Univerza v Ljubljani
Fakulteta za elektrotehniko

AVTONOMNI MOBILNI SISTEMI

Gregor Klančar

Ljubljana, 2013

Kazalo

Poglavje 1

Uvod v mobilne sisteme

Mobilni sistemi so zmožni premikanja (lokomocija) v okolju in niso fiksno vpeti v okolje. Mobilni sistemi je avtonomen, če je zmožen samostojnega delovanja v okolju, tako iz stališča energije (ima lasten vir energije), kot tudi odločitev in izvajanja akcij. V stvarnem okolju so mobilni sistemi mobilni roboti, ki so zmožni premikanja po tleh (kolesni robot), zraku (letala, rakete), vesolju (sateliti), vodi (podmornica) in podobno.

V Slovarju slovenskega knjižnega jezika najdemo pomen besede robota in robot. Starinski pomen besede robota se je uporabljal za izvajanje težkih del ("robota v rudniku jih je izčrpala"). Beseda robot pa je opisana kot elektronska naprava, ki enakomerno opravlja vnaprej programirana, pogosto človekovemu zdravju škodljiva dela.

Strokovno, pa je bila beseda robot prvič uporabljena v drami R.U.R - Razumni Univerzalni Roboti (Karel Čapek, 1920). Drama se odvija v tovarni, kjer izdelujejo umetne ljudi - robote, ki kmalu začno ogrožati človeštvo. Z delom avtor izrazi odpor do hitrega napredka, masovne proizvodnje in dehumanizaciji človeka.

V knjigi je nakazanih nekaj ključnih podpodočij, ki jih mobilni sistem mora vključevati pri svojem delovanju. Mobilni sistem, ki opravlja neko delo v okolju, mora imeti informacijo, kje v okolju se nahaja. Ta informacija je lahko podatek o poziciji in orientaciji, kar skupaj označuje *lego* sistema. Lego sistema in zaznavanje okolja (npr.: razdalja do ovir) sistem ocenjuje s pomočjo senzorjev, katerih informacija je bolj ali manj pošumljena (ni deterministična). Tu uporabljamo različne pristope *lokalizacije*, ki vključuje obdelavo in združevanje informacij iz različnih senzorjev za ocenjevanje lege sistema. Marsikdaj le podatek o legi sistema ni dovolj in potrebna je tudi informacija o samem okolju v katerem se sistem nahaja. Ta informacija je lahko podana z opisom okolja v obliki *zemljevida okolja* (zemljevid cest, sten v stavbi, ovir, itd.). Slednji je lahko v sistem vgrajen ali pa ga mora

mobilni sistem zgraditi in dopolnjevati sam. Postopek gradnje zemljevida imenuje kartiranje (angleško mapping). Marsikdaj pa mora sistem hkrati izvajati lokalizacijo in kartiranje, kar označuje kratica SLAM (Simultaneous localization and mapping).

Pridobivanje informacije o legi sistema za namen usmerjanja sistema proti cilju imenujemo navigacija, ki je v mobilnih sistemih ključnega pomena. S pomočjo zemljevida okolja in znane lokacije mobilnega sistema lahko sistem določi *planira* pot do cilja (trajektorijo) oziroma korake za izvedbo določene naloge. Sistem mora imeti tudi stabilne in učinkovite strategije vodenja, da lahko sledi načrtovani trajektoriji.

Pri lokalizaciji in vodenju sistema uporabljamo zanje o modelu sistema, ki je lahko podano s pomočjo kinematičnega in dinamičnega modela sistema.

Kadar pa imamo opravka z delovanjem več posameznih mobilnih sistemov oziroma *agentov* v nekem okolju, pa govorimo o *večagentnem sistemu*.

1.0.1 Razvrstitve mobilnih sistemov

Izmed številnih možnih delitev omenimo le nekaj osnovnejših. Ločimo sisteme **glede na teren** v katerem delujejo:

- Kopenski roboti oziroma kopenska brezpilotna vozila (UGV- Unmanned Ground Vehicles). Večinoma so s kolesnim pogonom, gosenicami, pa tudi humanoidni (dvonožni) ali večnožni roboti.
- Zračni roboti oziroma brezpilotna zračna plovila (UAV - Unmanned Aerial Vehicle).
- Podvodni roboti ali tudi avtonomna podvodna vozila (AUV- Autonomous Underwater Vehicles).

Mobilni sistemi imajo možnost **lokomocije** (premikanja), torej imamo sisteme s

- kolesnim pogonom, kar je v tehniki zelo pogosto,
- nogami (humanoidni roboti in ostali),
- gosenicami,
- krili za letenje,
- plovno konstrukcijo,...

V naravi srečamo vrsto možnih načinov premikanja, ki pa jih je v tehniki težko implementirati (hoja, skakanje, plazenje, drsenje/vijuganje, letenje). V tehniki pa imamo večinoma kolesni pogon, ki pa ga v naravi ne najdemo, čeprav je energijsko učinkovito in enostavno za izvedbo.

Mobilni sistemi imajo nek **namen uporabe**, zato lahko ločimo:

- industrijske in delovne robote (industrijski manipulatorji, roboti v kmetijstvu,...),
- domače ali hišne robote (sesalci, kosilnice),
- medicinski roboti (pomoč pri operacijah),
- servisne in strežne roboti (pomoč ostarelim, prenos lažjih bremen, hrane,...),
- vojaški in policijski roboti,
- zabavni roboti in
- raziskovalni roboti .

Nadalje lahko ločimo mobilne sisteme glede na njihovo **stopnjo avtonomije**, kjer imamo sisteme z daljinskim upravljanjem in vse do popolnoma avtonomnih robotov.

1.0.2 Zgradba mobilnih sistemov

Mobilne sisteme namenjene praktični uporabi imenujemo tudi mobilni roboti. Sestavljeni iz mehanskih delov in elektronike. Mobilni robot tako tipično vsebuje:

mehanska konstrukcija : kovinska, lego kocke in podobno ter mehanska izvedba pogona (kolesa, gosenice, noge,...),

aktuatorski pogon : motorni pogon (DC, koračni, servomotor,...)

senzorji : meritev zasuka koles, razdalje do ovir, globalni navigacijski sistem (GPS)...,

računalnik : mikrokrmilnik, prenosni osebni računalnik,...

sistem napajanja : akumulatorsko napajanje, sončne celice,...

elektronika : elektronika za pogon motorjev (pulznoširinsko upravljanje, ojačevalniki,...), telekomunikacijska elektronika,...

algoritmi za nižjenivojsko upravljanje : regulacija hitrosti, sledenje poti, fuzija in filtriranje senzorske informacije, obdelava slike,...

navigacija : ocenjevanje lege in planiranje premikov,

komunikacija : s človekom ali drugim sistemom,

inteligenca : vodenje na višjem nivoju, sprejemanje odločitev, strategija delovanja, učenje.

1.0.3 Motivacija in uporaba

Motivacija za razvoj mobilnih sistemov

Pomemben razlog za razvoj mobilnih robotov je človeška lenoba, ki jo lahko smatramo za gonilo napredka. Človeka namreč monotona, ponavljajoča dela opravlja z odporom, zato želimo marsikaj avtomatizirati in/ali robotizirati.

Mobilni roboti omogočajo človeku dostop no nevarnih okolij (minska polja, radioaktivna okolja, morske globine) in dostop do preveč oddaljenih ali nedostopnih okolij (Mars, nanoroboti v medicini).

Uvedba avtomatizacije, robotizacije in mobilnih sistemov pa omogoča tudi večjo produktivnost, kvaliteto (izdelkov ali storitev) in zmanjšanje stroškov dela.

Uporaba mobilnih sistemov

Mobilni sistemi se uporabljajo v številnih aplikacijah na različnih področjih, ki se stalno širijo zaradi hitrega razvoja tehnike na tem področju. Naštejmo nekaj možnih aplikacij:

- medicinske storitve, pomoč pri operacijah, opravljanje laboratorijskih analiz (npr. kjer je nevarnost okužbe),
- čiščenje raznih površin, večje površine (letališča, tovarniške halje), okna, sesanje tal v domovih,...
- dela v kmetijstvu, avtomatizirani obiralci sadja, sajenje, košnja,...
- gozdna dela, čiščenje gozdov,...
- prodaja blaga široke potrošnje,
- pregledovanje nevarnih področij (detekcija in deaktivacija min na minskih poljih, pregled jedrskih reaktorjev, roboti za čiščenje kanalizacijskih cevi)

- vesoljski roboti (sateliti, roboti za pregledovanje in servis satelitov, raziskovanje planetov)
- globine morja (roboti za polaganje kablov, pregledovanje morskega dna)
- roboti za natovarjanje in raztovarjanje blaga ali materiala (letala, ladje, kamionov)
- vojaški roboti (izvidniški roboti, letala in razni avtopilotski izstrelki)
- varnostni roboti (varnostniki za nadzor skladišč, zgradb,...)
- pomoč ostarelim in hendikepiranim (avtonomni invalidski vozički, rehabilitacijski roboti),
- zabavne aplikacije (robotski ljubljenčki, robotski nogomet,...),
- mobilni sistemi v raziskovalnih ustanovah, namenjeni učenju in razvoju novih algoritmov.

1.0.4 Kratka zgodovina

V nadaljevanju podajamo nekaj mejnikov v zgodovini mobilne robotike.

1898 je Nikola Tesla demonstriral brezžično radijsko vodeno plovilo, predstavljeno na sejmu elektronike na Madison Square Garden v NY. Občinstvo je bilo prepričano, da gre za trik, čarovnijo oziroma, da plovilo vodi dresirana opica. Dejanska uporaba daljinskega radijskega vodenja se je pojavila šele v prvi svetovni vojni.

okoli 1940 , med drugo svetovno vojno, so Nemci razvili avtopilotske rakete (projekt V1 in V2). Istočasno je Američan Norbert Wiener razvijal sistem za avtomatsko usmerjanje in proženje protiletalskega topa, ki je vseboval radar, analogni računalnik in algoritem.

1953 je W. Grey Walter razvil elektronsko želvo - avtonomnega robota, ki je zaznaval svetlobo (fotocelica) in oviro (kontakt) ter znal poiskati izvor svetlobe in se izogibati oviram.

1966–1972 so v Standford raziskovalnem centru razvili robota Shakey, ki je imel kamero, merilnik razdalje, senzor trka in brezžično povezavo. Bil je prvi mobilni robot, ki je znal planirati svoje akcije.

1970 je na luni pristalo prvo avtonomno vozilo, ki je bilo daljinsko vodeno.

1976 je NASA na Mars poslala dve avtonomni vesoljski sondi.

1977 so v Francoskem raziskovalnem centru LAAS naredili Hilare 1 mobilnega robota opremljenega z ultrazvočnim in laserskim merilnikom razdalje in kamero na robotski roki.

1980 je bil razvit komercialni robot HERO, ki je bil predvsem namenjen zabavi in raziskovanju. Na Standfordu pa so naredili vozilo, opremljeno s kamero, s pomočjo katere je gradil zemljevid okolice.

razvoj po letu 1980 pa je naglo naraščal do danes, ko imamo več podjetij in raziskovalnih ustanov, ki tržijo in razvijajo mobilne robote.

1.0.5 Prihodnost mobilnih avtonomnih sistemov

Zagotovo lahko trdimo, da bojo v prihodnosti mobilni robotski sistemi boj izpopolnjeni in bodo opravljali številna opravila. Imeli bodo večjo stopnjo avtonomije in inteligence. Lahko nam bodo v pomoč pri številnih domačih opravilih, ne le pri čiščenju ali košenju trave kot sedaj temveč v številnih koristnih vlogah in aplikacijah za zabavo. Več bo humanoidnih robotov, ki so človeku ljubši kot recimo kolesne različice.

Kljub številnim drznim napovedim v zadnjih desetletjih, ko smo bili priča razvoju računalnikov, so bile marsikatere napovedi, kako bodo računalniki postali umetni možgani kompleksnih mobilnih robotov, ki bodo za ljudi opravljali težka dela. Danes imamo številnih robotske kosilnic, čistilce ter robotizacijo v industriji (npr. avtomobilski) vendar je tovrstna avtomatizacija daleč od okretnosti, inteligence in mobilnosti avtonomnih stvaritev, ki so jih napovedovali. Največji razkorak predstavlja zagotovitev potrebne umetne inteligence. Danes smo priča avtonomnih robotov, katerih inteligenca je na stopnji primitivnejših živali. Glede na nekatere zdajšnje napovedi naj bi roboti po letu 2020 postali vsakdan pri opravljanju hišnih opravil in kot taki postali pomembno družinsko imetje, kot so danes avtomobili. Okoli leta 2040 naj bi bili roboti sposobni abstrakcije in planiranja za opravljanje večine ročnih del podobno kot človek.

Poglavje 2

Agent in večagentni sistemi

2.1 Uvod

Eden od načinov reševanja določenih nalog je vpeljava agenta, oz. entitete (smiselne zaključene celote), ki je zmožna sama bolj ali manj uspešno reševati določen problem. Agenti so lahko fizični (robot), ki vplivajo na stvarni svet ali pa virtualni (simulirani, programske komponente), ki vplivajo na virtualno okolje.

Večih agentov, ki deluje v nekem okolju predstavlja večagentni sistem (Multi-Agent System, MAS). Večagentni sistemi torej podajajo principe za gradnjo kompleksnih sistemov s pomočjo agentov in mehanizmov za koordinacijo delovanj neodvisnih agentov.

Osnovno vodenje oziroma delovanje agenta je potrebno, ne pa tudi zadostno, za usklajeno delovanje skupine agentov pri doseganju skupnega cilja. Vodenje večagentnega sistema je tako vedno kombinacija učinkovitega delovanja na nivoju osnovnih agentov in ustreznega sodelovanja med njimi.

V nadaljevanju podamo nekaj definicij in klasifikacij agenta in večagentnih sistemov.

2.2 Večagentni sistem

Večagentni sistemi so razmeroma mlada veda na področju umetne inteligence. Pristopi večagentnih sistemov posegajo na področje porazdeljene umetne inteligence (angleško: Distributed Artificial Intelligence, DAI) in področja umetnega življenja (angleško: Artificial Life). Namen prvega je razvoj organizacije sistemov, ki so zmožni reševati probleme z razmišljanjem (Cognitive agents), drugi pa skuša modelirati žive organizme, torej zmožnost preživetja in prilagajanja (adaptiranja) v običajno sovražnem okolju. Porazdeljeni sis-

temi so v računalništvu že uveljavljena vsakdanjost (večprocesorski sistemi), medtem ko metode za koordinacijo več agentov v robotiki pridobivajo na popularnosti.

Večagentni sistem torej sestoji iz več avtonomnih ali delno avtonomnih agentov, ki sestavljajo nek kompleksen sistem, in podaja mehanizme za koordinacijo delovanja teh neodvisnih agentov. Agenti izkazujejo določeno obnašanje, ki ga pogosto določajo enostavna pravila. Na to obnašanje pa vplivajo komunikacija z drugimi agenti in interakcije z okoljem in objekti v okolju. Izziv večagentnih sistemov je predvsem kooperativno delovanje več agentov. Da to dosežemo, je potrebnih nekaj korakov za zagotovitev njihove sinhronizacije, komunikacije (direktno: sistem sporočil, skupne tabele, ipd. in indirektno: z opazovanjem ostalih in sklepanjem) in pogajanj o delitvi dela.

Večagentni sistem lahko torej opredelimo kot sistem, ki v splošnem vsebuje naslednje elemente:

- okolje,
- množica pasivnih objektov,
- množico agentov (aktivni objekti v okolici) in
- množico odnosov in metod interakcije agentov z objekti okolice.

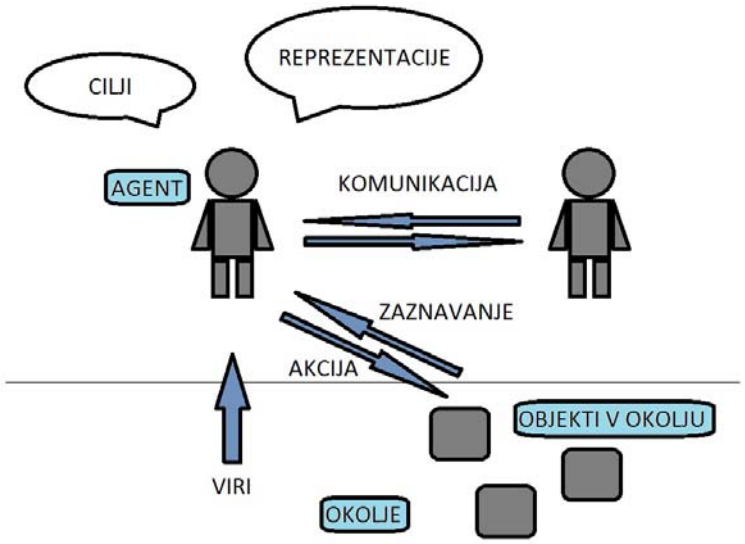
Ilustracijo večagentnega sistema podaja slika 2.1.

Večagentne sisteme, kjer so edini objekti agenti in okolje ni definirano, so *komunikacijski večagentni sistemi*. V tem primeru odnosi med agenti predstavljajo omrežje, kjer je vsak agent povezan z ostalimi agenti. Taki sistemi so pogosti na področju porazdeljene umetne inteligence (angleško: Distributed Artificial Intelligence, DAI), kjer so agenti tipično programski moduli.

V kolikor pa so agenti situirani v okolju in komunikacija poteka le indirektno preko zaznavanja in delovanja na okolje pa imamo *izključno situiran večagentni sistem*. Splošni večagentni sistemi pa imajo lastnosti obeh omenjenih skrajnih oblik.

2.3 Agent

Čeprav stroga, uveljavljena definicija agenta ne obstaja, agenta, kot smo že omenili, večinoma predstavlja entiteta v nekem okolju z možnostjo zaznavanja okolja, svojimi cilji, znanjem iz določenega področja, odločanjem in delovanjem v tem okolju. Agent ima senzorje s katerimi zaznava okolje



Slika 2.1: Izgled večagentnega sistema, kjer agent zaznava in vpliva na okolje ter na ostale agente v okolju.

(npr: senzor bližine zazna oviro), aktuatorje s katerimi na okolje vpliva (npr: kolesni pogon premakne robota in/ali odrine oviro) ter znanje iz okolja v katerem deluje, ki mu omogoča, da s pomočjo informacije iz senzorjev upravlja svoje aktuatorje, za dosego nekega cilja (npr. priti v želeno lokacijo).

Naštejmo nekaj lastnosti, ki opisujejo fizičnega ali virtualnega agenta:

- zmožnost delovanja v okolju,
- lahko komunicira z ostalimi agenti,
- ima nabor svojih teženj in ciljev,
- ima dostop do virov (napajanje, CPU, spomin, informacije),
- ima zmožnost zaznavanja okolice (do določene mere),
- ima svojo (delno) predstavitev okolice, ali pa je sploh nima,
- se lahko reproducira,
- njegovo delovaje stremi k dosegi svojih ciljev, kjer uporablja vire, svoja znanja, zaznave senzorjev, svojo predstavitev okolja (znanje o okolju) in komunikacijo.

Pomembna lastnost agenta je avtonomija, kar pomeni, da agent ni upravljan preko nekega drugega (npr. operaterja ali drugega agenta), ampak je sposoben samostojnega delovanja, glede na lastne cilje in situacije v katerih se znajde. Avtonomni agent ima tudi dostop do lastnih vire, ki so lahko napajanje, pomnilnik, informacije, itd. Agentovo zaznavanje je omejeno z lastnostmi senzorjev, ki jih ima, zato ima tudi le delno predstavitev okolja, saj ne more zaznavati vsega dogajanja v okolju. Na voljo so mu le lokalne informacije, torej tiste v dosegu njegovih senzorjev, zato so večagentni sistemi večinoma decentralizirani sistemi, kjer obnašanje agentov ni centralno nadzorovano. Svoje trenutno stanje v okolju ima predstavljeno s spremenljivkami. Njegovo delovanje pa je odvisno od stanja v katerem se nahaja, več ko ima možnih stanj, na več različnih načinov lahko deluje. Agenti v večagentnem sistemu so lahko različni tako po lastnostih, obnašanjih, virih, zmožnosti predstavitev, sposobnostjo pomnjenja dogodkov in interpretacij razpoložljivih informacij.

Da je agent v takem okolju uporaben, mora imeti sposobnosti prilagajanja. Znanje, ki mu je bilo vgrajeno, mora biti fleksibilno. Lahko ga dopolnjuje (adaptira) s spreminjanjem določenih parametrov, možni so tudi določeni algoritmi, ki slonijo na evoluciji živih bitij, ter ostali algoritmi strojnega učenja (genetski algoritmi, nevronske mreže, učenje z nagrajevanjem). Uspeh teh metod pogojuje dejstvo, da je problem umetne inteligence pogosto kombinacijsko preveč kompleksen, da bi bil rešljiv v stvarnem času. Zato inteligenca agenta skoraj vedno sestoji le iz dveh virov: znanja, pridobljenega na osnovi lastnih izkušenj (učenje, adaptiranje), ter znanja, ki je bilo agentu vgrajeno.

Agent ima za razliko od ostalih programov in objektno orientiranega programiranja naslednje lastnosti:

- zaznava okolje, v katerem se nahaja,
- ima sposobnost interakcije z ostalimi agenti in
- najpomembnejše: na poti k izpolnjevanju lastnih ciljev se odloča in deluje samoiniciativno.

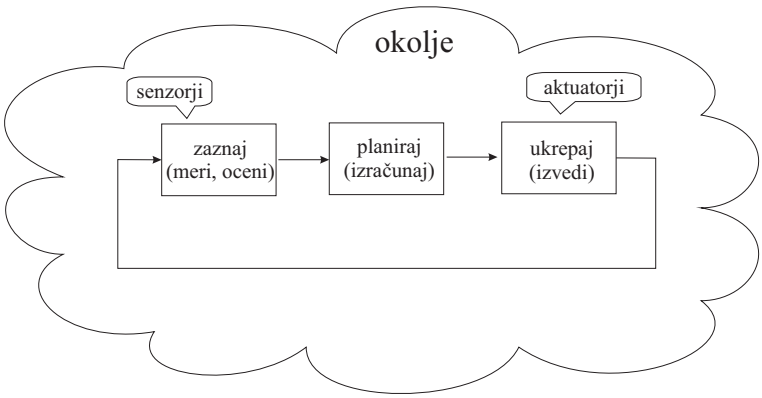
Objekti pa so pasivni elementi, ki nimajo možnosti izbire svojega delovanja, delujejo le na zunanjo iniciativo.

2.4 Način delovanja agentov

Klasičen in uveljavljen način vodenja (od leta 1985 dalje) v mobilni robotiki in avtomatiki temelji na načelu zaznaj-planiraj-ukrepaj (sense-plan-act,

SPA). Sistem torej najprej pridobi informacijo okolja s senzorji. Nato izgradi model z uporabo pridobljene informacije in planira oz. izračuna naslednji korak. Agent mora torej ugotoviti, kako se z vgrajeno strategijo odzvati na zaznane podatke. Na koncu agent ukrepa in izvede akcijo. SPA poteka v iteracijah, po zaznavanju, planiranju, ukrepanju se celoten cikel ponovi.

Osnovna ideja SPA je osnova avtomatskega vodenja, kjer se poskuša postopno zmanjševati pogrešek med želenim stanjem mobilnega sistema (ali kateregakoli drugega sistema) in med dejanskim stanjem (slika 2.2).



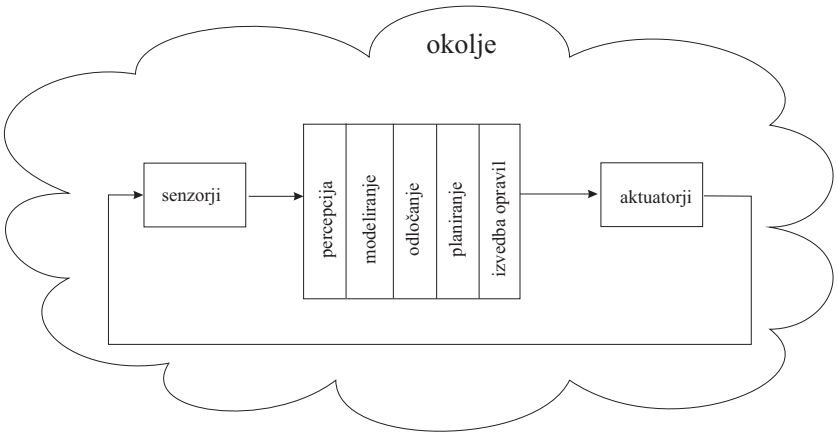
Slika 2.2: Osnovni način upravljanja agenta (SPA).

2.4.1 Kognitivni agenti

Kognitivni agenti (angleško deliberate agents) delujejo po principu SPA (sense-plan-act). Ko agent zazna okolico s pomočjo modela sveta (simbolična predstavitev okolice) naredi plan oz. načrt za izvedbo akcije.

Načrt reševanja problema (slika 2.3) se določi korak za korakom (interpretacija zaznav senzorjev, modeliranje, odločanje, planiranje, izvedba opravil, upravljanje aktuatorjev) na osnovi svojega zaznavanja okolja. Vsak tak agent navadno vsebuje bazo znanja, ki sestoji iz podatkov ter znanja, potrebnega za reševanje problemov. V nepredvidljivem dinamičnem okolju agent z izključno kognitivnimi sposobnostmi ni učinkovit, saj njegov načrt reševanja problemov ne more predvideti sprememb okolja in bi ga moral zato nenehno spreminjati.

Kognitivni agenti imajo nedvomno prednost v statičnih in poznanih okoljih. V primeru nepričakovanih dogodkov glede na njihov model sveta pa lahko odpovejo. Potrebujejo precej natančen model sveta (npr. zemljevid), ki ga je pogosto težko dobiti in vzdrževati. Za svoje delovanje potrebujejo veliko pro-



Slika 2.3: Kognitivni agenti na osnovi zaznav naredi načrt reševanja problema.

cesno moč, kar se posledično lahko odraža v počasnem odzivu na spremembe v oklici.

2.4.2 Odzivni agenti

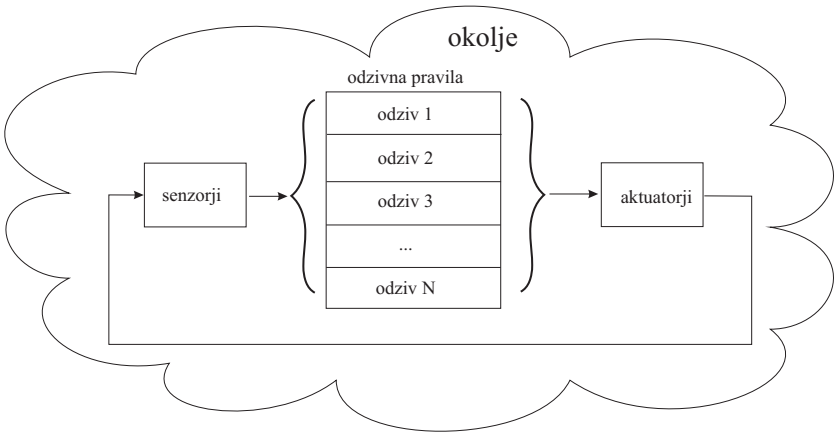
Odzivni agenti (slika 2.4) so zmožni povezati svoje zaznavanje okolja z akcijami vodenja (različna preddefinirana obnašanja) in s tem kar najbolje izvršiti naloge brez gradnje internega modela okolja (kar je sorodno obnašanju manjših živali, npr. mravelj).

Delujejo torej po principu zaznaj-deluj brez uporabe simbolične predstavitve okolja (modela sveta) in planiranja. Zanašajo se le na eno ali več enostavnih pravil, ki neposredno povežejo zaznave senzorjev z akcijami.

V osnovi odzivni agenti nimajo stanj (shranjevanja nekaterih preteklih podatkov), so brez spomina, brez internega modela okolice, nimajo možnosti planiranja akcij v naprej in niso zmožni učenja. Njihova prednost je ravno v njihovi preprostosti, kar jim omogoča hiter (trenuten) odziv.

2.4.3 Hibridni agenti

Ko že omenjeno so v nepredvidljivem dinamičnem okolju primernejši odzivni oz. hibridni agenti, ki združujejo dobre lastnosti obeh - tako odzivnih kot tudi kognitivnih. Obstajajo agenti, ki nimajo celotne ali obsežne simbolične predstavitve sveta okoli njih, ampak si, na primer, zapomnijo le nekaj pomembnejših parametrov. Kar jim lahko pomaga pri boljši asociaciji zazznav



Slika 2.4: Odzivni agent, reagira na zaznave brez planiranja.

z akcijami ali bolj dovršene akcije (npr. agent si lahko zapomni, da je v bližini stene).

Nadalje lahko ima agent zmožnost adaptivnosti. To pomeni, da glede na svoje prejšnje izkušnje spreminja vzorce delovanja (obnašanje) in se prilagaja spreminjajočim razmeram. Z drugimi besedami lahko temu rečemo tudi učenje. Za učenje na individualni ravni mora imeti agent spomin, torej pri popolnoma odzivnih agentih to ni mogoče.

Obstaja pa tudi adaptivnost na ravni sistema, ki je mogoča tudi pri večagentnih sistemih, ki so sestavljeni iz odzivnih agentov. Če se agenti v sistemu lahko reproducirajo, se lahko poveča število agentov tistih vrst, ki so bolj primerne za novonastale razmere.

2.4.4 Odzivno vedenjski agenti

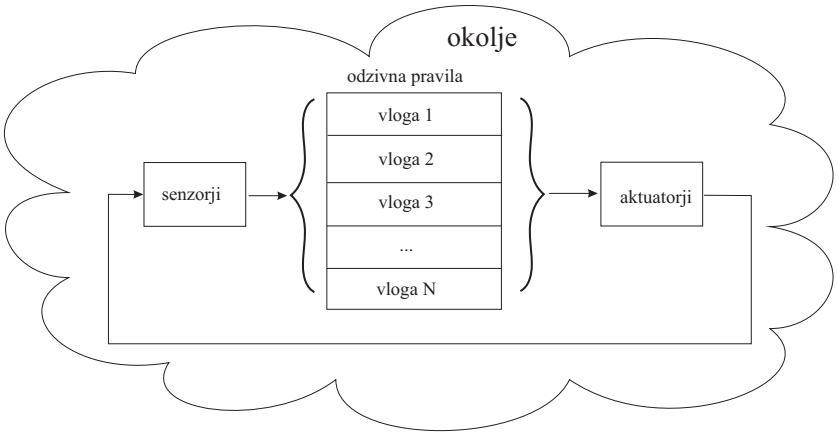
Njihovo delovanje je enako kot delovanje odzivnih agentov le da namesto enostavnih pravil uporabljajo vloge (angleško behaviours). Vloge predstavljajo module, ki se izvajajo paralelno (slika 2.5), saj ima vsak modul dostop do do senzorjev in lahko neposredno upravlja z aktuatorji. Vsaka vloga vsebuje neko znanje v obliki algoritmov vodenja, ki agentu omogoča primerno delovanje v določeni situaciji (sledenje steni, iskanje predmeta, izogibanje oviri, prihod v začetni položaj,...).

Podajmo nekaj lastnosti vedenjskih agentov:

- vsebujejo različne vloge za dosego različnih ciljev ali sledenja ciljev,
- vloge prejmejo vhodne informacije od senzorjev in lahko tudi od drugih vlog in posredujejo ukaze aktuatorjem,

- vloge so lahko kompleksne in so sestavljena iz različnih akcij. (akcije: stop, naprej, levo,... vloge: sledenje cilju, izogibanje oviri)

Ker se vloge izvajajo hkrati in neodvisno so taki agenti primerni za aplikacije v realnem času. Vloge lahko imajo stanja (si zapomnijo zgodovino), model okolice in zmožnost planiranja v naprej, kar omogoča izvedbo učinkovitih vlog.



Slika 2.5: Vedenjski agent se odziva na zaznave z izvajanjem vlog.

Primer 2.1. *Poglejmo si primer izvedbe enostavnega agenta ki bo deloval kognitivno in agenta, ki bo deloval odzivno. Agent je mobilni robot, ki želi iti skozi vrata, ki so zaklenjena.*
Rešitev

Kognitivni agent *lahko planira svoje delovanje, torej bo zgradil načrt v večih zaporednih korakih v obliki:*

```
Načrt odpiranja vrat:
 Grem do mesta, kjer je spravljen ključ,
 Vzamem ključ,
 Grem do vrat,
 Odprem vrata s ključem.
```

Odzivni agent *pa le reagira na situacije iz okolja brez planiranja oziroma razmišljanja. Njegovo obnašanja omogoča skupek enostavnih pravil Pi oziroma vlog:*

```
P1: Če sem pred vrati in imam ključ, potem odprem vrata.
P2: Če sem pred vrati in nimam ključa, potem poskusim
    odpreti vrata.
P3: Če se vrata ne odprejo in nimam ključa,
    potem grem iskat ključ.
P4: Če iščem ključ in vidim ključ pred sabo, potem vzamem
    ključ in grem proti vratom.
```

Vidimo, da kognitivni agent zgradi načrt, medtem ko ima odzivni agent že predhodno vgrajena pravila. Kognitivni agent bo gotovo odprl vrata hitreje, z manj akcijami, saj lahko predvidi zaporedje potrebnih akcij. Odzivni agent pa bo najprej šel do vrat in nato bo ugotovil, da nima ključa in da ga mora iti iskat. Seveda pa je odzivni agent bolj robusten, če so vrata odprta jih bo odprl takoj, ne da bi šel iskat ključ. Kognitivni agent, pa bo šel najprej po ključ, saj njegov model ne predvideva možnosti, da so vrata mogoče že odprta.

2.4.5 Osnovne vedenjske arhitekture

Izvedba arhitekture odzivno vedenjskih agentov je možna na veliko načinov. Osnovni arhitekturi (zgradbi) za izvedbo vedenjskih agentov sta tekmovalna shema in vsebovana shema.

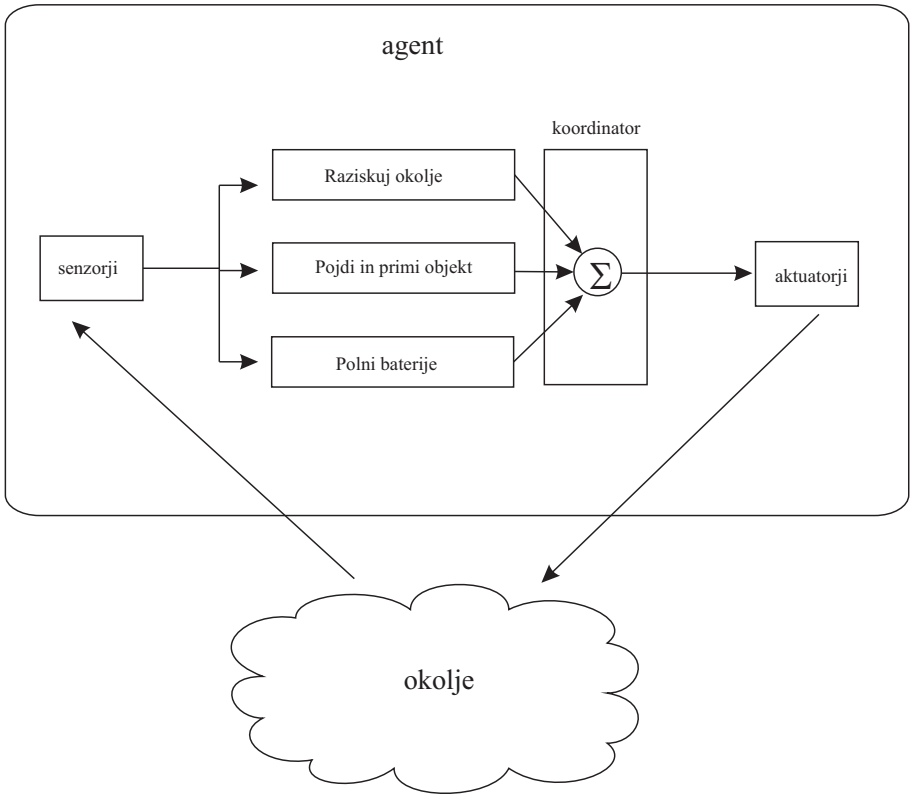
Tekmovalna shema

Tekmovalna shema (angleško:competitive architecture, motor shema architecture) je princip, ki ga je vpeljal [31]. Vloga sestoji iz sheme percepcije, ki procesirajo vhode iz senzorjev in jih posredujejo motornim shemam. Tekmovalne oz. motorne sheme pa generirajo izhodne za vodenje, ki povedo, kako naj se robot premika, da bo dosegel cilj. Gre za to, da več konkurenčnih vlog (shem) generira svoje ukaze (hitrost, smer gibanja,...) agentu, ukazi posameznih vlog so predstavljeni kot vektorji z uporabo potencialnega polja (vektorji so normirani glede na percepcijo in motorno shemo vloge). Prispevki posameznih vlog se nato združijo v končen ukaz, ki se posreduje aktuatorju agenta. Druga možnost pa je, da se izmed vseh vlog izbere le eno, ki je najbolj uspešna (na podlagi določenih parametrov se oceni uspešnost).

V osnovi gre za privlačna in odbojna polja. Predstavljajmo si primer, kjer agenta privlači cilj (vektor smeri vožnje je v smeri cilja), hkrati pa ga odbija ob ovire (vektor želene smeri vožnje kaže staran od ovire). Bliže, ko je agent oviri bolj prevladuje odbojni vektor smeri in se zmanjšuje privlačni

vektor proti cilju. Končna usmeritev je vektorska vsota teh dveh normiranih vektorskih polj.

Primer 2.2. *Poglejmo si primer vedenjskega agenta, katerega vloge so organizirane v tekmovalno oz. motorno shemo. Preprostega raziskovalnega robota, ki raziskuje okolje in ko zazna kak predmet gre ponj. V kolikor mu zmanjka energije pa gre polnit akumulatorje. Nabor vlog za izvedbo delovanja agenta povežemo v strukturo kot nakazuje slika.*



Vsebovana shema

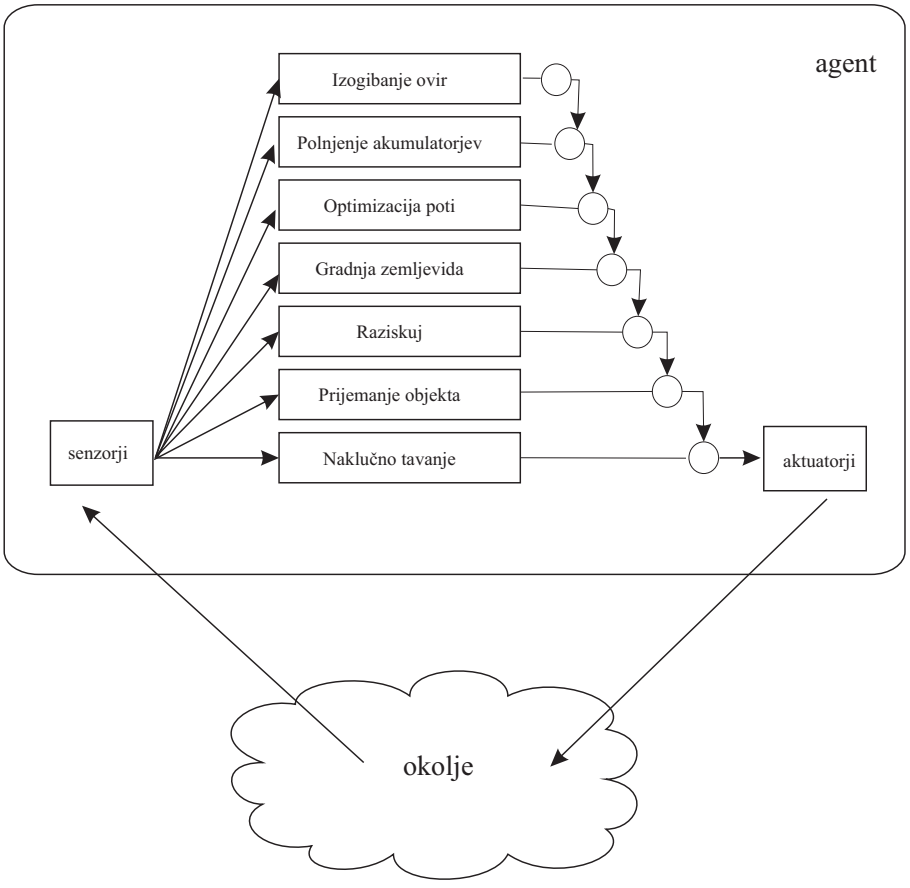
Vsebovana shema (angleško: subsumption architecture) je način dekompozicije inteligentnega obnašanja agenta v več preprostih vlog, ki so organizirane v nivojih po prioritetah. Posamezne nivoje lahko zgradimo z uporabo končnih avtomatov. Pojem je vpeljal Rodney Brooks [30]. Vse vloge se izvajajo hkrati in prejemajo informacijo od senzorjev ter posredujejo ukaze aktuatorjem. Vloge z višjo prioriteto posredujejo komande aktuatorjem. Tu

velja opomniti, da določene vloge (opravila) lahko onemogočijo ali spremenijo percepcijo ali povozijo akcije nižje prioritetnih vlog.

Vloge lahko onemogočijo svoje delovanje (onemogočijo izhode ali izhode), če na podlagi senzorjev ni izpolnjen pogoj za njihovo izvajanje. Vloga z višjo prioriteto lahko onesposobi vloge z nižjo prioriteto. Vloga z najvišjo prioriteto, ki ostane aktivna določa novo akcijo.

Vloge z višjimi prioritetami (v višjih slojih) so bolj abstraktne vloge in lahko dosežejo celotni cilj. Višje vloge vključujejo funkcije nižjih vlog. Vloge z nižjimi prioritetami (nižji sloji) pa so preprostejše in hitro odzivne (refleksi).

Primer 2.3. *Poglejmo si primer vedenjskega agenta, katerega vloge so organizirane v vsebovano shemo. Raziskovalnega robota z nekoliko nadgrajeno funkcionalnostjo primera 2.2 podaja spodnja slika. Nabor vlog za izvedbo delovanja agenta povežemo v vsebovano strukturo, kjer so vloge razdeljene v nivoje. Vloge v višjih nivojih lahko onemogočijo vloge v nižjih nivojih, kar nakazujejo krogci.*



2.4.6 Ostale delitve agentov in večagentnih sistemov

Večagentne sisteme lahko obravnavamo glede na štiri lastnosti agentov:

- granulacijo agentov (fina ali groba),
- raznolikost znanja agentov (splošno ali specializirano),
- znanje agenta (konstruktivno ali tekmovalno, ekipno ali hierarhično, statično ali dinamično spreminjanje vloge) in
- komunikacijo (oglasna deska ali sporočila, malo ali veliko komunikacije, direktna ali indirektna, vsebina).

Ponavadi imajo agenti grobo granulacijo (število agentov) in veliko stopnjo komunikacije, v ostalih lastnostih pa se razlikujejo. Skupina sodelujočih agentov je pri reševanju kompleksnega problema lahko bolj prilagodljiva in ekonomična kot en sam zmogljivejši agent, če le uspemo učinkovito rešiti oz. zagotoviti njihovo koordinacijo. Dejstvo pa je, da z večanjem števila agentov pri opravljanju nekega opravila ni smiselno pretiravati, ker je v tem primeru preveč energije vložene v njihovo koordinacijo, komunikacijo in pogajanja. Isto opravilo lahko opravi tudi manj agentov z enako ali boljšo učinkovitostjo.

2.5 Uporaba večagentnih sistemov

Področje uporabe večagentnih sistemov je zelo široko.

Tako imamo aplikacije v avtomatizaciji proizvodnje (avtomobilska proizvodnja, avtonomni vozički v skladiščih) in robote skavte (nevarna območja). Nekatere aplikacije pošiljajo robote skavte v izvidnico, le-ti med seboj sodelujejo in raziskujejo in kartirajo teren. To so lahko le simulacije, pa tudi resnične aplikacije (vojska, vesolje, nevarni tereni - globina, vulkani, minska polja, ...). Modele večagentnih sistemov lahko uporabljamo za simulacijo transporta in prometa, za raziskovanje potrošniških in finančnih trgov, preučevanje razširjanja epidemij [4], optimizacijo proizvodnje in logistike, simulacijo premikov bojnih enot na bojišču in simulacijo socialnih mrež. Večagentni sistemi se uporabljajo tudi v filmski industriji, za simulacijo velikih množic ljudi. V filmih, kot so Avatar, Lord of the Rings, King Kong in mnogih drugih, je bil uporabljen programski paket MASSIVE (Multiple Agent Simulation System in Virtual Environment). Nekateri modeli so enostavnejši in zajemajo le bistvene lastnosti sistemov, drugi so kompleksnejši in preverjeni tudi z uporabo podatkov iz realnega sveta. S pomočjo razvoja specialne programske opreme za modeliranje večagentnih sistemov

in povečevanjem računske moči računalnikov je možno ustvarjati vedno bolj napredne večagentne aplikacije, s pomočjo katerih pridemo do točnejših rezultatov in ugotovitev na raznovrstnih strokovnih področjih.

Poglavje 3

Modeliranje gibanja mobilnih sistemov

3.1 Uvod

V poglavju je predstavljeno modeliranje gibanja različnih mobilnih sistemov. Dobljeni modeli so koristni pri načrtovanju strategij lokomocije sistema. **Lokomocija** je proces gibanja mobilnega sistema iz enega mesta na drugo mesto.

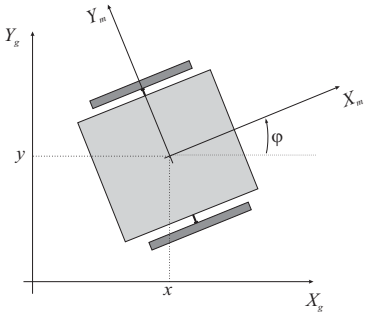
Modeli gibanja lahko opisujejo le kinematiko mobilnega sistema, kjer nas zanima matematičen zapis gibanja brez upoštevanja sil in navorov, ki v splošnem tako gibanje povzročijo. **Kinematični model** opisuje geometrijske relacije med vhodi sistema in obnašanjem sistema, ki je podano stanji sistema. Kineamatični model je ponavadi vezan na hitrostni prostor in je predstavljen z nizom diferencialnih enačb prvega reda.

Dinamični model pa opisuje gibanje sistema zaradi vplivanja sil in navor na system. Dinamični model vsebuje fizikalne veličine kot so sile, energije, masa, vztrajnost in hitrosti. Opisi dinamičnih modelov so podani z diferencialnimi enačbami drugega reda.

Pri modeliranju kolesnih mobilnih sistemov se večinoma uporablja kinematične modele za načrtovanje strategij gibanja. V primeru zahtevnejših aplikacij, in ostalih mobilnih sistemih kot so npr. nožni roboti, zračna plovila, hitra kolesna vozila in podobno pa uporabljamo dinamične modele gibanja.

3.2 Kinematika kolesnih mobilnih sistemov

Obstaja več različnih kinematičnih modelov, nekaj pommebnejših je našteto v nadaljevanju



Slika 3.1: Vozilo v ravnini.

- **Notranja kinematika** podaja relacije med internimi spremenljivkami sistema (npr. kako vrtenje koles vpliva gibanja vozila).

- **Zunanja kinematika** opisuje pozicijo in orientacijo vozila glede na nek referenčni koordinatni sistem.

- **Direktna kinematika** in **Inverzna kinematika**. Direktna kinematika modelira stanja sistema kot funkcijo vhodov (hitrosti koles, gibanje sklepov, zasuk krmilnega kolesa,...).). Inverzna kinematika pa podaja vhode v sistem, ki so potrebni za dosego želenega stanja sistema, kar pomeni, da se predvsem uporablja pri planiranju gibanja.

- **Omejitve gibanja** se tipično pojavijo, ko ima sistem manj vhodnih spremenljivk kot prostostnih stopenj (neholonomične omejitve). Holonomične omejitve omejujejo dosegljivost določenih stanj sistema, medtem ko neholonomične omejitve omejijo smeri možnih premikov sistema (kolesa robota se lahko kotalijo le v smeri njihove orientacije).

V nadaljevanju je podanih nekaj primerov določitve notranje in zunanje kinematike kolesnih mobilnih sistemov oz. robotov (WMR - wheeled mobile robot). Lega vozila v ravnini je podana z vektorjem stanj

$$q(t) = \begin{bmatrix} x(t) \\ y(t) \\ \varphi(t) \end{bmatrix}$$

v globalnih koordinatah X_g, Y_g kot je ilustrirano na sliki 3.1. Premikajoči koordinatni sistem X_m, Y_m pa je pripet na vozilo. Relacija med njim in globalnim koordinatnim sistemom (zunanja kinematika) je podana z vektorjem

translacije $[x, y]^T$ in rotacijsko matriko

$$R(\varphi) = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Kolesni mobilni robot (WMR) se giblje z uporabo koles, katera se kotalijo po podlagi zaradi trenja med kolesom in podlago. Pri zmernih hitrosti WMR lahko predpostavimo model **idealnega kotaljenja koles**, kjer se kolo lahko premika le zaradi rotacije (kotaljenja) ni pa premikov zaradi zdrsov v smeri kotaljenja in pravokotno na smer kotaljenja. Vsako kolo se lahko prosto vrti okoli svoje osi, kar pomeni, da torej mora obstajati točka, ki leži na presečišču vseh osi koles. Ta točka se imenuje **trenutni center rotacije** (ICR - instantaneous center of rotation) in definira točko okoli katere krožijo vsa kolesa z enako krožno hitrostjo ω glede na ICR.

3.2.1 Diferencialni pogon

Diferencialni pogon je zelo preprost mehanizem pogona in zato precej pogosto uporabljen, predvsem pri manjših vozilih ali mobilnih robotih. Vozilo s takim pogonom imajo ponavadi še dodatna podporna kolesa (castor), ki preprečijo prevračanje vozila. Kolesi diferencialnega pogona sta vpeta na isti osi, hitrost vrtenja vsakega kolesa pa je poljubna in gnana s svojim motorjem. Glede na sliko 3.2 so vhodne spremenljivke hitrosti levega kolesa $v_L(t)$ in desnega kolesa $v_R(t)$.

Pomen ostalih parametrov na sliki 3.2 je: r je radij kolesa, L je razdalja med kolesi in $R(t)$ je trenutni radij trajektorije vožnje vozila. $R(t)$ je razdalja med središčem vozila (središčna točka med kolesi) in točko ICR. V vsakem časovnem trenutku imata obe kolesi enako krožno hitrost $\omega(t)$ okrog ICR.

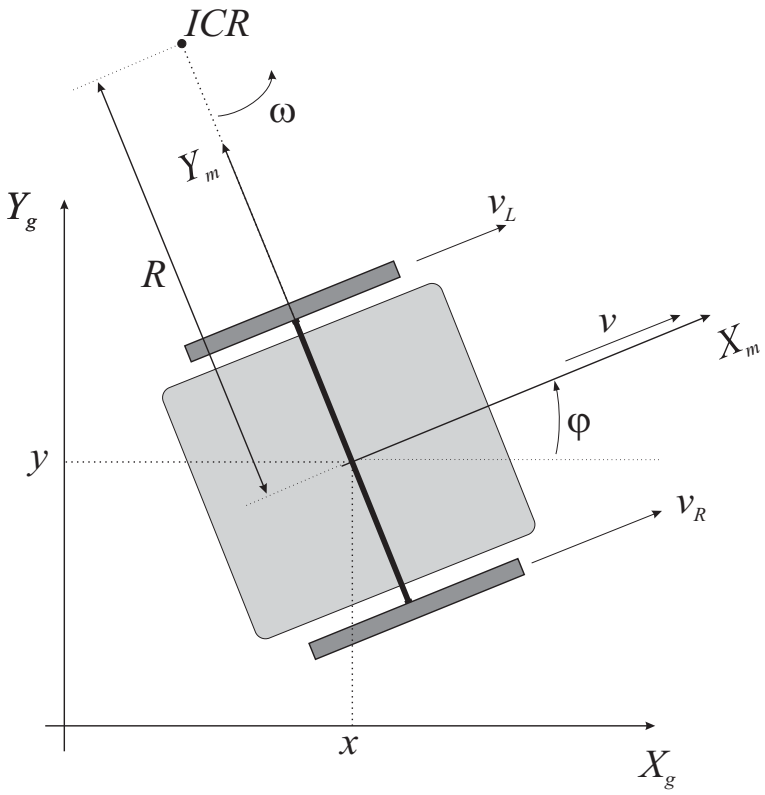
$$\omega = \frac{v_L(t)}{R(t)-\frac{L}{2}}$$
$$\omega = \frac{v_R(t)}{R(t)+\frac{L}{2}}$$

določimo $\omega(t)$ and $R(t)$ kot sledi

$$\omega(t) = \frac{v_R(t)-v_L(t)}{L}$$
$$R(t) = \frac{L}{2}\frac{v_R(t)+v_L(t)}{v_R(t)-v_L(t)}$$

Tangencialna hitrost vozila je

$$v(t) = \omega(t)R(t) = \frac{v_R(t)+v_L(t)}{2}$$



Slika 3.2: Kinematika diferencialnega pogona.

