SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Planiranje trajektorije

Osnove robotike

Laboratorijska vježba 3

Ivan Gudelj

Diplomski studij računarstva, DRB

Osijek,2022.

I. Cilj vježbe

Generirati trajektoriju robotskog manipulatora razmatranog u prvoj vježbi koja omogućuje robotu iscrtavanje zadanog znaka (početno slovo imena ili prezimena) na ploči postavljenoj u njegovom radnom prostoru.

II. Rad na vježbi

1. Primjenom funkcije hocook generirati trajektoriju robotskog manipulatora, razmatranog u prvoj vježbi, koja omogućuje robotu iscrtavanje zadanog znaka na ploči postavljenoj u njegovom radnom prostoru. Prvo zadati početnu i završnu točku putanje te nekoliko spojnih točaka trajektorije. Dodavati spojne točke dok se ne dobije željeni rezultat.

2. Rezultat prikazati funkcijom planecontact. Ova funkcija za putanju vrha alata zadanu matricom W, normalu ravnine n i udaljenost ravnine od ishodišta d prikazuje one točke putanje čija je udaljenost od zadane ravnine < 5 mm. Prikazati dobivenu trajektoriju u 3D pomoću funkcije plot.

3. Prikazati odzive svih varijabli zglobova tijekom izvođenja trajektorije te njihove prve derivacije (brzine) i druge derivacije (ubrzanja). Komentirati dobivene rezultate s obzirom na zahtjeve o neprekinutosti trajektorije te njezine prve i druge derivacije te s obzirom na zadana ograničenja brzina odnosno ubrzanja.

III. Rješenje

Zbog pogreške u softverskom rješenju Laboratorijske vježbe 2 (nepotpuna inverzna kinematika) za vlastiti primjer, korišten je primjer dan od strane voditelja Laboratorijske vježbe te je na njemu odrađena Laboratorijska vježba 3.

Prvo je bilo potrebno prepoznati ravninu na kojoj robotska ruka "piše" odnosno vrši svoju trajektoriju. Dimenzije ravnine su 400 x 400 mm (u kodu 0.0 - 0.4 x -0.2 - 0.2).

Zadatak je bio predati robotu točke kojima se iscrtava prvo slovo prezimena (G). Kod za dodavanje točki nalazi se na slici ispod;

```
# Robot velocity and acceleration limits.
    dqgr=np.pi*np.ones((1,6))
    ddqgr=10*np.pi*np.ones((1,6))

    # Trajectory.
    q_home = np.array([-np.pi/2, np.pi/2, 0, 0, 0, 0])

    T60_1 = np.identity(4)
    T60_1[:3,:3] = roty(np.pi)
    T60_1[:3,3] = np.array([0.40, 0.2, 0.03])
    q1 = invkin(rob.DH,T60_1,[1, 0, 0])

    T60_2 = T60_1.copy()
    T60_2[:3,3] = np.array([0.40, 0.2, 0.02])
    q2 = invkin(rob.DH,T60_2,[1, 0, 0])

    T60_3 = T60_1.copy()
    T60_3[:3,3] = np.array([0.20, 0.2, 0.02])
    q3 = invkin(rob.DH,T60_3,[1, 0, 0])

    T60_4 = T60_1.copy()
    T60_4[:3,3] = np.array([0.20, -0.2, 0.02])
    q4 = invkin(rob.DH,T60_4,[1, 0, 0])

    T60_5 = T60_1.copy()
    T60_5[:3,3] = np.array([0.40, -0.2, 0.02])
    q5 = invkin(rob.DH,T60_5,[1, 0, 0])

    T60_6 = T60_1.copy()
    T60_6[:3,3] = np.array([0.40, -0.05, 0.02])
    q6 = invkin(rob.DH,T60_6,[1, 0, 0])
```

```
T60_7 = T60_1.copy()
T60_7[:3,3] = np.array([0.30, -0.05, 0.02])
q7 = invkin(rob.DH,T60_7,[1, 0, 0])


Q = np.stack((q_home,q1, q2,q3,q4,q5,q6,q7, q_home), 1)
Ts = 0.02
Qc, dQc, ddQc, tc = hocook(Q, dqgr, ddqgr, Ts)
```

