Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Gyenes Zoltán Bálint

MOBILIS ROBOTOK MOZGÁSTERVEZÉSE DINAMIKUS KÖRNYEZETBEN A SAFETY VELOCITY OBSTACLES MÓDSZERREL

KONZULENS

Gincsainé Dr. Szádeczky-Kardoss Emese

BUDAPEST, 2017

Tartalomjegyzék

Kivonat

A mobilis robotok legfontosabb feladata, hogy egy kiindulási helyről a célpozícióba eljussanak. Legfőbb szempontként mindig szem előtt kell tartani, hogy a robot célig történő mozgása során ne ütközzön egy akadállyal sem, így biztosítva saját és környezete épségét. Ezen mozgás tervezésére szolgálnak a mozgástervező algoritmusok.

TDK dolgozatom célja egy olyan mozgástervező algoritmus fejlesztése és implementálása volt, mely egyaránt biztosítja a robot munkaterében jelenlevő statikus és mozgó akadályok elkerülését. Ehhez a *Velocity Obstacles* módszert használtam fel, amely alkalmas mobilis robotok pályatervezésére dinamikus környezetben.

A módszer a robot és az akadályok között elkerülő manővereket tervez figyelembe véve a robot és az akadályok – ismertnek feltételezett – aktuális pozícióját és sebességét. A témában született legtöbb korábbi publikációban a robotsebesség kiválasztásánál arra törekedtek, hogy azt a sebességet válasszák ki, amellyel a robot a leggyorsabban tudja elérni a kívánt célpozíciót. Egy ilyen pálya követése során a robot rendszerint nagyon megközelíti az akadályokat, ami ütközésveszélyt von maga után, ha az akadályok kiterjedése, pozíciója és sebessége csak pontatlanul áll rendelkezésre.

Munkám során a leggyorsabb célelérést eredményező algoritmus mellett bevezettem egy új módszert, amely a legbiztonságosabb mozgástervezést eredményezi. Ezt a módszert *Safety Velocity Obstacles* módszernek (*SVO*-nak) neveztem el.

A mozgástervezés eredményét minden esetben szimulációban vizsgáltam, a robot és az akadályok mozgásából videót is készítettem.

Ez a mozgástervező algoritmus alkalmazható autonóm járművek mozgásának tervezéséhez is, így a jövőben nagyon sok lehetőség nyílhat a továbbfejlesztésre.

Abstract

The main task of a mobile robot is to reach the goal from a start position. As a main criterion, we always need to guarantee a collision-free motion among the obstacles, while the robot is moving from the start to the target position. This way, we can ensure the safety of the robot and the environment. We can use the motion planning algorithms to plan this movement.

The main purpose of my work was to develop and implement a motion planning algorithm that can ensure the avoidance of static and moving obstacles in the workspace of the robot. I used the *Velocity Obstacles* method, which is suitable for motion planning of mobile robots in a dynamic environment.

With this method, I planned evasive maneuvers by using information about actual velocity and actual position of the robot and obstacles which information are supposed to be known. To choose the velocity of the robot, in the most former publications, the authors always tried to find the velocity, that is the fastest solution to reach the target position. While following such a path, the robot usually goes very close to the obstacles which can result a collision-risk, if only inaccurate size, position and velocity information are available about the obstacles.

In my work, next to the fastest target reaching method, I introduced a new method, that can ensure the safest motion planning. I named this method *Safety Velocity Obstacles* (*SVO)* method.

I always tested the result of the robot's motion planning by simulations and I made videos of the motion of the robot and the obstacles.

This motion planning algorithm is suitable for motion planning of autonomous vehicles, so in the future there are a lot of opportunities for further development.

1 Bevezetés

Korunk egyik legjobban fejlődő iparága a robotika. Ha belegondolunk, életünk minden területén találkozhatunk már robotokkal, hiszen a technológiai fejlődés következtében olyan technikai határokat dönthetünk és dönthettünk le, amelyre korábban legmerészebb álmainkban sem mertünk gondolni. Gyárak ezrei automatizálják gyártósoraikat robotok, robotkarok felhasználásával, ezzel is növelve a termelés hatékonyságát. Robotok felhasználásával megkönnyíthetjük mind a mérnöki, mind a mindennapi élet nehézségeit.

A robotoknak mindig van egy célfeladatuk, ezt kell megvalósítaniuk. Az autonóm robotok külső beavatkozás nélkül képesek elvégezni feladatukat, szenzoraikkal érzékelni tudják a környezet változásait, és ezen változásokra reagálni képesek.

Sok esetben ez a célfeladat a robot mozgása egy kezdő pozícióból egy célpozícióba (pl. gyárakban anyagok, vagy elkészült termékek szállítása egyik helyről a másikra). A robot munkaterében különböző akadályok fordulhatnak elő. Dinamikus környezetről akkor beszélhetünk, ha a térben nemcsak statikus, hanem mozgó akadályok is jelen vannak. A robotnak minden esetben úgy kell a célpozícióba eljutnia, hogy mozgása során egyetlen akadállyal se ütközzön. Ez az elsődleges célja a mozgásának. Természetesen az is fontos, milyen gyorsan tud eljutni a robot a kívánt pozícióba, illetve, hogy mennyi utat kell megtennie ehhez.

A mozgástervező algoritmusok segítségével meghatározhatjuk, hogy a robotok milyen pálya mentén és milyen sebességgel tudják elérni a megadott célt.

A robotok pályatervezésére nagyon sok kísérlet volt már, különösképpen az elmúlt két évtizedben. Adottak a robot dinamikai tulajdonságai, a környezet tulajdonságai, a robot kezdő helyzete és a célpozíció helyzete. A mozgástervezés legfőbb feladata tehát eljuttatni a robotot kezdő helyzetéből a kívánt pozícióba úgy, hogy közben engedelmeskedjünk a környezet szabályainak (ne ütközzünk az akadályokkal). Egy algoritmusnak, amely megoldja ezt a feladatot, véges idő alatt le kell futnia, és megvalósítható megoldással kell visszatérnie, amennyiben létezik, ellenkező esetben jelezni kell, hogy a mozgástervezés valamilyen oknál fogva nem megvalósítható. A mozgástervezés dinamikus környezetben nehéz problémának számít,

mivel egy pályatervezési és sebességprofil tervezési feladatot is végre kell hajtani. Az alap verziójú pályatervezési problémát „Generalised piano movers” problémának nevezzük, mely NP-nehéz probléma [3]. Habár létezik a teljes tervezési algoritmusa ennek a mozgástervezésnek, mégis olyan bonyolult, hogy nem használható gyakorlati alkalmazásokban.

A mozgástervezés dinamikus környezetben egyik megoldásaként a robot konfigurációs teréhez egy idődimenziót adtak hozzá, feltételezve az akadályok pályáját és sebességét. Korábbi munkákban poligon robotok és poligon mozgó akadályok esetére oldották meg a mozgástervezési problémát egy láthatósági gráf keresésével a konfiguráció-idő térben [4, 5].

Másik megközelítés alapján a mozgástervezés dinamikus környezetben felbontható kisebb alproblémákra: pályatervezés és sebesség tervezés. Ez a módszer először meghatározza az elérhető pályát az álló akadályok figyelembevételével, a sebességet a pálya mentén pedig ezután határozza meg, az alapján, hogy a mozgó akadályokat elkerülje a robot [6, 7].

Egy másik megoldási lehetőség, a lokális és visszaható navigáció abban különbözik az általános globális pályatervező megközelítésektől, mint pl. a [9-11], hogy ahelyett, hogy a teljes pályát megterveznék a célig, a robot kizárólagosan a közvetlen környezetével van kölcsönhatásban bármely időpillanatban. Néhány jól ismert visszaható navigációs módszer: a dinamikus ablak módszer [12], az elkerülhetetlen ütközési állapotok módszere [13], sebesség akadályok módszere (*Velocity Obstacles*) [1].

A TDK dolgozat során egy elsőrendű módszert mutatok be, mellyel meghatározom az időben változó környezetben mozgó robotom pályáját, felhasználva a sebesség információkat (mind az akadályok, mind a robotom sebességét), hogy meghatározzam az esetleges ütközéseket. Ez a módszer felhasználja a Velocity Obstacles (sebesség akadály) elgondolást [1], amely feltérképezi a változó környezetet a robot sebesség terével együtt, és így határozza meg a pálya geometriáját a sebességprofillal együtt. A Velocity Obstacles egy elsőrendű közelítése a robotsebességek halmazának, amelyek egy jövőbeli időpillanatban ütközéshez vezetnének valamely akadállyal. Az elkerülő manővereket úgy tudjuk meghatározni, hogy ezen Velocity Obstacles térrészen kívülről választunk sebességet a robotnak. Az

elkerülő manővernek minden esetben biztosítania kell, hogy dinamikailag elérhető legyen, teljesítve az összes fizikai kényszert.

A leggyorsabb célelérés mellett bemutatom az általam kifejlesztett *Safety Velocity Obstacles* módszert, mely biztosítja, hogy a robot a legbiztonságosabb úton érjen el a célhoz.

A dolgozat 2. fejezetében bemutatom a *Velocity Obstacles* módszert és tulajdonságait. Majd a 3. fejezetben részletezem a mozgástervezés megvalósítását az eredeti *VO* módszerrel. Ezt követően a 4. fejezetben bemutatom és elemzem a továbbfejlesztett *Safety Velocity Obstacles (SVO)* módszert. Utána az 5. fejezetben egy összetett példában mutatom be az algoritmusokat. Végezetül a 6. fejezetben értékelem a kapott eredményeket, és a továbbfejlesztés lehetőségeit nézem végig.

2 A Velocity Obstacles módszer bemutatása

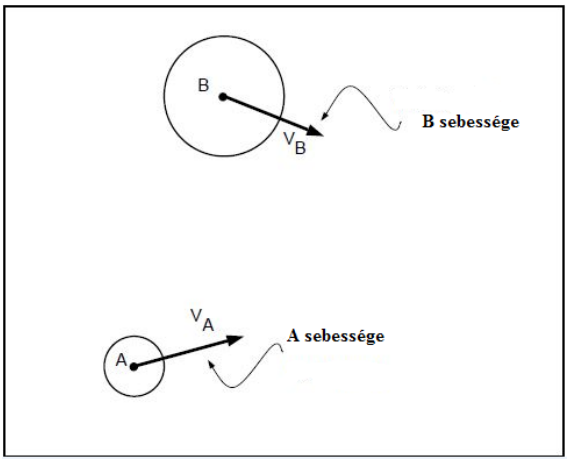
Ebben a fejezetben bemutatom a *Velocity Obstacles*, azaz a sebesség akadály módszert, melynek megismerésének forrásaként az [1] és a [8] szakirodalmat használtam. A módszer több akadály esetén is érvényes, figyelembe véve, hogy statikus vagy mozgó akadályról van-e szó.

A vizsgálatot kör alakú robotra és akadályokra korlátozzuk. Ám ez nem jelent nagyon szigorú korlátozást, amióta felfedezték, hogy általános sokszögeket minden esetben helyettesíteni tudunk több körrel [1].

Az akadályok pályáját úgy közelítjük, hogy egy időhorizonton belül állandó sebességvektort feltételezünk. Amennyiben ez nem elég pontos, akkor az *NLVO* használatával [18], tetszőleges akadály pályák is kezelhetővé válnak.

2.1 CC és VO meghatározása

Tegyük fel, hogy van 2 kör alakú objektum A és B, melyeket a következő ábrán láthatunk:



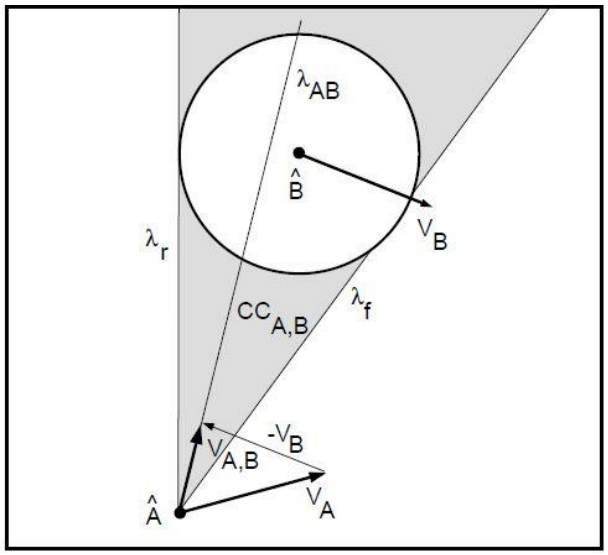
2-1. ábra: Robot és akadály helyzete, a hozzájuk tartozó sebességekkel [1]

A jelölje a robotot, *B* pedig az akadályt, a robotnak a sebességvektora egy adott időpillanatban v_A, az akadály sebességvektora ugyanebben az időpillanatban v_B. Minden egyes akadálynál, a robotnál is a sebességvektorukat a középpontból húzzuk.

Ezután változtassuk meg a konfigurációs terünket, csökkentsük a robotot egy pontra, az akadály méretét pedig növeljük a robot sugarának nagyságával. A robotot

jelölje $\hat{A}$, az akadályt $\hat{B}$. Az A robot és a B akadály mozgásuk során akkor és csak akkor fognak ütközni, ha $\hat{A}$ és $\hat{B}$ ütköznek.

A Collision Cone-t, azaz CC-t, melyet ütközési kúpnak fordíthatunk, úgy definiáljuk, hogy azon relatív sebességek halmaza a robot és az akadály között, amelyek ütközést okoznak. (Ez a tartomány csak térben alkot egy kúpot, síkban nem, de a megoldás térbeli kiterjeszthetősége miatt a szakirodalomban a cone elnevezés terjedt el, ezért a magyar fordításban is a kúpot alkalmazom.)

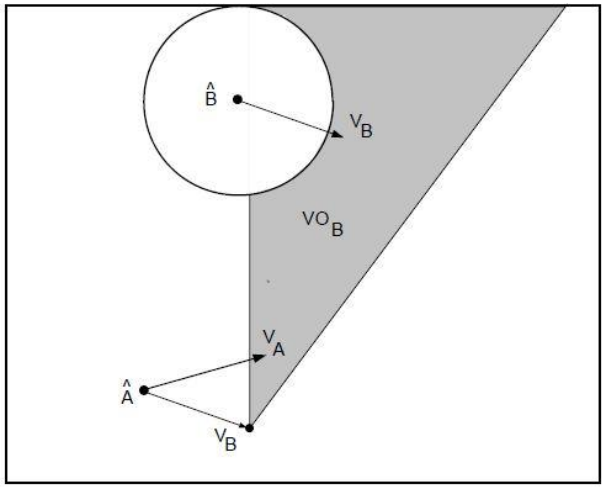


2-2. ábra: Ütközési kúp [1]

A 2-2. ábra mutatja a robotból húzott két érintő félegyenest az akadály felé (λ_r, λ_f). Az ütközési kúp csúcsa a robot pozíciója. Az ütközési kúp tehát nem más, mint a csúcs és a két érintő félegyenes közötti térrész. A képen látható a robot és az akadály sebességvektora (v_A, v_B), illetve a relatív sebesség ($v_{A,B} = v_A - v_B$), illetve $\lambda_{A,B}$ jelöli a relatív sebesség félegyenesét. Abban az esetben, ha ez a félegyenes, azaz a relatív sebesség vektora a két érintő között helyezkedik el, akkor ütközés fog történni a robot és az akadály között. Ha pedig a relatív sebesség ezen ütközési kúpon kívül esik, akkor ütközésmentes mozgást valósítunk meg, feltételezve, hogy az akadály ugyanilyen sebességvektorral rendelkezik a jövőben is. (Amennyiben λ_r mellől választ a robot sebességvektort, akkor az akadály mögött fog elmenni, λ_f mellől sebességvektort választva az akadály előtt fog elhaladni a robot (ha a sebességvektor a CC-n kívül esik)). Tehát az ütközési kúp a következőképpen írható le képlettel:

$$CC_{A,B} = \{v_{A,B} | \lambda_{A,B} \cap \hat{B} \neq 0\} \quad (2.1)$$

Minden egyes akadályhoz tartozik egy ilyen ütközési zóna. Ha több akadály van jelen a térben, akkor előnyös, ha a robotnak mindig csak az abszolút sebességével kell számolnunk, nem a relatív sebességekkel. Ezt úgy érhetjük el, hogy az adott ütközési kúphoz hozzáadjuk az adott akadály sebességét (eltoljuk az ütközési kúpot) így kapjuk meg a *Velocity Obstacles*-t, azaz *VO*-t, melyet sebesség akadálynak fordíthatunk.