Obodni hitrosti koles sta $v_L(t) = r\omega(t)_L$ in $v_R(t) = r\omega(t)_R$, kjer sta $\omega_L(t)$ in $\omega_R(t)$ krožna hitrost levega in desnega kolesa okoli njune osi. Upoštevajoč navedene relacije lahko zapišemo notranjo (interno) kinematiko kot

$$\begin{bmatrix} \dot{x_m}(t) \\ \dot{y_m}(t) \\ \dot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} v_X(t) \\ v_Y(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ -\frac{r}{L} & \frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix} \tag{3.1}$$

Zunanja (eksterna) kinematika (v globalnih koordinatah) pa je

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ \sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3.2}$$

kjer sta $v(t)$ in $\omega(t)$ vhodni (regulirni) spremenljivki. Model (3.2) lahko zapišemo v diskretni obliki (3.3), ki je veljavna za diskretne čase vzorčenja $t = kT_S$, $k = 0, 1, 2, \ldots$, kjer je T_s čas vzorčenja.

$$\begin{aligned} x(k+1) &= x(k) + v(k)T_s\cos(\varphi(k)) \\ y(k+1) &= y(k) + v(k)T_s\sin(\varphi(k)) \\ \varphi(k+1) &= \varphi(k) + \omega(k)T_s \end{aligned} \tag{3.3}$$

Direktna in inverzna kinematika

Lego vozila v trenutku t dobimo z integracijo kinematičnega modela, kar imenujemo **odometrija** (ang. odometry ali tudi dead reckoning). Postopek ocene lege vozila za podane vhodne veličine imenujemo direktna kinematika.

$$\begin{aligned} x(t) &= \int_0^t v(t)\cos(\varphi(t))dt \\ y(t) &= \int_0^t v(t)\sin(\varphi(t))dt \\ \varphi(t) &= \int_0^t \omega(t)dt \end{aligned} \tag{3.4}$$

Če predpostavimo konstante hitrosti v and *omega* med časi vzorčenja lahko integracijo v enačbah (3.4) izračunamo numerično z uporabe Eulerjeve metode integracije. Dobimo direktno kinematiko

$$\begin{aligned} x(k+1) &= x(k) + v(k)T_s\cos(\varphi(k)) \\ y(k+1) &= y(k) + v(k)T_s\sin(\varphi(k)) \\ \varphi(k+1) &= \varphi(k) + \omega(k)T_s \end{aligned} \tag{3.5}$$

Če uporabimo trapezno metodo integracije (Runge-Kutta) dobimo bolj točen rezultat numerične integracije

$$\begin{aligned} x(k+1) &= x(k) + v(k)T_s\cos\left(\varphi(k) + \tfrac{\omega(k)T_s}{2}\right) \\ y(k+1) &= y(k) + v(k)T_s\sin\left(\varphi(k) + \tfrac{\omega(k)T_s}{2}\right) \\ \varphi(k+1) &= \varphi(k) + \omega(k)T_s \end{aligned} \tag{3.6}$$

V primeru eksaktne integracije pa dobimo direktno kinematiko

$$\begin{aligned} x(k+1) &= x(k) + \tfrac{v(k)}{\omega(k)} \left(\sin\left(\varphi(k) + \omega(k)T_s\right) - \sin(\varphi(k))\right) \\ y(k+1) &= y(k) - \tfrac{v(k)}{\omega(k)} \left(\cos\left(\varphi(k) + \omega(k)T_s\right) - \cos(\varphi(k))\right) \\ \varphi(k+1) &= \varphi(k) + \omega(k)T_s \end{aligned} \tag{3.7}$$

kjer integriramo izraz (3.7) znotraj integrala vzorčenja in dobimo sledeče spremembe stanj

$$\Delta x(k) = v(k) \int_{kT_s}^{(k+1)T_s} \cos\left(\varphi(t)\right) dt = v(k) \int_{kT_s}^{(k+1)T_s} \cos\left(\varphi(k) + \omega(k)(t - kT_s)\right) dt$$

$$\Delta y(k) = v(k) \int_{kT_s}^{(k+1)T_s} \sin\left(\varphi(t)\right) dt = v(k) \int_{kT_s}^{(k+1)T_s} \sin\left(\varphi(k) + \omega(k)(t - kT_s)\right) dt$$

Bolj zahtevna je inverzna kinematika, kjer moramo določiti potrebne vhode, da se bo vozilo peljalo v želeno lego ali po želeni trajektoriji. Poljubno gibanje vozila ponavadi omejujejo neholonomične omejitve (poglavje 3.3), kar pomeni da niso možne poljubne smeri vožnje. Možnih je tudi več različnih rešitev (poti), da prispemo v želeno lego.

Preprosta rešitev inverzne kinematike je možna, če dovolimo le premo gibanje vozila ($v_R(t) = v_L(t) = v_R \Longrightarrow \omega(t) = 0$, $v(t) = v_R$) ali le kroženje na mestu ($v_R(t) = -v_L(t) = v_R \Longrightarrow \omega(t) = \frac{2v_R}{L}$, $v(t) = 0$) s konstantnimi hitrostmi. Za kroženje na mestu se enačbe gibanja (3.4) poenostavijo v

$$\begin{aligned} x(t) &= x(0) \\ y(t) &= y(0) \\ \varphi(t) &= \varphi(0) + \tfrac{2v_R t}{L} \end{aligned} \tag{3.8}$$

za premo gibanje pa se enačbe gibanja (3.4) poenostavijo v

$$\begin{aligned} x(t) &= x(0) + v_R \cos(\varphi(0))t \\ y(t) &= y(0) + v_R \sin(\varphi(0))t \\ \varphi(t) &= \varphi(0) \end{aligned} \tag{3.9}$$

Strategija gibanja potem vsebuje najprej usmeritev (rotacija) vozila proti ciljni poziciji, nato prema vožnja proti cilju in končno poravnava (rotacija) dejanske orientacije vozila z želeno orientacijo v cilju. Vhodi za vsako fazo (rotacija, premo gibanje, rotacija) se lahko enostavno izrazijo iz (3.8) in (3.9).

Če predpostavimo diskretno notacijo, kjer so hitrosti $v_R(k)$, $v_L(k)$ znotraj intervala vzorčenja Ts konstantne in se lahko spreminjajo le v časovnih trenutkih $t = kT_s$, potem lahko zapišemo enačbe gibanja kot sledi. Za kroženje

($v_R(k) = -v_L(k)$)) imamo

$$\begin{aligned} x(k+1) &= x(k) \\ y(k+1) &= y(k) \\ \varphi(k+1) &= \varphi(k) + \frac{2v_R(k)T_s}{L} \end{aligned} \tag{3.10}$$

in za premo gibanje ($v_R(k) = v_L(k)$)

$$\begin{aligned} x(k+1) &= x(k) + v_R(k)\cos(\varphi(k))T_s \\ y(k+1) &= y(k) + v_R(k)\sin(\varphi(k))T_s \\ \varphi(k+1) &= \varphi(k) \end{aligned} \tag{3.11}$$

Za želeno gibanje vozila znotraj intervala vzorčenja $t \in kT_s, (k+1)T_s$ lahko uporabimo inverzno kinematiko za izračun potrebnih vhodov v sistem. Vhode izrazimo iz (3.10) in (3.11).

Kot je že bilo omenjeno je možno več različnih gladkih poti, ki pripeljejo vozilo v želeno lego in je zato inverzna kinematika težavna za izvedbo. Inverzna kinematika pa je enostavna, če imamo predpisano želeno gladko trajektorijo ($x(t)$, $y(t)$), ki naj jo vozilo sledi, tako da bo njegova orientacija vedno tangentna na trajektorijo. Trajektorija je definirana v časovnem intervalu $t \in [0, T]$. Če predpostavimo, da je začetna lega vozila na želeni trajektoriji in če imamo idealni kinematični model, potem lahko izračunamo potrebne vhode v in ω kot sledi

$$v(t) = \pm\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \tag{3.12}$$

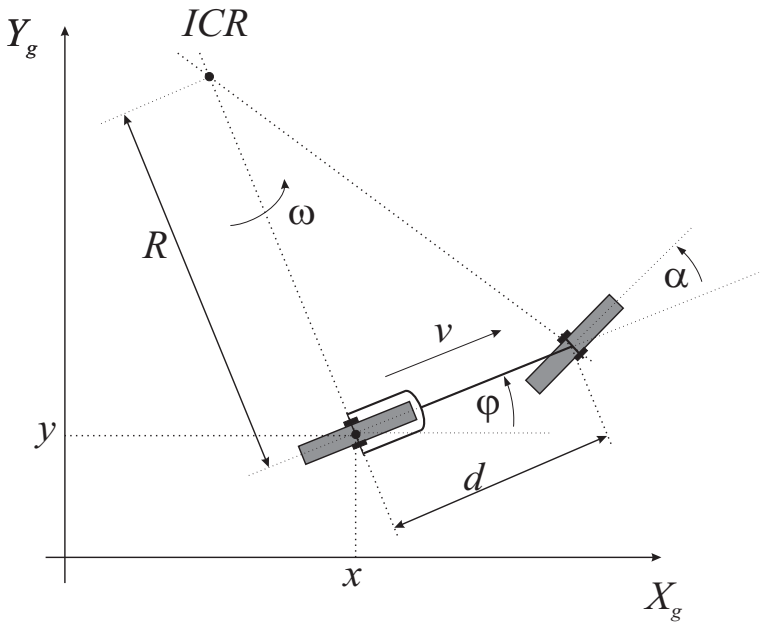
kjer predznak določa želeno smer vožnje (+ za naprej in – za vzvratno vožnjo). Kot tangente v vsaki točki na trajektoriji je določen z

$$\theta(t) = \arctan2(\dot{y}(t),\ \dot{x}(t)) + k\pi \tag{3.13}$$

kjer k=0,1 definira želeno smer vožnje (0 za naprej in 1 za nazaj) in funkcija arctan2 predstavlja štirikvadrantno inverzno funkcijo tangens. Z odvajanjem (3.13) po času dobimo kotno hitrost vozila $\omega(t)$

$$\omega(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{\dot{x}^2(t) + \dot{y}^2(t)} = v(t)\kappa(t) \tag{3.14}$$

kjer je $\kappa(t)$ ukrivljenost trajektorije. Z uporabo relacij (3.12) in (3.14) ter predpisane referenčne poti vozila $x(t)$, $y(t)$ lahko izračunamo potrebne krmilne vhode $v(t)$ and $\omega_(t)$. Potreben pogoj pri načrtovanju poti je, da je dvakrat odvedljiva in da se ne ustavi (tangencialna hitrost $v(t) \neq 0$). Če je pri nekem času t tangencialna hitrost $v(t)$=0 se robot vrti na mestu s krožno



Slika 3.3: Kinematika kolesnega pogona.

hitrostjo $\omega(t)$. Kot $\theta(t)$ ne moremo določiti iz enačbe (3.12) in mora biti torej eksplicitno podan. Prikazano inverzno kinematiko za znano trajektorijo lahko uporabimo pri vodenju kot predkrmiljenje (ang. feedforward), ki je dodatek povratnozančnemu vodenju, ki poskrbi za odpravo motenj, vplivov zaradi netočnega modela kinematike in začetnih napak lege vozila [1].

3.2.2 Kolesni pogon

Kolesni pogon, prikazan na sliki 3.3, ima krmilno kolo s kotom krmiljenja α in se kotali s kotno hitrostjo ω_s (pogon na prednje kolo). Točka ICR je določena s presečiščem osi prednjega in zadnjega kolesa. V danem trenutku kolo kroži (sledimo točko na osi zadnjega kolesa) okoli ICR s kotno hitrostjo ω in radijem R .

$$R(t) = d \tan\left(\frac{\pi}{2} - \alpha(t)\right) = \frac{d}{\tan\left(\alpha(t)\right)}$$

V danem trenutku velja, da okoli ICR kroži poljubna točka togega telesa s kotno hitrostjo ω. Krmilno kolo torej tudi kroži okoli ICR z ω, zato lahko zapišemo

$$\omega(t) = \dot{\varphi} = \frac{v_s(t)}{\sqrt{d^2 + R^2}} = \frac{v_s(t)}{d} \sin\left(\alpha(t)\right)$$

kjer je $v_s(t) = \omega_s(t)r$ obodna hitrost krmilnega kolesa in r je radij krmilnega kolesa.

Notranja kinematika vozila je določena z

$$\begin{aligned} \dot{x_m} &= v_s(t)\cos(\alpha(t)) \\ \dot{y_m} &= 0 \\ \dot{\varphi} &= \tfrac{v_s(t)}{d}\sin(\alpha(t)) \end{aligned} \tag{3.15}$$

zunanja kinematika pa z

$$\begin{aligned} \dot{x} &= v_s(t)\cos(\alpha(t))\, cos(\varphi(t)) \\ \dot{y} &= v_s(t)\cos(\alpha(t))\, sin(\varphi(t)) \\ \dot{\varphi} &= \tfrac{v_s(t)}{d}\sin(\alpha(t)) \end{aligned} \tag{3.16}$$

oziroma z matričnim zapisom

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ \sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3.17}$$

kjer $v = v_s(t)\cos(\alpha(t))$ in $\omega(t) = \frac{v_s(t)}{d}\sin(\alpha(t))$.

Kolesni pogon s pogonom na zadnje kolo

Večinoma imajo vozila (kolo, tricikel in nekateri avtomobili) pogon na zadnja kolesa. V teh primerih sta regulirni veličini hitrost zadnjega kolesa $v_r(t)$ in kot krmiljenja $\alpha_r(t)$ krmilnega kolesa. Notranjo kinematiko lahko enostavno izpeljemo iz (3.15) kjer upoštevamo $v_r = v_s(t)\cos(\alpha(t))$

$$\begin{aligned} \dot{x_m}(t) &= v_r(t) \\ \dot{y_m}(t) &= 0 \\ \omega(t) = \dot{\varphi} &= \tfrac{v_r(t)}{d}\tan(\alpha(t)) \end{aligned} \tag{3.18}$$

Zunanja kinematika je

$$\begin{aligned} \dot{x} &= v_r(t)\cos(\varphi(t)) \\ \dot{y} &= v_r(t)\sin(\varphi(t)) \\ \dot{\varphi} &= \tfrac{v_r(t)}{d}\tan(\alpha(t)) \end{aligned} \tag{3.19}$$

oziroma v matrični obliki

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ \sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r(t) \\ \omega(t) \end{bmatrix} \tag{3.20}$$

kjer je $\omega(t) = \frac{v_r(t)}{d}\tan(\alpha(t))$.

Direktna in inverzna kinematika

Upoštevajoč relacijo (3.17) lahko zapišemo direktno kinematiko kolesa s sprednjim pogonom z (3.4) kot pri diferencialnem pogonu.

Rešitev inverzne kinematike je v splošnem zelo zahtevna, lahko pa si problem precej olajšamo, če vpeljemo strategijo gibanja z dvema osnovnima načinoma premikov. Pri način predstavlja premo gibanja v smeri naprej ($\alpha(t) = 0$), drugi način pa kroženje na mestu ($\alpha(t) = \pm\frac{\pi}{2}$). Pri premem gibanji se hitrosti vozila poenostavijo v $v(t) = v_s(t)$ in $\omega(t) = 0$. Z vstavitvijo teh hitrosti in z diskretno integracijo dobimo sledeče enačbe gibanja

$$\begin{aligned} x(k+1) &= x(k) + v_s(k)cos\left(\varphi(k)\right)T_s \\ y(k+1) &= y(k) + v_s(k)sin\left(\varphi(k)\right)T_s \\ \varphi(k+1) &= \varphi(k) \end{aligned} \tag{3.21}$$

V primeru kroženja na mestu pa se hitrosti vozila poenostavijo v $v(t) = 0$ in $\omega(t) = \frac{v_s(t)}{d}$. Z vstavitvijo teh hitrosti v (3.17) in diskretizacijo dobimo sledeče enačbe gibanja

$$\begin{aligned} x(k+1) &= x(k) \\ y(k+1) &= y(k) \\ \varphi(k+1) &= \varphi(k) + \tfrac{v_s(t)}{d}T_s \end{aligned} \tag{3.22}$$

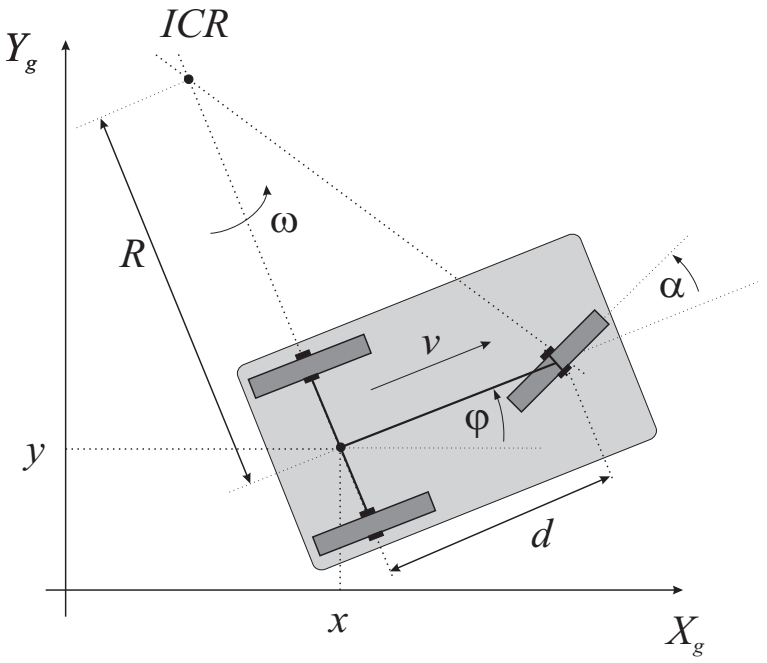
Potrebna vhoda v sistem (regulirni veličini) lahko določimo iz (3.21) in (3.22) za želeno gibanje med časi vzorčenja.

3.2.3 Trikolesni pogon

Trikolesni pogon (slika 3.4) ima enako kineamatiko kot kolesni pogon. Ker pri trikolesnem pogonu sledimo točko na sredini zadnje osi je kinematika enaka kot pri kolesnem pogonu. Kolesni pogon je le posebni primer trikolesnega pogona, ko je razdalja med zadnjimi kolesi enaka nič

$$\begin{aligned} \dot{x} &= v_s(t)\cos\left(\alpha(t)\right)cos\left(\varphi(t)\right) \\ \dot{y} &= v_s(t)\cos\left(\alpha(t)\right)sin\left(\varphi(t)\right) \\ \dot{\varphi} &= \tfrac{v_s(t)}{d}\sin\left(\alpha(t)\right) \end{aligned} \tag{3.23}$$

kjer je $v = v_s(t)\cos\left(\alpha(t)\right)$, $\omega(t) = \frac{v_s(t)}{d}\sin\left(\alpha(t)\right)$ in v_s je obodna hitrost krmilnega kolesa. Trikolesni pogon je v praksi pogost, saj tri kolesa zagotavljajo stabilnost vozila v vertikalni smeri (pomožna podporna oz. kastor kolesa zato niso potrebna).



Slika 3.4: Kinematika trikolesnega pogona.

3.2.4 Tricikel s priklopnikom

Kinematiko tricikla je določena v poglavju 3.2.3. Za priklopnik določimo točko ICR_2, ki leži na presečišču zadnje osi tricikla in osi priklopnika. Kotna hitrost s katero kolesa priklopnika krožijo okoli ICR_2 je

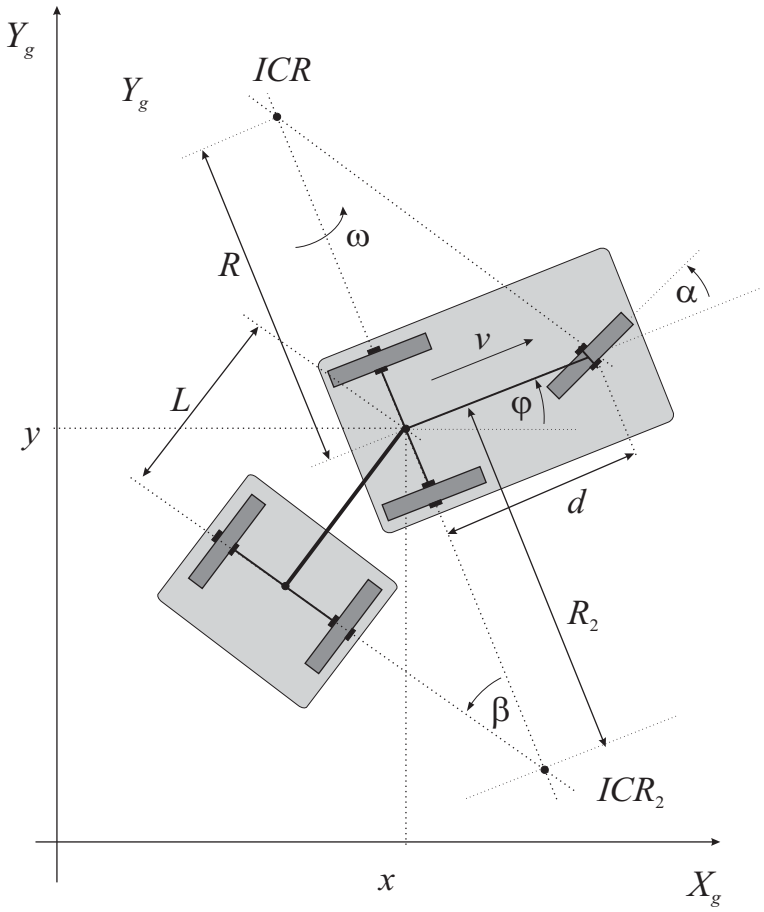
$$\omega_2(t) = \frac{v}{R_2} = \frac{v_s \cos\alpha}{R_2} = \frac{v_s \cos\alpha \sin\beta}{L} = \dot{\beta}$$

končna kinematika vozila na sliki 3.5 je določena z

$$\begin{aligned} \dot{x} &= v_s(t)\cos(\alpha(t))\, cos(\varphi(t)) \\ \dot{y} &= v_s(t)\cos(\alpha(t))\, sin(\varphi(t)) \\ \dot{\varphi} &= \tfrac{v_s(t)}{d}\sin(\alpha(t)) \\ \dot{\beta} &= \tfrac{v_s \cos\alpha \sin\beta}{L} \end{aligned} \tag{3.24}$$

3.2.5 Avtomobilski (Akerman) pogon

Avtomobilski pogon uporablja akermanov princip krmiljenja. Osnovna ideja akermanovega krmiljenja je, da ima notranje kolo (tisto bližje ICR) večji zasuk krmiljenja kot zunanje, saj to omogoča, da se vozilo vrti okoli središčne



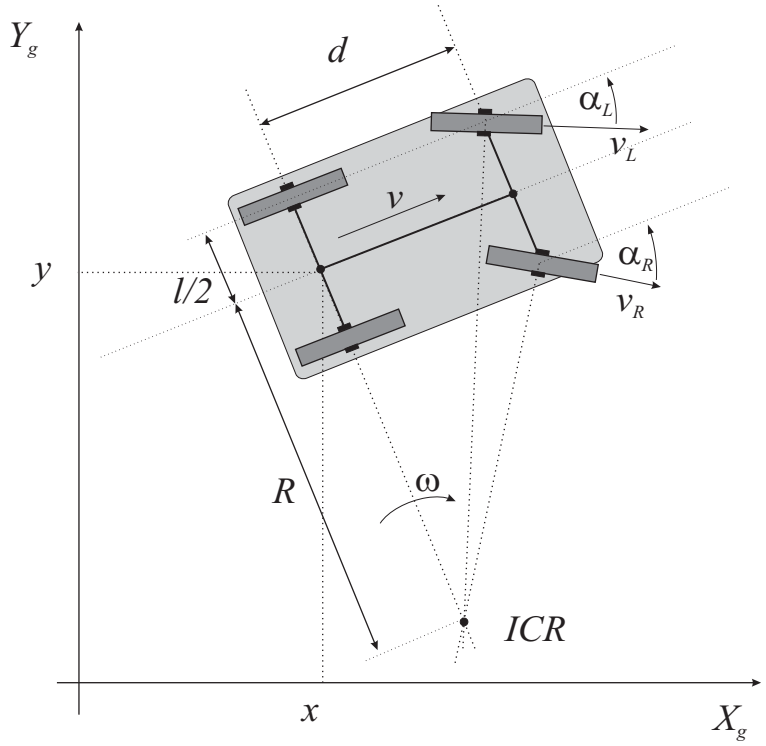
Slika 3.5: Kinematika tricikel s priklopnikom.

točke na osi zadnjih koles. Posledično ima notranje kolo manjšo obodno hitrost kot zunanje kolo. Akermanovo krmiljenje omogoča vrtenje zadnjih koles brez zdrsov in je zato točka ICR leži na premici, ki gre skozi zadnjo os. Ta krmilni mehanizem omogoča manjšo obrabo pnevmatik. Glede na sliko 3.6 lahko določimo orientacijo prednjih krmilnih koles iz

$$\begin{aligned} \tan\left(\tfrac{\pi}{2} - \alpha_{outer}\right) &= \tfrac{R+\frac{l}{2}}{d} \\ \tan\left(\tfrac{\pi}{2} - \alpha_{inner}\right) &= \tfrac{R-\frac{l}{2}}{d} \end{aligned}$$

od koder izrazimo kote krmiljenja

$$\begin{aligned} \alpha_{outer} &= \tfrac{\pi}{2} - \arctan \tfrac{R+\frac{l}{2}}{d} \\ \alpha_{inner} &= \tfrac{\pi}{2} - \arctan \tfrac{R-\frac{l}{2}}{d} \end{aligned} \tag{3.25}$$



Slika 3.6: Akermanov pogon.

Zunanje in notranje zadnje kolo krožita okoli ICR z enako kotno hitrostjo ω torej je njuna obodna hitrost enaka

$$\begin{aligned} v_{outer} &= \omega \left(R + \tfrac{l}{2}\right) \\ v_{inner} &= \omega \left(R - \tfrac{l}{2}\right) \end{aligned} \tag{3.26}$$

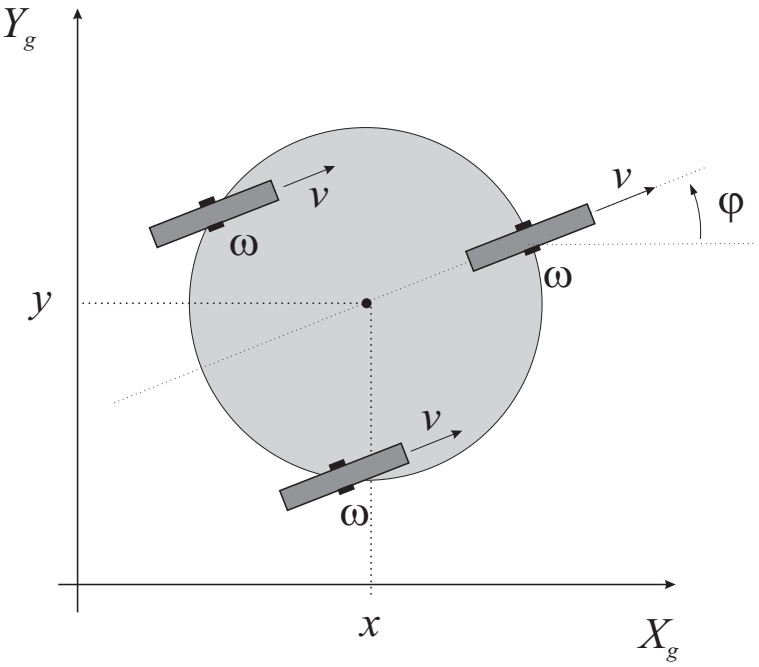
Na sliki 3.6 je notranje kolo desno kolo $v_{INNER} = v_R$ in zunanje kolo je levo kolo $v_{OUTER} = v_L$.

Akermanov kinematični pogon je primeren za modeliranje gibanja večjih vozil. Model gibanja pa lahko opišemo tudi z uporabo kinematike tricikla (3.23), kjer uporabimo povprečen akermanov kot krmiljenja $\alpha = \frac{\pi}{2} - \arctan \frac{R}{d}$. Inverzna kinematika akermanovega pogona je zahtevna in presega namen tega dela.

3.2.6 Sinhroni pogon

Sinhrono vozilo lahko krmili po smeri vsa svoja kolesa sinhrono (vsa kolesa imajo enako orientacijo v danem trenutku). Tipično ima vozilo s sinhronim pogonom tri kolesa razvrščena simetrično (v enakostraničnem trikotniku)

okoli središča vozila (slika 3.7). Vsa kolesa sinhronega pogona so krmiljena



Slika 3.7: Sinhroni pogon.

sinhrono, torej so njihove osi vrtenja zmeraj vzporedne in je zato točka ICR v neskončnosti. Vozilo lahko neposredno spreminja svojo orientacijo z orientacijo krmilnih koles. Vhodi (regulirne veličine) vozila so hitrost krmiljenja koles ω in njihova obodna hitrost v.

Kinematika vozila s sinhronim pogonom je podobna kot kinematika diferencialnega pogona.

$$\begin{bmatrix} \dot{x(t)} \\ \dot{y(t)} \\ \dot{\varphi(t)} \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ \sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3.27}$$

kjer sta $v(t)$ in $\omega(t)$ vhoda, ki jih pri vodenju lahko spreminjamo neodvisno (kar pri diferencialnem pogonu ni možno).

Direktna in inverzna kinematika

Direktno kinematiko dobimo z integracijo kinematičnega modela (3.27).

$$\begin{aligned} x(t) &= \int_0^t v(t)\cos(\varphi(t))dt \\ y(t) &= \int_0^t v(t)\sin(\varphi(t))dt \\ \varphi(t) &= \int_0^t \omega(t)dt \end{aligned} \tag{3.28}$$

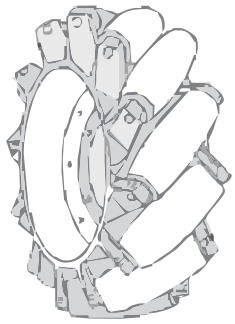
Splošna rešitev inverzne kinematike ni možna ker obstaja več možnosti, kako prispeti v želeno ciljno lego. Inverzna kinematika pa je enostavno rešljiva, če predpostavimo poseben primer kjer se vozilo ali vrti na mestu ali se le premo giblje v smeri trenutne orientacije (brez rotacije). Ko se vozilo vrti na mestu s konstantno krožno hitrostjo ω določen časovni interval Δt se njegova orientacija spremeni za $\omega\Delta t$. V primeru premega gibanja s konstantno hitrostjo $v\Delta t$, ki traja čas Δt, pa se vozilo premakne za $v\Delta t$ v smeri trenutne orientacije.

3.2.7 Večsmerni pogon

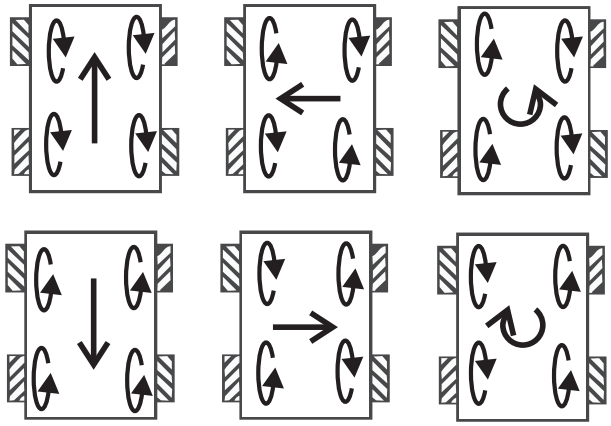
V predhodno opisanih kinematičnih modelih so bila uporabljena preprosta kolesa, ki se lahko le kotalijo v smeri njihove orientacije (npr. diferencialni pogon). Taka preprosta kolesa imajo le eno možno smer kotaljenja. Da dosežemo večsmerno kotaljenje (v več smereh) potrebujemo bolj kompleksen tip koles. Primer takega kolesa je kolo Mecanum oziroma Švedsko kolo, ki ima po obodu kolesa razvrščenih več valčkov. Osi valčkov niso vzporedne z osjo glavnega kolesa (tipično so pod kotom 45°). To omogoča številne različne načine uporabe. V osnovnem načinu so valjčki blokirani in glavno kolo tako predstavlja navadno kolo. Če blokiramo glavno kolo in se vrtijo valjčki to omogoča gibanje vozila v različnih smereh (pod kotom 45° glede na osnovni način). Poljubna kombinacija teh dveh osnovnih načinov pa omogoča gibanje v vseh smereh.

Z izmenjavo koles z levo in desnosučnimi valjčki, tako da vsako kolo povzroči silo približno pravokotno na diagonalo vozila, dosežemo stabilnost vozila in možnost njegovega premika v poljubni smeri in poljubno rotacijo. To dosežemo s spreminjanjem hitrosti in smeri vrtenja glavnih koles. Če vrtimo vsa štiri kolesa v isto smer, dosežemo gibanje vozila naprej ali nazaj. Z vrtenjem koles na levi strani v nasprotno smer, kot tista na desni, dosežemo rotacijo vozila. Z vrtenjem koles na eni diagonali v nasprotno smer kot kolesa na drugi diagonali dobimo gibanje v bočni smeri. Kombinacija vseh omenjenih gibanj omogoča gibanje v poljubni smeri in rotaciji.

Drug primer kompleksnega kolesa je kolo omni, ki omogoča večsmerno gibanje podobno kot kolo Mecanum. Kolo omni (slika 3.10) ima valčke



Slika 3.8: Kolo Mecanum z valjčki nameščenimi okoli oboda kolesa. Vsak valjček ima os vrtenja pod 45° kotom glede ravnino koles in pod kotom 45° glede na linijo vzporedno z osjo kolesa.



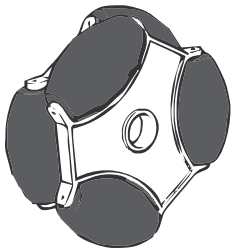
Slika 3.9: Osnovne smeri premikov vozila z večsmernim pogonom z uporabo koles Mecanum.

razvrščene po obodu glavnega kolesa, tako da je njihova os pravokotna glede na os kolesa.

3.2.8 Gosenični pogon

Njihovo gibanje lahko približno opišemo s kinematiko diferencialnega pogona. Diferencialni pogon predpostavlja idealno kotaljenje koles s točkovnim kontaktom kolesa in podlage, kar pa ne drži v primeru goseničnega pogona.

Gosenični pogon ima večjo kontaktno površino s tlemi, kar pomeni, da morajo gosenice drseti pri spreminjanju smeri. Zdrs med gosenicami in podlago ni konstanten, saj je odvisen od vrste podlage. Zato je odometrija



Slika 3.10: Kolo omni s šestimi prostovrtečimi valjčki razvrščenimi po obodu kolesa. kolo se lahko vrti in "drsi"(vrtijo s valjčki) bočno.

še manj primerna za ocenjevanje lege vozila kot pri diferencialnem pogonu. Prednost goseničnega pogona je, da se lahko giblje po zahtevnejših terenih, kjer so ostala kolesna vozila manj uspešna. To mu omogoča večja stična površina vozila s podlago.

3.3 Omejitve gibanja

Pri gibanju mobilnega kolesnega robota se srečamo z dinamičnimi in kinematičnimi omejitvami. *Dinamične omejitve* izvirajo iz dinamičnega modela sistema, ki ima omejeno odzivnost oziroma pospeševanje zaradi svoje vztrajnosti (mase) in omejitev motornega pogona (omejen navora motorja zaradi njegovih zmoglivosti ali preprečevanja zdrsavanja koles). *Kinematične omejitve* pa izvirajo iz zgradbe sistema oziroma iz njegovega kinematičnega modela. Predvsem so zanimive kinematične omejitve, ki jih ločimo na holonomične in neholonomične omejitve. *Neholonomične omejitve* omejujejo možne smeri premika mobilnega sistema. *Holonomične omejitve* pa se nanašajo le na dimenzijo opisa sistema s posplošenimi koordinatami z njihovo pomočjo lahko eliminiramo odvečne (odvisne od drugih) posplošene koordinate.

Nek sistem je holonomičen, če nima kinematičnih omejitev ali ima le holonomične omejitve. Za holonomične sisteme velja, da nimajo omejitev v smeri gibanja. Neholonomični sistem pa je sistem, ki ima neholonomične omejitve, torej se v danem primeru ne more premikati v poljubni smeri (npr. avtomobil se lahko premika le v smeri vrtenja koles, ne more pa se premikati bočno). Za holonomične sisteme lahko določimo nabor neodvisnih posplošenih koordinat, ki določajo prostor, v katerem so možne poljubne smeri gibanja. V neholonomičnih sistemih temu ni tako, saj gibanje v vsakem trenutku ni poljubno (prosto), dovoljena so le tista gibanja, ki ustrezajo neholonomičnim omejitvam. Za holonomične sisteme torej velja, da se z vrnitvijo notran-

jih spremenljivk sistema v začetno stanje (konfiguracija - zasuki koles, koti sklepov), sistem (mobilni robot manipulator) vrne v začetno pozicijo in orientacijo. V primeru neholonomičnih sistemov to ne drži, saj vrnitev notranjih spremenljivk v začetno konfiguracijo ne zagotavlja tudi vrnitve sistema v začetno lego (pozicijo in orientacijo). Povedano bolj splošno lahko rečemo, da je izhodno stanje neholonomičnih sistemov odvisno od opravljene poti.

Obravnavali bomo mehanske sisteme, katerih konfiguracijo (lega telesa na okolje in delov telesa med seboj) lahko opišemo z vektorjem posplošenih koordinat $\mathbf{q}$. Pri podani trajektoriji $\mathbf{q}(t)$ definiramo vektor posplošenih hitrosti $\dot{\mathbf{q}}(t)$

Holonomične omejitve lahko izrazimo v obliki enačb, ki povezujejo posplošene koordinate. Te enačbe lahko nato uporabimo pri eliminaciji nekaterih spremenljivk, tako da dobimo manjši prostor posplošenih spremenljivk, ki opišejo sistem. Take omejitve imenujemo holonomične omejitve. Neholonomične omejitve pa ne zmanjšujejo dimenzije prostora posplošenih spremenljivk, ampak vplivajo na zmanjšanje dimenzije prostora možnih posplošenih hitrosti. Holonomične omejitve tako bistveno ne spremenijo problema planiranja poti, medtem ko so neholonomične omejitve v tem smislu težavnejše za obravnavo. V zvezi z njimi se pojavljajo naslednja vprašanja:

- Kako ugotoviti ali je kinematična omejitev neholonomična? Če je omejitev integrabilna, se enačba, ki vsebuje hitrostne parametre (odvode posplošenih koordinat), lahko prevede v holonomično omejitev.

- Ali neholonomična omejitev omejuje množico dostopnih konfiguracij (leg) iz dane konfiguracije (lege)? Z uporabo orodij teorije vodenja lahko pridemo do preprostih pogojev, pod katerimi neholonomične omejitve ne vplivajo na območje dosegljivih leg.

- Kako zgraditi generator izvedljivih oz. možnih poti za robota z neholonomičnimi omejitvami?

3.3.1 Holonomične omejitve

So omejitve vezana na posplošene koordinate (stanja) sistema. Sistem z n posplošenimi koordinatami$\mathbf{q} = [q_1, \cdots, q_n]^T$ je holonomična omejitev v obliki

$$f(\mathbf{q}) = \mathbf{f}(q_1, \cdots, q_n) = 0 \tag{3.29}$$

kjer je f gladka funkcija z zveznim odvodom. Ta omejitev določa podmnožico vseh možnih konfiguracij v posplošenih koordinatah (delovni prostor), ki zadostijo omejitvi (3.29), oz. zmanjša število prostostnih stopenj sistema. Z

upoštevanjem (3.29) lahko namreč eliminiramo določeno posplošeno koordinato (izrazimo jo z $n-1$ ostalimi koordinatami).

V splošnem imamo lahko m holonomskih omejitev ($m < n$). Če so omejitve linearno neodvisne, potem določajo $n-m$ dimenzionalni podprostor, ki je dejanski konfiguracijski oz. delovni prostor sistema (sistem ima $n-m$ prostostnih stopenj).

3.3.2 Neholonomične omejitve

So omejitve možnih hitrosti sistema oziroma možnih smeri gibanja sistema, zapišemo jih v obliki

$$f(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{f}(q_1, \cdots, q_n, \dot{q}_1, \cdots, \dot{q}_n) = 0 \tag{3.30}$$

kjer je f gladka funkcija z zveznim odvodom in $\dot{\mathbf{q}}$ vektor hitrosti sistema v posplošenih koordinatah. V primeru, da sistem nima omejitev (3.30), potem se lahko giblje v poljubnih smereh.

Kinematična omejitev (3.30) je holonomična, če je integrabilna, to je, če lahko hitrosti $\dot{q}_1, \cdots, \dot{q}_n$ eliminiramo in enačbo (3.30) zapišemo v obliki (3.29). Če to ni mogoče (omejitev ni integrabilna), je omejitev neholonomična.

Če imamo m neodvisnih neholonomičnih omejitev v obliki (3.30), ima prostor dostopnih hitrosti dimenzijo $n-m$. Neholonomična omejitev torej omeji dovoljene hitrosti sistema. Za primer lahko vzamemo dvokolesni robot (invalidski voziček), ki se lahko premika v smeri vrtenja koles, v bočni smeri (pravokotno na kolesa) pa ne.

Če predpostavimo, da so omejitve (3.30) linearne v odvisnosti od $\dot{\mathbf{q}} = [\dot{q}_1, \cdots, \dot{q}_n)]^T$, lahko (3.30) zapišemo kot

$$f(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{a}^T(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} a_1(\mathbf{q}) & \cdots & a_n(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = 0 \tag{3.31}$$

kjer je $\mathbf{a}(\mathbf{q})$ vektor členov omejitve (nedovoljena smer pomika), ki so odvisni od odvoda posplošenih koordinat.

V kolikor imamo m neholonimičnih omejitev, lahko njihove člene zapišemo v matriko omejitev

$$\mathbf{A}(\mathbf{q}) = \begin{bmatrix} \mathbf{a}_1^T(\mathbf{q}) \\ \vdots \\ \mathbf{a}_m^T(\mathbf{q}) \end{bmatrix} \tag{3.32}$$

in vse neholonomične omejitve sistema podamo v matrični obliki

$$\mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0} \tag{3.33}$$

Nadalje definirajmo matriko dosegljivih smeri gibanja sistema (če imamo m neholonomičnih omejitev, je dosegljivih smeri $n-m$) $\mathbf{S}(\mathbf{q}) = [\mathbf{s}_1(\mathbf{q}), \cdots, \mathbf{s}_{n-m}(\mathbf{q})]$. Ta matrika definira kinematični model sistema za katerega velja

$$\dot{\mathbf{q}}(t) = \mathbf{S}(\mathbf{q})\mathbf{v}(t) \tag{3.34}$$

in kjer je $\mathbf{v}(t)$ vektor vhodnih spremenljivk (npr. glej kinamatični model (3.2)). Velja, da je produkt matrike omejitev $\mathbf{A}$ in matrike dosegljivih smeri $\mathbf{S}$ ničelna matrika, torej $\mathbf{AS} = \mathbf{0}$.

3.3.3 Integrabilnost omejitev

Če želimo ugotoviti, ali je določena omejitev resnično hitrostna oziroma neholonomična moramo preveriti, ali jo je mogoče integrirati in s tem prevesti v holonomično omejitev. V kolikor to ni mogoče, je omejitev neholonomična.

3.3.4 Vektorska polja, porazdelitev, Liejevi oklepaji

Najprej definirajmo pojem *porazdelitev*. Možne smeri premikov iz določene točke prostora $\mathbf{q}$ dobimo z linearno kombinacijo vektorskih polj v matriki dosegljivih smeri $\mathbf{S}$. Porazdelitev tako podaja dosegljiv podprostor iz določene točke prostora $\mathbf{q}$ s premiki, ki predstavljajo linearno kombinacijo vektorskih polj v matriki $\mathbf{S}$.

Vektorska polja so odvodi posplošenih koordinat, torej hitrosti oziroma smeri možnih premikov v prostoru. Vektorsko polje je zvezno odvedljiva preslikava, ki vsaki točki prostora priredi natanko določen vektor. Prikaz določitve dosegljivih vektorskih polj je podan v primeru 3.1.

Primer 3.1. *V primeru kinematike robota z diferencialnim pogonom (3.2) sta vektorja dosegljivih hitrosti oziroma smeri premikov*

$$\mathbf{s}_1(\mathbf{q}) = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{bmatrix} \quad \mathbf{s}_2(\mathbf{q}) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.35}$$

kar pomeni, da so možne smeri premika v danem trenutku, ko se nahajamo v legi $\mathbf{q}$

$$\dot{\mathbf{q}} = u_1\mathbf{s}_1(\mathbf{q}) + u_2\mathbf{s}_2(\mathbf{q})$$

kjer sta u_1 in u_2 poljubni realni števili, ki predstavljata vhoda v sistem (to je le preurejen zapis kinematičnega modela (3.2)).

V kolikor vektorskih polj $\mathbf{s}_i$ nimamo podanih, jih lahko določimo tudi iz znanih omejitev $\mathbf{a}_j$, kjer velja ortogonalnost $\mathbf{s}_i \perp \mathbf{a}_j$. Iz slike 3.2 lahko določimo omejitev, torej smer, kamer se robot ne more premikati - to je bočno na kolesa, torej

$$\mathbf{a}(\mathbf{q}) = \begin{bmatrix} -\sin\varphi \\ \cos\varphi \\ 0 \end{bmatrix}$$

kar je ravno pravokotno na vektor možnega premika $\mathbf{s}_1(\mathbf{q})$ (translatorni premik v smeri kotaljenja koles) in na vektor $\mathbf{s}_2(\mathbf{q})$ (rotacija okoli osi pravokotne na ravnino gibanja).

V kolikor porazdelitev določenih vektorskih polj definira celoten prostor, potem je *porazdelitev vsebovana*. Če pa osnovna porazdelitev ni vsebovana, potem lahko določimo nove vektorje, ki jih v osnovni porazdelitvi ni (so linearno neodvisni od vektorjev osnovne porazdelitve). Te nove smeri lahko dobimo s končnim številom preklopov pomikov (pomike limitiramo proti 0) v smeri vektorskih polj osnovne porazdelitve. Te nove smeri lahko določimo s pomočjo *Liejevih oklepajev*. Praktičen primer uporabe je paralelno parkiranja avtomobila (ali tudi vozila z diferencialnim pogonom). Neposredni bočni premik na parkirno mesto namreč ni mogoč zaradi neholonomičnih omejitev sistema (kolesa ne drsijo bočno), premik v stran pa lahko kljub temu dosežemo z zaporedno kombinacijo gibanja naprej in nazaj in zasukov, kar ilustrira primer 3.2.

Primer 3.2. *Paralelno parkiranja vozila z diferencilanim pogonom. Za podani vektorski polji $\mathbf{s}_1(\mathbf{q})$ in $\mathbf{s}_2(\mathbf{q})$ (glej enačbo (3.35)) določimo novo stanje sistema, če začnemo v stanju $\mathbf{q}_0 = \mathbf{q}(0)$ in se najprej za kratek čas ε gibljemo iz stanja $\mathbf{q}_0$ v smeri $\mathbf{s}_1$, nato v smeri $\mathbf{s}_2$ za čas ε, nato v smeri $-\mathbf{s}_1$ za čas ε ter za čas ε v smeri $-\mathbf{s}_2$. Povedano lahko izrazimo matematično z*

$$\mathbf{q}(4\varepsilon) = \phi_\varepsilon^{-\mathbf{s}_2}\left(\phi_\varepsilon^{-\mathbf{s}_1}\left(\phi_\varepsilon^{\mathbf{s}_2}\left(\phi_\varepsilon^{\mathbf{s}_1}\left(\mathbf{q}_0\right)\right)\right)\right)$$

kar predstavlja nelinearno diferencialno enačbo, katere rešitev (integracija kinematičnega modela) lahko aproksimiramo z razvojem v Taylerjevo vrsto (za podrobnosti glej [33]) kot

$$\mathbf{q}(4\varepsilon) = \mathbf{q}_0 + \varepsilon^2\left(\frac{\partial \mathbf{s}_2}{\partial \mathbf{q}}\mathbf{s}_1(\mathbf{q}_0) - \frac{\partial \mathbf{s}_1}{\partial \mathbf{q}}\mathbf{s}_2(\mathbf{q}_0)\right) + O(\varepsilon^3) \tag{3.36}$$

kjer so parcialni odvodi ovrednoteni v $\mathbf{q}_0$ *in* $O(\varepsilon^3)$ *predstavlja prispevek višjih odvodov, ki je zanemarljiv za kratke čase* ε*. Dobljeni končni premik manevra parkiranja je tako dolžine* ε^2 *v smeri vektorja, ki je podan v oklepaju in predstavlja operacijo Liejev oklepaj, ki ga bomo definirali v nadaljevanju.*

Povedano ilustrirajmo še na številčnem primeru. Predpostavimo, da je začetna lega robota $\mathbf{q}_0 = [0, 0, 0]^T$*. V prvem koraku izvajanja manevra paralelnega parkiranja pridemo v točko*

$$\mathbf{q}_1 = \mathbf{q}_0 + \varepsilon \mathbf{s}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \varepsilon \begin{bmatrix} \cos 0 \\ \sin 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \varepsilon \\ 0 \\ 0 \end{bmatrix}$$

v drugem koraku izvedemo rotacijo

$$\mathbf{q}_2 = \mathbf{q}_1 + \varepsilon \mathbf{s}_2 = \begin{bmatrix} \varepsilon \\ 0 \\ 0 \end{bmatrix} + \varepsilon \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \varepsilon \\ 0 \\ \varepsilon \end{bmatrix}$$

v tretjem koraku imamo translacijo v negativno smer

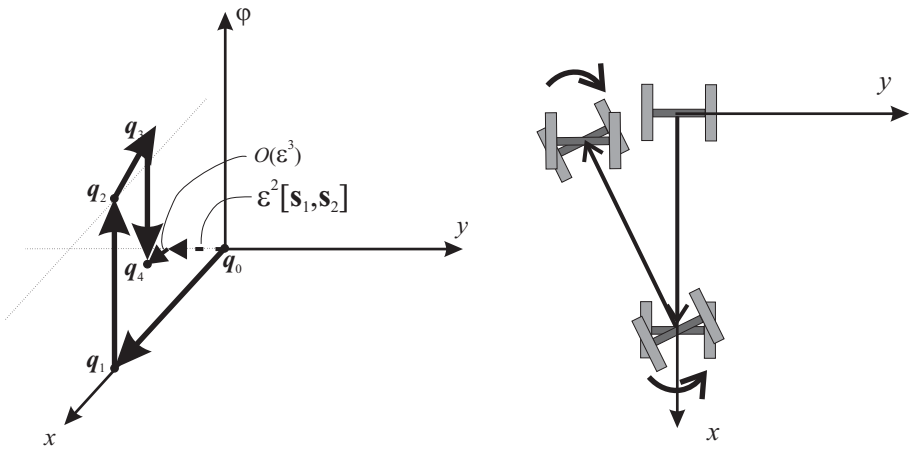
$$\mathbf{q}_3 = \mathbf{q}_2 - \varepsilon \mathbf{s}_1 = \begin{bmatrix} \varepsilon \\ 0 \\ \varepsilon \end{bmatrix} - \varepsilon \begin{bmatrix} \cos \varepsilon \\ \sin \varepsilon \\ 0 \end{bmatrix} = \begin{bmatrix} \varepsilon - \varepsilon \cos \varepsilon \\ -\varepsilon \sin \varepsilon \\ \varepsilon \end{bmatrix}$$

in v zadnjem koraku rotacijo v negativno smer

$$\mathbf{q}_4 = \mathbf{q}_3 - \varepsilon \mathbf{s}_2 = \begin{bmatrix} \varepsilon - \varepsilon \cos \varepsilon \\ -\varepsilon \sin \varepsilon \\ \varepsilon \end{bmatrix} - \varepsilon \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \varepsilon - \varepsilon \cos \varepsilon \\ -\varepsilon \sin \varepsilon \\ 0 \end{bmatrix}$$

Na sliki 3.11 so prikazani izračunani premiki in vmesne točke. Opazimo lahko, da je končni premik v stran, kar ni neposredno (v enem koraku) izvedljivo z možnimi smermi gibanja $\mathbf{s}_1$ *in* $\mathbf{s}_2$*, je pa izvedljivo v večih korakih s kombinacijo možnih smeri gibanj. Dobljeni končni premik ni točno v stran, zaradi končnega časa* ε *in člena* $O(\varepsilon^3)$ *v relaciji (3.36). V kolikor so premiki majhni (*$\varepsilon \to 0$*) je dobljeni premik točno v stran oz. končna točka je*

$$\mathbf{q}_4 = \begin{bmatrix} 0 \\ -\varepsilon^2 \\ 0 \end{bmatrix}$$



Slika 3.11: Manever paralelnega parkiranja, kjer je orientacija predstavljena z osjo z (levo) in prikaz manevra parkiranja (desno, pogled od zgoraj.)