Rješenje prikazuje postupak u kojem korisnik predaje koordinate točke do koje robot treba doći. Kako bi se to postiglo, potrebno je obaviti inverznu kinematiku robotske ruke kako bismo dobili parametre (kuteve zglobova robotske ruke). Navedeno se izvršava pomoću funkcije *invkin*() koju smo implementirali na prethodnoj Laboratorijskoj vježbi.

Nakon dodavanja svih točki i izračuna parametara q1-q7 (+q_home koji predstavlja početnu točku), koristimo ugrađenu funkciju *numpy* biblioteke *numpy.stack()* kojom se proširuje matrica po zadanoj osi. Ravnina u kojoj prepoznajemo i ispisujemo uzorkovane točke (ravnina u kojoj bilježimo točke robota) se određuje prema sljedećem kodu;

```
# Display plane contact.
    n_board = np.array([0, 0, 1])
    d_board = 0.02
    board_draw = planecontact(tool_tip_W, n_board, d_board)
    fig, ax = plt.subplots(1, 1)
    ax.plot(board_draw[:,0], board_draw[:,1], 'b.')
    ax.axis('equal')
    plt.show()
```

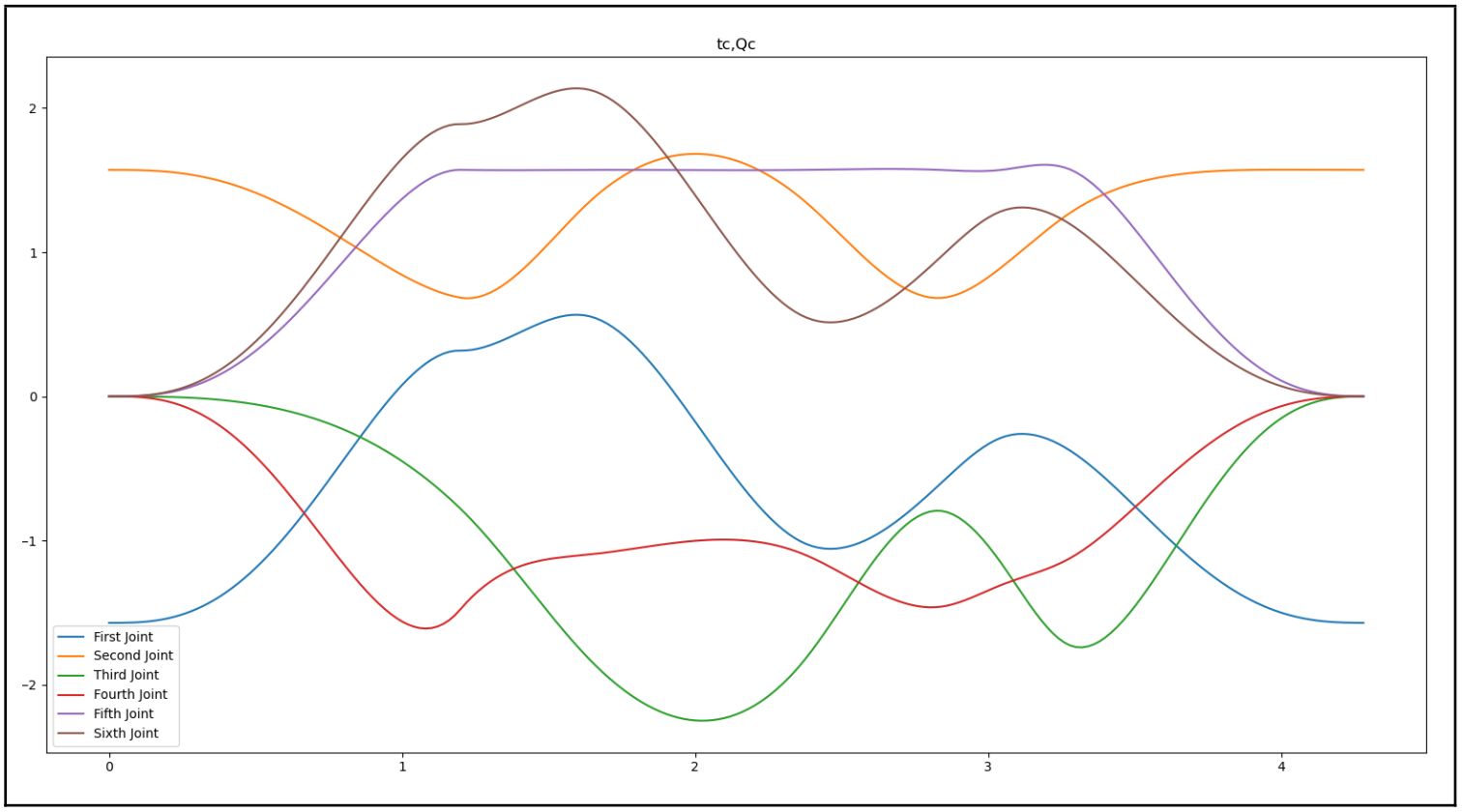
Vrijeme uzorkovanja (Ts) označava vrijeme svakih koliko će se uzeti uzorak pozicije u kojoj se robot nalazi tijekom trajektorije. Povećanjem vremena uzorkovanja dobijamo lošiji rezultat odnosno manji broj točaka na ravnini crtanja. Također limiti brzine i ubrzanja robota definiraju se sa sljedećim linijama koda