2-3. ábra VO térrész [1]

Tehát a *VO*-t a következőképpen definiálhatjuk:

$$VO = CC_{A,B} \oplus v_B \quad (2.2)$$

Ezen összeadás alatt a Minkowsky összeadást értjük [14].

Ha a robotnak a *VO*-n kívülről választunk robotsebességet, akkor megakadályozzuk az ütközést:

$$A(t) \cap B(t) = 0 \text{ ha} v_A(t) \notin VO(t) \quad (2.3)$$

Abban az esetben, ha álló akadályt vizsgálunk, akkor a *VO* megegyezik a *CC*-vel, mivel ekkor az akadály sebességvektorának hossza 0, így az ütközési kúp eltolása nem eredményez változást.

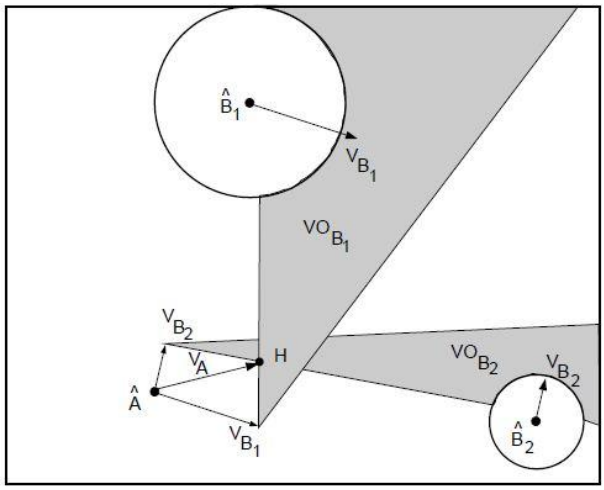
Ha *VO* határáról választunk sebességet, akkor érinteni fogja a robot mozgása során az akadályt, de nem okoz ütközést (2.3.1. fejezet). A valóságban úgy szoktunk ilyen esetekben tervezni, hogy a mozgástervező algoritmus elején az akadályok sugarának nagyságát a biztonsági távolsággal megnöveljük. Így az algoritmus

végrehajtása során, ha érintés történne, akkor sem fog a valóságban összeérni a robot és az akadály.

Több akadály esetén a VO nem más, mint az egyes akadályokhoz tartozó VO-k uniója.

$$VO = \cup_{i=1}^{n} VO_{Bi} \tag{2.4}$$

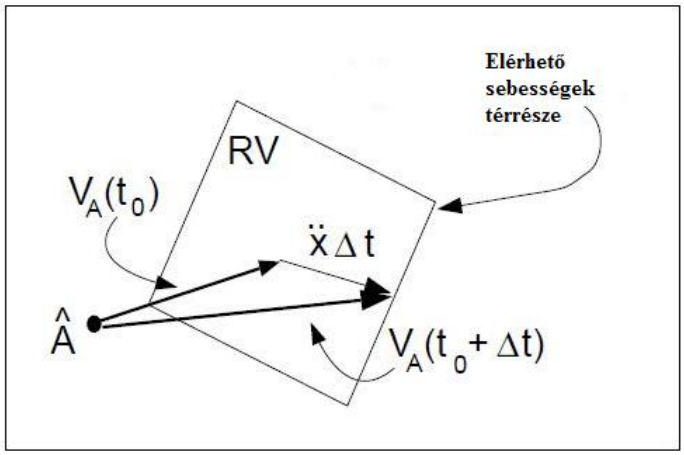
Az egyenletben n jelöli az akadályok számát, VO_{Bi} pedig az i. akadályhoz tartozó VO-t.



2-4. ábra: Több akadály esetén VO meghatározása [1]

2.2 Elérhető elkerülő sebességek térrésze

Az $\hat{A}$ robot által egy adott állapotból Δt idő alatt elérhető sebességeket ki tudjuk számítani a beavatkozó jelek és a mozgásegyenlet alapján.



2-5. ábra: Elérhető sebességek [1]

A *Feasisble Accelerations (FA(t))*, azaz elérhető gyorsulásokat kiszámíthatjuk:

$$FA(t) = \{\ddot{x} | \ddot{x} = f(x, \dot{x}, u),\ u \in U\} \tag{2.5}$$

Itt x jelöli a pozíciót, $f(x, \dot{x}, u)$ a robot dinamikáját, u a beavatkozó jel, U pedig a lehetséges beavatkozó jelek halmaza.

A Δt idő alatt elérhető sebességek, *Reachable Velocities* ($RV(t + \Delta t)$) kiszámíthatóak:

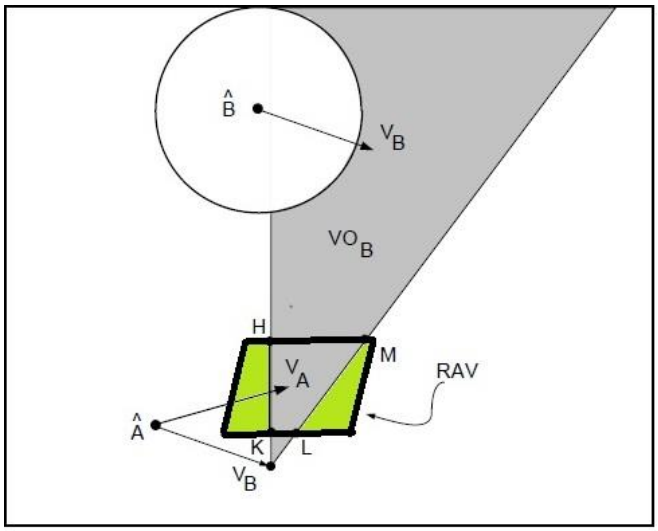
$$\mathrm{RV(t + \ \Delta t)} = \{\mathrm{v} \,|\mathrm{v} = \mathrm{v_A(t)} + \Delta \mathrm{t} * \mathrm{FA(t)}\} \tag{2.6}$$

A *Reachable Avoidance Velocities (RAV)*, azaz elérhető elkerülő sebességek pedig kiszámíthatóak, amennyiben ismerjük az elérhető sebességeket, illetve a *VO* térrészt, mivel minden elérhető elkerülő sebesség úgy definiálható, hogy elérhető és nem okoz ütközést, azaz *RV*-ben benne van, *VO*-ban pedig nincs. Tehát definíció szerint az elérhető elkerülő sebességek:

$$\mathrm{RAV(t + \ \Delta t)} = \mathrm{RV(t + \ \Delta t)} - \mathrm{VO(t)} \tag{2.7}$$

Az egyenletben a ”-” operátor a RV és VO térrészek különbségét képzi.

Ezáltal az akadály elkerülését szolgáló manővert úgy tudjuk számítani, hogy ebből az elérhető elkerülő sebességek térrészből választunk sebességet.



2-6. ábra: RAV bemutatása [1]

A képen vastagított fekete keretben látható a robotnak az elérhető sebesség térrésze (*RV*), amelyet adott időegység alatt elérni képes. Amennyiben ebből a térrészből eltávolítjuk a B akadályhoz tartozó *VO*-t, akkor meg is kapjuk a képen zölddel

satírozott elérhető elkerülő sebességek térrészét (*RAV*). Amennyiben ebből a térrészből választunk sebességvektort a robotnak, akkor mindenféleképpen ütközésmentes mozgást valósítunk meg.

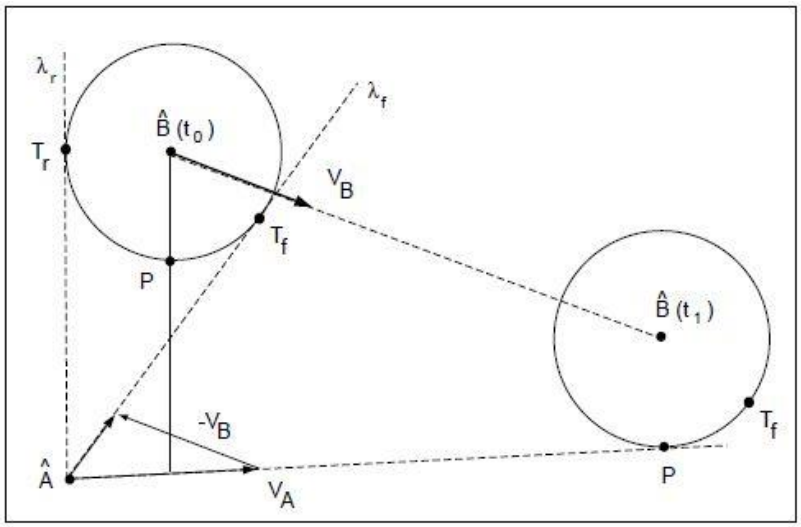
Korábbi munkám során [8] megvizsgáltam az elérhető sebességek térrészét differenciális meghajtású robotok esetén. A kinematikai egyenletek figyelembevételével megállapított elérhető sebességek térrésze során azt tapasztaltam, hogy minél nagyobb a robotnak a sebessége, oldal irányban egyre kevésbé képes elfordulni. Ám minden esetben az *RV* térrész teljes mértékben jól közelíthető volt egy 4 pont által közrezárt térrésszel. A TDK dolgozatban minden esetben ezzel a 4 pont által közrezárt térrésszel közelítem az elérhető sebességek térrészét, hiszen ez a kívánt elvárásoknak teljes mértékben eleget tesz.

2.3 Velocity Obstacles módszer tulajdonságai

A soron következő tételek bizonyítása megtalálható az [1]-es szakirodalom függelékében.

2.3.1 I. tétel

Tétel: A robot mozgása során érinti az akadályt, amennyiben a robot és az akadály közötti $v_{A,B}$ relatív sebesség az akadályhoz húzott érintők (λ_r, vagy λ_f) valamelyikére esik.

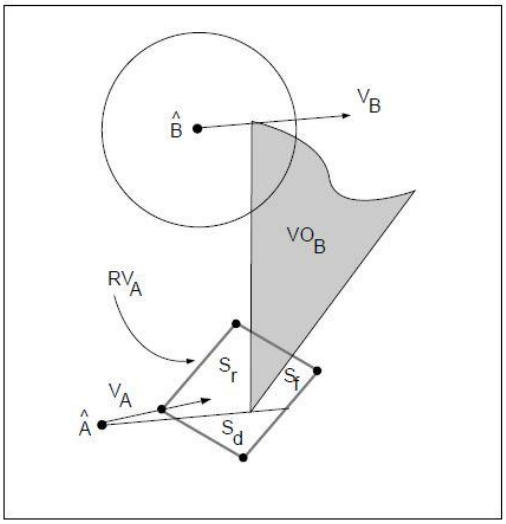


2-7. ábra Akadály érintése a mozgás során [1]

Az 2-7. ábra bemutatja, hogy egy adott t_0 időpillanatban az akadályhoz húzott két érintő félegyenes λ_r és λ_f. A félegyeneseknek és az akadály körvonalának metszéspontja rendre T_r és T_f. A kiválasztott v_A robotsebesség esetén a relatív sebesség λ_f érintő félegyenesre esik. Ekkor az akadály középpontjából merőlegest állíthatunk a robot által kiválasztott v_A sebességvektorra. Ezen merőleges és az akadály körvonalának metszéspontja a P pont. Feltételezzük, hogy nincs forgás, egyenes vonalú mozgást végez mind a robot, mind az akadály, és sebességük is állandó. Ebben az esetben egy adott t_1 időpontban a robot érinteni fogja az akadályt a P pontban, de nem történik ütközés. Természetesen abban az esetben is érvényes ez a tétel, ha VO térrész határáról kiválasztott abszolút sebességéről beszélünk. Ebben az esetben is érinteni fogja a robot az akadályt egy jövőbeli időpontban.

2.3.2 II. tétel

Tétel: Az elérhető elkerülő sebesség térrésze egy akadály esetén mindig felosztható legfeljebb 3 részre: S_f, S_r, S_d. Ezek a térrészek azon sebességeket tartalmazzák, melyek kiválasztása esetén az akadályt rendre elölről, hátulról, illetve kitérő irányban történő elkerülő manőverek valósulnak meg.



2-8. ábra: Elérhető elkerülő sebesség térrészének felosztása [1]

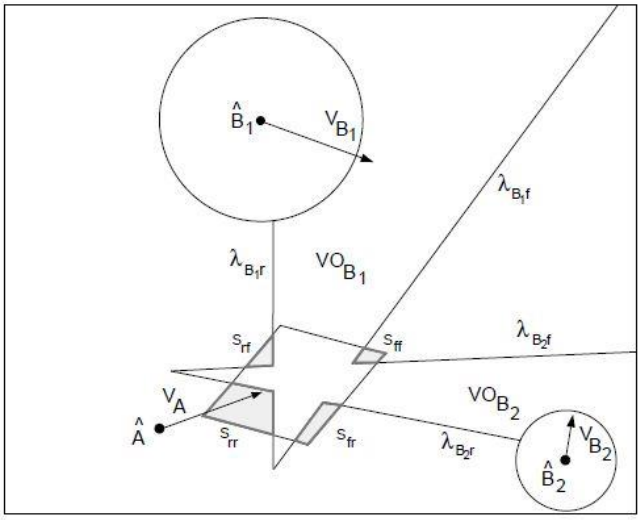
A képen látható elérhető elkerülő sebességek térrészét úgy osztjuk fel, hogy az akadályhoz tartozó VO_B csúcsát összekötjük a robotunkkal, így a VO_B határai és ezen félegyenes által 3 egymástól jól elkülöníthető térrészt kapunk: S_f, S_r, S_d. Ha az S_f térrészből választunk sebességet, akkor elkerülő manőverünk során az akadály előtt fogunk elhaladni, S_r térrészből választott sebesség esetén az akadály után fogunk

elhaladni, míg az alsó S_d térrész esetén teljesen másfelé fogunk menni, a robot és az akadály pályája nem keresztezi egymást.