Kot smo videli v primeru 3.2 lahko s pomočjo *Liejevih oklepajev* generiramo nove smeri gibanja, ki jih osnovna porazdelitev ne dovoljuje. Te nove smeri lahko praktično dosežemo s končnim številom neskončno kratkih premikov v smereh vektorskih polj v osnovni porazdelitvi. Liejevi oklepaji so operacija nad dvema vektorskima poljema $\mathbf{i}(\mathbf{q})$ in $\mathbf{j}(\mathbf{q})$, katere rezultat je vektorsko polje $[\mathbf{i}, \mathbf{j}]$. Definiramo ga z

$$[\mathbf{i}, \mathbf{j}] = \frac{\partial \mathbf{j}}{\partial \mathbf{q}}\mathbf{i} - \frac{\partial \mathbf{i}}{\partial \mathbf{q}}\mathbf{j} \tag{3.37}$$

kjer sta

$$\frac{\partial \mathbf{i}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial i_1}{\partial \mathbf{q}_1} & \frac{\partial i_1}{\partial \mathbf{q}_2} & \cdots & \frac{\partial i_1}{\partial \mathbf{q}_n} \\ \frac{\partial i_2}{\partial \mathbf{q}_1} & \frac{\partial i_2}{\partial \mathbf{q}_2} & \cdots & \frac{\partial i_2}{\partial \mathbf{q}_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial i_n}{\partial \mathbf{q}_1} & \frac{\partial i_n}{\partial \mathbf{q}_2} & \cdots & \frac{\partial i_n}{\partial \mathbf{q}_n} \end{bmatrix}$$

$$\frac{\partial \mathbf{j}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial j_1}{\partial \mathbf{q}_1} & \frac{\partial j_1}{\partial \mathbf{q}_2} & \cdots & \frac{\partial j_1}{\partial \mathbf{q}_n} \\ \frac{\partial j_2}{\partial \mathbf{q}_1} & \frac{\partial j_2}{\partial \mathbf{q}_2} & \cdots & \frac{\partial j_2}{\partial \mathbf{q}_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial j_n}{\partial \mathbf{q}_1} & \frac{\partial j_n}{\partial \mathbf{q}_2} & \cdots & \frac{\partial j_n}{\partial \mathbf{q}_n} \end{bmatrix}$$

Določena porazdelitev je vsebovana, če z Liejevimi oklepaji ne moremo generirati novih linearno neodvisnih vektorskih polj, kar pomeni, da je sistem popolnoma holonomičen (nima omejitev gibanja ali pa so vse omejitve holonomične). Ta zadnja ugotovitev izhaja it Frobeniousovega teorema: *Če*

je osnovna porazdelitev vsebovana, potem so vse morebitne omejitve, ki jih sistem ima, integrabilne in sistem je holonomičen.

Recimo, da je iz m omejitev k omejitev holonomičnih in $m-k$ neholonomičnih. Glede na Frobeniousov teorem imamo lahko tri možnosti:

- $k = m$ kar pomeni, da je dimenzija vsebovane porazdelitve (število linearno neodvisnih vektorjev) enaka osnovni porazdelitvi, kar je enako $n - m$.
- $0 < k < m$, imamo k integrabilnih omejite, torej lahko eliminiramo k posplošenih koordinat. V tem primeru je dimenzija vsebovane porazdelitve enaka $n - k$.
- Zadnja možnost pa je, da je $k = 0$ in v tem primeru je dimenzija vsebovana porazdelitve n (dimenzija prostora) in vse omejitve so neholonomične.

Primer 3.3. *Določite omejitve gibanja in smeri možnih premikov za primer enojnega kolesa, ki se kotali po podlagi. Določite število prostostnih stopenj sistema, število in vrsto omejitev.*

Rešitev

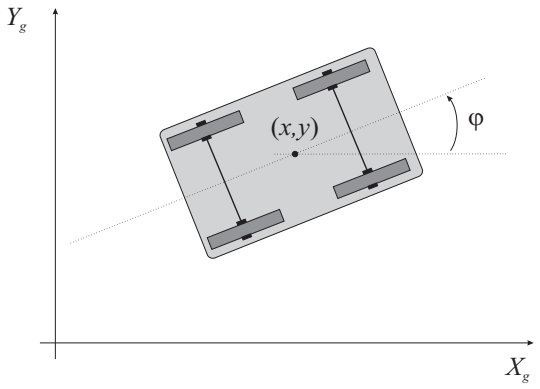
Kinematika enojnega kolesa je enaka kot kinematika diferencialnega pogona (kolesi damo skupaj), glej sliko 3.2 in kinematičen model 3.2. Omejitve smeri gibanja in možne smeri gibanja smo že določili v primeru 3.1. Imamo eno hitrostno omejitev, kar ne omejuje dosegljivih leg $\mathbf{q}$ *v prostoru, zato imamo tri prostostne stopnje. Da je omejitev res hitrostna oziroma neholonomična, lahko pokažemo z uporabo Liejevih oklepajev in Frobeniousovega teorema.*

Določimo najprej Liejev oklepaj za dosegljivi vektorski polji $\mathbf{s}_1$ *in* $\mathbf{s}_2$ *(3.35)*

$$\begin{aligned}
\mathbf{s}_3 = [\mathbf{s}_1, \mathbf{s}_2] &= \tfrac{\partial \mathbf{s}_2}{\partial \mathbf{q}}\mathbf{s}_1 - \tfrac{\partial \mathbf{s}_1}{\partial \mathbf{q}}\mathbf{s}_2 \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & -\sin\varphi \\ 0 & 0 & \cos\varphi \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \sin\varphi \\ -\cos\varphi \\ 0 \end{bmatrix}
\end{aligned}$$

Dobimo novo smer možnega pomika, ki je linearno neodvisna od prvotnih dveh in je ni v osnovni porazdelitvi. Zato sklepamo, da je omejitev neholonomična, dimenzija vsebovane porazdelitve je 3 (3 prostostne stopnje), kar je enako kot je dimenzija prostora. Z nadaljnjimi Liejevimi oklepaji ($[\mathbf{s}_1, \mathbf{s}_3]$, $[\mathbf{s}_2, \mathbf{s}_3]$*) namreč ni možno generirati novih linearno neodvisnih vektorskih polj.*

Primer 3.4. *Določite omejitve gibanja in smeri možnih premikov za primer avtomobila brez krmilnega mehanizma (kolesa so fiksno vpeta). Določite število in vrsto omejitev in število prostostnih stopenj sistema.*



Rešitev

Vozilo se ne more premikati bočno in se ne more vrteti (spreminjati orientacijo φ), torej imamo smeri omejitev gibanj

$$\mathbf{a}_1(\mathbf{q}) = \begin{bmatrix} \sin\varphi \\ -\cos\varphi \\ 0 \end{bmatrix} \quad \mathbf{a}_2(\mathbf{q}) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

imamo torej omejitvi gibanja

$$\mathbf{a}_1(\mathbf{q})^T\dot{(\mathbf{q})} = \dot{x}\sin\varphi - \dot{y}\cos\varphi = 0$$
$$\mathbf{a}_2(\mathbf{q})^T\dot{(\mathbf{q})} = \dot{\varphi} = 0$$

ki sta integrabilni omejitvi, saj jih lahko integriramo in dobimo

$$\varphi = \varphi_0$$
$$(x - x_0)\sin\varphi - (y - y_0)\cos\varphi = 0$$

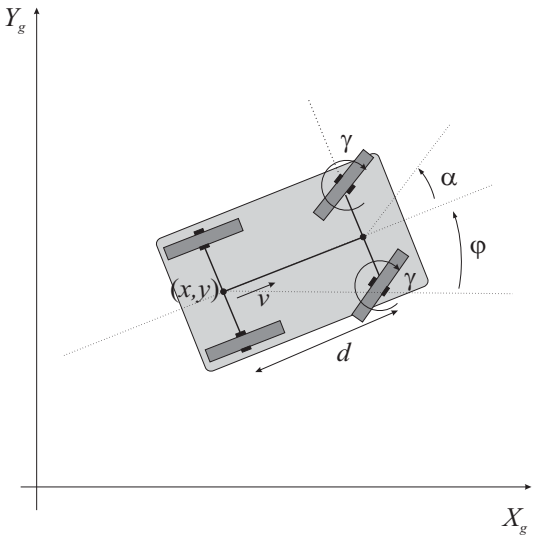
vozilo pa se lahko premika le v smeri vektorja

$$\mathbf{s}_1 = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{bmatrix}$$

Ker sta obe omejitvi holonomični je porazdelitev, ki jo določa $\mathbf{s}_1$ vsebovana in sistem je popolnoma holonomičen, dimenzija vsebovane porazdelitve je 1, sistem ima eno prostostno stopnjo. Z Liejevimi oklepaji ne moremo generirati novih smeri premikov, saj imamo le en vektor možnega premika.

Primer 3.5. *Določite omejitve gibanja in smeri možnih premikov za primer avtomobila s pogonom na zadnja kolesa. Avtomobil upravljamo z obodno hitrostjo vrtenja zadnjih koles v in hitrostjo vrtenja krmilnega mehanizma* γ *(*$\alpha(t) = \int \gamma dt$*).*

Določite število in vrsto omejitev, kinematičen model in število prostostnih stopenj sistema.



Rešitev

Vozilo opišemo s štirimi stanji $\mathbf{q} = [x, y, \varphi, \alpha]^T$*, saj smo kot vhod definirali poleg hitrosti v še hitrost zasuka krmilnega mehanizma* γ*.*

Velja opomniti, da smo v primeru določitve kinematike kolesa (3.19) z zadnjim pogonom (enaka kinematika kot v tem primeru) imeli le tri stanja, saj smo kot vhod definirali zasuk krmilnega mehanizma in ne njegovo kotno hitrost.

Iz slike vozila vidimo, da se vozilo ne more premikati v smeri bočno na zadnja in na prednja kolesa, torej sta vektorja smeri omejitev gibanja

$$\mathbf{a}_1(\mathbf{q}) = \begin{bmatrix} \sin\varphi \\ -\cos\varphi \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{a}_2(\mathbf{q}) = \begin{bmatrix} \sin(\alpha+\varphi) \\ -\cos(\alpha+\varphi) \\ d\cos\alpha \\ 0 \end{bmatrix}$$

Vektorji dovoljenih smeri premikov (gibanj v posplošenih koordinatah) pa so: smer kotaljenja zadnjih koles in vrtenje krmilnih koles (smer spremembe po-

zicije in kota v danem trenutku pri fiksnem zasuku krmilnih koles α).

$$\mathbf{s}_1 = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ \frac{1}{d}\tan\alpha \\ 0 \end{bmatrix} \quad \mathbf{s}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Kinematika vozila je podana z $\dot{\mathbf{q}} = [\mathbf{s}_1, \mathbf{s}_2][v, \gamma]^T$, kar je

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos\varphi & 0 \\ \sin\varphi & 0 \\ \frac{1}{d}\tan\alpha & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \gamma \end{bmatrix}$$

ker je podobno kot v kinematiki (3.19), le da imamo še dodatno stanje α, in vhod $\gamma = \dot{\alpha}$.

Imamo torej 4 stanja, 2 omejitvi in dva vhoda. Želimo imeti štiri linearno neodvisne smeri gibanja $\mathbf{s}_i$. Dve že imamo, poskušamo določiti še preostali dve z Liejevimi oklepaji

$$\begin{aligned} \mathbf{s}_3 = [\mathbf{s}_1, \mathbf{s}_2] &= \tfrac{\partial \mathbf{s}_2}{\partial \mathbf{q}}\mathbf{s}_1 - \tfrac{\partial \mathbf{s}_1}{\partial \mathbf{q}}\mathbf{s}_2 \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ \frac{1}{d}\tan\alpha \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & -\sin\varphi & 0 \\ 0 & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & \frac{1}{d\cos^2\alpha} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ \frac{1}{d\cos^2\alpha} \\ 0 \end{bmatrix} \end{aligned}$$

in

$$\begin{aligned} \mathbf{s}_4 = [\mathbf{s}_1, \mathbf{s}_3] &= \tfrac{\partial \mathbf{s}_3}{\partial \mathbf{q}}\mathbf{s}_1 - \tfrac{\partial \mathbf{s}_1}{\partial \mathbf{q}}\mathbf{s}_3 \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2\sin\alpha}{\cos^3\alpha} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ \frac{1}{d}\tan\alpha \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & -\sin\varphi & 0 \\ 0 & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & \frac{1}{d\cos^2\alpha} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \frac{1}{d\cos^2\alpha} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{\sin\varphi}{d\cos^2\alpha} \\ -\frac{\cos\varphi}{d\cos^2\alpha} \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

Vse dobljene štiri smeri so linearno neodvisne, zato ima sistem štiri prostostne stopnje in obe omejitvi sta neholonomični.

3.4 Dinamični model mobilnega sistema z omejitvami

Kinematični model opisuje le statično transformacijo hitrosti robota (psevdohitrosti) v osnovni koordinatni sistem podan s posplošenimi koordinatami.

Dinamični model gibanja mehanskega sistema pa podaja dinamične zakonitosti v prostoru posplošenih koordinat (gibanje sistema pod vplivom zunanje sile in inercije sistema).

Z uporabo Lagrange-ove formulacije, ki je še posebej primerna za opis mehanskih sistemov, lahko opišemo dinamični model sistema z enačbo

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} + \frac{\partial P}{\partial \dot{q}_k} + g_k + \tau_{d_k} = f_k \tag{3.38}$$

kjer indeks k teče po posplošenih koordinatah q_k ($k = 1 \cdots n$), z L je označena razlika med kinetično in potencialno energijo imenovana Lagrangian, P je močnostna funkcija zaradi trenja in dušenja v sistemu, g_k je sila zaradi gravitacije, τ_{d_k} predstavlja vse neznane motnje, ki jih z enačbo 3.38 nismo zajeli v modelu, f_k pa je posplošena sila (zunanji vpliv na sistem) povezana s posplošeno koordinato q_k. Enačba 3.38 velja za sisteme brez omejitev gibanja, torej za sisteme, ki imajo n prostostnih stopenj in brez hitrostnih omejitev. Za sisteme, ki imajo prisotne kinematične omejitve, lahko dinamične enačbe gibanja sistema zapišemo v obliki z Lagrange-ovimi multiplikatorji

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} + \frac{\partial P}{\partial \dot{q}_k} + g_k + \tau_{d_k} = f_k - \sum_{j=1}^{m} \lambda_j a_{jk} \tag{3.39}$$

kjer je m število linearno neodvisnih omejitvenih enačb gibanja, λ_j so Lagrange-evi multiplikatorji, povezani z j-to omejitveno enačbo, $a_j k$ ($j = 1 \cdots m$, $k = 1 \cdots n$) pa so koeficienti omejitvenih enačb.

Končni nabor enačb vsebuje $n + m$ diferencialnih in algebrajskih enačb (n Lagrange-ovih enačb in m omejitvenih enačb) z $n+m$ neznankami (n posplošenih koordinat q_k in m Lagrange-ovih multiplikatorjev λ_j. Enačbe so diferencialne v smislu posplošenih koordinat in algebrajske glede na Lagrange-ove multiplikatorje.

Splošni dinamični model mehanskega sistema z omejitvami lahko glede na formulacijo (3.39) v matrični obliki zapišemo z enačbo

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \boldsymbol{\tau}_d = \mathbf{E}(\mathbf{q})\mathbf{u} - \mathbf{A}^T(\mathbf{q})\boldsymbol{\lambda} \tag{3.40}$$

kjer so:

$\mathbf{q}$ vektor posplošenih koordinat (dimenzije $n \times 1$),

$\mathbf{M}(\mathbf{q})$ pozitivno definitna matrika mas in vztrajnosti (dimenzije $n \times n$),

$\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})$ vektor Coriolisovih in centrifugalnih členov (dimenzije $n \times 1$),

$\mathbf{F}(\dot{\mathbf{q}})$ vektor sil zaradi trenja oziroma dušenja podlage (dimenzije $n \times 1$),

$\mathbf{G}(\mathbf{q})$ vektor zaradi gravitacijskih sil oziroma navorov (dimenzije $n \times 1$),

$\boldsymbol{\tau}_d$ vektor neznanih motenj, vključno z dinamiko, ki ni zajeta v modelu (dimenzije $n \times 1$),

$\mathbf{E}(\mathbf{q})$ matrika preslikav aktuatorskega prostora v prostor posplošenih spremenljivk (dimenzije $n \times r$),

$\mathbf{u}$ vektor vhodov (dimenzije $r \times 1$),

$\mathbf{A}^T(\mathbf{q})$ matrika kinematičnih omejitev (dimenzije $m \times n$)

$\boldsymbol{\lambda}$ vektor omejitvenih sil (Lagrange-ovi multiplikatorji) (dimenzije $m \times 1$).

3.4.1 Predstavitev dinamičnega modela mobilnega sistema z omejitvami v prostoru stanj

V nadaljevanju bomo prikazali izpeljavo zapisa dinamičnega modela sistema podvrženega m kinematičnim omejitvam v prostoru stanj. Nadalje bomo izvedli delno linearizacijo sistema zapisanega v prostoru stanj z vpeljavo nelinearne povratnozančne relacije [34] in rezultirajoči sistem zapisali kot kinematični model drugega reda.

Dinamični sistem enačb, ki je podvržen m kinematičnim omejitvam, podaja oblika

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \mathbf{E}(\mathbf{q})\mathbf{u} - \mathbf{A}^T(\mathbf{q})\boldsymbol{\lambda} \tag{3.41}$$

kjer smo vpliv neznanih motenj na sistem $\boldsymbol{\tau}_d$ iz (3.40) zanemarili. Kinematični model gibanja pa podaja enačba

$$\dot{\mathbf{q}} = \mathbf{S}(\mathbf{q})\mathbf{v}(t) \tag{3.42}$$

Dinamični model (3.41) in kinematični model (3.42) lahko združimo v enoten zapis v prostoru stanj. Poenoteno obravnavo neholonomičnih in holonomičnih omejitev lahko najdemo v [35], kjer so holonomične omejitve izražene v diferencialni obliki (kot neholonomične).

Če odvajamo relacijo (3.42) po času, dobimo (zaradi enostavnosti zapisa smo v nadaljnjem izpeljevanju izpustili odvisnost od $\mathbf{q}$)

$$\ddot{\mathbf{q}} = \dot{\mathbf{S}}\mathbf{v} + \mathbf{S}\dot{\mathbf{v}} \tag{3.43}$$

z vstavitvijo dobljene relacije v (3.41) ter z zamenjavo posplošenih koordinat $\mathbf{q}$ s psevdohitrostmi $\mathbf{v}$ dobimo

$$\mathbf{M}\dot{\mathbf{S}}\mathbf{v} + \mathbf{M}\mathbf{S}\dot{\mathbf{v}} + \mathbf{V} + \mathbf{F} + \mathbf{G} = \mathbf{E}\mathbf{u} - \mathbf{A}^T\boldsymbol{\lambda} \tag{3.44}$$

Prisotnost Lagrange-ovih multiplikatorjev $\boldsymbol{\lambda}$ zaradi omejitev gibanja, ki jim je sistem podvržen, lahko eliminiramo z upoštevanjem transponirane relacije $\mathbf{AS} = \mathbf{0}$ oziroma transponirane relacije $\mathbf{S}^T\mathbf{A}^T = \mathbf{0}$. Z množenjem (3.44) z matriko $\mathbf{S}^T$ torej dobimo skrčeno obliko dinamičnega modela

$$\mathbf{S}^T\mathbf{M}\dot{\mathbf{S}}\mathbf{v} + \mathbf{S}^T\mathbf{M}\mathbf{S}\dot{\mathbf{v}} + \mathbf{S}^T\mathbf{V} + \mathbf{S}^T\mathbf{F} + \mathbf{S}^T\mathbf{G} = \mathbf{S}^T\mathbf{E}\mathbf{u} \tag{3.45}$$

s čimer smo eliminirali Lagrange-ove multiplikatorje $\boldsymbol{\lambda}$. Z vpeljavo zamenjav $\widetilde{\mathbf{M}} = \mathbf{S}^T\mathbf{M}\mathbf{S}$, $\widetilde{\mathbf{V}} = \mathbf{S}^T\mathbf{M}\dot{\mathbf{S}}\mathbf{v} + \mathbf{S}^T(\mathbf{V} + \mathbf{F} + \mathbf{G})$ in $\widetilde{\mathbf{E}} = \mathbf{S}^T\mathbf{E}$ lahko pregledneje zapišemo

$$\widetilde{\mathbf{M}}\dot{\mathbf{v}} + \widetilde{\mathbf{V}} = \widetilde{\mathbf{E}}\mathbf{u} \tag{3.46}$$

od koder izrazimo vektor psevdopospeškov $\dot{\mathbf{v}}$

$$\dot{\mathbf{v}} = \widetilde{\mathbf{M}}^{-1}\left(\widetilde{\mathbf{E}}\mathbf{u} - \widetilde{\mathbf{V}}\right) \tag{3.47}$$

Če je nadalje izpolnjen pogoj $\det \mathbf{S}^T\mathbf{E} \neq 0$ kar v večini realističnih primerih je, lahko iz enačbe (3.46) izrazimo vhod v sistem kot nelinearno povratnozančno relacijo

$$\mathbf{u} = \widetilde{\mathbf{E}}^{-1}\left(\widetilde{\mathbf{M}}\dot{\mathbf{v}} + \widetilde{\mathbf{V}}\right) \tag{3.48}$$

Če definiramo vektor stanj kot $\mathbf{x} = [\mathbf{q}^T\mathbf{v}^T]^T$ in zapišemo sistem v splošni nelinearni obliki $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}$ (matrika $\mathbf{f}(\mathbf{x})$ vsebuje nelinearno odvisnost od stanj in je zato ne moremo zapisati kot produkt matrike $\mathbf{f}$ in stanj $\mathbf{x}$), dobimo zapis sistema v prostoru stanj

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{S}\mathbf{v} \\ -\widetilde{\mathbf{M}}^{-1}\widetilde{\mathbf{V}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n\times r} \\ \widetilde{\mathbf{M}}^{-1}\widetilde{\mathbf{E}} \end{bmatrix}\mathbf{u} \tag{3.49}$$

kjer je r število vhodov v vektorju vhodov $\mathbf{u}$. Dimenzija vektorja $\mathbf{x}$ je torej enaka $(2n - m)$.

Z relacijo (3.48) smo določili inverzni model sistema, s katerim lahko za želene psevdohitrosti in pospeške sistema izračunamo potreben vhod v

sistem. Z uporabo tako izračunanega vhoda v sistem (3.49) dobimo naslednji skupni model

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{Sv} \\ \mathbf{0}_{(n-m)\times 1} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n\times(n-m)} \\ \mathbf{I}_{(n-m)\times(n-m)} \end{bmatrix} \mathbf{u}_z \tag{3.50}$$

kjer $\mathbf{u}_z$ predstavlja psevdopospeške sistema. Izraz (3.48) lahko torej uporabimo za izračun predikcij vhoda v sistem. Izračunane predikcije lahko pri vodenju uporabimo samostojno, torej odprtozančno vodenje oz. krmiljenje. Če jih uporabimo v kombinaciji z zaprtozančnim vodenjem, pa imamo regulacijo s predkrmiljenjem (angleško: feedforward).

3.4.2 Kinematični in dinamični model vozila z diferencialnim pogonom

Vozilo z differencilanim pogonom 3.2 ima kolesa gnana s pomočjo dveh elektromotorjev. Predpostavimo, da je težišče robota je v geometrijskem središču robota. Masa ohišja vozila je m_c, masa koles skupaj z rotorji elektromotorjev je m_k. Vztrajnostni moment ohišja robota okoli osi z označimo z J_c in vztrajnostni moment koles okoli osi kolesa z J_k.

V praksi se pogosto izkaže, da lahko pri modeliranju predpostavimo, da je masa koles in vztrajnost koles zanemarljiva glede na maso in vztrajnost ohišja, zato bomo v nadaljevanju uporabljali le skupno vztrajnost J in skupno maso robota m. Gibanje robota opišemo s tremi posplošenimi koordinatami $\mathbf{q} = [x, y, \varphi]$. Vhod v sistem pa predstavlja navor na levo in desno kolo τ_l, τ_d.

Kinematičen model in omejitve

Kinematičen model smo že izpeljali v (3.2) in je enak

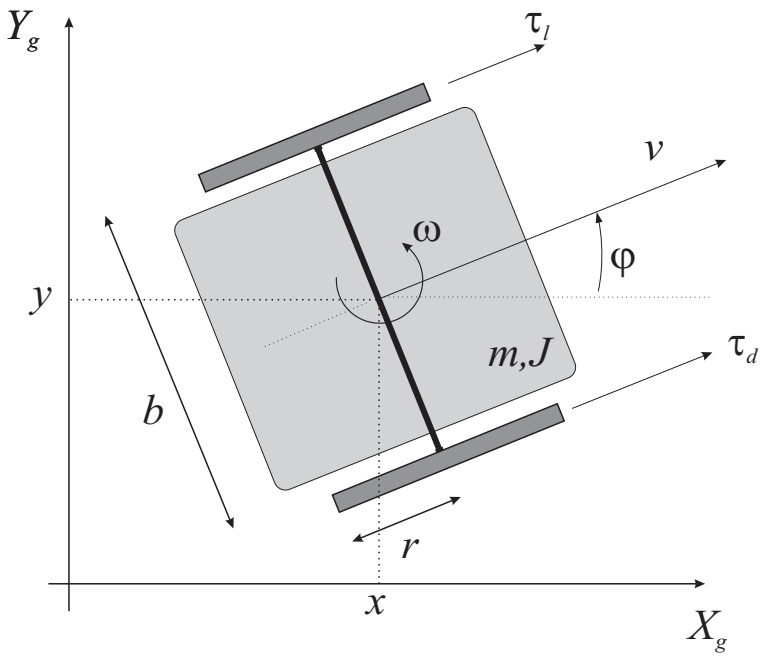
$$\begin{bmatrix} \dot{x(t)} \\ \dot{y(t)} \\ \dot{\varphi(t)} \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ \sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix}$$

Neholonomična omejitev gibanja pa je

$$-\dot{x}\sin\varphi + \dot{y}\cos\varphi = 0$$

torej je matrika dosegljivih smeri premikov

$$\mathbf{S} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ \sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \tag{3.51}$$



Slika 3.12: Vozilo z diferencialnim pogonom.

in matrika omejitev

$$\mathbf{A} = \begin{bmatrix} -\sin\varphi & \cos\varphi & 0 \end{bmatrix} \tag{3.52}$$

Dinamičen model

Dinamičen model izpeljemo z Lagrange-ovo formulacijo

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} + \frac{\partial P}{\partial \dot{q}_k} = f_k - \sum_{j=1}^{m} \lambda_j a_{jk} \tag{3.53}$$

kjer smo glede na (3.39)) opustili člena povezana z vplivi neznanih motenj τ_{d_k} in s silami in navori zaradi gravitacije g_k (gravitacija ne vpliva na gibanje v smeri posplošenih koordinat, saj je potencialna energija $\mathbf{V} = 0$ in $g_k = \frac{\partial \mathbf{V}}{\partial q_k} = 0$).

Skupno kinetično energijo sistema lahko opišemo z relacijo

$$W_K = \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2\right) + \frac{J}{2}\dot{\varphi}^2 \tag{3.54}$$

Ker je potencilana energija sistema $W_P = 0$ je Lagrangian enak

$$L = W_K - W_P = \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2\right) + \frac{J}{2}\dot{\varphi}^2 \tag{3.55}$$

Ker zanemarimo še vpliv dušenja pri kotaljenju koles je $P = 0$. Sile in navori zaradi vztrajnosti v enačbi (3.53) so

$$\begin{array}{l} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = m\ddot{x} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}}\right) = m\ddot{y} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) = J\ddot{\varphi} \end{array} \tag{3.56}$$

nadalje je

$$\begin{array}{l} \frac{\partial L}{\partial x} = 0 \\ \frac{\partial L}{\partial y} = 0 \\ \frac{\partial L}{\partial \varphi} = 0 \end{array} \tag{3.57}$$

Za dinamični model lahko glede na (3.53) zapišemo sledeče diferencialne enačbe

$$\begin{array}{c} m\ddot{x} - \lambda_1 \sin\varphi = F_x \frac{1}{r}(\tau_d + \tau_l) \\ m\ddot{y} + \lambda_1 \cos\varphi = F_y \\ J\ddot{\varphi} = M \end{array} \tag{3.58}$$

kjer rezultanta sil vsota sil na levem in desnem kolesu $F = \frac{1}{r}(\tau_d + \tau_l)$, kjer je r radij kolesa. Komponenta v smeri osi x je $F_x = \frac{1}{r}(\tau_d + \tau_l)\cos\varphi$ v smeri osi y pa $F_y = \frac{1}{r}(\tau_d + \tau_l)\sin\varphi$, navor, ki deluje na vozilo pa je $M = \frac{b}{2r}(\tau_d - \tau_l)$, kjer je b razdalja med kolesi. Imamo torej model

$$\begin{array}{l} m\ddot{x} - \lambda_1 \sin\varphi - \frac{1}{r}(\tau_d + \tau_l)\cos\varphi = 0 \\ m\ddot{y} + \lambda_1 \cos\varphi - \frac{1}{r}(\tau_d + \tau_l)\sin\varphi = 0 \\ J\ddot{\varphi} - \frac{b}{2r}(\tau_d - \tau_l) = 0 \end{array} \tag{3.59}$$

Dobljeni dinamični model lahko zapišemo v matrični obliki

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F}(\dot{\mathbf{q}}) = \mathbf{E}(\mathbf{q})\mathbf{u} - \mathbf{A}^T(\mathbf{q})\boldsymbol{\lambda}$$

z matrikami

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}$$

$$\mathbf{E} = \frac{1}{r}\begin{bmatrix} \cos\varphi & \cos\varphi \\ \sin\varphi & \sin\varphi \\ \frac{b}{2} & -\frac{b}{2} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} -\sin\varphi & \cos\varphi & 0 \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} \tau_d \\ \tau_l \end{bmatrix}$$

ostale matrike pa so ničelne.

Model v prostoru stanj, ki vključuje kinematiko in dinamiko je glede na (3.49) določen z matrikami

$$\widetilde{\mathbf{M}} = \begin{bmatrix} m & 0 \\ 0 & J \end{bmatrix}$$

$$\widetilde{\mathbf{V}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\widetilde{\mathbf{E}} = \frac{1}{r} \begin{bmatrix} 1 & 1 \\ \frac{b}{2} & -\frac{b}{2} \end{bmatrix}$$

od koder lahko glede na enačbo (3.49) zapišemo sistem v prostoru stanj v obliki $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$, kjer je vektor stanj določen z $\mathbf{x} = [\mathbf{q}^T, \mathbf{v}^T]^T$. Dobimo

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\varphi \\ v\sin\varphi \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{mr} & \frac{1}{mr} \\ \frac{b}{2Jr} & \frac{-b}{2Jr} \end{bmatrix} \begin{bmatrix} \tau_d \\ \tau_l \end{bmatrix} \tag{3.60}$$

Nadalje izrazimo inverzni model sistema, zapisanega v prostoru stanj. Inverzni model glede na relacijo (3.48) določimo kot

$$\begin{bmatrix} \tau_d \\ \tau_l \end{bmatrix} = \begin{bmatrix} \frac{\dot{v}mr}{2} + \frac{\dot{\omega}Jmr}{b} \\ \frac{\dot{v}mr}{2} - \frac{\dot{\omega}Jmr}{b} \end{bmatrix} \tag{3.61}$$

na podlagi katerega lahko za neke določene (želene) psevdohitrosti in pospeške robota izračunamo potrebna navora na obe kolesi. Dobljena izračunana vhoda lahko uporabimo za odprtozančno vodenje ali bolje v kombinaciji z zaprtozančnim vodenjem (regulacija s predkrmiljenjem) robota.

Poglavje 4

Vodenje kolesnih mobilnih sistemov

4.1 Uvod

Osnovno vodenje mobilnega kolesnega robota v okolju brez ovir lahko izvedemo z vodenjem od točke do točke (klasično sledilno vodenje, kjer vmeseni potek stanj sistema med začetnim stanjem in stanjem v referenčni točki ni predpisan) ali pa preko sledenja referenčni trajektoriji. V primeru neholonomičnih sistemov se izkaže, da je bolj težavno voditi sistem s povratno zanko k fiksni točki (klasično sledilno vodenje od točke do točke), kot pa proti referenčni trajektoriji (sledenje trajektorije), ki povezuje začetno stanje in stanje v ciljni točki. Za uspešno vodenje moramo uporabiti nezvezno ali časovno spremenljivo povratno zanko [17]. Nadalje mora robot pri gibanju upoštevati neholonomične omejitve, torej njegova pot ne more biti poljubna. Dodatni razlog se skriva tudi v dejstvu, da se mobilni robot pogosto giblje v prostoru, kjer so prisotne omejitve oz. ovire ter razne zahteve, ki v neki meri predpisujejo želeno pot, po kateri naj robot pride do cilja. Navedena dejstva govorijo v prid vodenju po referenčni krivulji, ki mora seveda zadostiti vsem neholonomičnim omejitvam. Gladko vodenje je v tem primeru izvedljivo, saj je linearni približek sistema v okolici trajektorije vodljiv, kar bomo pokazali v nadaljevanju. Neholonomične sisteme je smiselno voditi s predkrmiljenjem (angleško: feedforward), kjer za trajektorijo, ki je možna glede na vse kinematične omejitve, določimo oz. predhodno izračunamo potrebne vhode v sistem, da bo le-ta lahko trajektoriji sledil. Odprtozančno vodenje (samo predkrmiljenje) ni praktično uporabno, saj ni robustno na pogreške začetnih stanj kot tudi ne na motnje v delovanju sistema. Zato moramo za praktično uporabnost pri vodenju realnega robota dodati še zaprtozančni del. Omen-

jena kombinacija (regulacija s predkrmiljenjem) je naravna in prikladna za vodenje neholonomičnih mehanskih sistemov. Ta osnovni princip bo uporabljen v večini prikazanih primerov v nadaljevanju.

4.2 Vodenje vozila z akermanovim pogonom

Kinematičen model akermanovega pogona opiše gibanje kolesa, tricikla in avtomobilskega (ackermanov pogon) pod določenimi predpostavkami (glej podpoglavje 3.2.5). Kinematičen model s pogonom na zadnje kolo in krmiljenjem smeri prednjega kolesa je podan v enačbi (3.19) in prikazan na sliki 3.3 (ter slikah 3.4, 3.6) kot

$$\begin{aligned} \dot{x} &= v_r(t)\cos(\varphi(t)) \\ \dot{y} &= v_r(t)\sin(\varphi(t)) \\ \dot{\varphi} &= \tfrac{v_r(t)}{d}\tan(\alpha(t)) \end{aligned} \tag{4.1}$$

Na vožnjo mobilnega sistema vplivamo z vhodnima spremenljivkama $\alpha(t)$ in $v_r(t)$, ki predstavljata orientacijo prednjega krmilnega kolesa in obodno hitrost zadnjih koles (v primeru tricikla je to hitrost v sredinski točki osi zadnjih koles). Za sistem lahko ločeno obravnavamo vodenje orientacije in translatorne hitrosti.

4.2.1 Vodenje orientacije akermanovega pogona

V danem trenutku imamo podano referenčno (želeno) smer gibanja robota $\varphi_{ref}(t)$, ki jo lahko določimo glede na referenčno trajektorijo gibanja (smer je tangenta na referenčno trajektorijo $\varphi_{ref}(t) = \arctan\frac{\dot{y}_{ref}(t)}{\dot{x}_{ref}(t)}$) ali pa glede referenčno ciljno pozicijo x_{ref}, y_{ref}, kamor želimo, da se robot usmeri ($\varphi_{ref}(t) = \arctan\frac{y_{ref}-y(t)}{x_{ref}-x(t)}$).

Regulator mora zagotoviti, da pogrešek orientacije robota $e_\varphi(t) = \varphi_{ref}(t) - \varphi(t)$ vpliva na zasuk $\alpha(t)$ prednjega krmilnega kolesa. Zasuk krmilnega kolesa naj bo proporcionalen velikosti pogreška orientacije

$$\alpha(t) = K(\varphi_{ref}(t) - \varphi(t)) \tag{4.2}$$

kar definira proporcionalni regulator na procesu integrirnega značaja (glej tretjo vrstico enačbe (4.1)), torej vodenje brez pogreška v ustaljenem stanju pri konstantnem $\varphi_{ref}(t)$. Za majhne kote $\alpha(t)$ lahko določimo linearni model za orientacijo v enačbi (4.1)) kot

$$\dot{\varphi}(t) = \frac{v_r(t)}{d}\tan(\alpha(t)) \approx \frac{v_r(t)}{d}\alpha(t)$$

če predpostavimo konstantno hitrost zadnjih koles $v_r(t) = V$ in upoštevajoč regulator (4.2) lahko napišemo

$$\dot{\varphi}(t) = \frac{V}{d} K \left(\varphi_{ref}(t) - \varphi(t)\right)$$

ker definira prenosno funkcijo zaprte zanke prvega reda

$$G_z(s) = \frac{\phi(s)}{\phi_{ref}(s)} = \frac{1}{\frac{d}{VK}s + 1}$$

kar pomeni, da se orientacija eksponentno približa referenci (s časovno konstanto $T = \frac{d}{VK}$) brez pogreška v ustaljenem stanju (za konstantno referenco) saj je ojačenje zaprtozančne prenosne funkcije 1. Ker je zasuk krmilnega kolesa omejen $|\alpha| \leq \alpha_{max}$ je omejen tudi krivinski radij $R_{min} = \frac{d}{\tan(\alpha_{max})}$, katerega robot še lahko zvozi. To nadalje pomeni, da je regulator učinkovit pri odpravljanju motenj, dokler je razdalja do cilja večja od R_{min}.

V kolikor želimo hitrejši odziv pri vodenju orientacije, lahko definiramo regulator tako, da bo prenosna funkcija $G_z(s) = \frac{\phi(s)}{\phi_{ref}(s)}$ drugega reda. V tem primeru lahko definiramo, da pogrešek orientacije robota proporcionalno vpliva na kotno hitrost vrtenja krmilnega kolesa

$$\dot{\alpha}(t) = K(\varphi_{ref}(t) - \varphi(t)) \tag{4.3}$$

vendar ima ta regulator nedušen odziv sistema. Če odvajamo linearni model za smer vožnje $\dot{\varphi}(t) = \frac{V}{d}\alpha(t)$ ter vstavimo regulator (4.3) dobimo

$$\ddot{\varphi}(t) = \frac{V}{d} K \left(\varphi_{ref}(t) - \varphi(t)\right)$$

ker definira prenosno funkcijo zaprte zanke drugega reda

$$G_z(s) = \frac{\phi(s)}{\phi_{ref}(s)} = \frac{\frac{VK}{d}}{s^2 + \frac{VK}{d}}$$

z dvema konjugirano kompleksnim poloma na imaginarni osi, sistem je nedušen z lastno frekvenco nihanja $\omega_n = \sqrt{\frac{VK}{d}}$. V sistem moramo vpeljati dušenje, kar lahko storimo, če dodamo še diferencialni značaj k regulatorju (4.3)

$$\dot{\alpha}(t) = K_1(\varphi_{ref}(t) - \varphi(t)) - K_2\dot{\varphi}(t)$$

kar lahko z upoštevanjem približnega linearnega modela za smer vožnje $\dot{\varphi}(t) = \frac{V}{d}\alpha(t)$ zapišemo kot

$$\dot{\alpha}(t) = K_1(\varphi_{ref}(t) - \varphi(t)) - K_2\frac{V}{d}\alpha(t) \tag{4.4}$$

Z dobljenim regulatorjem (4.4) dobimo

$$\ddot{\varphi}(t) = K_1 \frac{V}{d} \left(\varphi_{ref}(t) - \varphi(t)\right) - K_2 \frac{V}{d} \dot{\varphi}(t)$$

in prenosno funkcijo

$$G_z(s) = \frac{\phi(s)}{\phi_{ref}(s)} = \frac{K_1 \frac{V}{d}}{s^2 + K_2 \frac{V}{d} s + K_1 \frac{V}{d}}$$

Iz prenosne funkcije vidimo, da smo dobili dušen odziv sistema z lastno frekvenco $\omega_n = \sqrt{K_1 \frac{V}{d}}$ in dušenjem $\zeta = \frac{K_2}{2} \sqrt{\frac{V}{dK_1}}$ ter zaprtozančnimi poli $s_{1,2} = -\zeta\omega_n \pm j\omega_n \sqrt{1-\zeta^2}$.

4.2.2 Vodenje hitrosti akermanovega pogona

Na hitrost gibanja robota vplivamo z obodno hitrostjo zadnjih koles $v_r(t)$. Hitrost $v_r(t)$ je lahko proporcionalna razdalji do referenčne pozicije $(x_{ref}(t), y_{ref}(t))$

$$v_r(t) = K \sqrt{(x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2} \tag{4.5}$$

Referenčna pozicija robota je lahko konstantna ali pa se spreminja glede na predpisano referenčno trajektorijo. Pri tem se moremo zavedati, da se robot lahko giblje z omejenimi hitrostmi $v_r \leq v_{max}$. Torej je smiselen vhod določen z

$$v_r(t) = \begin{cases} K \sqrt{(x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2} & , \quad v_r < v_{max} \\ v_{max} & , \quad v_r \geq v_{max} \end{cases} \tag{4.6}$$

Nadalje se v praksi hitrost koles $v_r(t)$ ne more v hipu spremeniti, saj neskončen pospešek zahteva neskončno moč pogonskih motorjev. Zato je njeno spreminjanje smiselno omejiti z dovoljenim oziroma dosegljivim pospeškom gibanja robota. Zadovoljiva možnost, da to storimo, je tudi, da se vhodna hitrost eksponentno približuje želeni vhodni hitrosti $v_r(t)$. Dejanska vhodna hitrost sistema $v_r^*(t)$ je filtrirana hitrost $v_r(t)$ s filtrom s časovno konstanto τ_f

$$\tau_f \dot{v}_r^*(t) + v_r^*(t) = v_r(t) \tag{4.7}$$

in prenosno funkcijo

$$G_f(s) = \frac{V_r^*(s)}{V_r(s)} = \frac{1}{\tau_f s + 1}$$

V primeru 4.1 je prikazan primer vodenja robota z akermanovim pogonom k podani referenčni legi.

Primer 4.1. *Trikolesnik s pogonom na zadnji par koles želimo voditi na referenčno pozicijo* $x_{ref} = 5m$ *in* $y_{ref} = 5m$. *Vozilo upravljamo z zasukom prednjega krmilnega kolesa* α *in s hitrostjo vrtenja pogonskih koles* v_r. *Medosna razdalja je* $d = 0.1m$. *Začetna lega vozila je* $[x(0), y(0), \varphi(0)] = [1, 0, -\pi]$. *Pri vodenju upoštevajte fizikalne omejitve sistema* $v_{max} = 0.8$ *in* $\alpha_{max} = \frac{\pi}{4}$.

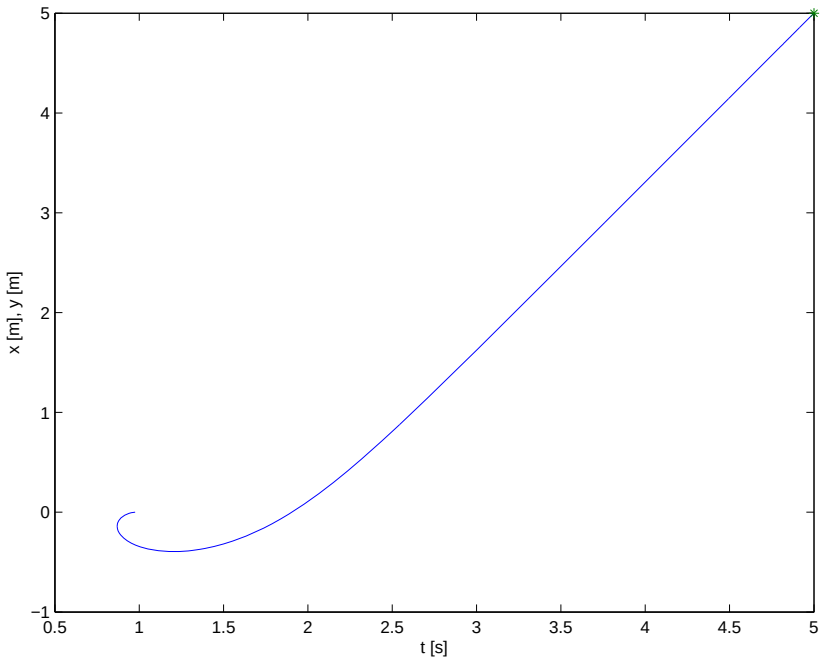
Napišite algoritem vodenja in prikažite rezultate.

Rešitev

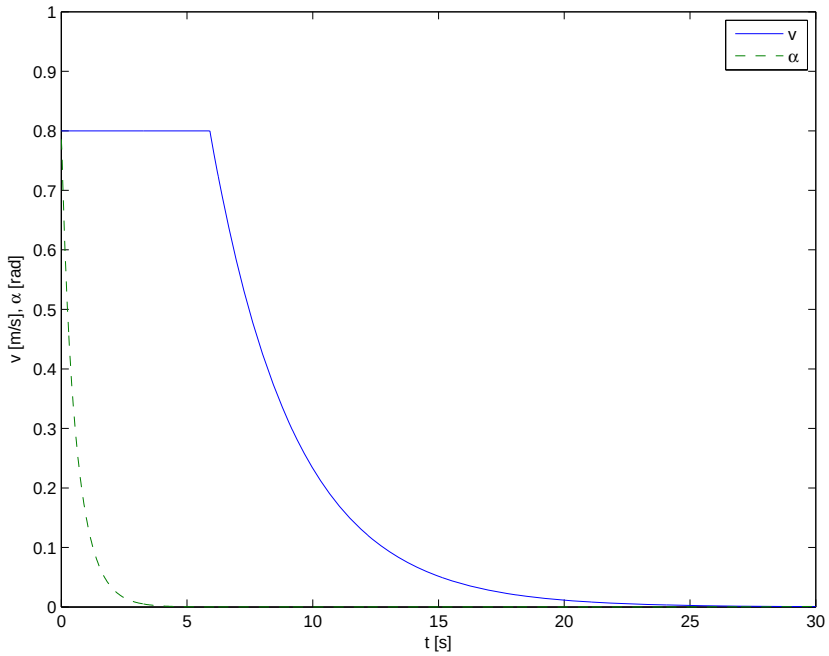
Uporabimo v tem poglavju predstavljen algoritem vodenja. V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.

```
function AckermanControl
close all, clear all %vodenje robota na referenčno pozicijo
Ts=0.03; % čas vzorčenja
d=0.1;   % medosna razdalja

tsim=0:Ts:30; % čas simulacije
xy_ref=[5,5];  % referenčna točka
q=[1 0 -pi];   % zacetno stanje robota

Q=[];Qr=[];U=[];Tvzorcenja=[]; % shranjevanje
t=0; for i=1:length(tsim)
    % referenca
    fi_ref=atan2(xy_ref(2)-q(2),xy_ref(1)-q(1)); % želeni kot
    qr=[xy_ref,fi_ref];

    e=(qr-q)';   % pogresek po x, y in kotu

    % regulator
    v     = 0.3*sqrt(e(1)^2+e(2)^2);
    alpha = 0.2*e(3);

    % fizikalne omejitve vozila
    if abs(alpha)>pi/4, alpha=pi/4*sign(alpha); end
    if abs(v)>0.8, v=0.8*sign(v); end

    Q=[Q;q]; U=[U;[v, alpha]];

    % simulacija kinematike z Euler integracijo
    Fi=q(1,3);
    dq=[ v*cos(Fi);
         v*sin(Fi);
         v/d*tan(alpha)];
    noise=0.00;
    q=q+Ts*dq'+randn(1,3)*noise;


    Tvzorcenja=[Tvzorcenja; t];
    t=t+Ts;
end

 figure plot(Q(:,1),Q(:,2),xy_ref(1),xy_ref(2),'*') xlabel('t
[s]'),ylabel('x [m], y [m]')
 figure,plot(Tvzorcenja,U(:,1),Tvzorcenja,U(:,2),'--') xlabel('t
[s]'),ylabel('v [m/s], \alpha [rad]'),legend('v','\alpha'),
axis([0 tsim(end) 0 1])
```

Rezultati simulacije primera 4.1 so podani v slikah 4.1 in 4.2 kjer vidimo, da vozilo prispe do referenčne lege in se tam ustavi. Na sliki 4.2 vidimo vhodne hitrosti, ki so omejene glede na fizikalne omejitve vozila.



Slika 4.1: Opravljena pot vozila do referenčne točke označene z $*$.



Slika 4.2: Izračunani vhodi.

4.2.3 Vodenje po referenčni trajektoriji

Pri vodenju robota po referenčni trajektoriji mora robot slediti poti, ki je podano parametrično s časom $x_{ref}(t)$, $y_{ref}(t)$, kjer je $t \in [0, T]$.

Pri tem lahko uporabimo pristop opisan v podpoglavjih 4.2.1 in 4.2.2. V trenutku t definiramo referenčno orientacijo robota $\varphi_{ref}(t)$ tako, da se bo le ta zapeljal v referenčno točko $x_{ref}(t)$, $y_{ref}(t)$, kar dosežemo z $\varphi_{ref}(t) = \arctan \frac{y_{ref}-y(t)}{x_{ref}-x(t)}$. Za regulacijo uporabimo regulator podan v (4.2)

$$\alpha(t) = K_\varphi(\varphi_{ref}(t) - \varphi(t)) \tag{4.8}$$

Pri vodenju hitrosti, pa je le-ta proporcionalna pogrešku razdalje do referenčne točke $x_{ref}(t)$, $y_{ref}(t)$. V trenutku t lahko hitrost izračunamo iz relacije (4.5) kot

$$v_r(t) = K_v\sqrt{(x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2} \tag{4.9}$$

Povratnozančnemu vodenju je smiselno dodati še predkrmilni del, katerega lahko določimo iz poznane referenčne trajektorije, kot je prikazano v nadaljevanju.

Vodenje s predkrmiljenjem

S predkrmiljenjem razbremenimo povratnozančno regulacijo in izboljšamo sledenje, kar je predvsem koristno ob prisotnosti pošumljenih signalov iz senzorjev. Z uporabo kinematičnega modela (3.19)

$$\begin{aligned} \dot{x} &= v_r(t)\cos(\varphi(t)) \\ \dot{y} &= v_r(t)\sin(\varphi(t)) \\ \dot{\varphi} &= \tfrac{v_r(t)}{d}\tan(\alpha(t)) \end{aligned} \tag{4.10}$$

kjer je $\dot{\varphi} = \omega(t) = \frac{v_r(t)}{d}\tan(\alpha(t))$, lahko določimo potrebne predkrmilne vhode $v_{ff}(t)$, $\alpha_{ff}(t)$, da se bo robot peljal po predpisani trajektoriji. Translatorna hitrost je določena z

$$v_{ff}(t) = \sqrt{(\dot{x}_{ref}(t))^2 + (\dot{y}_{ref}(t))^2} \tag{4.11}$$

Kotna hitrost $\omega_{ff}(t)$ pa je določena z odvodom orientacije tangente v referenčni točki trajektorije

$$\omega_{ff}(t) = \frac{d}{dt}\left[\arctan\left(\frac{\dot{y}_{ref}(t)}{\dot{x}_{ref}(t)}\right)\right] = \frac{\dot{x}_{ref}(t)\ddot{y}_{ref}(t) - \dot{y}_{ref}(t)\ddot{x}_{ref}(t)}{\dot{x}_{ref}(t)^2 + \dot{y}_{ref}(t)^2} \tag{4.12}$$

od koder (glej 4.10) izrazimo zasuk krmilnega kolesa

$$\alpha_{ff}(t) = \arctan \frac{d\,\omega_{ff}(t)}{v_{ff}(t)} \tag{4.13}$$

Uporaba le izračunanih predkrmilnih vhodov $v_{ff}(t)$, $\alpha_{ff}(t)$ zagotavlja ustrezno sledenje referenčni trajektoriji le v primeru, da ni motenj in začetnega pogreška lege robota. V praksi je to neizvedljivo, zato je smiselna kombinacija povratnozančnega vodenja in krmiljenja, kot to podaja primer 4.2.

