Deformación continua de la esfera

Enrique Carro Garrido

I. Introducción (motivación/objetivo de la práctica)

Un concepto fundamental en geometría diferencial son los homeomorfismos, deformaciones continuas incluyendo el estiramiento-contracción o distorsiones tales como pliegues y retorcimientos manteniendo siempre las propiedades topológicas de esquinas y agujeros. Dicho de otro modo, un homeomorfismo es un isomorfismo entre espacios topológicos.

La computación juega un papel muy importante para entender mejor cómo evolucionan geométricamente estas deformaciones continuas. En esta práctica se pretende utilizar herramientas de visualización para poder ver cómo se comporta la proyección estereográfica; ejemplo fundamental de deformación continua de la esfera sin el polo sur al plano.

II. Material usado (método y datos)

Consideramos $S_2^1 \setminus e_3 \subset \mathbb{R}^3$ como la variedad diferencial dada por una 2-esfera de radio unitario, de la cual se extrae el punto $e_3 := (0, 0, -1) \in S_1^2$.

Se utilizan las coordenadas cartesianas $(x, y, z)^T \in [-1, 1]$ para representar los espacios topológicos. Por ello, para poder definir la proyección $\Pi : S_1^2 \subset \mathbb{R}^3 \longrightarrow \mathbb{R}^2$ mediante la composición de las funciones tan y $\tan^{-1}$ se utiliza el cambio de coordenadas cartesianas a esféricas, ϕ y el cambio de coordenadas polares a cartesianas, γ.

El cambio de coordenadas cartesianas a esféricas viene dada por la siguiente expresión.

$$\begin{gathered}\phi : S_2^1 \longrightarrow (0, \pi) \times [0, 2\pi) \\ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \theta \\ \varphi \end{pmatrix}\end{gathered}$$

definida por las relaciones

$$\theta = \begin{cases} \tan^{-1}\left(\frac{\sqrt{x^2+y^2}}{z}\right) & z > 0 \\ \frac{\pi}{2} & z = 0 \\ \pi + \tan^{-1}\left(\frac{\sqrt{x^2+y^2}}{z}\right) & z < 0 \end{cases} \qquad \varphi = \begin{cases} \tan^{-1}\left(\frac{y}{x}\right) & x > 0,\ y > 0 \\ 2\pi + \tan^{-1}\left(\frac{y}{x}\right) & x > 0,\ y < 0 \\ \frac{\pi}{2} sgn(y) & x = 0 \\ \pi + \tan^{-1}\left(\frac{y}{x}\right) & x < 0 \\ undefined & x = 0,\ y = 0 \end{cases}$$

y que se calculan numéricamente teniendo en cuenta todas las distinciones mediante la función `arctan2` de la librería `numpy`. Este cambio viene reflejado en la Figura 1. Nótese que el ángulo θ se define a partir del polo norte, y como la proyección estereográfica la hacemos de forma que es el polo sur quien se envía al infinito tomamos $\theta' = \pi - \theta$ como el nuevo ángulo y $\phi'(x, y, z) = (\theta', \varphi)$ el nuevo cambio de coordenadas.

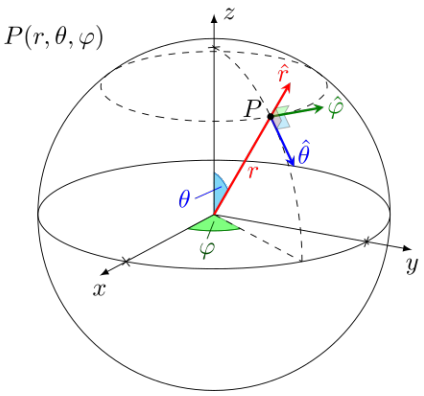


Figura 1: Coordenadas esféricas

Por otro lado, el cambio de coordenadas polares a cartesianas viene definido por la siguiente aplicación.

$$\begin{gathered}\gamma : \mathbb{R}_0^+ \times [0, 2\pi] \longrightarrow \mathbb{R}^2 \\ \begin{pmatrix} \rho \\ \omega \end{pmatrix} \mapsto \rho \cdot \begin{pmatrix} \sin(\omega) \\ \cos(\omega) \end{pmatrix}\end{gathered}$$

La proyección Π se podría expresar en coordenadas esféricas y polares de la siguiente forma

$$\gamma^{-1} \circ \Pi \circ \phi : \begin{pmatrix} \theta' \\ \varphi \end{pmatrix} \mapsto \begin{pmatrix} \frac{1}{\tan\left(\frac{\theta'}{2}\right)} \\ \varphi \end{pmatrix}$$

pero con los anteriores cambios ya estaríamos en condiciones de expresar la proyección Π en coordenadas cartesianas.

$$\Pi \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{\tan\left(\frac{\theta'}{2}\right)} \begin{pmatrix} \sin(\varphi) \\ \cos(\varphi) \end{pmatrix} = \frac{1}{\tan\left(\frac{arctan2\left(\frac{\sqrt{x^2+y^2}}{z}\right)}{2}\right)} \begin{pmatrix} \sin(arctan2\left(\frac{y}{x+\epsilon}\right) \\ \cos(arctan2\left(\frac{y}{x+\epsilon}\right) \end{pmatrix}$$

que como se puede ver es una composición de tan y $\tan^{-1}$. Nótese que el épsilon se pone para evitar que se de $x = y = 0$, ya que en este caso $arctan2$ no está definido.

