EÖTVÖS LORÁND TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

GEOMETRIAI TANSZÉK

Robotkarok útkeresési problémája

Témavezető:
Szeghy Dávid
adjunktus

Szerző:
Fischer Kornél
alkalmazott matematikus BSc

Budapest, 2020

Tartalom

Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, Szeghy Dávidnak, aki megismertetett ezzel a témával, figyelemmel kísérte a dolgozat megírását, és hasznos tanácsokkal látott el írás közben. Továbbá köszönöm a családomnak a támogatást, amit a tanulmányaim során nyújtottak felém.

1. Bevezetés

A robotika egy egészen új területe a mai modern tudománynak, mégis sokat találkozhatunk vele mindennapjainkban. A filmeknek és videó játékoknak köszönhetően rengeteg gyerek képzeletét ragadták meg az emberszerű, számítógép vezérelte gépek, melyek emberfeletti kitartással és precizitással képesek elvégezni a rájuk bízott feladatokat. Jelenleg viszont a legtöbbet használt robotok egyáltalán nem ilyenek. Leggyakrabban olyan munkákra használják őket, ahol nyersanyagokat vagy különböző alkatrészeket kell mozgatni, például autógyárakban. Az ilyen iparban dolgozó robotokat szokták robotkaroknak nevezni. Jelen dolgozatban is velük fogunk dolgozni.

A robotkarok egyik fő jellemzője az újraprogramozhatóság, ez azonban nem egy egyszerű feladat. Az egyik leggyakrabban előforduló probléma az útkeresés, aminek lényege, hogy a robotnak úgy kell helyet változtatnia, hogy közben nem ütközik bele semmilyen, a környezetében előforduló akadálynak. Sokszor egyéb feltételeknek is meg kell felelniük, például nem mozoghatnak túl gyorsan, nehogy sérüljön a rakományuk. Az alábbiakban ezt a problémát fogjuk körüljárni, és több megoldást is ismertetni fogunk rá.

1. ábra: Példa egy ipari robotkarra

1.1. Felhasznált irodalom és annak jelölései

A számolások során szükségünk lesz algebrai alapismeretekre. Szokásosan, a mátrixokat nagybetűvel jelöljük, a vektorokat és skalárokat kicsivel, utóbbiak esetén időnként a görög abc betűit használva. A dolgozat nagyban támaszkodik Spong, Hutchinson, és Vidyasagar Robot Modeling and Control című könyvének az úttervezésről szóló fejezetére, ideértve egy felhasznált képet is. Azon illusztrációk esetén, amik nem saját készítésűek, ez fel van tüntetve, és a felhasznált irodalom című fejezetben a forrás is meg van jelölve.

1.2. A feladat leírása

Általában a feladatunk útvonalat keresni egy adott kezdő- és végpontpár között. Miután ezt megtaláltuk, és a robot végigment rajta, új pontpárt fogunk kapni, ezért érdemes egy gyors algoritmust találnunk a probléma megoldására. Érdemes a korábbi útjaink során feltérképezett munkateret elmentenünk, hogy a későbbiekben is használhassuk a már felfedezett utakat, ezzel gyorsítva a későbbi kereséseket.

Bár a feladat meghatározása nem bonyolult, az robotkar útkeresési problémája az egyik legnehezebb feladat jelen korunk számítástudományában. Azt hogy az útkeresés feladat NP-nehéz Reif és Sharir mutatta meg 1985-ben, majd három évvel később Canny bebizonyította, hogy NP-teljes is (lásd [3] irodalom).

A mai ismert legjobb teljes algoritmus (azaz ami mindig megtalálja a megoldást, amennyiben létezik) futásideje exponenciálisan nő a szabadsági fokok számának növekedésével. Emiatt a gyakorlatban nem szoktak teljes algoritmusokat használni.

Megismerkedünk néhány olyan algoritmussal, amelyek feladata ennek a problémának gyors megoldása, a megvalósításuk egyszerű, ennek viszont az az ára, hogy nem mindig találnak megoldást. Célunk egy helyes megoldás megadása lesz, nem feltétlen egy akármilyen metrika szerint is optimális megoldás, bár erre törekedni fogunk, ha megtehetjük.

2. A robotok matematikai modellje

Mielőtt elkezdenénk az úttervezéssel és útkereső algoritmusok ismertetésével foglalkozni, alkotnunk kell egy matematikai modellt, amiben vizsgálhatjuk a kérdéskört.

2.1. A robot reprezentálása

A robotkaroknak két része van, a mozgó részeiket **csuklók**nak vagy **ízületek**nek nevezzük, a merev részeiket **szegmensek**nek, ezek együtt egy **kinematikai lánc**ot alkotnak. A lánc végén valamilyen eszköz van, amivel a robot meg tud fogni tárgyakat, ezt kéznek nevezzük. A csuklóknak két típusát különböztetjük meg, a forgó/**rotációs** csuklókat, illetve a eltoló/**prizmatikus** csuklókat. Előbbieket R-rel, utóbbiakat P-vel jelöljük. Minden csukló állapota leírható egy paraméterrel, rotációs csukló esetén ez az elfordulás szöge, prizmatikus csukló esetén az eltolás mértéke. A rotációs csuklókhoz tartozó változót θ (theta)-val jelöljük, a prizmatikus csukló változóját pedig d-vel, ezt hamarosan pontosítjuk. Ha a robotkar két végét egy szegmens-sorozat köti össze, zárt láncúnak nevezünk, ellenkező esetben nyílt láncúnak. Mi csak az utóbbiakkal fogunk most foglalkozni. Az ilyen robotok típusát úgy adjuk meg, hogy a kezdőpontjától haladva felsoroljuk a csuklók típusait.

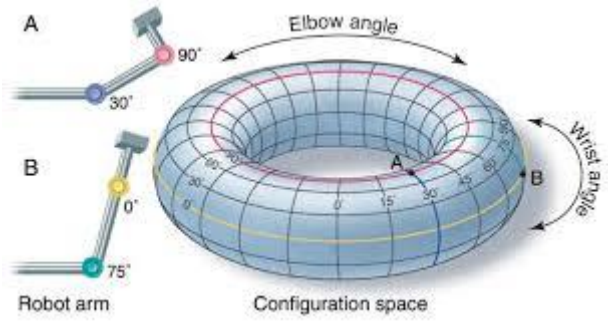
2.2. A konfigurációs tér

Amikor egy robotkar minden csuklójának koordinátáját megadjuk, azt a robot egy **konfiguráció**jának nevezzük. Az összes lehetséges konfiguráció halmaza a **konfigurációs tér,** jele **Q**. Látható, hogy a konfigurációs tér dimenziója megegyezik a csuklók számával. A mi esetünkben elég a csuklók változóit megadni, így már ismerjük a robotkar állását, egyrészt a szegmensek merevsége miatt, másrészt mert a kar alappontját fixnek tekintjük.

A továbbiakban **q**=$(q_1, q_2, \dots q_n)$ vektor fog jelölni egy konkrét konfigurációt, ahol a koordináták a csuklók változói, és $q_i = \theta_i$ ha a csukló rotációs, valamint $q_j = d_j$, ha prizmatikus.

Például egy olyan robotkar, aminek egy darab rotációs csuklója van, a konfigurációs tér reprezentánsa lehet az egységkör, azaz Q = S_1. Ilyenkor elég lesz egy paraméter, a θ_1.

Ha a robotkarunk egy síkbeli, két rotációs csuklóval rendelkező kar, akkor a reprezentáns lehetne a $Q = S_1 \times S_1 = T^2$, ahol T^2 a tóruszt jelképezi, ahogy a 2. ábrán látható. A kép forrása a dolgozat végén, a felhasznált irodalomban szerepel. Az ilyen robotkart RR-el jelöljük.



2. ábra: Példa egy RR robotkarra, és a konfigurációs terének reprezentálására. Az ábra forrását lásd a szakirodalomban.

Egy robotkarnak n a **szabadsági fok**a, ha a konfigurációja minimum n paraméterrel meghatározható. Ebből következik, hogy a szabadsági fok egyenlő a konfigurációs tér dimenziójával. Egy robotkar esetében a csuklók száma határozza meg a szabadsági fokot.

2.3. A munkatér

A későbbiekben a munkatér egy 2 vagy 3 dimenziós tér lesz, Descartes-féle koordináta rendszerrel. A tengelyek az x,y,z, az origó pedig O=(0,0) vagy O=(0,0,0). Ez szimbolizálja azt a környezetet, amiben a robot mozogni fog. A robot „teste” az origóból indul.

A munkatérben figyelembe vesszük a robot alakját, a konfigurációs térben viszont csak egy pontként reprezentáljuk. Látni fogjuk, hogy a robot alakja ilyenkor is számítani fog az úttervezésben, viszont a robot formája a térben lévő akadályok alakjában lesz tárolva.

2.4. Akadályok a konfigurációs térben

Ütközésnek nevezzük, amikor a robot érintkezik egy akadállyal, például egy fallal. Hogy ezt precízebbé tegyük, bevezetünk pár új jelölést. Ha a robotot A jelöli, akkor A(q) a munkatér egy részhalmaza, amit a robot kitölt, amikor a q konfigurációban van. Jelölje O_i a megfelelő akadályt a munkatérben, W pedig a munkateret.

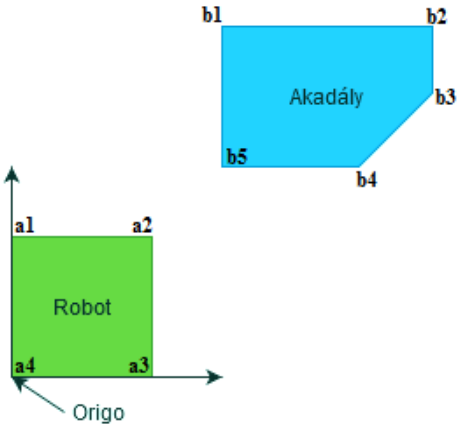
Ütközésmentes út tervezéséhez azt kell biztosítanunk, hogy a robot soha nem ér el olyan konfigurációt, amiben érintkezik egy akadállyal. A konfigurációk egy olyan halmazát, amiben ez megtörténik, **konfigurációs térbeli akadály**nak nevezünk, és így definiáljuk:

$$QO=\{q \in Q \mid A(q) \cap O \neq \emptyset\} \quad (1)$$

ahol $O = \bigcup_i O_i$. Az ütközésmentes konfigurációk halmazának a neve **szabad konfigurációs tér**, és így kapjuk:

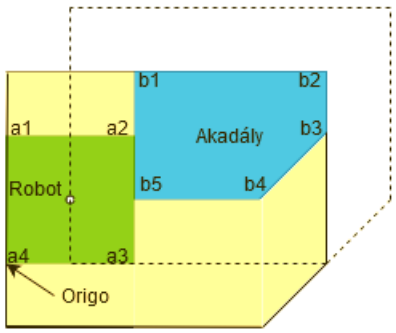
$$Q_{free} = Q \setminus QO \quad (2)$$

Nézzünk példának egy PP robotkart, tehát egy olyat, aminek kettő eltoló ízülete van. Emiatt a konfigurációs tér a sík, azaz $Q = R^2$. Legyen egy darab akadályunk, és legyen mind az akadály, mind a robotkar konvex sokszög, így könnyebb lesz a konfigurációs térbeli akadályok kiszámítása. Továbbá az egyszerűség miatt maga a kar legyen a sík fölött, így az nem ütközhet, csak a robotkar vége.



3. ábra: Robot és akadály a munkatéren

Az első ábrán a robot fejének csúcsai rendre $a_1, \dots a_4$, az akadály csúcsai pedig $b_1, \dots b_4$. Ezek segítségével definiálhatjuk azokat a pontokat, amik esetén ütközik a robotkar. A robot fejének geometriai közepét körbevezetjük az akadály körül, a 4. ábrán szaggatott vonal jelzi a pályáját. Mivel az egyetlen pont, amit használni akarunk, az az origó, ami itt az a_4 csúcs, azokat a helyeket jelöljük meg, ahova ha az origó kerül, akkor ütközés lesz. Az ábrán sárgával jelöltük, ez lesz a QO.



4. ábra: A robot fejét körbevezetve az akadály körül kijelöljük azt a területet, ahol felléphet az ütközés. Ezt a területet sárgával jelöltük

Amennyiben több konvex akadály is van, számozzuk be őket, és mindegyikre számoljuk ki a O_i-t, az uniójukból kiszámíthatjuk a QO-t. Amennyiben egy akadály nem konvex, akkor vegyük az akadály konvex burkát. Ennek több módja is van, a brute-force módszerrel n pont esetén ennek kiszámítása megoldható $O(nlog(n))$-ben. Ha ez az eljárás valamiért túl nehéz lenne, felvághatjuk az akadályt több konvex vagy „majdnem konvex" részre, és ezek konvex burkait vehetjük külön-külön. Ekkor ezeket mind külön akadályként kezelhetjük.

