Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Pályatervezési és mozgásirányítási algoritmusok fejlesztése mobil robotokhoz

Készítette
Csorvási Gábor, Nagy Ákos

Konzulens
Kiss Domokos

2014. október 19.

Tartalomjegyzék

Kivonat

A mobil robotok manapság egyre inkább feltörekvőben vannak. Már nem csak az ipar fedezi fel őket, hanem lassan a mindennapi életünk részévé válnak. Azonban még rengeteg elméleti és gyakorlati kérdés vár megoldásra, hogy az ilyen robotokkal rendszeresen találkozzunk. A mobil robotika egyik legalapvetőbb kérdése az akadályok jelenlétében történő mozgástervezés és mozgásvégrehajtás. A dolgozatban ezt a kérdéskört járjuk körül, foglalkozunk a globális és lokális geometriai pályatervezéssel, pályamenti sebességprofil kialakításával, valamint pályakövető szabályozással. Ezeket két, síkban mozgó, kerekeken guruló robotmodellre alkalmazzuk, mint szimulált, mint valós környezetben.

A dolgozatban bemutatjuk a leggyakrabban használt pályatervezési algoritmusokat, és az ezekhez kapcsolódó előnyöket és problémákat. Külön kitérünk az általunk vizsgált (differenciális és autószerű) robotmodelleknél felmerülő kinematikai korlátozásokra, és ezek hatásaira a pályatervezésben. Egy approximációs pályatervezési megközelítést mutatunk be a dolgozatunkban, amely egy globális és egy lokális tervező algoritmus együttes használatán alapszik.

Az általunk alkalmazott RTR (Rotate-Translate-Rotate) globális tervező a szakirodalomból jól ismert RRT (Rapidly Exploring Random Trees) eljáráson alapul. Az RTR lényege, hogy a kiindulási és a cél konfigurációból két topológiai fát épít, és amennyiben ezek elérik egymást, a keresett pálya könnyedén előállítható. A pálya forgásokból (R) és transzlációs mozgásokból (T) áll, így differenciális robotok számára közvetlenül is végrehajtható pályát eredményez. További lényeges tulajdonsága, hogy figyelembe veszi a robot pontos alakját. Ez hatékony tervezést tesz lehetővé szűk folyosókat tartalmazó környezet esetén is, szemben az elterjedtebb, a robot alakját körrel helyettesítő módszerekkel.

A megtervezett geometriai pálya még nem tartalmaz információt a mozgás időparaméterezésére (a robot sebességére, gyorsulására vagy szögsebességére) nézve. Ezért bemutatunk egy általunk kifejlesztett algoritmust a pályamenti sebességprofil meghatározására. Ezt a profilt a robot maximális sebessége, maximális gyorsulása, maximális szögsebessége és a robot kerekeinek maximális gyorsulása alapján számoljuk ki. Az így kialakuló pályát ezután újramintavételezzük, hogy időben egyenletes mintavételű pálya álljon rendelkezésre pályakövető szabályozás számára.

A pályakövető algoritmus a robot pályamenti sebességét és a szögsebességét függetlenül szabályozza. A szétcsatolt rendszer sebesség és szögsebesség beavatkozó jeleit a robot kinematikai egyenletei alapján átalakítjuk keréksebesség beavatkozó jelekre. A sebességszabályozási kör a robot tényleges pozíciója alapján egy PD szabályozón keresztül korri-

gálja az előírt sebességprofilt. A szögsebesség-szabályozás egy mozgás közbeni orientáció korrekciót hajt végre, melynek alapját a robot későbbi előírt pozíciói képezik.

A fent leírt algoritmusokat differenciális robotmodellt feltételezve alakítottuk ki. A dolgozatban bemutatjuk azokat a módosításokat, illetve kiegészítéseket, amelyek lehetővé teszik a pályatervezést és követést autószerű (kormányzott) robotok esetében is. Ennek keretében bemutatjuk a C*CS lokális pályatervező algoritmust, amely az RTR algoritmussal együtt alkalmazva olyan pályát eredményez, amely figyelembe veszi az autó minimális fordulási sugarát.

Az algoritmusokat a V-REP robotszimulációs környezetben implementáltuk és teszteltük, majd működésüket két valós roboton is vizsgáltuk.

Abstract

The research and application of mobile robots is nowadays increasingly widespread. Beyond industrial applications they are getting popular in personal usage as well. However, there are a lot of theoretical and practical issues to be resolved. One of the most fundamental aspects of mobile robotics is motion planning and control in environment populated with obstacles. In this paper we discuss global and local geometric path planning, velocity profile generation and motion control along the path. We simulated the investigated methods and tested with differential and car-like robots.

In this paper we present the most commonly used path planning algorithms, and their benefits and disadvantages. We discuss the kinematic constraints for the tested (differential and car-like) robot models and the consequences of these constraints. We present an approximation method for path planning, which is based on a global and a local planner algorithm.

We use the RTR (Rotate-Translate-Rotate) global path planner algorithm, which is based on the well-known RRT (Rapidly Exploring Random Trees) method. The RTR builds two search trees starting from the initial and the goal configuration. If these paths reach each other, the solution can be obtained easily. The path consists of rotation (R) and translation (T) motion primitives, so the path is directly applicable for differential drive robots. Furthermore, the RTR-planner takes the precise shape of the robot into account, which makes possible to find a path in an environment with narrow corridors. This is advantageous compared to other methods which assume circular robot shape.

The path generated by the RTR-planner does not contain any information about the robot's velocity, acceleration and angular velocity. Therefore, we present an algorithm developed to determine the velocity profile. The profile is based on parameters such as the robot's maximum velocity, maximum acceleration, maximum angular velocity along the path and the maximum acceleration of the robot's wheels. After velocity profile generation the geometric path needs to be re-sampled to obtain a path having uniform time-sampling for the motion controller.

The path following algorithm controls the robot translational and angular velocity in two independent control loops. The translational and angular velocity control signals of the decoupled system are converted to wheel velocity control signals based on the robot kinematic equations. The velocity control loop adjusts the velocity profile with a PD controller based on the robot current position. The angular velocity controller is based on orientation error between the current robot configuration and a future path point.

The algorithms described above are developed primarily for differential robots, but can be extended for car-like robots as well. In this context we introduce the C*CS local path planner method, which takes into account the minimal turning radius of the car and can be applied together with the RTR planner in order to obtain a feasible path for this robot class.

We have implemented and tested the algorithms in V-REP robot simulation framework and with real robots as well.

1. fejezet

Pályatervezés elmélete - 5 oldal CsG/NA

1.1. Megoldandó feladat

1.2. Lokális módszerek

1.3. Globális módszerek

2. fejezet

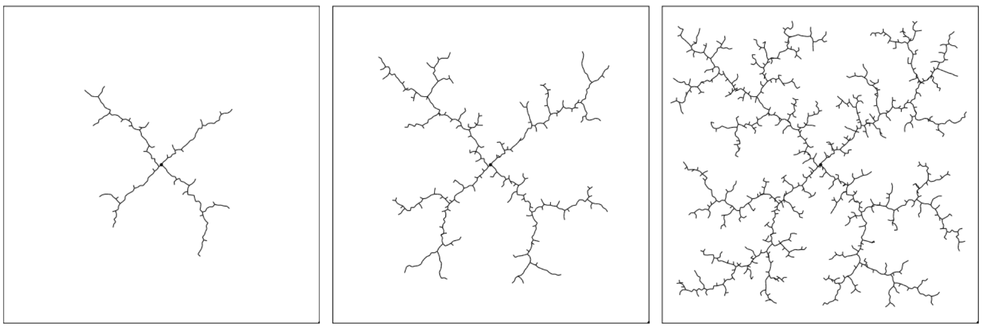
Az RTR algoritmus

Az RTR (Rotate-Translate-Rotate) algoritmust Kiss Domokos dolgozta ki [2]. A mi feladatunk az algoritmus implementálása volt C++ nyelven, majd az algoritmus tesztelése szimulációs környezetben és valós roboton.

Az algoritmus az irodalomból gyakran használt RRT algoritmuson alapszik, ezért ennek a bemutatásával kezdjük a fejezetet.

2.1. RRT algoritmus

Az RRT (Rapidly Exploring Random Trees) algoritmus lényege, hogy a kezdeti konfigurációból egy fát építünk a szabadon bejárható térben [4]. A fa terjesztését úgy irányítjuk, hogy a kívánt cél konfiguráció felé tartson. Ezután az utat a kezdeti konfigurációból a cél konfigurációba már könnyedén megkaphatjuk a fában. Létezik olyan változata az RRT algoritmusnak, ahol nemcsak a kezdeti konfigurációból építünk fát, hanem a cél konfigurációból vagy akár több köztes pontból is.



2.1. ábra. *Az RRT algoritmus három különböző iterációnál.*

A fa építés úgy kezdődik, hogy véletlen konfigurációkat veszünk a környezetből (q_{rand}). Ezt hívják mintavételezési szakasznak. Ezután meghatározzuk, hogy a fában melyik konfiguráció van a legközelebb a mintavételezett konfigurációhoz q_{near}. Ez a csomópont-választó szakasz. Előfordulhat, hogy nemcsak a fa csúcspontjában található konfigurációkat adjuk vissza, mint q_{near}, hanem a fa csúcspontjai közötti élek egy köztes konfigurációját.