Természetesen elképzelhető, hogy az elérhető elkerülő sebességek csak 1 vagy 2 részre tagolódnak. Utóbbira mutat példát a 2-6. ábra. Ebben az esetben az elérhető sebességek térrésze a VO csúcsa fölött helyezkedik el, és ennél fogva a VO csak 2 részre osztja ezen sebességek térrészét. Így egy S_f, S_r térrész képződik.

2.3.3 III. tétel

Tétel: N darab mozgó akadály esetén az elérhető elkerülő sebességek térrésze legfeljebb 3N részre osztható fel, mely részek tartalmazzák az elkerülő manőverekhez biztosított választható sebességeket.



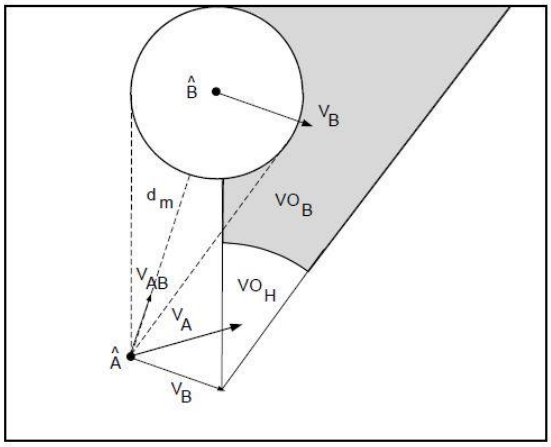
2-9. ábra: Több akadály esetén az elérhető elkerülő sebességek felosztása [1]

Több akadály esetén a felosztott elérhető elkerülő sebességek térrészeit jelölhetjük úgy, hogy a különböző akadályokhoz képest melyik irányból fog elhaladni a robot. A fenti ábrán így az S_{ff} térrész tehát azokat a sebességeket jelöli, amelyek választása esetén a robot mindkét akadály előtt fog elhaladni, míg például az S_{fr} térrész sebességei esetén a $\widehat{B}_1$-es akadály előtt, míg a $\widehat{B}_2$-es akadály mögött fog elmenni a robot.

2.4 Időhorizont

Az eddigiekben az összes akadályhoz tartozó VO térrész ütközést okozó térrésznek számított, azaz ezen térrészekről nem választhattunk a robotnak

sebességvektort, mivel ezek azon sebességkomponenseket tartalmazták, melyek egy adott jövőbeli időpillanatban ütközéshez vezetnének az adott akadállyal. Azonban bevezethetjük az időhorizont fogalmát. Ez azt fejezi ki, hogy csak azt vizsgáljuk, hogy egy adott időhorizonton belül történik-e ütközés vagy nem. Megengedjük olyan sebességvektor választását, ami csak a távoli jövőben fog ütközést okozni, mert addig még lesz a robotunknak lehetősége elkerülő sebesség választására. Mivel egyre nagyobb távolság esetén egyre kisebb az esélye annak, hogy ha a *VO* térrész „csúcsából" választunk sebességkomponenst, az valóban ütközést fog okozni, mivel valószínűtlen, hogy olyan hosszú ideig haladna a robot azzal a sebességkomponenssel, hogy valóban ütköznének. Ezáltal a *VO* térrészeken mindig meg tudunk határozni egy olyan kis részt, melyet olyan térrészekként definiálhatunk, melyből választott sebességkomponens nem okoz ütközést a jövőben a vizsgált időtartományon belül. A 2-10. ábra szemlélteti, hogy VO_H jelöli ezt a térrészt:



2-10. ábra: Időhorizont [1]

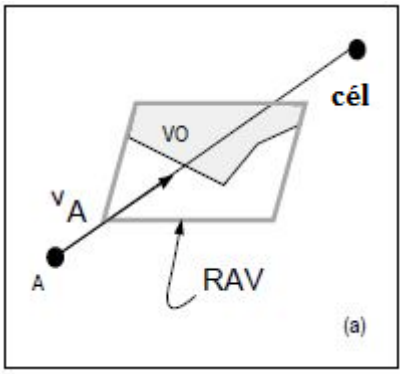
Tehát minden egyes akadály esetén megnézzük, hogy adott előre meghatározott idő múlva (időhorizont idő) hol helyezkedne el az adott akadály, és a robot aktuális pozíciójából érintőket húzhatunk. Amely térrészt ezen érintők lemetszenek, azon térrészeket nem tekintjük ütközést okozó sebességkomponenseknek.

Az időhorizont megválasztása minden pályatervezés esetén kényes kérdés. Ha túl kicsi az időhorizont, akkor növeljük az ütközések veszélyét, ha túl nagy, akkor az a hatékonyság rovására megy. Az algoritmus számítási igényét viszont nem befolyásolja az időhorizont nagysága.

2.5 Heurisztikus mozgástervezés

A heurisztikus mozgástervezés legfontosabb tulajdonsága, hogy „kis számítási igénye miatt” mozgás közben, valós időben tudunk tervezni, mindig meghatározott időközönként térképezzük fel az adott teret és választjuk ki a robotnak a megfelelő sebességet, amellyel egészen a következő időpillanatig haladunk. A heurisztikus mozgástervezés során korábbi munkámban [8] két fő stratégiát implementáltam: *TG (to goal),* illetve az *MV (maximum velocity)* stratégiákat.

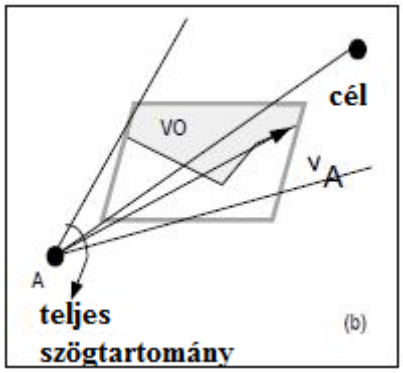
TG stratégia esetén minden adott időpillanatban kiválasztjuk a robotnak azt a legnagyobb elérhető sebességet, amely a cél irányába mutat, és nem okoz ütközést.



2-11. ábra: TG stratégia által kiválasztott legnagyobb sebesség [1]

A fenti ábrán jól látható, hogy a cél irányába mutató legnagyobb sebességet választjuk ki, amely nincs benne a *VO*-ban, azaz nem okoz ütközést, de elérhető sebesség. Tehát úgy is fogalmazhatunk, hogy kiválasztjuk a cél irányába mutató legnagyobb sebességet az elérhető elkerülő sebességek térrészéből.

MV stratégia esetén minden adott időpillanatban kiválasztjuk a robotnak a cél iránya körül egy adott szögtartományba eső legnagyobb elérhető elkerülő sebességet.



2-12. ábra: MV stratégia által kiválasztott legnagyobb sebesség [1]

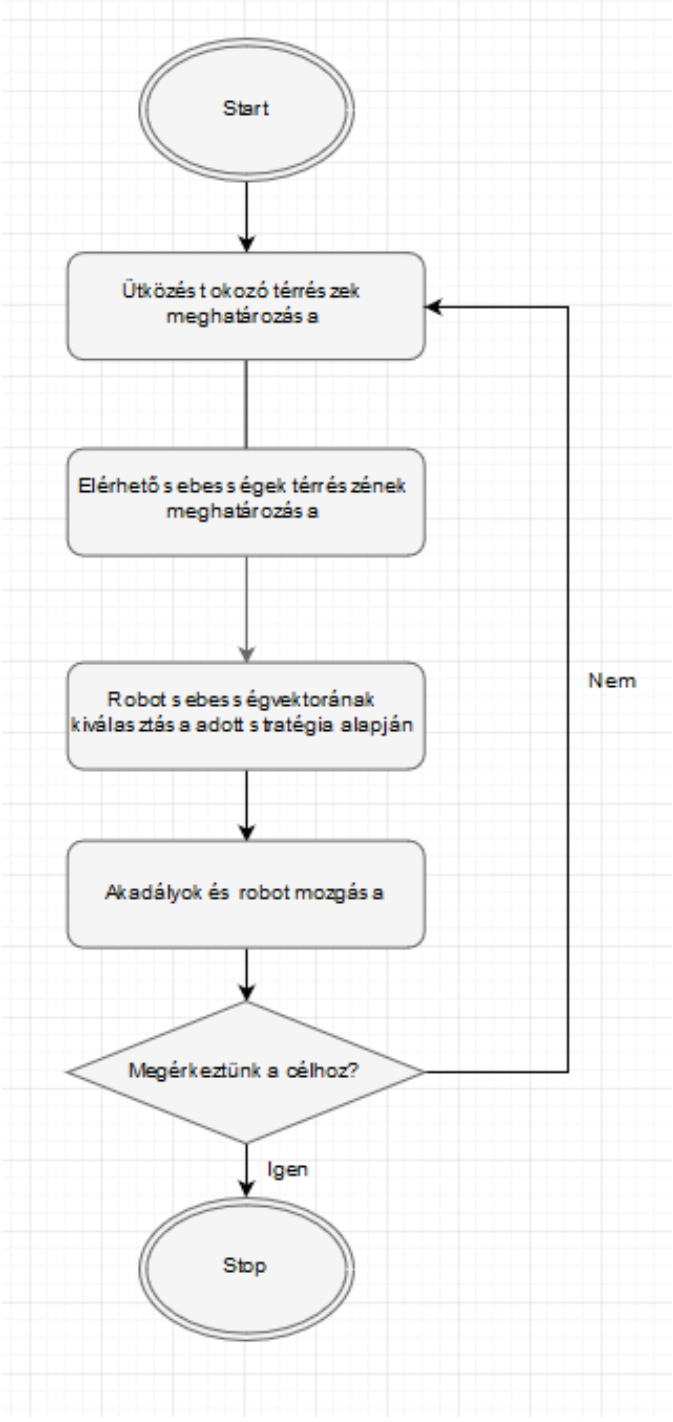
A 2-12. ábra mutatja, hogy a cél iránya körül egy adott szögtartományban végigpásztázzuk a környezetet, és kiválasztjuk a legnagyobb elérhető elkerülő sebességet ezen térrészen belül.

Természetesen semmi sem garantálja, hogy ezen algoritmusok felhasználásával a cél bármely időpillanatban elérhető. Lehetséges, hogy egyáltalán nem létezik megoldás, például, ha körbe van véve a robotunk statikus akadályokkal. Ekkor soha nem tudjuk elérni a célt. Ám az is elképzelhető, hogy létezne megoldás, de ezekkel a heurisztikákkal nem tudjuk megtalálni azt. Például, ha a cél és a robot között szorosan egymás mellett 2-3 nagy sugarú statikus akadály van, abban az esetben nagy kerüléssel el tudnánk érni a célt, de *TG* és *MV* stratégiákkal nem tudjuk megközelíteni.

A *TG* és *MV* stratégiák abban az esetben, ha egy akadály mozgása során keresztezi a robot útját a cél irányában, akkor a *VO* térrészek határáról választanak sebességvektort, amely minden esetben érintést eredményez az akadály és a robot között, mint ahogy a 2.3.1 tétel is bizonyítja.

3 A mozgástervező algoritmus

Az ütközésmentes mozgástervezés során a következő lépésekből álló algoritmust kell megvalósítanunk az eredményes célelérés érdekében.



3-1. ábra: Mozgástervező algoritmus folyamatábrája

Tehát első lépésben mindig az összes akadályhoz tartozó *VO* térrészeket kell meghatározni, amelyeket rögtön redukálunk az időhorizont figyelembevételével. Az így kapott egyes akadályokhoz tartozó *VO* térrészek uniója alkotja az ütközést okozó sebességek térrészét.

Majd a robot aktuális pozíciójának, előző sebességvektorának ismeretében meg tudjuk határozni az elérhető sebességek térrészét.

Ezt követően választunk a robotnak sebességet az elérhető, nem ütközést okozó sebességek térrészéből meghatározott szempontok alapján.

Ezzel a sebességvektorral haladunk előre meghatározott időegységig, majd amennyiben még nem értük el a célt, akkor újból kezdődik az egész környezet feltérképezése.

Így haladva jutunk el a kívánt célpozícióig.

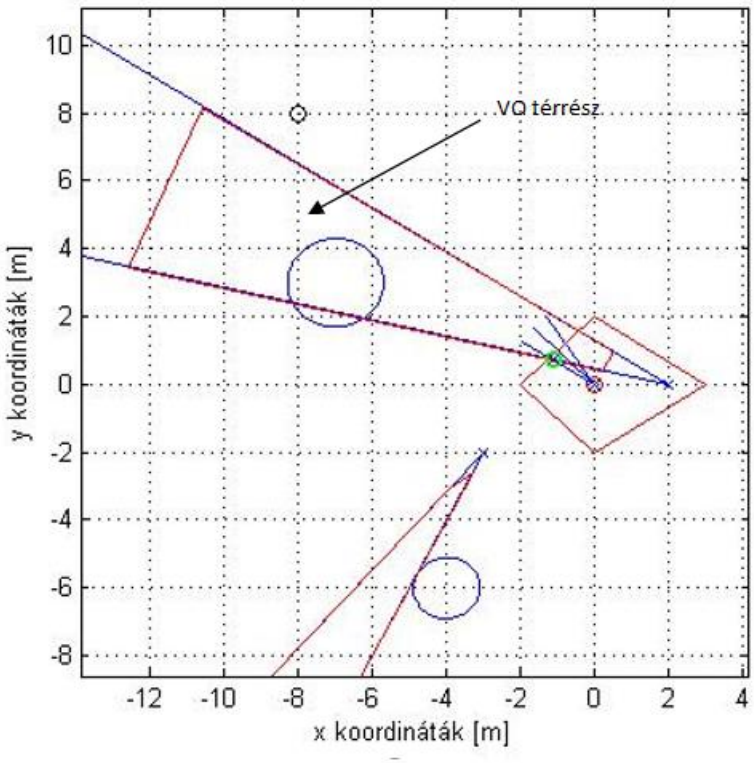
Az algoritmus implementálását MATLAB fejlesztői környezetben végeztem, a *VO* térrészek, elérhető sebességek térrészének pontos meghatározása, egy adott sebességkomponens elérhetőségének, ütközésének vizsgálata korábbi munkámban olvasható [8].