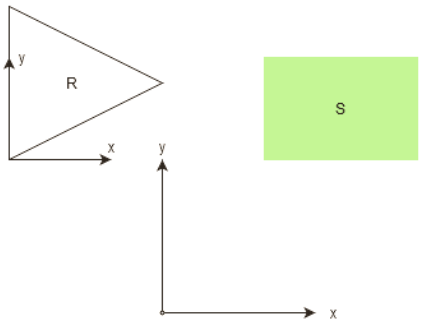
Az QO_i-k kiszámításának van más módja is, ehhez használhatjuk az úgynevezett Minkowski-összeget:

Definíció: Legyen A és B két R^d-beli halmaz. A két halmaz **Minkowski-összege:**

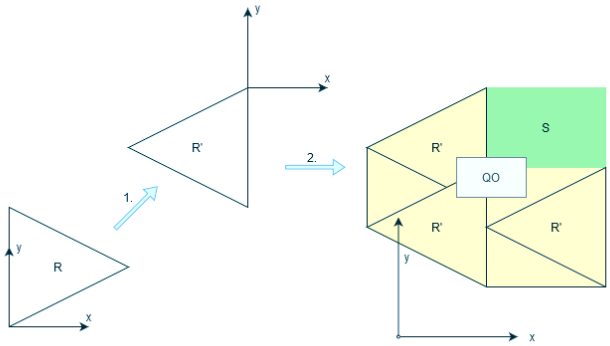
$$A+B=\{ a + b \mid a \in A, b \in B \} \quad (3)$$

Ennek a segítségével a következőképpen kaphatjuk meg a QO-t. Vegyük az A halmaznak egy a_0 pontját. Toljuk el az akadályt az a_0 vektorral. Innentől számolhatunk úgy, hogy a_0 az origóba esik, ezt az új halmazt jelölje A_0, továbbiakban vele dolgozunk. Az A_0 egy a_x pontja c-vel eltolva pontosan akkor kerül a B akadály egy b_x pontjába, ha $a_x + c = b_x$, ekkor pedig a megjelölt a_0 (vagyis az origo) pont a c-be kerülne. c-t kifejezhetjük az előző egyenletből: $c = b_x - a_x$.

E szerint, ha vesszük A_0 robotfej tükörképét az origóra, ami $(-A_0)$, akkor a $B + (-A_0)$ Minkowski összeg azon pontok halmaza lesz, ahova nem juthat a robotfej megjelölt a_0 pontja, mert ütközni fog B akadállyal. A folyamat lépéseit a 6-os mutatja be.

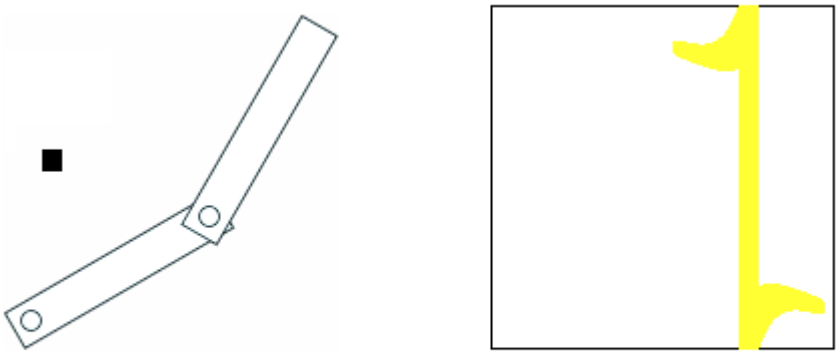


5. ábra: R robot és az S akadály a munkatéren



6. ábra: QO kiszámolásának lépései a Minkowski összeg segítségével. Először tükrözzük R robotot az origójára, majd összeadjuk az R'-t és S-t. Az így kapott QO az S-nek a reprezentációja a konfigurációs téren. Ezen a téren a robotot már nem R, hanem egy pont reprezentálja

Nézzünk egy másik példát, ezúttal legyen mindkét csukló rotációs, és legyen egy darab akadályunk, ahogy a 7-es ábra a) része mutatja. A QO az ábra b) részében van illusztrálva. A vízszintes tengely felel θ_1 változóért, a függőleges θ_2-ért. Látható, hogy amikor θ_1 nagyon közel van π/2-höz, az első szegmens ütközik az akadállyal. Ahogy θ_1 közelít π/2-höz, úgy θ_2 bizonyos értékei mellett szintén ütközik a kar. A négyzet minden pontjára volt egy ütközési teszt, és ahol a cella színezve van, ott bekövetkezett az ütközés. Ez tehát egy közelítő reprezentálása QO-nak. Ilyen kis példán is látható, hogy QO egyszerű akadályok esetén is egészen bonyolult formákat ölthet.



7. ábra: Az a) részben egy RR robotot láthatunk egy pontszerű akadállyal. A b) részben ennek a munkatérnek a konfigurációs térbeli alakját. Látható, hogy egyszerű akadálynak is bonyolult lehet a reprezentánsa a konfigurációs térben. A kép forrása az [1]-es szakirodalom

2.5. Úttervezési probléma formalizálása

Ahogy korábban említettük, az úttervezési feladat célja hogy találjunk egy utat a q_{init} kezdeti állapotból a q_{final} végső konfigurációba, úgy hogy a robotkar nem ütközik egyik akadállyal sem. Formalizálva, egy ütközésmentes út q_{init}-ből q_{final}-ba egy τ folytonos függvény, amelyre:

$$\tau:[0,1]\rightarrow Q_{free}$$

$$\tau(0)=q_{init}$$

$$\tau(1)= q_{final}$$

Ezt úgy fogjuk megoldani, hogy egy úttervező eljárással megadunk egy diszkrét konfigurációs sorozatot, majd erre illesztünk egy folytonosan differenciálható függvényt.

2.6. Robotgeometriai alapismeretek

Vegyük át, milyen robotgeometriai fogalmakra lesz szükségünk a továbbiakhoz.

Direkt kinematika feladatnak nevezzük azt a feladatot, amikor adott nekünk a robot egy q konfigurációja, és meg kell adnunk, hogy ekkor hol helyezkedik el a robotkar a munkatérben.

Direkt sebességkinematika feladat esetén adottak a paraméterek jelen pillanatban vett értékei és e paraméterek deriváltjai, célunk ezek segítségével kifejezni a kar adott pillanatbeli sebességvektorát. Ekkor szükségünk lesz tehát a paraméterek deriváltjaira, amit a Jacobi mátrix segítségével fogunk meghatározni. Ezt J-vel fogjuk jelölni.

A direkt kinematikai feladat megoldására használjuk az úgynevezett **Denavit-Hartenberg konvenciót**. Ennek lényege, hogy minden merev szegmenshez rögzítünk egy koordináta rendszert, jelölje ezt az i. szegmens esetén: (O_i, x_i, y_i, z_i), ahol O_i az origó, a maradék a három tengelyeket jelöli, amelyek jobbsodrásúak.

A koordináta rendszereknek 2 tulajdonságot kell kielégítenie:

1) A z_i tengely egybeesik az i. ízület hatástengelyével
2) Az x_{i+1} tengely merőleges a z_i tengelyre

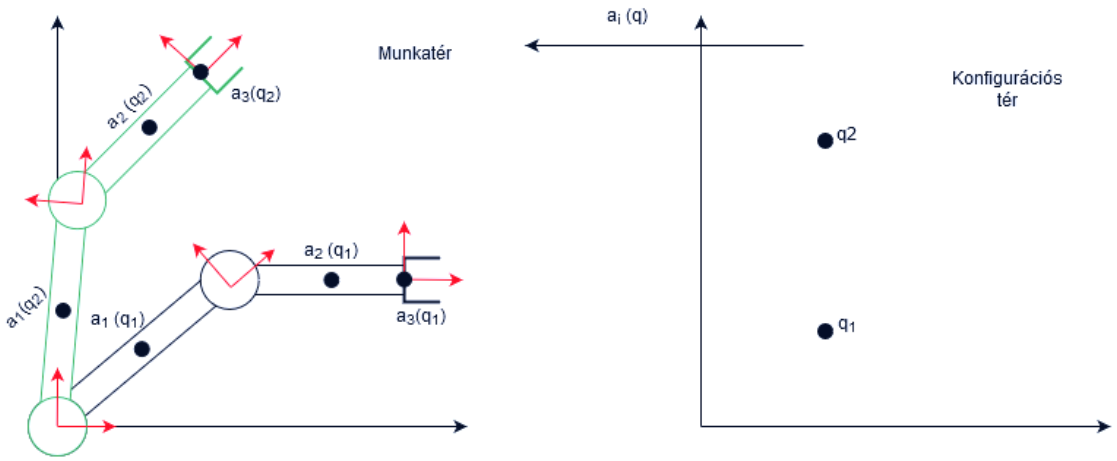
Az alaptesthez rögzített koordinátarendszert a világ koordinátarendszernek tekintjük. Kényelmi okok miatt célszerű egy ál-ízületet tenni a robotkar kezébe, és ahhoz egy ál-szegmenst csatlakoztatni úgy, hogy ennek az ál-szegmens koordinátarendszerének az origója pont a robotkar kezébe essen, azaz a robotkéz koordinátája ebben (0,0,0) legyen. Bebizonyítható, hogy ilyen koordináta rendszereket mindig fogunk találni, így használhatjuk őket a továbbiakban.

Felmerül a kérdés, hogy hogyan írhatjuk fel a különböző koordináta rendszerek közti áttérést a Denavit-Hartenberg konvenció segítségével?

A továbbiakban feltesszük, hogy a világkoordináta origója a kar bázisánál van. Bevezethető az M_i mátrix, ami az áttérést írja le az i+1. és az i. koordináta rendszer között. M_i tartalmaz négy paramétert (ha a munkatér 3 dimenziós, különben csak 3-at), de belátható, hogy ebből 3 fix, tehát csak 1 változója lesz (az is belátható, hogy ez a változó forgató ízület esetén épp a forgatás nagysága, eltoló ízület esetén pedig az eltolás nagysága lesz). Ha minden i-re meglesz az M_i mátrixunk, ezeket összeszorozva megkapjuk a M mátrixot, ami összefoglalja az összes áttérést. Ekkor, ha M-et megszorozzuk a robotkéz homogén koordinátáival, akkor megkapjuk a kéz helyzetét a világ koordinátarendszerben. Továbbá az előbbi meggondolás alapján ekkor a szorzatmátrixnak n változója lesz (az ál-ízület változója nem játszik szerepet). Ennyi elég lesz ahhoz, hogy a továbbiakban használhassuk a feladatunk megoldásához. Jelölésben $M(q)$ jelöli ezt a mátrixot, ahol q fogja össze a konfiguráció paramétereit (amelyek egyben a konfigurációs térbeli pont koordinátái is).

A robot karján ki fogunk jelölni úgynevezett kontrollpontokat. Ezeket a_i –vel jelöljük, és a munkatér elemei. Használni fogunk egy $a_i\,(q)$ függvényt, ami a konfigurációs térből képez a munkatérbe. Azt mondja meg, hogy egy adott q konfiguráció esetén hol tartózkodik az a_i pont. A mi példáinkban a kontrollpontok a karok tömegközéppontjánál, az egyszerűség kedvéért a kar közepénél fognak elhelyezkedni (feltesszük, hogy homogének, és a robotkar kézfejének nincs súlya).