A következő lépésben megpróbáljuk a q_{rand} és a q_{near} konfigurációkat interpolálással összekötni (összekötő szakasz). Itt több variációja is létezik az RRT-nek. Előfordul, hogy csak egy bizonyos fix Δq értékkel közelítünk q_{rand} felé q_{near} konfigurációból. A másik esetben addig terjesztjük a fát amíg el nem érjük q_rand konfigurációt vagy amíg nem ütközik a robot. Ebben az esetben a kapott konfiguráció a legmesszebb található ütközés mentes konfiguráció lesz q_rand irányában. Az újonnan kapott konfigurációt végül hozzáadjuk a fához.

Anholonom rendszerek esetén is használható az RRT eljárás. Ekkor az összekötésnél egyszerű interpolációt nem lehet alkalmazni, mert az azt feltételezné, hogy a robot minden irányba szabadon képes mozogni. Ehelyett az összeköttetést egy lokális tervező segítségével kell megoldani vagy egyszerűbb esetben itt is használható az a módszer, hogy csak egy adott Δq értékkel közelítünk q_{rand} irányába. Ehhez megfelelő beavatkozó jelet kell választanunk, amit Δt ideig alkalmazva elérhető Δq állapotváltozás.

2.2. RTR algoritmus

Differenciális robot esetében is használható ez előbb említett módszer, de ekkor a fa csúcspontjai között görbék lesznek. Ez nehézséget okozhat, ha olyan csomópont-választó eljárást alkalmazunk, ami köztes konfigurációt ad vissza. Természetesen alkalmazhatjuk azt az eljárást, hogy csak az élek végpontjait választjuk ki. Ehhez kis távolságú élek szükségesek, ami növeli a fa csomópontjainak számát és ezzel összefüggésben a csomópont kiválasztások számát is.

Differenciális robotnál is alkalmazhatunk lokális tervezőt két konfiguráció közti állapotváltozásra. A legegyszerűbb lokális tervező három lépésből áll:

- Egy helyben fordulás a kívánt konfiguráció irányába (R).
- Mozgás egyenes pályán a cél pozícióba (T).
- Egy helyben fordulás a cél konfiguráció irányába (R).

Ennek a tervezőnek az az előnye, hogy fa élei egyenes pályák lesznek, így egyszerűn tudjuk meghatározni a köztes konfigurációkat is és egzakt módon leírhatjuk a mozgásokat, nem kell mintavételezést alkalmaznunk.

A jelenleg ismertetett módon alkalmazva az algoritmust szűk folyosók esetében igen nehezen találna megoldást az eljárás. Abban az esetben is ha mind a kezdeti-, mind a célkonfigurációból növesztünk egy-egy fát. A problémát az okozza, hogy a összeköttetés fázisa gyakran nem ad eredményt és így a fák nem nőnek megfelelően. Ennek az az oka, hogy a lokális tervező esetében, fal vagy egyéb akadályok közelében az első egy helyben fordulásnál már ütközne a robot. Mivel az összeköttetés fázisa addig tart, amíg nem érjük el q_{rand}-ot vagy amíg nem ütközik a robot, így a fa további terjesztése nélkül választunk új q_{rand} értéket. A lokális tervező második lépése eredményezné a fa tényleges terjesztését.

Az RTR algoritmus felhasználva az RRT eljárás előnyös tulajdonságait igyekszik az előbbi problémára egy lehetséges megoldást bemutatni. Mind a kezdeti, mind a célkonfigurációból növeszt egy-egy fát, összeköttető fázisban a fent ismertetett Rotate-Translate-Rotate lokális tervezőt alkalmazza. Az RRT eljárás mind a három fázisa módosításra kerül az RTR tervező esetében.

Az RTR algoritmusnál alkalmazott fa struktúrában, az RRT-hez hasonlóan a csomópontokban konfigurációk találhatóak, az élek pedig transzlációs mozgást (TCI - Translation Configuration Interval) vagy egy helyben fordulást írnak le (Rotational Configuration Interval). Fontos megjegyezni, hogy mindkét éltípust egzakt módon írjuk le.

2.2.1. Mintavételezés

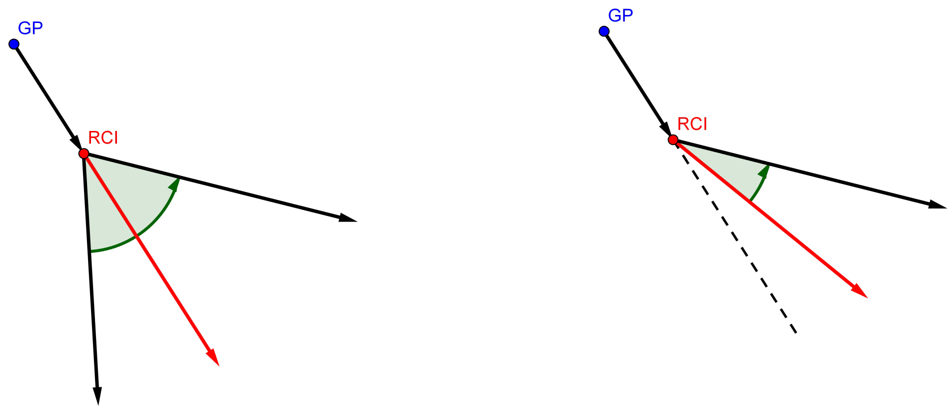
A mintavételezés fázisában különbséget jelent az eredeti RRT algoritmushoz képest, hogy nálunk a q_{rand}-nak megfelelő véletlen minta nem egy konfiguráció lesz, hanem egy pozíció a térben (GP - Guide Point). Ezt a pozíciót tekinthetjük egy folytonos, egy dimenziós konfigurációs listának, amelynek bármelyik eleme megfelelő cél konfigurációnak.

A mintavételezést kiegészíthetjük a pálya háromszög cellafelbontásából kapott mintákkal. Ezek a minták az akadályoktól távolabb találhatóak és szűk folyosók esetén is segítenek terjeszteni a fákat.

2.2.2. Csomópont-kiválasztás

Mivel a mintavételezésnél pozíciót használunk nem pedig konfigurációt, így a csomópont kiválasztás egyszerűbb lesz. Az eljárás lényege, hogy az adott GP esetén végigmegyünk a fa élein és meghatározzuk a legkisebb távolságát a GP és az adott él között. TCI él esetében ez távolságot jelent, RCI esetében pedig szögtávolságot. Így minden egyes élnél kapunk egy konfigurációt, amely esetében a távolság a GP-től minimális. A kapott konfigurációk közül azt választjuk amelynél legkisebb a távolság a GP-hez képest, ha így több megoldást is kapunk, akkor pedig azt a konfigurációt választjuk, amelynél a szögtávolság a legkisebb. Így egyértelműen meghatároztuk q_{near}-t.

Egy adott TCI és GP esetén a legkisebb távolsághoz tartozó konfigurációt a következőképpen határozzuk meg. Kiszámoljuk a GP vetületét a TCI-t alkotó egyenesre, ezután meghatározzuk, hogy a vetület a TCI-n, mint szakaszon belül található-e. Ha a szakaszon belül található, akkor egy köztes konfiguráció van a legközelebb a ponthoz és a köztes konfiguráció pozíciója a vetület, orientációja pedig a TCI orientációja. Ha a szakaszon kívül található a vetület, akkor a TCI közelebbi konfigurációja lesz a legkisebb távolságú konfiguráció.



2.2. ábra. *Csomópont-kiválasztás RCI esetén. A bal oldali ábra esetén köztes konfigurációt kapunk, míg a jobb oldali ábrán nem.*

RCI esetén a legkisebb (szög)távolságú konfiguráció kiválasztása a **??**. ábrán látható. Először kiszámoljuk a GP és az RCI pozíciójának irányát. Ha a kapott irány az RCI (szög)tartományába beleesik, akkor köztes konfigurációról van szó (bal oldali ábra) és ekkor a legközelebbi konfiguráció az RCI pozíciója és az előbb kiszámolt orientáció lesz. Abban az esetben mikor a kapott irány nem esik bele az RCI tartományába, akkor az RCI irányban közelebbi konfigurációt választjuk (jobb oldali ábra).

2.2.3. Kiterjesztés

2.2.4. Útvonal meghatározása, optimalizálás - 1 oldal

2.3. Eredmények - 3 oldal

3. fejezet

A C*CS, $\boldsymbol{c\bar{c}S}$ algoritmus - 10 oldal CsG

Működés Az általunk használt algoritmus egy approximációs módszert alkot, mely egy előzetes globális pályát rekurzív módon felbont kisebb szakaszokra, majd ezekre próbál illeszteni vagy két kör és egy egyenes kombinációját (CCS), vagy egy egyenes, egy körív, majd egy egyenes hármasát (SCS). Ahogy az (egyenlet)-nél beláttuk, az autó körpályán haladása során a kormányszög meghatározza a körpálya sugarát. Ha a kormányszöggel a nullához tartunk, akkor a sugár a végtelenbe tart, ami a gyakorlatban egy egyenest jelent, innen az algoritmus neve, C*CS, ahol a csillag jelzi a speciális körívet. Bár az algoritmus elsődlegesen autószerű robotok számára lett kitalálva, a megoldást egy differenciális robot is képes lekövetni, mivel az kevesebb korlátozással rendelkezik.