3.1 Eredeti VO módszer

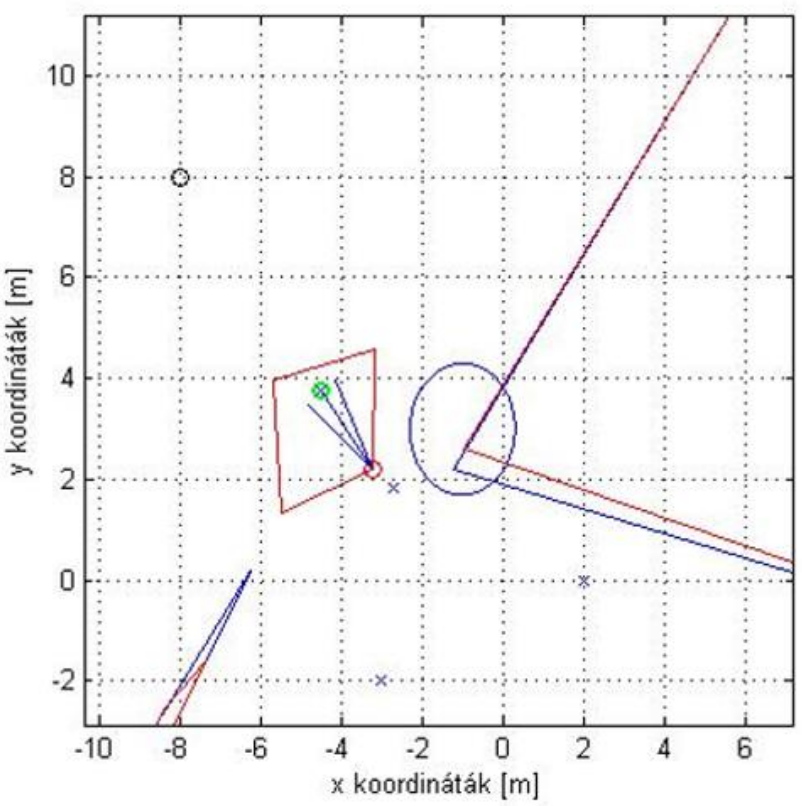
Az eredeti *VO* módszer esetében mindig a leggyorsabb célelérést preferáljuk, azaz, mint ahogyan a *TG* és *MV* stratégiák esetén is bemutattam mindig azt a legnagyobb sebességvektor komponenst választjuk ki, amellyel vagy a cél irányába vagy egy adott szögtartományban a leggyorsabban tudunk haladni. Az MV stratégia megvalósításához a véges számítási igény miatt a cél iránya körül 3 különböző irányú sebességvektor komponenst vizsgálok.

A 3-2. ábra egy példán szemlélteti, hogy a gyorsaság érdekében a robot az akadályhoz tartozó *VO* térrész határáról választ sebességkomponenst. Kék körök jelzik az akadályokat, piros kör mutatja a robot aktuális pozícióját. Kezdeti helyzetben mindig az origóban van a robot. (Amennyiben máshol lenne, egy egyszerű transzformációval a koordinátarendszer középpontjába mozgatható). A robottól húzott három vonal jelöli, hogy MV stratégiával milyen irányú sebességvektorok közül választhat a robot. Zöld kör jelzi a kiválasztott sebességvektor végpontját, amellyel előre meghatározott időegységig haladva a legközelebb jutunk a célhoz. (-8,8-as koordinátában fekete kör

jelzi a célt). A robot körül látható egy piros körvonallal körbevett 4 pont által közrezárt elérhető sebességek térrésze. Az ábrán látható a *VO* térrész is, mely az ütközést okozó sebességkomponenseket tartalmazza (2 kék félegyenes által közrezárt térrész). A 2.4. fejezetnek megfelelően nem a teljes *VO* térrészen vizsgáljuk az ütközést, hanem az időhorizont figyelembevételével azokat a sebességkomponenseket nem tekintjük ütközést okozó komponenseknek, melyeket kiválasztva csak nagyon sok idő elteltével következne be ütközés, illetve a megvalósíthatatlanul nagy robotsebességekkel sem foglalkozunk. Ezáltal az ütközést okozó sebességeket egy zárt térrésszel közelítjük, melyet a teljes *VO* térrészből négy pont által körülvett piros színű térrész jelöl az ábrán.



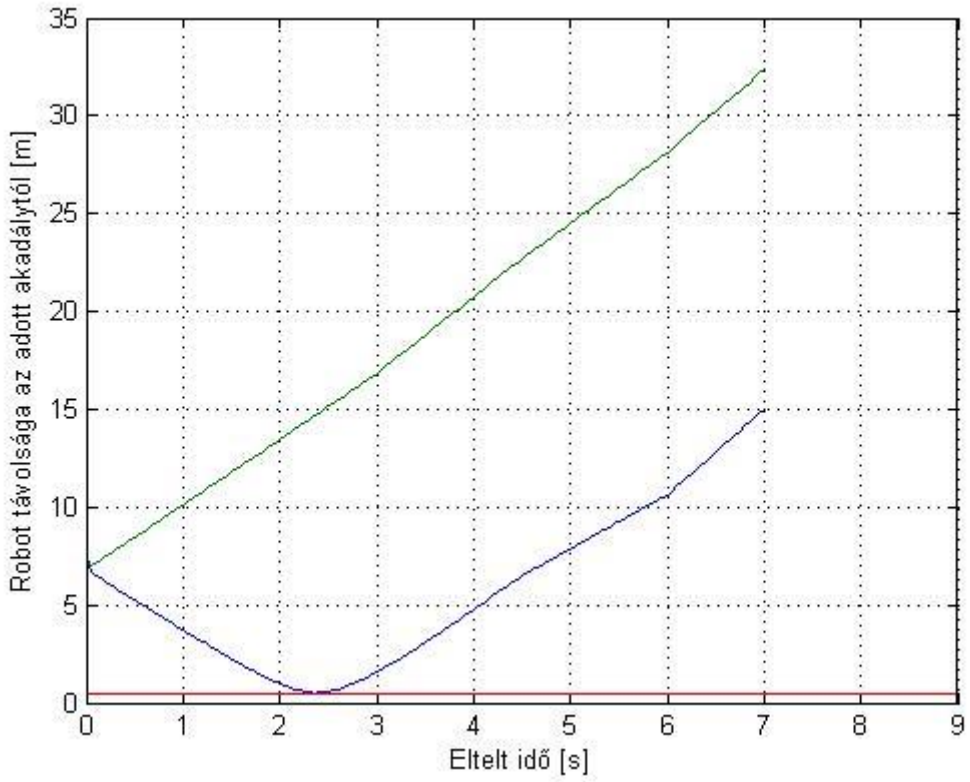
3-2. ábra: Robot sebességvektorának kiválasztása eredeti VO módszerrel MV stratégiával (két akadály esetén)



3-3. ábra: Sebességválasztás abban az esetben, ha teljesen szabad az út a cél felé

A 3-3. ábra mutatja, amikor a robot elhaladt az akadály mellett és teljesen szabaddá vált az útja a cél felé, akkor egyenes vonalban legnagyobb elérhető referencia sebességgel egyenesen a cél irányába haladt már. Minden időpillanatban azért van szükség egy maximális referencia sebességre, hogy ahogy közeledünk a célpozícióhoz, egyre lassuljon a robot annak érdekében, hogy minden esetben meg tudjon állni. Így haladva elérjük a célpozíciót.

Érdemes szemléltetni a robot és az akadályok távolságának alakulását az idő függvényében (ld. 3-4. ábra), mivel jól látszódik, hogy az utazás során egy adott időpillanatban 0 a távolság az egyik akadállyal, érintik egymást, de nem ütköznek, majd ahogy halad tovább az adott akadály és a robot, ismét növekszik a köztük lévő távolság. A távolság meghatározásánál a robot és az akadály távolságából minden esetben kivontam az adott akadály sugarát, míg a robot sugarát egy konstans piros vonallal jelöltem, ezáltal, ha a görbe érinti a piros vonalat, akkor történik érintés az akadály és a robot között. Amennyiben a görbe a piros vonal alá megy, akkor ütközés következik be az akadály és a robot között. (A mozgásról készült videó: https://youtu.be/rLcTp-10dEM)



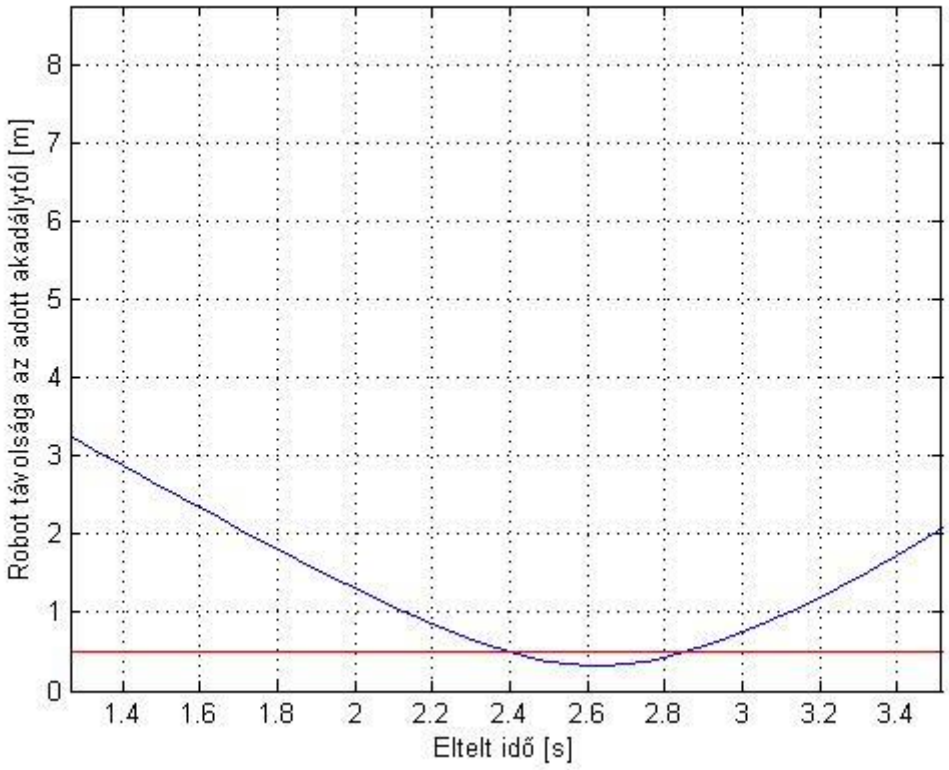
3-4. ábra: A robot középpontjának távolsága az egyes akadályok legközelebbi pontjától

A 3-4. ábra mutatja, hogy ebben az esetben két akadállyal végeztem a szimulációt, az egyiktől való távolságot kék, a másiktól pedig zöld vonallal szemléltetem. Tehát ezzel a módszerrel a legrövidebb útvonalon, leggyorsabban tudjuk elérni a kívánt pozíciót a robot számára, ez a módszer legnagyobb előnye.

Vizsgáljuk meg, mi történik akkor, ha a szenzorbizonytalanság következtében az akadályok pontos sebessége kis mértékben eltér attól, mint amivel mi az algoritmus során számoltunk.

Tekintsük az előző példát, és a kiszámított robotsebességek mellett változtassuk meg annak az akadálynak a sebességvektorát kb. 5 %-kal, amellyel korábban érintkezett a robot mozgása során.

Ekkor megtekinthetjük a robot és az akadályok közötti távolságot, melyet a 3-5. ábra mutat be: (A mozgásról készült videó: https://youtu.be/_oTQ5p2-IMQ)



3-5. ábra: Robot és akadályok távolsága a szenzorbizonytalanság figyelembevételével

Tehát látható, hogy ahol korábban érintés történt a robot és az adott akadály között, ott most a piros vonal (robot sugár) alá kerül a távolság görbe, azaz ütközés történik a két fél között.

A szenzorbizonytalanság által okozott esetleges ütközés elkerülése érdekében fejlesztettem ki a *Safety Velocity Obstacles* módszert.

4 A Safety Velocity Obstacles módszer

A *Safety Velocity Obstacles (SVO)* módszer legfontosabb tulajdonsága, hogy minden egyes mintavételezési időpillanatban a robot számára a legbiztonságosabb elérhető elkerülő sebességkomponenst választja ki.

Mindig meg kell vizsgálni, hogy az adott sebességvektor végpontja milyen távolságban van az akadályokhoz tartozó *VO* térrészektől. Minél távolabb esik egy adott sebességvektor végpontjának távolsága a *VO* térrészektől, annál biztonságosabbnak számít azt a sebességvektort választani a robotnak, hiszen ez az jelenti, hogy az adott akadálytól annál messzebb fog eltávolodni.

Egy adott sebességvektor végpont esetén mindig meg kell vizsgálni, hogy az összes *VO* térrésztől mi a legkisebb távolság. Egy bizonyos távolságnál messzebb lévő *VO*-kkal nem foglalkozunk, mert azok még szenzorbizonytalanság esetén se okozhatnak ütközést. Ezért bevezettem egy maximális távolságot, ennél messzebb lévő *VO*-k távolságát is ez fogja jelölni. Ezt a maximális távot úgy határoztam meg, hogy maximális sebesség nagysága (amit a robot elérni képes) szorozva az előre meghatározott időegységgel (amennyi ideig egy adott sebességgel halad a robot).

Képlettel:

$$If\ \ VO_{tav\_i} > maxsebesseg * t \quad Then\ VO_{tav\_i} = maxsebesseg * t \tag{3.1}$$

A (3.1) képletben VO_{tav_i} jelöli az i. akadályhoz tartozó *VO* térrésztől vett távolságot, $maxsebesseg$ a robot által elérhető maximális sebesség nagyságát, t pedig, hogy egy kiválasztott sebességvektorral mennyi ideig haladunk, mielőtt újra sebességet választanánk. Amennyiben meghatároztuk a *VO-távolságot* (összes *VO* távolság közül a legkisebb), melyet jelöljünk VO_{tav}-val, utána egy normálást hajtunk végre a (*maxsebesseg*t*)-vel való osztással annak érdekében, hogy egy [0-1] közötti értéket kapjunk, ezt nevezzük VO_{arany}-nak. Tehát a VO_{arany} kiszámítható:

$$VO_{arany} = \frac{VO_{tav}}{maxsebesseg * t} \tag{3.2}$$

Bevezethetünk egy költség értéket *(ktg_{ertek})*, amit minimalizálni szeretnénk, mely nem más, mint:

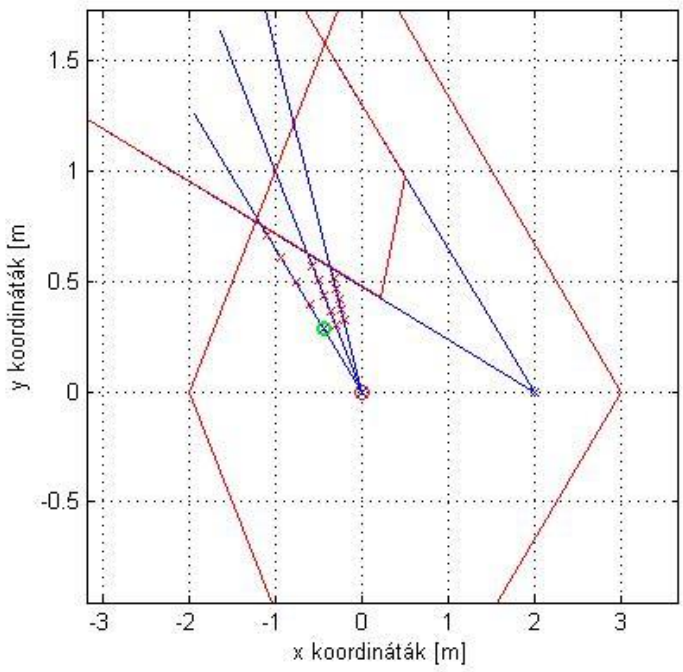
$$SVOktg_{ertek} = (1 - (VO_{arany})) \tag{3.3}$$

Ekkor minél kisebb a költség érték, annál jobb az adott sebesség vektor, hiszen annál távolabb helyezkedik el az összes *VO* térrésztől.