Primer 4.2. *Trikolesnik, podan v primeru 4.1 želimo voditi po referenčni trajektoriji* $x_{ref} = 1.1 + 0.7\sin(\frac{2\pi t}{30})$ *in* $y_{ref} = 0.9 + 0.7\sin(\frac{4\pi t}{30})$. *Začetna lega vozila je* $[x(0), y(0), \varphi(0)] = [1.1, 0.8, 0]$. *Napišite algoritem vodenja in prikažite rezultate.*

Rešitev

Uporabimo v tem poglavju predstavljen algoritem vodenja. Algoritem vsebuje vodenje in predkrmiljenje za translatorno hitrost. Predkrmiljenje za kot krmiljenja pa ni potrebno in je zato v algoritmu pomnoženo z ničlo. V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.

```
function AckermanControl
close all, clear all %vodenje robota po trajektoriji

Ts=0.03; % cas vzorčenja
d=0.1;

t=0:Ts:30; % referenčna trajektorija
frek=2*pi*1/30; xr=(1.1+.7*sin(frek*t))';
yr=(0.9+.7*sin(2*frek*t))'; dxr=(frek*.7*cos(frek*t))';
dyr=(2*frek*.7*cos(2*frek*t))'; ddxr=-frek*frek*.7*sin(frek*t)';
ddyr=(-4*frek*frek*.7*sin(2*frek*t))';

Fir=atan2(dyr,dxr); qr=[xr yr Fir];

uR1=sqrt(dxr.^2+dyr.^2);
uR2=(dxr.*ddyr-dyr.*ddxr)./(dxr.^2+dyr.^2); uR=[uR1 uR2];

q=[1.1 0.8 0];          % zacetno stanje
Q=[];Qr=[];E=[];Etrans=[];U=[];Tvzorcenja=[]; EE=[];

tt=0; for i=1:length(t)
    % referenca
    fi_ref=atan2(qr(i,2)-q(2),qr(i,1)-q(1));
    q_ref= [qr(i,1:2), fi_ref];

    e=(q_ref-q)';   % pogresek po x, y in kotu

    % izberemo pravi pogresek kota zaradi šuma in cikličnosti kota
    iii=q_ref(3) - (q(3) + [0;2*pi;-2*pi]);
    [tmp,index]=min(abs(iii));
    e(3)=iii(index);

    % predkrmiljenje
    v_ff     = uR(i,1);
    alpha_ff = atan(uR(i,2)*d/uR(i,1));

    %regulator
    alpha_fb = e(3)*2;  % regulator alfe
    v_fb     = sqrt(e(1)^2+e(2)^2)*0.05*5;

    v_=v_ff*cos(e(3))+v_fb;
    alpha_=alpha_ff*0+alpha_fb;

    % fizikalne omejitve vozila
    if abs(alpha_)>pi/4, alpha_=pi/4*sign(alpha_); end
```

```
    if abs(v_)>0.8, v_=0.8*sign(v_); end

    Q=[Q;q]; Qr=[Qr; qr(i,:)]; U=[U;[v_, alpha_]];

    % simulacija kinematike z Euler integracijo
    Fi=q(1,3);
    dq=[ v_*cos(Fi);     % simulacija gibanja vozila
         v_*sin(Fi);
         v_/d*tan(alpha_)];
    noise=0.000;
    q=q+Ts*dq'+randn(1,3)*noise;

    % popravi kot q(3)
    q(3)=PopraviCiklicnostKota(q(3));

    Tvzorcenja=[Tvzorcenja; tt];
    tt=tt+Ts;
end

figure plot(Q(:,1),Q(:,2),Qr(:,1),Qr(:,2),'--')
xlabel('t[s]'),ylabel('x [m], y [m]') figure
plot(Tvzorcenja,U(:,1),Tvzorcenja,U(:,2),'--'), xlabel('t[s]')
ylabel('v [m/s], \alpha [rad]'),legend('v','\alpha')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = PopraviCiklicnostKota(a)
    a=atan2(sin(a),cos(a));   % prevedemo kot na območje [-pi,pi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati simulacije primera 4.2 so podani v slikah 4.3 in 4.4, kjer vidimo, da vozilo dobro sledi referenčni trajektoriji.

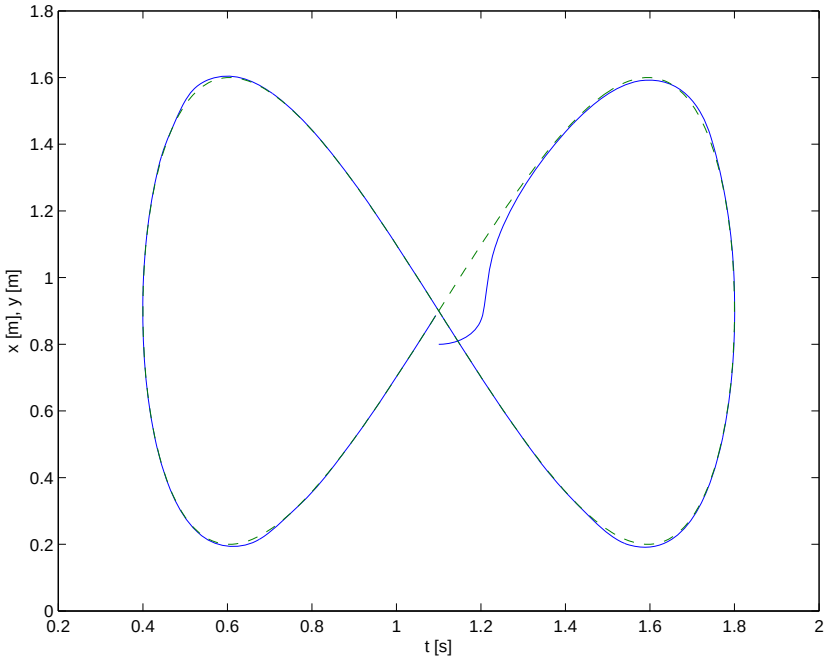
4.3 Vodenje vozila z diferencialnim pogonom

V nadaljevanju bomo obravnavali nekaj različnih pristopov za vodenje kolesnega mobilnega robota z diferencialnim pogonom podanim z modelom (3.2).

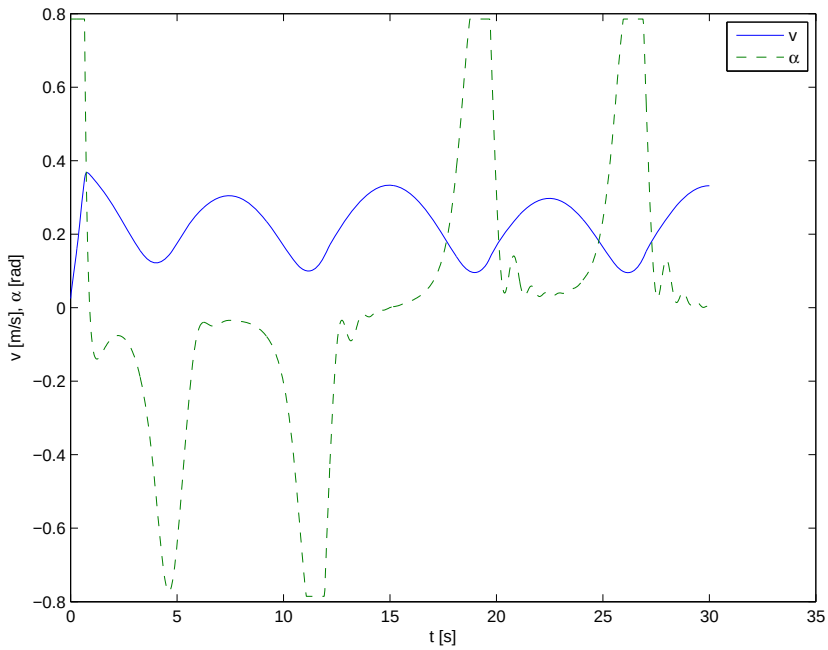
4.3.1 Vodenje v referenčno lego

Pri vodenju mobilnih robotov je izbira primerne oz. optimalne krivulje poti glede na nek kriterij (čas, dolžina, ukrivljenost, poraba energije na poti, ...) zahteven problem. V kolikor imamo zahtevo, da robot sledi vnaprej znani trajektoriji (cesta, črtne označbe v skladiščih,...) se nam z načrtovanjem poti ni potrebno ukvarjati. Če pa je predpisana le ciljna lega robota je potrebno po neki izvedljivi poti prispeti do cilja. To pot lahko eksplicitno določimo in jo med vožnjo tudi sproti prilagajamo ali pa je implicitno podana z izvedbo algoritma vodenja v referenčno lego.

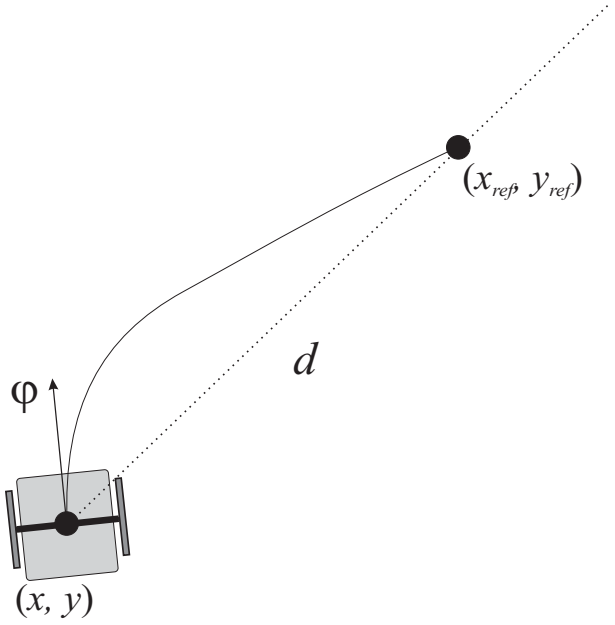
V literaturi obstaja veliko možnih pristopov določitve poti, predvsem je to optimizacija glede na neko kriterijsko funkcijo, uvedba potencialnega oz. gradientnega polja ali vnaprej predpisana oblika poti (po nekem receptu).



Slika 4.3: Sledenje vozila referenčni trajektoriji (črtkano).



Slika 4.4: Izračunani vhodi pri sledenju trajektorije.



Slika 4.5: Vodenje v referenčno pozicijo.

V nadaljevanju bomo prikazali nekaj enostavnih pristopov vodenje v referenčno lego. Predstavljeni primeri vodenja so splošno uporabni in ne le za vozilo z diferencialno kinematiko.

Vodenje v referenčno pozicijo

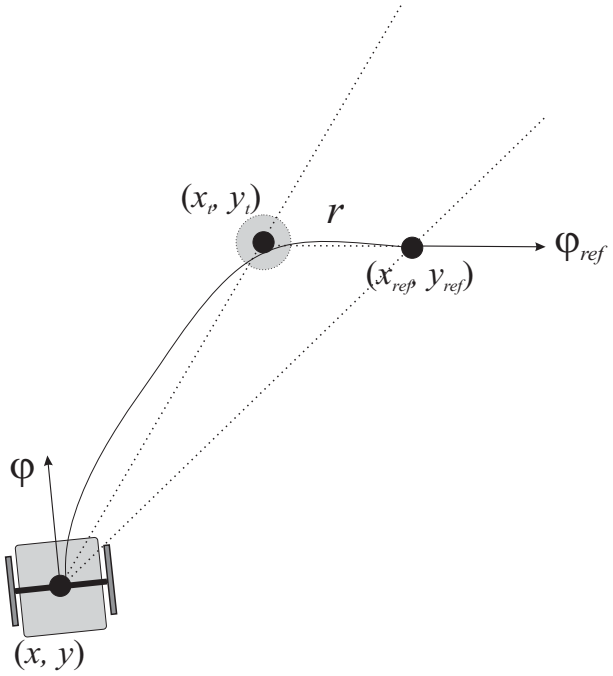
Imamo le zahtevo za referenčno pozicijo robota $(x_{ref}(t), y_{ref}(t))$, orientacija robota v referenčni točki pa ni predpisana. Da se bo robot peljal v smeri proti cilju, moramo regulirati orientacijo robota, tako da bo le ta sledil zveznici, ki povezuje pozicijo robota in ciljno točko. Kotno hitrost lahko definiramo kot

$$\omega(t) = K(\varphi_{ref}(t) - \varphi(t)) \tag{4.14}$$

kjer kot $\varphi_{ref}(t) = \arctan \frac{y_{ref} - y(t)}{x_{ref} - x(t)}$ definira želeno usmeritev robota proti cilju $(x_{ref}(t), y_{ref}(t))$ in K je ojačenje regulatorja. Osnovni princip vodenja prikazuje slika 4.5. Translatorna hitrost gibanja robota pa je lahko ali proporcionalna razdalji do cilja

$$v(t) = K\sqrt{(x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2}$$

ali pa robot pospešuje z maksimalnim dovoljenim pospeškom, pri katerem še ni zdrsov koles, in pred ciljem zavira do želene končne hitrosti.



Slika 4.6: Vodenje v referenčno lego z vpeljavo vmesne točke.

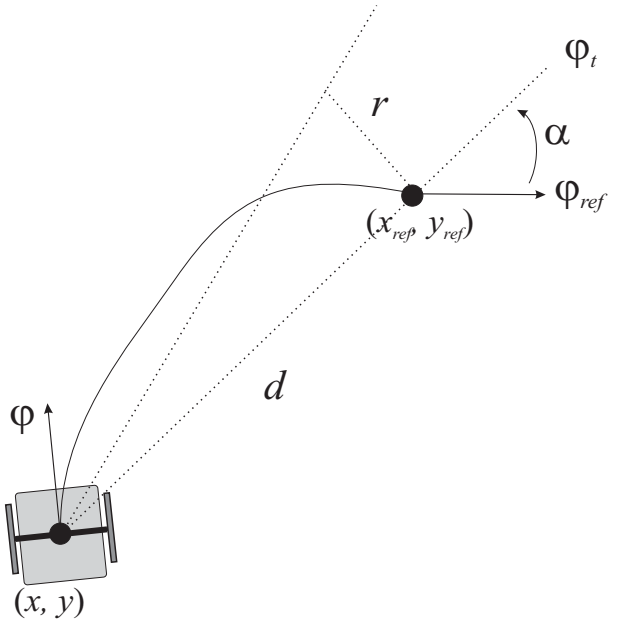
Vodenje v referenčno lego z vpeljavo vmesne točke

Tako vodenje je zelo enostavno izvesti saj uporabimo enostavni regulator (4.14), ki robota pripelje v želeno točko. Ker imamo poleg $(x_{ref}(t), y_{ref}(t))$ tudi zahtevo za referenčno smer φ_{ref} pa vpeljemo vmesno točko (x_t, y_t), postavljeno razdaljo r za referenčno točko v nasprotni smeri referenčne orientacije, kot prikazuje slika 4.6. Vmesno točko določimo z

$$\begin{aligned} x_t &= x_{ref} - r\cos\varphi_{ref} \\ y_t &= y_{ref} - r\sin\varphi_{ref} \end{aligned}$$

V prvem delu robota vodimo po smeri proti tej vmesni točki, ko pa se ta približa vmesni točki $\sqrt{(x-x_t)^2+(y-y_t)^2} < d_{tol}$ pa naredimo preklop in robota vodimo proti referenčni točki. Na ta način dosežemo, da se robot v referenčno točko pripelje z orientacijo približno enako referenčni. Možne so različne variacije algoritma in vpeljava več vmesnih točk za boljše delovanje.

Opisani algoritem je enostaven in se marsikje uporablja. Glede na primer uporabe je potrebno smiselno izbrati razdaljo r ter tolerančno področje d_{tol} za detekcijo bližine testne točke.



Slika 4.7: Vodenje v referenčno lego z vmesno usmeritvijo.

Vodenje v referenčno lego z vpeljavo vmesne usmeritve

Robot se mora pripeljati iz svoje začetne lege v referenčno lego, kjer je definirana pozicija $(x_{ref}(t), y_{ref}(t))$ in orientacija φ_{ref}. Podobno kot v primeru vodenja z vmesno točko je algoritem razdeljen v dva dela. V prvem delu robota vodimo v smeri referenčne pozicije, v drugem delu pa poskrbimo še zato, da se robot v referenčno pozicijo pripelje z referenčno orientacijo. Preklop med tema dvema režima je izveden mehko s pomočjo kotnega pogoja.

Smer robota vodimo z regulatorjem

$$\omega(t) = K\varphi_{err}(t)$$

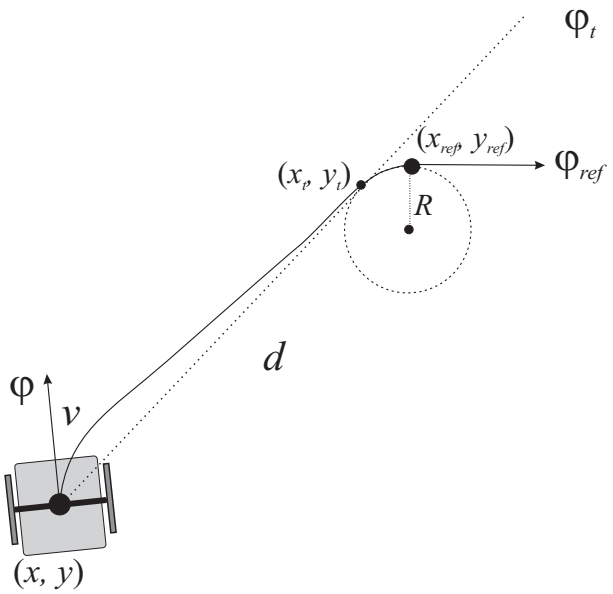
kjer je kotni pogrešek $\varphi_{err}(t)$ glede na sliko 4.7 določen z relacijo [10]

$$\varphi_{err}(t) = \varphi_t(t) - \varphi(t) + \min\left(\alpha(t), \arctan\frac{r}{d}\right)$$

kjer je $\varphi_t(t) = \arctan\frac{y_{ref}-y(t)}{x_{ref}-x(t)}$, $\alpha(t) = \varphi_t(t) - \varphi_{ref}$ in

$$d = \sqrt{(x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2}$$

Parameter r definira smer vožnje v prvem delu algoritma torej vožnjo v smeri, ki je za kot $\arctan\frac{r}{d}$ premaknjena glede na smer $\varphi_t(t)$. V prvem delu je je kotni pogrešek določen z $\varphi_{err}(t) = \varphi_t(t) - \varphi(t) + \arctan\frac{r}{d}$.



Slika 4.8: Vodenje po odsekoma zvezni poti sestavljeni iz premice in kroga.

Ko se robot dovolj približa referenci začne krožiti, dokler se ne poravna z referenčno smerjo $\varphi(t) = \varphi_{ref}$. V drugem delu je kotni pogrešek določen z $\varphi_{err}(t) = \varphi_t(t) - \varphi(t) + \alpha(t)$.

Translatorna hitrost gibanja robota je lahko proporcionalna razdalji do cilja ali definirana na kak drug način (konstantna, pospešeno gibanje,...).

Vodenje po odsekoma zvezni poti (premica in krog)

Osnovna ideja je, da robot najprej sledi premici, ki se v bližini referenčne točke tangentno poveže na krožnico. Krožnica pa je določena tako, da je tangenta v referenčni točki referenčna smer. Sestavljeno pot prikazuje slika 4.8.

Prvi del, ki predstavlja sledenje premici je določen z regulatorjem

$$\omega(t) = K(\varphi_t(t) - \varphi(t))$$

kjer je $\varphi_t(t) = \arctan \frac{y_t - y(t)}{x_t - x(t)}$ in (x_t, y_t) je vmesna točka, kjer se premica poveže na krožnico. Drugi del, kjer pa sledimo krožnici pa z

$$\omega(t) = \frac{v(t)}{R} + K(\varphi_{ref}(t) - \varphi(t))$$

kjer je R radij krožnice in $v(t)$ želena translatorna hitrost vožnje, $\varphi_{ref}(t)$ pa kot tangente na krožnico v točki, kjer se robot nahaja. Prvi del regulatorja za vožnjo po krožnici predstavlja predkrmiljenje, drugi del pa regulacijo.

Za dosego večje robustnosti, se referenčna pot izračuna v vsaki iteraciji, kar zagotovi, da je robot vedno na referenčni premici oziroma na referenčni krožnici. Končna prevožena pot se zato nekoliko razlikuje od poti sestavljen iz premice in kroga. Razlika v gibanju nastane zaradi neujemanja začetnih pogojev (orientacija robota ne ustreza tangenti), šuma in motenj (zdrs koles). Pot je enostavno sestavljena iz premice in kroga zato je določljiva v realnem času. Pot je odsekoma zvezna, kjer sta zvezna dela poti premica in krog, prehod iz premice na krog pa je nezvezen, saj se mora krožna hitrost robota pri tem prehodu hipoma spremeniti iz nič (na premici) na $\frac{v(t)}{R}$ (prekrmiljenje za krožnico), da robot sledi poti brez pogreška. V praksi to ni izvedljivo, zato imamo nekaj sledilnega pogreška na prehodu iz premice na krog.

Tovrstno vodenje je primerno, ko mora robot hitro in točno prispeti v ciljno lego, oblika vmesne poti pa ni pomembna, vendar želimo, da je čim krajša (npr. robotski nogomet). Podoben način vodenja mobilnih robotov je v praksi pogost. Za robota z akermanovim pogonom, ki ima omejen najmanjši radij zavoja, je pokazano [11] [12], da je najkrajša možna pot do cilja (v katerem je definirana tudi orientacija) prav kombinacija premice in kroga. Problem lahko opišemo ločeno: najprej gibanje po premici, pred ciljem pa prehod na krožnico. Radij krožnice je določen z želeno hitrostjo robota v ciljni točki in z dovoljenim kotnim pospeškom oziroma z minimalnim radijem zaradi krmilnega mehanizma.

4.3.2 Vodenje po referenčni trajektoriji

V nadaljevanju bo podanih nekaj pristopov uporabnih pri vodenju mobilnega robota z diferencialnim pogonom s kinematičnim modelom

$$\begin{bmatrix} \dot{x(t)} \\ \dot{y(t)} \\ \dot{\varphi(t)} \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ \sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{4.15}$$

po trajektoriji $x_{ref}(t), y_{ref}(t)$, ki je podana parametrično s časom t.

Predkrmiljenje

Upoštevajoč kinematičen model lahko izračunamo potrebne predkrmilne vhode $v_{ff}(t)$, $\omega_{ff}(t)$, da se bo robot peljal po predpisani trajektoriji (brez povratne zanke). To je seveda izvedljivo le v idealnem primeru, ko model točno opisuje robota, in ko ni motenj in začetnega pogreška lege robota. V praksi to ni neizvedljivo, zato je smiselna kombinacija povratnozančnega vodenja in krmiljenja kot bo podano v nadaljevanju.

Translatorna hitrost je določena z

$$v_{ff}(t) = \sqrt{(\dot{x}_{ref}(t))^2 + (\dot{y}_{ref}(t))^2} \tag{4.16}$$

Kotna hitrost $\omega_{ff}(t)$ pa je določena z odvodom orientacije tangente v referenčni točki trajektorije

$$\omega_{ff}(t) = \frac{d}{dt}\left[\arctan\left(\frac{\dot{y}_{ref}(t)}{\dot{x}_{ref}(t)}\right)\right] = \frac{\dot{x}_{ref}(t)\ddot{y}_{ref}(t) - \dot{y}_{ref}(t)\ddot{x}_{ref}(t)}{\dot{x}_{ref}(t)^2 + \dot{y}_{ref}(t)^2} \tag{4.17}$$

4.3.3 Linearni regulator

V okolici trajektorije lahko ocenimo linearni, časovno spremenljiv model z uporabo linearizacije. Model podaja dinamiko sledilnega pogreška, kot sledi. Sledilni pogrešek definiramo v koordinatnem sistemu robota, tako da transformiramo globalni pogrešek lege robota v lokalne koordinate robota z

$$\begin{bmatrix} e_x(t) \\ e_y(t) \\ e_\varphi(t) \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & \sin(\varphi(t)) & 0 \\ -\sin(\varphi(t)) & \cos(\varphi(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{q}_{ref}(t) - \mathbf{q}(t)) \tag{4.18}$$

kjer je $\mathbf{e}(t) = [e_x(t), e_y(t), e_\varphi(t)]^T$ sledilni pogrešek v koordinatah robota, $\mathbf{q}_{ref}(t) = [x_{ref}(t), y_{ref}, \varphi_{ref}]^T$ je referenčna lega robota v trenutku t (na sliki 4.9 prikazano kot namišljeni referenčni robot) ter $\mathbf{q}(t) = [x, y, \varphi]^T$ je lega robota. Iz kinematike (4.15) in odvoda transformacije sledilnega pogreška (4.18), upoštevajoč, da ima navidezni referenčni robot tudi kinematičen model (4.15), dobimo

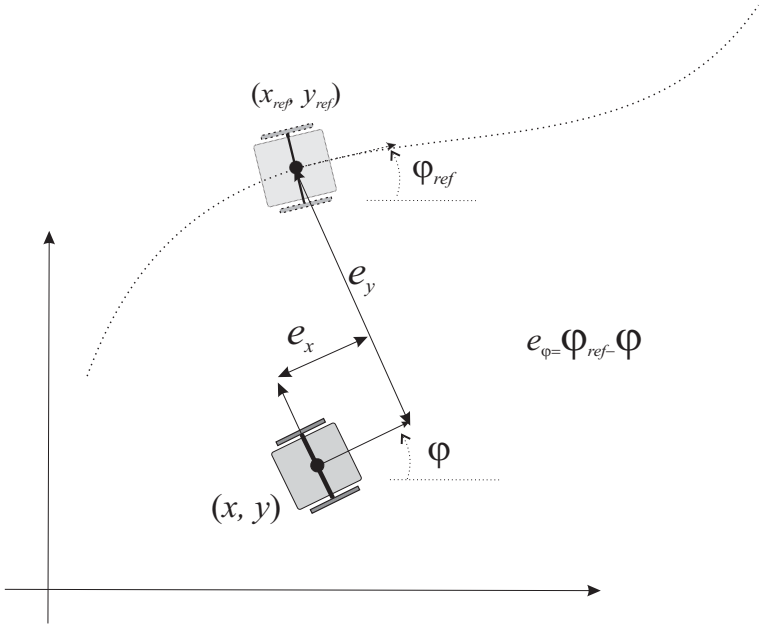
$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\varphi \end{bmatrix} = \begin{bmatrix} \cos e_\varphi & 0 \\ \sin e_\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{ff} \\ \omega_{ff} \end{bmatrix} + \begin{bmatrix} -1 & e_y \\ 0 & -e_x \\ 0 & -1 \end{bmatrix} \mathbf{u} \tag{4.19}$$

kjer je $\mathbf{u} = [v, \omega]^T$ vhodni signal, ki ga določi regulator. Zaradi boljše preglednosti smo izpustili označbo za časovno odvisnost spremenljivk (t). Vhodni signal $\mathbf{u}$ se zelo pogosto ([14], [13]) določi kot vsoto predkrmiljenja in povratnozančnega signala (v_{fb}, ω_{fb}) regulatorja

$$\mathbf{u} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_{ff}\cos e_\varphi + v_{fb} \\ \omega_{ff} + \omega_{fb} \end{bmatrix} \tag{4.20}$$

Če vstavimo (4.20) v (4.19) dobimo

$$\begin{aligned} \dot{e}_x &= \omega_{ff} e_y - v_{fb} + e_y \omega_{fb} \\ \dot{e}_y &= -\omega_{ff} e_x + v_{ff} \sin e_\varphi - e_x \omega_{fb} \\ \dot{e}_\varphi &= -\omega_{fb} \end{aligned} \tag{4.21}$$



Slika 4.9: Sledilni pogrešek v lokalnih koordinatah.

dobljeni nelinearni model lahko linearizirамo v okolici trajektorije ($e_x = e_y = e_\varphi = 0$, $v_{fb} = \omega_{fb} = 0$) in dobimo linearni model sledilnega pogreška

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\varphi \end{bmatrix} = \begin{bmatrix} 0 & \omega_{ff} & 0 \\ -\omega_{ff} & 0 & v_{ff} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\varphi \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \mathbf{u}_{fb} \tag{4.22}$$

ki pa je časovno spremenljiv sistem, saj sta $v_{ff}(t)$ in $\omega_{ff}(t)$ časovno odvisna. Krajše lahko zapišemo $\dot{\mathbf{e}} = \mathbf{A}\mathbf{e} + \mathbf{B}\mathbf{u}_{fb}$ in od tod vodljivostno matriko $\begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} \end{bmatrix}$, ki je polnega ranga le če sta v_{fb} ali ω_{fb} različna od nič. Polna rank vodljivostne matrike je zadosten pogoj za vodljivost sistema le, če sta v_{fb} in ω_{fb} konstantna (vožnja po premici ali po krogu). V tem primeru je možno sistem stabilizirati s statično povratno zanko (konstantna ojačanja regulatorja stanj). V splošnem (v_{fb} in ω_{fb} sta časovno spremenljiva) pa statična povratna zanka ne zagotavlja asimptotično stabilnost ($\mathbf{e}(t) \to 0$, ko $t \to \infty$) sledilnega pogreška, ker je sistem (4.22) spremenljiv.

Definiramo lahko linearni regulator stanj s statično matriko ojačenj

$$\mathbf{u}_{fb} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & k_\varphi \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\varphi \end{bmatrix} \tag{4.23}$$

kjer vidimo, da pogrešek v smeri vožnje e_x vpliva na v_{fb}, pogrešek v ortogonalni smeri vožnje e_y in pogrešek orientacije e_φ pa vplivata na kotno hitrost robota ω_{fb}.

Ojačenja regulatorja (k_x, k_y, k_φ) lahko določimo s poskušanjem ali pa jih določimo tako, da dosežemo želene zaprtozančne pole sistema. Za želeni zaprtozančne pole določimo želeni karakteristični polinom

$$(s + s_1)(s + s_2)(s + s_3) = 0$$

in primerjamo člene pri enakih potencah s z dejanskim karakterističnim polinomom

$$\det(s\mathbf{I} - \mathbf{A} + \mathbf{BK}) = 0$$

Rešitev poiščemo v obliki [13]

$$\begin{aligned} k_x &= k_\varphi = 2\zeta\omega_n \\ k_y &= \frac{\omega_n^2 - \omega_{ff}^2}{|v_{ff}|} \end{aligned} \tag{4.24}$$

kjer mora biti lastna frekvenca ω_n večja kot je maksimalna kotna hitrost robota ($\omega_n \geq \omega_{ffMAX}$). Dušenje pa izberemo v območju $0 < \zeta < 1$. Opazimo, da je regulator singularen, ko $v_{ff} = 0$ zato je v literaturi predlagan regulator

$$\begin{aligned} k_x &= k_\varphi = 2\zeta\sqrt{\omega_{ff}^2 + g v_{ff}^2} \\ k_y &= g|v_{ff}| \end{aligned} \tag{4.25}$$

kjer je $g > 0$ vpeljan kot dodatni parameter.

Primer 4.3. *Vozilo z diferencilanim pogonom s pogonom želimo voditi po referenčni trajektoriji* $x_{ref} = 1.1 + 0.7\sin(\frac{2\pi t}{30})$ *in* $y_{ref} = 0.9 + 0.7\sin(\frac{4\pi t}{30})$. *Čas vzorčenja je* $T_s = 0.033$ *s. Začetna lega vozila je* $[x(0), y(0), \varphi(0)] = [1.1, 0.8, 0]$. *Napišite algoritem vodenja in prikažite rezultate.*

Rešitev

Algoritem vsebuje vodenje in predkrmiljenje. V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.

```
function DifferentialControl
close all, clear all %vodenje robota po trajektoriji
Ts=0.033; % cas vzorčenja
t=0:Ts:30; % referenčna trajektorija
frek=2*pi*1/30; xr=(1.1+.7*sin(frek*t))';
yr=(0.9+.7*sin(2*frek*t))'; dxr=(frek*.7*cos(frek*t))';
dyr=(2*frek*.7*cos(2*frek*t))'; ddxr=-frek*frek*.7*sin(frek*t)';
ddyr=(-4*frek*frek*.7*sin(2*frek*t))';

qr=[xr yr atan2(dyr,dxr)]; % referenčna trajektorija

uR1=sqrt(dxr.^2+dyr.^2);
uR2=(dxr.*ddyr-dyr.*ddxr)./(dxr.^2+dyr.^2);
uR=[uR1 uR2];  % predkrmiljenje
```

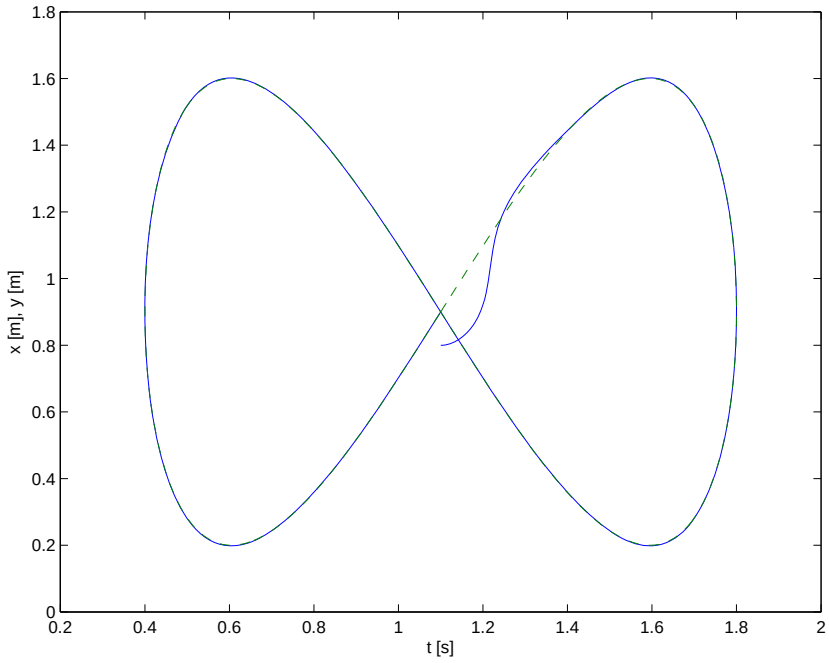
```
q=[1.1 0.8 0];          % zacetno stanje

Q=[];Qr=[];E=[];Etrans=[];U=[];Tvzorcenja=[]; EE=[];

tt=0; for i=1:length(t)

    q_ref=qr(i,:); % trenutna referencna lega

    e=[cos(q(3)) sin(q(3)) 0;...
       -sin(q(3)) cos(q(3)) 0;...
       0           0     1]     * (q_ref-q)';   % pogresek po x, y in kotu
    e(3)=PopraviCiklicnostKota(e(3));  % popravimo pogrešek za kot (cikličnost)

    % predkrmiljenje
    v_ff = uR(i,1)*cos(e(3));
    w_ff = uR(i,2);

    % vodenje
    ex=e(1); ey=e(2); efi=e(3);
    ceta=0.9;
    g=30;
    Kx=2*ceta*sqrt(w_ff^2+g*v_ff^2);
    Kfi=Kx;
    Ky=g;        % lahko tudi konstante Kx=Kfi=3; Ky=30;

    % predkrmiljenje in regulacija (linearna)
    v_ = v_ff + Kx*ex;
    w_ = w_ff + Ky*ey + Kfi*efi ;

    Q=[Q;q]; Qr=[Qr; q_ref]; U=[U;[v_, w_]];

    % simulacija kinematike z Euler integracijo
    Fi=q(1,3);
    dq=[ v_*cos(Fi);     % simulacija gibanja vozila
         v_*sin(Fi);
         w_];
    noise=0.000;
    q=q+Ts*dq'+randn(1,3)*noise;
    q(3)=PopraviCiklicnostKota(q(3));    % popravi kot q(3)

    Tvzorcenja=[Tvzorcenja; tt];
    tt=tt+Ts;
end

figure plot(Q(:,1),Q(:,2),Qr(:,1),Qr(:,2),'--')
xlabel('t[s]'),ylabel('x [m], y [m]') figure
plot(Tvzorcenja,U(:,1),Tvzorcenja,U(:,2),'--')
xlabel('t [s]'),ylabel('v [m/s], \omega [rad/s]'),legend('v','\omega'),% axis([0 Tvzorcenja(end) 0 1])


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = PopraviCiklicnostKota(a)
    a=atan2(sin(a),cos(a));   % prevedemo kot na območje [-pi,pi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati simulacije primera 4.3 so podani v slikah 4.10 in 4.11, kjer vidimo, da vozilo dobro sledi referenčni trajektoriji.

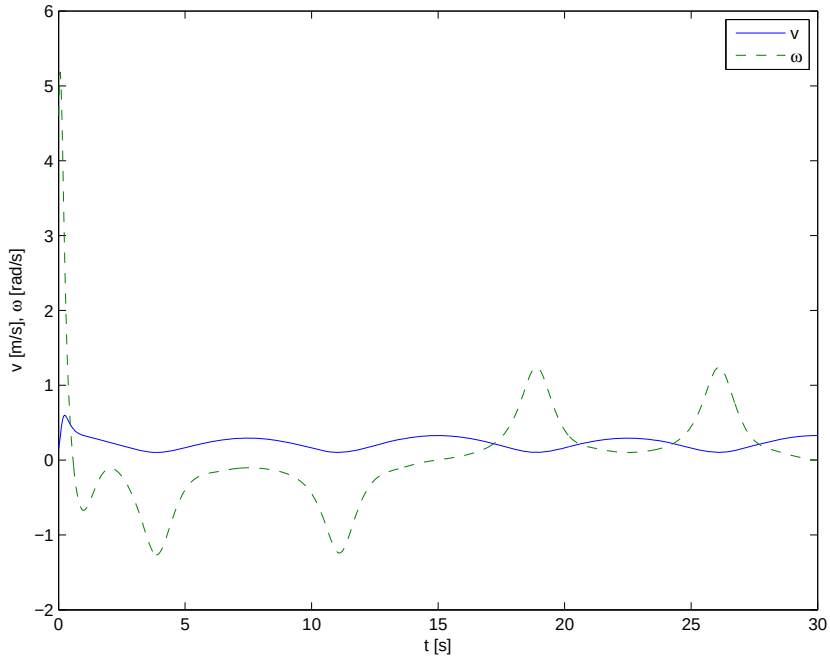
4.3.4 Nelinearni regulator

Nelinearni regulator za sledenje trajektoriji, ki je v literaturi [13], [15] pogosto uporabljen, je

$$\begin{aligned} v_{fb} &= k_x e_x \\ \omega_{fb} &= k_y v_{ff} \frac{\sin e_\varphi}{e_\varphi} e_y + k_\varphi e_\varphi \end{aligned} \tag{4.26}$$



Slika 4.10: Sledenje diferencialnega pogona referenčni trajektoriji (črtkano).



Slika 4.11: Izračunani vhodi pri sledenju diferencialnega pogona trajektoriji.

kjer je $\mathbf{e}(t) = [e_x(t), e_y(t), e_\varphi(t)]^T$ sledilni pogrešek v koordinatah robota, kot v enačbi (4.18). Nadalje je k_y pozitivna konstanta, k_x in k_φ pa sta pozitivni, zvezni in omejeni funkciji. Ena od primernih možnosti za definicijo ojačenj je podana pri izpeljavi linearnega regulatorja v (4.25) kot

$$\begin{aligned} k_x &= k_\varphi = 2\zeta\sqrt{\omega_{ff}^2 + g v_{ff}^2} \\ k_y &= g|v_{ff}| \end{aligned}$$

Upoštevajoč predkrmiljenje (4.16) in (4.17) lahko zapišemo celotni (vsota predkrmiljenja in regulacije) vhodni signal kot

$$\begin{aligned} v &= v_{ff}\cos e_\varphi + k_x e_x \\ \omega &= \omega_{ff} + k_y v_{ff}\frac{\sin e_\varphi}{e_\varphi} e_y + k_\varphi e_\varphi \end{aligned} \tag{4.27}$$

Analiza stabilnosti

V nadaljevanju bomo pokazali, da nelinearni regulator (4.26) zagotavlja globalno asimptotično stabilnost, kar pomeni, da poljubni začetni pogrešek $\mathbf{e}(0)$ konvergira k nič (sledenje referenci brez pogreška), ko čas narašča proti neskončnosti ($\mathbf{e}(0) \to 0$, ko $t \to \infty$).

Z uporabo Lyapunove analize lahko za določen sistem dokažemo asimptotično stabilnost, če obstaja pozitivno definitna funkcija $V(\mathbf{e})$, tako da je njen odvod $\dot{V}(\mathbf{e})$ negativno definiten (za vse $\mathbf{e} \neq 0$ je $\dot{V}(\mathbf{e}) < 0$). Če je $V(\dot{\mathbf{e}})$ le semi-negativno definiten (za določene vrednosti $\mathbf{e} \neq 0$ je $\dot{V}(\mathbf{e}) = 0$) pa je sistem stabilen. Sistem je globalno asimptotično stabilen, če je asimptotično stabilen in če je $V(\mathbf{e})$ zvezno odvedljiva in omejena funkcija. $V(\mathbf{e})$ si lahko predstavljamo kot energijo sistema in če je njen odvod $\dot{V}(\mathbf{e})$ negativen, potem se bo energija sistema s časom zmanjševala proti nič. Torej se bo tudi pogrešek sistema $\mathbf{e}$ zmanjševal proti nič, saj je $V(\mathbf{e})$ pozitivno definitna funkcija pogreška. Pri dokazovanju stabilnosti, moramo torej najti primerno energijsko funkcijo $V(\mathbf{e})$, tako da je njen odvod negativno definiten. Če tako funkcijo uspemo najti (kar je lahko kar zahtevno) smo dokazali stabilnost sistema.

Za sledenje trajektoriji z regulatorjem (4.26) uporabimo Lyapunovo funkcijo

$$V(\mathbf{e}) = \frac{k_y}{2}(e_x^2 + e_y^2) + \frac{1}{2}e_\varphi^2 \tag{4.28}$$

katere časovni odvod je

$$\dot{V}(\mathbf{e}) = k_y e_x \dot{e}_x + k_y e_y \dot{e}_y + e_\varphi \dot{e}_\varphi \tag{4.29}$$

odvode pogreškov izrazimo s pogreški, regulacijskimi in predkrmilnimi signali upoštevajoč relacijo (4.21), dobimo

$$\begin{aligned}\dot{V}(\mathbf{e}) &= k_y e_x \left(\omega_{ff} e_y - v_{fb} + e_y \omega_{fb}\right) + k_y e_y \left(-\omega_{ff} e_x + v_{ff} \sin e_\varphi - e_x \omega_{fb}\right) + e_\varphi \left(-\omega_{fb}\right) \\ &= -k_y e_x v_{fb} + k_y v_{ff} e_y \sin e_\varphi - e_\varphi \omega_{ff}\end{aligned} \tag{4.30}$$

vstavimo nelinerni regulator (4.26) in dobimo končni odvod Lyapunove funkcije

$$\begin{aligned}\dot{V}(\mathbf{e}) &= -k_y e_x (k_x e_x) + k_y v_{ff} e_y \sin e_\varphi - e_\varphi (k_y v_{ff} \tfrac{\sin e_\varphi}{e_\varphi} e_y + k_\varphi e_\varphi) \\ &= -k_x k_y e_x^2 - k_\varphi e_\varphi^2\end{aligned} \tag{4.31}$$

ki je negativno definiten za pogreška e_x in e_φ, ki torej s časoma asimptotično konvergirata k nič. Za pogrešek e_y pa tega ne moremo sklepati, saj ga v (4.31) ni.

Da pokažemo asimptotično stabilnost sistema, določimo integral

$$\int_0^t \dot{V}(t)dt = V(t) - V(0) = -\int_0^t k_x k_y e_x^2 dt - \int_0^t k_\varphi e_\varphi^2 dt \tag{4.32}$$

od koder sklepamo, da je

$$V(0) \geq \int_0^t k_x k_y e_x^2 dt + \int_0^t k_\varphi e_\varphi^2 dt = F(t) \tag{4.33}$$

saj je $V(t)$ pozitivno definitna funkcija. Nadalje velja (Barbalat lema), da če ima funkcija $F(t)$ končno limito, ko gre $t \to \infty$ in če je $\dot{F}(t)$ uniformo zvezna potem velja, da $\dot{F}(t) \to 0$, ko $t \to \infty$. Ker $\dot{F}(t) = -\dot{V}$ konvergira k nič, potem morata tudi e_x in e_φ konvergirati proti nič. Ker torej e_φ ima končno limito (v našem primeru 0) in ker je $\dot{e}_\varphi$ uniformno zvezna funkcija (sledi iz zveznosti referenčne trajektorije) tudi $\dot{e}_\varphi$ konvergira k nič.

Ker $\dot{e}_\varphi$ konvergira proti nič, iz (4.21) sledi, da tudi ω_{fb} konvergira k nič. Nadalje iz definicije regulatorja (4.26)

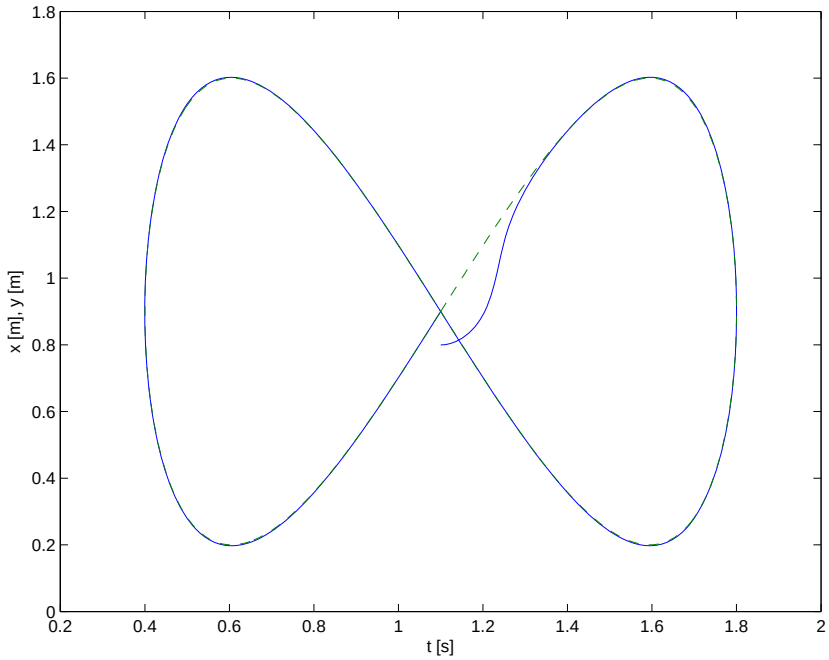
$$\omega_{fb} = k_y v_{ff} \frac{\sin e_\varphi}{e_\varphi} e_y + k_\varphi e_\varphi \tag{4.34}$$

sledi, da če ω_{fb} konvergira k nič in če e_φ konvergira k nič mora k nič konvergirati tudi e_y saj izraz $\frac{sine_\varphi}{e_\varphi}$ konvergira k 1 ($\lim_{t\to\infty} \frac{sine_\varphi}{e_\varphi} = 1$). Dodaten pogoj za konvergenco e_y, ki tudi sledi iz (4.34) je, da $v_{ff} \neq 0$. V kolikor $v_{ff} = 0$ (referenčna trajektorija se ustavi), potem e_x in e_θ še zmeraj konvergirata proti 0, e_y pa konvergira k nekem konstantnem pogrešku. S tem smo pokazali, da je regulator (4.26) zagotavlja globalno asimptotično stabilnost.

Primer 4.4. *Vozilo z diferencilanim pogonom s pogonom želimo voditi po referenčni trajektoriji* $x_{ref} = 1.1 + 0.7\sin(\frac{2\pi t}{30})$ *in* $y_{ref} = 0.9 + 0.7\sin(\frac{4\pi t}{30})$. *Čas vzorčenja je* $T_s = 0.033$ *s. Začetna lega vozila je* $[x(0), y(0), \varphi(0)] = [1.1, 0.8, 0]$. *Napišite algoritem vodenja in prikažite rezultate.*
Rešitev