Finalmente, nos queda diseñar la familia paramétrica $f_t : S_1^2 \setminus e_3 \longrightarrow \mathbb{R}^3$ que para $t = 0$ sea la identidad y para $t = 1$, $g_1 \simeq \Pi$. Para ello, se coge el haz de curvas paramétricas que recorren el segmento que une cada punto de la esfera con su correspondiente proyección. De esta forma, nos queda

$$f_t : S_1^2 \setminus e_3 \longrightarrow \mathbb{R}^3 \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ z \end{pmatrix} \cdot (1-t) + \frac{1}{\tan\left(\frac{arctan2\left(\frac{\sqrt{x^2+y^2}}{z}\right)}{2}\right)} \begin{pmatrix} \sin(arctan2\left(\frac{y}{x+\epsilon}\right) \\ \cos(arctan2\left(\frac{y}{x+\epsilon}\right) \\ 0 \end{pmatrix} t, \quad t \in [0,1]$$

que satisface todas las condiciones mencionadas. Para los cálculos matemáticos y uso de funciones matemáticas se utiliza la librería `numpy` y para las animaciones se utiliza la librería `matplotlib`.

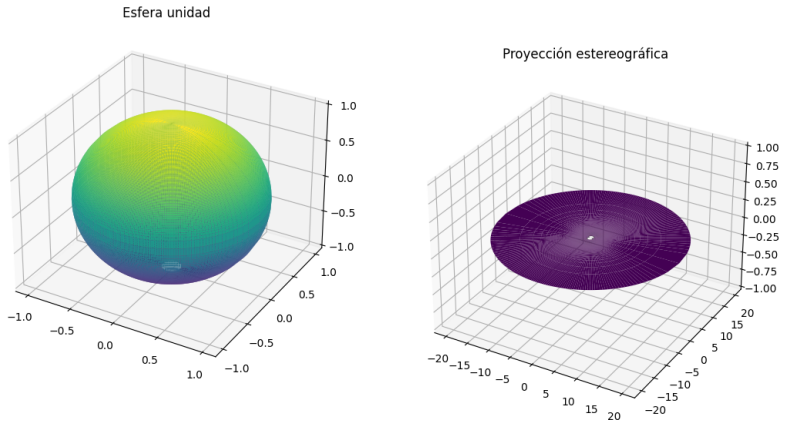
Los datos utilizados en la práctica son:

- *Matrices de coordenadas cartesianas de* S_1^2, a partir de esta discretización de la esfera unidad se representa la deformación continua.
- Se ha tomado referencia de la plantilla de animaciones "*GCOMP-practica_animacion_esfera*"para el proceso de obtención de los datos y posterior animación.

III. Resultados

La esfera unidad se ha definido a partir de sus paralelos, es decir, dado un conjunto $\mathcal{R} := 0, h, 2h, ..., 1$ de radios, tal que $|\mathcal{R}| = R$, se obtiene un conjunto de tres matrices de coordenadas cartesianas, $\{X, Y, Z\} \subset \mathcal{M}_{N\times R}$ donde la fila i es un paralelo de radio i expresado en coordenadas cartesianas.

El resultado de la aplicar la proyección estereográfica se puede ver en la siguiente figura.



(a) (b)

Se han obtenido dos animaciones de la deformación, una que en la que se ve toda la región del plano en la que se ha estirado la esfera - *Proyección estereográfica.gif* - y otra en la que se ve el estiramiento en la misma escala - *Proyección estereográfica zoom.gif*.

IV. Conclusión

La geometría computacional nos permite trabajar con deformaciones continuas de espacios topológicos muy complejos, en esta práctica hemos podido ver como se va estirando la esfera en un plano, pero se podrían haber cogido otra multitud de deformaciones continuas.

