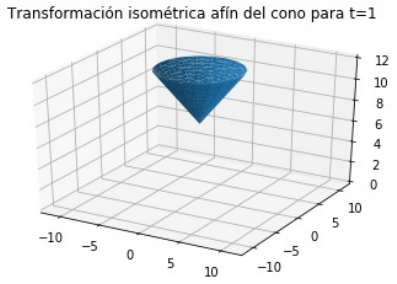
para $u \in [0, 2\pi]$ y $v \in [0, 5]$. Posteriormente he realizado una animación del cono parametrizado S según una familia paramétrica continua con parámetro $t \in [0, 1]$ que reproduce desde la posición original del cono hasta la transformación final, compuesta por una rotación de $\theta = 3\pi$ entorno al centroide C de S y una traslación con $v = (0, 0, d)$ donde d es el diámetro mayor de S.
La composición de la rotación con la traslación respecto a $t \in [0, 1]$ transforma cada punto $q = (x, y, z) \in S$ en $q' = (x', y', z')$ tal que:

$$q' = C + (q - C)M_t + vt$$

siendo

$$M_t = \begin{pmatrix} cos(t\theta) & -sin(t\theta) & 0 \\ sen(t\theta) & cos(t\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad C = \begin{pmatrix} \frac{\sum_{i=1}^{N} x_i}{N} \\ \frac{\sum_{i=1}^{N} y_i}{N} \\ \frac{\sum_{i=1}^{N} z_i}{N} \end{pmatrix} \tag{1}$$



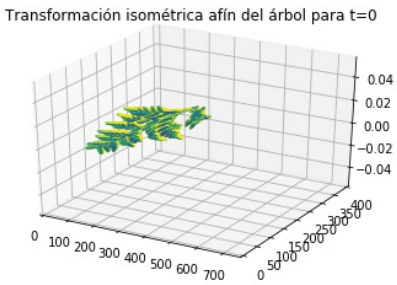


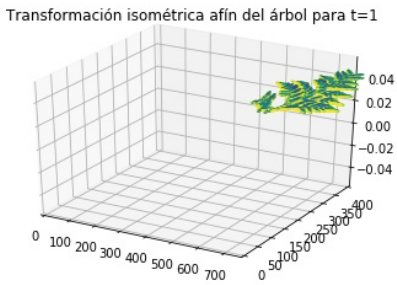
Adjunto animación 'p4i.gif' que transforma el cono desde $t = 0$ hasta $t = 1$

2. Considerando el subsistema σ repesentado por los píxeles de la imagen digital 'arbol.png' que tienen su segundo color $verde < 240$, he obtenido que su centroide es:

$$C_\sigma = \begin{pmatrix} 173{,}6979 \\ 204{,}5345 \\ 0 \end{pmatrix}$$

Además he aplicado una transformación isométrica afín que consiste en una rotación con $\theta = 3\pi$ entorno al centroide C_σ compuesta con una traslación del vector $v = (d, d, 0)$ siendo d el diámetro mayor de σ





Adjunto animación 'p4ii.gif' que transforma el subsistema σ desde $t = 0$ hasta $t = 1$

4. Conclusión

Cabe destacar que efectivamente una transformación isométrica afín preserva la forma y tamaño, o lo que es lo mismo, la distancia entre los puntos que forman el sistema transformado. Se puede observar claramente en las animaciones.