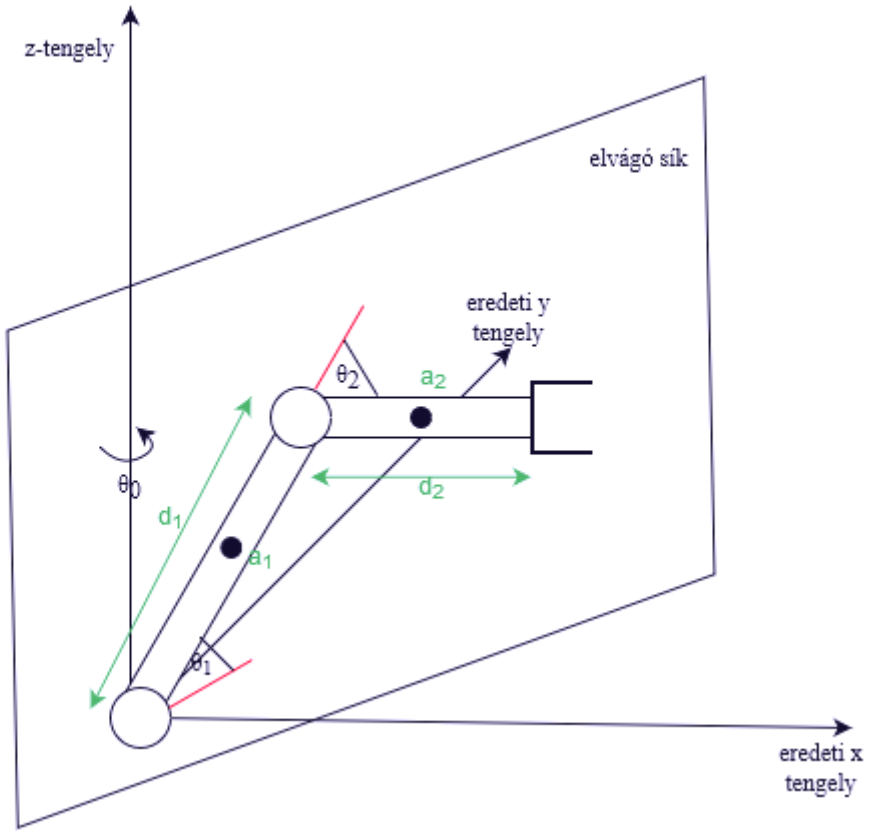
Az a_i helyzetének meghatározásához nem lesz szükségem a teljes M mátrixra, csak az $M_1 M_2 \ldots M_i$ mátrixra, mivel az a_i kontrollpont az i-dik szegmensen van. Tehát a pont a szegmenssel együtt mozog, ő egy fix koordináta a szegmenshez rögzített koordináta rendszerben, lásd 8. ábra.



8. ábra: A példán egy RR robotkar két állapota látható. A piros koordinátákat a D-H konvenció adta. Látható, hogy a_2 pont minden q esetén fix a szegmensének koordináta rendszerében

2.6.1. Példa egy számolásra

Tekintsünk egy RRR robotkart a 3 dimenziós munkatéren ahol az xy síkbeli RR robotkarunkat forgatjuk a z-tengely körül, tehát az első forgó ízület a ezen mentén van, lásd 9. ábra. Oldjuk meg a direkt kinematika feladatot a Denavit-Hartenberg konvenciót használva. Ha egy q_{final}-on átmenő, z-t tartalmazó síkkal elmetsszük a teret, a feladat visszavezethető egy RR robotkar direkt kinematika feladat megoldására, amit ezután el kell még forgatnunk a z-tengely körül valamilyen szöggel.

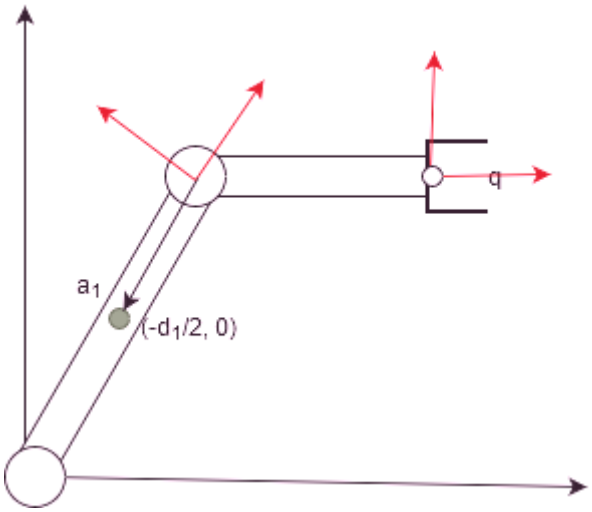


9. ábra RRR robotkar a háromdimenziós munkatéren. A cél konfiguráció a bejelölt sík eleme. Ha a feladatot ezen a síkon oldjuk meg, visszavezethető egy RR robotkar direkt kinematika feladat megoldására

Szóval először oldjuk meg RR robotkarra a feladatot. Vagyis adott egy q=(θ_1, θ_2) konfiguráció, meg kell adnunk a kontrollpontok helyét a munkatérben (ami most még szintén csak 2 dimenziós, hiszen az elvágó síkban vagyunk).

A két kontrollpontunk a_1 és a_2, a szegmenseinken rendre $d_1/2$ illetve $d_2/2$ távolságra vannak a megfelelő koordináta rendszerektől.

Legyen $v_1 = (-\frac{d_1}{2}, 0)$ az a_1 pont koordinátája a 2 ízület koordináta rendszerében, ez leolvasható. Homogén koordinátákkal felírva $v_1 = (-\frac{d_1}{2}, 0, 1)$.



10. ábra: Az a_1 pont koordinátája a második koordinátarendszerben állandó

Az első R karhoz tartozó változó θ_1, ezért az áttérést leíró M_1 mátrix az alábbi:

$$M_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & d_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & d_1\sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Elvégezve a szorzást:

$$M_1 v_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & d_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & d_1\sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{d_1}{2} \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{d_1}{2}\cos\theta_1 \\ \frac{d_1}{2}\sin\theta_1 \\ 1 \end{bmatrix} \tag{5}$$

Vagyis a világkoordinátában a_1 homogén koordinátái $\left(\frac{d_1}{2}\cos\theta_1 \quad \frac{d_1}{2}\cos\theta_1 \quad 1\right)$.

Az M_2 mátrix, ha a második szög θ_2 és a szegmens hossza d_2:

$$M_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & d_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & d_2\sin\theta_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Könnyen látható, hisz az átállás hasonló, mint az első esetben: az x tengelyt el van forgatva θ_2 -vel, és el van tolva d_2-vel.

A második kontrollpont koordinátái a robotkar fejéhez illesztett koordináta rendszerben, és homogén koordinátájúvá kiegészítve:

$$v_2 = \left(-\frac{d_2}{2}, 0, 1\right) \tag{7}$$

Ha most meg akarjuk kapni ennek a pontnak a helyét a világ koordináta rendszerében (x és y), akkor ezt a mátrixszorzást kell elvégeznünk:

$$M_1 M_2 v_2 = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{8}$$

Elvégezve a számolást:

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & d_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & d_1\sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & d_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & d_2\sin\theta_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{d_2}{2} \\ 0 \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta_1\cos\theta_2 - \sin\theta_1\ \sin\theta_2 & -\cos\theta_1\sin\theta_2 - \sin\theta_1\cos\theta_2 & d_2\cos\theta_2\cos\theta_1 - \sin\theta_1 d_2\sin\theta_2 + d_1\cos\theta_1 \\ \cos\theta_2\ \sin\theta_1 + \cos\theta_1\sin\theta_2 & -\sin\theta_1\ \sin\theta_2 + \cos\theta_1\cos\theta_2 & d_2\cos\theta_2\ \sin\theta_1 + \cos\theta_1 d_2\sin\theta_2 + d_1\sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{d_2}{2} \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{d_2}{2}(\cos\theta_1\cos\theta_2 - \sin\theta_1\ \sin\theta_2) + d_2\cos\theta_2\cos\theta_1 - \sin\theta_1 d_2\sin\theta_2 + d_1\cos\theta_1 \\ -\frac{d_2}{2}(\cos\theta_2\ \sin\theta_1 + \cos\theta_1\sin\theta_2) + d_2\cos\theta_2\ \sin\theta_1 + \cos\theta_1 d_2\sin\theta_2 + d_1\sin\theta_1 \\ 1 \end{bmatrix} = \tag{9}$$

$$= \begin{bmatrix} \frac{d_2}{2}(\cos\theta_1\cos\theta_2 - \sin\theta_1\ \sin\theta_2) + d_1\cos\theta_1 \\ \frac{d_2}{2}(\cos\theta_2\ \sin\theta_1 + \cos\theta_1\sin\theta_2) + d_1\sin\theta_1 \\ 1 \end{bmatrix}$$

Vagyis az a_2 kontrollpont helyzete a világkoordinátában:

$$a_2(q) = \begin{bmatrix} \frac{d_2}{2}(\cos\theta_1\cos\theta_2 - \sin\theta_1\ \sin\theta_2) + d_1\cos\theta_1 \\ \frac{d_2}{2}(\cos\theta_2\ \sin\theta_1 + \cos\theta_1\sin\theta_2) + d_1\sin\theta_1 \end{bmatrix} = \tag{10}$$

$$= \begin{bmatrix} \mathrm{d}_1\cos\theta_1(\mathrm{t}) + \frac{\mathrm{d}_2}{2}\cos(\theta_1(\mathrm{t}) + \theta_2(\mathrm{t})) \\ \mathrm{d}_1\sin\theta_1(\mathrm{t}) + \frac{\mathrm{d}_2}{2}\sin(\theta_1(\mathrm{t}) + \theta_2(\mathrm{t})) \end{bmatrix}$$

Tehát megvan, hogy kell kiszámolnunk a direkt kinematikai feladatot RR robotkar esetén. Ahhoz, hogy a példában szereplő RRR robotkarra is megoldjuk a feladatot, először át kell írnunk mindent homogén koordinátájúvá, vagyis kell egy negyedik koordináta. Minden mátrixba be kell szúrnunk egy 3. oszlopot és egy harmadik sort, mindkettő $(0 \quad 0 \quad 1 \quad 0)$ legyen. Mivel ez a koordináta 0 volt végig, az eredményen ez nem változtat, de minden mátrix már 4×4-es lesz. Ezután az eddigi eredményt meg kell szoroznunk M_0-val, ahova szintén beszúrtunk egy 4. sort és oszlopot, mindkettő legyen $(0 \quad 0 \quad 0 \quad 1)$. Homogén koordináták esetén mindig egyel több koordinátát írunk fel, mint amennyink van.

$$\mathrm{M}_0 = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & 0 & 0 \\ \sin\theta_0 & \cos\theta_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

3. A gradiens módszer

A következő fejezetben az egyik legelterjedtebb megoldási módszert fogjuk kielemezni, a gradiens módszert. Az eddig ismertetett számolásokat fogjuk kihasználni, valamint alkotunk egy potenciál mezőt, melynek segítségével leírjuk, milyen útvonalon kell végigmennie a robotkarnak.

3.1. A potenciálmezők

Mostanra példán is láthattuk, hogyan kell egy adott q konfigurációhoz megkeresni a munkatéren a robot jelenlegi állását. Vagyis ha a konfigurációs téren találunk egy utat, az út pontjai megadják a robot útját a munkatéren is.

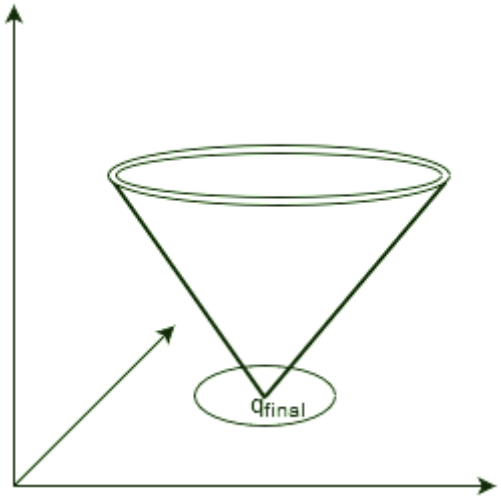
A potenciálmezők a következő ötleten alapszanak. A robotot a konfigurációs térben egy pont jelöli. Erre a pontra hatnak erők, amik miatt mozogni fog. A mezőt (és ez által az erőket) mi alkotjuk meg, szem előtt tartva, hogy a kar elkerülje az akadályokat, és megérkezzen a célba. Emiatt két részből fogjuk összeállítani a mezőt: egy taszító (rep) mezőből, és egy vonzó (att) mezőből. A taszító miatt nem ütközünk akadályba, a vonzó miatt haladunk a végcél fele.

Az erőket úgy fogjuk megkapni, hogy vesszük a gradiensét (deriváltakból álló vektorát) a mezőnek. A gradiens egy pontban azt mondja meg, hogy melyik irányba nő legjobban a függvény egységnyi távolságon belül. Mi úgy fogjuk megalkotni a mezőt, hogy a végpontban minimuma legyen a függvénynek. Vagyis nekünk a gradiens mínuszegyszeresére lesz szükségünk, hogy melyik irányban csökkenünk leggyorsabban.