4.1 Implementáció

A sebességvektor kiválasztása vizsgálatánál felosztottam az elérhető vizsgált sebességkomponenseket (az *MV* stratégia által meghatározott irányokkal dolgozom) 5 részre és ezekre a komponensekre értékeltem ki a költség értékeket, majd kiválasztottam a legkisebbet, ez biztosítja a legbiztonságosabb mozgást a robot számára.

A 4-1. ábra szemlélteti, hogy a robotot piros kör jelzi, mely az origóban található, mint a 3.1. fejezetben bemutatott példa kezdetén is. Körülötte a 4 pont által közrezárt elérhető sebességek térrészét pirossal lett ábrázolva. Látható az eltolt *VO* térrész egy darabja is, felhasználva az időhorizontot. A robotból húzott 3 kék vonal jelzi az adott vizsgálandó sebességvektorokat. Piros x-ek jelölik a vizsgált sebességvektor komponenseket, zöld kör jelzi a legbiztonságosabb sebesség vektor végpontot, amely a legtávolabb helyezkedik el a *VO* térrészektől, ezáltal a legbiztonságosabb mozgást valósítja meg.

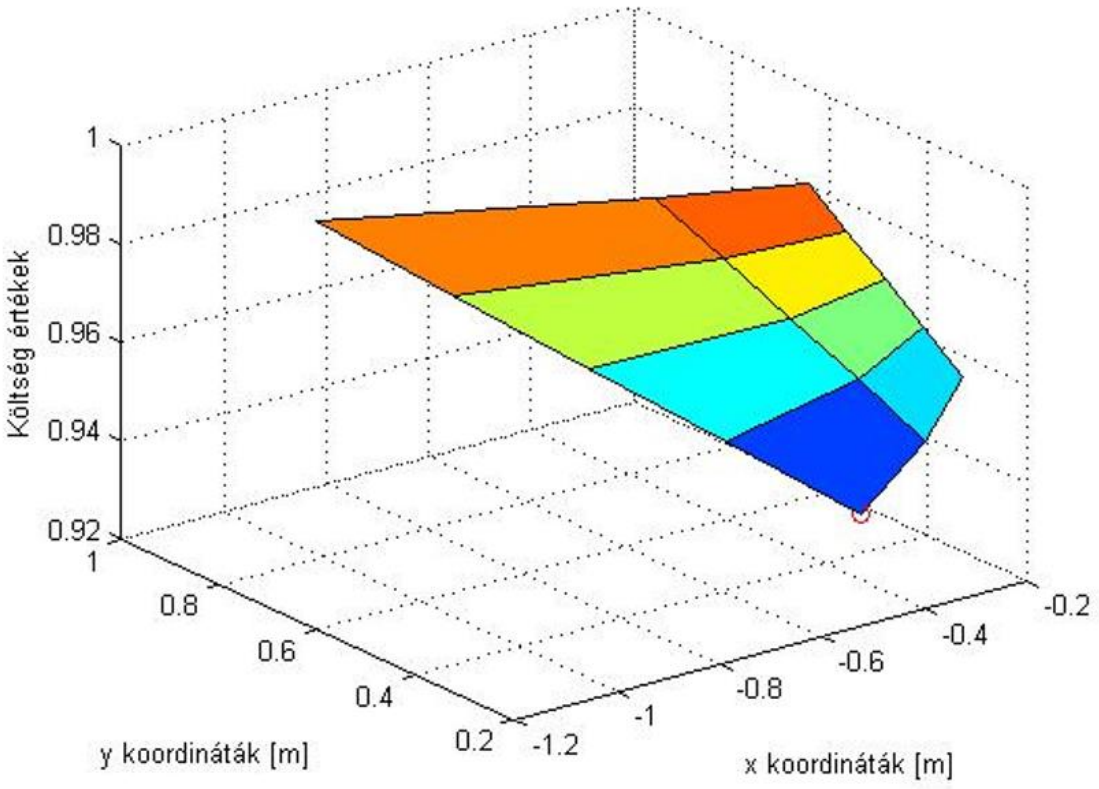


4-1. ábra: Sebesség választás *SVO* esetében

Látható, hogy jelen esetben a kiválasztott komponens a legkisebb nagyságú sebességkomponens a vizsgált elemek közül, viszont ez valósítja meg azt, hogy az adott akadályt a lehető legjobban elkerüljük (az *MV* stratégia által meghatározott szögtartományban), és úgy haladjunk a cél felé. Ebben az esetben úgy lehet felfogni, mintha az adott *VO* térrész (ezáltal az akadály mozgása) és a robot két pozitív mágneses pólus lenne, melyek taszítják egymást, a robot azt a sebességkomponenst választja ki, mellyel a leginkább elkerüli az akadályt.

A módszer kis mértékben hasonlít a potenciálmező alapú pályatervezési módszerre (*APF*-Artificial Potential Field). Ennél a módszernél az akadályok taszítják a robotot, kialakul minden akadály körül egy olyan tér, potenciálmező, ahol az akadályok körül nagy potenciálértékek jelennek meg [17]. Sok esetben ennél a módszernél egy lokális optimumban tud ragadni az algoritmus, erre is találtak megoldást [15]. Hosszú ideig a potenciálmező alapú pályatervezési módszert csak statikus környezetben használták, de a közelmúltban egyre több hatékony megoldás született a dinamikus környezetben történő hatékony mozgástervezésre is [16]. Van hasonlóság az *APF* és az *SVO* módszerek között, de az *APF* nem használja az akadályok sebességét.

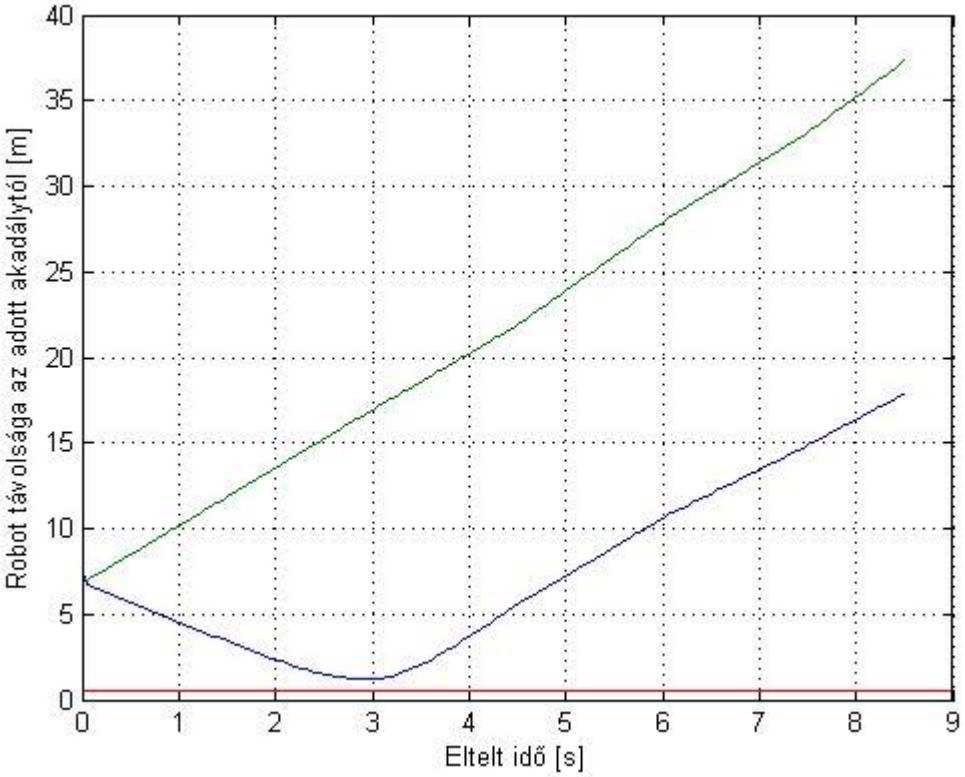
Visszatérve az *SVO* módszerre, minden robotsebesség választás esetén szemléletesebb a választás eredményének áttekintése, ha az adott sebességvektor x, y koordinátáját és a hozzá tartozó költség értéket egy 3 dimenziós ábrán szemléltetjük. Ezt mutatja be a 4-2. ábra, ahol a függőleges z tengely tartalmazza az adott pontokhoz tartozó költség értékeket. Látható, hogy a legkisebb költség értékhez tartozó sebességvektor végpont került kiválasztásra, ezt piros kör jelzi:



4-2. ábra: Adott sebesség komponensekhez tartozó költség értékek

Az *SVO* módszerrel is megvalósítottam a 3.1. fejezetben bemutatott mozgástervezés példáját. A legnagyobb különbségként azt lehet konstatálni, hogy míg a sima *VO* módszernél érintés történt az adott akadály és a robot között, ebben az esetben nem megy olyan „közel” az akadályhoz a robot, hanem elkerülve azt, jut el a célhoz.

Ebben az esetben is ábrázolhatjuk a robot és az akadályok távolságát az idő függvényében (ld. 4-3. ábra), a kék és a zöld görbe mutatja a szimulációban megvalósított 2 akadály távolságát a robottól. Itt a robot és az adott akadály középpontjának távolságából kivontam az akadály sugarát. A robot sugarát konstans piros vonal jelöli: (A mozgásról készült videó: https://youtu.be/r23dHkqEbIs)

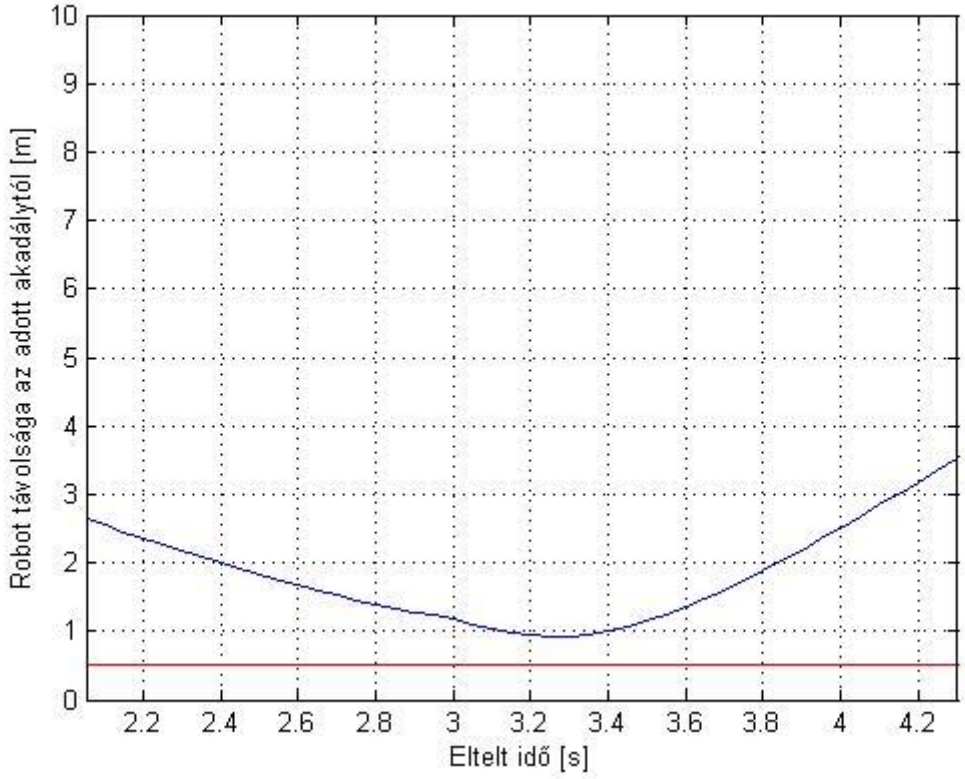


4-3. ábra: Robot és akadályok távolsága SVO esetén

Látható, hogy a távolság sohasem redukálódik olyan kicsire, hogy érintené a piros vonalat, ezáltal érintés következne be a felek között. Még amikor a legközelebb kerül a robot az akadályhoz, akkor is van kb. 1m távolság az adott akadálytól.

Érdemes ebben az esetben is megvizsgálni, hogy mi történik akkor, ha figyelembe vesszük a szenzorbizonytalanságot. Ennek érdekében ugyanolyan mértékben változtattam meg az adott akadály sebességvektorát, mint ahogy az előző példában (3.1. fejezet) láttuk. Abban az esetben azt tapasztalhattuk, hogy változtatva az akadály sebességén, már ütközés történt a robottal a mozgás során. Ám most lévén, hogy a legbiztonságosabb sebességkomponenst választottuk ki a robot számára, még a megváltoztatott akadálysebesség esetén sem történik ütközés, sőt érintés sem történik a felek között. Természetesen most közelebb kerülnek, mint a kezdeti esetben megvalósított *SVO* által számoltak alapján, de nem történik semmiféle kontaktus robot és akadály között (a mozgásról készült videó: https://youtu.be/33GIEhXWd8k)

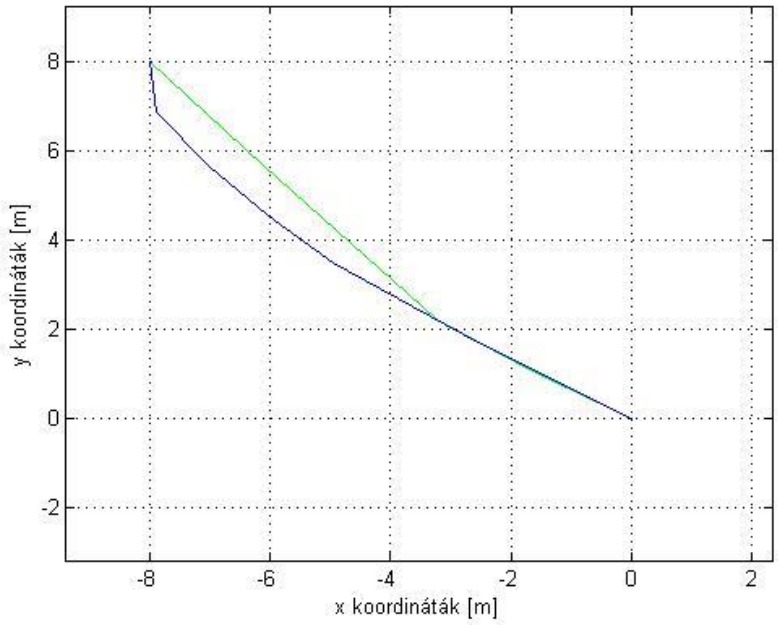
Ezt mutatja a 4-4. ábra ráközelítve:



4-4. ábra: Robot és akadály távolsága *SVO* módszerrel figyelembe véve a szenzorbizonytalanságot

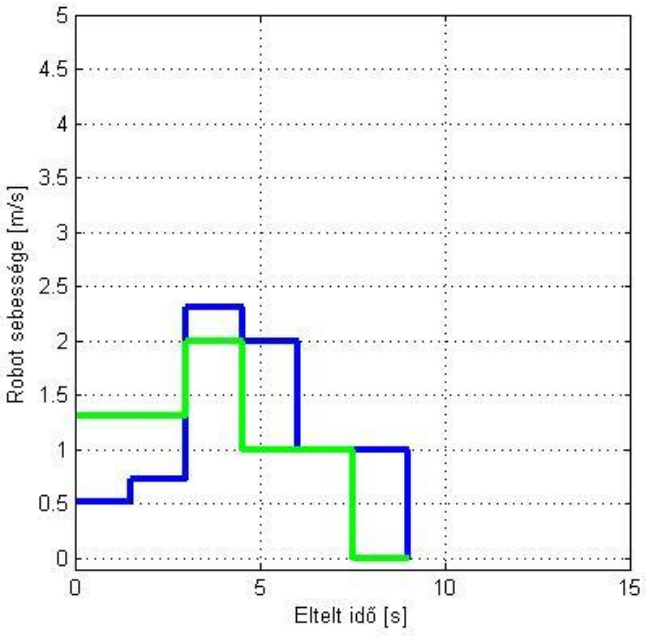
Tehát látható, hogy abban az esetben, ha az *SVO* módszert használjuk, még a szenzorbizonytalanság figyelembevételével is biztonságos pályatervezést tudunk megvalósítani. Ebben az esetben a megtett út lassabb, mint az eredeti az *VO* módszer esetén, mivel a tervezés kezdetén kisebb nagyságú sebességkomponenseket tudtunk csak választani, de ennek következtében biztonságban tartottuk a robot és környezete épségét.