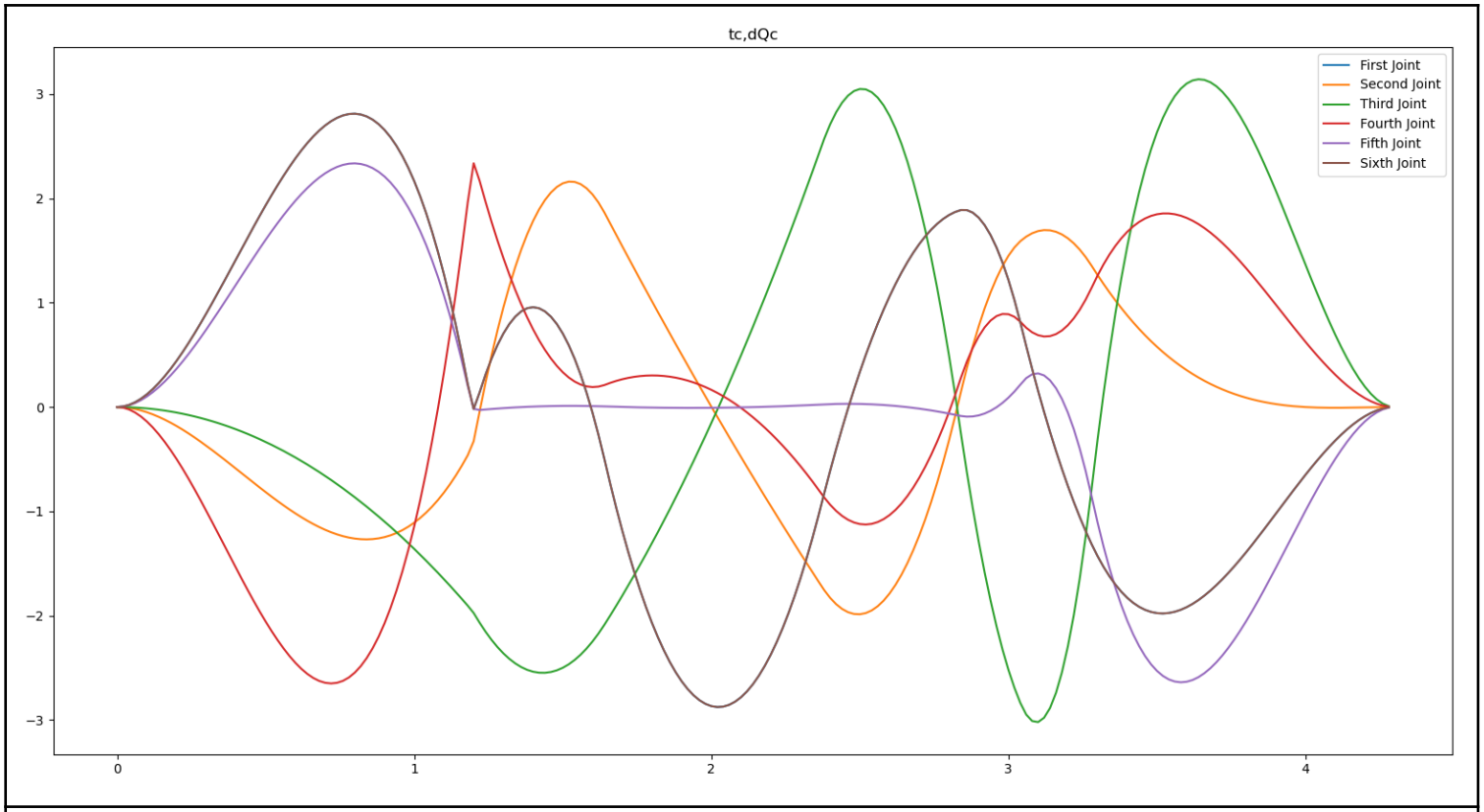
```
    # Robot velocity and acceleration limits.
    dqgr=np.pi*np.ones((1,6))
    ddqgr=10*np.pi*np.ones((1,6))
```