Itt persze feltettük, hogy a mezőnek egyetlen minimumhelye van, a q_{final}. Ez sajnos sokszor nem igaz, létezhetnek lokális minimumok, amikben elakadhat az algoritmus. Ennek a problémának megoldásával később fogunk foglalkozni.

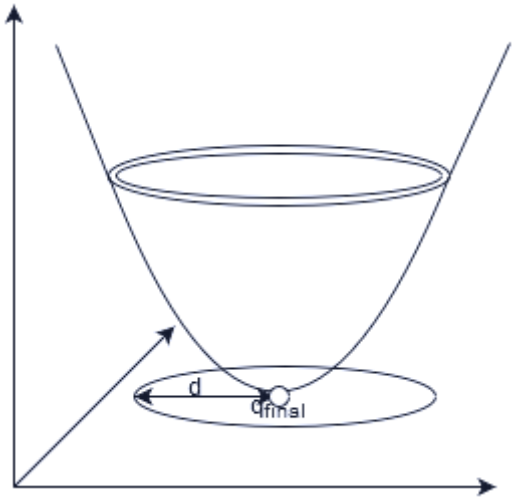
3.1.1. A vonzó mező

Különböző módszerekkel adhatjuk meg a mezőket. Egyik ilyen a conic well, ami egy kúp, aminek a csúcsa a q_{final} . Ezzel az lehet a probléma, hogy a célpontban nem lesz deriválható, és nem csökken a meredeksége a cél közelében, azaz nem fog az algoritmus lelassítani, rövidebbet, „óvatosabbat” lépni a cél közelében.



11. ábra: Látható, hogy a kúpos vonzó mező nem lesz deriválható a célban

Nem kell teljesen elvetnünk ezt az ötletet, csak meg kell javítanunk a cél környékén, az ötlet pedig a következő: a cél „közelében” vágjuk le a csúcsát a kúpnak, hogy parabolikus íve legyen. Figyeljünk rá, hogy a 2 ív csatlakozzon, és a deriváltjuk is illeszkedjen, hogy az áttérés elég sima legyen, és ne zavarjon bele az erők számításába. Az oka, hogy nem csak a parabolikus ívet használjuk az, hogy ha messze vagyunk a céltól, túl erősen hat, túl nagyot lépne az algoritmus és esetleg e miatt ütköznénk akadályba. Ezért kell a kettőt kombinálni.



12. ábra: A vonzó mező a konfigurációs téren

Ahhoz hogy formalizálni tudjuk, szükségünk lesz egy metrikára, legegyszerűbb az euklideszit használni:

$$\rho(q) := \|q - q_{final}\| = \left(\sum_{i=1}^{n} (|q_i - q_{final_i}|)^2\right)^{\frac{1}{2}} \quad (12)$$

Vagyis a ρ a konfigurációs téren van értelmezve. Először nézzük meg csak a parabolikus esetet. Mivel parabolát akarunk, a távolság négyzetét használjuk a mező definiálásához:

$$U_{att}(q) = \frac{1}{2}\zeta \rho^2(q) \tag{13}$$

A ζ –t azért fogjuk használni, hogy szabályozzuk a vonzás erősségét. Ez az U mező a konfigurációs téren van értelmezve. Elméletileg itt is kiszámolhatnánk ehhez a potenciálfüggvényhez tartozó vonzóerőt és hasonlóan az akadályoknak valamilyen taszítását, de a konfigurációs téren már maguknak az akadályoknak a számolása is elég nehéz. Itt kiszámolni ezeket az erőket és ezek szerint leírni a q_{init} pont mozgását amíg eljut q_{final}-ba egy nehéz feladat. Ezért inkább megpróbáljuk a munkatéren kivitelezni ezt az ötletünket.

Tehát valahogy ezt az egyenletet át kell írnunk, hogy a munkatéren értelmezhessük. Használjuk ehhez a korábban megismertetett $a_i(q)$ függvényt. Emlékeztetőül, ez megmondja, hogy egy adott q konfiguráció esetén hol helyezkedik el az a_i pont a munkatéren. Az a_i munkatérbeli pont célja elérni azt a pontot, amit akkor vesz fel, amikor elérjük q_{final} konfigurációt. Ez a pont az $a_i(q_{final})$. Azaz definiáljuk a fenti U_{att} függvényt külön-külön minden a szegmenseken elhelyezett a_i kontrollpontra, és ezek célpozíciója lesz a saját potenciálfüggvényük $U_{att,i}$ minimum helye. Az $a_i(q_{final})$ ponttól vett távolságának a négyzetével arányos a mező nagysága. Tovább gondolva az előző egyenletet:

$$U_{att,i}(\underline{x}) = \frac{1}{2}\zeta_i \left|\left|\underline{x} - a_i(q_{final})\right|\right|^2 \tag{14}$$

Hogy értelmezzük ezt az egyenletet? Minden pontra adott lesz egy ζ_i, ugyanaz lesz a haszna, mint a ζ-nak. Mivel nem elvárás, hogy minden kontrollpontra ugyanaz legyen a ζ_i, ezért szokás „súlyozni” a pontokat. Néhányat jobban vonz a célja, gyorsabban halad arra, ezzel (heurisztikusan) maga után húzva a többi pontot. Ez az $U_{att,i}$ függvény a munkatérről képez a valós számokra és x helyett majd $a_i(q)$-t fogjuk helyettesíteni, hiszen a q konfigurációtól függ, hogy hol vannak a robotkar kontrolpontjai.

De mi nem szeretnénk, ha mindenhol négyzetes lenne a mező, csak egy bizonyos távolságig. Jelölje ezt a távolságot az a_i kontrollpont esetén d_i. Ezt a környezetet is a kontrollponttól függően választhatjuk meg, azaz az adott indextől függ, hogy a kontrolpont a cél milyen környezetében „lassítson le”. Vagyis az $a_i(q_{final})$-nak a d_i sugarú körében a fenti egyenlet legyen, de utána a távolságtól lineárisan függjön a vonzás. De azt is szeretnénk, ha a két

egyenlet deriváltja is összeérne. Számoljuk ki, mi a deriváltja (gradiense) a fenti mezőnek. Ehhez nézzük meg, koordinátánként mit kapunk deriváláskor:

$$U_{att,i}\left(\underline{x}\right) = \frac{1}{2}\,\zeta_i\,\left|\left|\underline{x} - a_i(q_{final})\right|\right|^2 = \frac{1}{2}\,\zeta_i\,(\sum_{j=1}^{n}(x_j - a_i(q_{final})_j)^2) \tag{15}$$

Hogy ne bonyolódjunk bele a sok indexbe, legyen $a_i(q_{final}) = (c_1, c_2, \dots c_n)$. Ekkor a k-dik változó szerinti parciális derivált:

$$\delta_k U_{att,i}\,(x_1, x_2, \dots x_n) = \frac{1}{2}\,\zeta_i\,2\,(x_k - c_k) = \zeta_i\,(x_k - c_k) \tag{16}$$

Ebből pedig a gradiens:

$$\begin{aligned}\nabla\, U_{att,i}\,(x_1, x_2, \dots x_n) &= \left(\,\delta_1 U_{att,i}, \delta_2 U_{att,i}, \dots\, \delta_n U_{att,i}\,\right) = \\ &= \left(\zeta_i\,(x_1 - c_1), \zeta_i\,(x_2 - c_2), \dots\, \zeta_i\,(x_n - c_n)\right) = \zeta_i\,\left(\underline{x} - \underline{c}\right) \\ &= \zeta_i\,(\underline{x} - a_i(q_{final}))\end{aligned} \tag{17}$$

Most szeretnénk átírni $U_{att,i}$ –t, hogy d_i-nél távolabb lineáris legyen, és d_i távolságra a deriváltak megegyezzenek. Ezt pedig ez az egyenlet teljesíti:

$$U_{att,i}(\underline{x}) = d_i\,\zeta_i\left|\left|\underline{x} - a_i(q_{final})\right|\right| - \tfrac{1}{2}\,\zeta_i\,{d_i}^2 \qquad \text{ha } \left|\left|\underline{x} - a_i(q_{final})\right|\right| > d_i \tag{18}$$

Ellenőrizzük a gradienst ekkor:

$$U_{att,i}(\underline{x}) = d_i\,\zeta_i\left|\left|\underline{x} - \underline{c}\right|\right| - \frac{1}{2}\,\zeta_i\,{d_i}^2 = d_i\,\zeta_i\sqrt{\sum_{j=1}^{n}(x_j - c_j)^2} - \frac{1}{2}\,\zeta_i\,{d_i}^2 \tag{19}$$

A korábbi jelöléseket használva megint:

$$\delta_k U_{att,i}\left(\underline{x}\right) = d_i\,\zeta_i\frac{1}{2}\,\frac{1}{\sqrt{\sum_{j=1}^{n}(x_j - c_j)^2}}\,2\,(x_k - c_k) = d_i\,\zeta_i\,\frac{(x_k - c_k)}{\sqrt{\sum_{j=1}^{n}(x_j - c_j)^2}} \tag{20}$$

Ebből pedig a gradiens:

$$\nabla U_{att,i}\left(\underline{x}\right) = \left(\delta_1 U_{att,i}, \delta_2 U_{att,i}, \dots \delta_n U_{att,i} \right) =$$

$$= \left(d_i\, \zeta_i \frac{(x_1 - c_1)}{\sqrt{\sum_{j=1}^{n}(x_j - c_j)^2}}, \dots , d_i\, \zeta_i \frac{(x_n - c_n)}{\sqrt{\sum_{j=1}^{n}(x_j - c_j)^2}} \right) = \tag{21}$$

$$= d_i\, \zeta_i \frac{\underline{\mathrm{x}} - a_i(q_{final})}{\left|\left|\underline{\mathrm{x}} - a_i(q_{final})\right|\right|}$$

Ebből most már összerakhatjuk a teljes munkatérre értelmezett vonzó mezőt:

$$\mathrm{U_{att,i}}(\underline{\mathrm{x}}) = \begin{cases} \frac{1}{2}\,\zeta_{\mathrm{i}}\, \left|\left|\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})\right|\right|^2 & ha\ \left|\left|\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})\right|\right| \le \mathrm{d}_i \\ \mathrm{d}_i\, \zeta_{\mathrm{i}} \left|\left|\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})\right|\right| - \frac{1}{2}\,\zeta_{\mathrm{i}}\, {\mathrm{d}_i}^2 & ha\ \left|\left|\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})\right|\right| > \mathrm{d}_i \end{cases} \tag{22}$$

Látható, hogy ha a távolság pont d_i, a két egyenlet egybeesik.

Az erőt pedig úgy kapjuk meg, hogy vesszük a potenciál gradiensét, és szorozzuk mínusz eggyel, hogy a legnagyobb csökkenés irányába lépjünk:

$$F_{att,i}(\underline{\mathrm{x}}) = -\nabla \mathrm{U_{att,i}}(\underline{\mathrm{x}}) =$$

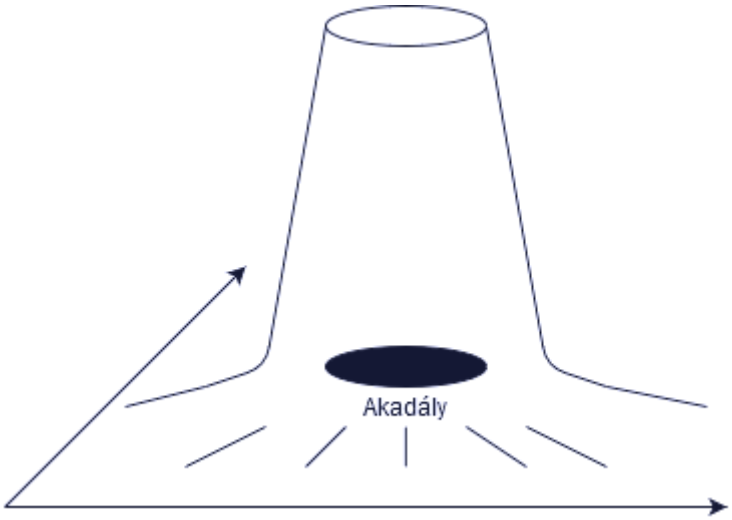
$$= \begin{cases} -\zeta_{\mathrm{i}}\, \left(\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})\right) & ha \left|\left|\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})\right|\right| \le \mathrm{d}_i \\ -\mathrm{d}_i\, \zeta_{\mathrm{i}} \frac{\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})}{\left|\left|\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})\right|\right|} & ha \left|\left|\underline{\mathrm{x}} - a_{\mathrm{i}}(\mathrm{q_{final}})\right|\right| > \mathrm{d}_i \end{cases} \tag{23}$$

Itt is látható, hogy ha a távolság pont d_i, a két egyenlet ugyanazt adja. Vagyis a bevezetett $\mathrm{U_{att,i}}$ tényleg folytonos, és a gradiense is szépen csatlakozik.

