Propuneri proiect

opțional SPER

21 martie 2024

Cuprins

1 Elemente comune **2**

2 Propuneri proiecte **3**

2.1 Parcurgerea de puncte intermediare folosind funcții B-spline 3
2.2 Ocolirea unui obstacol folosind funcții B-spline 4
2.3 Parcurgere de puncte intermediare printr-o traiectorie Dubins . . . 5
2.4 Parcurgere de puncte intermediare printr-o traiectorie Reeds-Shepp 6
2.5 Ocolire de obstacole folosind formulări cu variabile mixte 7
2.6 Generare referințe ale forței de tracțiune/unghiuri pentru o dronă . 8
2.7 Mecanism de menținere a unei formații 9
2.8 Implementare algoritm BCD 10
2.9 Implementare algoritm RRT 11

compilat la: 21/03/2024, 09:30

1 Elemente comune

- proiectul presupune realizarea și prezentarea unei aplicații (fie din lista de mai jos, fie propusă direct de echipă);
- fiecare echipă (formată din 1-4 persoane) trebuie să încarce până în deadline o arhivă cu script-uri și o prezentare (ppt/pdf) de 15 min pe Moodle;
- prezentarea va fi susținută în fața colegilor (fiecare membru al echipei trebuie să participe la prezentare și să își explice contribuția);
- două sau mai multe echipe nu pot avea aceeași temă; în caz de suprapunere, voi varia conținutul cerințelor/propun altă temă;
- pentru unele proiecte v-am recomandat (sau puteți găsi și voi) librării Python ce deja realizează cerințele; dacă/unde folosiți aceste librării trebuie să explicați ideea din spate/să analizați comportamentul pentru diverse valori ale parametrilor ⇐ să arătați că ați înțeles, nu doar că ați rulat...;
- dificultatea proiectului este proporțională cu gradul de interes/dimensiunea echipei ⇐ e perfect posibil să estimez greșit complexitatea subiectului, dacă vi se pare prea ușor, contactați-mă :D

2 Propuneri proiecte

2.1 Parcurgerea de puncte intermediare folosind funcții B-spline

Funcțiile B-spline sunt definite recursiv de relația

$$B_{\ell,d,\zeta}(\tau) = \frac{\tau - \tau_\ell}{\tau_{\ell+d} - \tau_\ell} B_{\ell,d-1,\zeta}(\tau) + \frac{\tau_{\ell+d+1} - \tau}{\tau_{\ell+d+1} - \tau_{\ell+1}} B_{\ell+1,d-1,\zeta}(\tau), \tag{1a}$$

$$B_{\ell,0,\zeta}(\tau) = \begin{cases} 1, & \tau \in [\tau_\ell, \tau_{\ell+1}), \\ 0, & \text{altfel.} \end{cases}, \quad \ell = 1 \dots n. \tag{1b}$$

Parametrii d și $\zeta = \{\underbrace{\tau_1, \dots \tau_1}_{d+1}, \tau_2, \dots, \tau_{n-1}, \underbrace{\tau_n, \dots \tau_n}_{d+1}\}$ definesc ordinul funcției B-spline respectiv “knot vector”-ul acesteia.
Pentru modelul matematic al mașinii Dubins,

$$\begin{cases} \dot{x} &= u_V \cos\phi, \\ \dot{y} &= u_V \sin\phi, \\ \dot{\phi} &= \frac{u_V}{L} \tan u_\phi, \end{cases} \tag{2}$$

se cer următoarele:

i) Pentru o listă de puncte intermediare date, găsiți traiectoria de energie minimă ce trece prin ele. Cu alte cuvinte, găsiți punctele de control P_i ca rezultat al problemei de optimizare

$$\min_{P_i} \int_{t_i}^{t_f} \left\| \sum_{i=0}^{n} P_i \dot{B}_{i,d,\zeta}(t) \right\|^2 dt \tag{3a}$$

$$\sum_{i=0}^{n} P_i B_{i,d,\zeta}(t_j) = w_j, \forall j, \tag{3b}$$

unde traiectoria este dată de $z(t) = \sum_{i=0}^{n} P_i B_{i,d,\zeta}(t)$.

ii) Ilustrați grafic u_V și u_ϕ în funcție de $z(t)$ și derivatele sale folosind formula:

$$\begin{cases} u_V &= \sqrt{\dot{z}_1^2 + \dot{z}_2^2} \\ u_\phi &= \arctan\left(\frac{L\dot{\phi}}{u_V}\right) = \arctan\left(L \frac{\ddot{z}_2 \dot{z}_1 - \dot{z}_2 \ddot{z}_1}{\left(\dot{z}_1^2 + \dot{z}_2^2\right)^{\frac{3}{2}}}\right) \end{cases} \tag{4}$$

Indicație: funcțiile B-spline se pot calcula pe baza relației (1) sau folosind librăria Python geomdl.

Dificultate: 3-4 persoane.