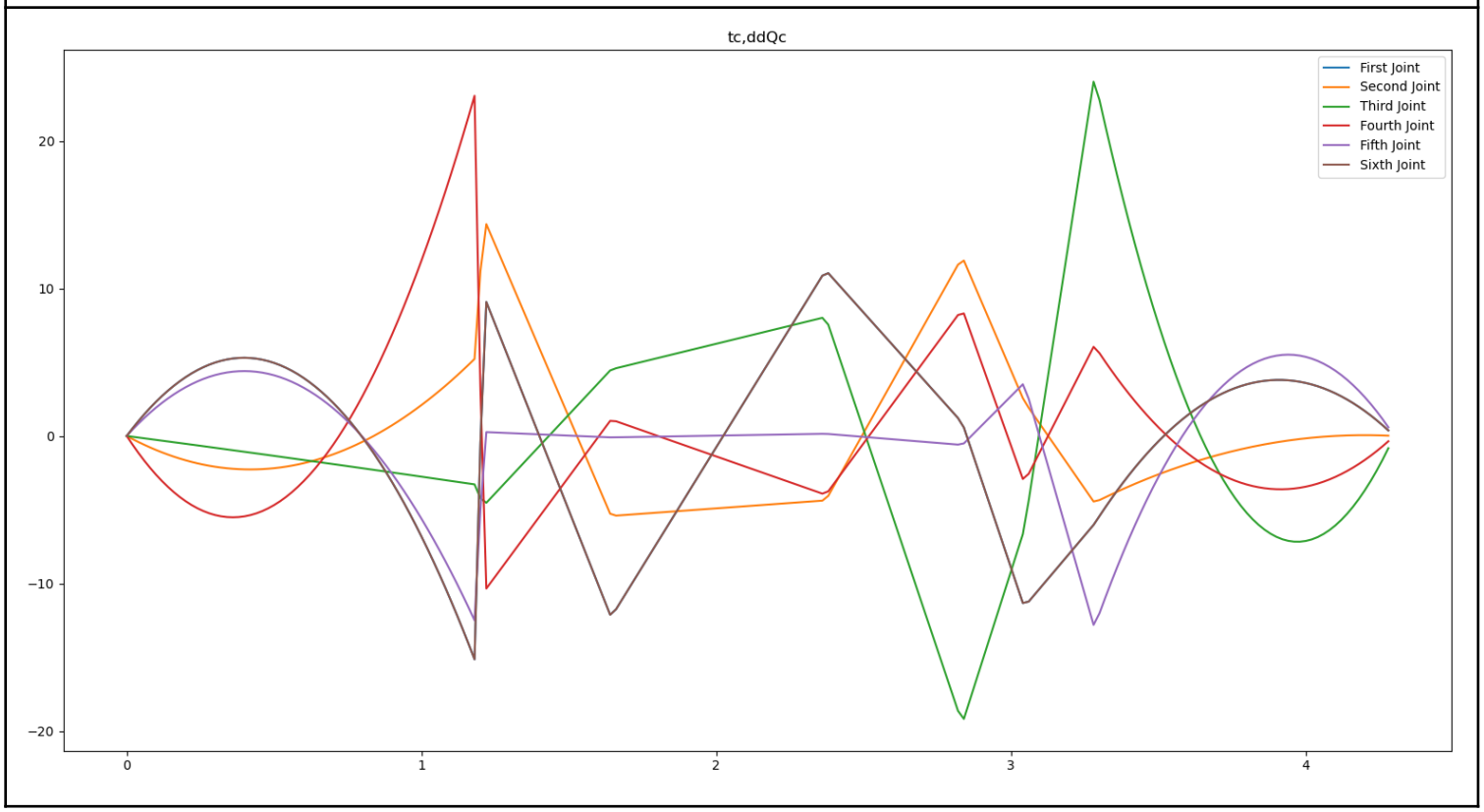
Tolerancija(*Threshold*) u kojoj mi bilježimo(uzorkujemo) točke trajektorije po kojoj se robotska ruka kreće se podešava u datoteci *planescontact* sljedećim linijama koda gdje 0.075(nije 0.05 jer u tom slučaju iz nekog razloga ne zabilježava točke) predstavlja prostor tolerancije od 7.5 mm oko ravnine koju smo definirali kao ploha za crtanje;