Algoritem vsebuje vodenje in predkrmiljenje. V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.

```
function DifferentialControlNelin
close all, clear all %vodenje robota po trajektoriji
Ts=0.033; % cas vzorčenja
t=0:Ts:30; % referenčna trajektorija
frek=2*pi*1/30; xr=(1.1+.7*sin(frek*t))';
yr=(0.9+.7*sin(2*frek*t))'; dxr=(frek*.7*cos(frek*t))';
dyr=(2*frek*.7*cos(2*frek*t))'; ddxr=-frek*frek*.7*sin(frek*t)';
ddyr=(-4*frek*frek*.7*sin(2*frek*t))'; qr=[xr yr atan2(dyr,dxr)];

uR1=sqrt(dxr.^2+dyr.^2);
uR2=(dxr.*ddyr-dyr.*ddxr)./(dxr.^2+dyr.^2); uR=[uR1 uR2];

q=[1.1 0.8 0];          % zacetno stanje

Q=[];Qr=[];E=[];Etrans=[];U=[];Tvzorcenja=[]; EE=[];

tt=0; for i=1:length(t)

    q_ref=qr(i,:); % trenutna referencna lega

    e=[cos(q(3)) sin(q(3)) 0;...
       -sin(q(3)) cos(q(3)) 0;...
       0            0      1]     * (q_ref-q)';   % pogresek po x, y in kotu
    e(3)=PopraviCiklicnostKota(e(3)); % popravek zaradi cikličnosti pogreška po kotu

    % predkrmiljenje
    v_ff = uR(i,1)*cos(e(3));
    w_ff = uR(i,2);

    % vodenje
    ex=e(1); ey=e(2); efi=e(3);
    ceta=0.9;
    g=85;

    Kx=2*ceta*sqrt(w_ff^2+g*v_ff^2);
    Kfi=Kx;
    Ky=g;

    % predkrmiljenje in regulacija (nelinearna)
    v_ = v_ff + Kx*ex;
    w_ = w_ff + Ky*v_ff*sinc(efi/pi)*ey+Kfi*efi ;

    Q=[Q;q]; Qr=[Qr; q_ref]; U=[U;[v_, w_]];

    % simulacija kinematike z Euler integracijo
    Fi=q(1,3);
    dq=[ v_*cos(Fi);     % simulacija gibanja vozila
         v_*sin(Fi);
         w_];
    noise=0.000;
    q=q+Ts*dq'+randn(1,3)*noise;

    % popravi kot q(3)
    q(3)=PopraviCiklicnostKota(q(3));
    Tvzorcenja=[Tvzorcenja; tt];
    tt=tt+Ts;
end

 figure, plot(Q(:,1),Q(:,2),Qr(:,1),Qr(:,2),'--'), xlabel('t[s]'),ylabel('x [m], y [m]')
 figure, plot(Tvzorcenja,U(:,1),Tvzorcenja,U(:,2),'--')
 xlabel('t [s]'),ylabel('v [m/s], \omega [rad/s]'),legend('v','\omega')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = PopraviCiklicnostKota(a)
    a=atan2(sin(a),cos(a));   % prevedemo kot na območje [-pi,pi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

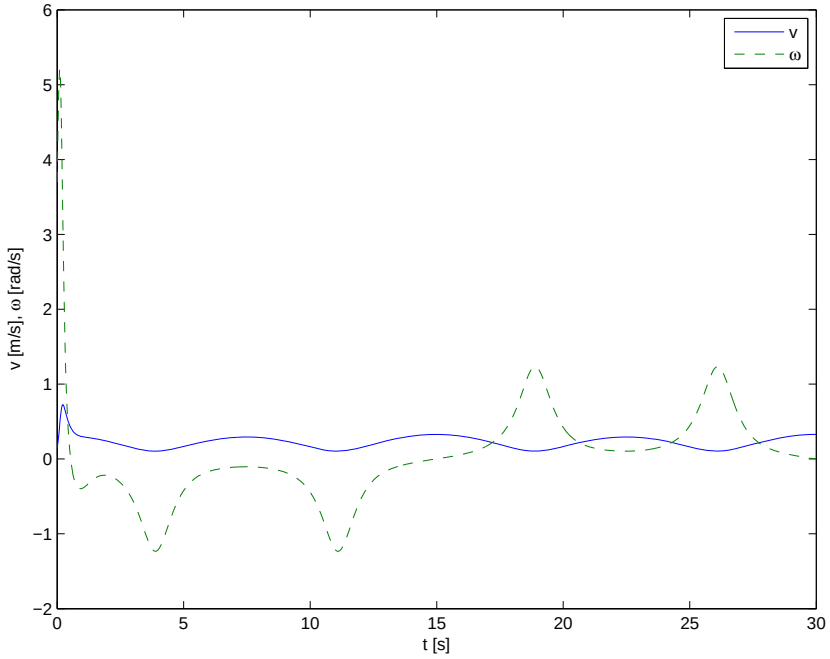


Slika 4.12: Sledenje diferencialnega pogona referenčni trajektoriji (črtkano) z nelinearnim regulatorjem.

Rezultati simulacije primera 4.4 so podani v slikah 4.12 in 4.13, kjer vidimo, da vozilo sledi referenčni trajektoriji podobno kot v primeru linearnega regulatorja. Prednost uporabe nelinearnega regulatorja je, da zagotavlja globalno asimptotično stabilnost sistema.

4.3.5 Povratnozančna linearizacija

Pri postopku povratnozančne linearizacije moramo najprej ugotoviti ali je sistem diferencialno plosk (ang. differentially flat system) [13], [16]. Za diferencialno plosk sistem obstaja set spremenljivk $\mathbf{z}_f$, imenovanih ploski izhodi (ang. flat outputs), če lahko vsa stanja sistema in vhode sistema opišemo



Slika 4.13: Izračunani vhodi pri sledenju diferencialnega pogona trajektoriji z nelinearnim regulatorjem.

s temi ploskimi izhodi in končnim številom njihovih zaporednih odvodov. Veljati mora torej

$$\mathbf{x} = f(\mathbf{z}_f, \dot{z}_f, \ddot{z}_f, \cdots \tfrac{d^p}{dt^p}\mathbf{z}_f)$$
$$\mathbf{u} = g(\mathbf{z}_f, \dot{z}_f, \ddot{z}_f, \cdots \tfrac{d^p}{dt^p}\mathbf{z}_f)$$

kjer so $\mathbf{x}$ stanja sistema, $\mathbf{u}$ vhodi sistema in p zaporedno število odvodov ploskih izhodov.

Sam postopek načrtovanja povratnozančne linearizacije lahko strnemo v sledeče korake:

- Izberemo primerne ploske izhode sistema, njihovo število mora biti enako številu vhodov v sistem.
- Postopno odvajamo ploske izhode, dokler niso odvisni od vseh vhodov sistema.
- Postopek končamo, ko je dobljeni sistem invertabilen iz izbranih izhodov. Z inverzijo izrazimo vhode oz. odvode vhodov sistema. Vhode, ki nastopajo kot odvodi, integriramo, da dobimo vhode v sistem.

Za kolesnega robota z diferencialnim pogonom sta ploska izhoda $x(t)$ in $y(t)$. Njun prvi odvod je glede na kinematiko (4.15) enak

$$\dot{x} = v\cos\varphi$$
$$\dot{y} = v\sin\varphi$$

opazimo, da v odvodih nastopa le translatorna hitrost v, zato ponovno odvajamo

$$\ddot{x} = \dot{v}\cos\varphi - v\sin\varphi\dot{\varphi}$$
$$\ddot{y} = \dot{v}\sin\varphi + v\cos\varphi\dot{\varphi}$$

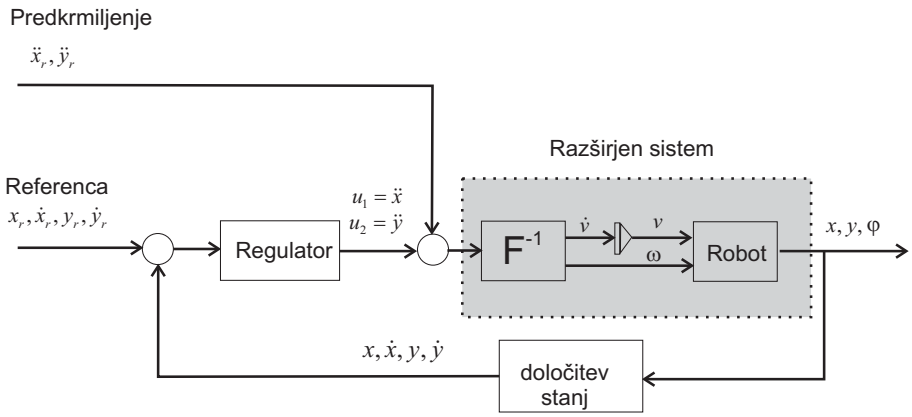
ker je $\omega = \dot{\varphi}$ v dobljenih izrazih nastopata oba vhoda. Nadalje zapišemo

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -v\sin\varphi \\ \sin\varphi & v\cos\varphi \end{bmatrix} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = \mathbf{F} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} \tag{4.35}$$

Z inverzijo izrazimo vhode sistema

$$\begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = \mathbf{F}^{-1} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\frac{\sin\varphi}{v} & \frac{\cos\varphi}{v} \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{4.36}$$

kjer je enačba (4.36) nesingularna za $v \neq 0$. Vhodi v dobljeni povratnozančno lineariziran sistem so $[u_1, u_2]^T = [\ddot{x}, \ddot{y}]^T$, stanja sistema, ki jih reguliramo, so ploski izhodi in njihovi prvi odvodi $\mathbf{z} = [x, y, \dot{x}, \dot{y}]^T$. Imamo štiri stanja, kar



Slika 4.14: Povratnozančno lineariziran sistem, kjer s kompenzatorjem razširimo sistem v linearno obliko.

ustreza vsoti reda originalnega sistema ($\dot{x}$, $\dot{y}$, $\dot{\varphi}$, glej (4.15)) ter enim dodatnim integratorjem ($\dot{v}$) v postopku linearizacije. Dobljeni povratnozančno lineariziran sistem je podan na sliki 4.14.

Razširjen sistem na sliki 4.14 je linearen (model direktne veje brez regulatorja in predkrmiljenja) in brez križnih povezav in ga lahko zapišemo kot

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.37}$$

kjer sta vhoda $u_1 = \ddot{x}$ in $u_2 = \ddot{y}$. Sistem (4.37), ki ga kompaktno zapišemo kot $\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{u}$, je vodljiv saj je rank vodljivostne matrike $[\mathbf{B}, \mathbf{AB}]$ enak številu stanj sistema (polni rank). Nadalje lahko sistem (4.37) zapišemo kot dva univariabilna sistema (zaporedna vezava dveh integratorjev t. i. verižna oblika)

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_1 \tag{4.38}$$

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_2 \tag{4.39}$$

Pri regulaciji na sliki 4.14 primerjamo referenco $\mathbf{z}_r(t) = [x_r, \dot{x}_r, y_r, \dot{y}_r]^T$ in izhode linearnega sistema $\mathbf{z}(t) = [x, \dot{x}, y, \dot{y}]^T$, kjer pa odvodi ponavadi niso direktno merljivi. Odvode lahko ocenimo s pomočjo observatorja ali numeričnega odvajanja. Lahko jih tudi izračunamo, če merimo orientacijo robota φ, $\dot{x} = v\cos\varphi$, $\dot{y} = v\sin\varphi$. Določiti moramo še regulator za vsak

univariabilen sistem (4.38) in (4.39). Določimo lahko regulator PD ali pa regulator stanj za izbrane želene zaprtozančne pole. Primer izvedbe vodenja je podan v primeru 4.5.

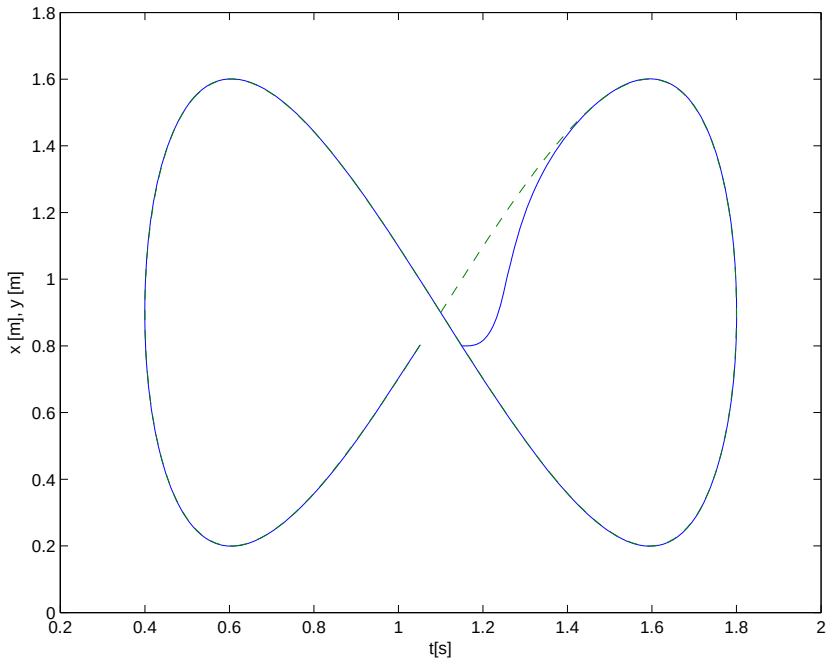
Primer 4.5. *Vozilo z diferencilanim pogonom s pogonom želimo voditi po referenčni trajektoriji* $x_{ref} = 1.1 + 0.7\sin(\frac{2\pi t}{30})$ *in* $y_{ref} = 0.9 + 0.7\sin(\frac{4\pi t}{30})$. *Čas vzorčenja je* $T_s = 0.033$ *s. Začetna lega vozila je* $[x(0), y(0), \varphi(0)] = [1.1, 0.8, 0]$. *Napišite algoritem vodenja s povratnozančno linearizacijo in prikažite rezultate.*

Rešitev *V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.*

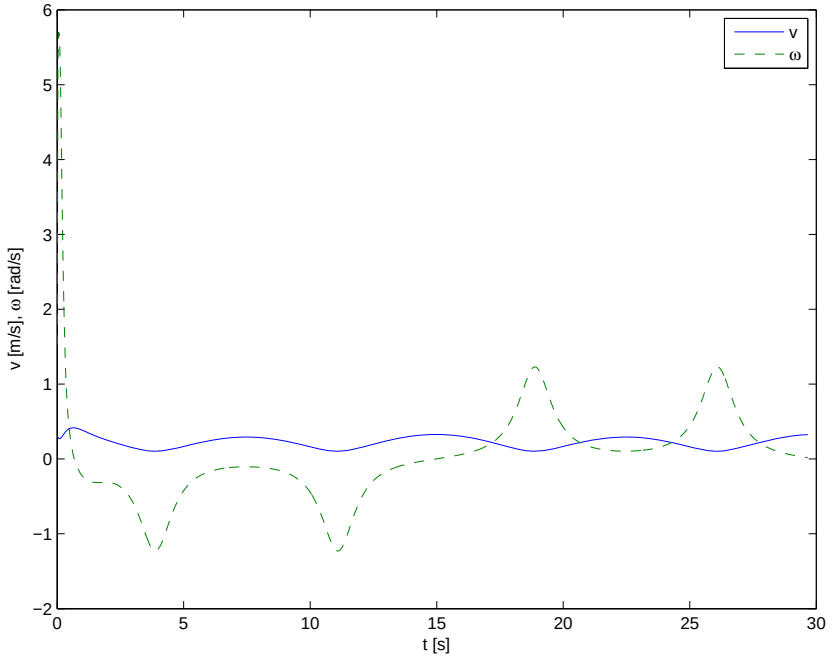
```
close all,clear all
Ts=0.033; t=0:Ts:30; frek=2*pi*1/30;
xr=(1.1+.7*sin(frek*t))'; yr=(0.9+.7*sin(2*frek*t))';
dxr=(frek*.7*cos(frek*t))'; dyr=(2*frek*.7*cos(2*frek*t))';
ddxr=(-frek*frek*.7*sin(frek*t))';
ddyr=(-4*frek*frek*.7*sin(2*frek*t))';
thetar=atan2(dyr,dxr);

qr=[xr yr thetar]; uR=[ddxr ddyr];

q=[xr(1)+.05 , yr(1)-.1 , 0]; % zacetno stanje robota
z1=[q(1) dxr(1)]';            % zacetna stanja lin. sistema [x x']
z2=[q(2) dyr(1)]';            % zacetna stanja lin. sistema [y y']
vv=sqrt(z1(2)^2+z2(2)^2);     % zacetno stanje integratorja za hitrost
v=[ddxr(1);ddyr(1)];          % flat vhod v sistem so pospeški v x in y smeri

% matrike lineariziranega sistema
A1=[0 1; 0 0]; B1=[0;1]; C1=[1 0]; A2=[0 1; 0 0]; B2=[0;1]; C2=[1
0];
% regulator stanj
zeleniPoli=[-2-1*j; -2+1*j];
K1 = acker(A1,B1,zeleniPoli);
K2 = acker(A2,B2,zeleniPoli);
%observator
H1=place(A1',C1',[-15 -16])';
H2=place(A2',C2',[-15 -16])';

Q=[];Qr=[];E=[];Etrans=[];U=[];

for i=1:length(xr)-10   % simulacijska zanka
    Theta=q(1,3);
    % referečna stanja zr
    zr1=[xr(i) dxr(i)]';
    zr2=[yr(i) dyr(i)]';

    % regulacijski pogrešek in vodenje
    ez1=(zr1-z1);
    ez2=(zr2-z2);
    v(1)= K1*ez1;
    v(2)= K2*ez2;
    v= [ddxr(i) ; ddyr(i)] + v;   % dodamo še feedforward

    % določimo vhode v dejanskega robota
    F=[cos(Theta),  -vv*sin(Theta);...
        sin(Theta),   vv*cos(Theta)];
    vvv=F^(-1)*v;     % translatorni pospešek in kotna hitrost
    vv=vv+ Ts*vvv(1);  % integracija translatornega pospeška, da dobis translatorno hitrost
    u=[vv; vvv(2)]; % dejanski vhod v robota (translatorna in kotna hitrost)

    sum=0;
    Brobo=[cos(Theta) 0;sin(Theta) 0;0 1];
    Q=[Q;q]; U=[U;u'];
    dq=Brobo*u;
    q=q+Ts*dq'+rand(1,3)*sum;   % Euler integracija stanj robota

    if (0)   %%%%% stanja z ocenimo s pomočjo observatorja
        yz1=q(1);     % meritev
        yz2=q(2);     % meritev
        yz1_=C1*z1;   % ocene preko modela
        yz2_=C2*z2;   % ocene preko modela

        dz1=A1*z1+B1*v(1)+H1*(yz1-yz1_); % observator
        z1=z1+dz1*Ts;  % Euler integracija
        dz2=A2*z2+B2*v(1)+H2*(yz2-yz2_); % observator
```

```
        z2=z2+dz2*Ts;  % Euler integracija
    else
        % vzamemo kar meritve pozicije in hitrosti namesto observatorja
        %(observator je koristen v primeru, ko je dejanska hitrost robota zaradi zdrsavanja precej različna
        % glede na hitrost podano robotu)
        x=q(1); y=q(2); kot=q(3);
        z1=[x;u(1)*cos(kot)];
        z2=[y;u(1)*sin(kot)];
    end
    Qr=[Qr; qr(i,:)];
end