Globális pálya Az előzetes pálya bármilyen globális tervező eredménye lehet. Elsődleges célja egy mankó nyújtása a későbbi tervező számára, a megoldásnak nem feltétele, hogy az előzetes pálya akár egyetlen pontját is tartalmazza. Az egyik megoldás egy celladekompozíciós eljárás. Ez a környezet háromszögekre bontja, majd ezeknek a háromszögeknek az oldalfelező pontjait összeköti a szomszédos háromszögekkel. Az így kialakult gráfba beszúrja a kezdő és célkonfigurációt, majd hozzáveszi az őket körbevevő háromszög oldalfelező pontjait. Az éleket a pontok egymástól való távolságával súlyozzuk, majd ebben a gráfban keresünk egy legrövidebb utat. Ez a megoldás igen hasznos, mivel a szabad terület közepén alkot pályát, így ha az autó ezt a pályát követi, akkor bármilyen irányú manőverezésre nyújt lehetőséget. További előnye, hogy véges időn belül képes megmondani, hogy létezhet-e megoldás. Az eljárás egyik fő hibája, hogy csak sokszögekkel vagy más exakt módon leírható testekkel képes dolgozni, és még ebben a formájában nem veszi figyelembe az autó kiterjedését. Ezen könnyen lehet segíteni, ha figyelembe vesszük a kialakult szakaszok távolságát az akadályoktól, és ha a pálya- és az akadályél túl közel vannak egymáshoz, akkor töröljük az élt a gráfból.

Lokális pályák A továbbiakban az előzetes pálya két konfigurációját választjuk ki, és a fentebb említett pályákat keresünk köztük. Az eljárás először a pálya két végpontja közt keres megoldást, ami egyszerű esetekben akár rögtön megoldásra is vezethet, felgyorsítva az algoritmus működését. Ha ez a keresés nem járt sikerrel, akkor az előzetes pályát megfelezi az algoritmus és az első konfiguráció és az új célkonfiguráció közt keres megoldást. Ezt egészen addig ismétli, míg van köztes konfigurációs pont, ha elfogyott, további pontokat illeszt a pályába.

C*CS Bizonyítható [hivatkozás], hogy végtelen sok olyan megoldás található, mely körök és egyenesek hármasát használja fel. Ez akadályok jelenlétében előnyös tulajdonság, mivel így kiválasztható egy olyan megoldás, mely végrehajtható. Természetesen az összes megoldást nincs lehetőségünk kipróbálni, így valamilyen mintavételezéssel kell próbálkozzunk.

ARM A kezdő konfigurációból köríveken haladva elérhető pontokat hívjuk ARM-nak (Arc Reachable Manifold) Ha a környezetet felosztjuk egységnyi távolságokra, akkor kiszámítható véges sok lehetőséget kapunk. A kiszámítás ideje természetesen függ a választott távolság egységtől, és a környezet méretétől. Az eredmények azt mutatják, hogy az algoritmus futásának ez a leghosszabb része, ami nem meglepő, mivel a körívek kiszámítása komplex művelet, és ezt egy adott pont esetén a robot testének minden csúcsára ki kell számoljuk, majd ütközést kell detektáljunk. A művelet hatékonyságán lehet javítani, ha előre készítünk egy foglaltsági mátrixot, ami megmondja az adott pont akadályon beül van-e, így az ilyen pontokra nem kell a számítást elvégezni. Mivel az ARM egy adott kezdő-konfigurációhoz tartozik, további javítási lehetőség, ha a globális pálya felbontásakor inkább a célkonfiguráció pontját mozgatjuk, így nem kell újra és újra kiszámolni a köríven elérhető sokaságot.

CS Az algoritmus további részében az ARM-nál kiszámolt elérhető köztes konfigurációk és a célkonfiguráció közt kell meghatározni a lehetséges útvonalakat. Mivel a pálya minden esetben egy egyenes szakasszal végződik, így a célunk olyan további körök keresése, melyek érintik a köztes konfiguráció és a célkonfiguráció által meghatározott egyeneseket. Ilyen körökből egy köztes konfigurációhoz két megoldás is található, mindkét eset megfelelő megoldást nyújthat számunkra. Azonban figyelembe kell vegyük, hogy a két különböző körív mentén való haladás különböző végső orientációt okoz. Egy köztes szakasz számításakor ez nem jelent feltétlenül problémát, de a teljes pálya utolsó célkonfigurációnál el kell vessük az ellenkező irányt, mivel itt már nincs lehetőség megfordulásra. Bár a globális pálya tervezésekor ellenőriztük az egyenes szakaszok végrehajthatóságát, az érintő körök keresésekor nem volt feltétel, hogy az érintési pontok ezeken a szakaszokon helyezkedjenek el. Ezért ellenőriznünk kell az utolsó, egyenes szakasz végrehajthatóságát is.

Végül az így keletkező végrehajtható pályák közül ki kell választanunk egyet. Ezt többféle képpen megtehetjük, talán a legkézenfekvőbb a legrövidebb megoldás felhasználása. ide jön az indok miért Ezért a kiválasztott megoldásunk második köztes konfigurációs pontot be kell szúrjuk a globális pályába, majd innen folytathatjuk a szakaszok megoldását.

3.1. Reeds-Shepp lokális pályák - 1 oldal

3.2. C*CS lokális pályák - 6 oldal

3.2.1. ARM mintavételezés - 3 oldal

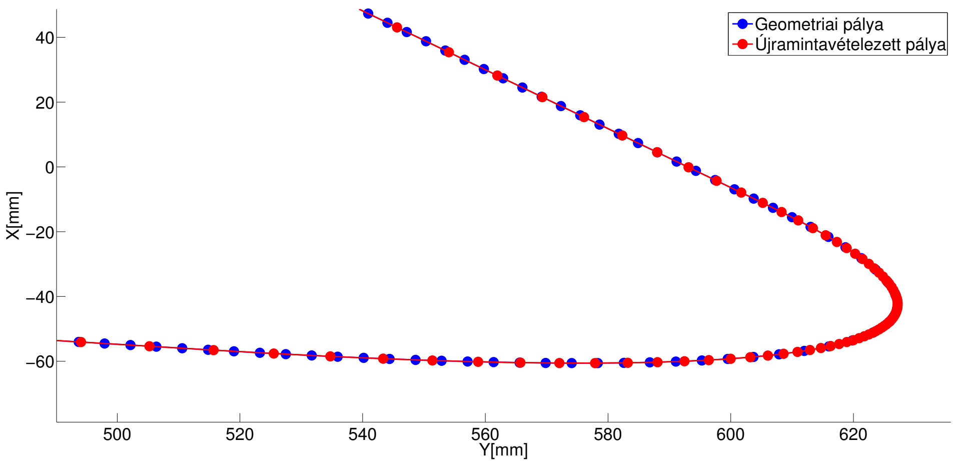
3.3. $c\bar{c}S$ - 3 oldal

4. fejezet

Pálya időparaméterezése

A pályatervező által elkészített ütközésmentes pálya nem tartalmaz semmilyen idővel kapcsolatos információt. Ebben a fejezetben a pálya pontjaihoz sebesség értékeket rendelünk hozzá. Ezt a többlet információt a pályakövető algoritmus használja fel, hogy mozgás során a robot kinematikai korlátai ne okozzanak problémát. Tehát az időparaméterezés elsősorban a robot korlátait használja fel, de arra is alkalmas, hogy meghatározzuk a pálya bejárásának idejét.

Az időparaméterezés két fő lépésből áll. Elsőként a kapott geometriai pályához sebesség értékeket rendelünk hozzá, majd ezután újramintavételezzük a pályát. Az újramintavételezés után a pálya időben egyenletes lesz, tehát az egymást követő pálya pontok között azonos idő telik el. A mintavételezés idejét a pályakövető algoritmus mintavételi ideje határozza meg. A geometriai pályát általában távolságban egyenletesen mintavételezzük, de ez nem szükséges az időparaméterezéshez.



4.1. ábra. *A pálya időparaméterezése.*

Az irodalomban nem sok időparaméterezéssel kapcsolatos munka található. Egy hasonló megközelítést Christoph Sprunk munkájában találhatunk [5]. A legfontosabb eltérés, hogy Sprunk külön korlátozza a robot tangenciális és centripetális gyorsulását, míg mi a robot kerekeinek eredő gyorsulását korlátozzuk. Ez a megoldás a valóságot jobban közelíti, hiszen attól, hogy a gyorsulás két komponense a korlátok alatt marad, nem biztos, hogy az eredő gyorsulás sem haladja a korlátot meg.

Az időparaméterezés során nem használjuk ki a pályatervező által tervezett pálya tulajdonságait, a célunk egy olyan algoritmus készítése, amely tetszőleges geometriai pályából képes sebesség információval ellátott, időben egyenletes mintavételű pályát készíteni. Emiatt nem építhetünk a pályatervező által használt geometriai elemekre (körív, egyenes) és ezek speciális tulajdonságaira.