Eddig tehát definiáltunk minden kontrollpontra egy erőt a munkatéren, amit ez a vonzó mező határozott meg. De mint korábban említettük, szükségünk lesz arra is, hogy a robotot taszítsuk az akadályoktól. Szerencsére hasonló meggondolásokkal azt a mezőt is meg tudjuk alkotni.

3.1.2. A taszító mező

Ezúttal azt szeretnénk, ha a mező nagysága egy akadályon, illetve a közelében nagyon nagy lenne, lehetőleg tartson végtelenhez, de távolodva tőle csökkenjen. Sőt, bizonyos távolság után már egyáltalán ne legyen hatása, nem szeretnénk ugyanis, ha a taszító erő ellökné a robotot a q_{final} közeléből. Egy olyan függvény, ami ilyen tulajdonságokkal rendelkezik az $\frac{1}{x}$.



13. ábra: Minél közelebb kerülünk az akadályhoz, annál erősebb a taszítása

Ismét csak a távolság függvényében adjuk meg a mezőt leíró képletet. Ehhez szükségünk van egy ponthalmaz (az akadály) és egy külső pont távolságára. Ezt visszavezetjük két pont távolságára, mégpedig úgy, hogy a külső ponthoz legközelebbi akadálybeli ponttól mérjük a távolságot.

Ezt általános esetben elég nehéz lehet megadni. Ha az akadály széle „szép” lenne, például egyenes, akkor erre kéne merőlegest állítani a ponton keresztül, és így kapnánk meg a távolságot. Általánosságban is, akkor távolodunk leggyorsabban egy akadálytól, megkeressük az akadály azon pontját, mely a legközelebb van a pontunkhoz, majd ezen egyenes mentén kezdünk távolodni az akadályunktól.

Állítsunk tehát elő egy olyan függvényt, ami a távolság reciprokjától függ, de egy d távolság után lenullázódik:

$$U_{rep,i}(\underline{x}) = \begin{cases} \frac{1}{2}\eta_i \left(\frac{1}{\rho(\underline{x})} - \frac{1}{d_{r,i}}\right)^2 & \text{ha } \rho(\underline{x}) \leq d_{r,i} \\ 0 & \text{ha } \rho(\underline{x}) > d_{r,i} \end{cases} \quad (24)$$

A képletben $\rho(\underline{x})$ az $\underline{x}$ és az i. akadály közti legrövidebb távolságot jelöli. Látható, hogy ez a munkatérből képező függvény rendelkezik a kívánt tulajdonságokkal. Az indexben az i jelöli, hogy ez az i-dik akadályhoz tartozó függvény.

Ebben a képletben $d_{r,i}$ szerepe hasonló, mint a vonzó mezőnél volt, de nem feltétlen ugyanaz az érték, mint ott. A taszító mezők hatótávjának megválasztásakor arra szokás figyelni, hogy egyik akadály hatósugarában se legyen egyik kontrollpont végállapota se.

A η_i szerepe ugyanaz, mint a vonzó mező esetén a ζ_i -nek volt, szabályozza a taszítás mértékét. Nem szükséges minden i-re ugyanakkorának lennie, szokásos eljárás, hogy a célhoz közeli akadályoknak kisebb a taszításuk, nehogy ellökjék a robotot a céljától.

Ahhoz, hogy megkapjuk a taszító erőt, ki kell számolnunk ennek az $U_{rep,i}$ –nek a gradiensét. Hasonlóan számolunk, mint eddig, koordinátánként nézzük meg:

$$\begin{aligned}\delta_k U_{rep,i}(\underline{x}) &= \frac{1}{2}\eta_i 2\left(\frac{1}{\rho(\underline{x})} - \frac{1}{d_{r,i}}\right)\delta_k \frac{1}{\rho(\underline{x})} = \\ &= \eta_i\left(\frac{1}{\rho(\underline{x})} - \frac{1}{d_{r,i}}\right)(-1)\frac{1}{\rho(\underline{x})^2}\,\delta_k\rho(\underline{x})\end{aligned} \tag{25}$$

Ebből látható, hogy a gradiens így fog kinézni:

$$\begin{aligned}\nabla U_{rep,i}(\underline{x}) &= -\eta_i\left(\frac{1}{\rho(\underline{x})} - \frac{1}{d_{r,i}}\right)\frac{1}{\rho(\underline{x})^2}\left(\delta_1\rho(\underline{x}), \delta_2\rho(\underline{x}), \dots\, \delta_n\rho(\underline{x})\right) = \\ &= -\eta_i\left(\frac{1}{\rho(\underline{x})} - \frac{1}{d_{r,i}}\right)\frac{1}{\rho(\underline{x})^2}\,\nabla\rho\end{aligned} \tag{26}$$

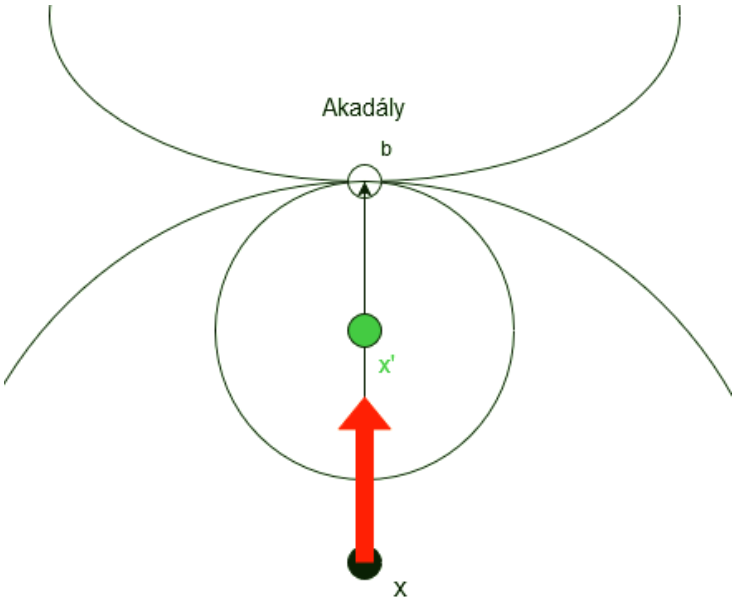
Látható, hogy a képletben az utolsó átalakításnál egy vektorban megjelennek $\rho(\underline{x})$ parciális deriváltjai. Ez pedig definíció szerint ρ gradiense. Vagyis ezt is ki kéne számolnunk, ez viszont egy bonyolult számítás. Ami segíthet minket, az a korábbi heurisztikánk: a függvény akkor nő a leggyorsabban, ha az akadály felé megyünk a lehető legrövidebb úton.

Egyszerű példán szemléltetve: legyen a pontunk x. Van mellette egy A akadály, annak x-hez legközelebbi pontja b. Ez azt jelenti, hogy más akadálybeli pont nincs a x középpontú, |x-b| sugarú körben. Ha elindulunk b fele, és eljutunk x'-be, továbbra is b lesz a legközelebbi pont, mert ezúttal egy kisebb körben keresünk további akadálybeli pontokat. Ez a kisebb kör része a nagyobbnak, és már a nagyobban sem volt más akadálybeli pont. Ebből látszik, hogy a gradiens mínuszegyszeresének erre fele kell mutatnia.

E szerint a gondolat szerint tehát a ρ gradiense felírható így:

$$\nabla\rho(\underline{x}) = \frac{\underline{x} - b}{||\underline{x} - b||} \tag{27}$$

ahol b az ábrán is látható legközelebbi akadálybeli pont. Ez tehát egy b felé mutató egységnyi hosszú vektor.



14. ábra: Gradiens keresése egyszerű példán. A piros nyíl irányába mutat a gradiens

Tehát ezt összefoglalva, a taszító mező megad egy taszító erőt a munkatéren, mégpedig ezt:

$$F_{rep,i}(\underline{x}) = \begin{cases} -\eta_i \left(\frac{1}{\rho(\underline{x})} - \frac{1}{d_{r,i}}\right) \frac{1}{\rho(\underline{x})^2} \nabla\rho(\underline{x}) & \text{ha } \rho(\underline{x}) \leq d_{r,i} \\ 0 & ha\ \ \rho(\underline{x}) > d_{r,i} \end{cases} \quad (28)$$

3.1.3. Az erők összegzése

Mostanra megadtunk egy a_i-re ható vonzó erőt, és minden akadály részéről egy a_i-re ható taszító erőt. Ha egy adott q konfigurációban az $a_i(q)$ kontrolpontunkra ható erőket akarjuk kiszámítani, akkor az $F_{rep,j}(x)$ függvényekbe és az $F_{att,i}(x)$ vonzó függvényünkbe az x helyett az $a_i(q)$ pontot kell helyettesítenünk. Ezeket összeadva megkapjuk azt az erőt, ami az a_i kontrolpontunkra fog hatni a munkatéren.

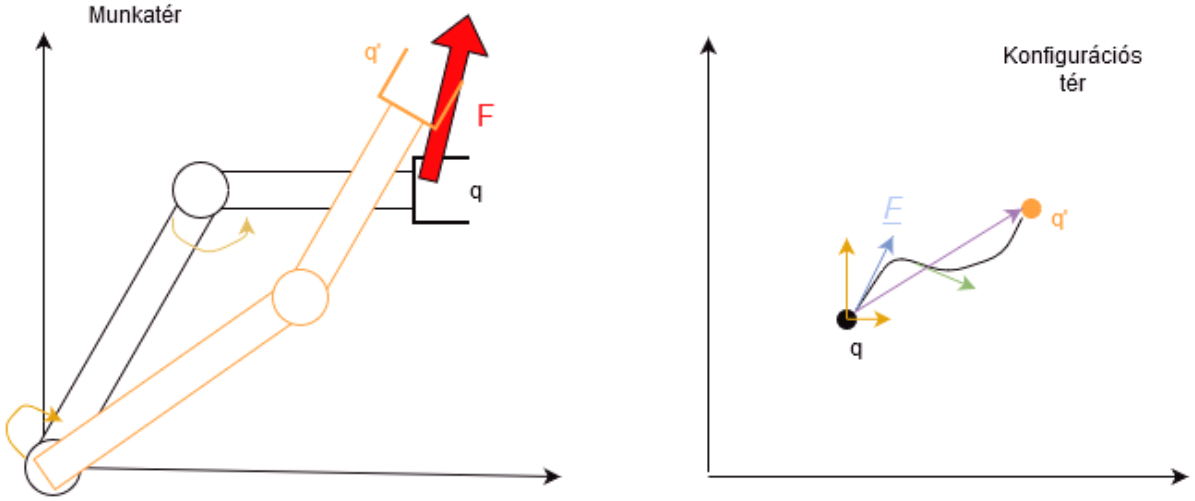
$$F\left(a_i(\underline{q})\right) = F_{att,i}\left(a_i(\underline{q})\right) + \sum_j F_{rep,j}\left(a_i(\underline{q})\right) \quad (29)$$

A képletben a szumma indexe az akadályokhoz tartozik. Ha ezt minden kontrollpontra megcsináljuk, kiszámolhatjuk majd a robot elmozdulását a munkatéren.

3.1.4. Számolás az erőkkel

Jelenleg tehát van egy q konfigurációnk a konfigurációs térben, egy ehhez tartozó állás a munkatérben, ott a robot szegmensein kontrollpontok, és ezen pontok mindegyikére hat egy erő, amit kiszámoltunk (mostanra összeadtuk a taszító és vonzó erőket). Most először megvizsgáljuk egy pontra, hogy mit fog ez kiváltani, majd megnézzük, hogyan kell ezeket a hatásokat összegezni az egész karra.

Gondoljunk egy RR robotkarra, ami éppen egy q konfigurációban van. Kezdjük elmozgatni a robot kezét egy F erővel. Tegyük fel, hogy nincs súrlódás, a csuklók anélkül tudnak elfordulni. Ahogy az F erő hat a robotra, az másik konfigurációkba kerül át. Minden pillanatban, amikor az erő hat, a konfigurációs térben is lesz egy úgynevezett virtuális erő, ami megváltoztatja a konfigurációt. De amíg a munkatérben ugyanaz az erő hat, a Q térben lehet, hogy minden pillanatban más erő lép fel.