figure, plot(Q(:,1),Q(:,2),Qr(:,1),Qr(:,2),'--'), xlabel('t[s]'), ylabel('x [m], y [m]')
figure, plot(t(1:length(xr)-10),U(:,1),t(1:length(xr)-10),U(:,2),'--')
xlabel('t [s]'),ylabel('v [m/s], \omega [rad/s]'),legend('v','\omega')
```

Rezultati simulacije primera 4.5 so podani v slikah 4.15 in 4.16, kjer vidimo, da vozilo sledi referenčni trajektoriji podobno kot v primeru linearnega in nelinearnega regulatorja. Pri izvedbi regulacije določimo primerne ploske izhode sistema in določimo kompenzator, s pomočjo katerega sistem linearizziramo. Dobra stran uporabljenega regulatorja je, da se ne rabimo ukvarjati s problemi, ki nastanejo zaradi cikličnosti kota.



Slika 4.15: Sledenje diferencialnega pogona referenčni trajektoriji (črtkano) z regulatorjem na osnovi povratnozančne linearizacije.



Slika 4.16: Izračunani vhodi pri sledenju diferencialnega pogona trajektoriji z regulatorjem na osnovi povratnozančne linearizacije.

Poglavje 5

Planiranje poti

5.1 Uvod

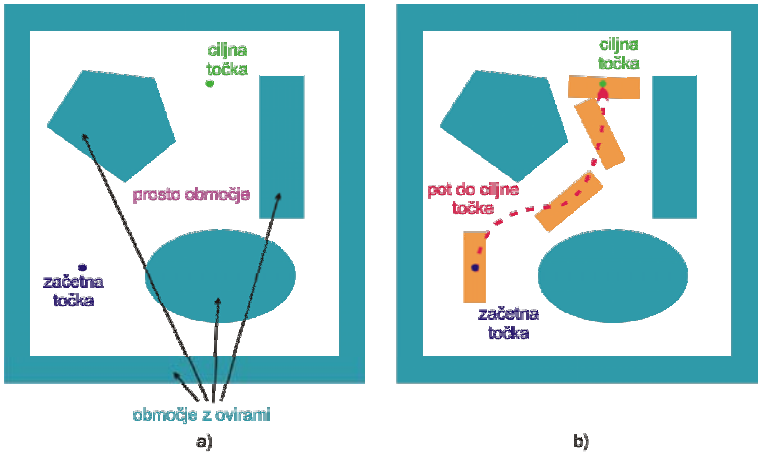
Načrtovanje poti od točke A do točke B, hkratno izogibanje oviram in upoštevanje sprememb v okolju je za človeka enostavna naloga, za stroj pa omejitev, ki jo mora vsaj delno preseči vsak robot, ki želi biti mobilen. Robot s pomočjo senzorjev z določeno negotovostjo zaznava prostor okoli sebe in tako izdeluje ali dopolnjuje svoj zemljevid. Za izračun premikov za dosego nekega cilja se s pomočjo algoritmov odloča in načrtuje potrebna dejanja. Pri tem se upošteva dinamične omejitve robota in kinematične omejitve.

Načrtovanje poti se uporablja v okoljih, ki so v celoti ali delno vnaprej znana, lahko pa tudi v celoti neznana. Kljub številnim raziskavam iz načrtovanja poti znotraj znanega okolja to ostaja osnova tudi za bolj kompleksne primere, ko okolje ni vnaprej znano. V nadaljevanju je prikazanih nekaj uveljavljenih metod načrtovanja poti, ki so povzeta po [33] in [36].

Najprej podajmo definicije nekaj osnovnih pojmov pri načrtovanju poti.

5.1.1 Okolica robota

Okolica robota je sestavljena iz prostega območja in območja z ovirami (slika 5.1). V prostem območju se nahajata začetna in ciljna pozicija, ki navadno nista samo točki v prostoru, ampak definirata vse prostostne stopnje (DOF – degree of freedom) robota, torej pozicijo in orientacijo robota oz. njegovih sklepov. Okolje, ki vsebuje premikajoče se ovire, imenujemo dinamično okolje; okolje, ki se časovno ne spreminja, pa statično okolje. Znano okolje je tisto, pri katerem je pozicija ovir vnaprej znana. V nasprotnem primeru govorimo o neznanem okolju.



Slika 5.1: Okolica robota z ovirami, začetno točko in želeno ciljno točko (a), in eno izmed možnih poti od začetne do ciljne točke (b).

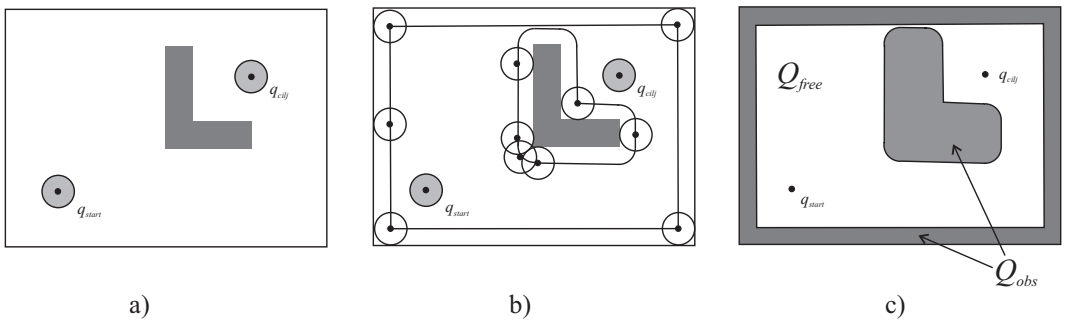
5.1.2 Načrtovanje poti

Načrtovanja poti je proces iskanja zvezne poti, ki bo robota pripeljala od začetne točke do ciljne točke, tako da bo njegova celotna pot ležala v prostem območju, kot je prikazano na sliki 5.1). Pri načrtovanju poti mobilni sistem uporablja zemljevid okolje, ki je shranjen v njegovem spominu.

Stanje je katerakoli točka ali konfiguracija, v kateri se mobilni sistem (robot) lahko znajde. Iz enega stanja v drugo preide s pomočjo akcij. Ustrezno pot opišemo z zaporedjem akcij, ki vodijo robota od začetne točke oz. začetnega stanja skozi potrebna stanja za doseg ciljne točke oz. ciljnega stanja. Katero akcijo bomo v določenem stanju izbrali in katero bo naslednje stanje, je odvisno od izbranega algoritma. Ta se odloča tako, da iz množice stanj, ki lahko sledijo trenutnemu, izbere najprimernejšega glede na cenilko, npr. glede na najmanjšo Evklidsko razdaljo do ciljne točke.

Med začetno in ciljno točko pogosto obstaja več poti. Poleg zahteve, da robot na svoji poti ne trči z ovirami, nam pri izbiri primerne poti lahko pomagajo dodatne zahteve (kriteriji), ki določajo želeno optimalnost:

- dolžina poti naj bo najkrajša,
- ustrezna pot naj bo tista, ki jo robot lahko prevozi v najkrajšem možnem času,
- pot naj bo čim bolj oddaljena od ovir,
- pot naj bo gladka, brez ostrih zavojev,



Slika 5.2: a) krožni robot v okolju z oviro ter začeta in ciljna konfiguracija, b) določitev konfiguracijskega prostora in c) načrtovanje poti, ki je predstavitve okolja v a) prevede v točkovno obravnavo robota v konfiguracijskem prostoru.

- pot naj upošteva robotove omejitve gibanja (primer neholonomičnosti, kjer v danem trenutku niso možne vse smeri vožnje).

5.1.3 Konfiguracija in konfiguracijski prostor

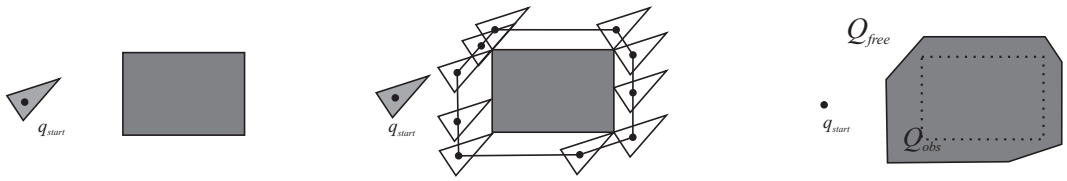
Stanje mobilnega sistema v nekem okolju imenujemo *konfiguracija* in jo opišemo z n podatki, ki predstavljajo vektor stanj $\mathbf{q} = [q_1, \cdots, q_n]^T$, kjer je n število prostostnih stopenj. Stanje $\mathbf{q}$ je točka v n dimenzionalnem prostoru, ki ga imenujemo *konfiguracijski prostor* Q in predstavlja vse možne konfiguracije mobilnega sistema glede na njegovo kinematiko.

Del konfiguracijskega prostora, ki predstavlja ovire O_i (tam se tam robot ne more nahajati) označimo z $Q_{obst} = \bigcup_i O_i$. Torej predstavlja prostor brez ovir Q_{free} celotni konfiguracijski prostor brez konfiguracij Q_{obst}

$$Q_{free} = Q - Q_{obst}$$

torej predstavlja Q_{free} prosti prostor, kjer mobilni sistem lahko načrtuje svoje gibaje.

Predpostavimo, da imamo robota krožne oblike, ki je zmožen le translacij v ravnini (dve prostostni stopnji $\mathbf{q} = [x, y]^T$). Njegovo konfiguracijo lahko obravnavamo točkovno in jo enostavno predstavimo s točko x, y njegovega centra. Konfiguracijski prostor Q pa določimo tako, da robota pomikamo ob oviri, tako da je ves čas v stiku z njo, kar prikazuje 5.2, pri tem pa točka robota, ki določa njegovo pozicijo, opiše mejo med in Q_{free} in Q_{obs}. Na ta način razširimo dimenzije ovir za dimenzijo robota (njegov radij) in nato lahko robota obravnavamo kot točko.



Slika 5.3: a) trikotni robot s točko, ki določa njegovo konfiguracijo $\mathbf{q}$ in pravokotno oviro, b) določitev konfiguracijskega prostora, c) prosti konfiguracijski prostor Q_{free} in prostor ovir Q_obs.

Na sliki 5.3 prikazujemo še primer konfiguracijskega prostora za robota trikotne oblike, ki je zmožen translatornega gibanja v smeri x in y ($\mathbf{q} = [x, y]^T$) in pravokotno oviro.

Če za robota na sliki 5.3 predpostavimo, da je zmožen tudi rotiranja v ravnini $\mathbf{q} = [x, y, \varphi]^T$ je njegov konfiguracijski prostor kompleksnejši in predstavljen v treh dimenzijah. Poenostavljeno pa lahko konfiguracijski prostor določimo tako, da obliki robota očrtamo krog, ki ima središče v točki robota, ki predstavlja njegovo pozicijo. Dobljeni prostor Q_{free} je v tem primeru nekoliko manjši (očrtan krog ima večjo površino kot oblika robota), kot dejanski prosti prostor, vendar nam to poenostavi problem načrtovanja poti.

5.1.4 Matematični zapis oblike in lege ovir v okolju

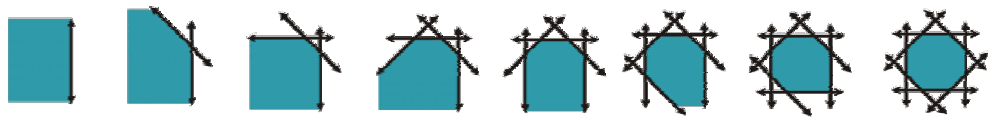
Za izračun konfiguracijskega prostora robota in uporabo nadaljnjih poenostavitev okolja je potreben način zapisa oblike in lege ovir v prostoru. Najpogosteje to storimo na dva načina: z zapisom meje in z zapisom celega telesa.

Predstavitev ovir z zapisom meje

Ovira na tlorisu predstavlja m-strani poligon oz. m-kotnik, katerega mejo zapišemo z nizanjem oglišč v obratni smeri urinega kazalca; luknje v ovirah in okoliško steno tako zapišemo v nasprotni smeri, tj. v smeri urinega kazalca. To velja tako za zapis konveksnih kot tudi nekonveksnih poligonov.

Predstavitev ovir s presekom polravnin

Oviro, m-kotnik, lahko zapišemo kot presek m-tih polravnin, od katerih je vsaka določena s svojo enačbo premice $f(x, y) \leq 0$ oziroma ravnine $f(x, y, z) \leq$



Slika 5.4: Primer zapisa osemkotnika s polravninami.

0 za tridimenzionalne ovire. To velja samo za zapis konveksnih poligonov. Slika 5.4 prikazuje, kako na ta način zapišemo osemkotnik. Nekonveksne like in like z luknjami dobimo s pomočjo operacij nad množicami, npr. unija, presek, razlika množic, itd.

5.2 Predstavitve okolja za namen planiranje poti

Pred samim načrtovanjem poti, moramo okolje predstaviti na poenoten matematičen način, primeren za obdelavo z algoritmi iskanja poti.

5.2.1 Graf prehajanja stanj

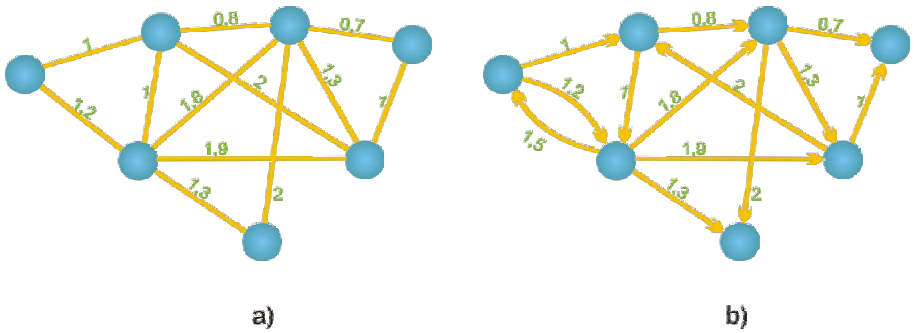
Konfiguracijsko okolje sestoji iz prostega območja, ki predstavlja vse možne konfiguracije mobilnega sistema, in območja z ovirami. Če prosto območje reduciramo in ga predstavimo s podmnožico konfiguracij oz. stanj sistema (so npr. lahko središča nekih področij oziroma celic), ki sestoji iz začetnega stanja, ciljnih stanj in želenih vmesnih stanj ter prehodov med njimi dobimo *graf prehajanja stanj*. Kjer so stanja prikazana s krogi (vozlišča grafa), povezave med njimi pa s črtami, ki predstavljajo akcije, potrebne za prehod med stanji.

Graf je *utežen*, če vsaki povezavi pripišemo neko utež ali ceno, ki je potrebna za izvršitev akcije pri prehodu med stanjema te povezave. V *usmerjenem grafu* pa povezave označimo še s smerjo, če je cena odvisna od smeri prehoda ali pa povezava možna v obeh smereh. Slika 5.5 prikazuje utežen graf in usmerjen utežen graf.

Iskanje poti v predstavitvi problema z grafom je možno z različnimi algoritmi iskanja (A*, Dijkstrov algoritem,...).

5.2.2 Razcep na celice

Okolje lahko razdelimo na celice, ki predstavljajo enostavne geometrijske like. Celice so konveksne, saj mora vsaka daljica, ki povezuje poljubni točki



Slika 5.5: Primer zapisa osemkotnika s polravninami.

v celici ali njene robu v celoti ležati v celici. Po razcepu okolja na celice lahko izvedemo graf prehajanja stanj, kjer je stanje točka v celici (npr. težišče), povezave med stanji (vozlišča grafa) pa so možne le med sosednjimi celicami, ki imata skupen rob.

Natančen razcep na celice

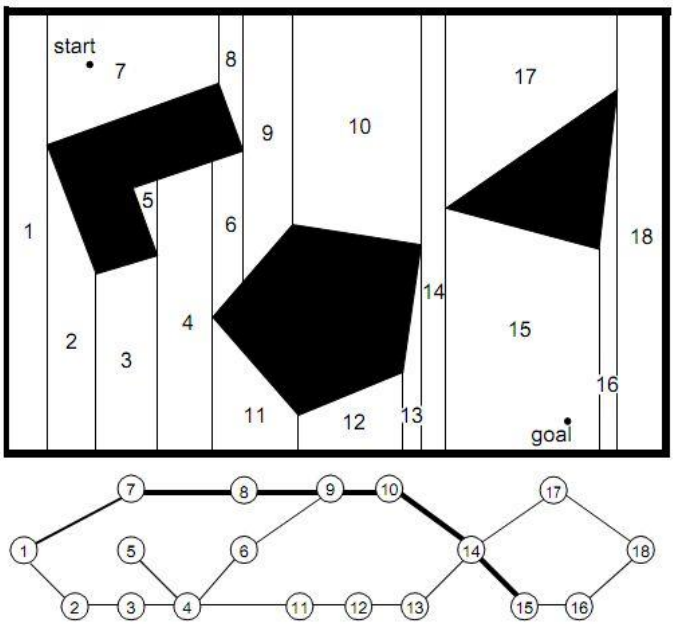
Razcep okolja na celice je natančen, če celice v celoti ležijo ali v prostem območju ali v območju z ovirami. Natančen razcep je brezizguben, saj je unija vseh prostih celic enaka prostem območju oz. prostem konfiguracijskem prostoru Q_{free}.

Primer natančnega razcepa na celice je navpičen razcep na celice, je prikazan na sliki 5.6. Z navidezno navpično črto se pomikamo čez okolico od leve proti desni. Vsakič, ko prečkamo oglišče katerega izmed večkotnikov, ki predstavljajo ovire, ustvarimo navpično črto, tj. mejo med celicama, od oglišča navzgor ali navzdol do ovire. Kompleksnost tega pristopa je močno odvisna od geometrije okolice: če je okolica enostavna, bo število celic in povezav med njimi majhno, z večanjem števila poligonov in oglišč, pa se le-ta veča.

Natančen razcep na celice predstavimo z grafom prehajanja stanj, kjer so vozlišča lahko središča celic, prehodi med središči celic pa gredo skozi točke na središču mej med celicami.

Približen razcep na celice

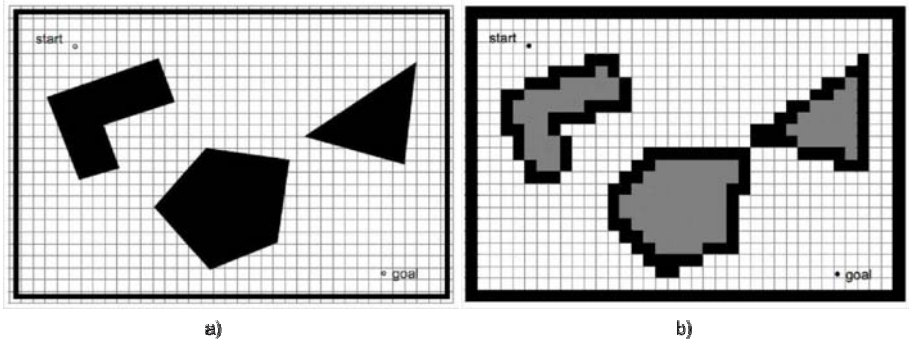
Razcep je približen, ker je možno, da je v posamezni celici tako prosti konfiguracijski prostor kot tudi del ovire. Celice, ki vsebujejo vsaj del ovire označimo kot zasedene, ostale pa so proste. Večinoma se uporablja razcep na celice konstantne velikosti, kjer dobimo *mrežo zasedenosti* (slika 5.7 angleško:



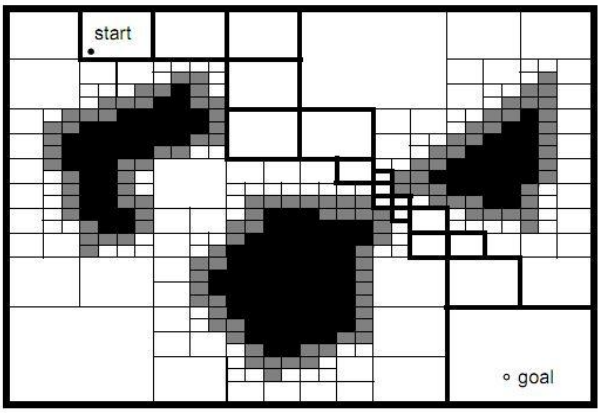
Slika 5.6: Navpičen razcep na celice in pripadajoč graf z vrisano potjo med začetno in ciljno točko.

occupancy grid). Center vsake celice je vozlišče grafa, povezave med celicami pa so možne v štirih ali osmih smereh, odvisno od tega, ali je dovoljeno prehajanje med celicami (vozlišči) v diagonalni smeri. Omenjen pristop je zelo enostaven za uporabo, vendar zaradi konstantne velikosti celice, ki ni odvisna od okolice, lahko pride do izgube informacij: ovire se povečajo in ozki prehodi med njimi se izgubijo. Glavna cena tega pristopa je poraba pomnilnika, ki je za večja okolja velika, ne glede na to, ali so enostavna ali kompleksna.

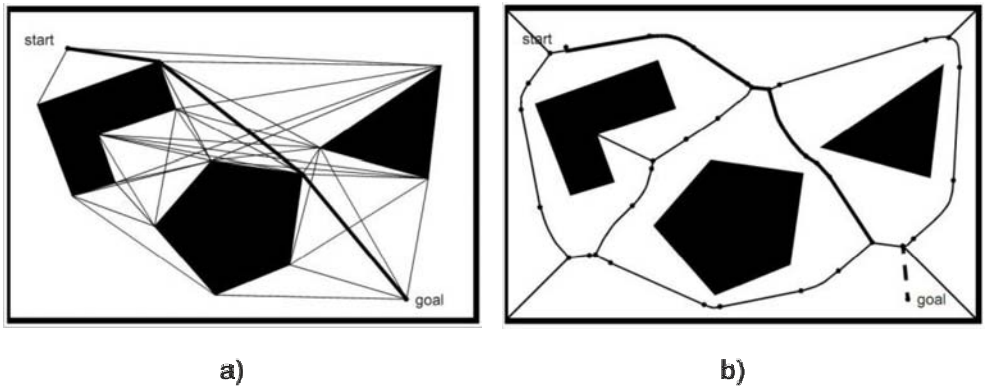
Do manjše izgube informacije pride pri uporabi spremenljive velikosti celic. Na okolje položimo eno celico, ki pokrije celotno okolje. Če je celotna celica v prostem območju ali območju z ovirami, ostane takšna kot je. Če pa je le del nje pokrit z oviro, celico razdelimo na 4 manjše celice. Postopek, v literaturi imenovan *štiriško drevo* (quadtree), ponavljamo do določene resolucije. Dobljeno razdelitev okolja na celice, prikazano na sliki 5.8, lahko prav tako pretvorimo v graf. Približen razcep na celice je enostavnejši od natančnega, vendar lahko zaradi izgube informacij vodi do problema, ko proces načrtovanja poti ne najde rešitve, čeprav le-ta obstaja.



Slika 5.7: Približen razcep na celice: a) na okolico položimo mrežo celic konstantne velikosti, b) celice označimo kot zasedene in proste.



Slika 5.8: Približen razcep na celice s spremenljivo velikostjo celic- štiriško drevo.



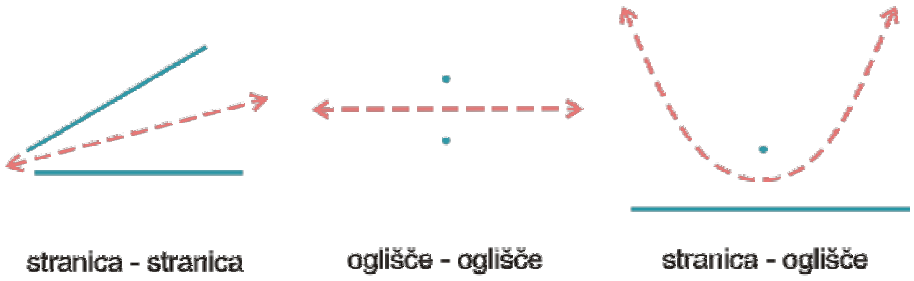
Slika 5.9: Zemljevid cest: a) graf vidljivosti in b) Veronoiev graf.

5.2.3 Zemljevid cest

S pomočjo zemljevida cest (angleško: road map), črt in krivulj ter njihovih stičišč, je prikazana povezljivost prostega območja okolja. Proces načrtovanja poti ima nalogo povezati začetno in ciljno točko s preostalimi cestnimi povezavami in poiskati povezujoče zaporedje cest. Postavitev cest je odvisna od geometrije okolice. Cilj je postaviti minimalno število cest, ki robotu omogočajo dostop do kateregakoli dela prostega območja. V nadaljevanju sta predstavljena dva načina izdelave zemljevida cest.

Graf vidljivosti

Graf vidljivosti (visibility graph) je sestavljen iz vseh povezav med dvema ogliščema območja z ovirami, ki v celoti ležijo v prostem območju. Povedano drugače: za vsako oglišče preverimo, katera oglišča je z njegove pozicije možno videti in ustvarimo vmesne povezave. Pri tem začetno in ciljno točko obravnavamo kot dodatni oglišči. Povezave ustvarimo tudi med dvema sosednjima ogliščema istega poligona. Graf vidljivosti je prikazan na sliki 5.9 a). Pot dobljena s pomočjo grafa vidljivosti je najkrajša možna, saj so ceste speljane kar se da blizu oviram. Da rešimo problem dotika robota in ovire, lahko le-te povečamo vsaj za polmer robota, kot je opisano podpoglavju 5.1.3. Grafi vidljivosti so dokaj enostavni za uporabo, vendar z večanjem števila ovir in njihove kompleksnosti narašča število cestnih povezav in vozlišč, zato se manjša njihova učinkovitost. Grafe vidljivosti pa lahko poenostavimo, tako da odstranimo določene povezave, ki jih je možno nadomestiti z drugo krajšo povezavo.



Slika 5.10: Veronoieve črte.

Veronoiev graf

Veronoiev graf, ki ga prikazuje slika 5.9 b), je sestavljen iz odsekov poti, ki imajo največjo oddaljenost od ovir, kar hkrati pomeni, da je oddaljenost ceste, ki poteka med dvema ovirama, enaka do obeh. V okolici zgrajeni iz poligonov sestavimo zemljevid cest iz treh različnih odsekov, imenovanih Veronoieve črte, glede na postavitev oglišč in stranic, kot to prikazuje slika 5.10. Veronoievo črto, ki je določena z enako oddaljenostjo med

- stranico in stranico predstavlja poltrak ali premica,
- ogliščem in ogliščem predstavlja premica,
- stranico in ogliščem predstavlja parabola.

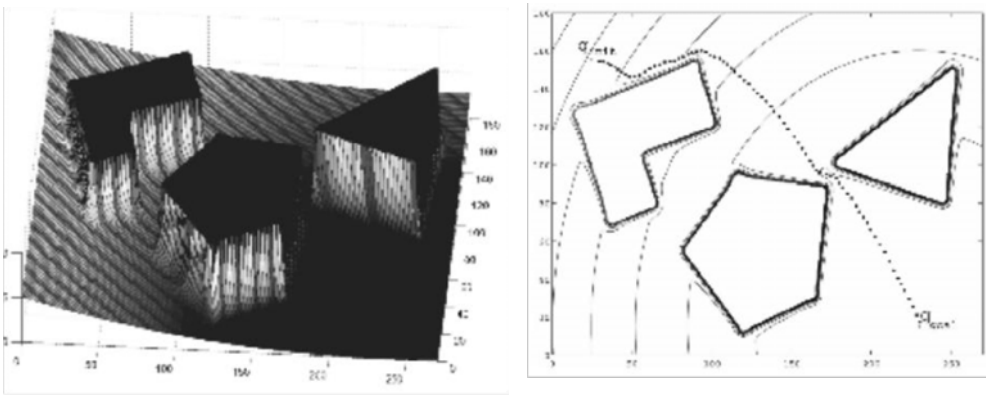
V cestno omrežje povežemo še začetno in ciljno točko, tako dobimo graf, po katerem iščemo rešitev.

Pristop maksimira oddaljenost robota do ovir, vendar je glede dolžine poti daleč od optimalnega. Robot s senzorji oddaljenosti, npr. ultrazvočnimi ali laserskimi, lahko z enostavno regulacijo, kjer se giba na enaki oddaljenosti do vseh okoliških ovir, sledi cestam po Veronoievem grafu, roboti z bližinskimi tipali ali senzorji za kratke razdalje pa imajo pri tem pristopu probleme z lokalizacijo, ki jih pri grafu vidljivosti ne bi imeli.

Iz pristopov, ki brezizgubno upodobijo okolico, je s pomočjo kasneje predstavljenih algoritmov možno v končnem času dobiti tako informacijo, da iskana pot obstaja, kot tudi, da ne obstaja. Pravimo, da so ti pristopi popolni (complete).

5.2.4 Potencialna polja

Okolico predstavimo kot potencialno polje po katerem se robot premika kot žoga, ki se kotali po hribu navzdol do doline. Prikazano je na sliki 5.11.



Slika 5.11: a) Potencialno polje, b) Izohipse in pot od začetne do ciljne točk.

Potencialno polje je vsota privlačnega polja ciljne točke $U_{atr}(\mathbf{q})$, ki je hkrati globalni minimum, in odbojnega polja ovir $U_{rep}(\mathbf{q})$.

$$U(\mathbf{q}) = U_{atr}(\mathbf{q}) + U_{rep}(\mathbf{q}) \tag{5.1}$$

Privlačni potencial $U_{atr}(\mathbf{q})$ je lahko določen tako, da je sorazmeren kvadratu Evklidske razdalje do ciljne točke $D(\mathbf{q}, \mathbf{q}_{goal}) = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2}$, npr.

$$U_{atr}(\mathbf{q}) = k_{atr}\frac{1}{2}D^2(\mathbf{q}, \mathbf{q}_{goal}) \tag{5.2}$$

kjer je k_{atr} konstanta.

Odbojni potencial $U_{rep}(\mathbf{q})$ naj bo zelo močan v bližini ovir, po določeni razdalji od ovir D_0, pa naj nima vpliva na skupno potencialno polje. Lahko je predstavljen kot

$$U_{rep}(\mathbf{q}) = \left\{ \begin{array}{lll} \frac{1}{2}k_{rep}\left(\frac{1}{D(\mathbf{q},\mathbf{q}_{obst})} - \frac{1}{D_0}\right)^2 & ; & D(\mathbf{q}) \leq D_0 \\ 0 & ; & D(\mathbf{q}) > D_0 \end{array} \right\} \tag{5.3}$$

kjer je k_{rep} konstanta, $D(\mathbf{q}, \mathbf{q}_{obst})$ razdalja do ovire.

Robot se pomika v smeri negativnega gradienta potencialnega polja $(-\nabla U(\mathbf{q}))$ in na koncu doseže globalni minimum – ciljno točko.

Primer 5.1. *Določimo negativni gradient polja (5.1).*
Rešitev
Negativni gradient privlačnega polja (5.2) je enak

$$-\nabla U_{atr}(\mathbf{q}) = -k_{atr}\frac{1}{2}[2(x - x_{goal}), 2(y - y_{goal})]^T = k_{atr}(\mathbf{q}_{goal} - \mathbf{q})$$

in kaže v smeri od robota $\mathbf{q}$ *do cilja* $\mathbf{q}_{goal}$*, njegova dolžina pa je proporcionalna razdalji od* $\mathbf{q}$ *do* $\mathbf{q}_{goal}$*.*

Določimo še negativni gradient odbojnega polja (5.3) za primer, ko je $D(\mathbf{q}) \leq D_0$

$$\begin{aligned} -\nabla U_{rep}(\mathbf{q}) &= -k_{rep}\left(\frac{1}{D_{obst}} - \frac{1}{D_0}\right)\frac{-1}{D_{obst}^2}\nabla D_{obst} \\ &= k_{rep}\left(\frac{1}{D_{obst}} - \frac{1}{D_0}\right)\frac{1}{D_{obst}^2}(\mathbf{q} - \mathbf{q}_{obst}) \end{aligned}$$

kjer je $D_{obst} = D(\mathbf{q}, \mathbf{q}_{obst}) = \sqrt{(x - x_{obst})^2 + (y - y_{obst})^2}$*. Vidimo, da smer odbojnega polja kaže stran od ovire, njegova jakost pa se manjša z oddaljenostjo od ovire. Za primer, ko je* $D(\mathbf{q}) > D_0$ *pa je* $-\nabla U_{rep}(\mathbf{q}) = 0$*.*

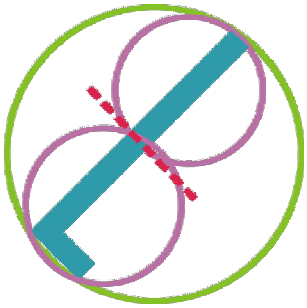
Slabost planiranja poti s potencialnim poljem je, da se lahko robot pri iskanju rešitve ujame v lokalni minimum, pri konkavnih ovirah pa lahko zaradi več enako oddaljenih točk od ovire oscilira.

5.2.5 Metode vzorčenja prostora

Do sedaj predstavljene metode za predstavitev okolja za namen planiranja poti so zahtevale znano eksplicitno predstavitev prostega konfiguracijskega prostora. Z večanjem dimenzije konfiguracijskega prostora postanejo te metode časovno preveč potratne in v teh primerih lahko uporabimo metode vzorčenja prostora.

Pri načrtovanju poti z vzorčenjem (sampling-based path planning) se naključno zajemajo točke iz okolja oz. stanja robota, nato se s pomočjo zaznavanja trka preverja, če le-ta ležijo v prostem območju. Izmed množice takšnih točk in povezav med njimi, ki prav tako v celoti ležijo v prostem območju, poiščemo pot med začetno in ciljno točko.

Z metodami vzorčenja prostora se torej izognemo izračunu prostega konfiguracijskega prostora Q_{free}, ki pri kompleksni postavitvi ovir in visokih prostostnih stopnjah postane zamuden, ter neodvisno od geometrije okolice najdemo rešitev za širok spekter problemov. Prav tako se izognemo velikemu številu celic, ki ga dobimo pri opisu z razcepom na celice, in zamudni implementaciji ter računanju, ki spremljata uporabo natančnega razcepa na celice. Zaradi dodatka naključnega sprehoda (random walk) nekaterim algoritmom, npr. pri RPP (random path planner), ko se iz točke, v kateri smo ujeti, rešimo s pomočjo premika po naključnih vzorcih iz prostega območja, pa močno omilimo problem lokalnih minimumov, ki nas pesti pri uporabi potencialnega polja.



Slika 5.12: Razdelitev večjega lika na dva manjša pri hierarhičnem zaznavanju trka robota L-oblike.

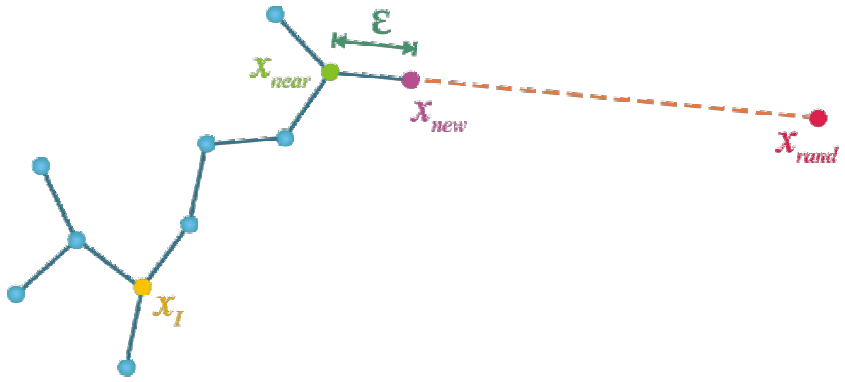
Da bi zaznavanje trka zavzemalo čim manj računskega časa, ga preverjamo samo za ovire, ki so dovolj blizu, da bi lahko bile v trku z robotom, za bolj oddaljene pa ne. Robota in ovire si predstavljamo omejene z enostavnimi liki, tako da kompleksnejše preverjanje trka izvajamo samo, ko se dva lika prekrivata. V tem primeru lahko trk zaznavamo na hierarhičen način. Večji lik, ki obkroža celotnega robota, zamenjamo z dvema manjšima, ki obkrožata vsak svojo polovico robota, kot je to prikazano na sliki 5.12. Nato preverimo, če se kateri izmed likov prekriva z oviro in ga v tem primeru razdelimo na dva manjša ustrezna lika. S tem nadaljujemo dokler ne izključimo ali potrdimo trka oz. dosežemo določene resolucije.

Tovrstne pristope delimo na tiste, ki so primerni za enkratno iskanje poti, in tiste, ki so primerni za večkratno iskanje poti. Pri prvih želimo čim hitreje poiskati pot med eno začetno in eno ciljno točko, zato se osredotočimo na dele okolice, ki obetajo rešitev, in sproti z dodajanjem novih točk ter povezav v drevesno strukturo tudi iščemo rešitev. Pri drugih pa se pred samim načrtovanjem poti izvede enkraten postopek izdelave neusmerjenega grafa oz. zemljevida cest, ki predstavlja povezanost prostega območja in s pomočjo katerega lahko nato rešimo problem načrtovanja poti za več parov začetnih in ciljnih točk. V nadaljevanju sta opisana predstavnika iz obeh skupin.

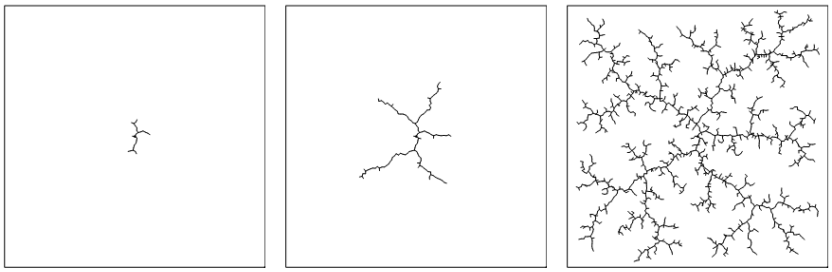
RRT metoda

RRT (rapidly-exploring random tree) je metoda iskanja poti med eno začetno in eno ciljno točko. Metoda sproti dodaja povezave v smeri naključnih točk najbližjim točkam, ki so že v grafu - drevesu.

Začetna konfiguracija x_i v prvem koraku predstavlja drevo. Ob vsakem nadaljnjem koraku se izbere naključno konfiguracijo x_{rand}, kateri se poišče najbližje vozlišče iz grafa x_{near}. V smeri od x_{near} do x_{rand} se na vnaprej



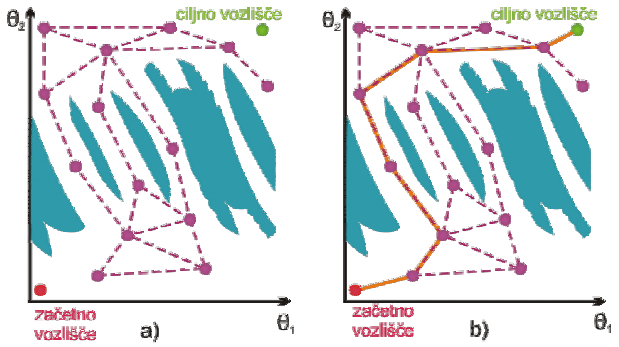
Slika 5.13: Razširitev grafa pri metodi RRT z novo točko x_{new} v smeri naključno izbrane točke x_{rand}.



Slika 5.14: Drevo ustvarjeno z algoritmom RRT se hitro širi po še neraziskanem prostem območju.

določeni razdalji ε izračuna lego morebitnega novega vozlišča x_{new}. Če x_{new} in povezava med njim in x_{near} ležita v prostem območju, potem drevesu dodamo vozlišče x_{new} in povezavo le-tega z x_{near}. Postopek je prikazan na sliki 5.13.

Rešitev najdemo tako, da na določeno število korakov, npr. na vsakih sto, ali z določeno verjetnostjo (10%) namesto novega vzorca izberemo ciljno točko in jo poskusimo povezati z drevesom. Na ta način se izdeluje drevo, ki se hitro širi na največja še neraziskana področja prostega območja, kar je prikazano na sliki 5.14. Pri tej metodi vplivamo samo na parametra dolžina koraka in resolucija ali čas, pri katerem algoritem končamo, zato je obnašanje algoritma konsistentno in njegova analiza lažja.



Slika 5.15: a) faza učenja in b) faza iskanje poti.

PRM metoda

PRM (probabilistic roadmap) je metoda iskanja poti med več začetnimi in več ciljnimi točkami, ki poteka v dveh fazah.

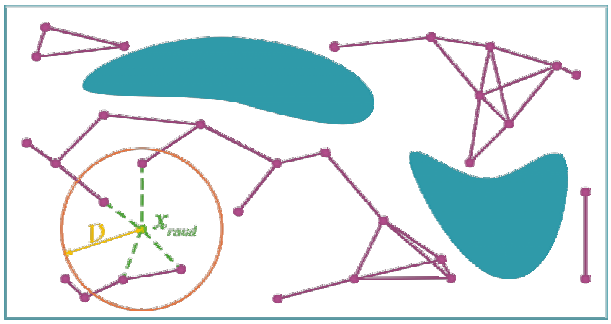
Prva je faza učenja, v kateri se izdela povezan zemljevid cest oz. neusmerjen graf prostega območja, kot ga prikazuje slika 5.15 a), druga pa faza iskanja, v kateri se trenutni par začetne in ciljne točke poveže z grafom in se s pomočjo iskalnih algoritmov poišče pot, kot je možno videti na sliki 5.15 b).

V fazi učenja izdelamo zemljevid cest, ki je na začetku prazna množica, s ponavljanjem v nadaljevanju naštetih korakov. Naključno izbrano konfiguracijo x_{rand}, ki leži v prostem območju, dodamo v zemljevid in določimo vozlišča X_n za razširitev zemljevida. To lahko storimo tako, da izberemo K najbližjih sosednjih vozlišč ali pa vsa sosednja vozlišča, katerih oddaljenost od x_{rand} je manjša od vnaprej določenega parametra D, kot prikazuje slika 5.16 (v prvem oz. prvih korakih sosednjih vozlišč mogoče ne najdemo). V zemljevid nato dodamo vse enostavne povezave od x_{rand} do vozlišč iz X_n, ki v celoti ležijo v prostem območju. S tem nadaljujemo, dokler zemljevid ne vsebuje želenega števila vozlišč N.

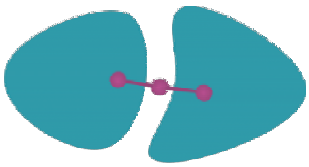
V fazi iskanja začetno in ciljno točko preko prostega območja povežemo s čim bližjima možnima vozliščema iz zemljevida in nato z iskalnim algoritmom poiščemo pot med njima.

Za ti dve fazi ni nujno, da ju izvedemo ločeno, lahko ju večkrat ponavljamo in tako po potrebi, ko število vozlišč ni zadovoljivo za odkritje rešitve, zemljevid bolj dodelamo ter se na ta način iterativno približujemo čim bolj ustrezni predstavitvi prostega območja.

Metoda je zelo učinkovita pri visokih prostostnih stopnjah, vendar ima probleme poiskati povezavo med dvemi območji preko ozkih prehodov, kar



Slika 5.16: Možne povezave do sosednjih vozlišč dodamo v zemljevid.



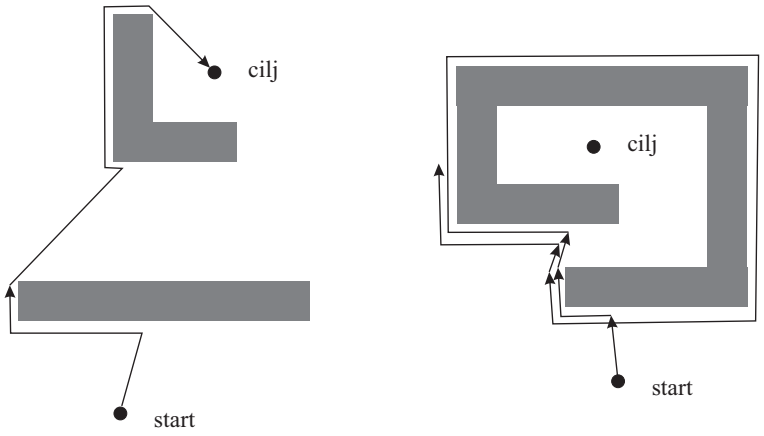
Slika 5.17: Preizkus mostu.

uspešno premagujemo z dodajanjem vozlišč s pomočjo preizkusa mostu (bridge test) v katerem izberemo tri naključne blizuležeče točke, ki ležijo na daljici, kot to prikazuje slika 5.17. Če sta krajni točki v trku z ovirami in srednja ne, potem srednjo točko dodamo kot vozlišče v zemljevid ter jo nato skušamo na enak način kot ostale povezati s sosednjimi vozlišči.

5.3 Enostavni algoritmi planiranja poti - hrošč

Najbolj enostavni algoritmi planiranja poti tipa hrošč (angleško Bug) za planiranje ne potrebujejo zemljevida okolice, zato so primerni, ko zemljevid okolice ni poznan ali pa se okolica stalna spreminja. Ti algoritmi uporabljajo le lokalno informacijo o okolju (senzorji) in na globalno podan cilj, ne potrebujejo pa globalnega znanja v obliki zemljevida okolja. Njihovo delovanje sestoji iz dveh enostavnih obnašanja: gibanje po direktni liniji proti cilju in sledenje obrisu ovire.

Roboti, ki uporabljajo te algoritme se izogibajo oviram in se premikajo prti cilju, imajo majhno porabo spomina, njihova pot pa je lahko daleč od optimalne. V nadaljevanju so predstavljeni trije tovrstni algoritmi.



Slika 5.18: Hroščo algoritem najde pot do cilja na levi sliki in je neuspešen na desni sliki.

5.3.1 Hroščo

Hroščo algoritem deluje v naslednjih dveh korakih:

1. V ravni liniji se premika proti cilju, dokler ne naleti na oviro ali cilj.
2. Če naleti na oviro ob oviri zavije levo (oz. vedno desno, če je tako določeno v algoritmu) in sledi obrisu ovire, dokler ni možno ponovno nadaljevati proti cilju.

Primer delovanja algoritma hroščo je prikazan na sliki 5.18.

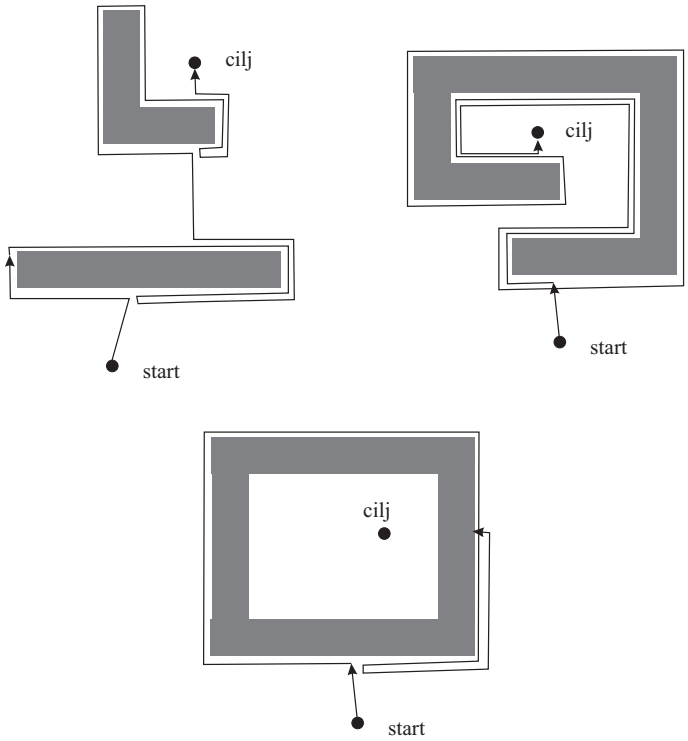
5.3.2 Hroščı

Hroščı uporablja glede na Hroščo nekaj spomina in malo več računanja. Namreč stalno računa Evklidsko razdaljo do cilj in si zapomni najbližjo točko na obodu ovire do cilja. Njegovo delovanje podajata koraka:

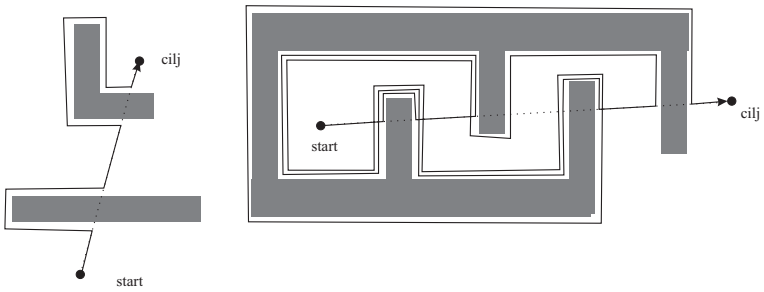
1. V ravni liniji se premika proti cilju, dokler ne naleti na oviro ali cilj.
2. Če naleti na oviro ob oviri zavije levo in sledi celotnemu obrisu ovire ter ves čas meri Evklidsko razdaljo do cilja. Ko prispe do točke, kjer je naletel na oviro, gre po krajši poti ob obodu ovire do točke, ki je bila najbliže cilju.

Primer delovanja algoritma hroščı je prikazan na sliki 5.19.

Dobljena pot ni optimalna. V najslabšem primeru je za $\frac{3}{2}$ obsega vseh ovir do cilja daljša kot samo Evklidska razdalja med začetno in ciljno točko.



Slika 5.19: Hrošč1 algoritem najde pot do cilja v obeh primerih (zgornji sliki). V najslabšem primeru je njegova pot za $\frac{3}{2}$ obsega vseh ovir daljša od Evklidske razdalje med začetno in ciljno točko. Algoritem zna ugotoviti, kdaj cilj ni dosegljiv.



Slika 5.20: Hrošč2 algoritem sledi glavni liniji do cilja. Na isto oviro lahko naleti večkrat (desna slika), zato pride do kroženja. Hrošč2 zna razbrati, kdaj ni mogoče priti do cilja.

Algoritem za vsako oviro, na katero naleti na poti do ciljne točke, najde samo eno točko naleta na oviro in samo eno točko, v kateri zapusti obod ovire, tako na nobeno oviro ne naleti več kot enkrat in zaradi tega nikoli ne ustvari ciklov med istimi ovirami. Ko algoritem naleti na isto oviro več kot enkrat, je to znak, da je ali začetna ali ciljna točka ujeta znotraj ovire, kot je prikazano na spodnjem primeru slike 5.19. Takrat se algoritem konča.

5.3.3 Hrošč2

Algoritem hrošč2 se vedno poskuša premikati po glavni liniji, tj. ravni liniji, ki povezuje začetno in ciljno točko. Deluje s ponavljanjem naslednjih korakov

1. Robot naj se premika po glavni liniji, dokler ne naleti na oviro ali ciljno točko. V zadnjem primeru se iskanje zaključi.

2. Robot sledi obodu ovire toliko časa, da doseže glavno linijo, kjer je Evklidska razdalja do cilja manjša kot Evklidska razdalja točke, kjer je trenutno (zadnjič) naletel na oviro.

Čeprav deluje hrošč2 veliko bolj učinkovit kot hrošč1, zaradi dejstva, da ni več zagotovila, da na določeno oviro naleti največ enkrat, lahko pri določenih postavitvah in oblikah ovir po prostoru dolgo časa po nepotrebnem kroži, preden prispe do ciljne točke, kot je prikazano na sliki 5.20, in s tem izniči vse navidezne prednosti. Algoritem lahko razbere, da ciljne točke ni možno doseči, če večkrat v isti točki naleti na isto oviro.

Iz primerjave algoritmov hrošč1 in hrošč2 lahko zaključimo sledeče:

- hrošč1 je bolj temeljit algoritem iskanja, saj preišče vse možnosti pred izvršitvijo naslednjega koraka,

- hrošč2 je požrešen algoritem, saj vzame prvo opcijo, ki izgleda bolje,
- v večini primerov je hrošč2 bolj učinkovit kot hrošč1, toda
- hrošč1 ima bolj predvidljivo delovanje.

5.4 Metode iskanja poti v grafu

Ko imamo okolje z ovirami ustrezno predstavljeno z grafom (npr. prostor stanj, razcep na celice, zemljevid cest) lahko uporabimo enega izmed algoritmov, ki poišče pot od začetnega do ciljnega stanja.

V nadaljevanju bo podano nekaj algoritmov iskanja poti v grafu. V splošnem iskanje poteka tako, da za začetno vozlišče preverimo, če je hkrati tudi ciljno vozlišče. Ponavadi temu ni tako, zato razširimo iskanje na vozlišča, ki sledijo sedanjemu. Odločimo se za enega od njih, če to vozlišče ni ciljno, raziskujemo naslednja vozlišča, ki sledijo temu novemu vozlišču. Tako nadaljujemo, dokler ne najdemo rešitve ali dokler ne preiščemo celotnega grafa.

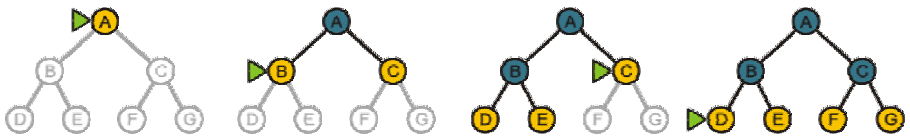
Pri iskanju v grafu vodimo sezname vozlišč, ki smo jih med iskanjem že obiskali, s čimer preprečimo, da bi isto vozlišče obiskali večkrat. Vozlišča iz katerih še lahko nadaljujemo iskanje hranimo v seznam odprtih (open list) ali živih (alive nodes) vozlišč. Vozlišča, ki ali nimajo naslednikov ali pa smo jih že vse preverili, pa shranimo v seznam zaprtih (closed list) ali mrtvih vozlišč (dead notes).

Od strategije algoritma je odvisno po kakšnem zaporedju izbiramo vozlišča za razširitev. Seznam odprtih vozlišč razvrstimo glede na določen kriterij in ob izbiranju naslednjega vozlišča za razširitev iskanja vzamemo prvo iz seznama, torej tisto ki najbolj ustreza razvrščevalnemu kriteriju.

Na začetku je v seznamu odprtih vozlišč Q samo začetno vozlišče. Ko izračunamo vozlišča, ki sledijo začetnemu vozlišču, so le ta vozlišča v seznamu odprtih. Ko iskanje razširimo s prvim izmed teh vozlišč in izračunamo njegove naslednike, so v seznamu preostali nasledniki, razen prvega že raziskanega vozlišča, in pravkar izračunani nasledniki prej izbranega vozlišča. Postopek se lepo vidi na sliki 5.21, kjer so odprta vozlišča prikazana z rumenim krogom in kjer se lepo vidi, zakaj se ta vozlišča med iskanjem v drevesni strukturi imenujejo listi.

Ločimo *neinfomirana* in *informirana* algoritme iskanja po grafu. Pri neinformiranem ali slepem iskanju o stanjih oz. vozliščih ne posedujejo drugih informacij kot tistih, ki so podane z definicijo problema. Graf pregledujejo sistematično in ne razlikujejo bolj obetavnih od manj obetavnih vozlišč.

Pri informiranem ali hevrističnem iskanju je zaradi dodatnih informacij o vozliščih zmožno razlikovati med bolj in manj obetajočimi vozlišči, zaradi



Slika 5.21: Iskanje v širino: Najprej raziščemo najbližja vozlišča.

česar lahko algoritem na učinkovitejši način najde rešitev.

5.4.1 Iskanje v širino

Iskanje v širino (breadth-first search) je algoritem neinformiranega iskanja. Najprej raziščemo najbolj plitva vozlišča, torej vozlišča, ki so bliže začetnemu vozlišču. Vsa vozlišča, do katerih lahko dostopamo v k korakih obiščemo prej kot katerokoli vozlišče, do katerega pridemo v $k+1$ korakih, kar je prikazano na sliki 5.21.

V seznam odprtih vozlišč Q razvrščamo po metodi prvi noter – prvi ven (FIFO – first in - first out): na novo odprta vozlišča dodajamo na konec vrste Q, vozlišča za razširjanje iskanja pa jemljemo z začetka vrste.

Algoritem je popoln, saj pri končnem faktorju razvejitve najde rešitev, če le-ta obstaja, če je možnih več, pa najde tisto, ki je najmanj korakov oddaljena od začetnega vozlišča. To ne pomeni, da je najdena rešitev hkrati tudi optimalna, saj ni nujno, da imajo vsi prehodi med vozlišči enako ceno.

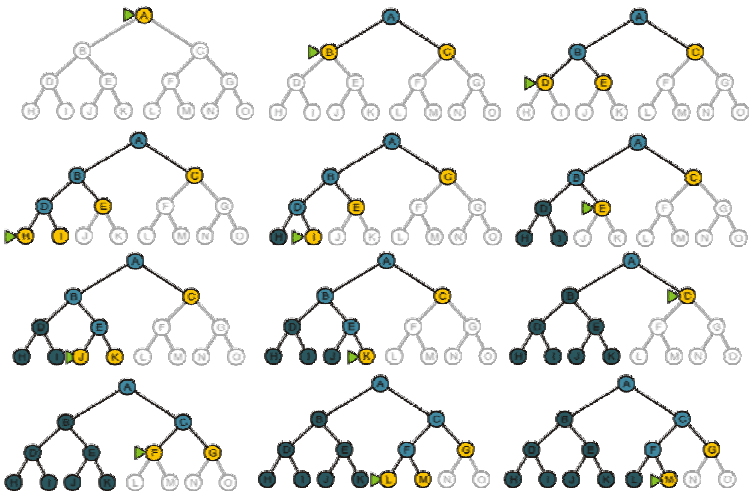
Poraba spomina in računski čas sta pri tej metodi velika, saj z razvejanostjo drevesa eksponentno naraščata.

5.4.2 Iskanje v globino

Iskanje v globino (depth-first search) je algoritem neinformiranega iskanja. Tu iskanje širimo v globino. Vozlišče, ki je najbolj oddaljeno od začetnega, razširimo in iskanje na ta način nadaljujemo v globino, dokler neko vozlišče nima nobenih naslednikov več. Takrat iskanje nadaljujemo z naslednjim najglobljim vozliščem, katerega nasledniki še niso bili vsi raziskani, kot je to prikazano na sliki 5.22

To storimo tako, da seznam odprtih vozlišč Q obravnavamo kot sklad (stack), torej ga razvrščamo po metodi zadnji noter – prvi ven (LIFO – last in - first out): na novo odprta vozlišča dodajamo na začetek vrste Q, od koder tudi jemljemo vozlišča za razširjanje iskanja.

Algoritem iskanja v globino ni popoln, saj bi v primeru neomejene globine neskončno časa raziskoval samo eno vejo grafa. Iskanje lahko omejimo le do



Slika 5.22: Iskanje v globino ima majhno porabo spomina, saj hranimo le liste (rumena) in na poti razširjena vozlišča (svetlo modra).

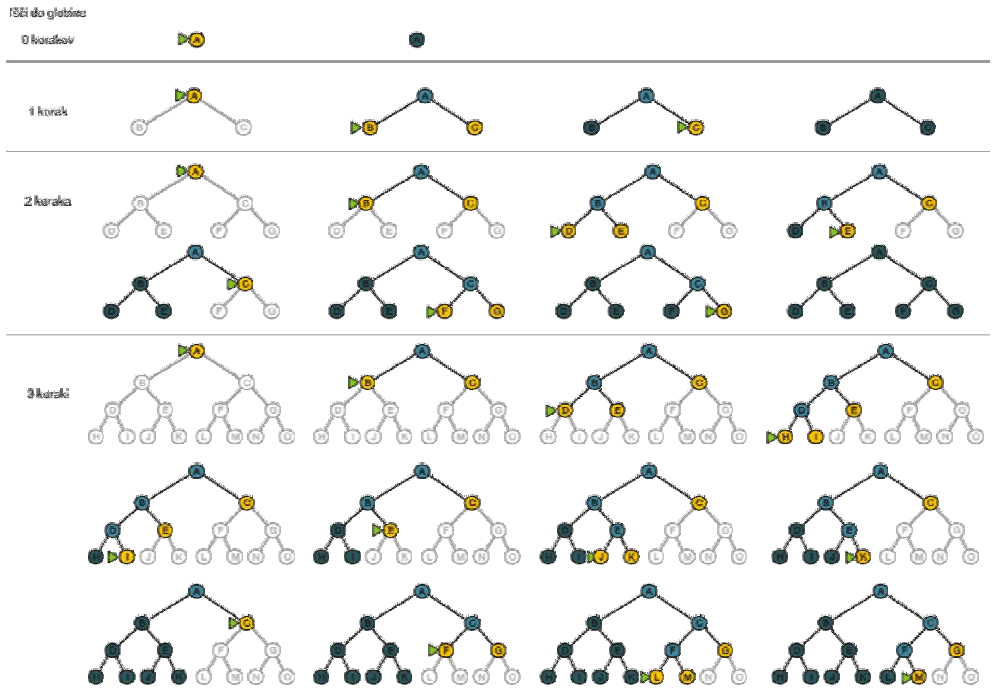
določene globine. Prav tako algoritem ni optimalen, saj najdena pot ni nujno hkrati tudi najkrajša.

Metoda ima majhno porabo spomina, saj mora hraniti samo pot od začetnega vozlišča do trenutnega vozlišča in vmesna vozlišča, s katerimi še nismo nadaljevali iskanja. Ko neko vozlišče in vse njegove naslednike raziščemo, lahko to vozlišče prenehamo hraniti v spominu.

5.4.3 Iterativno poglabljanje iskanja v globino

Algoritem združi prednosti iskanja v širino in iskanja v globino. Pri iterativnem poglabljanju iskanja v globino (iterative deepening depth-first search) po korakih večamo globino, do katere raziskujemo z iskanjem v globino, dokler ne najdemo ciljnega vozlišča: najprej izvedemo iskanje v globino za vozlišča oddaljena nič korakov od začetnega, nato v primeru, da ne najdemo ciljnega vozlišča, iskanje v globino izvedemo za vozlišča, oddaljena en korak od začetnega, itd.

Pri majhni porabi spomina je popoln, saj najde rešitev, če le-ta obstaja, in optimalen, saj pri enakih oz. nepadajočih cenah prehodov v odvisnosti od globine vozlišča najde najkrajšo pot. Če imajo vsa vozlišča približno enak faktor razvejitve tudi večkratno računanje stanj ni pretirano potratno, saj je večina vozlišč v dnu drevesa.



Slika 5.23: Prikaz iterativnega poglabljanja iskanja v globino do globine treh korakov in ciljnega vozlišča M.

5.4.4 Dijkstrov algoritem

Dijkstrov algoritem (neinformiranega) je bil ustvarjen za iskanje najkrajše poti od enega začetnega vozlišča v grafu do vseh preostalih vozlišč, zato je v primeru iskanja poti med eno začetno in eno ciljno točko zaradi izračunavanja vseh nepotrebnih poti neučinkovit in ga spremenimo tako, da se konča, ko izračuna najkrajšo želeno pot.

Algoritem najde najkrajšo pot od začetnega vozlišča do ciljnega vozlišča, saj temelji na računanju cene poti od začetnega do trenutnega vozlišča, ki jo imenujmo cena-do-sem.

Ceno poti do trenutno obravnavanega vozlišča izračunamo kot vsoto cene celotne poti do vozlišča, iz katerega smo prišli do trenutnega vozlišča, in cene povezave med njima. V primeru več najkrajših poti algoritem vrne eno, nas pa ne zanima katero.

Za izvajanje algoritma je potrebno označiti povezave med vozlišči in jim določiti ceno.

Za vsako obiskano vozlišče shranjujemo ceno trenutno najkrajše poti do njega in po kateri izmed povezav, ki vodijo v vozlišče, smo do te cene prišli. Pri iskanju vodimo tudi seznam odprtih in zaprtih vozlišč.

V nadaljevanju sledi opis algoritma. Na začetku je v seznamu odprtih vozlišč samo začetno vozlišče, katerega cena-do-sem je nič in je brez predhodne povezave. Seznam zaprtih vozlišč je prazen. Nato ponavljamo naslednje korake, ki so ilustrirani na sliki 5.24

1. Iz seznama odprtih vozlišč vzamemo prvo vozlišče, to naj bo trenutno vozlišče. Seznam naj bo urejen naraščajoče glede na ceno-do-sem, torej: prvo vozlišče je tisto z najmanjšo ceno-do-sem.

2. Vsem vozliščem, do katerih lahko pridemo iz trenutnega vozlišča in niso na seznamu zaprtih vozlišč, s pomočjo cene-do-sem trenutnega vozlišča in vsote cene vmesne povezave izračunamo ceno-do-sem.

3. Za vsako izmed teh vozlišč, ki izračunane cene-do-sem in ustrezne vmesne povezave od trenutnega vozlišča še nima shranjene, jo shranimo.

4. Če je v prejšnjem koraku katero izmed teh vozlišč že imelo shranjeno ceno-do-sem in ustrezno povezavo v grafu iz katere od prejšnjih iteracij, ti dve ceni primerjamo in kot končen podatek shranimo manjšo ceno in ustrezno povezavo.

5. Vozlišča dodamo na seznam odprtih vozlišč in ga uredimo po naraščajoči vrednosti cene-do-sem. Takšen seznam, ki ga imenujemo vrsta s pred-

nostjo (priority queue), omogoča, da hitreje najdemo vozlišče z najmanjšo vrednostjo cene-do-sem kot pa z iskanjem po neurejenem seznamu v vsaki iteraciji. Trenutno vozlišče premaknemo na seznam zaprtih vozlišč.

Prvotno naj bi se Dijkstrov algoritem zaradi izračunavanja najkrajše poti od enega do vseh preostalih vozlišč končal takrat, ko je seznam odprtih vozlišč prazen. Če potrebujemo samo najkrajšo pot do končnega vozlišča, iskanje zaključimo, ko dodamo ciljno vozlišče na seznam zaprtih vozlišč.

Ustrezno pot izluščimo tako, da preberemo in izpišemo po kateri povezavi smo prišli do ciljnega vozlišča in ji sledimo do prejšnje povezave, kjer prav tako preberemo in izpišemo po kateri povezavi smo prišli do tja. S tem nadaljujemo, dokler ne dosežemo začetnega vozlišča. Nato seznam izpisanih povezav le še obrnemo.

Dijkstrov algoritem je popoln (če pot obstaja, jo najdejo) in optimalen (najdena pot je najkrajša), če so vse uteži povezav večje od nič.

5.4.5 A* algoritem

Algoritem A$\star$ je informiran oz. hevrističen algoritem iskanja, saj vsebuje dodatno informacijo (hevristika oz. ocena cene poti od vozlišča do cilja) zaradi česar lahko razlikuje med bolj ali manj obetavnimi vozlišči in je učinkovitejši pri iskanju rešitve. Za posamezno vozlišče, zračunamo vrednost izbrane hevristične funkcije, ki predstavlja oceno cene, potrebne za pot od tega vozlišča do cilja; imenujmo jo cena-do-cilja. Hevristična funkcija je lahko npr. Evklidska razdalja, razdalja Manhattan (sešteje premike v smereh navpično in vodoravno) ali kakšna druga primerna funkcija.

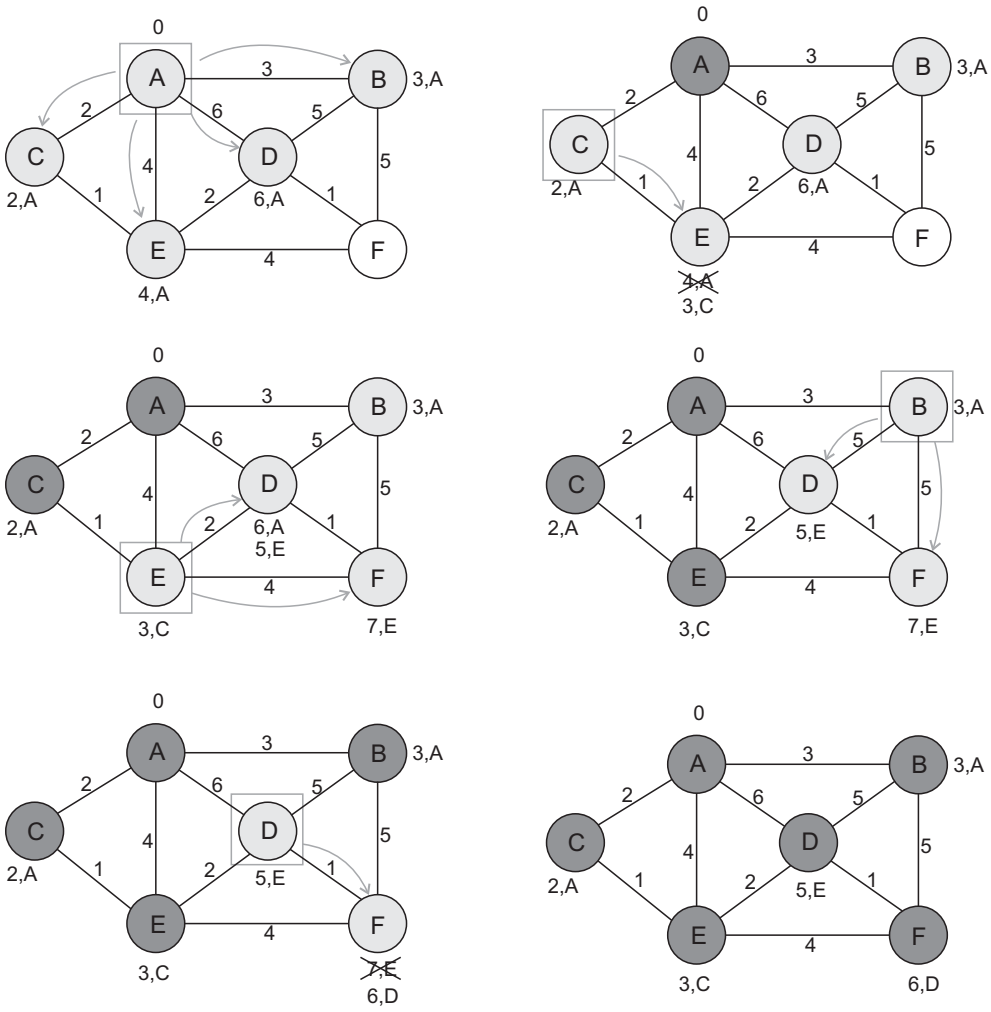
Tekom izvajanja algoritma za vozlišča računamo ceno-celotne-poti tako, da za določeno vozlišče seštejemo ceno-do-sem in njegovo ceno-do-cilja. Vodimo tudi seznama odprtih in zaprtih vozlišč. V nadaljevanju sledi opis algoritma.

Na začetku je v seznamu odprtih vozlišč samo začetno vozlišče, katerega cena-do-sem je nič in je brez predhodne povezave. Seznam zaprtih vozlišč je prazen. Nato ponavljamo naslednje korake, ki so ilustrirani na sliki 5.25.

1. Iz seznama odprtih vozlišč vzamemo prvo vozlišče, to naj bo trenutno vozlišče. Seznam naj bo urejen naraščajoče glede na ceno-celotne-poti, torej: prvo vozlišče je tisto z najmanjšo ceno-celotne-poti.

2. Vsem vozliščem, do katerih lahko pridemo iz trenutnega vozlišča izračunamo

   - ceno-do-cilja,



Slika 5.24: Dijkstrov algoritem za iskanje najkrajše poti med vozliščem A in ostalimi oglišči. Trenutno oglišče je označeno s sivim kvadratom, njegovi nasledniki pa s puščicami. Cene poti so označene ob povezavah. Ob vozliščih je označena cena-do-sem in povezava do predhodnega vozlišča. Odprta vozlišča s označena s svetlo sivo, zaprta vozlišča pa s temno sivo. Če nas zanima najkrajša pot med vozliščema A in F, vidimo, da je $Cena_{F-D-E-C-A} = 6$, pot pa gre med vozlišči $A \rightarrow C \rightarrow E \rightarrow D \rightarrow F$.

- ceno-do-sem kot vsoto cene do-sem trenutnega vozlišča in vmesne povezave ter
- ceno-celotne-poti kot vsoto cene-do-sem in cene-do-cilja.

3. Za vsako izmed teh vozlišč, ki izračunane cene-do-sem, ustrezne vmesne povezave od trenutnega vozlišča, cene-do-cilja in cene-celotne-poti še nima shranjene, jo shranimo.

4. Če je v prejšnjem koraku katero izmed teh vozlišč že imelo shranjene izračunane vrednosti iz katere od prejšnjih iteracij, primerjamo obe ceni-do-sem in kot končen podatek shranimo manjšo ceno-do-sem, temu dodamo ustrezno povezavo in ustrezno ceno-celotne-poti.

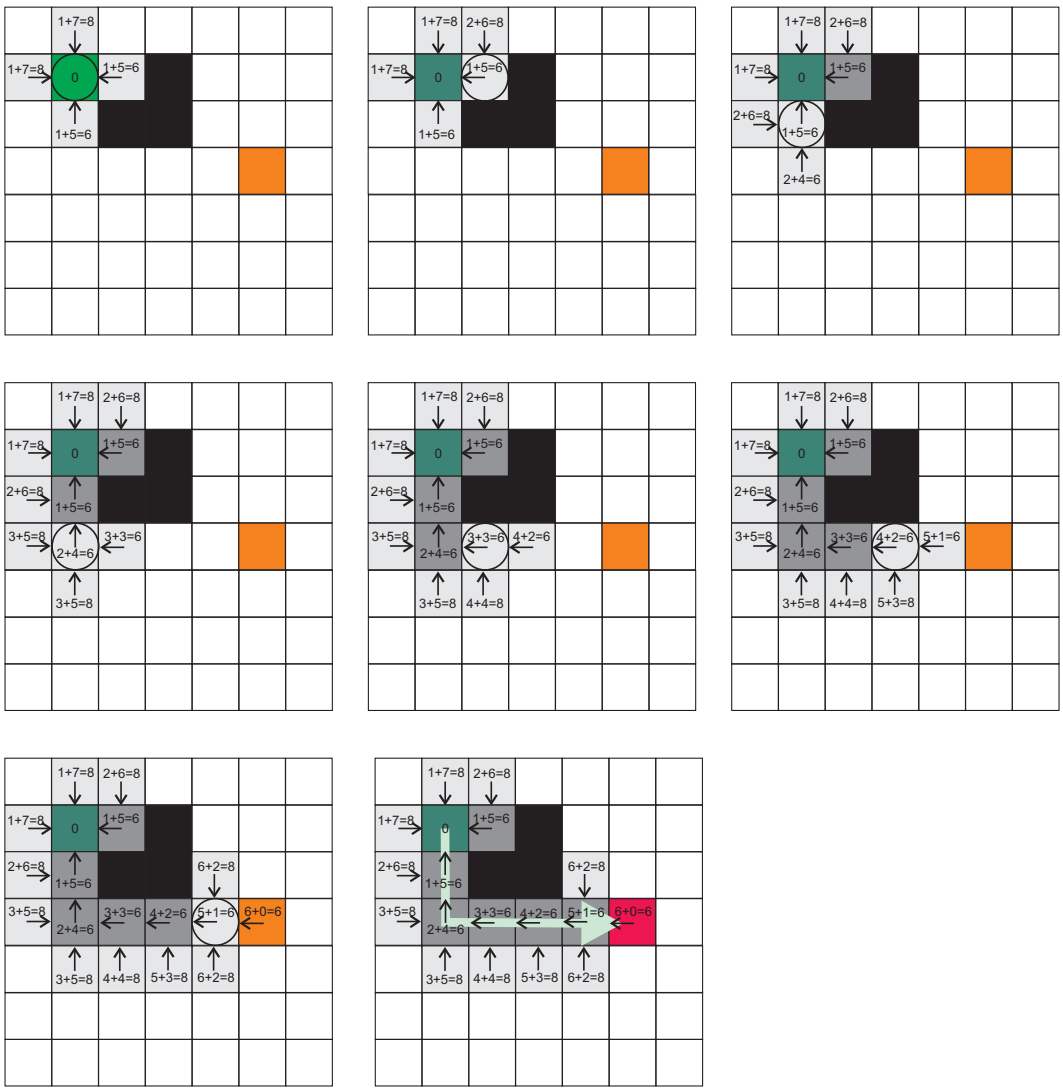
5. Vozlišča, katerih vrednosti smo računali prvič, dodamo na seznam odprtih vozlišč. Vozlišča, ki smo jim posodobili vrednosti in so že bila na seznamu odprtih vozlišč, jih tam tudi obdržimo. Vozlišča, ki so bila na seznamu zaprtih vozlišč in katerih vrednosti smo posodobili (do njega smo našli pot z manjšo ceno-do-sem), premaknemo na seznam odprtih vozlišč, ki ga nato uredimo po naraščajoči vrednosti cene-celotne-poti. Trenutno vozlišče premaknemo na seznam zaprtih vozlišč.

Algoritem A* zagotavlja optimalnost najdene poti v grafu, v kolikor je hevristika (cena-do-cilja) optimistična, kar pomeni, da je cena-do-cilja za vsako vozlišče manjša ali kvečjemu enaka resnični ceni-do-cilja. Algoritem končamo takrat, ko dodamo ciljno vozlišče na seznam zaprtih vozlišč.

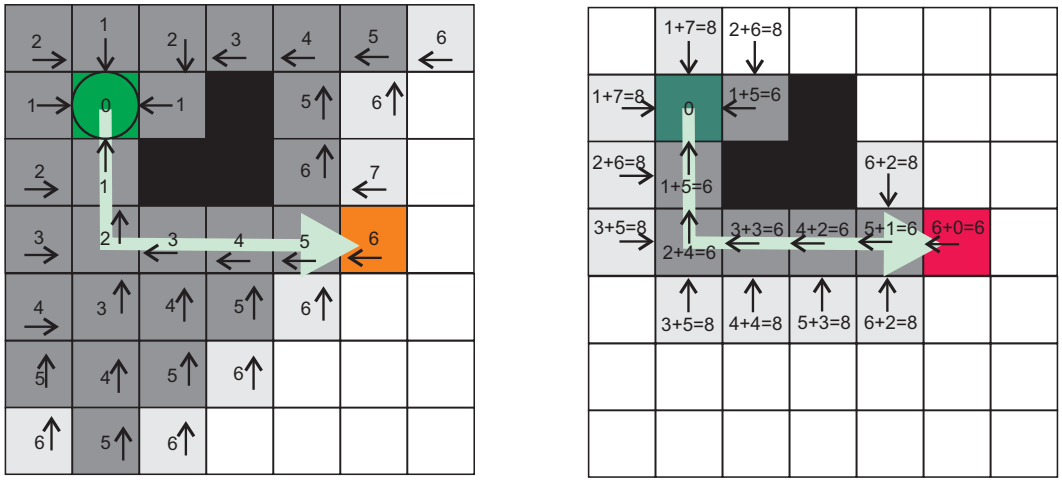
Algoritem A* je popoln, saj vedno najde pot, če le-ta obstaja, pri uporabi optimistične hevristike pa je tudi optimalen. Njegova slabost je velika poraba spomina. V primeru, da vse cene-do-cilja izenačimo z nič, dobimo algoritem, ki je enak Dijkstrovemu. Če uporabljena hevristika ni optimistična, potem dobimo algoritem A. Na sliki 5.26 je prikazana primerjava delovanja algoritma Dijkstrov in A*.

5.4.6 Pohlepno iskanje na način najprej najboljši

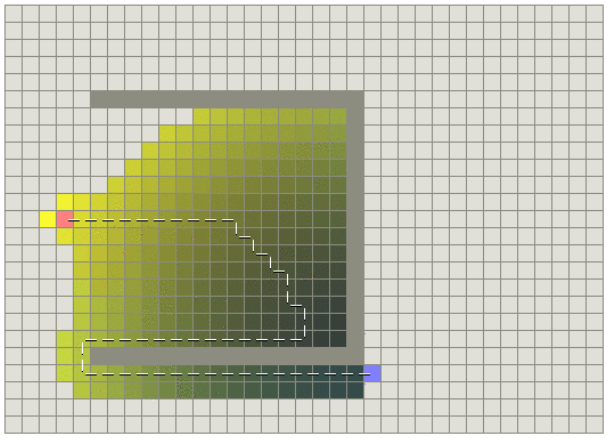
Pohlepno iskanje na način najprej najboljši (greedy best-first search) je informiran oz. hevrističen algoritem. Seznam odprtih vozlišč razvrščamo po naraščajoči ceni-do-cilja. Tako iskanje razširimo na tisto odprto vozlišče, ki je najbliže cilju, torej z najmanjšo ceno do cilja, kar je podobno kot pri iskanju v globino, dobljena pot pa prav tako ni najkrajša, kot prikazuje slika 5.27 in zato ni pomembno ali je hevristika dopustna ali ne, kot je bilo pomembno pri algoritmu A*.



Slika 5.25: Začetno vozlišče je zeleno, končno pa oranžno. Trenutno vozlišče je označeno s krogom, odprta vozlišča so svetlo siva, zaprta pa temno siva. V vsaki celici je označena smer do trenutnega vozlišča in cena poti, ki je vsota cene-do-sem in cene-do-cilja. Za izračun cene je uporabljena razdalja Manhattan. Najdeno pot razberemo s sledenjem povezav.



Slika 5.26: Primerjava algoritmov iskanja poti: Dijkstrov (levo) in A* (desno). Oba najdeta najkrajšo pot, vendar A* porabi precej manj potrebnih iteracij zaradi uporabe dodatne informacije pri iskanju poti.



Slika 5.27: Pot, ki jo najdemo s pohlepnim iskanjem na način najprej najboljši, ni optimalna.

Poglavje 6

Senzorji v mobilnih sistemih ???

Poglavje 7

Satelitski mobilni sistemi

7.1 Uvod

Okoli Zemlje kroži veliko naravnih in umetnih satelitov. Umetni sateliti so mobilni sistemi, ki jih je v vesolje izstrelil človek in služijo različnim namembnostim. Imamo telekomunikacijske satelite, televizijske in radijske (ASTRA, HOTBIRD, W2, HELLASSAT,...), satelite za telefonsko komunikacijo (IRIDIUM, INMARSAT, GLOBALSTAR), vremenske satelite (GOES, MTSAT, METEOSAT, NOAA,...), satelite za globalno pozicioniranje (GPS), satelite za daljinsko zaznavanje, kartiranje, astronomije, vojaške satelite, različne radioamaterske satelite in raziskovalne satelite.

Skupno večini satelitom je, da morajo biti za izvajanje svoje naloge pravilno orientirani. Torej morajo imeti senzorski sistem (magnetni senzor, giroskop, sledilnik zvezd,...) za določevanje svoje orientacije in aktuatorski pogon (reakcijska kolesa, potisni motorji,...) ter regulacijski algoritem za dosego želene orientacije.