Az egyik legalapvetőbb tulajdonságuk az lenne, hogy a görbületüket analitikusan ki tudnánk számolni (a konkrét elemeknek ráadásul triviális). Általános esetben azonban nem tudjuk a pálya görbületét analitikusan kiszámolni, így görbület becslést kell alkalmaznunk. Az irodalomban sok cikket találhatunk görbület becslésről, főleg képfeldolgozással kapcsolatos témákban, a mi dolgozatunknak azonban nem ez a témája. Az algoritmus fejlesztésekor több becslőt is kipróbáltunk, ezeket úgy teszteltük, hogy olyan pályát adtunk meg nekik, amelynek a görbülete analitikusan is számolható, így össze tudtuk hasonlítani az ideális megoldással a becslést. Ez alapján választottunk egy eljárást [3].Természetesen abban az esetben, ha a pályatervező rendelkezik már a pálya görbületével, az időparaméterező algoritmus azt fogja használni a becslés helyett.

4.1. Jelölések

Ebben a fejezetben a 4.1. táblázatban megadott jelöléseket fogjuk használni. Azokban az esetekben, ahol fontos megkülönböztetni a geometriai pályát és az (újra)mintavételezett pályát, ott a felső indexben található **g** betű a geometriai pályát jelöli, az **s** betű pedig a mintavételezett pályát. A pálya pontjait 1-től számozzuk.

$$\begin{aligned}
\Delta t(k) &: \text{A } k \text{ és a } k+1 \text{ pontok között eltelt idő} \\
t(k) &: \text{A } k\text{. pontban az addig eltelt idő} \\
\Delta s(k) &: \text{A } k \text{ és a } k+1 \text{ pontok között megtett távolságot} \\
s(k) &: \text{A } k\text{. pontban az addig megtett távolság} \\
v(k) &: \text{A } k\text{. pontban a robot sebességének nagysága} \\
\omega(k) &: \text{A } k\text{. pontban a robot szögsebességének nagysága} \\
a_t(k) &: \text{A } k\text{. pontban a robot tangenciális gyorsulásának nagysága} \\
c(k) &: \text{A } k\text{. pontban a görbület nagysága} \\
N &: \text{A pálya pontjainak száma}
\end{aligned} \tag{4.1}$$

Azokban az esetekben, amikor a robot kerekére vonatkozó mennyiségekről beszélünk, külön jelöljük, hogy bal (l) vagy jobb (r) kerékről van szó. Ezenkívül a kerekeknél megkülönböztetjük, hogy tangenciális (a_t), centripetális (a_c) vagy eredő (a_e) gyorsulásról beszélünk.

Fontos megjegyezni, hogy a $\Delta s(k)$ távolságot úgy kell értelmezni, hogy a k. és $k+1$. pont között egy körív található és az ezen mért távolság lesz $\Delta s(k)$. A körívet a $c(k)$ görbület határozza meg. Ha nem köríveket használnánk, hanem egyenessel kötnénk össze a pálya pontokat, akkor a görbületnek szükségszerűen 0-nak kellene lennie.

4.2. Differenciális robotmodell

Ebben a részben az időparaméterezést differenciális robotmodellhez készítjük el. A differenciális meghajtása a következő két kinematikai egyenlettel írható le [1]:

$$\begin{aligned} v(k) &= \frac{v_r(k) + v_l(k)}{2} \\ \omega(k) &= \frac{v_r(k) - v_l(k)}{W}, \end{aligned} \tag{4.2}$$

ahol W a robot kerekei közti távolság.

A 4.2. egyenletet átírhatjuk úgy, hogy a kerekek sebességeit fejezzük ki akár a szögsebesség, akár a pálya adott görbülete alapján:

$$\begin{aligned} v_l(k) &= v(k) - \frac{W \cdot \omega(k)}{2} = v(k) \cdot p_l(k) \\ v_r(k) &= v(k) + \frac{W \cdot \omega(k)}{2} = v(k) \cdot p_r(k) \\ p_l(k) &= 1 - \frac{W \cdot c(k)}{2} \\ p_r(k) &= 1 + \frac{W \cdot c(k)}{2}, \end{aligned} \tag{4.3}$$

ahol felhasználtuk, hogy $v(k) \cdot c(k) = \omega(k)$.

4.2.1. Korlátozások

A robot mozgását általános esetben a **??**. ábra mutatja be. Az időparaméterezés során figyelembe vesszük a robot pályamenti sebességét és szögsebességét és a robot kerekeinek tangenciális és eredő gyorsulását. Adott robot esetében ezekre a mennyiségekre határozunk meg korlátozásokat:

$$\begin{aligned} v^{max} &: \text{A robot pályamenti sebesség korlátja} \\ \omega^{max} &: \text{A robot szögsebesség korlátja} \\ {a_{lt}}^{max} &: \text{A robot bal kerekének tangenciális gyorsulás korlátja} \\ {a_{rt}}^{max} &: \text{A robot jobb kerekének tangenciális gyorsulás korlátja} \\ {a_l}^{max} &: \text{A robot bal kerekének eredő gyorsulás korlátja} \\ {a_r}^{max} &: \text{A robot jobb kerekének eredő gyorsulás korlátja} \end{aligned} \tag{4.4}$$

Mivel a robot kerekeinek tangenciális gyorsulásából már adódik a robot tangenciális gyorsulása is, így a robot gyorsulását nem szükséges külön korlátozni.

$$a_t^{max} = \frac{{a_{lt}}^{max} + {a_{rt}}^{max}}{2} \tag{4.5}$$

Ugyanez a helyzet a kerekek sebesség korlátjával, ami meghatározható a robot sebesség és szögsebesség korlátaiból.

$$v_l^{max} = v^{max} - \frac{W \cdot \omega^{max}}{2} \tag{4.6}$$

$$v_r^{max} = v^{max} + \frac{W \cdot \omega^{max}}{2} \tag{4.7}$$

A kerekek maximális eredő gyorsulását a maximális tapadási súrlódási együttható ($\mu_{tap_{max}}$) határozza meg, amelynél a robot kerekei még nem csúsznak meg. A maximális gyorsulás és a tapadási együttható között a következő egyszerű összefüggés áll fent:

$$a_{max} = \mu_{tap_{max}} \cdot g, \tag{4.8}$$

ahol g a nehézségi gyorsulás

Írjuk fel a **??**. ábra alapján a robot kerekeire ható erőket:

$$\sum F(k) = m \cdot a(k) = m \cdot \sqrt{a_c(k)^2 + a_t(k)^2} \leq m \cdot g \cdot \mu_{tap_{max}}, \tag{4.9}$$

ahol m a robot kerekének tömege, $F(k)$ a robot kerekeire ható eredő erő a pálya k-adik pontjában, $a_c(k)$ a kerék centripetális gyorsulása, $a_t(k)$ a kerék tangenciális gyorsulása

A 4.9. egyenletben azzal a feltevéssel élünk, hogy a robot kerekei és a talaj között a tapadási súrlódási együttható állandó és nem függ az erő irányától. Az általunk használt differenciális robotnál ez a közelítés megengedhető, mivel a gumikerekek homogénnek tekinthetők. Ha barázdákat tartalmaznának, akkor már nagyobb eltérést okozna ez a közelítés.

Fontos megjegyezni, hogy a kerék gyorsulás korlátokat lassulásnál is alkalmazzuk. Tehát a kerék gyorsulásának abszolút értékét korlátozzák ezek a korlátozások. Így azt tesszük fel, hogy a kerekek viselkedése gyorsulás és lassulás esetében megegyezik. A robot sebességénél viszont nem engedünk negatív értékeket, a robot végig előre haladhat. A tervező viszont megadhat olyan pályát ahol tolatnia kell a robotnak, vagy egy helyben megfordulnia, de ezt a pályatervező algoritmus kezeli.

4.2.2. Geometriai sebességprofil

Első lépésként a geometriai pályapontokhoz rendelünk a korlátoknak megfelelő sebességeket és a későbbiekben ezt a sebességprofilt használjuk fel a pálya újramintavételezésekor.

A pályamenti sebességeket úgy határozzuk meg, hogy a robot gyorsulása a lehető legnagyobb legyen. A 4.5. egyenlet alapján ezt megtehetjük úgy, hogy a robot kerekeinek tangenciális gyorsulását maximalizáljuk. Több hatás miatt nem tudjuk a kerekek gyorsulását folyamatosan növelni.

Egyrészt a robot sebesség és szögsebesség korlátját nem sérthetjük meg. Ebből a két korlátból a pálya minden pontjára kiszámolhatunk egy maximális sebességet függetlenül az előző pályapont sebességétől:

$$v^{max}(k) = \min\left(v^{max}, \frac{\omega^{max}}{c(k)}\right) \tag{4.10}$$

Valamint a kerekek centripetális gyorsulása nem haladhatja meg az előírt eredő gyorsulás korlátot, különben a robot kereke megcsúszna. A pálya adott k. pontjában a kerekek centripetális gyorsulást a következőképpen számolhatjuk ki:

$$\begin{aligned} a_{lc}(k) &= (v(k) \cdot p_l(k))^2 \cdot c(k) \\ a_{rc}(k) &= (v(k) \cdot p_r(k))^2 \cdot c(k) \end{aligned} \tag{4.11}$$

Fontos megjegyezni, hogy mivel mi a robot gyorsulását határozzuk meg a k. pontban, így a $v(k)$ már rendelkezésünkre áll a $k-1$. pontban számított gyorsulásból.