15. ábra: A munkatéren végig ugyanaz a F erő hat. Ez elmozdítja a csuklókat, átviszi őket q' konfigurációban. De amíg odaér, különböző F virtuális erők lépnek fel a konfigurációs térben

Az F mindkét csuklót mozgásra fogja késztetni, ezt mutatják a nyilak a munkatéren. A csuklókra hatni fognak forgató erők. Ezek konfigurációs térben úgy jelennek meg, mint egy-egy erő, ami az adott paraméter tengelye mentén hat, az ábra bal felén a 2 narancssárga nyíl ezt ábrázolja. Az első csuklóra ható forgató erő a vízszintes változásért felel, a második a függőlegesért. Ezeknek az erőknek az eredője lesz a F erő, abba az irányba fog elmozdulni a q pont. Később, amikor már máshelyzetben van, az eredő erő is megváltozhat, ezt mutatja a zöld nyíl. Mire a munkatéren befejeződik a mozgás, a robotkar átér a q' konfigurációba.

Így láthattunk egy példát rá, hogy a munkatérben fellépő erők milyen virtuális erőket indukálnak a konfigurációs téren.

Most továbbra is a robotkarnak csak egy pontját fogjuk meg, a többi kontrollpontról még elfeledkezünk.

Az F és F közti kapcsolat megértéséhez szükségünk lesz a virtuális munka tételére, amihez pedig szükségünk lesz a munka fogalmára is. Fizikából ismert, hogy a munka az erő és elmozdulás skaláris szorzata:

$$W =< F; s > \tag{30}$$

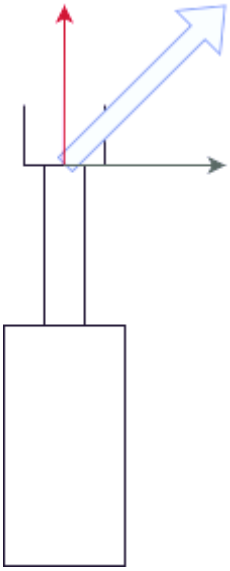
A virtuális munka tételének bizonyítása rendkívül bonyolult, meghaladja ennek a dolgozatnak a kereteit. Bizonyítása megtalálható az [1] szakirodalomban. Mi csak a következményeit fogjuk használni. A tétel röviden összefoglalva a következő:

A konfigurációs térben fellépő F erő hatására meg fog változni a q konfiguráció valamennyivel. Ez a megváltozás felírható így:

$$\delta q = (q_{0+\varepsilon} - q_0) \tag{31}$$

Vagyis a q_0 állapotból „arrébb mentünk egy keveset”. Ez a pici δq elmozdulás attól következett be, hogy F erő egy „kicsi” ideig hatott a q pontra.

Hasonlóan a munkatérben, a vizsgált pont el fog mozdulni valamilyen irányban az erő hatására. Nem feltétlen arra, amerre az erő hat, példa erre az alábbi P robotkar:



16. ábra: A karra a kék erő hat, megváltozás mégis csak a piros nyíllal jelzett irányban fog történni. Feltesszük, hogy a robot bázisa nem tud mozogni.

Figyelni kell, hogy a ható erőnek melyik az a vetülete, amelyik irányban felléphet az elmozdulás. A példában szereplő kar csak a függőleges tengelymentén tud változni, így a ható F erőnek a függőleges komponense határozza meg a vizsgált pont (most a robot keze) elmozdulását. Összefoglalva, a kényszerfeltételek miatt a munkatérben az erő és az elmozdulás nem lesz párhuzamos, nem úgy, mint egy szabad pont esetén, amit ha valamerre húzok egy F erővel, akkor arra fog elmozdulni. Azonban a konfigurációs térben már igaz lesz, hogy az ottani erő és q pont elmozdulása párhuzamos és egyirányú is lesz.

A virtuális munka tétele viszont azt mondja ki, hogy a két erő által végzett munka megegyezik. Vagyis mindegy, hogy a munkatérben, vagy a konfigurációs térben nézem a pont elmozdulását, a két munka azonos. A munkatérben vizsgált pontot jelölje $\underline{x}$, ennek elmozdulása $\delta\underline{x}$. Így felírva a tétel által adott képlet:

$$F\,\delta x = \underline{F}\,\delta q \tag{32}$$

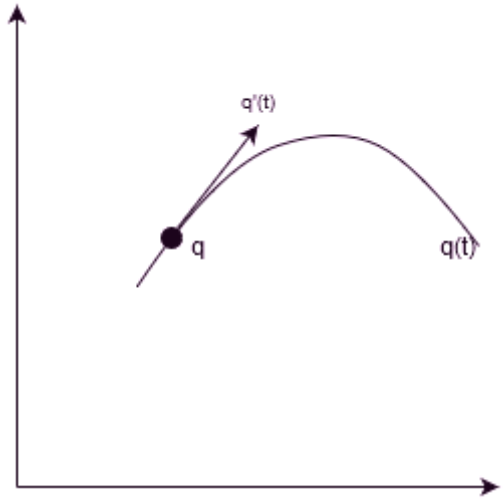
A képletben F egy $1 \times m$ –es vektor, δx az $m \times 1$ –es, $\underline{F}$ az $1 \times n$-es, δq pedig $n \times 1$ –es, ahol m a munkatér dimenziója, n pedig a csuklók száma, vagyis a konfigurációs tér dimenziója.

Ezt a képletet, ha elfogadjuk, tovább alakíthatjuk, könnyítve a számításainkat.

A 2.6.1. fejezetben, illetve az utána következő példán láthattuk, hogy ha adott egy q konfiguráció, akkor megadhatunk M_i mátrixokat, amelyeket egymással összeszorozva, és

összeszorozva egy adott pont homogén koordinátájával, megkaphatjuk annak a pontnak a helyét a világkoordináta rendszerben a munkatéren. Ez volt a direkt kinematikai feladat megoldása, és ehhez használtuk az $a_i(q)$ függvényt. Vagyis az input a q konfiguráció volt, az output pedig egy $\left[a_i^{x_1}(q), a_i^{x_2}(q), \ldots\ a_i^{x_m}(q)\right]$ vektor volt.

Most meg szeretnénk mondani, hogy ha a q kicsit megváltozik, akkor milyen sebességvektor fog tartozni ehhez a megváltozáshoz. Ezt neveztük direkt sebesség kinematikai feladatnak. Itt feltettük, hogy q megváltozása egy paramétertől függ.



17. ábra: A t paramétertől függő q(t) út megváltozását a q'(t) írja le

Emlékezzünk, hogy a q koordinátáiban θ_i szögek illetve l_i hosszok vannak, attól függően, hogy az adott csukló, amihez a koordináta tartozik, forgó vagy prizmatikus. Ha $q(t)$ alakban írjuk fel a konfigurációt, akkor az azt jelenti, hogy ezek a koordináták is függnek t-től, azaz:

$$q(t) = (\,\theta_1(t), l_2(t), l_3(t), \ldots \theta_n(t)\,) \tag{33}$$

Ekkor ennek a deriváltja:

$$q'(t) = (\,\theta_1{}'(t), l_2{}'(t), l_3{}'(t), \ldots \theta_n{}'(t)\,) \tag{34}$$

A koordináták a csuklók paramétereinek megváltozásának sebességét írják le.

Most nézzük meg, hogy mi lesz a deriváltja például az $a_i^{x_1}(q(t))$-nek ekkor. Összetett függvényt deriválunk, vagyis össze fogjuk adni a belső és külső függvények deriváltjainak szorzatát:

$$a_i^{x_1}{}'\big(q(t)\big) = \delta_1 a_i^{x_1}{}'\big(q(t)\big) * \theta_1^{\prime(t)} + \delta_2 a_i^{x_1}{}'\big(q(t)\big) * l_1^{\prime(t)} \ldots + \delta_n a_i^{x_1}{}'\big(q(t)\big) * \theta_n^{\prime(t)} \tag{35}$$

Vagyis a külső függvény deriváltjai a parciális deriváltak, a belső pedig a $q'(t)$ koordinátái, amit jelöltünk δq-val is. Ha a munkaterünk m dimenziós, a konfigurációs tér pedig n, akkor az elmozdulás felírásához használhatunk egy $m \times n$-es Jacobi mátrixot, amit J-vel fogunk jelölni. Az k-dik sorban az k-dik koordinátafüggvény parciális deriváltjai vannak, 1-től n-ig. Ezt a mátrixot kell megszoroznunk tehát δq-val, hogy megtudjuk, mekkora volt az elmozdulás a munkatérben:

$$J\,\delta q = \delta \underline{\mathrm{x}} \tag{36}$$

Ha ezt visszahelyettesítjük a virtuális munka tétele által adott egyenletbe, megszabadulhatunk $\delta \underline{\mathrm{x}}$-től:

$$\mathrm{F}\,J\delta q = \underline{\mathrm{F}}\,\delta q \tag{37}$$

Mindkét oldalon egy mátrixot szorzunk egy vektorral, mégpedig ugyanazzal. Az előbbi gondolatmenetből látszik, hogy ez az egyenlőség minden δq esetén fennáll, amiből az következik, hogy a két mátrix ugyanaz. Jelen esetben mindkét mátrix egy sorvektor lesz.

$$\mathrm{F}\,J = \underline{\mathrm{F}} \tag{38}$$

Ezzel pedig kijött, hogy kapcsolatot tudok teremteni a két erő között. Ha megvannak a valódi munkatérbeli erőim (amiket korábban már ki tudtam számolni), kiszámolhatóak a konfigurációs térben indukálódott erők is.

Ezt tudván már meg tudjuk mondani, hogy fogják a fellépő erők elmozdítani a robotkart. Az eljárás a következő: az összes kontrollpontnál fellépő eredő erőt egyesével számoljuk át a konfigurációs térbe a q ponthoz. Tehát ha volt k darab kontrollpontom, akkor k darab erő fog hatni a q konfigurációs pontra. Ezután pedig vegyük ezeknek az eredőjét, és ezzel az erővel mozgassuk el egy picit a konfigurációs térbeli pontot. Ez adja meg, hogy a munkatérben a robotkarunk hogyan fog elmozdulni egy rövid idő alatt (azt, hogy mennyit mozgassuk, vagy mennyi ideig, most nem határozzuk meg). Amikor elértünk egy új q′ konfigurációba, akkor pedig a korábban használt a_i függvényekkel fejtsük vissza, mi az a robotkar új állása a munkatérben.

Összefoglalva, mi a munkatéren csak kiszámoljuk a ható erőket. Ezeket az erőket átfordítjuk virtuális erőkké, amik a konfigurációs térben hatnak. Ott megnézem, milyen új konfigurációba kerül a robotkar, és ezt ismervén visszajövök a munkatérbe, ahol az új konfiguráció szerint beállítjuk a robotkart.