Ti sistemi satelitu omogočajo, da se obrne proti želeni lokaciji na Zemlji, jo opazuje s senzorji (slikovna kamera) oz. komunicira z zemeljsko postajo na tej lokaciji. Nadalje morajo biti omenjeni sistemi energijsko učinkoviti, saj ima satelit v vesolju omejeno avtonomijo in jo lahko pridobiva le iz svojih baterij in/ali rezervoarjev z omejeno kapaciteto oz. polni akumulatorje preko sončnih celic, ko je obsijan s soncem.

Stabilizacijo in orientacijo satelita uravnava regulacijska zanka (pravzaprav tri, ker je treba satelit stabilizirati po treh oseh), ki temelji na senzorjih orientacije satelita, aktivatorjih za njegovo vrtenje in algoritmih za vodenje sistema. Senzorji orientacije običajno temeljijo na določitvi dveh linearno neodvisnih vektorjev znanih smeri, običajno sta to smer sonca in smer zemeljskega magnetnega polja. Ko je satelit v Zemljini sonci pa lahko uporabljamo

druge merilnike za orientacijo giroskopi, sistemi za razpoznavanje zvezd in horizonta.

Ko je orientacija satelita znana in ni enaka želeni, jo je potrebno spremeniti z aktivatorji.

Zelo pogosti aktuatorji so reakcijska kolesa (električni vztrajniki). Ker leti "nabirajo"moment (se vrtijo vedno hitreje), jih je potrebno razbremeniti, kar lahko naredimo z mikro-potisnimi motorji (micro-thruster), ki temeljijo na MEMS tehnologiji in omogočajo izjemno natančno kontrolo položaja satelita s potisno silo velikostnega reda mN. V nizki orbiti, kakršna je predvidena, so možne tudi razbremenitve z magnetnimi tuljavami.

Takšen stabilizacijski in orientacijski podsistem je lahko zelo kompleksen in zahteva različna znanja, od kompleksnega programiranja in modeliranja, do finomehanike in elektrooptike.

7.2 Senzorji in aktuatorji

7.2.1 Senzorji

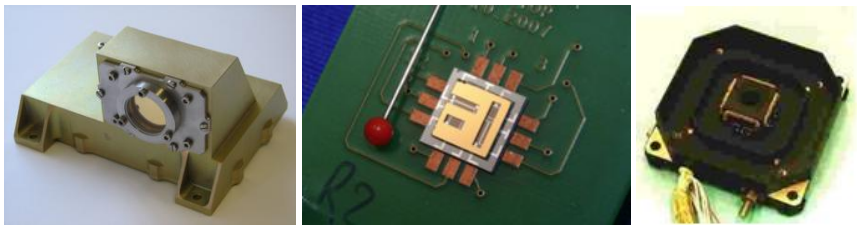
Satelit lahko vsebuje senzorje za oceno njegove orientacije v vesolju in druge senzorje, povezane z njegovo misijo (različne kamere pri vremenskih satelitih, daljinskem zaznavanju, meritve radiacije, temperature,...).

V nadaljevanju opišemo najbolj pogosto uporabljene senzorje in metodologijo potrebno za določitev orientacije satelita iz meritev teh senzorjev.

Želeno orientacijo satelita dosežemo lahko le če poznamo trenutno orientacijo, ki jo lahko izmerimo z različnimi senzorji. Večina teh senzorjev daje absolutno informacijo, saj dajo vektor orientacije do objekta zaznavanja, katerega pozicija je znana (sonce, zemlja, zvezde). Kadar informacija absolutnih senzorjev ni na voljo, pa orientacijo lahko ocenjujemo s senzorji z relativno informacijo (giroskopi), ki podajo le spremembo zasuka od prejšnje meritve. Za določitev orientacije satelita sta potrebne dve različni meritvi absolutnih senzorjev (dva nekolinearna vektorja znanih smeri) ali pa si pomagamo s preteklo informacijo. Najbolj pogosto uporabljen pristop združevanja informacij večih senzorjev in upoštevanja preteklih meritev je Kalmanov filter.

Senzor sonca

Najbolj preprost in praktičen senzor sonca, ki se uporabljajo na manjših satelitih (Cubesat) lahko predstavlja kar sistem sončnih celic montiranih na stranicah satelita, ki hkrati zagotavlja potrebno električno energijo za delovanje satelita. Iz razmerja osvetlitev oz. tokov, ki jih proizvajajo posamezne



Slika 7.1: Levo je napredni senzor večjih dimenzij in desno izgled dveh manjših senzorjev.

sončne celice na stranicah satelita lahko določimo smerni vektor v smeri sonca. Bolj napredne rešitve, z večjo točnostjo, predstavljajo digitalni senzorji na osnovi CCD senzorjev oz. svetlobno občutljivih elementov. Dva primera sta podana na sliki 7.1. Prvi senzor je profesionalni in nekoliko večji (1.4kg) z območjem pogleda 110° in natančnostjo pod 0.2°, druga dva pa sta manjša dvoosna senzorja z natančnostjo cca. 1° na os. Senzor ima to slabost, da ne deluje, kadar je satelit v senci Zemlje, kar je za satelite v nizki zemljini orbiti skoraj polovico časa. Nadalje je njegova natančnost omejena na približno 0.1°,

Dobljen vektor smeri sonca primerjamo z referenčnim vektorjem, ki ga izračunamo na osnovi modela sončnega obsevanja. Ta primerjava je del sistema za ocenjuje orientacije satelita.

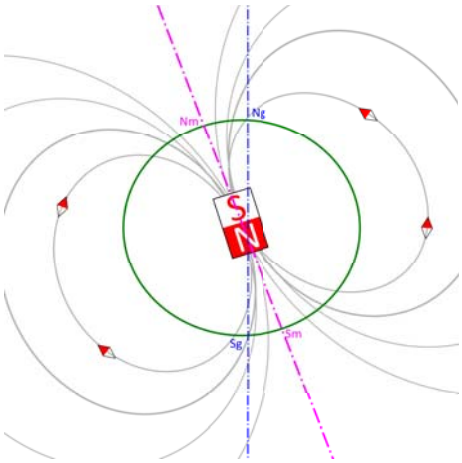
Senzor magnetnega polja

Meritev zemeljskega magnetnega polja z magnetometrom je mogoča le v nizkih tirnicah (do 1000 km), kjer je polje še dovolj močno. Tak senzor sestoji iz treh ortogonalnih senzorjev in izmeri smerni vektor Zemeljskega magnetnega polja v koordinatnem sistemu senzorja. Dobljena meritev je primerjana z referenčnim vektorjem polja, ki ga določimo iz poznanega modela zemeljskega polja in poznane oz. ocenjene točke v orbiti. Ta primerjava je del sistema za ocenjuje orientacija satelita.

Obstaja veliko izvedb magnetometrov od najpreprostejše, ki predstavlja navitje žice oz. tuljava do manjših, varčnejših in točnejših izvedb.

Na točnost meritve magnetometra vplivajo:

- motnje zaradi elektronike v satelitu,
- točnost referenčnega modela zemeljskega polja (točnost določitve referenčne orientacije),



Slika 7.2: Dipolni model zemeljskega magnetnega polja.

- zunanje motnje kot so nepredvidljivi ionosferski tokovi.

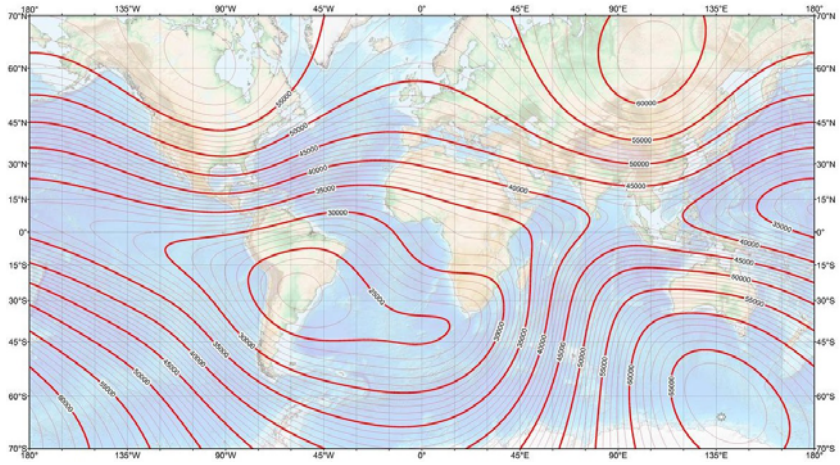
Najbolj osnoven model zemeljskega magnetnega polja je magnetni dipol s poloma v Zemljinih magnetnih polih (slika 7.2).

Dejansko zemeljsko magnetno polje se s časom spreminja in bolj slabo ustreza modelu magnetnega dipola. Izgled magnetnega polja Zemlje podaja slika 7.3). Magnetno polje zato opisujemo z bolj kompleksnim matematičnim modelom, ki temelji na eksperimentalno določenih koeficientih za vrsto sferičnih harmonikov. V ta namen se uporabljajo IGRF modeli (International Geomagnetic Reference Field), ki so ocenjeni na vsakih 5 let.

Sledilnik zvezd

Sledilniki zvezd so zelo natančni senzorji za določitev orientacije z natančnostjo do nekaj tisočink stopinje. V osnovi gre za senzor, kjer se orientacija senzorja določi s pomočjo zajete slike zvezd in njene primerjave s katalogi zvezdnega neba.

Sledilnik zvezd s pomočjo kamere posname zvezde v vidnem polju kamere. Nato s pomočjo primerjave (svetilnosti, vzorcev,...) poišče te iste zvezde v podanem katalogu ozvezdja (npr. hipparcos catalog), kjer so za posamezne zvezde podane oznaka, pozicija, svetilnost, razdalja, itd. Ker je pozicija zvezde v inercialnem koordinatnem sistemu (npr. J2000) fiksna se ohranja tudi medsebojna lega zvezd. S pomočjo posnetih zvezd na pogledu kamere, ustreznega algoritma in istih zvezd v podanem katalogu, lahko oce-



Slika 7.3: Magnetno polje na površju zemlje, model za leto 2010. Podana je absolutna vrednost magnetne poljske jakosti v nT.

nimo orientacijo pogleda satelita v inercialnem koordinatnem sistemu.

To so večinoma dragi, računsko in energetsko potratni ter veliki senzorji.

Senzor horizonta

Z njimi določimo relativno pozicijo Zemlje glede na satelit. Ponavadi merijo infrardeči pas Zemlje (14-16 μm), ki predstavlja zgornjo Zemljino CO2 atmosfero. Ta plast je uporabna tako podnevi, kot ponoči ali oblačnem vremenu. To so dragi in veliki senzorji.

Giroskop

Meri spremembe orientacije okoli koordinatnih osi. Končna orientacija satelita je integral premikov. Pri giroskopih je stalno prisotno lezenje, zato so ti senzorji primerni le v kombinaciji z ostalimi absolutnimi senzorji, ki lahko rekalibrirajo žiroskop.

Mehanski giroskopi merijo spremembo orientacije. Mehanski giroskopi so dragi.

Večinoma jih nadomeščajo piezoelektrični in optični giroskopi, ki merijo hitrost rotacije. Piezoelektrični giroskopi merijo spremembo frekvence vibracij, ki nastane pri rotaciji zaradi Coriolisove sile.

Predvsem pa so popularni optični giroskopi saj nimajo mehanskih delov in ne potrebujejo vzdrževanja. Vsebujejo dva laserska žarka, ki potujeta v

senzor	točnost [°]	prednost	slabost
senzor sonca	0.1	preprost, zanesljiv, poceni	ni uporaben v Zemljini senci
senzor horizonta	0.03	uporaben podnevi in ponoči	drag, slaba ocena kota yaw
magnetometer	1	poceni, stalna pokritost	uporaba le v nizkih orbitah
sledilnik zvezd	0.001	zelo točen	težak, kompleksen, drag
giroskop	0.01/h	hiter	časovno lezenje

Tabela 7.1: Primerjava lastnosti senzorjev

nasprotnih smereh po sklenjeni poti (zanka iz steklenega vlakna). Laserski žarek, ki potuje v smeri rotacije ima nekoliko krajšo pot, kar povzroči višjo frekvenco. Sprememba frekvence obeh žarkov je proporcionalna kotni hitrosti vrtenja.

Globalna navigacija

Globalna navigacija (GPS) nudi možnost za precej natančno in ekonomično rešitev (iz vidika stroškov, mase, energije in volumna) določitev orientacije satelita v kombinaciji z ostalo navigacijo na satelitu. Problem predstavljata vsaj dva razloga, ki govorita proti tovrstni uporabi. Prvi je, da si odvisen od delovanja GPS satelitov, s katerimi upravlja ameriška vojska. Drugi problem, pa je v tako imenovanem COCOM dogovoru, ki pravi, da so GPS sprejemniki ne delujejo nad 18 km višine in hitrosti nad 515,4m/s. Satelit v nizki orbiti kroži na več kot 20x večjih višinah in z več kot 10x večjo hitrostjo. V kolikor lahko lahko dobimo GPS sprejemnik brez teh omejitev, potem ga je možno uporabiti. Možni načini uporabe GPS so, da dobimo precej točno pozicijo satelita, ki jo potrebujemo za določitev referenčnega vektorja zemeljskega magnetnega polja in nato orientacije satelita. V kolikor pa uporabimo dve GPS anteni na neki medsebojni razdalji (čim večji) lahko iz določitve faznega zamika nosilca v GPS signalu določimo orientacijo satelita, razen orientacije okoli osi, ki povezuje anteni.

Primerjava lastnosti senzorjev

V tem poglavju so našteti najbolj pogosto uporabljeni senzorji za določitev orientacije satelita. Njihove lastnosti primerjamo v tabeli 7.1.

7.2.2 Aktuatorji

Aktuatorji so na satelitu potrebni, da spremenijo njegovo orientacijo oz. lego v orbiti. Sama izvedba satelita, namen in orbita pogojujejo uporabo aktua-

torjev. Zemeljsko magnetno polje lahko koristno uporabljamo le v nizkih orbitah, ko je jakost polja dovoj velika. Majhni (pico) sateliti so omejeni z velikostjo, tako da je uporaba vztrajnikov in micro-potisnih motorjev (microthrusters) zelo redka. Po drugi strani pa sateliti za daljinsko opazovanje zahtevajo zelo točno orientacijo, kar narekuje uporabo vztrajnikov in potisnih motorjev. V nadaljevanju podajamo nekaj osnovnih aktuatorjev uporabnih predvsem v nizkih orbitah, ostali, pa so le bežno omenjeni.

Elektromagnet

Elektomagnet (ang. magnetic tourquers) izkorišča zemeljsko magnetno polje in je torej uporaben v nizkih orbitah. Lahko so le navitja bakrene žice (tuljave), kjer je rezultirajoči navor T, ki ga povzoča tok i skozi tuljavo z N ovoji in ploščino A v zemeljskem magnetnem polju B

$$T = B \times iNA \tag{7.1}$$

Če tuljavo navijemo okoli feromagnetnega jedra z relativno permeabilnostjo μ (do 106) dobimo navor

$$T = B \times iN\mu A \tag{7.2}$$

S tem dosežemo isti navor pri veliko manjšem toku. Slabost pa je večja teža zaradi feromagnetnega jedra in histereza jedra, ki vstopa v povratno zanko.

Elektromagneti so praktično edini uporabni aktuatorji v pico satelitih, v mikro in večjih satelitih pa se uporabljajo za razbremenitev reakcijskih koles.

Potisni motorji

Izkoriščajo pojav, ki ga opisuje zakon o vzajemnem učinku (3. Newtonov zakon ali zakon o akciji in reakciji). Pri teh motorjih (thrusters, microthrusters) pulzirajoči izhodni plin skozi šobe motorja povzroči gibanja satelita v nasprotno smer. V kolikor smer pulzirajočega potisnega plina ni točno v nasprotni smeri zveznice motorja in težišča satelita, to povzroči (tudi) navor oz. rotacijo satelita. Pri micro potisnih motorjih so dobljene sile velikostnega reda mN. Slabost tovrstnega pogona so potrebni rezervoarji potisnega plina, torej dodatna teža satelita in omejen čas delovanja.

Reakcijska kolesa

Izkoriščajo pojav, ki ga opišemo z rotacijskim zakon o vzajemnem učinku. Pospeševanje vrtenja reakcijskega kolesa (vztrajnik) v satelitu bo povzročilo kotni pospešek satelita v nasprotni smeri. Za poljubno orientacijo satelita

potrebujemo tri reakcijska kolesa (električni vztrajniki) postavljena medsebojno ortogonalno, oz. štiri kolesa postavljena v oglišča tetraedra, kar omogoča večjo zanesljivost v primeru odpovedi enega od njih. Reakcijska kolesa omogočajo najbolj točno orientacijo satelita. Zaradi velikosti in teže tovrstni aktuatorji niso primerni za najmanjše (pico) satelite so pa vsekakor prisotni v večjih (od micro dalje).

Momentna kolesa

Momentna kolesa se od reakcijskih razlikujejo po tem, da se stalno vrtijo in imajo zaradi tega giroskopsko stabilnost (držanje smeri). Torej lahko odpravijo vpliv kratkotrajnih zunanjih motenj.

Giroskopska momentna kolesa

Pri giroskopskih momentnih kolesih gre za isti pojav, le da imamo tu le eno momentno kolo, katerega os rotacije lahko spreminjamo (control moment gyros, gimballed momentum wheel).

Pasivni načini orientacije satelita

Njihova skupna lastnost je, da za dosego želene orientacije ne potrebujejo dodatne energije (elektrika, plin, ...).

Gradientno težnostno vodenje deluje na principu zmanjšanja težnosti z oddaljenostjo od Zemlje. Podolgovato telo se tako zaradi različnih velikosti gravitacijskih sil, ki povzročijo navore v različnih točkah telesa, postavi z daljšo osjo v smeri središča Zemlje. Ponavadi je to izvedeno z dolgim drogom z maso na koncu. Točnost orientacije, ki jo lahko dosežemo je nekaj stopinj (1 do 10°). Slabost v primeru manjših satelitov je tudi potreben dolg tog drog, kateri mora biti zložen med izstrelitvijo v orbito.

Pasivno magnetno vodenje je mogoče zaradi internega magnetnega polja satelita, zaradi katerega je plovilo magnetni dipol, ki se uskladi z Zemeljskim magnetnim poljem. Magnetno polje v satelitu lahko dosežemo s trajnim magnetom v satelitu ali nastane zaradi električnih virov v njem in prevodnostnih površin. Točnost orientacije, ki jo lahko dosežemo je podobna kot pri gradientnem vodenju.

Vodenje in stabilizacija z vrtenjem je mogoče, če se satelit vrti okoli ene osi. Giroskopski efekt odpravi vpliv zunanjih motilnih navorov. Vrtenje satelita lahko dosežemo na več načinov. Z različno pobarvanimi

metoda	točnost [°]	prednost	slabost
stabilizacija z vrtenjem	0.1-1	preprost, pasiven, poceni	inercijsko orientiran
gradientno težnostna	1-5	pasiven, poceni, preprost	drag, težiščno orientiran
potisni motorji	0.01-1	hiter	cena, potrošen
elektromagneti	1-2	poceni, lahek, počasen, samo LEO	
reakcijska kolesa	0.001- 1	hiter, natančen	cena, teža

Tabela 7.2: Primerjava lastnosti aktuatorjev

stranicami satelita je navor zaradi sonca (solar pressure) večji na svetlih kot na temnejših površinah. Nadalje lahko vrtenje dosežemo s potisnim motorjem in ga vzdržujemo z magnetnimi tuljavami (elektromagneti). Namesto vrtenja celotnega satelita, lahko uporabimo momentno kolo, ki nam da isti efekt.

Primerjava aktuatorjev

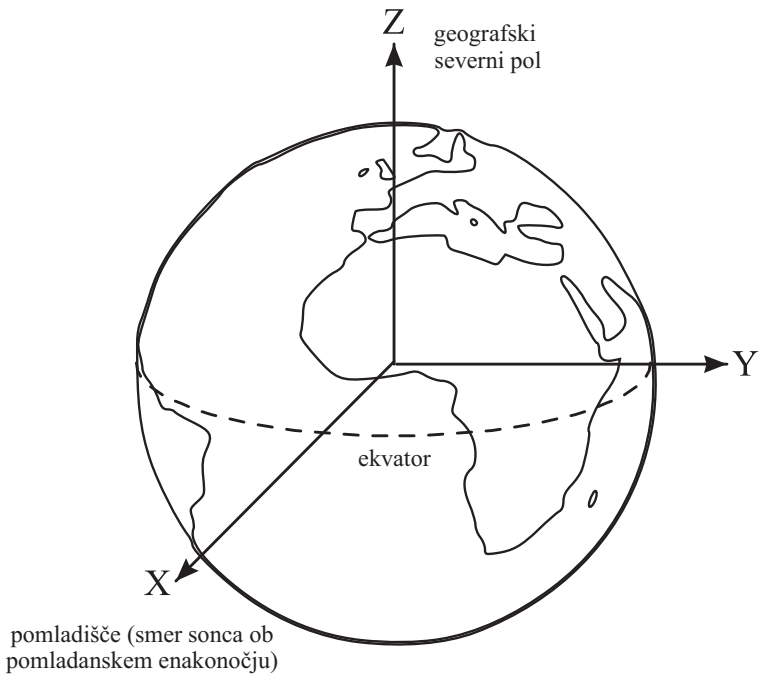
Najbolj pogosto uporabljeni aktuatorji na satelitih in njihova primerjava je podana tabeli 7.2

7.3 Referenčni koordinatni sistemi

Referenčni koordinatni sistemi so pomembni, ker so v njih podani parametri okolja satelita, njegov položaj in orientacija. Poglavje je vzeto iz [21].

Inercialni koordinatni sistem z izhodiščem v središču Zemlje (Earth Centered Inertial Frame - ECI)

Inercialni sistemi so sistemi, ki mirujejo oz. se enakomerno gibljejo glede na oddaljene zvezde. Pomembni so zato, ker v njih veljajo Newtonovi fizikalni zakoni. ECI sistem ima izhodišče v masnem središču Zemlje, njegova x os kaže v smeri pomladišča (smer sonca ob pomladanskem enakonočju), njegova z os gre skozi severni tečaj, y os pa je definirana tako, da vse tri osi tvorijo desnoročni sistem. Strogo gledano ta sistem ni čisto inercialen tudi glede vrtenja, saj se zaradi precesije zemeljske osi pomladišče vrti glede na zvezde s periodo 25800 let.



Slika 7.4: ECI koordinatni sistem

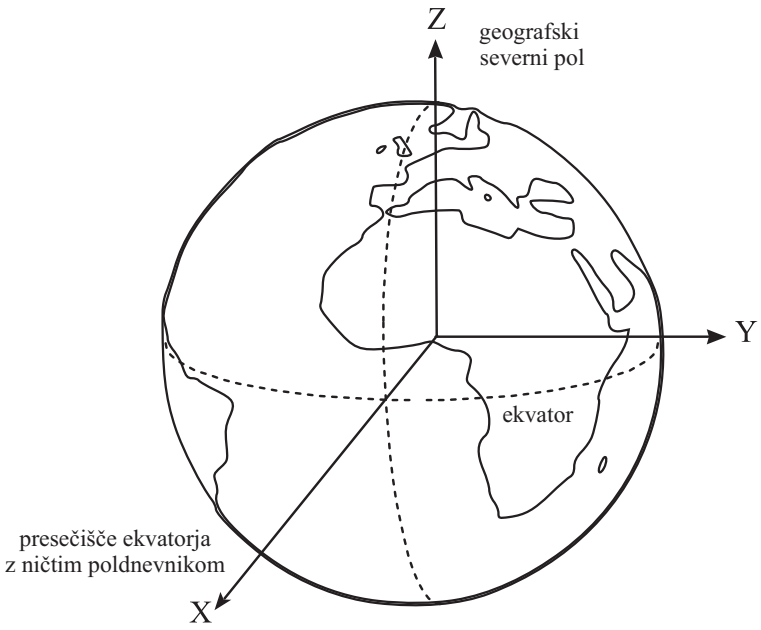
Na Zemljo pritrjen koordinatni sistem (Earth Centered Earth Fixed Frame - ECEF)

.

Ta sistem ima izhodišče v središču Zemlje, njegova x os kaže v presečišče ekvatorja z ničtim poldnevnikom, njegova z os gre skozi severni tečaj, y os pa je definirana tako, da vse tri osi tvorijo desnoročni sistem. ECEF sistem se glede na ECI sistem vrti s kotno hitrostjo $\omega_0 = 7,2921 \times 10^{-5}\ rad/s$. Pomemben je za izračun položaja satelita glede na točke na Zemlji in za izračun orientacije satelita glede na smer zemeljskega magnetnega polja.

Orbitalni koordinatni sistem z izhodiščem v središču zemlje (Earth Centered Orbit Frame - ECO)

ECO koordinatni sistem je pomemben, ker so v njem podani Keplerjevi elementi tirnice. Izhodišče ima v središču Zemlje, njegova x os kaže proti perigeju, njegova y os v smeri manjše polosi elipse, njegova z os pa pravokotno na ravnino tirnice. Ta sistem je proti sistemu ECI zavrten za rektascenzijo dvižnega vozla Ω, naklon tirnice i in argument perigeja ω.



Slika 7.5: ECEF koordinatni sistem.

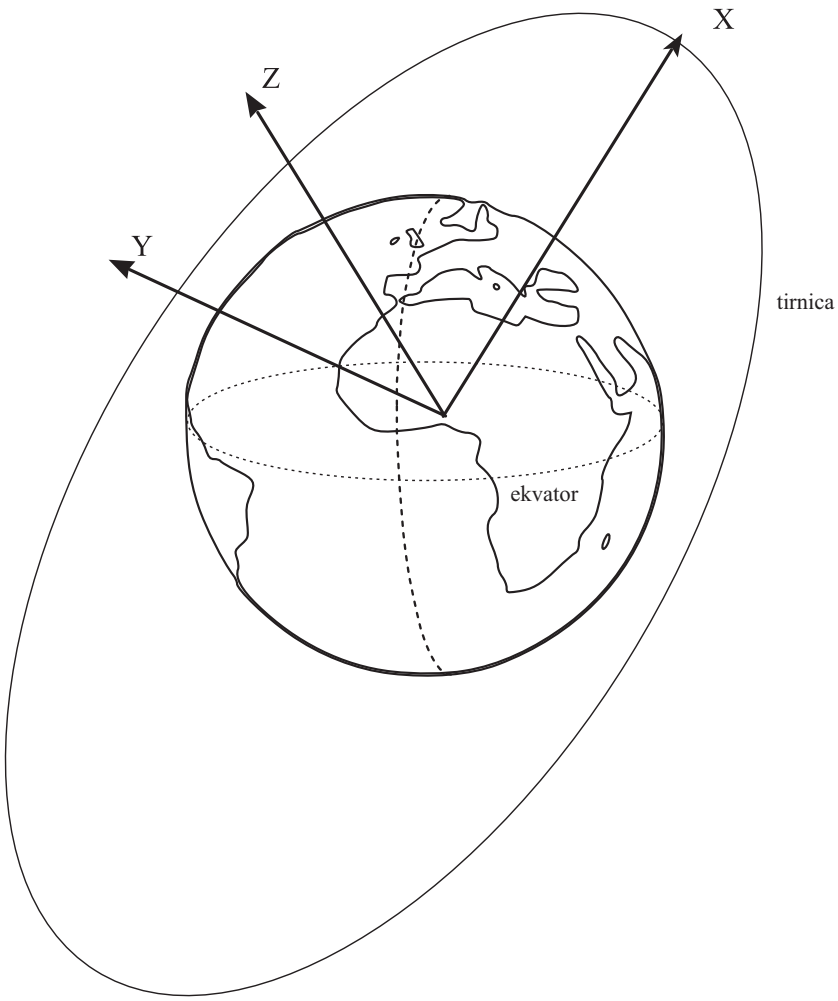
Orbitalni koordinatni sistem (Orbit Frame - O)

Izhodišče tega sistema v masnem središču satelita in rotira glede na ECO koordinatni sistem s kotno hitrostjo $\dot{\varphi}$, kjer je φ prava anomalija. Njegova z os je v smeri središča Zemlje, y os je pravokotna na ravnino tirnice, njegova x os pa kaže v smeri gibanja satelita tako, da vse tri osi tvorijo desnoročni sistem. Poudariti je potrebno, da sovpada os x s smerjo vektorja hitrosti (tangencialno na tirnico) le pri krožnih tirnicah.

Ta koordinatni sistem je pomemben za izračun orientacije satelita glede na Zemljo.

Lokalno vertikalen lokalno horizontalen koordinatni sistem (Local Vertical Local Horizontal Frame - LVLH)

Tako kot pri **O** koordinatnem sistemu je tudi pri LVLH sistemu izhodišče v masnem središču satelita in rotira glede na ECO koordinatni sistem s kotno hitrostjo $\dot{\varphi}$, kjer je φ prava anomalija. Njegova x os je radialno v smeri nasprotni proti središču Zemlje, z os je pravokotna na ravnino tirnice, njegova y os pa kaže v smeri gibanja satelita tako, da vse tri osi tvorijo desnoročni sistem. Gibanja v smeri osi x, y in z imenujemo tudi radialno, v smeri tirnice in prečno na tirnico (radial, in-track, cross-track). Poudariti je potrebno,



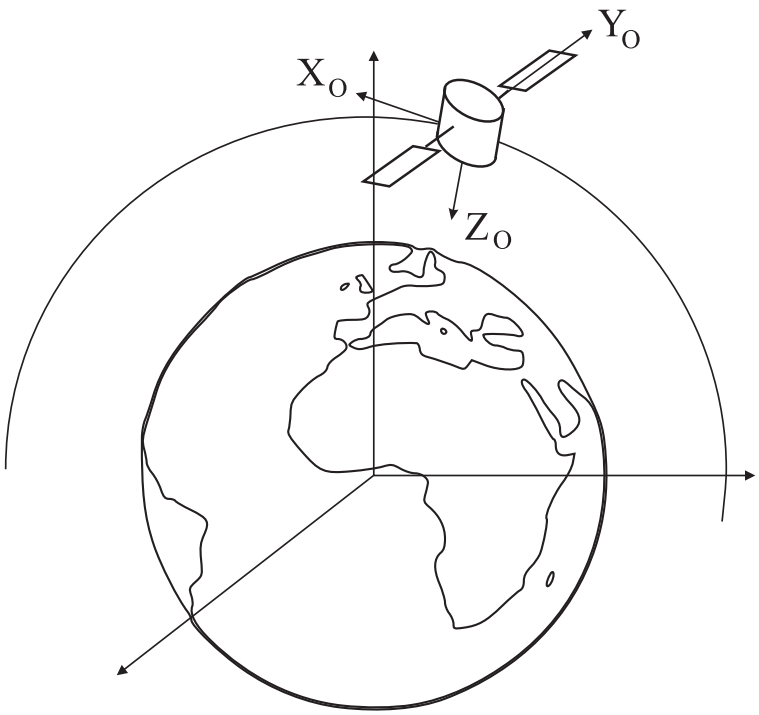
Slika 7.6: ECO koordinatni sistem.

da sovpada os y s smerjo vektorja hitrosti (tangencialno na tirnico) le pri krožnih tirnicah. Ugotovimo, da je LVLH koordinatni sistem konstantno obrnjen proti O koordinatnemu sistemu.

Ta koordinatni sistem se uporablja za obravnavo relativnih pozicij satelitov pri letenju v formaciji.

Koordinatni sistem satelita (Body Frame - B)

Izhodišče tega sistema je v masnem središču satelita, njegova orientacija pa je določena z geometričnimi lastnostmi satelita. V tem koordinatnem sistemu so podane orientacije senzorjev, aktuatorjev, kamer in usmerjenih anten.



Slika 7.7: Ilustracija O koordinatnega sistema.

7.4 Matematične predstavitve orientacije

Orientacijo satelita spreminjamo z njegovim vrtenjem. Vzemimo desno sučni koordinatni sistem, ki je pritrjen na satelit in katerega enotski vektorji so u,v,w. Orientacijo satelita opišemo z orientacijo koordinatnega sistema u,v,w proti referenčnemu koordinatnemu sistemu x,y,z. Referenčni koordinatni sistem je običajno ECI. To storimo tako, da podamo komponente vektorjev u,v,w vzdolž smeri x,y,z. Tako dobimo devet parametrov, ki jih združimo v matriko 3×3.

$$\mathbf{R} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \tag{7.3}$$

Enotske vektorje koordinatnega sistema, ki je pritrjen na satelit opišemo v referenčnem koordinatnem sistemu kot sledi: $\mathbf{u} = [u_1, u_2, u_3]^T$, $\mathbf{v} = [v_1, v_2, v_3]^T$, $\mathbf{w} = [w_1, w_2, w_3]^T$ in

$$\mathbf{u} \times \mathbf{v} = \mathbf{w} \tag{7.4}$$

Vektor v koordinatnem sistemu satelita $\mathbf{v}_{uvw}$ izrazimo z vektorjem v referenčnem koordinatnem sistemu $\mathbf{v}_{xyz}$s transformacijo

$$\mathbf{v}_{uvw} = \mathbf{R}\mathbf{v}_{xyz} \tag{7.5}$$

Matrika $\mathbf{R}$ torej transformira vektor iz referenčnega koordinatnega sistema v koordinatni sistem satelita. Elementi matrike R so kosinusi kotov med posameznimi osmi obeh koordinatnih sistemov, zato matriko R imenujemo tudi matriko smernih cosinusov (Direction cosine matrix - DCM). Ker ima orientacija tri prostostne stopnje, matrika DCM pa devet elementov, seveda ti elementi niso medsebojno neodvisni. Ker so vektorji $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ enotski vektorji, velja da je vsota kvadratov elementov vsake vrstice enaka 1, ker pa so medsebojno pravokotni, je njihov skalarni produkt enak nič. Tako dobimo šest dodatnih omejitvenih enačb. Njihova posledica je da so matrike DCM ortogonalne (njihov inverz je enak transponirani matriki). Zato so absolutne vrednosti lastnih vektorjev enake 1, in obravnavana transformacija predstavlja rotacijo. Ena lastna vrednost oz. en lastni vektor sta realna in zanju velja

$$\mathbf{R}\mathbf{e} = \mathbf{e} \tag{7.6}$$

kjer je $\mathbf{e}$ realni lastni vektor. Njegova smer se pri transformaciji (rotaciji) ne spremeni, zato predstavlja os vrtenja. Elementarne transformacije dobimo z vrtenjem okrog osi x, y, z, zato obstajajo tri elementarne rotacijske matrike

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{7.7}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{7.8}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.9}$$

kjer oznaka $\mathbf{R}_{os}(kot)$ indicira rotacijo za kot okrog osi. Sukcesivne rotacije opišemo s produktom rotacijskih matrik - važen je vrstni red rotacij! DCM je elementarna parametrizacija orientacije togega telesa.

V nadaljevanju bomo opisali še dve metodi parametrizacije, ki so včasih bolj primerne.

7.4.1 Eulerjevi koti

Eulerjevi koti opisujejo orientacijo togega telesa z vrtenjem okrog osi x, y, z. Uveljavljen način označevanja kotov izhaja iz letalske tehnike, kjer so koti označeni kot

- ϕ - kot valjanja (angl. roll)
- θ- kot prevračanja (angl. pitch)
- ψ - kot sukanja (angl. yaw, heading).

Možnih je 12 kombinacij zasukov okrog osi x, y, z [22]. Najbolj uveljavljena je kombinacija 3-2-1, pri kateri dobimo orientacijo satelita glede na referenčni koordinatni sistem tako, da iz začetne lege, ko sta oba koordinatna sistema poravnana, zavrtimo koordinatni sistem satelita v naslednjem vrstnem redu.

1. Najprej okrog osi z za kot ψ (kot sukanja - yaw)
2. Nato okrog nove osi y za kot Θ (kot prevračanja - pitch)
3. Nato okrog nove osi x za kot ϕ (kot valjanja - roll)

DCM matrika takšne transformacije je

$$R = R_x(\phi)R_y(\theta)Rz(\psi) =$$
$$\begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{7.10}$$

Obratna transformacija se glasi

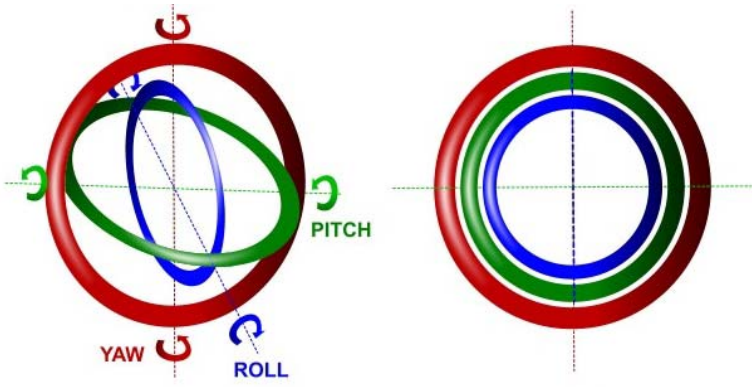
$$\begin{aligned} \phi &= \arctan(\frac{R_{23}}{R_{33}}) \\ \theta &= -\arcsin(R_{13}) \\ \psi &= \arctan(\frac{R_{12}}{R_{11}}) \end{aligned} \tag{7.11}$$

Parametrizacija orientacije z Eulerjevimi koti ni redundantna (ima samo tri parametre), ima pa slabost, ker postane pri zasuku $\theta = \pi/2$ singularna. Takrat imata namreč vrtenji okrog z in x osi isti učinek (sovpadeta). Pri klasičnih letalskih žiroskopih takrat omenjeni osi tudi sovpadeta in ta efekt je poznan pod imenom "Gimbal Lock". Pri simulaciji kinematike leta z Eulerjevimi koti se ta singularnost odraža tako, da se v imenovalcih izrazov pojavi $\cos\theta$ (glej poglavje 7.6).

7.4.2 Kvaternioni

Parametrizacija orientacije z Eulerjevimi koti, kot opisano zgoraj, je kolikor možno kompaktna (trorazsežni vektor parametrov za predstavitev orientacije v trorazsežnem prostoru), vendar ima to slabost, da pri zasuku $\theta = \frac{\pi}{2}$ postane singularna - zasuka okoli osi z ali x takrat sovpadata (Slika 7.8). Pri simulaciji kinematike (izračun kotnih hitrosti) je to očitno, ker se v imenovalcu izraza za $\dot{\Psi}$ pojavi $cos\,\theta$. To velja za kombinacijo zasukov 3-2-1, vendar tudi pri ostalih izmed 12 možnih kombinacijah pride do istega pojava na drugih oseh.

Analogijo problemu predstavlja problem predstavitve položaja na zemeljski površini z dvema parametroma (geografsko širino in dolžino) - v geografskem severnem oz. južnem polu eden od parametrov - geografska dolžina - nima pomena. Tu problem rešimo tako, da predstavitev pozicije razširimo na tri parametre s kartezičnim koordinatnim sistemom in dodatno omejitvijo $\sqrt{x^2 + y^2 + z^2} = R_E$, tako da imamo še vedno dve stopnji prostosti. Podobno s kvaternioni razširimo predstavitev 3D orientacij iz treh na štiri parametre in vpeljemo en dodaten pogoj, s čimer se izognemo zgoraj opisanemu problemu singularnosti. Slika 7.8 ponazarja problem singularnosti Eulerjevih kotov na primeru troosnega giroskopa - ko so vse osi poravnane, se lahko sistem premika samo v dve smeri, torej izgubi prostostno stopnjo.



Slika 7.8: Singularnost Eulerjevih kotov v troosnem giroskopu.

Matematično gledano so kvaternioni sistem hiperkompleksnih števil, ki predstavlja nekomutativno razširitev kompleksnih števil [25]. Splošna oblika zapisa kvaterniona je

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{7.12}$$

kjer za kompleksne elemente i, j in k velja zveza

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \tag{7.13}$$

Kvaternion oblike $q_0 + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$ (q_0 je realno število), se imenuje realni del kvaterniona. Kvaternion, ki ima obliko $0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ (q_1, q_2 in q_3 so realna števila), se imenuje čisti imaginarni kvaternion. Če je kvaternion $q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$, potem imenujemo q_0 skalarni del kvaterniona in $q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ imenujemo vektorski del. Čeprav je vsak kvaternion vektor v štirirazsežnem vektorskem prostoru, lahko definiramo vektor kot čisti imaginarni kvaternion.

Konjugirana vrednost kvaterniona se določi podobno kot se določi konjugirana vrednost kompleksnega števila. Kadar je kvaternion enak $\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$, je njegova konjugirana vrednost enaka $\mathbf{q}^* = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}$. Označujemo jo kot $\mathbf{q}^*$ ali $\bar{\mathbf{q}}$. Konjugacija je involucija, kar pomeni, da pri dvakratni konjugaciji dobimo prvotni element. Konjugacija produkta je produkt konjugiranih vrednosti v obratnem vrstnem redu

$$(\mathbf{pq})^* = \mathbf{q}^*\mathbf{p}^* \tag{7.14}$$

Norma kvaterniona je kvadratni koren iz zmnožka kvaterniona z njegovo konjugirano vrednostjo.

$$\|\mathbf{q}\| = \sqrt{\mathbf{q}\mathbf{q}^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{7.15}$$

Enotski kvaternion je kvaternion z normo 1. Inverzno vrednost kvaterniona določimo s pomočjo konjugirane vrednosti in norme

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} \tag{7.16}$$

Po odkritju matrik so kvaternioni šli v pozabo, danes pa jih uporabljamo v 3D računalniški grafiki in pri izračunu 3D rotacij.

Prostorske rotacije parametriramo z enotskimi kvaternioni tako, da definiramo kvaternion zasuka

$$\begin{aligned} q_0 &= \cos \Delta\Phi/2 \\ q_1 &= e_1 \sin \Delta\Phi/2 \\ q_2 &= e_2 \sin \Delta\Phi/2 \\ q_3 &= e3 \sin \Delta\Phi/2 \end{aligned} \tag{7.17}$$

kjer je vektor $\mathbf{e} = [e_1,\ e_2,\ e_3]$ enotski vektor osi vrtenja in $\Delta\Phi$ kot zasuka. Seveda velja

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{7.18}$$

Transformacija

$$\mathbf{v}' = \mathbf{q}^{-1}\mathbf{v}\mathbf{q} \tag{7.19}$$

zavrti vektor, ki je podan s kvaternionom

$$\mathbf{v} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{7.20}$$

okrog osi $\mathbf{e}$ za kot $\Delta\Phi$ v vektor

$$\mathbf{v}' = x'\mathbf{i} + y'\mathbf{j} + z'\mathbf{k}. \tag{7.21}$$

Pri enostkih kvaternionih je inverz enak konjugirani vrenosti. Iz en. (7.17) vidimo, da predstavlja konjugacija vrtenje v nasprotno smer. Interpretacija vrtenja je v tem delu povzeta po [22]; v literaturi (npr. v Wikipediji) se pojavljajo tudi interpretacije, kjer so kvaternioni konjugirani.

Največkrat nas zanima le orientacija, zato uporabimo parametrizacijo zasuka glede na nek osnovni zasuk samo s kvaternion zasuka (7.17).

Eulerjevi simetrični parametri (kvaterninoi) so zelo priročna parametrizacija orientacije satelita. So bolj strnjeni kot matrika DCM - le štirje parametri namesto devetih. So tudi primernejši od Eulerjevih kotov (tudi če spregledamo singularnost Eulerjevih kotov pri $\theta = \pm\pi/2$), saj ne vsebujejo kotnih funkcij, ki predstavljajo numerično zahtevne operacije.

Naslednja prednost kvaternionov je relativno enostavna oblika njihove kombinacije pri zaporednih vrtenjih. Kvaternion, ki ustreza produktu dveh matrik DCM, je enostavno produkt obeh kvaternionov [22]. Torej če sta

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{7.22}$$

in

$$\mathbf{q}' = q'_0 + q'_1\mathbf{i} + q'_2\mathbf{j} + q'_3\mathbf{k} \tag{7.23}$$

ter zavrtimo nek vektor iz osnovnega zasuka najprej za zasuk definiran s $\mathbf{q}'$ nato pa še za zasuk definiran s $\mathbf{q}$, potem je

$$\begin{aligned} \mathbf{q}'' = \mathbf{q}'\mathbf{q} &= (q'_0q_0 - q'_1q_1 - q'_2q_2 - q'_3q_3) \\ &+ \mathbf{i}(q'_1q_0 + q'_2q_3 - q'_3q_2 + q'_0q_1) \\ &+ \mathbf{j}(-q'_1q_3 + q'_2q_0 + q'_3q_1 + q'_0q_2) \\ &+ \mathbf{k}(q'_1q_2 - q'_2q_1 + q'_3q_0 + q'_0q_3) \end{aligned} \tag{7.24}$$

oziroma v vektorski - matrični obliki

$$\begin{bmatrix} q''_0 \\ q''_1 \\ q''_2 \\ q''_3 \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \begin{bmatrix} q'_0 \\ q'_1 \\ q'_2 \\ q'_3 \end{bmatrix} \tag{7.25}$$

za kar je potrebno 16 operacij množenja in 12 operacij seštevanja oz. odštevanja. Za primerjavo, za izračun zaporednih rotacij s transformacijskimi matrikami $\mathbf{R}$ je za zaporedno vrtenje potrebno 27 operacij množenja in 18 operacij seštevanja oz. odštevanja.

Povezava med kvaternioni in matriko DCM je naslednja:

$$R(\mathbf{q}) = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_0q_3+q_1q_2) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_3q_0) & +q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_0q_1+q_2q_3) \\ 2(q_0q_2+q_1q_3) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \quad (7.26)$$

oziroma

$$q_0 = \pm\frac{1}{2}\sqrt{1+R_{11}+R_{22}+R_{33}} \quad (7.27)$$

$$q_1 = \frac{1}{4q_0}\big(R_{23}-R_{32}\big) \quad (7.28)$$

$$q_2 = \frac{1}{4q_0}\big(R_{31}-R_{13}\big) \quad (7.29)$$

$$q_1 = \frac{1}{4q_0}\big(R_{12}-R_{21}\big) \quad (7.30)$$

Povezavo med kvaternioni in Eulerjevimi koti (vrstni red rotacij 3-2-1) pa dobimo na naslednji način: Matrikam $R_x(\phi)$, $R_y(\theta)$ in $R_z(\psi)$ ustrezajo kvaternioni $[\cos(\phi/2)+\mathbf{i}\sin(\phi/2)]$, $[\cos(\theta/2)+\mathbf{j}\sin(\theta/2)]$ in $[\cos(\psi/2)+\mathbf{k}\sin(\psi/2)]$. Kvaternion rotacije 3-2-1 je

$$\mathbf{q} = [\cos(\psi/2)+\mathbf{k}\sin(\psi/2)][\cos(\theta/2)+\mathbf{j}\sin(\theta/2)][\cos(\phi/2)+\mathbf{i}\sin(\phi/2)] \quad (7.31)$$

oziroma v vektorski obliki

$$\mathbf{q} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2)+\sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2)-\cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2)+\sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2)-\sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix} \quad (7.32)$$

Obratna transformacija se glasi

$$\phi = \arctan\left(\frac{2(q_1q_0+q_2q_3)}{q_0^2-q_1^2-q_2^2+q_3^2}\right) \quad (7.33)$$

$$\theta = -\arcsin(2(q_1q_3-q_2q_0)) \quad (7.34)$$

$$\psi = \arctan\left(\frac{2(q_3q_0+q_1q_2)}{q_0^2+q_1^2-q_2^2-q_3^2}\right) \quad (7.35)$$

Primer 7.1. *Imamo referenčni koordinatni sistem (G) in koordinatni sistem togega telesa (L). Na začetku sta oba poravnana, nato togo telo zavrtimo okoli osi x osi za kot $\alpha_x = 90°$ in nato še enkrat okoli nove osi z za kot $\alpha_z = 90°$. Odgovorite na vprašanja:*

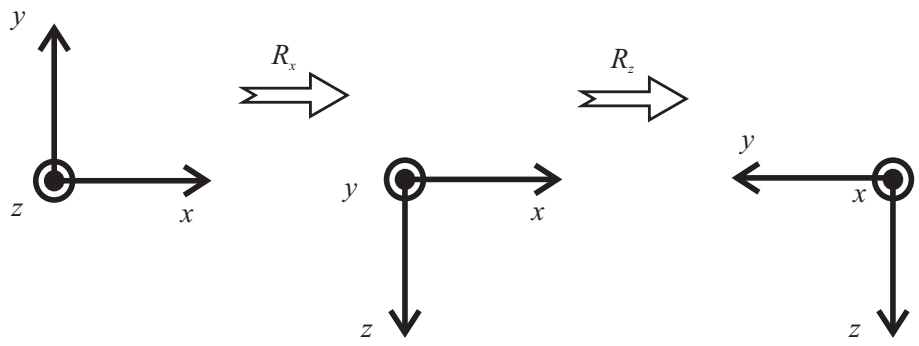
1. *Kakšna je orientacija togega telesa (DCM matrika $\mathbf{R}_G^L$) glede na referenčni koordinatni sistem.*
2. *Določite Eulerjeve kote (notacija 3-2-1), ki opisujejo to transformacijo.*
3. *Določite kvaternion $\mathbf{q}_G^L$, ki opisuje to transformacijo.*
4. *Kakšna je predstavitev vektorja $\mathbf{v} = [0, 0, 1]^T$, podanega v referenčnih koordinatah, v koordinatah togega telesa?*

Rešitev

1. *Končno orientacijo opisuje skupna matrika rotacije, kjer je pomemben vrstni red množenj.*

$$\mathbf{R}_G^L = \mathbf{R}_z(\alpha_z)\mathbf{R}_x(\alpha_x) =$$
$$= \begin{bmatrix} \cos(\alpha_z) & \sin(\alpha_z) & 0 \\ -\sin(\alpha_z) & \cos(\alpha_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_x) & \sin(\alpha_x) \\ 0 & -\sin(\alpha_x) & \cos(\alpha_x) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Vrstice v matriki rotacije $\mathbf{R}_G^L$ predstavljajo komponente novih osi togega telesa izražene v referenčnem koordinatnem sistemu.



2. *Eulerjeve kote za notacijo 3-2-1 ocenimo iz matrike $\mathbf{R} = \mathbf{R}_G^L$*

$$\phi = \arctan(\tfrac{R_{23}}{R_{33}}) = NAN$$
$$\theta = -\arcsin(R_{13}) = -90°$$
$$\psi = \arctan(\tfrac{R_{12}}{R_{11}}) = NAN$$

Opazimo, da je $\theta = -90°$, kar pomeni, da je parametrizacija z Eulerjevimi koti singularna in sta zato ϕ in ψ nedefinirana. Z Eulerjevimi koti torej te orientacije ne moremo zapisati.

3. *Kvaternion $\mathbf{q}_G^L$, ki opisuje to transformacijo dobimo s produktom kvaternionov posameznih rotacij.*

$$\mathbf{q}_G^L = \mathbf{q}_x \mathbf{q}_z$$

kjer je $\mathbf{q}_x$ glede na enačbo (7.17) definiran s kotom vrtenja $\Delta\Phi_x = 90°$ okoli osi vrtenja $\mathbf{e}_x = [1,0,0]^T$ in $\mathbf{q}_z$ s kotom vrtenja $\Delta\Phi_z = 90°$ okoli osi vrtenja $\mathbf{e}_z = [0,0,1]^T$.

$$\mathbf{q}_x = \begin{bmatrix} \cos\Delta\Phi_x/2 \\ e_{x1}\sin\Delta\Phi_x/2 \\ e_{x2}\sin\Delta\Phi_x/2 \\ e_{x3}\sin\Delta\Phi_x/2 \end{bmatrix} = \begin{bmatrix} 0.7071 \\ 0.7071 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{q}_z = \begin{bmatrix} \cos\Delta\Phi_z/2 \\ e_{z1}\sin\Delta\Phi_z/2 \\ e_{z2}\sin\Delta\Phi_z/2 \\ e_{z3}\sin\Delta\Phi_z/2 \end{bmatrix} = \begin{bmatrix} 0.7071 \\ 0 \\ 0 \\ 0.7071 \end{bmatrix}$$

končni kvaternion je (produkt kvaternionov je definiran v (7.25))

$$\mathbf{q}_G^L = \mathbf{q}_x \mathbf{q}_z = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ 0.5 \end{bmatrix}$$

ker ustreza vrtenju za kot $\Phi = 2\arccos(0.5) = 120°$ okoli osi vrtenja $\mathbf{e} = \frac{1}{\sin\frac{\Delta\Phi}{2}}[q_1, q_2, q_3]^T = [0.5774, -0.5774, 0.5774]^T$

4. *Vektor $\mathbf{v}_G = [0,0,1]^T$ je v koordinatah togega telesa izražen kot*

$$\mathbf{v}_L = \mathbf{R}_G^L \mathbf{v}_G = [1,0,0]^T$$

kar dobimo tudi s kvaternioni, kjer je vektor po rotaciji zapisan s qvaternionom

$$\mathbf{q}_{v_L} = {\mathbf{q}_G^L}^{-1} \mathbf{q}_{v_G} \mathbf{q}_G^L$$

in je $\mathbf{q}_{v_G} = [0, \mathbf{v}_G^T]^T$, inverz kvaterniona $\mathbf{q}_G^L$ je definiran v relaciji (7.16) in produkt med kvaternioni z (7.25). Dobimo

$$\mathbf{q}_{v_L} = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

kar ustreza vektorju $\mathbf{v}_L = [1, 0, 0]^T$ *(skalarni del qvaterniona pri zapisu vektorja nima pomena, zanima nas le vektorski del).*

Primer 7.2. *Imamo referenčni koordinatni sistem (G) in koordinatni sistem togega telesa (L). Na začetku sta oba poravnana, nato togo telo zavrtimo okoli osi x osi za kot* $\alpha_x = 90°$ *in nato še enkrat okoli nove osi y za kot* $\alpha_y = 45°$*. Odgovorite na vprašanja:*

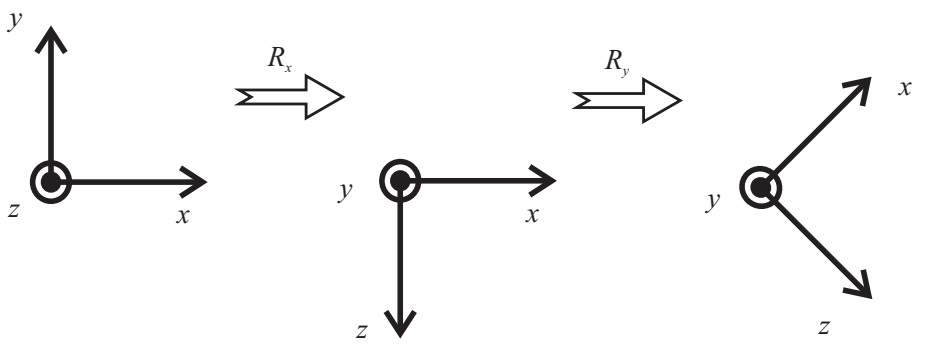
1. *Kakšna je orientacija togega telesa (DCM matrika* $\mathbf{R}_G^L$*) glede na referenčni koordinatni sistem.*
2. *Določite Eulerjeve kote (notacija 3-2-1), ki opisujejo to transformacijo.*
3. *Določite kvaternion* $\mathbf{q}_G^L$*, ki opisuje to transformacijo.*
4. *Kakšna je predstavitev vektorja* $\mathbf{v} = [0, 0, 1]^T$*, podanega v referenčnih koordinatah, v koordinatah togega telesa?*

Rešitev

1. *Končno orientacijo opisuje skupna matrika rotacije, kjer je pomemben vrstni red množenj.*

$$\mathbf{R}_G^L = \mathbf{R}_y(\alpha_y)\mathbf{R}_x(\alpha_x) =$$
$$= \begin{bmatrix} \cos(\alpha_y) & 0 & -\sin(\alpha_y) \\ 0 & 1 & 0 \\ \sin(\alpha_y) & 0 & \cos(\alpha_y) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_x) & \sin(\alpha_x) \\ 0 & -\sin(\alpha_x) & \cos(\alpha_x) \end{bmatrix} = \begin{bmatrix} 0.7071 & 0.7071 & 1 \\ 0 & 0 & 1 \\ 0.7071 & -0.7071 & 0 \end{bmatrix}$$

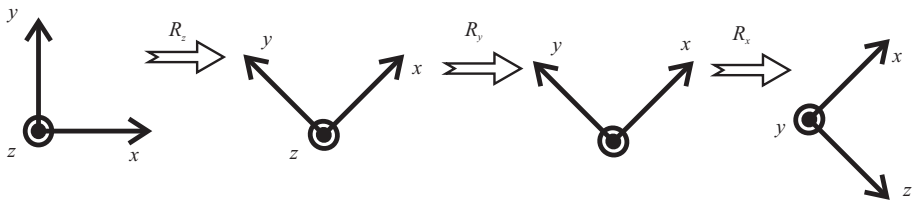
Vrstice v matriki rotacije $\mathbf{R}_G^L$ *predstavljajo komponente novih osi togega telesa izražene v referenčnem koordinatnem sistemu.*



2. *Eulerjeve kote za notacijo 3-2-1 ocenimo iz matrike* $\mathbf{R} = \mathbf{R}_G^L$

$$\begin{aligned}\phi &= \arctan(\tfrac{R_{23}}{R_{33}}) = 90^\circ \\ \theta &= -\arcsin(R_{13}) = -0^\circ \\ \psi &= \arctan(\tfrac{R_{12}}{R_{11}}) = 45^\circ\end{aligned}$$

kjer matriko rotacije dobimo z $\mathbf{R}_G^L = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}z(\psi)$, *kjer so matrike* $R_x(\phi)$, $R_y(\theta)$ *in* $Rz(\psi)$ *definirane v enačbi (7.9).*



3. *Kvaternion* $\mathbf{q}_G^L$, *ki opisuje to transformacijo dobimo s produktom kvaternionov posameznih rotacij.*

$$\mathbf{q}_G^L = \mathbf{q}_x\mathbf{q}_y$$

kjer je $\mathbf{q}_x$ *glede na enačbo (7.17) definiran s kotom vrtenja* $\Delta\Phi_x = 90^\circ$ *okoli osi vrtenja* $\mathbf{e}_x = [1,0,0]^T$ *in* $\mathbf{q}_y$ *s kotom vrtenja* $\Delta\Phi_y = 45^\circ$ *okoli osi vrtenja* $\mathbf{e}_z = [0,1,0]^T$.

$$\mathbf{q}_x = \begin{bmatrix} \cos\Delta\Phi_x/2 \\ e_{x1}\sin\Delta\Phi_x/2 \\ e_{x2}\sin\Delta\Phi_x/2 \\ e_{x3}\sin\Delta\Phi_x/2 \end{bmatrix} = \begin{bmatrix} 0.7071 \\ 0.7071 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{q}_y = \begin{bmatrix} \cos\Delta\Phi_z/2 \\ e_{y1}\sin\Delta\Phi_y/2 \\ e_{y2}\sin\Delta\Phi_y/2 \\ e_{y3}\sin\Delta\Phi_y/2 \end{bmatrix} = \begin{bmatrix} 0.9239 \\ 0 \\ 0.3827 \\ 0 \end{bmatrix}$$

končni kvaternion je (produkt kvaternionov je definiran v (7.25))

$$\mathbf{q}_G^L = \mathbf{q}_x\mathbf{q}_y = \begin{bmatrix} 0.6533 \\ 0.6533 \\ 0.2706 \\ 0.2706 \end{bmatrix}$$

ker ustreza vrtenju za kot $\Phi = 2\arccos(0.5) = 98.41^\circ$ *okoli osi vrtenja* $\mathbf{e} = \frac{1}{\sin\frac{\Delta\Phi}{2}}[q_1,q_2,q_3]^T = [0.8630, 0.3574, 0.3574]^T$

4. *Vektor* $\mathbf{v}_G = [0,0,1]^T$ *je v koordinatah togega telesa izražen kot*

$$\mathbf{v}_L = \mathbf{R}_G^L \mathbf{v}_G = [0,1,0]^T$$

kar dobimo tudi s kvaternioni, kjer je vektor po rotaciji zapisan s qvaternionom

$$\mathbf{q}_{v_L} = \mathbf{q}_G^{L\,-1} \mathbf{q}_{v_G} \mathbf{q}_G^L$$

in je $\mathbf{q}_{v_G} = [0, \mathbf{v}_G^T]^T$*, inverz kvaterniona* $\mathbf{q}_G^L$ *je definiran v relaciji (7.16) in produkt med kvaternioni z (7.25). Dobimo*

$$\mathbf{q}_{v_L} = \begin{bmatrix} 0.6533 \\ -0.6533 \\ -0.2706 \\ -0.2706 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0.6533 \\ 0.6533 \\ 0.2706 \\ 0.2706 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

kar ustreza vektorju $\mathbf{v}_L = [0,1,0]^T$ *(skalarni del qvaterniona pri zapisu vektorja nima pomena, zanima nas le vektorski del).*

7.5 Pretvorbe med koordinatnimi sistemi

Pretvorbe med posameznimi koordinatnimi sistemi vršimo s pomočjo rotacijskih matrik DCM. Transformacije med posameznimi koordinatnimi sistemi podamo z matriko R_{iz}^v, kjer iz pomeni izhodiščni koordinatni sistem, v pa ciljnega.

Transformacija med ECEF in ECI

ECEF koordinatni sistem se glede na ECI sistem vrti s kotno hitrostjo $\omega_0 = 7,2921 \times 10^{-5}\ rad/s$. Koordinatna sistema ECEF in ECI sta enkrat na dan - ob zvezdnem času osnovnega poldnevnika (Greenwich Mean Sidereal Time) 0 - popolnoma poravnana po vseh oseh. Transformacija je podana z

$$R_{ECEF}^{ECI} = R_z(-\Theta) \quad (7.36)$$
$$R_{ECI}^{ECEF} = R_z(\Theta) \quad (7.37)$$

kjer je $\Theta = \omega_0 t$ in je t zvezdni čas ničtega poldnevnika (GST=Greenwich Siderial Time)

Transformacija med ECO in ECI

Sistema ECI in ECO imata isto izhodišče v središču Zemlje, vendar se razlikujeta v tem, da je ECI orientiran glede na oddaljene zvezde, ECO pa glede na tirnico opazovanega satelita. Kot je opisano v [23] dobimo ECO sistem tako, da zavrtimo ECI sistem najprej za rektascenzijo dvižnega vozla Ω okrog osi z, nato okrog nove osi x za naklon tirnice (inklinacija i) in nato zopet okrog nove osi z za argument perigeja ω Rotacijski matriki sta tako:

$$R_{ECO}^{ECI} = R_z(-\Omega)R_x(-i)R_z(-\omega) \quad (7.38)$$
$$R_{ECI}^{ECO} = R_z(\omega)R_x(i)R_z(\Omega) \quad (7.39)$$

Transformacija med ECO in O

O je edini od referenčnih koordinatnih sistemov, ki nima izhodišča v središču Zemlje in je odvisen tako od tirnice opazovanega satelita kot njegovega trenutnega položaja. Za določitev orientacije O glede na ECO tako potrebujemo pravo anomalijo satelita φ - Keplerjev element, ki določa trenuten položaj satelita v svoji tirnici. Pri tej transformaciji moramo satelit zavrteti tako, da kaže z os proti središču zemlje, x os pa v smeri gibanja. To dosežemo z vrtenjem okrog osi z za $\frac{\pi}{2}$ in nato okrog nove osi x za $-\frac{\pi}{2}$. Satelit je potrebno zavrteti okrog osi z tudi za pravo anomalijo φ. Rotacijski matriki sta tako:

$$R_{ECO}^{O} = R_x(-\frac{\pi}{2})R_z(\varphi + \frac{\pi}{2}) \quad (7.40)$$
$$R_{O}^{ECO} = R_z(-\varphi - \frac{\pi}{2})R_x(\frac{\pi}{2}) \quad (7.41)$$

Transformacija med ECO in LVLH

Pri tej transformaciji moramo satelit zavrteti tako, da kaže x os v smer, nasprotni proti središču zemlje, z os pa v smeri gibanja. To dosežemo z vrtenjem okrog osi z za pravo anomalijo φ. Rotacijski matriki sta tako:

$$R_{ECO}^{LVLH} = R_z(\varphi) \quad (7.42)$$
$$R_{LVLH}^{ECO} = R_z(-\varphi) \quad (7.43)$$

Transformacija med O in LVLH

Kot omenjeno v razdelku 7.3 je zasuk koordinatnih sistemov O in LVLH (drug proti drugemu) konstanten. Ustrezni transformacijski matriki sta

$$R_O^{LVLH} = \begin{pmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \tag{7.44}$$

$$R_{LVLH}^O = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix} \tag{7.45}$$

Povezava med B in ECI

Orientacijo satelita glede na inercialni koordinatni sistem določata dinamika in kinematika satelitov, ki ju obravnavata poglavji 7.6 in 7.7.

7.5.1 Keplerjevi elementi tirnic

Keplerjevi elementi tirnic ([18], [19], [23], [20]), imenovani po astronomu Johannesu Keplerju, izhajajo iz njegovih zakonov o gibanju planetov. Keplerjevi zakoni so dobljeni eksperimentalno in primarno opisujejo gibanje planetov okoli Sonca. Izkazalo se je, da so Keplerjevi zakoni splošno uporabni za opis kroženja lažjega telesa okoli mnogo težjega telesa (npr. satelita okoli Zemlje).

Pozicijo satelita na tirnici lahko določimo iz Keplerjevih elementov, ki so podani za določeno točko ob določenem času (Epoch). Če poznamo Keplerjevih elemente orbite za nek čas, lahko ocenimo predikcijo pozicije satelita v prihodnosti.

Šest Keplerjevih elementov enoznačno določa tirnico satelita [22] (slika 7.9):

- *Naklon tirnice* i (inclination) je kot med ravnino tirnice in ravnino Zemljinega ekvatorja. Presečišče obeh ravnin (ekvatorialne in orbitne) definira linijo vozlov, dvižni vozel (tirnica se dviga nad ekvatorialno ravnino) in spustni vozel (tirnica gre pod ekvatorialno ravnino). Naklon $i = 0°$ pomeni ekvatorialno tirnico, po kateri satelit kroži proti vzhodu, naklon $i = 90°$ polarno tirnico in naklon $i = 180°$ retrogradno tirnico - kroženje proti zahodu.

- *Rektascenzija dvižnega vozla* Ω_0 (Right Ascension of Ascending Node (RAAN)) je kot med pomladiščem in točko v tirnici, kjer satelit prečka

ravnino ekvatorja tako, da preide iz južne v severno poloblo (dvižni vozel).

- *Velika polos a* (semimajor axis) opisuje velikost elipse in predstavlja polovico razdalje med periapsido in apoapsido. Periapsida je točka na elipsi (tirnici), ki je najbližje gorišču elipse, v katerem se nahaja Zemlja. Točka na elipsi najbolj oddaljena od Zemlje pa je apoapsida.

- *Ekscentričnost* ε (eccentricity), podaja obliko elipse in je vrednost med 0 in 1 definirana z $\varepsilon = \sqrt{1 - \frac{b^2}{a^2}}$, kjer je b mala polos elipse.

- *Argument periapside* ω_0 (argument of perigee) je kot med smerjo proti dvižnemu vozlu in smerjo proti periapsidi (točka, kjer je telo najbliže gorišču v katerem se nahaja Zemlja). Kot merimo v smeri premikanja satelita po tirnici.

- *Srednja anomalija v epohi* M_0 (mean anomaly at epoh) definira kje na elipsi je pozicioniran satelit. M_0 je kot z vrednostmi 0-360° (0° je v periapsidi in 180° v apoapsidi). Apoapsida je točka na elipsi, ki je najdlje od Zemlje.

  V primeru nekrožnih elips M_0 definira lego satelita na tirnici le v periapsidi in apoapsidi, saj satelit nima konstantne kotne hitrosti. V tem kontekstu so definirane še ekscentrična anomalija E in prava anomalija φ. Prava anomalija podaja dejanski kot od periapside do satelita.

TLE - standardni format zapisa elementov tirnic satelitov

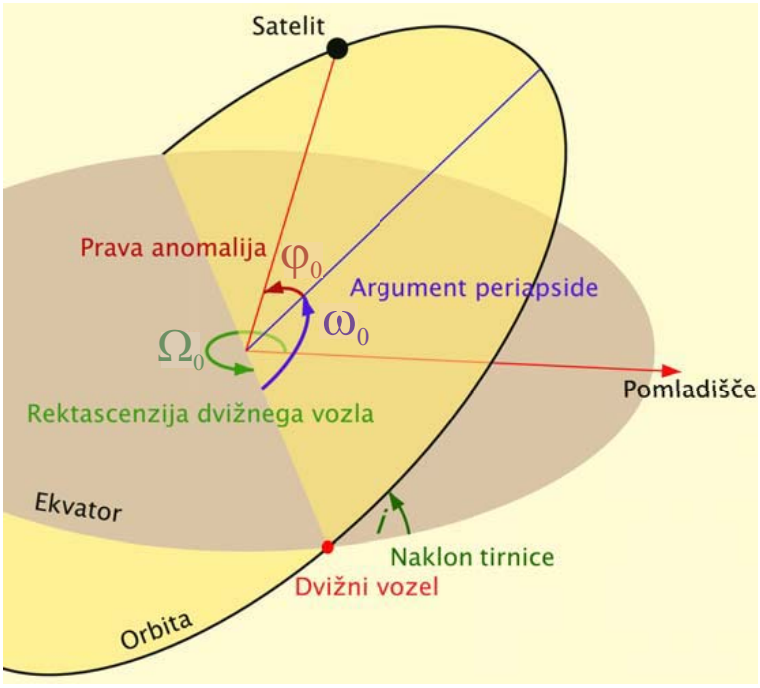
TLE oz. Two Line Element-set je standardni format za računalniški zapis elementov tirnic satelitov. Sestoji iz naslovne vrstice in dveh vrstic s predpisanim vrstnim redom zapisa. Zapis TLE na primeru tirnice satelita "Iridium-33":