Amennyiben a kiszámolt centripetális gyorsulások már önmagukban is meghaladják az előírt eredő gyorsulás korlátot, úgy $v(k)$ értékét addig kell csökkenteni, hogy a centripetális gyorsulás az eredő gyorsulás korlátot már ne haladja meg.

Ezután a kerekek tangenciális gyorsulását a 4.12. egyenlet alapján határozhatjuk meg.

$$a_{lt}(k) = \min\left(\sqrt{a_l{}^{max})^2 - a_{lc}(k)^2}, a_{lt}{}^{max}\right) \tag{4.12}$$

$$a_{rt}(k) = \min\left(\sqrt{a_r{}^{max})^2 - a_{rc}(k)^2}, a_{rt}{}^{max}\right) \tag{4.13}$$

Eddig a két kerék gyorsulást teljesen függetlenül tárgyaltuk, azonban mindkét gyorsulást nem választhatjuk meg szabadon, mert a pálya görbülete meghatározza a köztük lévő arányt. Ezt a következőképpen láthatjuk be (a 4.3 alapján könnyedén belátható, hogy sebességek aránya is ugyanez lesz):

$$a_{lt}(k) = \beta(k) \cdot (r(k) - \frac{W}{2}) \tag{4.14}$$

$$a_{rt}(k) = \beta(k) \cdot (r(k) + \frac{W}{2}) \tag{4.15}$$

$$\frac{a_{lt}(k)}{a_{rt}(k)} = \frac{r(k) - \frac{W}{2}}{r(k) + \frac{W}{2}} = \frac{p_l(k)}{p_r(k)}, \tag{4.16}$$

ahol $\beta(k)$ a robot szöggyorsulása, $r(k)$ a pálya görbületi sugara a robot középpontjához viszonyítva.

A 4.16. és a 4.12. egyenletek alapján 2-2 lehetséges kerék gyorsulást tudunk számolni. Ezek közül azt a gyorsulás párt fogjuk választani, amelyiknek egyik eleme sem sérti a 4.12. egyenletek által meghatározott korlátokat.

Miután kiszámoltuk, hogy az adott pályapontnál mekkora legyen a robot kerekeinek tangenciális gyorsulása már könnyedén számolható a a robot gyorsulása és sebessége:

$$a_t(k) = \frac{a_{lt}(k) + a_{rt}(k)}{2} \tag{4.17}$$

$$v(k+1) = \min\left(v^{max}(k+1), \sqrt{v(k) + 2 \cdot a_t(k) \cdot \Delta s_c(k)}\right) \tag{4.18}$$

Profil visszaterjesztés

Két esetben előfordulhat, hogy az előző pálya ponthoz meghatározott sebesség értéket módosítani kell. Egyrészt ha a centripetális gyorsulás önmagában meghaladja a megengedhető maximális gyorsulást, akkor az előző pálya ponthoz tartozó sebességet mindenképp csökkenteni kell. Másrészt a 4.18. egyenlet esetében előfordulhat, hogy a robot gyorsulás korlátját megsértjük és így módosítani kell az előző ponthoz tartozó sebességet. Ez például a pálya végpontjában fordulhat elő, ahol előírjuk, hogy a robot álljon meg, tehát $v^{max}(N) = 0$. Ha nem terjesztenénk vissza a profilt, akkor az utolsó pontnál lévő fékezés meghaladhatja az előírt korlátot, hiszen az előző pontokban nem tudtuk, hogy meg kell állni a robotnak.

Mindkét esetben ugyanazt az eljárást alkalmazhatjuk a visszaterjesztéshez. Azért beszélünk visszaterjesztésről, mivel addig kell visszafelé haladni a pályán, amíg minden korlátot betartunk.

Kezdetnek kiszámoljuk, hogy a megváltozott sebesség következtében hogyan alakulnak a kerekek tangenciális gyorsulásai. A 4.19. egyenletben felhasználjuk a 4.3. egyenlet összefüggését a robot és kerék sebesség kapcsolatára.

$$\begin{aligned} a_{lt}(k) &= \frac{v_l(k+1)^2 - v_l(k)^2}{2 \cdot \Delta s_l(k)} = \frac{v(k+1)^2 - v(k)^2}{2 \cdot \Delta s_l(k)} \cdot p_l(k)^2 \\ a_{rt}(k) &= \frac{v_r(k+1)^2 - v_r(k)^2}{2 \cdot \Delta s_l(k)} = \frac{v(k+1)^2 - v(k)^2}{2 \cdot \Delta s_r(k)} \cdot p_r(k)^2 \end{aligned} \tag{4.19}$$

Amennyiben a kapott tangenciális gyorsulások megsértik a tangenciális vagy eredő gyorsulásra vonatkozó korlátokat kiszámoljuk, hogy mekkora robot sebesség esetében teljesülnének a korlátok. Ezt mindkét kerék esetén megtesszük és a szigorúbb sebesség korlátot fogjuk választani, mint robot sebesség. Ezt az eljárást mindaddig megtesszük visszafelé a pályán, amíg azt nem kapjuk, hogy egyik kerék sem sérti meg a korlátokat.

Most vizsgáljuk meg, hogy ha a kerék gyorsulás egy adott korlátot megsért, akkor hogyan kapjuk meg belőle azt a robot sebességet, amely esetében még nem sértjük meg a korlátot.

Először tekintsük a tangenciális gyorsulásra vonatkozó korlátot. A 4.19. egyenletet fejezzük ki $v(k)$-ra mindkét kerék esetén:

$$
\begin{aligned}
{v_l}^t(k) &= \sqrt{v(k+1)^2 + \frac{2 \cdot {a_{lt}}^{max} \Delta s_l(k)}{p_l(k)^2}} \\
{v_r}^t(k) &= \sqrt{v(k+1)^2 + \frac{2 \cdot {a_{rt}}^{max} \Delta s_r(k)}{p_r(k)^2}},
\end{aligned} \tag{4.20}
$$

ahol a ${v_l}^t(k)$, ${v_r}^t(k)$ jelölések arra utalnak, hogy a sebességek a tangenciális korlátból adódnak a bal és jobb kerék esetén.

Az eredő gyorsulásra vonatkozó korlát esetén pedig a 4.21. összefüggést használhatjuk. Ehhez felhasználjuk a 4.11. egyenletet (az egyszerűség kedvéért most elhagyjuk a kereket azonosító indexet, a két kerék esetén ugyanúgy történik a számítás):

$$
a_t(k) = \frac{v(k+1)^2 - v^c(k)^2}{2 \cdot \Delta s(k)} \cdot p(k)^2 = \sqrt{(a^{max})^2 - ({a_c}^{max})^2} = \sqrt{(a^{max})^2 - \left((v^c(k) \cdot p(k))^2 \cdot c(k)\right)^2} \tag{4.21}
$$

ahol a $v^c(k)$ jelölés arra utal, hogy a sebesség az eredő gyorsulásra vonatkozó korlátból adódik.

A 4.21. egyenletet kifejezhetjük $v^c(k)$-re. Ekkor egy negyedfokú egyenletet kapunk, ami a következőképpen épül fel:

$$
\begin{aligned}
d(k) &= \frac{p(k)^4}{4 \cdot \Delta s(k)^2} + c(k) \cdot p(k)^2 && (4.22) \\
e(k) &= -\frac{2 \cdot v(k+1)^2 \cdot p(k)^4}{4 \cdot \Delta s(k)^2} \\
f(k) &= \frac{v(k+1)^4 \cdot p(k)^4}{4 \cdot \Delta s(k)^2} - {a_{max}}^2 \\
0 &= v^c(k)^4 \cdot d(k) + v^c(k)^2 \cdot e(k) + f(k) && (4.23)
\end{aligned}
$$

A 4.23. egyenlet valós, pozitív megoldásait keressük. Felmerülhet a kérdés, hogy mi garantálja, hogy mindig lesz valós, pozitív megoldás. A Viete-formula felírásával belátható, hogy mindig pozitív megoldása van az egyenletnek, a másodfokú egyenlet diszkriminánsának felírásával pedig, hogy lesz valós megoldás. Amennyiben több pozitív valós megoldása van az egyenletnek, akkor a legnagyobb megoldást választjuk.

Miután meghatároztuk $v^c(k)$ és $v^t(k)$ értékeit mindkét kerékre, $v(k)$ értéke ezek közül a legkisebb lesz, hiszen így biztosíthatjuk, hogy a robot egyik kereke sem fogja megsérteni a két gyorsulás korlátot.

A visszaterjesztés során a sebesség és szögsebesség korlátokkal nem kell foglalkoznunk, hiszen mindkét esetben mikor módosítjuk a sebességet, csökkentjük az értékét.

4.2.3. Újramintavételezés

Miután elkészítettük a geometriai pályához tartozó sebességprofilt, létrehozzuk a végleges pályát, amit majd a pályakövető egység bemenetként megkap. Ez a végleges pálya már időben egyenletesen lesz mintavételezve (mintavételezett pálya).