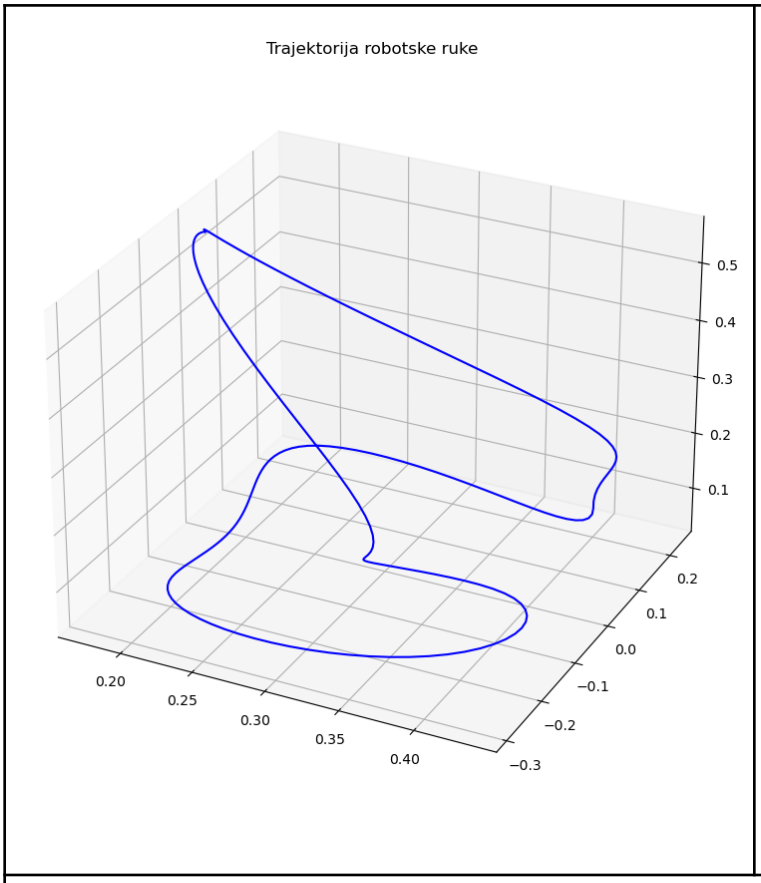
```
def planecontact(W, n, d):
    e = (n * W).sum(1) - d
    return W[np.abs(e) < 0.075,:2]
```

