3.Praktika: 3D Biraketak

Praktika hau burutzeko, aurreko praktikan egindako transbordadorea erabiliko dugu. Honi bi biraketa mota aplikatuko dizkiogu, bata Rodrigues formularekin eta beste koaternoien bitartez.

Jadanik zenbait funtzio ematen dizkigute eginda:

- rodrigues() : Rodrigues-en formularekin biraketa matrizea egiteko.
- quaternion(): Koaternoia sortzeko funtzioa da.
- quat_dot_quat(): Bi koaternoi biderkatzen dituen formula da.
- quat_dot_curve(): Kuaternoi bat eta 3D kurba bat biderkatzeko balio du.
- SLInterp(): Interpolazioa egiteko funtzioa da.

Honekin batera, diskretizatu() deitzen den funtzio berri bat sortu dut, honek, denbora n alditan diskretizatzeko balioko du.

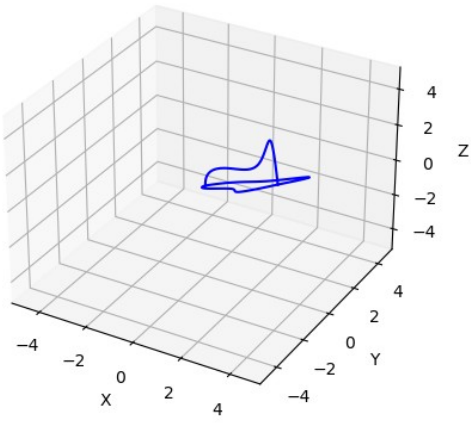
Rodrigues-ekin egindako biraketa:

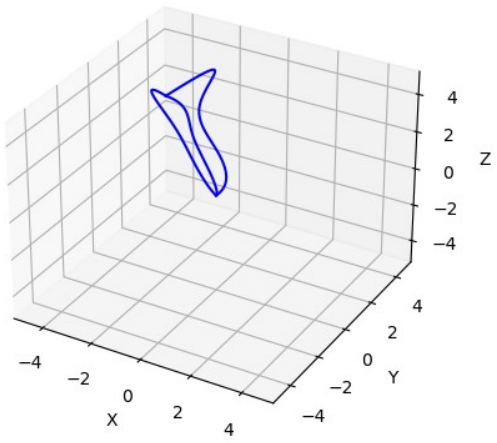
Lehenik eta behin, azken praktikaren transbordadorearen puntuekin bi irudi sortu dira, bakoitza angelu eta norabide batekin. Irudia ondo irudikatzeko, spline bakoitza Rodrigues formularekin biraketa matrizea egin behar da. Ondoren, aurreko praktikan bezala, B-Splineak eta kurbakl atera beharko ditugu. Hau guztia bi aldiz egin beharko dugu, bi irudi baitira. Horretarako hurrengo kodea erabili dezakegu:

```
#Rodrigues aplikatu
theta = pi/4
w1 = [-1, -1, 0]

puntuak_R1 = np.dot(bir.rodrigues(theta,w1), puntuak1)
puntuak_R2 = np.dot(bir.rodrigues(theta,w1), puntuak2)
puntuak_R3 = np.dot(bir.rodrigues(theta,w1), puntuak3)

basis_splinea_PR1, t_PR1 = splprep(puntuak_R1, k=3, s=0)
basis_splinea_PR2, t_PR2 = splprep(puntuak_R2, k=3, s=0)
basis_splinea_PR3, t_PR3 = splprep(puntuak_R3, k=3, s=0)

bs_kurba_R1_1 = splev(t_kurba, basis_splinea_PR1)
bs_kurba_R2_1 = splev(t_kurba, basis_splinea_PR2)
bs_kurba_R3_1 = splev(t_kurba, basis_splinea_PR3)

fig = plt.figure()
ax = fig.add_subplot(projection = '3d')

ax.plot(bs_kurba_R1_1[0], bs_kurba_R1_1[1], bs_kurba_R1_1[2], 'b-')
ax.plot(bs_kurba_R2_1[0], bs_kurba_R2_1[1], bs_kurba_R2_1[2], 'b-')
ax.plot(bs_kurba_R3_1[0], bs_kurba_R3_1[1], bs_kurba_R3_1[2], 'b-')

ax.set_xlim(-5, 5)
ax.set_ylim(-5, 5)
ax.set_zlim(-5, 5)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```