3.1.5. Példa a számolásra

Vegyünk egy RR robotkart a síkon. A két kontrollpontunk a szegmensek középpontjai, hasonlóan, mint a 2.6.1-es fejezetben. Nézzük a második kontrollpontot. Már kiszámoltuk a világkoordináta rendszerben adott koordinátáit, ezért használhatjuk azt. Jelöljük a t-től való függést is, a következőképpen:

$$\begin{bmatrix} d_1 \cos\theta_1(\mathrm{t}) + \frac{d_2}{2}\cos(\theta_1(t)+\theta_2(t)) \\ d_1 \sin\theta_1(\mathrm{t}) + \frac{d_2}{2}\sin(\theta_1(\mathrm{t})+\theta_2(t)) \end{bmatrix} = a_2(q(t)) \tag{39}$$

Számoljuk ki a Jacobi mátrixot. Az első sorában először θ_1, aztán θ_2 szerint deriváljuk az $a_2(q(t))$ első koordinátájának, majd a második sorban a második koordinátát ugyanígy:

$$J_{a_2} = \begin{bmatrix} -d_1 \sin\theta_1(\mathrm{t}) - \frac{d_2}{2}\sin(\theta_1(\mathrm{t})+\theta_2(\mathrm{t})) & -\frac{d_2}{2}\sin(\theta_1(\mathrm{t})+\theta_2(\mathrm{t})) \\ d_1 \cos\theta_1(\mathrm{t}) + \frac{d_2}{2}\cos(\theta_1(\mathrm{t})+\theta_2(\mathrm{t})) & \frac{d_2}{2}\cos(\theta_1(\mathrm{t})+\theta_2(\mathrm{t})) \end{bmatrix} \tag{40}$$

És hogy ha a munkatéren ható erőt felírjuk $F = [F_x \quad F_y]$ alakban, akkor a virtuális erő a korábbi képlet alapján:

$$F = F J_{a_2} =$$

$$= [F_x \quad F_y] \begin{bmatrix} -d_1 \sin\theta_1(\mathrm{t}) - \frac{d_2}{2}\sin(\theta_1(t)+\theta_2(t)) & -\frac{d_2}{2}\sin(\theta_1(t)+\theta_2(t)) \\ d_1 \cos\theta_1(\mathrm{t}) + \frac{d_2}{2}\cos(\theta_1(t)+\theta_2(t)) & \frac{d_2}{2}\cos(\theta_1(\mathrm{t})+\theta_2(t)) \end{bmatrix} \tag{41}$$

A szorzás eredménye egy sorvektor, amit az átláthatóság miatt egy oszlopvektor transzponáltjaként írunk most fel:

$$\begin{bmatrix} -F_x\left(d_1 \sin\theta_1(\mathrm{t}) + \frac{d_2}{2}\sin(\theta_1(\mathrm{t})+\theta_2(\mathrm{t}))\right) + F_y\left(d_1 \cos\theta_1(\mathrm{t}) + \frac{d_2}{2}\cos(\theta_1(\mathrm{t})+\theta_2(\mathrm{t}))\right) \\ -F_x\left(\frac{d_2}{2}\sin(\theta_1(\mathrm{t})+\theta_2(\mathrm{t}))\right) + F_y\frac{d_2}{2}\cos(\theta_1(\mathrm{t})+\theta_2(\mathrm{t})) \end{bmatrix}$$

3.2. A gradiens módszer lépései

Az eddigiekben láthattuk, hogyan tudunk megalkotni egy potenciál mezőt, amiben a robotkarunk a kívánt cél fele fog haladni. Továbbá láttuk, hogyan kell kiszámolni a munkatérben fellépő erők és az indukált virtuális erők közti kapcsolatot. Most adunk egy

pszeudo-kódot, ami leírja, hogyan lehet ennek segítségével egy q konfigurációkból álló sorozatot megadni, ami mentén eljuthatunk q_{init}-ből q_final-be:

1. q_0:= q_{init}; i:= 0
2. HA $q_i \neq q_{final}$

   q_{i+1} := $q_i + \alpha_i * \frac{F(q_i)}{||F(q_i)||}$

   i:= i + 1

   KÜLÖNBEN return <q_0, q_1, …. q_i>
3. GO TO 2

A kódban szereplő $F(q_i)$ az az F, ami a konfigurációs térben hat a q_i pontra, aminek kiszámolásához szükséges lépéseket levezettük. Azért osztunk le a normájával, hogy egység hosszú vektort kapjunk. Az α_i skalár felel a lépéseink hosszáért. Különböző heurisztikák léteznek ennek a jó megválasztásához. Lehet úgy megválasztani például, hogy ha a kontrollpontok messze vannak az akadályoktól, ugorhatunk nagyobbat, mert biztonságos területen maradunk úgy is. Viszont ha közel vannak akadályokat, inkább kisebbeket lépjünk, hogy elkerülhessük őket. Persze lehet α_i végig egységesen ugyanaz is, azzal is találhatunk megoldást.

4. Valószínűségi roadmap módszer (PRM)

Ebben a fejezetben bemutatunk egy másik fajta megközelítést a probléma megoldásához. Célunk olyan módszer használata, ami során nem kell minden egyes új lekérdezés esetén újra felfedezni Q_{free}-t, hanem már korábbról megismert utakat használhatunk ismét. Ez hasznos, ha a robotkarunk hosszabb ideig fog ugyanazon a munkatéren. Az egyes lépések vizsgálatakor nagyban támaszkodunk az [1] forrás javasolt megoldásaira.

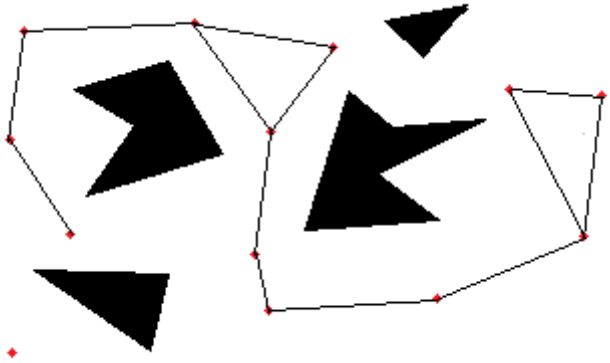
A probabilistic roadmap method (továbbiakban PRM) egy mintavételen alapuló útvonaltervező eljárás. Megvalósításához két feltételezést kell tennünk. Az első, hogy létezik egy ütközés-ellenőrző algoritmusunk. Ennek az inputja konfigurációs térbeli elemek, az outputja pedig 1, ha az input a Q_{free}-nek eleme, és 0 különben.

A másik feltétel, hogy adott egy úgynevezett lokális tervező, aminek inputja két konfigurációtérbeli pont, az algoritmus pedig visszatér egy egyszerű, nem feltétlen optimális úttal a 2 pont között. Két dimenzióban ez legtöbbször a 2 pontot összekötő szakasz.

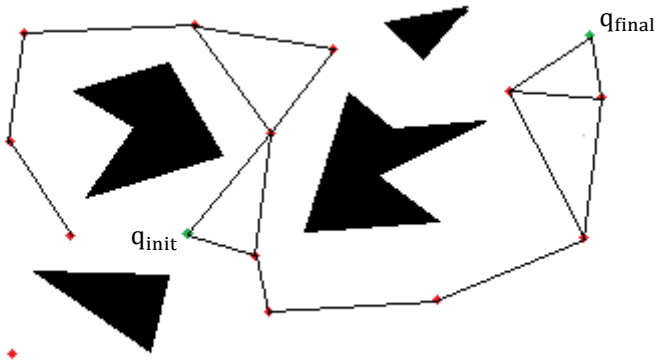
Az úgynevezett roadmap-et pedig így alkotjuk meg:

1) Vegyünk a konfigurációs térből mintát, N darab pontot, nevezzük ezeket mérföldköveknek. A kezdeti számukat mi választhatjuk, felesleges túl sokat választani, mert ha kevés lesz, akkor a későbbiekben veszünk majd még fel újakat.
2) Minden mérföldkőre futtassuk le az ütközés-ellenőrzést. Ahol az input 1 volt, azokat a pontokat megtarthatjuk, a többit vessük el.
3) A megmaradt mérföldköveket próbáljuk meg összekötni egymással a lokális tervezőt használva.
4) Szükség esetén beiktatunk egy javító fázist, erről részletesebben később

18. ábra: A piros pontok a véletlen választott minta a konfigurációs térből. Az ütközéstesztek után csak azok a pontok maradnak meg, amik a fehérrel jelzett Q_{free}területén vannak.



19. ábra: Az eljárás harmadik lépése, megpróbáltuk összekötni egymással a közeli konfigurációkat



20. ábra: Lekérdezéskor felvettük a két végpontot, aztán bekötöttük őket a roadmapbe, majd utat keresünk köztük a mérföldköveken keresztül

Ezt a roadmap-et felhasználva az útkeresési feladatot az alábbi módon oldjuk meg: Vegyük fel a konfigurációs térben q_{init}-et és q_{final}-t. A lokális tervező segítségével ezeket a pontokat kössük hozzá a roadmap-hez. Most pedig keressünk utat q_{init}-ből q_{final}-ba, a mérföldköveken lépkedve. Ez utóbbi megvalósítható egy gráfkereséssel, ahol a csúcsok a mérföldkövek és a 2 hozzávett pont, valamint az élek a lokális tervezővel adott összekötő utak. Ezzel az eljárással az eredeti útkeresési problémát visszavezettük arra, hogy a 2 pontot bekössük a roadmap-be, és azon belül találjunk egy utat a két pont között. Ez pedig általában sokkal egyszerűbb feladat.

Vegyük végig, milyen problémák léphetnek fel a PRM alkalmazása során, hogyan tudjuk őket megoldani, valamint mik a módszer előnyei és hátrányai.

4.1. Mintavétel

A legegyszerűbb módja a mintavételnek, ha egyenletes eloszlás szerint választjuk a pontokat. Ennek viszont következménye, hogy a minták száma a Q_{free} egy adott területén arányos a terület térfogatával. Emiatt kicsi a valószínűsége, hogy mérföldkő kerül az akadályok közti szűkösebb területekre. Ezt a jelenséget narrow passage problémának nevezik az irodalomban. Általában úgy oldják meg, hogy vagy más eloszlás szerint vesznek mintát, vagy beiktatnak egy javító fázist a roadmap kiépítésébe. Erről részletesebben a 4.3-as fejezetben lesz szó.

4.2. A pontpárok összekötése

Több megközelítést is használhatunk a pontok összekötéséhez. A legelterjedtebb módszerek a következők: összekötünk minden pontot minden ponttal; mindenkit összekötünk az ő k darab

legközelebbi szomszédjával; minden olyan pontot összekötünk, amik legfeljebb egy adott R távolságra vannak egymástól. Az első eljárás a legidőigényesebb, sok pont esetén nem tanácsos azt használni. Ha meg szeretnénk határozni, hogy egy pontnak melyik a k legközelebbi szomszédja, szükségünk lesz távolságfüggvényre. Az alábbi táblázat összefoglalja a két leggyakrabban használt normát a konfigurációs térben:

2 norma	$\| q' - q \| = \left[\sum_{i=1}^{n} (q_i' - q_i)^2\right]^{\frac{1}{2}}$
végtelen norma	$\max_n \lvert q_i' - q_i \rvert$

A táblázatban feltettük, hogy n darab csukló van, a két konfiguráció q és q', és q_i jelöli a i-dik csukló konfigurációját. Ahogy említettük, a legegyszerűbb lokális tervező csak összeköti a két konfigurációt egy egyenes vonallal. Ezután ellenőrzi, ez az út ütközik-e akadállyal. Ezt megteheti úgy, hogy az egyenesen felvesz „elég sűrűn" pontokat, és ezekre lefuttatja az ütközés-ellenőrzést. Ha egyik pont se ütközik, akkor az utat elfogadhatjuk, bevehetjük a roadmapbe.

4.3. Javító fázis

Miután összekötöttük azokat a pontokat, amiket lehetett, elég valószínű, hogy több komponensünk lesz. Ilyenkor nem tudunk mindig utat találni, ezért össze kell valahogy kötnünk a komponenseket. Általában ilyenkor megpróbáljuk megkeresni a legnagyobb komponenst, és megpróbáljuk hozzákötni a többit. A kisebb komponensekből megkeressük a legközelebbi pontot a nagy komponenshez, és megpróbáljuk összekötni valamelyik pontjával. Esetleg új pontokat is felvehetünk a konfigurációs téren, és megismételhetjük a kötögetést, bízva benne, hogy ezzel pár komponenst sikerül egyesíteni.

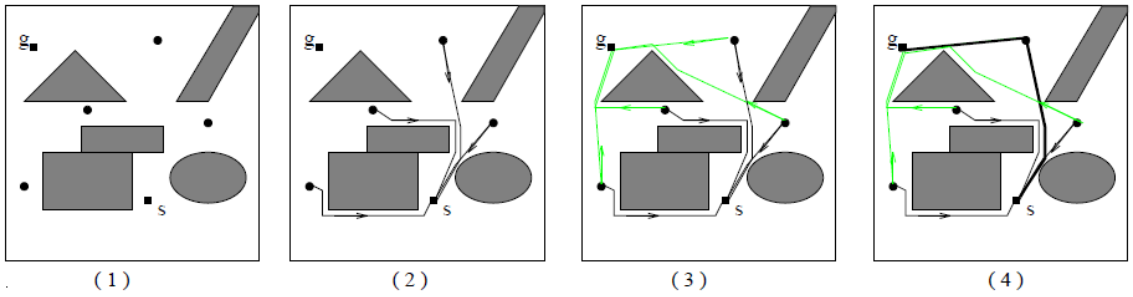
4.4. Megjegyzés a PRM használatáról

A PRM-nek nagy előnye, hogy könnyen használható akár magasabb dimenziójú konfigurációs térben is. Továbbá ha többször is lekérdezünk, minden feladatnál egyre jobb lesz a roadmap, abban az értelemben, hogy a már korábbi q_{init}-ek és q_{final}-ok is be vannak kötve a roadmapbe, így később lehet, hogy gyorsabban találunk megfelelő utakat. Vagyis ha tudjuk, hogy sokszor fogunk utat keresni ugyanabban a térben, hasznos lehet PRM-et használni.