2.2 Ocolirea unui obstacol folosind funcții B-spline

Funcțiile B-spline sunt definite recursiv de relația

$$B_{\ell,d,\boldsymbol{\zeta}}(\tau) = \frac{\tau - \tau_\ell}{\tau_{\ell+d} - \tau_\ell} B_{\ell,d-1,\boldsymbol{\zeta}}(\tau) + \frac{\tau_{\ell+d+1} - \tau}{\tau_{\ell+d+1} - \tau_{\ell+1}} B_{\ell+1,d-1,\boldsymbol{\zeta}}(\tau), \tag{5a}$$

$$B_{\ell,0,\boldsymbol{\zeta}}(\tau) = \begin{cases} 1, & \tau \in [\tau_\ell, \tau_{\ell+1}), \\ 0, & \text{altfel.} \end{cases}, \quad \ell = 1 \dots n. \tag{5b}$$

Parametrii d și $\zeta = \{\underbrace{\tau_1, \dots \tau_1}_{d+1}, \tau_2, \dots, \tau_{n-1}, \underbrace{\tau_n, \dots \tau_n}_{d+1}\}$ definesc ordinul funcției B-spline respectiv "knot vector"-ul acesteia.

Pentru o traiectoria definită ca

$$z(t) = \sum_{i=0}^{n} P_i B_{i,d,\boldsymbol{\zeta}}(t), \tag{6}$$

avem proprietatea că acestă traiectorie se regăsește în uniunea regiunilor definite de $d+1$ puncte de control consecutive. Mai precis, pentru un sub-interval de timp $t \in [\tau_\ell, \tau_{\ell+1})$ avem că

$$z(t) \in R\left(\{P_{\ell-d}, \dots, P_\ell\}\right). \tag{7}$$

Prin urmare, pentru a garanta că traiectoria nu intersectează un obstacol S definit prin punctele sale extreme $\{V_1, \dots, V_m\}$ este suficient să rezolvăm problema de optimizare

$$\min_{P_i, c_\ell} \int_{t_i}^{t_f} \left\| \sum_{i=0}^{n} P_i \dot{B}_{i,d,\boldsymbol{\zeta}}(t) \right\|^2 dt \tag{8a}$$

$$c_\ell^\top V_j \leq 1, \; j = 1 \dots m, \tag{8b}$$

$$c_\ell^\top P_j \geq 1, \; j = \ell - d \dots \ell. \tag{8c}$$

Cu alte cuvinte, forțăm ca pentru fiecare listă de $d+1$ puncte de control consecutive să existe hiperplanul $c_\ell^\top x = 1$ care să le separe de obstacolul S.
Rezolvați problema de optimizare și ilustrați rezultatul.

Indicație: funcțiile B-spline se pot calcula pe baza relației (1) sau folosind librăria Python geomdl.

Dificultate: 2-3 persoane.

2.3 Parcurgere de puncte intermediare printr-o traiectorie Dubins

Pentru o listă de puncte intermediare dată, construiți traiectoria formată din primitive Dubins astfel încât să minimizați lungimea traiectoriei.

Se cer următoarele:

i) Alegeți ordinea de parcurgere a punctelor intermediare astfel încât să minimizați lungimea traiectoriei (de exemplu printr-un algoritm de tipul traveling salesman problem).

ii) Ilustrați traiectoriile rezultate pentru diverse valori ale valorii de rază de întoarcere minimă.

Indicație: Pentru generarea traiectoriei pentru o listă dată de puncte puteți, spre exemplu, să folosiți codul din https://github.com/fgabbert/dubins_py

Dificultate: 3-4 persoane.

2.4 Parcurgere de puncte intermediare printr-o traiectorie Reeds-Shepp

Pentru o listă de puncte intermediare dată, construiți traiectoria formată din primitive Reeds-Shepp astfel încât să minimizați lungimea traiectoriei.

Se cer următoarele:

i) Alegeți ordinea de parcurgere a punctelor intermediare astfel încât să minimizați lungimea traiectoriei (de exemplu printr-un algoritm de tipul traveling salesman problem).

ii) Ilustrați traiectoriile rezultate pentru diverse valori ale valorii de rază de întoarcere minimă.

Indicație: Pentru generarea traiectoriei pentru o listă dată de puncte puteți, spre exemplu, să folosiți codul din `https://github.com/nathanlct/reeds-shepp-curves`

Dificultate: 3-4 persoane.