Szemléletes képet ad a mozgásról, ha ábrázoljuk a robot pályáját mind a *VO*, mind az *SVO* stratégiák esetében, ezt mutatja a 4-5. ábra. A kék görbe az *SVO* stratégia által megvalósított mozgástervezés eredményét mutatja, míg a zöld a *VO* stratégia végeredményét. Látható, hogy míg a *VO* stratégia esetén majdnem egyenesen ment a robot a cél irányába, *SVO* stratégia esetén a biztonság érdekében egy kis kerülőt beiktatva érte el a célját.



4-5. ábra: Robot mozgásának pályája a célpozícióig *VO*, illetve *SVO* stratégia felhasználásával

Érdemes ábrázolni a két stratégia esetében az adott mintavételi időpillanatokban kiválasztott sebességvektorok hosszát.



4-6. ábra: Sebességek hosszának ábrázolása az idő függvényében *VO* és *SVO* módszer esetében

A 4-6. ábra által bemutatott kék görbe a *SVO* módszer által kiválasztott sebességek nagysága az idő függvényében, a zöld görbe pedig a *VO* stratégia által kiválasztott sebességnagyságok.

Látható, hogy kezdetben az SVO stratégia által megvalósított mozgástervezés esetén kisebb nagyságú sebességeket választ ki a robot a biztonság érdekében, ám, amikor „kitisztul a terep”, akkor már nagy sebességgel indul el a cél felé, 1 mintavételi időponttal később (jelen esetben t=1.5 s volt a mintavételi idő) éri el a célt, mint a VO stratégia által megvalósított mozgástervezés esetén.

Amennyiben egyszerre szeretnénk figyelembe venni a biztonsági és a gyorsasági tényezőket, akkor egy többkomponensű költségfüggvényt kell bevezetni.

4.2 Költségfüggvény meghatározása

A költségfüggvényben tehát egyszerre kell figyelembe venni az eredeti *VO* módszert (a gyorsaság érdekében) és az *SVO* módszert (a biztonság érdekében).

Először is figyelembe kell venni a távolság arányt, melyet úgy tudunk kiszámítani, hogy megvizsgáljuk, hogy az adott sebesség vektorral előre meghatározott időegyég alatt milyen közel kerülünk a célhoz, és ezt normáljuk az eredeti céltávolsággal, így egy [0,1] közötti számot kapunk (abban az esetben, ha a cél fele történik elmozdulás). Minél kisebb a távolságarány, annál közelebb jutunk a célhoz, annál jobb választás, ha a gyorsaságot vesszük fontos tényezőnek.

Tehát a távolságarány kiszámítása:

$$Tav_{arany} = \frac{táv(érkezett_{poz} - cél)}{celtav} \tag{3.4}$$

A (3.4) képletben Tav_{arany} jelöli a távolság arányt, a $táv(érkezett_{poz} - cél)$ azt fejezi ki, hogy az adott sebességkomponenssel ahova érkeznénk egy előre meghatározott idő alatt, milyen távolságban van a célpozíciótól. A $celtav$ a célpozíció távolsága a robottól kezdeti helyzetben.

Ha a biztonságot szeretnénk figyelembe venni a mozgástervezés során, akkor a költségfüggvényben teljes mértékben felhasználható a (3.3) egyenletben megfogalmazott $SVOktg_{ertek}$.

Bevezetve egy α aránytényezőt (értéke egy [0,1] közötti valós szám), mely megmondja, hogy melyik szempontot (gyorsaság, biztonság) milyen mértékben vegyük figyelembe számításkor, a (3.5) egyenletben megfogalmazott költségfüggvényt tudjuk felírni:

$$\text{ktgfv\_ertek} = \alpha * \text{Tav}_{\text{arany}} + (1 - \alpha) * SVOktg_{ertek} \tag{3.5}$$

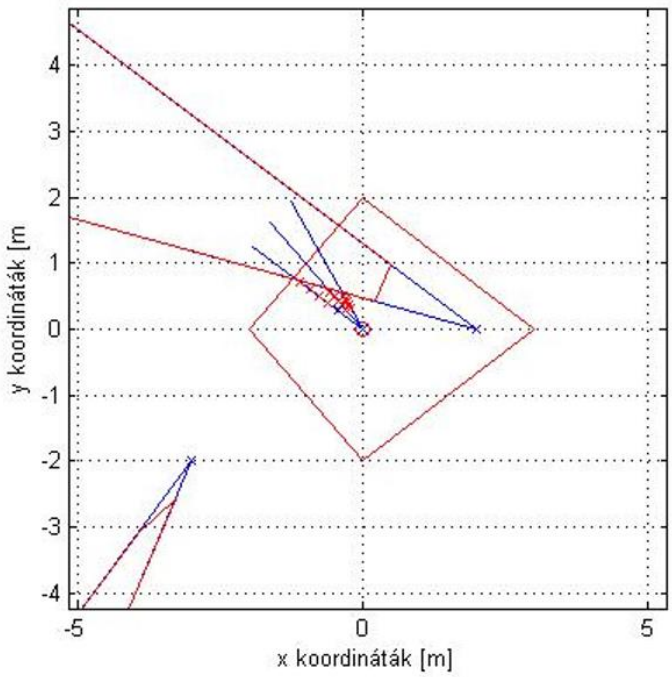
Abban az esetben, ha α értéke 0, akkor a legbiztonságosabb megoldást valósítjuk meg. Amennyiben α értéke 1, akkor pedig a leggyorsabb mozgástervezést érjük el.

Vizsgáljuk meg a különböző α értékek mellett milyen eredményeket kapunk!

4.2.1 Költségfüggvény értékei különböző α arányténező értékek mellett

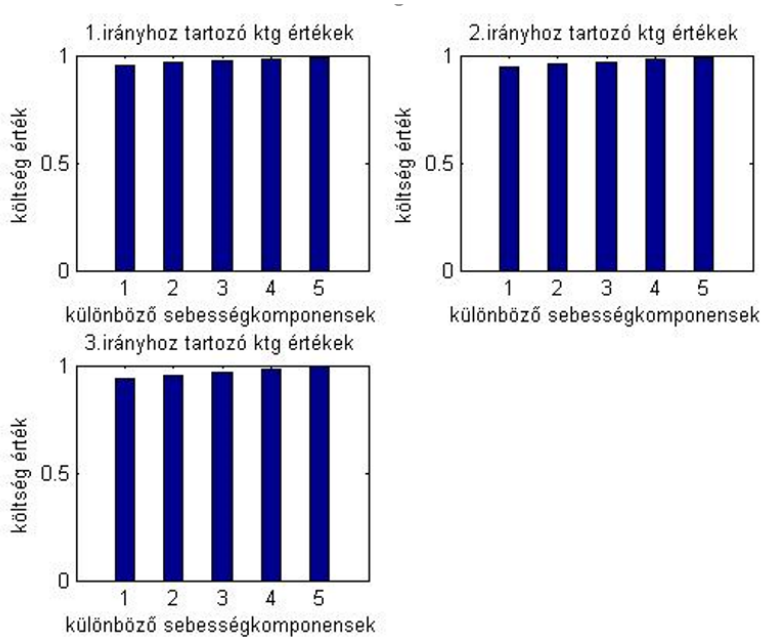
Amennyiben az arányténező 0.5, vagy ennél nagyobb értékű, akkor az algoritmus a vizsgált példában a leggyorsabb sebességkomponenst választja ki, azt a sebességvektort, mellyel adott időegység alatt legközelebb jutunk a célhoz.

Amennyiben 0 az arányténező, akkor a legbiztonságosabb sebességkomponenst választja ki az algoritmus, ahogy láthattuk a 4.1.-es fejezetben. Ám a vizsgált komponensek közel vannak egymáshoz, és közel vannak a legközelebbi *VO* térrészhez, ezt mutatja a 4-7. ábra.



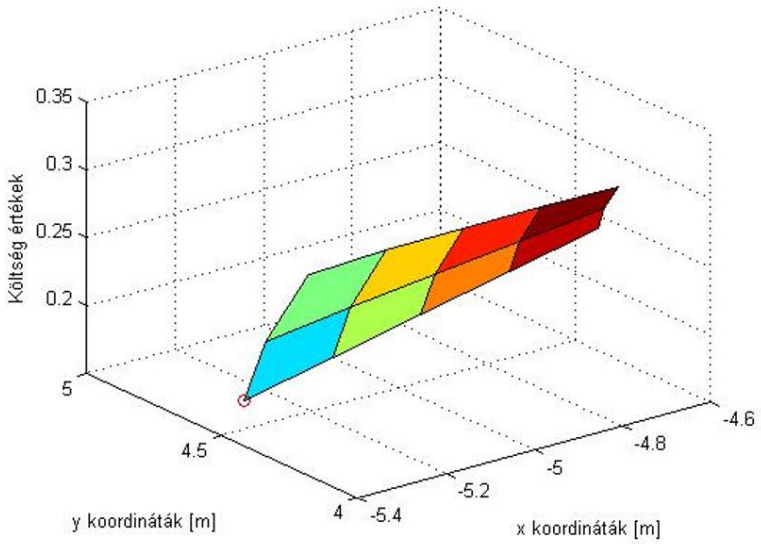
4-7. ábra: Vizsgált sebességkomponensek közel vannak egymáshoz, piros x-ek jelölik a vizsgált sebességvektor végpontokat

Ennek következtében, ha kiszámoljuk az $SVOktg_{ertek}$-eket a különböző komponensekre, akkor azt tapasztaljuk, hogy a normálás következtében is nagyon közel esnek egymáshoz (ld. 4-8. ábra):



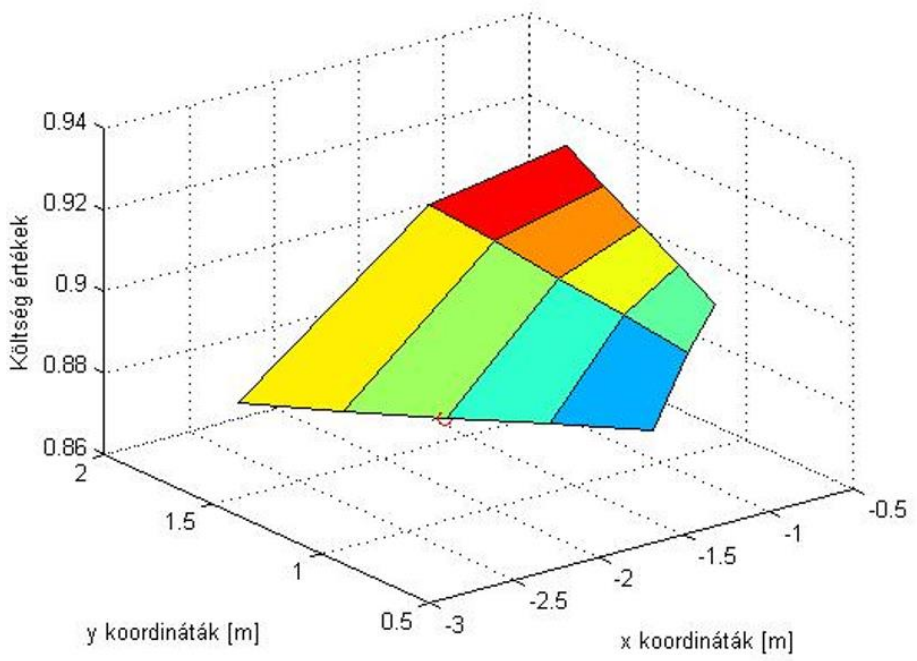
4-8. ábra: Különböző sebességkomponensek esetén az $SVOktg_{ertek}$ alakulása (három irányban, 5-5 különböző abszolútértékű sebességvektor vizsgálata)

Ha $\alpha = 0.4$ az aránytényező akkor előzetesen arra számítanánk, hogy legelső lépésben a legbiztonságosabb komponenst választja ki az algoritmus, de mivel az $SVOktg_{ertek}$-ek nagyon közel esnek egymáshoz, ezért ebben az esetben még felértékelődik, hogy az adott sebességvektorral milyen közel kerülünk a célhoz, ezért első lépésben a leggyorsabb komponenst választja ki az algoritmus, majd ezután már mindig a legbiztonságosabb, leggyorsabb komponenst:



4-9. ábra: Legnagyobb, legbiztonságosabb sebességkomponens kiválasztása, az aránytényező értéke: 0.4.

Az $\alpha = 0.3$ értékű aránytényezőnél először a legbiztonságosabb sebességet választja, majd utána azt tapasztaljuk, hogy a költségfüggvény minimuma éppen az egyik vizsgált sebességirány középső pontja (ld 4-10. ábra). Ahogy halad tovább a robot, közelebb érve a célhoz, a legbiztonságosabb irányban a legnagyobb abszolút értékű sebességkomponenst választja.