5. Párhuzamos keresés

Az alábbi módszert Caigong Qin és Dominik Henrich dolgozta ki, akiknek célja a megoldás megtalálásának felgyorsítása volt. Az eredeti cikk megtalálható az irodalomjegyzékben, valamint a 21-es ábra is az említett cikkből származik.

Eddigiekhez hasonlóan adott q_{init} és q_{final} , keresünk egy konfigurációs pontokból állós sorozatot, ami elvezeti a robot egyikből a másikba. Most a számítások felgyorsítására párhuzamosan fogunk utakat keresni, a következőképpen: egy sor útkereső algoritmus próbálja összekötni q_{init}-et és q_{final}-t, mindegyik egy-egy különböző köztes ponton keresztül. Ezeket a pontokat maguknak generálják a folyamatok. Az egyes algoritmusok az általuk generált köztes pontot próbálják meg összekötni a kezdőponttal. Ha azzal végeztek, keresnek még egy utat, ezúttal a végpontba. Az output a két út megfelelő konkatenációja (összekötése). Amikor az első algoritmus (processz szál) sikeresen visszaad egy ilyen utat, az összes többi algoritmus (többi processz szál) is leáll.



21. ábra: A párhuzamos keresés lépései.

Az ábra első részén azt látjuk, hogy az eljárások véletlenszerű pontokat generáltak a konfigurációs térben. A másodikban ezek az algoritmusok elkezdik keresni az utat az s-el jelölt q_{final} pontba. A harmadikban ugyanezt teszik q_{init} felé. Végül az első algoritmus, ami talál egy helyes utat, leállítja az összes többi keresést, és az ő megoldását fogadjuk el. Az egyes algoritmusok a saját maguk által generált pontot próbálják meg összekötni a kezdő és végponttal, se nem használják, se nem ismerik a többi futó processz szál köztes pontját.

A módszerben a keresések a generált pontból indulnak, nem pedig valamelyik adott pontból. Ennek oka, hogy ha a vég vagy a kezdőpont egy lokális minimumban van, nagyon nehéz onnan kiszökni, még a korábban ismertetett módszer használatával is. Ebben az esetben, ha

különböző köztes pontokból indítjuk a kereséseket, megkönnyíthetjük a keresést. Különösen, hogy párhuzamosan több keresés is fut.

Könnyen látható, hogy egy így megkapott út általában nem lesz optimális, akkor se, ha a részutak maguk optimálisak. De ha elég sok párhuzamos számítást végzünk egyszerre, akkor a kapott eredmény tart a legjobb megoldáshoz, állítják a szerzők a [2] cikkben.

Ha kevés az akadály a konfigurációs téren, lehet jobban járnánk, ha ehelyett a megközelítés helyett magukat a megadott pontokat próbálnánk összekötni. Ezért szoktak még két processzt berakni a sorba, az egyik a végpontból keres utat a kezdőbe, a másik fordítva. Ebből látható, hogy egy darab útkereséssel megoldani a problémát nem gyorsabb, mint egyszerre többet használni.

6. Trajektória tervezés

Az alapfeladatunk továbbra is az, hogy keressünk egy utat a q_{init} kezdeti konfigurációból egy kívánt q_{final} végállapotba. Az út egy τ folytonos függvény lesz, ami a [0,1] intervallumból a Q konfigurációs térbe, továbbá teljesíti az alábbi két feltételt:

$$\tau(0)=q_{init} \tag{42}$$

$$\tau(1)= q_{final} \tag{43}$$

Ezt a feladatot úgy is felírhatjuk, hogy a használt paraméterünk az idő, amit t-vel jelölünk. Ez esetben nem útnak, hanem pályának nevezzük a függvényt, és nem τ-val jelöljük, hanem q-val. Ilyenkor megadjuk a t_0-lal jelzett kezdeti időt, és a t_f-el jelölt befejezési időt.

$$q(t_0) = q_{init} \tag{44}$$

$$q(t_f) = q_{final} \tag{45}$$

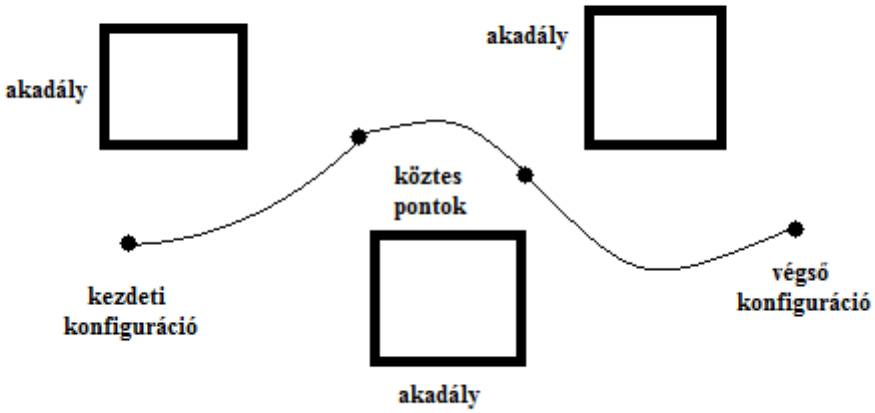
Ilyenkor fontos, hogy $t_f - t_0$ pozitív legyen (úgy is gondolhatunk erre, hogy a pálya megtételéhez időre lesz szükségünk).

A kapcsolat a két megközelítés között az, hogy az útra (τ) gondolhatunk úgy, mint egy olyan pálya (q), amin egy egységnyi idő alatt fogunk végigmenni (bármi is legyen az időegység).

Egyik oka, amiért érdemes az idővel parametrizálnunk, mert ilyenkor a függvény deriválásával megkaphatjuk a sebességet, még egyszeri deriválás után pedig a gyorsulást is. Ezek a sebességek és gyorsulások a konfigurációs téren vannak értelmezve, vagyis nem a robot munkatérbeli gyorsulását adják meg. Annak a kiszámolásához használhatjuk a Denavit-

Hartenberg konvencióval kapott direkt kinematikai feladatokat. Korábban a 3.1.4-es fejezetben már láthattuk, hogy a sebességet ki tudjuk számolni, a gyorsulást hasonlóan kell. Ezt kihasználva ezekre is megadhatunk feltételeket, amiket elvárunk, hogy a kapott pálya kielégítsen.

Az eddigiekben láthattuk, hogy egy úttervező algoritmus általában nem a τ útfüggvényt adja meg nekünk, hanem konfigurációs pontok egy sorozatát az út mentén.



22. ábra: Példa egy mozgás tervezésére köztes pontok segítségével

Ezen az úton végig tudunk úgy menni, hogy az egymást követő q_i tagokat egyenes szakaszokkal összekötjük. Használva a korábban bevezetett $a_i(q)$ függvényt, meg tudjuk mondani, hogy adott konfigurációban hol kell lennie a robot pontjainak. A probléma az lehet, hogy ha a q(t) függvény nem elég sima, akkor a robot mozgása a munkatérben döcögős lesz. Hirtelen fog megállni, majd teljesen más irányba elindulni, mint amerre eddig ment.

Ez azt jelenti, hogy szeretnénk ezt az utat elsimítani, hogy elkerüljük az ilyen rándulásokat a robotkarnál. Ha a q(t) egyszeresen folytonosan differenciálható, akkor azzal a sebességét simítjuk, ha pedig kétszeresen folytonosan differenciálható, akkor a gyorsulását is. Ennél magasabb rendű deriváltakat nem szoktunk szabályozni, mert a gyakorlatban annak a hatását már nem szoktuk érzékelni.

Ez a feladat gyakori a matematika különböző területein, sokféle megoldás létezik rá. Megtehetjük, hogy polinomokat illesztünk a szomszédos köztes pontokra, úgy hogy ezeknek a

polinomoknak a deriváltjaik, és a második deriváltjaik is illeszkedjenek. Ehhez használhatunk Bezier görbéket, megadva, hányad rendben kell illeszkedniük a polinomoknak.

7. Útkeresés láthatósági gráf segítségével

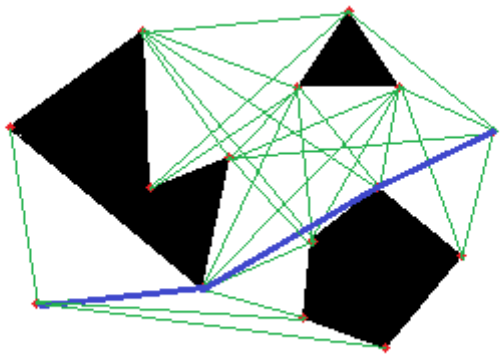
A láthatósági gráf, angolul visibility graph egy elterjedt modell a számítástudomány geometriával foglalkozó területein, és a robotok úttervezésében egyaránt. A módszert Nilsson dolgozta ki 1969-ben.

Feltesszük, hogy az akadályok politóp alakúak a konfigurációs térben (ez feltehető, ha politópok uniójával közelítjük őket). A gráfunk úgy keletkezik, hogy az akadályok összes csúcsát tekintjük a gráf csúcsainak,valamint a q_{init} és q_{final} csúcsokat. Két csúcsot pedig akkor kötünk össze, ha „a csúcsok látják egymást”, formalizálva:

$$V = \{q_{init}, q_{final}\} \cup \{v \mid v \text{ a QO egy csúcsa}\} \quad (46)$$

$$E = \{(u,v) \mid (1-t)u + tv \in Q_{free}, u, v \in V, t \in [0,1]\} \quad (47)$$

Mivel az akadályok politópok, annak az ellenőrzése, hogy egy csúcsokat összekötő szakasz elmetsz-e egy akadályt, gyorsan és könnyen számítható. Miután megvan a konstrukciónk, a Dijkstra algoritmus segítségével keresünk utat q_{init}-ből q_{final}-be. Szükség szerint akár az akadályok éleit is behúzhatjuk élként a gráfban is.



23. ábra Pirossal a csúcsokat, zölddel a gráf éleit, kékkel pedig a legrövidebb utat jelöltük

Ez az algoritmus teljes, vagyis ha létezik megoldás, akkor vissza fog térni eggyel, ha pedig nem, akkor hibát jelez, és leáll. Ez megtörténhet, például ha az egyik pontot teljesen körbeveszi egy akadály. Továbbá bizonyítható, hogy ez az algoritmus a 2 pont közötti legrövidebb utat találja meg.

Mivel valószínűleg nem túl hasznos, ha az út ennyire közel helyezkedik el az akadályokhoz, a gyakorlatban felteszik, hogy a robot nagyobb, mint igazából, és így számolják ki Q_{free}-t. Más szavakkal az akadályok megvastagodnak a konfigurációs térben. Így marad egy biztonságos távolság a robot és az akadályok között.

A módszer hátránya, hogy egy elég erős feltételt is megkövetel, mégpedig hogy explicit ismerjük a konfigurációs teret, és benne az akadályokat. Ez pedig, mint korábban láthattuk, sokszor nem igaz.

8. Összefoglalás

A dolgozat során különböző megoldási módszereket kerestünk egy robotkar útkeresési problémájának megoldására. Részletesebben végigvettük a gradiens módszert, melynek megalkotásához szükségünk volt a konfigurációs tér bevezetésére, a Denavit-Hartenberg konvencióra, és a virtuális munka elvére.

A részletes számolások után átvettünk egyéb módszereket is, ahol kevesebb számítással, valamint más megközelítésekkel szintén meg tudtuk oldani a feladatot, például hogy visszavezettük egy gráfban való útkeresésre. Valamint röviden megnéztük, miért lehet fontos szabályozni a robotkar sebességét illetve gyorsulását, és hogy ezt hogyan kell megtennünk, ha adottak a pontjaink, melyeket utunk során be kell járnunk.

9. Felhasznált irodalom

Szakirodalom és szakcikkek:

[1] Mark W. Spong, Seth Hutchinson, és M. Vidyasagar: Robot Modeling and Control

[2] Caigong Qin és Dominik Henrich: Path planning for indutrial robot arms

[3] John Reif és Micha Sharir: Motion Planning in the presence of moving obstacles: https://users.cs.duke.edu/~reif/paper/sharir/movingrobot.pdf

Egyéb források:

[4] CS 294-115 Algorithmic Human-Robot Interaction Lecture 3 and 4, Fall 2016 (by Molly Nicholas, Chelsea Zhang, Xinlei Pan and Daniel Seita)

Felhasznált ábrák:

[5] A 2. ábra forrása Dana Mackenzie Shape Shifters Tread a DauntingPath Toward Reality című cikke: https://www.isi.edu/robots/press/science123.pdf

[6] A 7. ábra a [3] forrásból származik.

[7] A 21-es ábra a [3] forrásból származik