2.5 Ocolire de obstacole folosind formulări cu variabile mixte

Pentru o colecție de obstacole date în forma "half-space":

$$S_i = \{x \in \mathbb{R}^2 : \ F_i x \leq \theta_i\}, \text{ cu } (F_i, \theta_i) \in \mathbb{R}^{N_i \times 2} \times \mathbb{R}^{N_i}, \tag{9}$$

implementați o problemă de tipul MPC pentru a calcula o traiectorie ce le ocolește (folosind variabile binare pentru a forța ocolirea obstacolelor):

$$\min_{u_k \dots u_{k+N-1}} \sum_{i=1}^{N} (x_{k+i} - \bar{x})^\top Q \, (x_{k+i} - \bar{x}) + \sum_{i=1}^{N-1} (u_{k+i} - u_{k+i-1})^\top R \, (u_{k+i} - u_{k+i-1}) \tag{10a}$$

$$\text{s.t.} \quad x_{k+i+1} = A x_{k+i} + B u_{k+i}, \tag{10b}$$

$$|u_{k+i}| \leq \bar{u}, \tag{10c}$$

$$|x_{k+i+1}| \leq \bar{x}, \tag{10d}$$

$$x_{k+i+1} \notin S_j, \qquad \forall i = 1:N, \forall j, \tag{10e}$$

unde

$$A = \begin{bmatrix} I & T \cdot I \\ 0 & I \end{bmatrix}, \ B = \begin{bmatrix} \frac{T^2}{2} \cdot I \\ T \cdot I \end{bmatrix}, \ Q = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}, \ R = 0.1 \cdot I \text{ și } T = 0.1.$$

Dificultate: 3-4 persoane.

2.6 Generare referințe ale forței de tracțiune/unghiuri pentru o dronă

Considerați un profil de referință a cărei derivată de ordinul 4 ("jounce") este dată de formula

$$z^{(4)}(t) = \begin{cases} a\sin\left(\frac{\pi t}{b}\right), 0 \leq t \leq b \\ -a\sin\left(\frac{\pi t}{2b} - \frac{\pi}{2}\right), b \leq t \leq 3b \\ a\sin\left(\frac{\pi t}{b} - 3\pi\right), 3b \leq t \leq 4b. \end{cases} \tag{11}$$

Considerați o dronă controlată prin forța de tracțiune și unghiurile de roll/pitch. Aceste mărimi sunt cosntruite pe baza unui profil dat de accelerație de referință:

$$T = m\sqrt{\ddot{z}_1^2 + \ddot{z}_2^2 + (\ddot{z}_3 + g)^2}, \tag{12a}$$

$$\phi_{ref} = \arcsin\left(\frac{\ddot{z}_1 \sin z_4 - \ddot{z}_2 \cos z_4}{\sqrt{\ddot{z}_1^2 + \ddot{z}_2^2 + (\ddot{z}_3 + g)^2}}\right), \tag{12b}$$

$$\theta_{ref} = \arctan\left(\frac{\ddot{z}_1 \cos z_4 + \ddot{z}_2 \sin z_4}{\ddot{z}_3 + g}\right). \tag{12c}$$

Se cere:

i) Integrați succesiv termenul (11) pentru a obține derivatele de ordinul 3, 2, 1 și 0 ("jerk", accelerație, viteză și poziție);

ii) Aplicați profilul de accelerație $z^{(2)}(t)$ în relațiile (12) și ilustrați valorile rezultate;

iii) Pentru o listă de puncte intermediare, alegeți parametrii (a, b) astfel încât să construiți o traiectorie ce trece prin aceste puncte (fiecărui segment de două puncte intermediare consecutive îi corespunde o pereche (a, b)).

Dificultate: 3-4 persoane.

2.7 Mecanism de menținere a unei formații

Pentru o colecție de agenți definiți de poziție/viteză, considerați o formație caracterizată prin distanțe și viteze relative:

$$\begin{aligned}&\text{între agentul i și vecinii săi } j \in \mathcal{N}_i \text{ se mențin relațiile:}\\&\|p_i - p_j\| \to d_{ij}, \quad \|v_i - v_j\| \to 0.\end{aligned} \tag{13}$$

Se cere:

i) Implementați un mecanism de tip gradient ce penalizează erorile de poziție și viteză relative astfel încât agenții, plecând din poziții inițiale arbitrare să conveargă către formația dorită.

ii) Ilustrați evoluția în timp a erorilor de poziție și viteză.

Dificultate: 2-3 persoane.

2.8 Implementare algoritm BCD

Unul din algoritmii utilizați pentru acoperirea unei regiuni cu obstacole este BCD (boustrophedon cellular decomposition).

Se cere:

i) Explicați algoritmul.

ii) Implementați-l/aplicați-l pentru o suprafață cu obstacole (generați/cautați hărți de spații interioare).

Indicație: Pentru generarea traiectoriilor BCD puteți spre exemplu folosi `https://gitlab.com/Mildoor/boustrophedon`

Dificultate: 2-3 persoane.

2.9 Implementare algoritm RRT

Unul din algoritmii utilizați pentru găsirea unei traiectorii într-un mediu cu obstacole este RRT (rapidly-exploring random trees).

Se cere:

i) Explicați algoritmul.

ii) Implementați-l/aplicați-l pentru o suprafață cu obstacole (generați/cautați hărți de spații interioare).

Indicație: Există foarte multe variante. Spre exemplu, aveți cod și explicații în `https://github.com/nimRobotics/RRT`

Dificultate: 2-3 persoane.