Először számoljuk ki az eltelt időt a geometriai pálya mentén. A számolás alapja, hogy két pályapont között a robot állandó gyorsulással halad.

$$\Delta t^g(k) = \frac{2\Delta s^g(k)}{v^g(k) + v^g(k+1)} \tag{4.24}$$

$$t^g(k+1) = t^g(k) + \Delta t^g(k) \tag{4.25}$$

A következő lépésben meghatározzuk, hogy az újramintavételezett pályánk hány pontból álljon. Ezt könnyedén megtehetjük, hiszen adott számunkra a kívánt mintavételi idő(t_s). Így a következő képlet adódik a mintavételezett pálya pontjainak számára:

$$N^s = \lceil t^g(N^g)/t_s \rceil + 1 \tag{4.26}$$

A pontok számába beleértjük a kezdő és végpontot is. A 4.26. egyenletből következik, hogy amennyiben $t(N^g)$ és t_s nem egymás többszörösei, a mintavételezett pálya utolsó pontjához olyan időpont tartozik, amely nagyobb mint $t(N^g)$. A pálya végpontját még a későbbiekben tárgyaljuk, ott vissza térünk erre az eltérésre is.

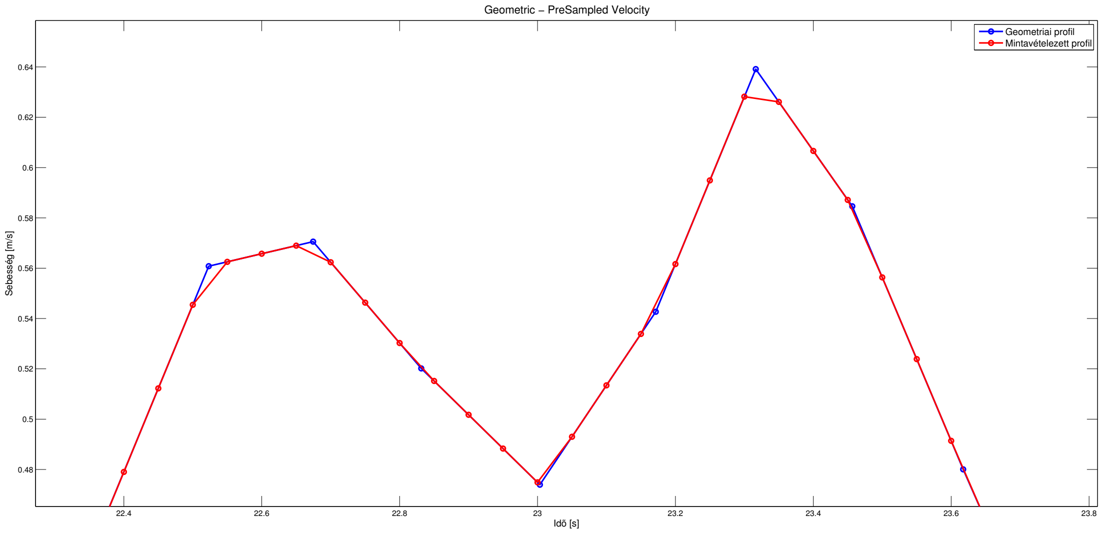
Most pedig meghatározzuk a mintavételezett pálya pontjaiban a sebességet. Ezt a geometriai pálya alapján tesszük, figyelembe véve, hogy a mintavételezett pálya esetén is két pont között állandó gyorsulást feltételezünk. A számítás egy egyszerű lineáris interpolációt valósit meg:

$$v^s(k) = v^g(j) + v^g(j+1) \cdot it(k) \tag{4.27}$$

$$it(k) = \frac{t^s(k) - t^g(j)}{t^g(j+1) - t^g(j)}, \tag{4.28}$$

ahol j jelöli a legkisebb indexet amelyre teljesül, hogy $t^s(k) < t^g(j)$

A lineáris interpoláció miatt teljesül az a feltétel, hogy két pont között állandó gyorsulással mozogjon a robot.



4.2. ábra. *A geometriai és mintavételezett sebességprofil.*

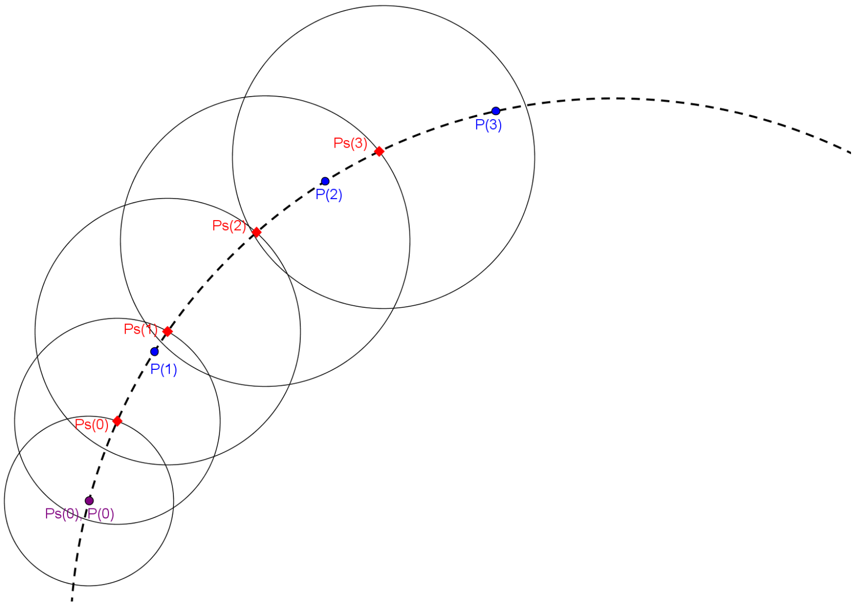
A kiszámított sebességprofil alapján könnyedén adódik a megtett út is:

$$\Delta s^s(k) = \frac{v^s(k) + v^s(k+1)}{2} \cdot t_s \tag{4.29}$$

$$s^{s+1}(k) = s^s(k) + \Delta s^s(k) \tag{4.30}$$

Így már rendelkezésünkre áll a robot kívánt sebessége, a megtett út, valamint az idő a mintavételezett pálya összes pontjában. Már csupán a pálya pontjainak koordinátáit kell ezek alapján meghatároznunk.

Mivel ismerjük a pálya pontok közötti távolságot, iteratív eljárással az előző pálya pont koordinátái alapján az aktuális pontról tudjuk, hogy egy körpályán helyezkedik el. További feltételünk, hogy a pont az eredeti, geometriai pályán rajta legyen. Ha vesszük a geometriai pálya pontjai közötti görbületből adódó köríveket, akkor az ívek és a kör metszéspontjai közül kell kiválasztanunk a keresett pontot. A kiválasztás egyszerű, ha megjegyezzük, hogy az előző pontnál melyik szakasz alapján találtuk meg a pontot, így csak attól a szakasztól kezdve kell keresni a metszéspontokat. Az algoritmus menete látható a 4.3. ábrán. Minden vizsgált szakasznál arra kell figyelni, hogy a metszéspont a szakasz határpontjai között helyezkedjen el. Az első szakasz vizsgálatánál még az is fontos, hogy az előző pont előtti metszéspontot ne vegyük figyelembe. Az ábrán a $Ps(1)$ pontban ezért nem választhatjuk a másik metszéspontot. A legelső mintavételezett pontot a geometriai pálya első pontjába helyezzük el.



4.3. ábra. *A mintavételezett pontok meghatározása. $P(x)$ a geometriai pálya pontjait jelöli, $Ps(y)$ pedig a keletkező mintavételezett pályát.*

Mintavételezett pálya végpontja

Az lenne az optimális esett ha a mintavételezett pálya utolsó pontja egybeesne az eredeti pálya végpontjával, ahogyan a kezdőpontjaik ténylegesen egybeesnek. Alapvetően mi úgy hoztuk létre a mintavételezett pályát, hogy az a geometriai pálya sebességprofiljának megfeleljen, ez viszont nem garantálja az előző feltétel teljesülését.

Három hatás azt eredményezi, hogy nem fog teljesülni ez a feltétel a pálya utolsó pontjára:

1. Ahogy már említettük korábban nem biztos, hogy a két pályát ugyanannyi idő alatt járja be a robot. Ez maximum t_s időkülönbséget okozhat, és minden esetben távolabbi végpontot eredményez, mint az eredeti végpont.

2. A mintavételezett pálya sebességprofiljának elkészítésekor nem tökéletesen követi az eredeti sebességet a robot a mintavételezésből adódóan. Ez látszik a 4.2. ábrán is. A hiba megegyezik a két görbe alatti terület közötti különbséggel. Ez a hiba okozhat távolabbi és közelebbi végpontot is.

3. A harmadik hiba a koordináták meghatározásánál keletkezik. Ez a hatás is mindig távolabbi végpontot okoz.

A legtöbb esetben célszerű, ha a végpontok egybeesnek, így ezt a mintavételezett pálya meghatározásánál biztosítanunk kell. Ha egyszerűen az utolsó pályapontot az eredeti pálya végpontjába tesszük nem biztos, hogy betartjuk a robot gyorsulás korlátait, így más módszerhez kell folyamodnunk.