4-10. ábra: A költségfüggvény minimuma éppen az egyik vizsgált sebességvektor közepén található, az aránytényező értéke: 0.3

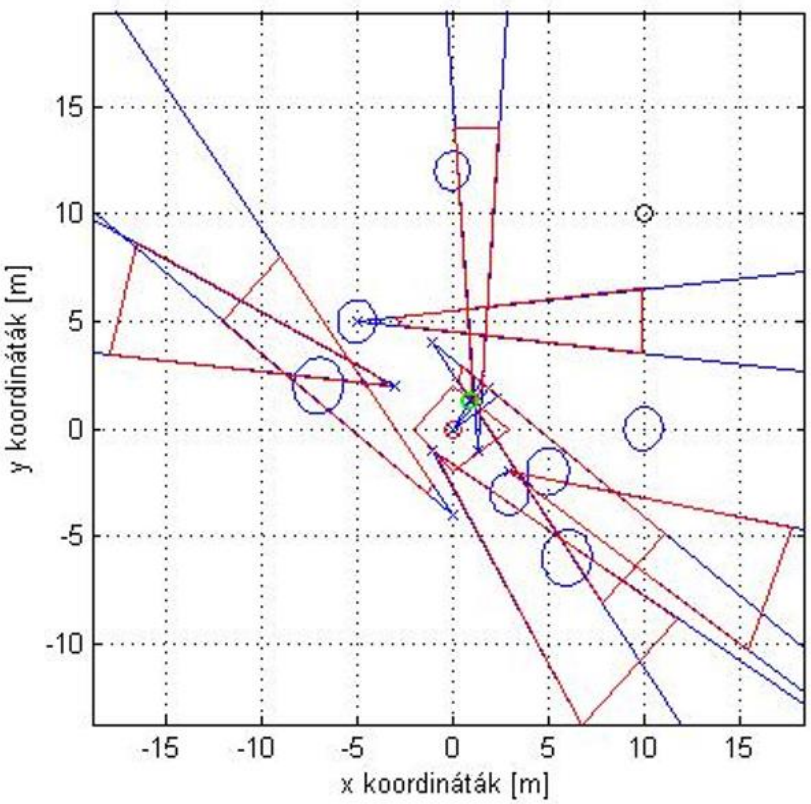
Az $\alpha = 0.3$ értékű aránytényező alatt már az első két esetben a legbiztonságosabb sebességet választja (kicsi abszolút értékű, elkerülő), majd utána ahogy közeledünk a célhoz a legbiztonságosabb irányból a legnagyobb komponenst választja ki. Később is a legbiztonságosabb komponenst választja ki, de ahogy elhaladtunk az akadály mellett, a legbiztonságosabb sebességkomponens már nem a legkisebb nagyságú, hanem a *VO*-tól legtávolabb eső legnagyobb sebességkomponens, ahogy az a 4-9. ábrán is látható.

5 Összetett mozgástervezés sok akadály esetén

A következő példában 7 mozgó akadály található a konfigurációs térben, így kell megvalósítania a robotnak a mozgástervezést.

A 3.1.-es fejezethez hasonlóan először az eredeti *VO* stratégia által megvalósított mozgástervezést mutatom be (aránytényező először $\alpha = 1$ értékű), majd a *SVO* megoldását szemléltetem (az aránytényező $\alpha = 0$ értékű), és hasonlítom össze a két megoldást. Ezt követően legvégül bemutatom a mozgástervezést abban a köztes esetben, ha az aránytényező, $\alpha = 0.5$ értékű.

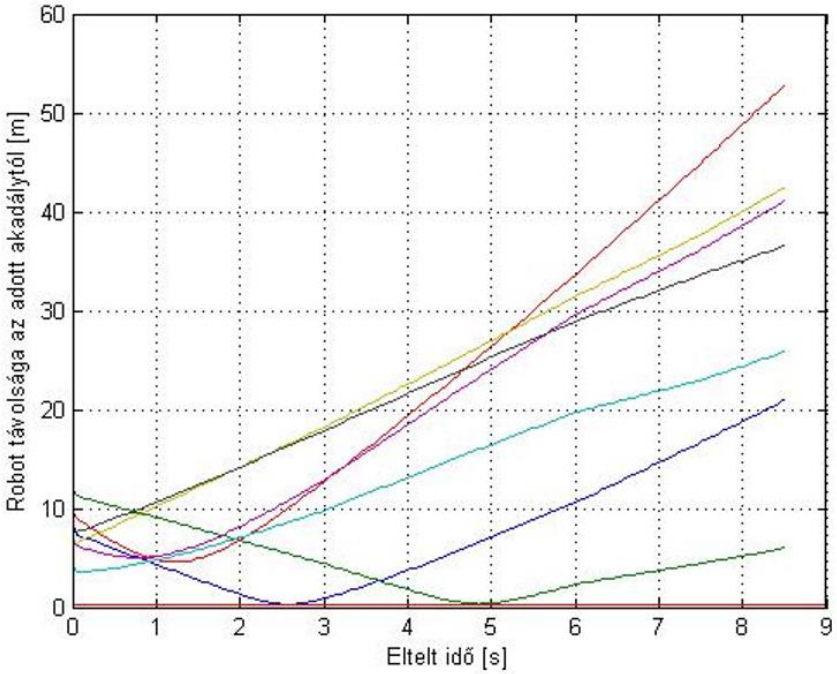
Az 5-1. ábra szemlélteti a hét akadályt (kék körökkel), a hozzájuk tartozó 4 pont által közrezárt zárt *VO* térrészek, fekete kör jelzi a (10,10) pozícióban elhelyezkedő célpozíciót. Zöld kör jelzi a kiválasztott sebességet.



5-1. ábra: Eredeti *VO* módszer sok akadály esetén

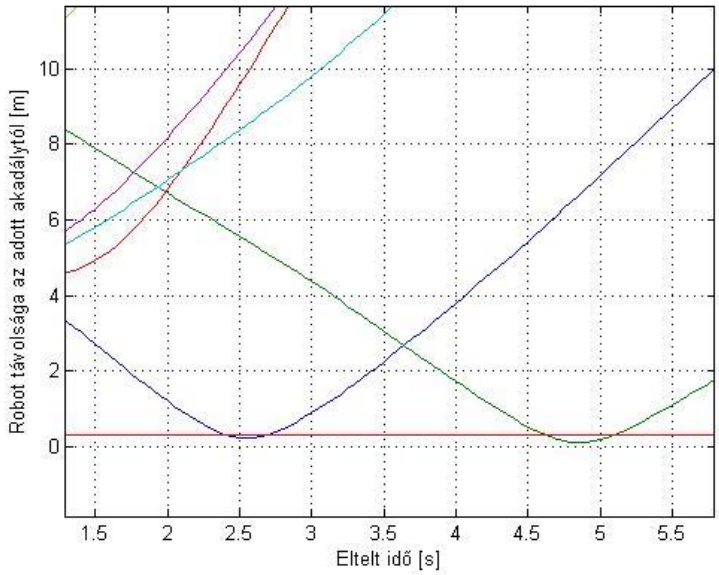
Mozgás során több akadály (3) is keresztezi a robot útját a célpozícióhoz. Két esetben történik a robot és az akadályok között érintés a leggyorsabb célelérés

érdekében, ezt mutatja be a 5-2. ábra (a mozgásról készült videó: https://youtu.be/2UrkGUCmEyQ):



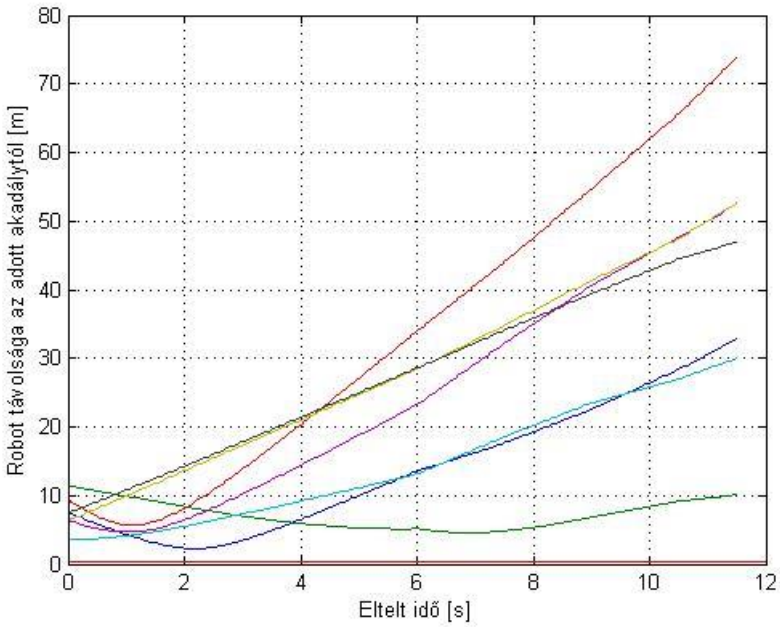
5-2. ábra: Sok akadály esetén az akadályoktól való távolság VO stratégia esetén.

Piros konstans vonal jelzi a robot sugarát, a különböző színű görbék az egyes akadályok középpontjának távolságát a robot középpontjától, ebből kivonva mindig az adott akadály sugarát. Érintés történik akadály és robot között, ha a görbe érinti a piros vonalat. Ám ha figyelembe vesszük a szenzorbizonytalanságot, akkor a 2 akadállyal történő érintés esetén azonnal ütközés történik. Látható az 5-3. ábra, mely bemutatja, hogy az adott görbék a piros vonal alá mennek, jelezve az ütközést (a mozgásról készült videó: https://youtu.be/HbUAmd6SL6A):



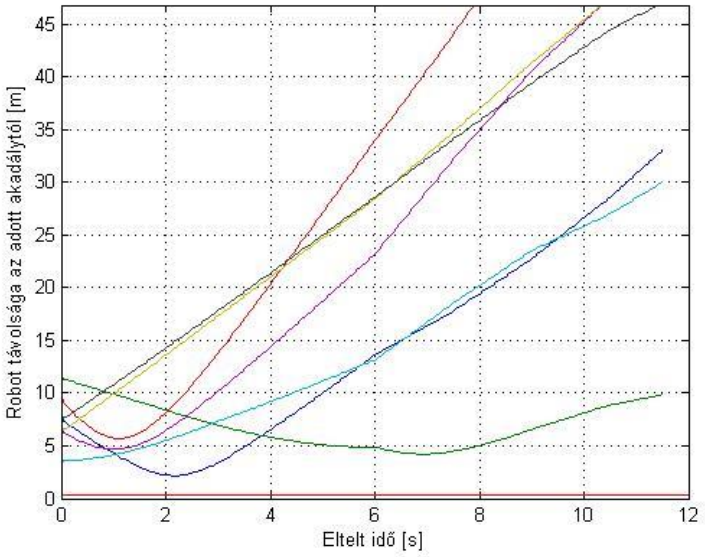
5-3. ábra: A szenzorbizonytalanság figyelembevételével már ütközést történik két akadállyal is

Az *SVO* módszer esetén (a legbiztonságosabb mozgást valósítjuk meg, az aránytényező értéke $\alpha = 0$) a legbiztonságosabb sebességkomponenst választja ki a robot, ennek függvényében még érintés sem történik az akadályok és a robot között. Lassabban indul el a robot, nevezhetjük biztonsági játékosnak is. Ebben az esetben a robot távolsága az akadályoktól az 5-4. ábra által figyelhető meg (a mozgásról készült videó: https://youtu.be/g6CyageUR7k):



5-4. ábra: Akadályok távolsága a robottól *SVO* módszer esetén

Ha figyelembe vesszük szintén a szenzorbizonytalanságot, abban az esetben is ütközésmentes mozgástervezést valósítunk meg (ld. 5-5. ábra) (a mozgásról készült videó: https://youtu.be/_2REp_n3cmU):

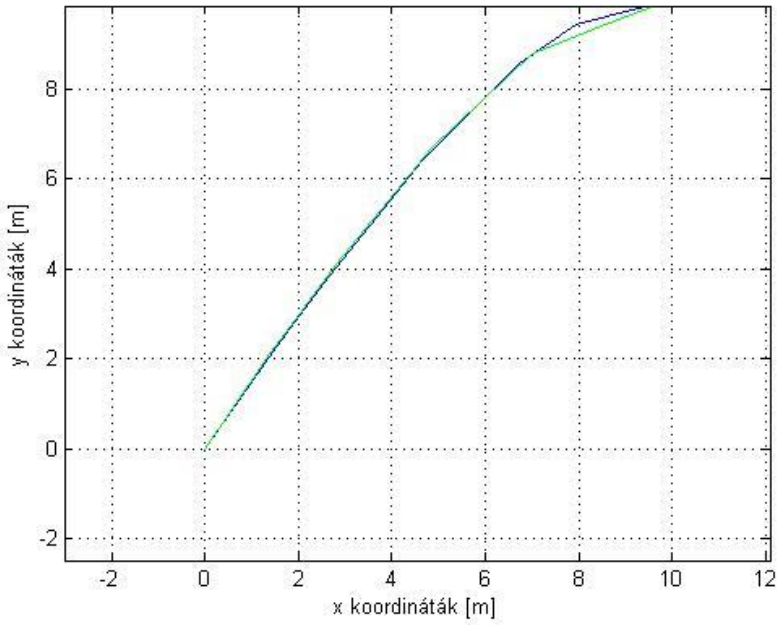


5-5. ábra: Akadályok távolsága a robottól SVO módszer esetén figyelembe véve a szenzorbizonytalanságot

Az *SVO* módszer esetén a robot megvárja, míg a sűrű terep kicsit kitisztul, addig csak kis sebességekkel halad a cél irányába, majd amikor már az akadályok nem veszélyeztetik az útját a cél felé, akkor nagyobb sebességgel indul a kívánt pozíció felé. Előnyös ez a mozgástervezés, mert még a szenzorbizonytalanságok figyelembevételével is biztonságos mozgástervezést tudunk megvalósítani a robotunk és környezete számára. Ám mindennek „ára van", ebben az esetben több ideig tart a mozgás, a robot az előző példában 20%-kal később érte el a kívánt célpozíciót, a mozgásának útja viszont „csak" 2 %- kal volt hosszabb. Tehát ha nem az időben leggyorsabb célelérést tartjuk legfontosabb prioritásként szem előtt a mozgástervezés során, akkor mindenképpen érdemesebb ezt a *Safety Velocity Obstacles (SVO)* választani a robot trajektóriájának meghatározásához. A valóságban is sokszor előfordulhat, hogy nem a leggyorsabb útvonal a legfőbb prioritás, hanem a legbiztonságosabb útvonal, például, ha gyúlékony, robbanékony anyagot szállítunk, vagy amennyiben sérülékeny, nagy értékű tárgyat vitetünk a robotunkkal.

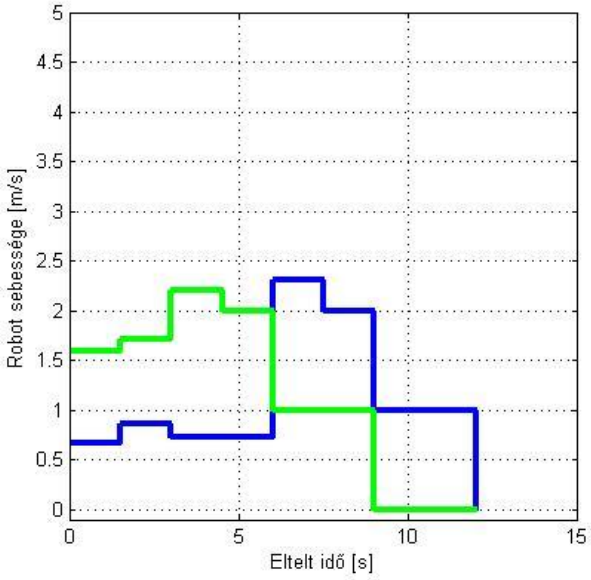
Ezen mozgástervezés esetén is, mint a 4.1. fejezetben, érdemes ábrázolni a robot mozgásának pályáját, illetve a sebességeinek nagyságát az idő függvényében, mind a *VO*, mind az *SVO* stratégiák esetében.