```
#Rodrigues bigarren aldiz aplikatu
theta2 = pi/2
w2 = [0, 1, 1]

puntuak2_R1 = np.dot(bir.rodrigues(theta2,w2), puntuak1)
puntuak2_R2 = np.dot(bir.rodrigues(theta2,w2), puntuak2)
puntuak2_R3 = np.dot(bir.rodrigues(theta2,w2), puntuak3)

basis_splinea2_PR1, t2_PR1 = splprep(puntuak2_R1, k=3, s=0)
basis_splinea2_PR2, t2_PR2 = splprep(puntuak2_R2, k=3, s=0)
basis_splinea2_PR3, t2_PR3 = splprep(puntuak2_R3, k=3, s=0)

bs_kurba2_R1 = splev(t_kurba, basis_splinea_PR1)
bs_kurba2_R2 = splev(t_kurba, basis_splinea_PR2)
bs_kurba2_R3 = splev(t_kurba, basis_splinea_PR3)

fig = plt.figure()
ax = fig.add_subplot(projection = '3d')

ax.plot(bs_kurba2_R1[0], bs_kurba2_R1[1], bs_kurba2_R1[2], 'b-')
ax.plot(bs_kurba2_R2[0], bs_kurba2_R2[1], bs_kurba2_R2[2], 'b-')
ax.plot(bs_kurba2_R3[0], bs_kurba2_R3[1], bs_kurba2_R3[2], 'b-')

ax.set_xlim(-5, 5)
ax.set_ylim(-5, 5)
ax.set_zlim(-5, 5)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```

Honekin hurrengo bi irudiak lortuko ditugu:





Kuaternoien bidez egindako biraketa:

Zati honekin hasteko, lehenik eta behin erabiliko ditugun bi kuaternoiak definituko ditugu, hurrengoa erabiliz:

```
#Koaternoiak aplikatu

p = bir.quaternion(theta, w1)
q = bir.quaternion(theta2, w2)
```

Ondoren, t diskretizatu egin behar da. 50 alditan egitea gomendagarria da, modu honetan gero sortuko den animazioa ondo ikusiko da. Hau egiteko hurrengo kodea erabiltzen da:

```
s = []
n = 50
t = diskretizatu(n)

for i in range(len(t)):
    s.append(bir.SLInterp(p,q,t[i]))
```

Gero, aurreko atalaren antzera, puntuak, splineak eta kurbak atera beharko ditugu. Kodea nahiko antzekoa da, dena den, koaternoiak erabiltzen hari garenez, hasieran aipatutako bi biderkadura funtzioak erabili beharko ditugu. Hemendik sortuko ditugun irudiak, *figures* deitutako karpeta batean gordeko ditugu:

```
for i in range(len(s)):
    p1quater = bir.quat_dot_curve(s[i], puntuak1)
    p2quater = bir.quat_dot_curve(s[i], puntuak2)
    p3quater = bir.quat_dot_curve(s[i], puntuak3)

    basis_splinea1_quat, t_puntuak1_quater = splprep(p1quater, k=3, s=0)
    basis_splinea2_quat, t_puntuak2_quater = splprep(p2quater, k=3, s=0)
    basis_splinea3_quat, t_puntuak3_quater = splprep(p3quater, k=3, s=0)

    bs_kurba1_quater = splev(t_kurba, basis_splinea1_quat)
    bs_kurba2_quater = splev(t_kurba, basis_splinea2_quat)
    bs_kurba3_quater = splev(t_kurba, basis_splinea3_quat)

    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')

    ax.plot(bs_kurba1_quater[0], bs_kurba1_quater[1], bs_kurba1_quater[2], 'b-')
    ax.plot(bs_kurba2_quater[0], bs_kurba2_quater[1], bs_kurba2_quater[2], 'b-')
    ax.plot(bs_kurba3_quater[0], bs_kurba3_quater[1], bs_kurba3_quater[2], 'b-')

    ax.set_xlim(-5, 5)
    ax.set_ylim(-5, 5)
    ax.set_zlim(-5, 5)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    plt.savefig("./figures/"+str(i+1))
    plt.close()
```

Azkenik, irudiak izanda, bi alditan ordenatuko ditugu, bata alfabetikoki, eta bestea kontrako ordena, modu honetan, animazioa aurrera eta atzeraka joango da:

```
#Irudiak karpetatik atera
file_list = glob.glob('./figures/*.png')

#Alfabetikoki ordenatu
seq = natsort.natsorted(file_list)

#Kontrako ordenean
seq_ald = natsort.natsorted(file_list, reverse = True)

#Sekuentzia sortu
clip = mpy.ImageSequenceClip(seq+seq_ald, fps=24)
clip.write_gif('animazioa.gif')
```

Sortutako animazioa, gif bat bezala gordetzen da karpetan bertan.