V. Anexo con el código utilizado

```
    import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation

# En primer lugar definimos la función que obtiene las coordenadas cartesianas
# de la esfera unidad menos el polo sur.
def paralelos(R, N):
  """
  Función que define 2*N puntos de cada circunferencia de radios r \in R
  de la esfera unidad.
  """
  X, Y, Z = np.empty(shape=(2*len(R) - 2, 2*N)), np.empty(shape=(2*len(R) - 2, 2*N)), np.empty(shape=(2*len(R
  for i in range(len(R)):
    # Coordenada X
    X[i,:N] = np.linspace(-np.sqrt(R[i]), np.sqrt(R[i]), N)
    X[i,N:] = -X[i,:N]
    if 0 < i < len(R) - 1: X[2*(len(R) - 1) - i,:] = X[i,:]
    # Coordenada Y
    Y[i,:N] = np.sqrt(np.abs(R[i] - X[i,:N]**2))
    Y[i,N:] = -Y[i,:N]
    if 0 < i < len(R) - 1: Y[2*(len(R) - 1) - i,:] = Y[i,:]
    # Coordenada Z
    Z[i,:] = np.sqrt(np.abs(1. - X[i,:]**2 - Y[i,:]**2))
    if 0 < i < len(R) - 1: Z[2*(len(R) - 1) - i,:] = -Z[i, :]
  return X, Y, Z

# Obtenemos las coordenadas cartesianas
R = np.linspace(0., 1., 100)
N = 100
X, Y, Z = paralelos(R, N)

fig=plt.figure(figsize=(6,6))
ax = plt.axes(projection='3d')

ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
                cmap='viridis', edgecolor='none')

ax.set_xticks(np.linspace(-1.0, 1.0, 5))
ax.set_yticks(np.linspace(-1.0, 1.0, 5))
ax.set_zticks(np.linspace(-1.0, 1.0, 5))

plt.title('Esfera unidad')
plt.savefig("Esfera unidad.png")
plt.show()

# Definimos la familia paramétrica de funciones
def ft(t, x, y, z):
  # Para que se pueda calcular siempre el ángulo phi
  eps = 1e-16
  # Cambiar cartesianas a esféricas
  theta = np.pi - np.arctan2(np.sqrt(x**2 + y**2), z)
  phi = np.arctan2(y,x + eps)
  # Deformamos la esfera
  xt = x*(1 - t) + np.cos(phi)*t / np.tan(theta/2)
  yt = y*(1 - t) + np.sin(phi)*t / np.tan(theta/2)
  zt = z*(1 - t)
  return xt, yt, zt

Xt, Yt, Zt = ft(0.5, X, Y, Z)
```

```
# Podemos ver el resultado de la proyección estereográfica
fig  = plt.figure(figsize=(6,6))
ax = plt.axes(projection='3d')

X1, Y1, Z1 = ft(1, X, Y, Z)

ax.plot_surface(X1, Y1, Z1, rstride=1, cstride=1, cmap='viridis', edgecolor='none')
ax.set_zlim(-1., 1.)

plt.title('Proyección estereográfica')
plt.savefig("Proyección estereográfica.png")

# Representamos 15 fotogramas de la proyección estereográfica, haciendo zoom o sin hacerlo

# Haciendo zoom
# Creamos la figura sobre la que dibujaremos la animación
fig = plt.figure(figsize=(6,6))
ax = plt.axes(projection='3d')

# Creamos la animación
def animate(t):
    # Obtenemos la imagen de la transformación afín en el instante t
    Xt, Yt, Zt = ft(t, X, Y, Z)
    # Borramos lo que había antes en Axes
    ax.clear()
    # Determinamos los límites del plot
    ax.set_xlim(-1., 1.)
    ax.set_ylim(-1., 1.)
    ax.set_zlim(-1., 1.)
    # Ploteamos el resultado
    ax.plot_surface(Xt, Yt, Zt, rstride=1, cstride=1, cmap='viridis', alpha=0.5, edgecolor='none')
    # Actualizamos el contador del tiempo
    ax.text(0.2,0.2,0.0, "t = {:,.3f}".format(t), transform=ax.transAxes, ha="right",
            bbox={'facecolor':'w', 'alpha':0.5, 'pad':5})
    return ax,

def init():
  return animate(0),

ani = animation.FuncAnimation(fig, animate, frames=np.arange(0,1 + 1/15,1/15), init_func=init)
ani.save("Proyección estereográfica zoom.gif", fps = 5)

# Sin hacerlo
# Creamos la figura sobre la que dibujaremos la animación
fig = plt.figure(figsize=(6,6))
ax = plt.axes(projection='3d')

# Creamos la animación
def animate(t):
    # Obtenemos la imagen de la transformación afín en el instante t
    Xt, Yt, Zt = ft(t, X, Y, Z)
    # Borramos lo que había antes en Axes
    ax.clear()
    # Determinamos los límites del plot
    ax.set_zlim(-1., 1.)
    # Ploteamos el resultado
    ax.plot_surface(Xt, Yt, Zt, rstride=1, cstride=1, cmap='viridis', alpha=0.5, edgecolor='none')
    # Actualizamos el contador del tiempo
    ax.text(0.2,0.2,0.0, "t = {:,.3f}".format(t), transform=ax.transAxes, ha="right",
            bbox={'facecolor':'w', 'alpha':0.5, 'pad':5})
    return ax,

def init():
  return animate(0),
```

```
ani = animation.FuncAnimation(fig, animate, frames=np.arange(0,1 + 1/15,1/15), init_func=init)
ani.save("Proyección estereográfica.gif", fps = 5)
```