Zadaća 1

Inteligentni robotski sustavi

Rok predaje: 10. ožujka u 23:59

Način predaje: Na gitlab stvoriti repozitorij "irs2022" te dodati korisnika jmaltar s ulogom "developer". Rješenja zadataka smjestiti u mapu `irs2022/dz01`.

Komentari: Radi lakšeg snalaženja, osi gibajućeg koordinatnog sustava možete obojati različitim bojama, npr. `colors=["r", "g", "b"]`.

Zad. 1 (5 bodova)

Po uzoru na implementaciju s vježbi, implementirajte gibanje gdje se za orijentaciju koriste Eulerovi kutevi. Neka se orijentacija i pozicija, odnosno konfiguracija/poza/stanje, okvira j i okvira i spremaju u homogene transformacijske matrice ${}^{j}T_{j}$ i ${}^{j}T_{i}$, a ne u matrice rotacija i vektore pozicija $\left({}^{j}R_{j}, {}^{j}p_{j}\right)$ i $\left({}^{j}R_{i}, {}^{j}p_{i}\right)$ *(2 boda)*. Na odgovarajući način modificirajte vektore okvira *(1 bod)* i na odgovarajući način vršite transformaciju s homogenom transformacijskom matricom *(1 bod)*. Koristite brzine ${}^{i}\omega_{i}$ i ${}^{j}\dot{p}_{i}$ definirane s:

```python
def fake_algorithm_transl_vel(i):
    j_p_i_dot = np.array([1, 1, 1], dtype=float)
    j_p_i_dot[2] = 2 * np.sin(4 * i / 120 * 2 * np.pi)
    return j_p_i_dot

def fake_algorithm_angular_vel(i):
    if i < 60:
        i_omega_i = np.array([np.pi, 0, 0], dtype=float)
    else:
        i_omega_i = np.array([0, 2 * np.pi, 0], dtype=float)
    return i_omega_i
```

Rješenje spremite u skriptu `zad_1.py`, a gif animaciju u `zad_1.gif` *(1 bod)*.

Zad 2. (10 bodova)

Gibajte se kao u prvom zadatku. Umjesto nadogradnje orijentacije putem Eulerovih kuteva, te pozicije uobičajeno, nadograđujte konfiguraciju mapirajući os vijka $\mathcal{S}$ u novu homogenu transformaciju *(9 bodova)*. Neka je za to u Pythonu definirana funkcija:

`T_update(T, j_p_i_dot, i_omega_i, dt)`.

Rješenje spremite u skriptu `zad_2.py`, a gif animaciju u `zad_2.gif` *(1 bod)*.

Zad. 3 (5 bodova)

Nastavno na drugi zadatak, implementirajte gibanje u trajanju od tri sekunde prikazano u priloženom gifu `zad_3_rjesenje.gif` *(1 bod za prvu sekundu, 3 boda za drugu i treću)*. Definirajte odgovarajuće brzine u funkcijama `fake_algorithm_angular_vel` i `fake_algorithm_transl_vel`. Kretanje u zadnje dvije sekunde moguće je postići tako da upravljamo translacijskom brzinom gibajućeg koordinatnog sustava s obzirom na gibajući koordinatni sustav – ${}^{i}v_{i}$ – pri čemu svo vrijeme želimo ostati udaljeni s obzirom na ishodište fiksnog koordinatnog sustava jednako. Stoga, razmislite koje je argumente, uz broj iteracije `i`, potrebno proslijediti ovim funkcijama. Rješenje spremite u skriptu `zad_3.py`, a gif animaciju u `zad_3.gif` *(1 bod)*.

Zad. 4 (5 bodova)

Nadopunite treći zadatak tako da postoji još jedan gibajući okvir, okvir k, čija će se konfiguracija pohranjivati u ${}^{i}T_{k}$ koji je u početku poravnat s i. Time okvir i postaje fiksni okvir za okvir k, iako je okvir i gibajući okvir s obzirom na okvir j. Neka je u početku okvir i translacijski otklonjen od okvira j za ${}^{j}p_{i} = [1.5, 0, 0]^{T}$[1]. Neka se i s obzirom na j giba kao u priloženom gifu `zad_4_k_static.gif`. *(2 boda)* Neka se k s obzirom na svoj fiksni okvir i giba kako se u trećem zadatku gibao i s obzirom na j *(2 boda)* – gif `zad_4_i_static.gif` prikazuje kako se konfiguracija okvira k mijenja s obzirom na i kada se i ne giba).

Pri gibanju oba okvira rješenje mora izgledati kao na gifu `zad_4_rjesenje.gif`. Rješenje spremite u skriptu `zad_4.py`, a gif animaciju u `zad_4.gif` *(1 bod)*.

[1] Tada i pri instanciranju strijelica okvira treba napraviti odgovarajuću transformaciju jer je ${}^{j}T_{i}(0) \neq I$.