5. Anexo con el script/código utilizado

```
1
2  """
3  Practica 4 - GCOM: Transformacion isometrica afin
4  NOMBRE: Gabriel Alba Serrano
5  SUBGRUPO: U1
6  """
7
8  import numpy as np
9  import matplotlib.pyplot as plt
10 from mpl_toolkits.mplot3d import axes3d
11 from matplotlib import animation
12 from skimage import io, color
13 from scipy.spatial import distance
14
15 ###   APARTADO 1   ###
16
17 #Parametrizacion de un cono
18 def param_cone(u,v):
19     u,v = np.meshgrid(u,v)
20     x = v*np.cos(u)
21     y = v*np.sin(u)
22     z = v
23     return x.reshape(-1), y.reshape(-1), z.reshape(-1)
24
25 #Transformacion isometrica afin de un sistema de 3 dimensiones que consiste en
26 # una rotaci n entorno al centroide compuesta con una traslaci n del vector v
27 def transf_isom(x,y,z,M,C,v):
28     N = len(x)
29     xt = x*0
30     yt = y*0
31     zt = z*0
32     for i in range(N):
33         q = np.array([x[i],y[i],z[i]])
34         xt[i], yt[i], zt[i] = C + np.matmul(M, (q-C)) + v
35     return xt, yt, zt
36
37 #Grafica de la transformacion del cono para cada tiempo t
38 def animate1(t):
39     x,y,z = param_cone(np.linspace(0, 2*np.pi, 20),np.linspace(0, 5, 20))
40     theta = 3*np.pi
41     #Matriz de rotacion de angulo theta en sentido antihorario segun t
42     Mt = np.array([[np.cos(t*theta), -np.sin(t*theta), 0],
43                    [np.sin(t*theta), np.cos(t*theta),0],[0,0,1]])
44     N = len(x)
45     #Centroide del cono
46     C = np.array([sum(x)/N, sum(y)/N, sum(z)/N])
47     #Diametro mayor del cono, calculado aplicando el Teorema de Pitagoras,
48     #ya que el valor maximo de las coordenadas x,y,z es igual a 5 para cada
49     #una de ellas, entonces al cortar el cono por el eje Z, obtenemos un
50     #triangulo rectangulo en el plano XZ cuya hipotenusa es el diametro mayor d
51     d = np.sqrt(2)*5
52     vt = np.array([0,0,d])*t
53     xt, yt, zt = transf_isom(x,y,z,Mt,C,vt)
54
55     ax = plt.axes(xlim=(-12,12), ylim=(-12,12), zlim=(0,12), projection='3d')
56     #ax.view_init(60, 30)
57     ax.plot_trisurf(xt,yt,zt)
58     return ax,
59
60
61 def init1():
62     return animate1(0),
63
64 """
65 cono_t0 = plt.figure()
66 animate1(0)
67 plt.title('Transformaci n isom trica af n del cono para t=0')
68 cono_t0.savefig('cono_t=0.jpg')
```

```
cono_t1 = plt.figure()
animate1(1)
plt.title('Transformaci n isom trica af n del cono para t=1')
cono_t1.savefig('cono_t=1.jpg')


fig = plt.figure(figsize=(8,8))
ani1 = animation.FuncAnimation(fig, animate1, frames=np.arange(0,1,0.025),
                               init_func=init1, interval=20)
ani1.save("p4i.gif", fps = 10)
"""

###   APARTADO 2   ###

#Obtenemos las coordenadas de la imagen 'arbol.png'
img = io.imread('arbol.png')
fig = plt.figure()
xyz = img.shape
x = np.arange(0,xyz[0],1) #coordenadas x de cada pixel
y = np.arange(0,xyz[1],1) #coordenadas y de cada pixel
xx,yy = np.meshgrid(x, y)
xx = np.asarray(xx).reshape(-1)
yy = np.asarray(yy).reshape(-1)
z  = img[:,:,1] #componente de color verde de cada pixel
zz = np.asarray(z).reshape(-1)

#Obtenemos las coordenadas del subsistema sigma, formado por los p xeles con
#el color verde menor que 240
x0 = xx[zz<240]
y0 = yy[zz<240]
z0 = zz[zz<240]/256.

#Calculamos el di metro mayor de los pixeles del sistema sigma
l=zip(x0,y0)
dist=[]
for p in l:
    for q in l:
        dist.append(distance.euclidean(p,q))
d = max(dist)

#Grafica de la transformacion del arbol para cada tiempo t
def animate2(t):
    theta = 3*np.pi
    #Matriz de rotaci n de  ngulo  theta en sentido antihorario seg n t
    Mt = np.array([[np.cos(t*theta), -np.sin(t*theta), 0],
                   [np.sin(t*theta), np.cos(t*theta),0],[0,0,1]])
    N = len(x0)
    #Centroide del arbol: la componente Z del centroide es 0, porque la imagen
    #'arbol.png' es un sistema en 2 dimensiones, que no tiene altura
    C = np.array([sum(x0)/N, sum(y0)/N, 0])
    v = np.array([d,d,0])*t

    ax = plt.axes(xlim=(0,750), ylim=(0,400), projection='3d')
    #ax.view_init(60, 30)

    XYZ = transf_isom(x0, y0, z0, Mt, C, v)
    col = plt.get_cmap("viridis")(np.array(0.1+z0))
    ax.scatter(XYZ[0],XYZ[1],c=col,s=0.1,animated=True)
    return ax,

def init2():
    return animate2(0),

"""
arbol_t0 = plt.figure()
animate2(0)
plt.title('Transformaci n isom trica af n del  rbol  para t=0')
arbol_t0.savefig('arbol_t=0.jpg')

arbol_t1 = plt.figure()
```

```
animate2(1)
plt.title('Transformaci n  isom trica  af  n  del   rbol   para t=1')
arbol_t1.savefig('arbol_t=1.jpg')


fig = plt.figure(figsize=(8,8))
ani2 = animation.FuncAnimation(fig, animate2, frames=np.arange(0,1,0.025),
                               init_func=init2, interval=20)

ani2.save("p4ii.gif", fps = 10)
"""
```