Az 5-6. ábra szemlélteti a robot mozgásának pályáját. Kék görbe mutatja az *SVO*, zöld görbe a *VO* stratégia által történt mozgástervezést. Jelen esetben kis mértékben tér el csak a két görbe (a mozgástervezés végén). Ez annak a következménye, hogy a munkatérben sok akadály volt a robot pályájának mindkét oldalán, így a legbiztonságosabb sebességtényező a 3 irány közül mindig a középső volt, amely a cél irányába vezetett, így majdnem ugyanazon az útvonalon jutott el a robot a célig, mint *VO* stratégia esetén, csak több idő alatt.



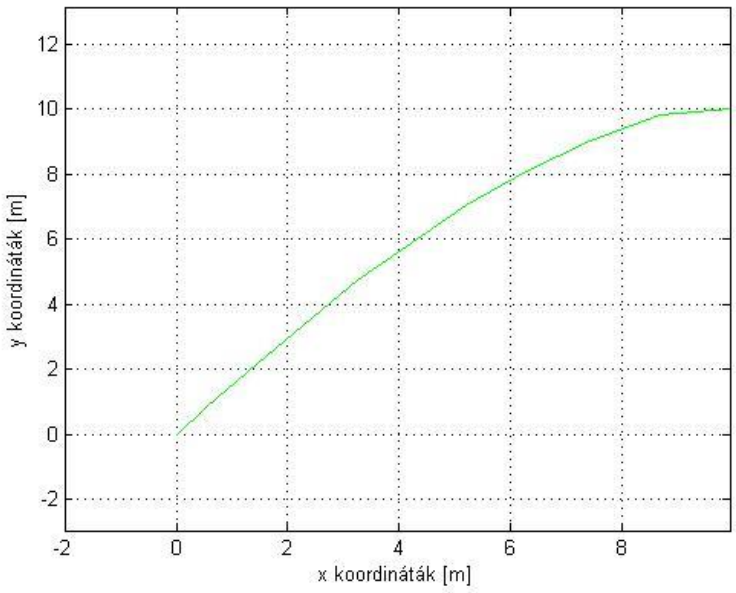
5-6. ábra: Robot mozgásának a pályája SVO és VO stratégiák esetén

Az 5-7. ábra a sebességnagyságot mutatja az idő függvényében az *SVO* (kék görbe) és *VO* (zöld görbe) stratégiák esetén. Kezdetben az SVO stratégia sokkal lassabb, kisebb nagyságú sebességvektorokat választott a robotnak a biztonság érdekében. Majd, amikor „kitisztult a tér”, akkor már gyorsabb sebességvektorokat választva érkezik el a célhoz a robot. A mintavételi idő t=1.5 s volt a példában.



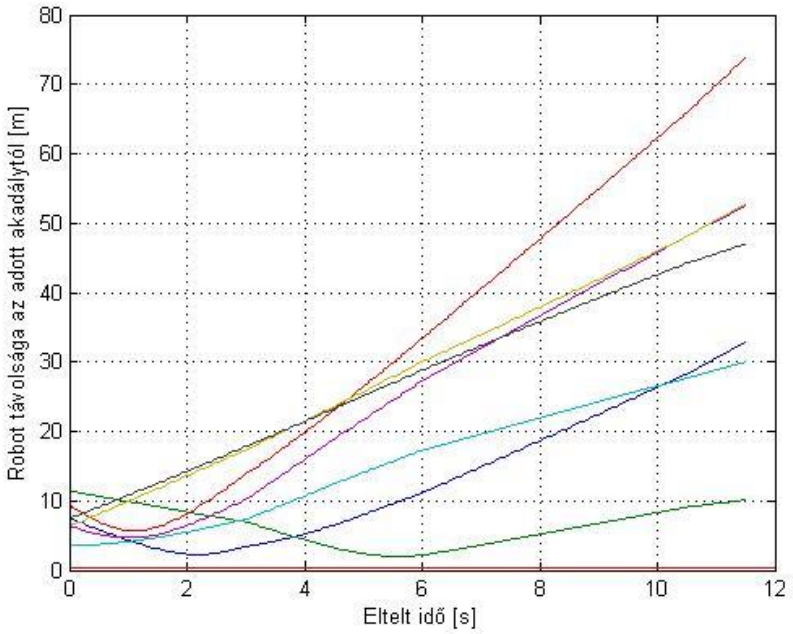
5-7. ábra: Robotsebességek nagysága az idő függvényében VO illetve SVO stratégiák esetében

Végezetül vizsgáljuk meg azt az esetet, amikor az aránytényező, $\alpha = 0.5$ értékű. Ekkor a robot mozgástervezésében ugyanolyan mértékben vesszük figyelembe a gyorsaságot, mint a biztonságot. Az 5-8. ábra mutatja a robot mozgásának pályáját. Látható, hogy majdnem megegyezik ez a pálya a 5-6. ábra által bemutatott mozgástervezések eredménye által kapott pályákkal, melyeknél az aránytényező: $\alpha = 0$, illetve $\alpha = 1$ volt.



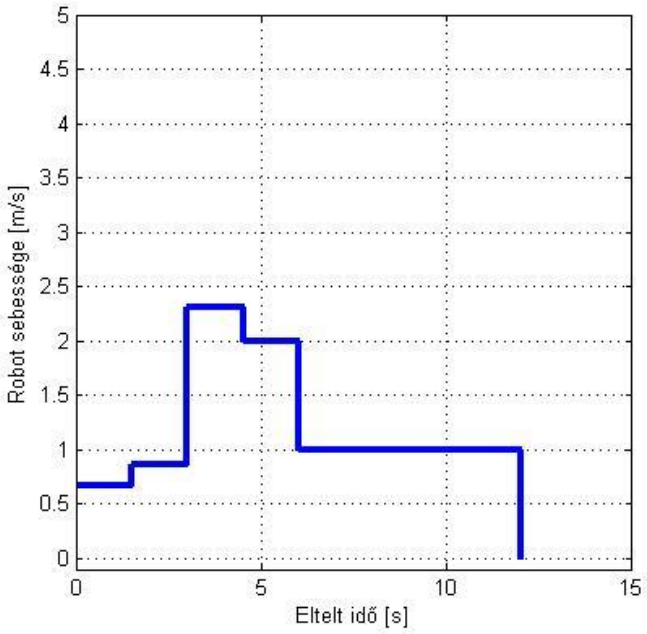
5-8. ábra: A robot mozgásának pályája abban az esetben, ha az aránytényező értéke $\alpha = 0.5$

A robotnak az akadályoktól vett távolságát az 5-9. ábra szemlélteti, figyelembe véve a szenzorbizonytalanságot. Ugyanúgy, mint a legbiztonságosabb mozgástervezés esetén ($\alpha = 0$), most sem történik ütközés a robot és az akadályok között.



5-9. ábra: A robotnak az akadályoktól vett távolsága, figyelembe véve a szenzorbizonytalanságot, ha az aránytényező értéke $\alpha = 0.5$

Érdemes ebben az esetben is ábrázolni a robotsebességek nagyságát az idő függvényében (ld. 5-10. ábra). Itt tapasztalhatjuk a legnagyobb eltérést a korábban bemutatott *VO* és *SVO* stratégiákhoz képest. Ebben az esetben, mivel ugyanolyan mértékben vesszük figyelembe a gyorsaságot és a biztonságot, kezdetben a legbiztonságosabb sebességkomponenst választja ki a robot, de mozgása során korábban kezd el gyorsítani, mint amit az 5-7. ábra által bemutatott SVO stratégia esetén láthattunk. Ezáltal azt tapasztalhatjuk, hogy a mozgás kezdetén a legbiztonságosabb sebességvektort választja ki a robot, amíg távol vagyunk a céltól, majd ugyanolyan fontos szerepet játszik a gyorsaság is, tehát ezt követően már nagy abszolút értékű sebességeket választ a robot mozgása során.



5-10. ábra: Robotsebességek nagysága az idő függvényében, ha az aránytényező értéke α =0.5

6 Értékelés, továbbfejlesztési lehetőségek

Munkám során egy mozgástervező algoritmust fejlesztettem ki és implementáltam, mely alkalmas mobilis robotok mozgástervezésére dinamikus környezetben. Ehhez a mozgástervező algoritmushoz alapvetően a *Velocity Obstacles* módszert használtam fel, mely egyszerre térképezi fel a változó környezetet a robot sebességterével, és így határozza meg a pálya geometriáját a sebességprofillal együtt.

Az eredeti *VO* módszer minden esetben a leggyorsabb célelérést valósította meg, melynek következtében amennyiben a robot cél felé történő mozgása során egy akadály keresztezte a mozgó robot útját, akkor mindig érintés történt a felek között. A szenzorbizonytalanság figyelembe vételével érintésből nyomban ütközés keletkezett. Tehát ez a módszer abban az esetben használható, ha nagyon pontos szenzor értékek állnak rendelkezésre a környezetről, akadályok sebességeiről és azok pozíciójáról.

A szenzorbizonytalanság által okozott hiba kiküszöbölésére teljes mértékben megoldásul szolgált a *Safety Velocity Obstacles (SVO)* módszer, melynek során minden egyes sebességválasztás esetén a legbiztonságosabb sebességkomponens került kiválasztásra. Ennek következtében nem történt érintés sem a robot sem az adott akadály között. A szenzorbizonytalanság figyelembevételével is ütközésmentes mozgástervezést tudtunk megvalósítani. A pálya bejárásához több időre volt szükség, de amennyiben a pályatervezés esetén a robot mozgásának időtartama nem számít olyan fontos tényezőnek, akkor érdemes az *SVO* módszert választani a biztonság érdekében.

A költségfüggvény bevezetésével figyelembe lehet venni azt is, hogy a robot mozgása során milyen preferenciával rendelkezzen, azaz a biztonságot, vagy a gyorsaságot tartsa-e fontosabb tényezőnek.

A későbbiekben továbbfejlesztésként figyelembe lehetne venni a szenzorbizonytalanságot, megnövelve a *VO* ütközést okozó térrészeket egy ε sugarú környezettel. Érdemes lenne megvizsgálni a mintavételi idő nagyságának hatását a módszerekre. Az algoritmust tovább lehetne fejleszteni téglalap alakú akadályok és robot esetére is, közelítve, hogy autószerű járművekről beszélhessünk. Több robot esetére is ki lehetne terjeszteni a módszert.

Ez a mozgástervezési algoritmus a jövőben nagy lehetőségként szolgálhat az önvezető autók mozgástervezése esetében is, illetve az otthoni háztartásban is felhasználható lehet robotok pályatervezésére, akár idősek segítésére kifejlesztett robotok esetén.

Köszönetnyilvánítás

Hálás köszönettel tartozom Gincsainé Dr. Szádeczky-Kardoss Emesének, konzulensemnek, aki minden alkalommal idejét és energiáját rám áldozva tanácsaival ellátott, mindig támogatott, inspiráló közeget biztosított az előrehaladás érdekében.

Köszönettel tartozom szüleimnek is, akik szüntelenül biztosítják a lehetőséget a tanulásra, fejlődésre. Szeretetükkel és gondoskodásukkal mindig támogatnak, amelyért hálás a szívem.

Irodalomjegyzék

1. Paolo Fiorini and Zvi Shiller: Motion Planningin Dynamic Environments using Velocity Obstacles
   The International Journal of Robotics Research July 1998 vol. 17 no. 7 760-772.
2. Jamie Snape, Jur van den Berg, Stephen J. Guy, and Dinesh Manocha: The Hybrid Reciprocal Velocity Obstacle, IEEE TRANSACTIONS ON ROBOTICS, VOL. 27, NO. 4, AUGUST 2011.
3. Canny, J. and Reif, J. (1987). New lover bound techniques for robot motion planning problems. In 28th IEEE Symposium on Foundation of Computer Science.
4. Reif, J. and Sharir, M. (1985). Motion planning in the presence of moving obstacles. In 25th IEEE Symposyum on Foundation of Computer Science, pages 144-153.
5. Fujimura, K. and Samet, H. (1989b). Time-minimal path among moving obstacles. In IEEE International Conference on Robotics and Automation, pages 1110-1115,Scottsdale, AZ.
6. Kant, K. and Zucker, S. (1986). Towards efficient trajectory planning the path-velocity decomposition. The International Journal of Robotic Research, 5(3):72-89.
7. Lee, B and Lee, C. (1987). Collision-free motion planning of two robots. IEEE Transactions on System Man and Cybernetics, SMC-17(1): 21-32.
8. Gyenes Zoltán: Mobilis robotok pályatervezése dinamikus környezetben, Budapesti Műszaki és Gazdaságtudományi Egyetem, BSc Szakdolgozat 2016. december
9. D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, “Randomized kinodynamicmotion planning withmoving obstacles,” *Int. J. Robot. Res.*, vol. 21, . 3, pp. 233–255, Mar. 2002.
10. L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilisticroadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
11. J. J. Kuffner Jr and S. M. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, Apr. 2000, vol. 2, pp. 995–1001.
12. D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
13. S. Petti and T. Fraichard, “Safe motion planning in dynamic environments,” in *Proc. IEEE RSJ Int. Conf. Intell. Robot. Syst.*, Edmonton, AB, Canada, Aug. 2005, pp. 2210–2215.
14. https://en.wikipedia.org/wiki/Minkowski_addition
15. Qidan Zhu, Yongjie Yan, and Zhuoyi Xing: Robot Path Planning Based on Artificial Potential Field Approach withSimulated Annealing, Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06) 0-7695-2528-8/06 2006 IEEE

16. Cao Qixin, Huang Yanwen, Zhou Jingliang: An Evolutionary Artificial Potential Field Algorithm for Dynamic Path Planning of Mobile Robot, Proceedings of the 2006 IEEE/RSJ,International Conference on Intelligent Robots and Systems,October 9 - 15, 2006, Beijing, China
17. https://www.youtube.com/watch?v=r9FD7P76zJs
18. Frederic LArge, Sepanta Sekhavat, Zvi Shiller and Christian Laughier: Using Non-Linear Velocity Obstacles to Plan Motions in a Dynamic Environment, Senventh International Conference on Control,Automation, Robotics And Vision (ICARCV'02),Dec 2002, Singapore

Függelék

1. A robot mozgása *VO* stratégia esetén (2 akadály): https://youtu.be/rLcTp-10dEM
2. A robot mozgása *VO* stratégia esetén, figyelembe véve a szenzorbizonytalanságot (2 akadály): https://youtu.be/_oTQ5p2-IMQ
3. A robot mozgása *SVO* stratégia esetén (2 akadály): https://youtu.be/r23dHkqEbIs
4. A robot mozgása *SVO* stratégia esetén, figyelembe véve a szenzorbizonytalanságot (2 akadály): https://youtu.be/33GIEhXWd8k
5. A robot mozgása *VO* stratégia esetén (7 akadály) https://youtu.be/2UrkGUCmEyQ
6. A robot mozgása *VO* stratégia esetén, figyelembe véve a szenzorbizonytalanságot (7 akadály) https://youtu.be/HbUAmd6SL6A
7. A robot mozgása *SVO* stratégia esetén (7 akadály) https://youtu.be/g6CyageUR7k
8. A robot mozgása *SVO* stratégia esetén , figyelembe véve a szenzorbizonytalanságot (7 akadály) https://youtu.be/_2REp_n3cmU