Az általunk használt algoritmus lényege, hogy a sebességprofilnak egy részét egy adott sebességgel eltoljuk úgy, hogy a két pálya végpontja pontosan egybeessen. Az eltolás mértékét (Δv_{corr}) a következő képlettel kapjuk meg:

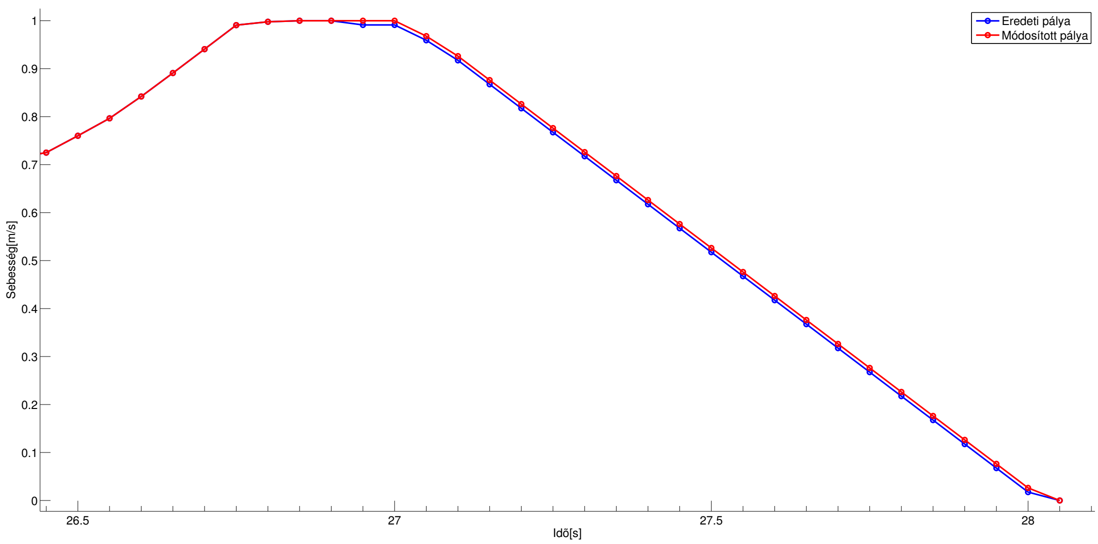
$$\Delta v_{corr} = \frac{\Delta s_{corr}}{t_s \cdot n}, \tag{4.31}$$

ahol Δs_{corr} a mintavételezett és a geometriai pálya végpontjai közötti távolság előjelesen. Ha a mintavételezett pálya utolsó pontja van távolabb, akkor negatív a távolság, különben pozitív. n pedig azoknak a sebességpontoknak a száma, amiket eltolunk.

A 4.31. egyenlet egyszerűen belátható ha felírjuk az eltolásból adódó területkülönbséget. A Δs_{corr} útkülönbséget azért kell előjelesen megadnunk, hogy mindkét esetben használható legyen az algoritmus, akkor is ha a mintaévtelezett pálya végpontja van távolabb és akkor is ha a geometriai pályáé.

A továbbiakban meghatározzuk azokat a sebességpontokat, amelyeket Δv_{corr} sebességgel eltolunk. Mivel a megváltozott sebességponthoz tartozó koordinátákat újra ki kell számolnunk, így minél kevesebb pontot szeretnénk eltolni a sebességprofilon. Viszont a sebesség és gyorsulás korlátokat be kell tartanunk, így nem tolhatunk el tetszőlegesen kevés pontot.

Vizsgáljuk külön a két alapesetet Δs_{corr} előjele alapján. Kezdjük azzal az esettel amikor Δs_{corr} negatív, tehát a mintavételezett pálya végpontja van távolabb (4.4. ábra). Ekkor a módosítandó szakasz kezdő pontjához tartozó gyorsulásnak pozitívnak kell lennie, hiszen mi csökkenteni fogjuk a soron következő pont sebességét és ha a gyorsulás pozitív, akkor csökken a robot gyorsulása a szakasz kezdőpontjában. Ha a gyorsulás negatív lenne a kezdőpontban, akkor könnyedén előfordulhat olyan eset, hogy a sebességcsökkentés után megszegjük a gyorsulás korlátot.

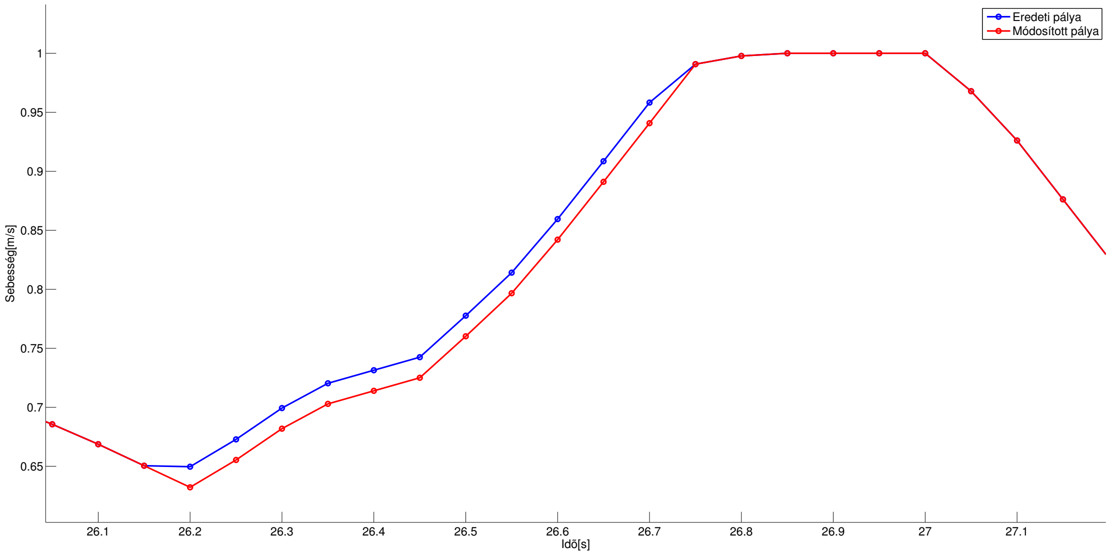


4.4. ábra. *A módosított mintavételezett sebességprofil ha Δs_{corr} negatív.*

A szakasz végpontjánál pedig negatív gyorsulás szükséges, hiszem a következő pont gyorsulása meg fog nőni a módosítás hatására, és ha pozitív lenne a gyorsulás, a gyorsulásra vonatkozó korlátunkat könnyedén megszegnénk.

Tehát a legegyszerűbb esetben a szakasz kezdőpontja a pálya végén található lassító szakasz eleje, mielőtt lassítani kezd a robot és a végpontja pedig a pálya utolsó előtti pontja. Ennek a szakasznak a pontjait fogjuk a 4.31. egyenletből adódó Δv_{corr} sebességgel csökkenteni és így a robot pontosan a geometriai pálya végpontjában áll meg.

A másik eset, mikor Δs_{corr} pozitív, tehát a mintavételezett pálya végpontja messzebb van a geometriai pálya végpontjához képest. Ekkor mivel meg fogjuk növelni a szakasz sebességét pont fordítva kell szakaszt választanunk, a kezdőpontjánál negatív gyorsulás szükséges, a végpontjánál pedig pozitív. Így kerülhető el leginkább a gyorsulás korlát megszegése. Itt pedig egy megfelelő szakasz a pályán található utolsó gyorsító rész.



4.5. ábra. *A módosított mintavételezett sebességprofil ha Δs_{corr} pozitív.*

Abban az esetben ha valamiért az előbb leírt triviális szakaszok mégsem jók, másik szakaszt kell választanunk. Első lépésként válaszunk ki egy megfelelő végpontot a keresendő szakaszhoz. Ha Δs_{corr} negatív akkor megfelelő választás a pálya utolsó előtti pontja, ha pozitív akkor pedig a pálya utolsó olyan pontja, ahol a gyorsulás pozitív. Ezután keressünk ehhez a kiválasztott végponthoz egy kezdőpontot, de most már vegyünk figyelembe a robot korlátozásait és természetesen azt, hogy az útkülönbség az előírt Δs_{corr} legyen. Miután megkaptuk a kezdőpontot is még ellenőriznünk kell, hogy a végpontnál a robot korlátozásai nem sértjük-e meg. Ezt az első lépésben nem tudtuk megtenni, mivel nem ismertük a végpontot, így Δv_{corr} értékét sem. Ha a végpont megsérti a korlátokat, új végpontot kell keresnünk és ahhoz új kezdőpontot. Ezt addig kell folytatnunk, amíg a robot korlátozásait betartjuk.

Miután a módosított sebességprofil elkészült a szakasz elejétől kezdve újra kell számolnunk a mintavételezett pálya koordinátáit. Ezt teljesen ugyanúgy történik, ahogyan már egyszer megkaptuk a mintavételezett pályát. Azért volt fontos, hogy a lehető legkevesebb sebességpontot toljuk el, hogy a koordináták újraszámlálását is kevesebb pontnál kelljen megtenni.