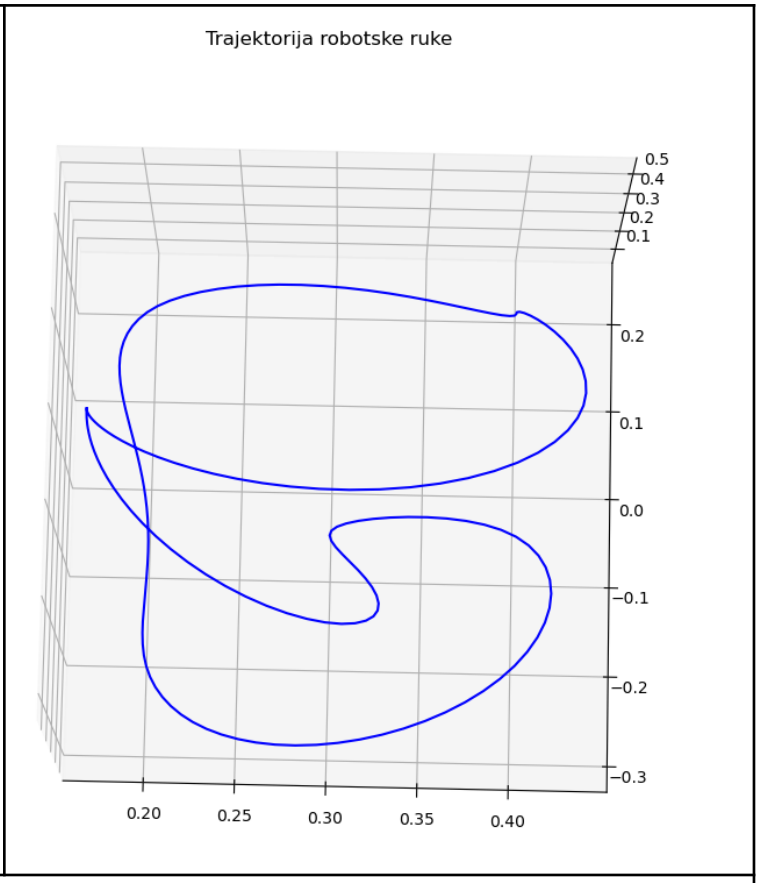
Rezultati pokretanja pripremljene skripte sa zadanim točkama nalaze se na slikama ispod;