Habár a fenti iteratív eljárás hosszadalmas tűnik vegyük figyelembe, hogy általában igen kis távolságot kell kompenzálnunk, amihez kis sebességkülönbség tartozik. Ebből adódóan nagy valószínűséggel a triviális szakasz is megfelelő lesz számunkra.

Szintén fontos megjegyezni, hogy mivel a tárgyalt három hatás elsősorban negatív $\Delta_{s_{corr}}$-t eredményezz, így a gyakorlatban ez az eset fordul elő. A gyakorlatot tekintve még megemlítendő, hogy a $\Delta_{s_{corr}}$ nagyságrendje igen csekély a pálya teljes hosszához képest, nehezen elképzelhető akárcsak 1%-ot meghaladó arány a teljes pálya hosszához képest.

4.3. Autószerű robotmodell

Ebben a részben áttekintjük a különbségeket az időparaméterezésben, ha autószerű robotmodellt alkalmazunk. A lényeges különbségek a modellben és a korlátozásokban mutatkoznak. Ezek csak a geometriai sebességprofil alkotásakor mutatkoznak meg, a további lépések teljesen megegyeznek a fentebb részletezettel.

Az autószerű robot modellje a következő.

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\phi,\end{aligned} \tag{4.32}$$

ahol L az első és hátsó tengelyek távolsága, ϕ a kormányszög, v pedig a hátsó tengely középpontjának tangenciális sebessége, melyet a robot referenciapontjának nevezünk.

Könnyen belátható, hogy az egyes kerekek sebességkülönbsége a megtett utak különbségéből adódik, mely arányos az egyes kerekekhez tartozó elfordulási sugárral, így elegendő felírnunk ezeket a sugarakat, illetve ezek arányát. A **??**. ábrán látható hogy egy ilyen robot esetén ezek a sugarak hogyan származtathatók.

A 4.32 harmadik egyenletéből könnyen adódik, hogy a referenciapont által bejárt kör sugara

$$\rho = \frac{L}{\tan\phi} \tag{4.33}$$

alapján számolható, innen a hátsó kerekek által bejárt kör sugara a következőképpen adódik:

$$\begin{aligned}\rho_{rl} &= \rho - \frac{d}{2} \\ \rho_{rr} &= \rho + \frac{d}{2},\end{aligned} \tag{4.34}$$

ahol a d az egy tengelyen található kerekek távolsága. (Bár a 4.33 alapján a sugár lehet negatív, de a profilozás során ezt nem használjuk ki. A továbbiakban az egyenleteket mindig pozitív sugárra írjuk fel, de ezt később részletezett okok miatt nem használjuk ki.)

Ahhoz hogy fordulás közben ne csússzanak meg oldal irányba az első kerekek, a két oldali keréknek különböző szögben kell állnia. Ezt nevezzük Ackermann hajtásnak. Ez a különbség ugyan csak a különböző fordulókörrel áll összefüggésben, és a következőképpen számolható a kormányszögből:

$$\begin{aligned}\phi_r &= \arctan\left(\frac{L}{\rho - \frac{d}{2}}\right) \\ \phi_l &= \arctan\left(\frac{L}{\rho + \frac{d}{2}}\right)\end{aligned} \tag{4.35}$$

Az is könnyen belátható, hogy az első kerekek a hátsó párjukhoz képest a kerékhez tartozó kormányszög koszinuszával fordítottan arányos, ebből számítható az első kerekek fordulási sugara:

$$\begin{aligned}\rho_{fl} &= \frac{\rho - \frac{d}{2}}{\cos\phi_l} \\ \rho_{fr} &= \frac{\rho + \frac{d}{2}}{\cos\phi_r}\end{aligned} \tag{4.36}$$

4.3.1. Korlátozások

Az autószerű robot esetén is nagyon hasonló korlátozásokkal kell számolnunk, mint egy differenciális robot esetén, viszont némelyek egy másikból származtathatók:

$$\begin{aligned}v^{max} &: \text{A robot pályamenti sebesség korlátja} \\ \phi^{max} &: \text{A robot maximális kormányszöge} \\ {a_{wheel}}^{max} &: \text{A robot bármely kerekének eredő gyorsulás korlátja}\end{aligned} \tag{4.37}$$

A differenciális robotnál használt ω^{max} helyett itt ϕ^{max} szerepel, mivel ez egy fizikai korlátja az autónak, de ez szükség esetén egyszerűen átszámítható a maximális sebesség ismeretében. A gyorsulások közül csak a a^{max} jelent igazi korlátozást, mivel egy egyenes pályán haladva a centripetális gyorsulás értéke nulla, így ebben az esetben ez megegyezik a tangenciális gyorsulással. Körpálya esetén pedig a 4.9 alapján származtatható.

Jelentős különbség, hogy ebben az esetben nem határozzuk meg a maximális sebességet minden kerékre, mivel ez nagyon elbonyolítaná a számításokat, de szerencsére erre nincs is szükség. Mivel a különböző keréksebességek a sugarak arányiból számíthatók, így nekünk elegendő mindig csak a legnagyobb sugárral számolni. Ez a 4.34 és a 4.36 egyenletek alapján látható hogy pozitív sugár esetén a bal, míg negatív esetén a jobb oldali első kerék esetén teljesül. Az algoritmus során, pontosan ezért mi csak a sugár abszolút értékével számolunk.

Az autószerű robot esetén is azzal a feltételezéssel élünk, hogy a kerekek tapadási tényezője irányfüggetlen, bár ez a feltételezés egy rendes autónál már nem feltétlen állja meg a helyét, de az általunk használt robotautó kerekei esetén ez igen jó közelítést mutat.

4.3.2. Geometriai sebességprofil

A sebességprofil meghatározása teljesen analóg módon történik az eddig látottakkal. Meghatározzuk a maximális sebességet, majd az aktuális sebességből kiszámítjuk a centripetális gyorsulás értékét a leginkább terhelt kerék esetén. Ha ez nem sérti meg a korlátokat akkor ebből számítható a terhelt kerék tangenciális gyorsulása, majd abból a kerék sebessége. Innen már egy egyszerű arányosságból adódik a robot sebessége is a következő időpontban.

A profil visszaterjesztés ugyancsak hasonlóan működik, mint a differenciális robot esetén, egyetlen apró kivétel, hogy a 4.19 egyenletet ebben az esetben csak a legnagyobb sugáron mozgó kerékre írjuk fel. Másik változás, hogy a megtett utat is a sugarak arányából származtatjuk, így a következő egyenlet adódik:

$$a_t(k) = \frac{v(k+1)^2 - v(k)^2}{2 \cdot \Delta s(k)} = \frac{v(k+1)^2 - v(k)^2}{2 \cdot \Delta s(k)} \cdot p(k), \tag{4.38}$$

ahol $p(k)$ a maximális sugáron mozgó kerék és a referencia pont sugarának aránya.

Innen átrendezve, és kifejezve $v^c(k)$-t, a következőt kapjuk:

$$\begin{aligned} d(k) &= \frac{p(k)^2}{4 \cdot \Delta s(k)^2} + c(k) \cdot p(k)^2 \qquad (4.39) \\ e(k) &= -\frac{2 \cdot v(k+1)^2 \cdot p(k)^2}{4 \cdot \Delta s(k)^2} \\ f(k) &= \frac{v(k+1)^4 \cdot p(k)^2}{4 \cdot \Delta s(k)^2} - {a_{max}}^2 \\ 0 &= v^c(k)^4 \cdot d(k) + v^c(k)^2 \cdot e(k) + f(k) \qquad (4.40) \end{aligned}$$

Ez a módosítás nem érinti az egyenlet megoldhatóságát, hiszen a két egyenlet ekvivalens, csak a távolság kerékre átszámítása máskor történik meg.

5. fejezet

Pályakövető szabályozás - 10 oldal CsG/NA

5.1. Differenciális robotmodell - 4 oldal NA

5.1.1. Sebesség szabályozás - 2 oldal

5.1.2. Orientáció szabályozás - 2 oldal

5.2. Autószerű robotmodell - 4 oldal CsG

5.2.1. Virtuális vonalkövező szabályozás - 2 oldal

6. fejezet

Algoritmusok megvalósítása - 4 oldal CsG/NA

6.1. Szimuláció

6.2. Valós robotok

7. fejezet

Összegzés - 1 oldal CsG/NA

7.1. Értékelés

7.2. Jövőbeli fejlesztések

Irodalomjegyzék

[1] Kiss Domokos. *Autonóm robot fejlesztése az Eurobot 2007 versenyre, Diplomamunka.* BME-VIK Automatizálási és Alkalmazott Informatikai Tanszék, 2007.

[2] Domokos Kiss. *The RTR Path Planner for Differential Drive Robots.* 2014.

[3] Dirk-Jan Kroon. 2d line curvature and normals. `http://www.mathworks.com/matlabcentral/fileexchange/32696-2d-line-curvature-and-normals/content/LineCurvature2D.m`.

[4] Steven M. LaValle. *Rapidly-exploring random trees: A new tool for path planning.* Computer Science Dept., Iowa State University, 1998.

[5] Christoph Sprunk. *Planning Motion Trajectories for Mobile Robots Using Splines.* Albert-Ludwigs-Universitat Freiburg, 2008.