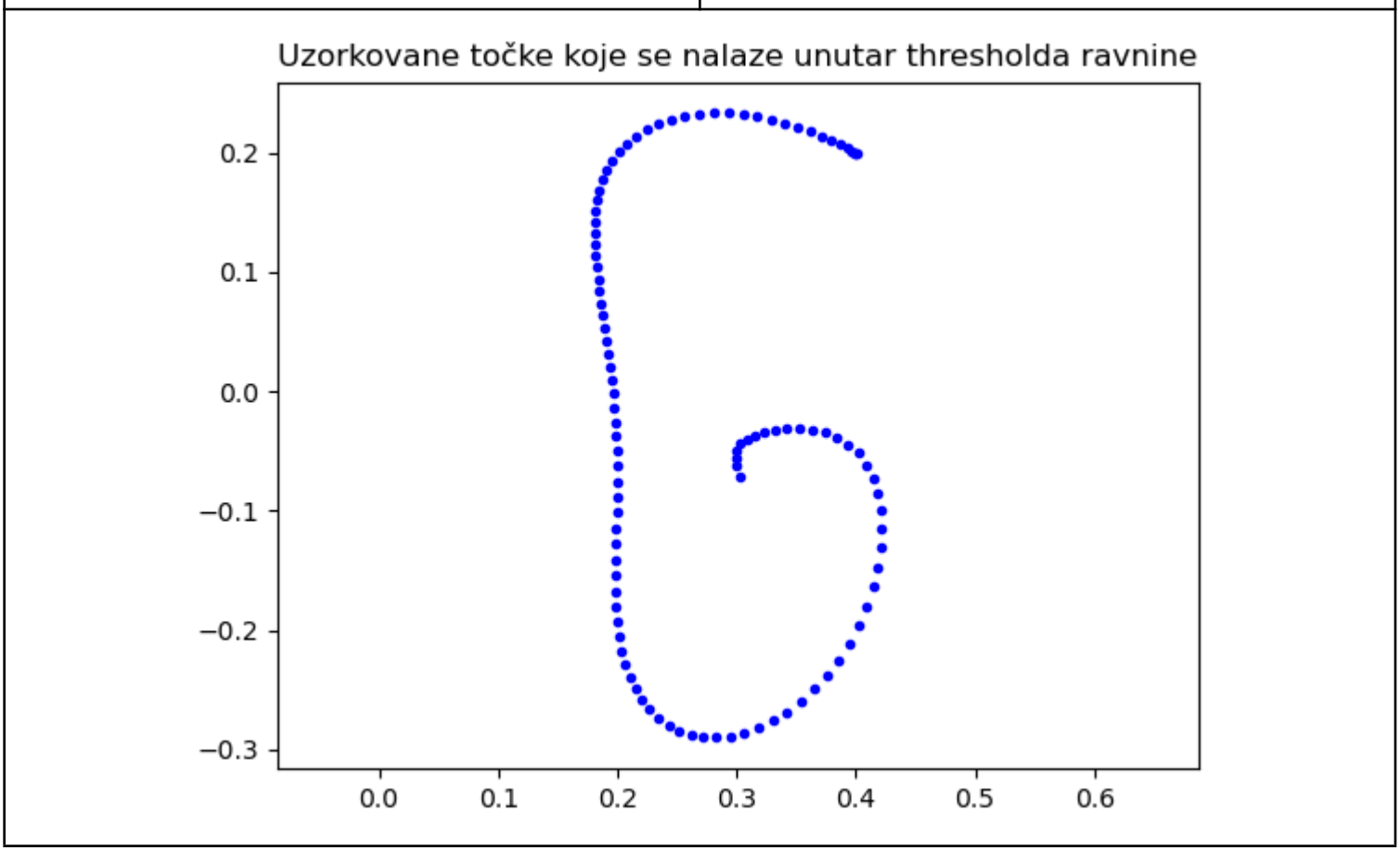



tc,dQc
First Joint
Second Joint
Third Joint
Fourth Joint
Fifth Joint
Sixth Joint
3
2
1
0
−1
−2
−3
0
1
2
3
4


tc,ddQc
First Joint
Second Joint
Third Joint
Fourth Joint
Fifth Joint
Sixth Joint
20
10
0
−10
−20
0
1
2
3
4


Trajektorija robotske ruke
0.20
0.25
0.30
0.35
0.40
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.1
0.2
0.3
0.4
0.5


Trajektorija robotske ruke
0.20
0.25
0.30
0.35
0.40
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.1
0.2
0.3
0.4
0.5


Uzorkovane točke koje se nalaze unutar thresholda ravnine
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−0.3
−0.2
−0.1
0.0
0.1
0.2

Kao posljednju stavku bilo je potrebno analizirati promjenu ograničenja brzine i ubrzanja robotske ruke. **Prepolovljavanjem brzine i ubrzanja** dobijamo puno veći broj točaka za isto vrijeme uzorkovanja (robot napravi manji pomak za 0.02s), te samim time dobijamo jasniju (čistiju) sliku koja se nalazi ispod. Također vrijeme potrebno da se trajektorija izvrši se mijenja sa 4.5s na približno 8.5s