```
IRIDIUM 33
 1 24946U 97051C 13215.44831351 .00000191 00000-0 62768-4 0 5336
 2 24946 86.3935 158.1918 0009163 154.3465 344.2201 14.32933672831537
```

Obrazložitev oznak [19]:

- 1 - oznaka prve vrstice.

- 24946 -unikatna številčna oznaka satelita.



Slika 7.9: Keplerjevi elementi tirnice satelita.

- U - kategorija objekta (U - unclassified).
- 97 - zadnji dve cifri leta izstrelitve (1997).
- 051 - raketa, na kateri je bil satelit izstreljen (51. izstrelitev v letu 1997).
- C - oznaka tovora, če je na raketi več kot en satelit.
- 13 - zadnji dve cifri trenutnega leta (2013).
- 215.44831351 - trenuten dan leta, del za decimalnim ločilom pomeni čas dneva.
- .00000191 - balistični koeficient - prvi časovni odvod števila tirnic, ki jih satelit opravi v enem dnevu, deljen z 2.
- 00000-0 - drugi časovni odvod števila tirnic, ki jih satelit opravi e enem dnevu, deljen z 6.
- 62768-4 - koeficient, povezan z motnjo tirnice satelita s strani sevalnega tlaka sonca.

- 0 - tip zapisa, 0 je edini uporabljen
- 533 - zaporedje elementov TLE - število se poveča za 1, vsakič, ko se za satelit generira nov TLE.
- 6 - kontrolna vsota vrstice po modulu 10.
- 2 - oznaka druge vrstice
- 24946 - unikatna številčna oznaka satelita.
- 86.3935 - naklon tirnice v stopinjah
- 158.1918 - rektascenzija dvižnega vozla v stopinjah
- 0009163 - ekscentricnost (predpostavljena decimalna vejica, dejanska vrednost 0.0009163)
- 154.3465 - argument periapside v stopinjah
- 344.2201 - srednja anomalija v stopinjah
- 14.3293367283153 - število tirnic, ki jih satelit opravi v enem dnevu
- 7 - kontrolna vsota vrstice po modulu 10.

Izračun pozicije in vektorja hitrosti satelita iz Keplerjevih elementov

Keplerjevimi elementi opisujejo lego satelita na tirnici v določenem časovnem trenutku (epohi). Obstaja več različnih algoritmov za izračun pozicije satelita v orbiti v določenem času. V nadaljevanju podajamo SGP4 (Simplified General Perturbations Version 4) model [24].

Iz podanih Keplerjevih elementov (iz TLE dvovrstičnega zapis) in podano epoho t_0 (začetni čas) ter $\frac{1}{2}\frac{dn}{dt}$ (n je povprečno število obkrožitev tirnice na dan) določimo pozicijo satelita ob času t kot sledi:

1. Izračunamo povprečno število dnevnih obkrožitev n, ki ga izrazimo iz izraza za veliko polos a

$$a = \sqrt[3]{\frac{\mu}{n^2}} \qquad \mu = 3.986005 \cdot 10^{14} \tag{7.46}$$

2. Izračunamo povprečno anomalijo M pri času t

$$M = M_0 + n(t - t_0) + \frac{1}{2}\frac{dn}{dt}(t - t_0)^2 \tag{7.47}$$

3. Določimo ekscentrično anomalijo E iz povprečne anomalije M upoštevajoč Keplerjevo enačbo

$$M = E - \varepsilon \sin(E) \tag{7.48}$$

ki jo lahko rešimo z uporabo Newtnovo iteracijske metode kjer iščemo ničlo funkcije $f(E) = E - \varepsilon \sin E - M$ in je iterativna rešitev

$$E_{i+1} = E_i - \left[\frac{f(E)}{\frac{df(E)}{dE}}\right]_{E=E_i} = E_i - \frac{E_i - M - \varepsilon \sin E_i}{1 - \varepsilon \cos E_i} \tag{7.49}$$

kjer za začetno oceno vzamemo $E_0 = M$.

4. Določimo pravo anomalijo φ

$$\varphi = \arctan\left[\frac{\sin(E)\sqrt{1-\varepsilon^2}}{\cos(E) - \varepsilon}\right] \tag{7.50}$$

5. Določimo dejanski argument periapside ω in RAAN Ω upoštevajoč geopotencialni coefficient $J_2 = 1.08263\ 10^{-3}$

$$\omega = \omega_0 + \frac{d\omega}{dt}(t - t_0) \tag{7.51}$$

$$\Omega = \Omega_0 + \frac{d\Omega}{dt}(t - t_0) \tag{7.52}$$

kjer

$$\frac{d\omega}{dt} = \frac{3}{4} n \left(\frac{a_E}{a}\right)^2 \frac{5\cos^2 i - 1}{(1-\varepsilon^2)^2} J_2 \tag{7.53}$$

$$\frac{d\Omega}{dt} = -\frac{3}{2} n \left(\frac{a_E}{a}\right)^2 \frac{\cos i}{(1-\varepsilon^2)^2} J_2 \tag{7.54}$$

in $a_E = 6378 \cdot 10^3$ je velika polos zemeljskega elipsoida.

6. Izračunamo pozicijo satelita v ECO koordinatnem sistemu

$$\mathbf{P}_{ECO} = \begin{bmatrix} \frac{a(1-\varepsilon^2)\cos(\varphi)}{1+\varepsilon\cos(\varphi)} \\ \frac{a(1-\varepsilon^2)\sin(\varphi)}{1+\varepsilon\cos(\varphi)} \\ 0 \end{bmatrix} \tag{7.55}$$

7. Transformiramo ECO pozicijo satelita v pozicijo v ECI kordinatah

$$\mathbf{P}_{ECI} = \mathbf{R}_{ECO}^{ECI}\mathbf{P}_{ECO} \tag{7.56}$$

kjer

$$\begin{aligned}&\mathbf{R}_{ECO}^{ECI} = \\ &= \begin{bmatrix} c(-\Omega) & s(-\Omega) & 0 \\ -s(-\Omega) & c(-\Omega) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(-i) & s(-i) \\ 0 & -s(-i) & c(-i) \end{bmatrix} \cdot \\ &\cdot \begin{bmatrix} c(-\omega) & s(-\omega) & 0 \\ -s(-\omega) & c(-\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix}\end{aligned}$$

in c označuje cos in s označuje sin.

7.6 Kinematika orientacije satelitov

Kinematika gibanja togega telesa (satelita) bomo obravnavali s pomočjo parametrizacije s kvaternioni, podali pa bomo tudi enačbe za parametrizacijo z Eulerjevimi koti.

7.6.1 Parametrizacija kinematike s kvaternioni

Časovno odvisnost (diferencialno enačbo) za vrtenje togega telesa najenostavneje izvedemo iz relacije za produkt dveh kvaternionov (7.25). Če je orientacija togega telesa $\mathbf{q}(t)$ v času t znana, potem lahko orientacijo togega telesa $\mathbf{q}(t + \Delta t)$ v času $t + \Delta t$ zapišemo kot

$$\mathbf{q}(t + \Delta t) = \mathbf{q}(t)\Delta\mathbf{q}(t) \tag{7.57}$$

kjer $\Delta\mathbf{q}(t)$ podaja spremembo rotacije iz $\mathbf{q}(t)$ v $\mathbf{q}(t + \Delta t)$ oziroma je to orientacija telesa v času $t + \Delta t$ relativno na njegovo orientacijo v času t. Torej do končne orientacije telesa $\mathbf{q}(t + \Delta t)$ pridemo tako, da telo glede na nek referenčni koordinatni sistem zavrtimo za rotacijo $\mathbf{q}(t)$ in nato še za rotacijo $\Delta\mathbf{q}(t)$ glede na $\mathbf{q}(t)$. $\Delta\mathbf{q}(t)$ je določena z enačbo (7.17)

$$\Delta\mathbf{q}(t) = \begin{bmatrix} \cos\Delta\Phi/2 \\ e_x \sin\Delta\Phi/2 \\ e_y \sin\Delta\Phi/2 \\ e_z \sin\Delta\Phi/2 \end{bmatrix} \tag{7.58}$$

kjer je $\mathbf{e}(t) = [\ e_x \quad e_y \quad e_z\ ]^T$ vektor rotacije podan v koordinatnem sistemu togega telesa (satelita) v trenutku t in $\Delta\Phi$ kot zasuku v času Δt. Če predpostavimo, da sta $\mathbf{e}(t)$ in $\Delta\Phi$ konstantna med časom rotacije Δt, lahko produkt kvaternionov (7.57) glede definicijo produkta kvaternionov (7.25) (matrična oblika produkta) zapišemo kot

$$\mathbf{q}(t+\Delta t) = \left\{ \cos\left(\frac{\Delta\Phi}{2}\right)\mathbf{I} + \sin\left(\frac{\Delta\Phi}{2}\right) \begin{bmatrix} 0 & -e_x & -e_y & -e_z \\ e_x & 0 & e_z & -e_y \\ e_y & -e_z & 0 & e_x \\ e_z & e_y & -e_x & 0 \end{bmatrix} \right\} \mathbf{q}(t) \tag{7.59}$$

kjer je I enotirna matrika reda 4×4. Za kratek časovni interval Δt lahko $\Delta\Phi$ aproksimiramo z $\Delta\Phi = \sqrt{{\omega_x}^2 + {\omega_y}^2 + {\omega_z}^2}\Delta t$, kjer je $\boldsymbol{\omega}(t) = [\ \omega_x \quad \omega_y \quad \omega_z\ ]^T$ vektor trenutnih kotnih hitrosti vrtenja satelita, nadalje velja $\boldsymbol{\omega}(t) = \sqrt{{\omega_x}^2 + {\omega_y}^2 + {\omega_z}^2}\mathbf{e}$. Za majhne kote lahko enačbo (7.59) aproksimiramo z

$$\mathbf{q}(t+\Delta t) = \left(1 + \frac{\Delta t\boldsymbol{\Omega}}{2}\right)\mathbf{q}(t) \tag{7.60}$$

kjer je

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{7.61}$$

Diferencialno enačbo dobimo z limitiranjem časovnega intervala Δt proti nič

$$\frac{d\mathbf{q}}{dt} = \lim_{\Delta t\to 0}\frac{\mathbf{q}(t+\Delta t) - \mathbf{q}(t)}{\Delta t} = \frac{1}{2}\boldsymbol{\Omega}\mathbf{q} \tag{7.62}$$

Potrebno je podariti, da so kotne hitrosti podane v koordinatnem sistemu togega telesa.

7.6.2 Parametrizacija kinematike z Eulerjevimi koti

Izpeljimo še diferencialno enačbo za orientacino podano z DCM matriko. Podobno kot zgoraj lahko zapišemo

$$\mathbf{R}(t+\Delta t) = \Delta\mathbf{R}(t)\mathbf{R}(t) \tag{7.63}$$

kjer je $\mathbf{R}(t)$ orientacija togega telesa v času t, $\mathbf{R}(t+\Delta t)$ orientacija togega telesa v času $t+\Delta t$ in $\Delta\mathbf{R}(t)$ spremembo rotacije oziroma je to orientacija telesa pri $t = t+\Delta t$ glede na orientacijo čas v času t.

Spremembo orientacije $\Delta\mathbf{R}(t)$ lahko zapišemo kot

$$\Delta\mathbf{R}(t) = e^{\int_t^{t+\Delta t} \mathbf{\Omega}' dt} \tag{7.64}$$

kjer je

$$\mathbf{\Omega}' = \begin{pmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{pmatrix} \tag{7.65}$$

in je $\boldsymbol{\omega}(t) = [\ \omega_x \quad \omega_y \quad \omega_z\ ]^T$ vektor trenutnih kotnih hitrosti vrtenja satelita. Izraz $\int_t^{t+\Delta t} \Omega'(t)$ lahko ob predpostavki, da je $\mathbf{\Omega}'$ konstantna matrika v času Δt, aproksimiramo z $B = \mathbf{\Omega}'\Delta t$ in nadalje lahko eksponent v relaciji (7.64) razširimo v Taylorjevo vrsto in dobimo

$$\begin{aligned} &\Delta\mathbf{R}(t) = e^B = \\ &= \left(I + B + \tfrac{B^2}{2!} + \tfrac{B^3}{3!} + \cdots\right) = \\ &= \left(I + B + \tfrac{B^2}{2!} - \tfrac{\sigma^2 B}{3!} - \tfrac{\sigma^2 B^2}{4!} + \tfrac{\sigma^4 B}{5!} + \cdots\right) = \\ &= \left(I + \tfrac{\sin\sigma}{\sigma} B + \tfrac{1-\cos\sigma}{\sigma^2} B^2\right) \end{aligned} \tag{7.66}$$

kjer je I is enotina matrika reda 3×3 in $\sigma = \Delta t\sqrt{{\omega_x}^2 + {\omega_y}^2 + {\omega_z}^2}$. Za majhne kote σ lahko (7.66) aproksimiramo z

$$\Delta\mathbf{R}(t) = I + B = I + \Omega'\Delta t \tag{7.67}$$

torej lahko predikcijo rotacijske matrike (7.63) zapišemo kot

$$\mathbf{R}(t + \Delta t) = (I + \mathbf{\Omega}'\Delta t)\,\mathbf{R}(t) \tag{7.68}$$

Diferencialno enačbo dobimo z limitiranjem časovnega intervala Δt proti nič

$$\frac{d\mathbf{R}}{dt} = \lim_{\Delta t \to 0} \frac{\mathbf{R}(t + \Delta t) - \mathbf{R}(t)}{\Delta t} = \mathbf{\Omega}'\mathbf{R} \tag{7.69}$$

Navedimo še ekvivalentno enačbo pri parametrizaciji z Eulerjevimi koti [26]

$$\dot{\Phi} = \omega_x + \omega_y \sin\Phi \tan\theta + \omega_z \cos\Phi \tan\theta \tag{7.70}$$

$$\dot{\theta} = \omega_y \cos\Phi - \omega_z \sin\Phi \tag{7.71}$$

$$\dot{\Psi} = \frac{\omega_y \sin\Phi + \omega_z \cos\Phi}{\cos\theta} \tag{7.72}$$

Vidimo lahko, da postaneta prva in tretja enačba singularni pri $\theta = \pm\pi/2$.

7.7 Dinamika gibanja satelitov

Newtonov zakon pravi, da je odvod vrtilne količine enak vsoti momentov, ki učinkujejo na telo. Vrtilna količina je enaka produktu tenzorja vztrajnostnih momentov J in vektorja kotnih hitrosti $\boldsymbol{\omega}$, $\boldsymbol{\Gamma} = \mathbf{J}\boldsymbol{\omega}$.

J imenujemo tenzor, ker ima specifične transformacijske lastnosti, glej [22]. Tu bo dovolj, da ga zapišemo v obliki realne simetrične matrike reda 3×3. Realne simetrične matrike 3×3 imajo tri realne lastne vektorje in tri realne lastne vrednosti.

$$\mathbf{J}\boldsymbol{\theta}_i = \lambda_i \boldsymbol{\theta}_i \quad i = 1, 2, 3 \tag{7.73}$$

Lastne vrednosti predstavljajo glavne vztrajnostne momente in lastni vektorji tri glavne osi. Če uporabimo za osi koordinatnega sistema glavne osi (lastni vektorji realnih simetričnih matrik so med seboj ortogonalni), postane tenzor vztrajnostnih momentov diagonalna matrika.

Newtonovi zakoni seveda veljajo samo v inercialnih koordinatnih sistemih, zato lahko zapišemo

$$\mathbf{M}_{ECI} = \frac{d\boldsymbol{\Gamma}_{ECI}}{dt} \tag{7.74}$$

Za nadaljnjo obravnavo je smiselno izbrati tisti koordinatni sistem, v katerem je tenzor vztrajnostnih momentov konstanten - to pa je koordinatni sistem togega telesa. Zato zapišemo transformacijsko enačbo za vrtilni količini v koordinatnih sistemih togega telesa (B) in inercialnega sistema z izhodiščem v središču zemlje (ECI).

$$\boldsymbol{\Gamma}_B = \mathbf{R}^B_{ECI}\boldsymbol{\Gamma}_{ECI} \tag{7.75}$$

$\boldsymbol{\Gamma}_B$ je vektor definiran v vrtečem se koordinatnem sistemu (B), zato je pri določitvi njegovega odvoda potrebno upoštevati, da se spreminja tudi $\mathbf{R}^B_{ECI}$. Časovni odvod (7.75) je

$$\frac{d\boldsymbol{\Gamma}_B}{dt} = \frac{d\mathbf{R}^B_{ECI}}{dt}\boldsymbol{\Gamma}_{ECI} + \mathbf{R}^B_{ECI}\frac{d\boldsymbol{\Gamma}_{ECI}}{dt} \tag{7.76}$$

kjer upoštevajoč enačbo za odvod DCM matrike (7.69) in transformacijo (7.75) velja $\frac{d\mathbf{R}^B_{ECI}}{dt}\boldsymbol{\Gamma}_{ECI} = \boldsymbol{\Omega}'\mathbf{R}^B_{ECI}\boldsymbol{\Gamma}_{ECI} = \boldsymbol{\Omega}'\boldsymbol{\Gamma}_B$ in enačbo (7.76) zapišemo kot

$$\frac{d\boldsymbol{\Gamma}_B}{dt} = \boldsymbol{\Omega}'\boldsymbol{\Gamma}_B + \mathbf{R}^B_{ECI}\frac{d\boldsymbol{\Gamma}_{ECI}}{dt} \tag{7.77}$$

ker je $\mathbf{J}$ konstantna v koordinatah togega telesa velja $\frac{d\mathbf{\Gamma}_B}{dt} = \mathbf{J}\frac{d\boldsymbol{\omega}}{dt}$ in za drugi člen na desni strani enačbe (7.77) lahko zapišemo $\mathbf{R}^B_{ECI}\frac{d\mathbf{\Gamma}_{ECI}}{dt} = \mathbf{R}^B_{ECI}\mathbf{M}_{ECI} = \mathbf{M}$. $\mathbf{M}$ je vektor vseh zunanjih navorov, navorov zaradi vodenja in vsota vseh motilnih navorov (sončni pritisk, navor zaradi gradienta gravitacije, navor zaradi trenja z ozračjem in navor zaradi preostalega oz. remanentnega magnetizma), ki delujejo na togo telo. Končno enačbo gibanja podaja diferencialna enačba, kjer upoštevamo $\mathbf{\Omega}'\mathbf{\Gamma}_B = -\boldsymbol{\omega} \times \mathbf{\Gamma}_B = -\boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega})$

$$\frac{d\boldsymbol{\omega}}{dt} = \mathbf{J}^{-1}(\mathbf{M} - \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega})) \tag{7.78}$$

V tej enačbi so vse veličine (tudi $\mathbf{M}$) izražene v koordinatnem sistemu togega telesa. $\boldsymbol{\omega}$ je trenutna kotna hitrost telesa glede na inercialni sistem ECI, vendar izražena v koordinatnem sistemu togega telesa.

7.8 Regulacija orientacije

Satelit, ki leti v svoji tirnici ima določeno misijo, kjer se mora pogosto orientirati v neko referenčno orientacijo. Nekaj možnih primerov je: komunikacija z zemeljsko postajo, kjer se mora obrniti proti njej ali opazovanje določene pozicije na Zemlji, polniti akumulatorjev preko sončnih celic, ki jih je potrebno obrniti proti soncu in podobno. Pri regulaciji želimo odpraviti pogrešek orientacije.

Regulacijo orientacije je možno izvesti pri različnih parametrizacijah orientacije. Danes so se največ uporablja parametrizacija s kvaternioni. Kot je opisano v razdelku 7.4 opišemo zaporedna vrtenja s produktom dveh kvaternionov. Trenutno orientacijo satelita opišemo s kvaternionom $\mathbf{q}(t)$, ki predstavlja zasuk proti njegovi začetni orientaciji. Na enak način opisemo tudi želeni (referenčni) zasuk $\mathbf{q}^r(t)$; lahko ga n.pr. dobimo iz Eulerjevih kotov (ki si jih lažje predstavljamo) s transformacijo (7.32). Kvaternion potrebnega zasuka (pogrešek zasuka) $\mathbf{q}^e(t) = q^e_0 + q^e_1\mathbf{i} + q^e_2\mathbf{j} + q^e_3\mathbf{k}$ dobimo tako s produktom konjugirane vrednosti kvaterniona položaja (spomnimo, da to predstavlja vrtenje v nasprotno smer) in kvaterniona reference.

$$\mathbf{q}^e(t) = \mathbf{q}^*(t)\mathbf{q}^r(t).$$

Potreben zasuk (Φ) in os vrtenja (e_1, e_2, e_3) dobimo iz en.(7.17)

$$\Phi = 2\arccos(q_0^e) \tag{7.79}$$

$$e_1 = \frac{q_1^e}{\sin\frac{\Phi}{2}} \tag{7.80}$$

$$e_2 = \frac{q_2^e}{\sin\frac{\Phi}{2}} \tag{7.81}$$

$$e_3 = \frac{q_3^e}{\sin\frac{\Phi}{2}} \tag{7.82}$$

Opozoriti je potrebno, da os vrtenja ni definirana pri zasuku $\Phi = 0$. Prav tako je smiselno uporabiti absolutno vrednost "realnega" dela dela kvaterniona q_0^e, saj negativni q_0^e ustreza zasukom večjim kot 180°, zato je bolj smiselno uporabiti kot vrtenja, ki je razlika do 360°, in obrniti njegovo smer.

Potreben zasuk okrog posamezne osi (ponovno poudarjamo, da so vse veličine izražene v koordinatnem sistemu telesa - satelita) je tako produkt potrebnega kota zasuka in pa ustrezne komponente osi

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \Phi \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{7.83}$$

Zasuk okrog posamezne osi zagotovijo reakcijska kolesa. Seveda lahko pride v primeru neenakih komponent tenzorja do medsebojnih vplivov med vrtenji okrog posameznih osi.

7.9 Upravljanje orientacije satelita z reakcijskimi kolesi

Satelit, ki leti v svoji tirnici v vesolju mora imeti možnost nastavljanja orientacije, če želi opazovati določeno področje na Zemlji ali komunicirati z zemeljsko postajo. Ko je orientacija satelita znana in ni enaka želeni, jo je potrebno spremeniti z aktivatorji. Reakcijska kolesa (angl. reaction wheel, torquers)) so zelo pogost aktivator za dosego želene usmeritve satelita. Najbolj pogosta je uporaba treh reakcijskih koles, ki so nameščena v ortogonalnih oseh. Reakcijsko kolo sestoji iz vztrajnika pritrjenega na elektromotor. Če na satelit ne deluje zunanji motilni navor, potem vrtenje vztrajnika v določeno smer povzroči vrtenje satelita v nasprotni smeri, saj se vrtilna količina sistema ohranja.

Ker reakcijska kolesa med obratovanjem "nabirajo"moment (se vrtijo vedno hitreje), jih je potrebno razbremeniti, kar lahko naredimo z mikropotisnimi motorji (micro-thruster) ali pa magnetnimi tuljavami (magnitic torquers) v nizkih orbitah kjer je zemeljsko magnetno polje še dovolj močno.

7.9.1 Model satelita z reakcijskimi kolesi

Model vrtenja satelita, ki je vzbujan z zunanjim navorom smo opisali v poglavju 7.7, sedaj pa ga razširimo še z modelom reakcijskih koles, ki ustvarjajo navor na satelit (v oseh, kjer so montirani).

Celotno vrtilno količino satelita [27] lahko ločimo na vrtilno količino statičnih delov satelita $\mathbf{\Gamma}_B$ in vrtilno količino n poljubno orientiranih reakcijskih koles $\mathbf{\Gamma}_{W\to B}$ (njihovi vrteči deli - vztrajnik in rotor motorja).

$$\mathbf{\Gamma} = \mathbf{\Gamma}_B + \mathbf{\Gamma}_{W\to B} \tag{7.84}$$

kjer je vrtilna količina $\mathbf{\Gamma}_B$ produkt tenzorja vztrajnostnih momentov satelita in kotne hitrosti vrtenja satelita okrog njegovih osi, torej

$$\mathbf{\Gamma}_B = \mathbf{J}\boldsymbol{\omega} \tag{7.85}$$

Vsako reakcijsko kolo je montirano v svoji osi g_i $(i = 1 \cdots n)$, ki so stolpci v matriki $3 \times n$ postavitve vztrajnikov $\mathbf{G} = [g_1, \cdots, g_n]$. Komponento vrtilne količine zaradi vztrajnikov $\mathbf{\Gamma}_{W\to B}$ izrazimo s projekcijo vrtilnih količin posameznih vztrajnikov $\mathbf{\Gamma}_W = [\Gamma_1, \cdots, \Gamma_n]^T$ (okoli njihovih osi g_i) kot

$$\mathbf{\Gamma}_{W\to B} = \; G\mathbf{\Gamma}_W \tag{7.86}$$

Vztrajnostne momente reakcijskih koles označimo z J_i in njihove kotne hitrosti vrtenja z Ω_i. Vrtilna količina posameznega vztrajnika i podana v osi vztrajnika je produkt J_i in skupne kotne hitrosti vztrajnika, ki predstavlja vsoto kotne hitrosti vztrajnika in komponente zaradi kotne hitrosti satelita $(\mathbf{G}^T\boldsymbol{\omega})$. Vektor vrtilnih količin posameznih vztrajnikov okoli njihovih osi torej dobimo z

$$\mathbf{\Gamma}_W = \mathbf{J}_W \left(\mathbf{G}^T\boldsymbol{\omega} + \mathbf{\Omega}_W\right) \tag{7.87}$$

kjer je $\mathbf{\Omega}_W = [\Omega_1, \cdots, \Omega_n]^T$ vektor kotnih hitrosti vztrajnikov in $\mathbf{J}_W$ diagonalna matrika vztrajnostnih momentov vztrajnikov (dimenzije $n \times n$)

$$\mathbf{J}_W = \begin{bmatrix} J_1 & 0 & \cdots & 0 \\ 0 & J_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & J_n \end{bmatrix}$$

Enačbo za gibanje sistema dobimo z odvajanjem enačbe (7.84). Pri odvajanju moramo upoštevati, da se koordinatni sistem satelita vrti glede na ECI koordinatni sistem (Eulerjeva enačba gibanja). Odvod enačbe (7.85) je glede na (7.78) sledeč

$$\frac{d\mathbf{\Gamma}_B}{dt} = -\boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) + \mathbf{M} \tag{7.88}$$

kjer zunanji navor zanemarimo M=0 saj predstavlja motilni navor (sončni pritisk, trenja ozračja,...), ki ga ne bomo upoštevali. Podobno naredimo še odvod vrtilne količine satelita zaradi vrtenja vztrajnikov (7.86)

$$\frac{d\mathbf{\Gamma}_{W\to B}}{dt} = \frac{d\mathbf{G}}{dt}\mathbf{\Gamma}_W + \mathbf{G}\frac{d\mathbf{\Gamma}_W}{dt} = \frac{d\mathbf{G}}{dt}\mathbf{\Gamma}_W - \mathbf{G}\mathbf{M}_W \tag{7.89}$$

kjer je $\mathbf{M}_W$ vektor navorov reakcijskih koles okoli svojih osi. Navor $\mathbf{M}_W$ nastopa z minusom, saj vrtenje vztrajnikov v pozitivno smer povzroči vrtenje satelita v nasprotno smer. Za odvod smerne matrike vztrajnikov $\mathbf{G}$ pa velja podobno kot za odvod DCM matrike (7.69), torej $\frac{d\mathbf{G}}{dt} = \mathbf{\Omega}'\mathbf{G}$. Odvod vrtilne količine satelita zaradi vrtenja vztrajnikov je torej

$$\frac{d\mathbf{\Gamma}_{W\to B}}{dt} = \Omega'\mathbf{G}\mathbf{\Gamma}_W - \mathbf{G}\mathbf{M}_W = -\boldsymbol{\omega} \times \mathbf{G}\mathbf{\Gamma}_W - \mathbf{G}\mathbf{M}_W \tag{7.90}$$

Z vstavitvijo enačb (7.88) in (7.90) v odvod enačbe (7.84) dobimo končno enačbo gibanja satelita z reakcijskimi kolesi

$$\frac{d\boldsymbol{\omega}}{dt} = \mathbf{J}^{-1}\left(-\boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) - \boldsymbol{\omega} \times \mathbf{G}\mathbf{\Gamma}_W - \mathbf{G}\mathbf{M}_W\right) \tag{7.91}$$

oziroma, če vektorski produkt realiziramo z matričnim zapisom

$$\frac{d\boldsymbol{\omega}}{dt} = \mathbf{J}^{-1}\left(\Omega'\mathbf{J}\boldsymbol{\omega} + \Omega'\mathbf{G}\mathbf{\Gamma}_W - \mathbf{G}\mathbf{M}_W\right) \tag{7.92}$$

Dinamiko vrtenja reakcijskih koles pa lahko zadovoljivo opišemo z

$$\frac{d\mathbf{\Omega}}{dt} = \mathbf{J}_W^{-1}\mathbf{M}_W - \mathbf{G}^T\frac{d\boldsymbol{\omega}}{dt} \tag{7.93}$$

7.9.2 Vodenje satelita z reakcijskimi kolesi

Regulacijski algoritem na podlagi pogreška orientacije izračuna potrebne navore v oseh satelita $\mathbf{M}$, katere lahko izrazimo z navori v oseh reakcijskih koles kot

$$\mathbf{M} = \mathbf{G}\mathbf{M}_W \tag{7.94}$$

Glede na (7.94) mora biti matrika $\mathbf{G}$ polnega ranga, da lahko z reakcijskimi kolesi generiramo poljuben navor na osi satelita. V kolikor imamo več kot tri reakcijska kolesa, je možnih več vektorjev navorov $\mathbf{M}_W$ v oseh reakcijskih koles, ki povzročijo v isti navor v oseh satelita $\mathbf{M}$. Označimo z $\mathbf{M}_W^*$ rešitev, ki ima minimalno normo in jo dobimo s pseudo inverzom relacije (7.94)

$$\mathbf{M}_W^* = \left(\mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{M} \tag{7.95}$$

Ta rešitev je v praksi uporabna [27], saj so navori reakcijskih koles omejeni. Vse ostale rešitve enačbe (7.94) izrazimo kot

$$\mathbf{M}_W = \mathbf{M}_W^* + \mathbf{M}_{Ws} \tag{7.96}$$

kjer je $\mathbf{M}_W$ skupni navor reakcijskih koles in $\mathbf{M}_{Ws}$ dodatni navor, ki ga je potrebno določiti tako, da ne spremenimo obnašanje satelita. Navor $\mathbf{M}_W$ mora torej povzročiti enako gibanje satelita kot navor $\mathbf{M}_W^*$, kar pomeni, da mora veljati

$$\mathbf{G}\mathbf{M}_{Ws} = \mathbf{0} \tag{7.97}$$

Dodatni navor je torej v ničnem prostoru matrike $\mathbf{G}$. S tem dosežemo, da je dodatni navor $\mathbf{M}_{Ws}$ razklopljen glede na navor vodenja v oseh satelita $\mathbf{M}$, kar omogoča dodatno svobodo pri načrtovanju regulatorja z želenim optimalnim delovanjem.

Poglavje 8

Letalni sistemi

8.1 Uvod

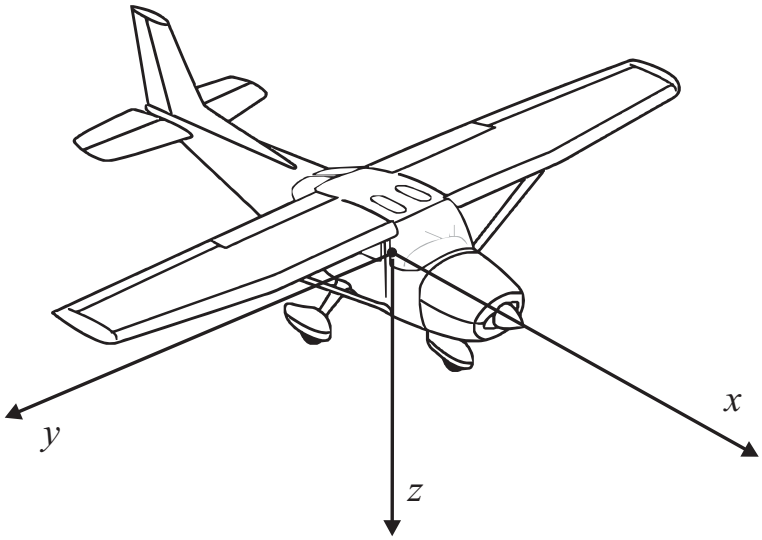
Obkrožajo nas številne letalne naprave oziroma sistemi od katerih so nam še najbolj domača letala. Poled potniških letal in vojaških letal se uveljavljajo tudi avtonomna letala, ki so večinoma manjša in avtonomna (brezpilotna). Njihova misija je lahko zgolj raziskovalna (testiranje algoritmov za stabilizacijo leta - avtopilotov) ali pa opazovanje, reševanje, snemanje iz zraka in podobno.

V nadaljevanju nakažemo matematično modeliranje gibanja letala, saj je to v praksi precej obsežno in zahtevno. Podamo le splošne enačbe gibanja togega telesa in nekatere zakonitosti leta. To je potrebno razširiti še z modeliranjem aerodinamike, kar pa ni cilj tega dela. Večinoma je potrebno dobljene teoretične modele nekega letala še ovrednotiti in jih dopolniti z različnimi eksperimentalno določenimi koeficienti (testi v vetrovnikih).

8.2 Eulerjeve enačbe gibanja togega telesa

Letalo lahko obravnavamo kot togo telo, s težiščem, na katerega je pripet koordinatni sistem telesa kot podaja slika 8.1. Njegova x os je vzdolžno skozi trup letala in kaže v smeri leta, y os kaže v smeri desnega krila in z os je usmerjena navzdol, tako da osi definirajo desnosučni koordinatni sistem.

Gibanje letala lahko razdelimo na translatorno gibanje in na kroženje [28]. Translatorno gibanje povzročajo sile, ki delujejo na njegovo težišče. Sile, ki imajo prijemališče izven težišča, pa povzročajo navor okoli težišča, kar povzroči vrtenje letala okoli težišča. Med letom na letalo delujejo različne sile, ki so prikazane na sliki 8.2. Te sile predstavljajo rezultante realnih sil, katere je potrebno za dani model oceniti.



Slika 8.1: Koordinatni sistem letala.

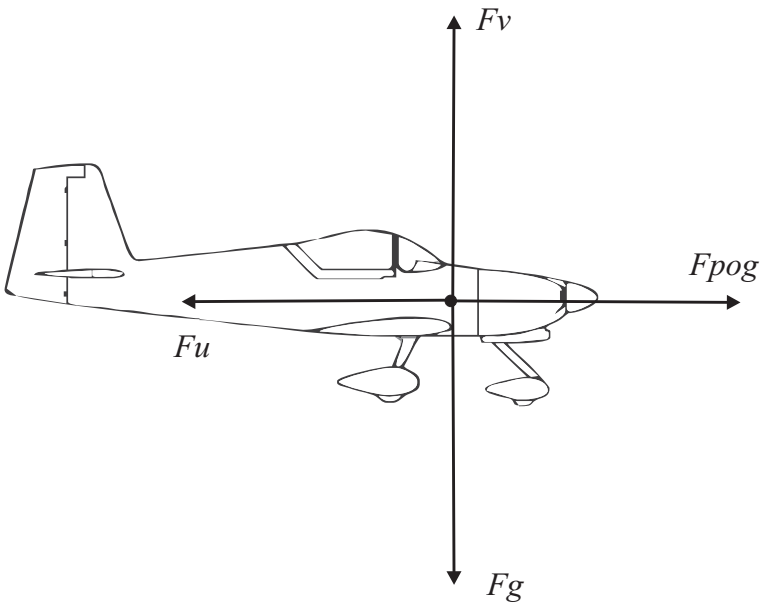
Slika 8.2 prikazuje primer leta letala, kjer so vse sile v ravnovesju in imajo prijemališče v težišču letala, zato ni navorov, ki bi povzročali vrtenje letala. Letalo torej v tem primeru leti translatorno in nepospešeno. Na je $Fpog$ sila, ki jo povzroča pogon letala. V nasprotni smeri deluje rezultanta sil upora $Fpog$, ki vključuje prispevke sile zaradi upora kril, trupa, repnih stabilizatorjev, zakrilc, krmil,... Podobno velja tudi za rezultanto sile vzgona Fv. Obe sili ocenimo s pomočjo aerodinamičnega modela letala, ki je večinoma precej kompleksen (več podrobnosti o modeliranju lahko najdete v [29]) in ga zato tu ne bomo obravnavali. Teža letala povzroči, da v težišču letala deluje sila zaradi gravitacije Fg.

Gibanje letala, v translatorni smeri, povzroči rezultanta vseh sil F ki deluje nanj, kroženje pa rezultanta vseh navorov M. Splošni enačbi za omenjeno gibanje sta

$$\mathbf{F} = m\frac{d\mathbf{v}}{dt} + m\boldsymbol{\omega} \times \mathbf{v} \tag{8.1}$$

$$\mathbf{M} = \mathbf{J}\frac{\boldsymbol{\omega}\mathbf{v}}{dt} + \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) \tag{8.2}$$

kjer je $\boldsymbol{\omega}$ vektor kotnih hitrosti letala okoli njegovih osi vpetih v težišču letala, $\mathbf{v}$ vektor translatornih hitrosti težišča letala, m je masa letala, $\mathbf{J}$ je tenzor vztrajnostnih momentov letala, $\mathbf{F}$ je vektor vsote vseh zunanjih sil v smereh koordinatnih osi letala in $\mathbf{M}$ vektor vsote vseh zunanjih navorov okoli osi letala. Opazimo, da je rotacijski del enak kot pri satelitu, saj gre za



Slika 8.2: Sile na letalo med letom: $Fpog$ sila pogona, Fg sila gravitacije, Fu sila upora in Fv vzgon.

splošne Eulerjeve enačbe gibanja. V kolikor želimo simulirati model, enačbe prevedemo v obliko

$$\frac{d\mathbf{v}}{dt} = \frac{\mathbf{F}}{m} - \boldsymbol{\omega} \times \mathbf{v} \tag{8.3}$$

$$\frac{d\boldsymbol{\omega}}{dt} = \mathbf{J}^{-1} \left(\mathbf{M} - \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) \right) \tag{8.4}$$

Poglavje 9

Nedeterminističnost v mobilnih sistemih

9.1 Uvod

V realnem svetu deterministični dogodki (skoraj) ne obstajajo. Podajmo nekaj primerov.

Nikoli ne moremo povsem z gotovostjo ugotoviti s kakšno hitrostjo se peljemo, saj imajo merilniki večjo ali manjšo negotovost (območje, občutljivost, resolucija, fizikalne omejitve in podobno). Dodatno so meritve senzorjev podvržene šumu, motnjam iz okolice in možnostim nastopa okvar. Vsi ti dejavniki nam omejujejo koristno informacijo.

Podobno je z aktuatorji, kot na primer motorni pogon. Zaradi šuma, obrabe ali mehanske okvare ne moremo natančno določiti njegove obrate pri danem vzbujanju.

Del negotovosti izvira tudi iz programske opreme, saj so vsi modeli (taki ali drugačni opisi okolja) le približni.

Negotovost je prisotna tudi v delovanju različnih algoritmov, ki dajejo rezultate z neko zadovoljivo natančnostjo. Natančnost je omejena tudi zato, ker morajo algoritmi delovati v realnem času. Torej je čas izračuna omejen pri omejenih procesnih zmožnostih in želeni odzivnosti sistema.

Tudi samo okolje kjer mobilni agent (kolesni robot) deluje je nepredvidljivo, stopnja nepredvidljivosti je sicer manjša v strukturiranih okoljih (umetno zgrajena okolja z ravnini stranicami,...) in je večja v dinamično spremenljivih okoljih (domovi, cestni promet, bližina ljudi,...).

Gradnja robustnih sistemov se mora v prvi vrsti uspešno kosati ravno z različnimi negotovostmi v realnem svetu.

9.2 Osnove verjetnosti

Označimo z X naključno spremenljivko in z x vrednost, ki jo X lahko zavzame. Če X lahko zavzame končno število vrednosti (npr.: izid meta kovanca je cifra ali lik), potem govorimo o **diskretni slučajni spremenljivki** drugače pa o **zvezni slučajni spremenljivki** (npr.: teža kovanca).

V primeru diskretne spremenljivke označimo verjetnost, da X zavzame vrednost x z

$$P(X = x)$$

oziroma krajše $P(x)$. Vsota verjetnosti vseh posameznih izidov diskretne spremenljivke X je 1.

$$\sum_x P(X = x) = 1$$

V primeru zvezne naključne spremenljivke pa je $P(X = x) = 0$, saj ima X neskončno zalogo vrednosti. Pri opisu verjetnosti verjetnosti zato predvsem uporabljamo funkcijo gostote **porazdelitve verjetnosti**. Verjetnost da x zavzame vrednost nekje znotraj intervala $\in [a, b]$ je

$$P(a < X < b) = \int_a^b p(x)dx$$

Podobno kot v primeru diskretnih slučajnih spremenljivk velja za zvezne, da je integral porazdelitve verjetnosti po celotni zalogi enak

$$P(-\infty < X < +\infty) = \int_{-\infty}^{+\infty} p(x)dx = 1$$

Pogosta je uporaba **normalne porazdelitve**, podane z Gaussovo funkcijo

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

kjer je μ srednja vrednost (matematično upanje) in σ^2 varianca. Srednja vrednost oz. matematično upanje je določeno z

$$E[X] = \mu = \int_{-\infty}^{+\infty} xp(x)dx$$

varianca pa je matematično upanje kvadratov odstopanja od srednje vrednosti

$$E[(x-\mu)^2] = \sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 p(x)dx$$

diskretne spremenljivke	zvezne spremenljivke
$\sum_x P(x) = 1$	$\int p(x)dx = 1$
$P(x) = \sum_y P(x, y)$	$p(x) = \int p(x, y)dy$
$P(x) = \sum_y P(x\|y)P(y)$	$p(x) = \int p(x\|y)p(y)dy$

Tabela 9.1: Teorem polne verjetnosti

V primeru, da je naključna spremenljivka $\mathbf{x}$ vektor (ima več dimenzij) potem je njena normalna porazdelitev enaka

$$p(\mathbf{x}) = \det(2\pi\mathbf{\Sigma})^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)}$$

kjer je $\mathbf{\Sigma}$ kovariančna matrika.

Verjetnost, da se hkrati zgodita dva dogodka (da spremenljivka X zavzame vrednost x in Y zavzame vrednost y), opisuje **skupna porazdelitev** verjetnosti

$$p(x, y) = p(X = x, Y = y)$$

Če sta X in Y neodvisna

$$p(x, y) = p(x)p(y)$$

Pogojna verjetnost je verjetnost $p(x|y)$ da X zavzame vrednost x, če predhodno vemo, da je Y zavzel vrednost y. Velja

$$p(x|y) = \frac{p(x, y)}{p(y)}$$

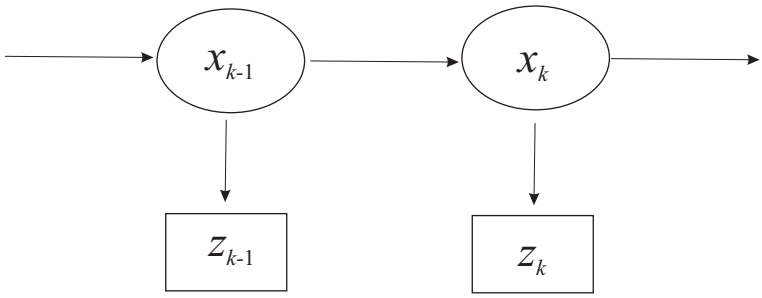
Torej Y nosi informacijo o X, če pa sta spremenljivki X in Y neodvisni pa

$$p(x|y) = \frac{p(x, y)}{p(y)} = \frac{p(x)p(y)}{p(y)} = p(x)$$

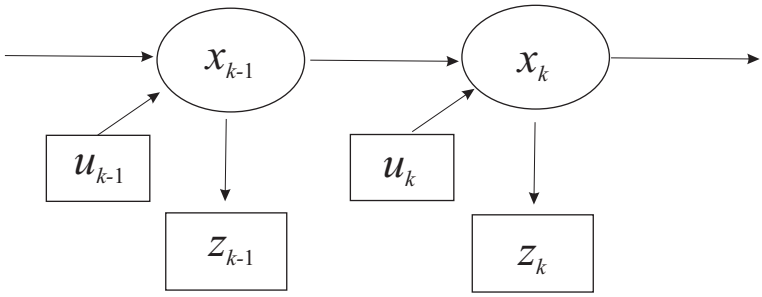
Podajmo še teorem **polne verjetnosti** s tabelo 9.1

Obravnavali bomo procese, za katere velja predpostavka, da je **stanje vsebovano**, kar pomeni, da stanja vsebuje celotno informacijo o sistemu, ki jo potrebujemo za njegov opis v danem trenutku. Za opis sistema rabimo poznati le trenutno vrednost stanja, kar je lastnost Markovih procesov. Slika 9.1 prikazuje prikriti Markov proces (vrednosti stanj niso dostopna, lahko jih le ocenimo preko meritve), kjer je izid meritve stohastično odvisen od trenutne vrednosti stanj. Zaradi te predpostavke (Markov proces) je trenutno stanje pogojno odvisno le od preteklega stanja, ne pa od prejšnjih stanj

$$p(x_k|x_0, \cdots, x_{k-1}) = p(x_k|x_{0:k-1}) = p(x_k|x_{k-1})$$



Slika 9.1: Prikriti Markov proces, meritev z_k je stohastično odvisna od stanja x_k. Stanje ni direktno dostopno, odvisno je od pretekle vrednosti stanja.



Slika 9.2: Prikriti Markov proces, meritev z_k je stohastično odvisna od stanja x_k. Stanja ni direktno dostopno, odvisno pa je od pretekle vrednosti stanja in trenutnega vhoda.

Podobno je izid meritve neodvisen od vrednosti vseh preteklih stanj, če poznamo trenutno stanje

$$p(z_k|x_0, \cdots, x_k) = p(z_k|x_{0:k}) = p(z_k|x_k)$$

Prikriti Markov proces, kjer je trenutno stanje x_k odvisno od preteklega stanja x_{k-1} in trenutnega vhoda u_k prikazuje slika 9.2

9.2.1 Bayesovo pravilo

Pomembno vlogo v verjetnostni robotiki ima **Bayesovo pravilo**

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{9.1}$$

oziroma za diskretne spremenljivke

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{9.2}$$

ki nam omogoča, da določeno verjetnost, ki jo je težje določiti, izrazimo z drugimi, lažje določljivimi verjetnostmi. Bayesovo pravilo lahko napišemo tudi v krajši obliki

$$P(x|y) = \eta P(y|x)P(x)$$

kjer imenujemo $\eta = \frac{1}{p(y)} = \frac{1}{\int p(y|x)p(x)dx}$ normirni koeficient.

Primer 9.1. *Podajmo primer uporabe Bayesovega pravila (9.1) v mobilnih sistemih. Naj X predstavlja stanje našega robota, ki ga želimo oceniti (npr. razdaljo robota do ovire) na podlagi meritve $Y = y$. Ker je izid meritve negotov želimo dobiti oceno kako verjetna je ocena stanja x na podlagi opravljene meritve y.*

Rešitev

Zanima nas $p(x|y)$, kar lahko ocenimo iz (9.1). Verjetnost $p(x)$ vsebuje znanje o X pred opravljeno meritvijo y, zato jo imenujemo ***napovedna ocena*** *(s tujko* ***a priori****). Verjetnost $p(x|y)$ imenujemo* ***trenutna ocena*** *(s tujko* ***a posteriori****) in podaja znanje o X po opravljeni meritvi. Verjetnost $p(y|x)$, nam posreduje informacijo, kako stanje X povzroči meritev Y in predstavlja model senzorja (npr. porazdelitev verjetnosti izda meritve razdalje do ovire, če se nahajamo na določeni razdalji pred oviro. $p(y)$ pa podaja verjetnost opravljene meritve y. Torej zaupanje v opravljeno meritev. $p(y)$ je neodvisna od x in jo lahko določimo s teoremom polne verjetnosti $p(y) = \int p(y|x)p(x)dx$. Torej za trenutno oceno $p(x|y)$ rabimo poznati le statističen model senzorja (podan z $p(y|x)$) ter predhodno (napovedno) oceno verjetnosti stanja $p(x)$.*

Primer 9.2. *Do cilja obstajajo tri možne poti. Robot s 70% verjetnostjo izbere prvo, z 10% drugo in z 20% tretjo pot. Na prvi poti ima 5%, na drugi 10% in na tretji 8% možnost naleta na oviro.*

1. *Kako verjetno je, da robot naleti na oviro?*

2. *Robot naleti na oviro. Kakšna je verjetnost, da se je to zgodilo na prvi poti?*

Rešitev

1. *Verjetnost, da izberemo določeno pot je $P(A_1) = 0.7$, $P(A_2) = 0.1$, $P(A_3) = 0.2$. Verjetnost da naleti na oviro na prvi poti $P(B|A_1) =$*

0.05, *na drugi* $P(B|A_2) = 0.1$ *in na tretji* $P(B|A_3) = 0.08$. *Verjetnost, da robot naleti na oviro je*

$$P(B) = P(B|A_1)P(A_1) + P(B|A_2)P(A_2) + P(B|A_3)P(A_3)$$
$$P(B) = 0.05 \cdot 0.7 + 0.1 \cdot 0.1 + 0.08 \cdot 0.2 = 0.061$$

2. *Verjetnost, da obtičimo na prvi poti je*

$$P(A_1|B) = \frac{P(B|A_1)P(A_1)}{P(B)} = \frac{0.05 \cdot 0.7}{0.05 \cdot 0.7 + 0.1 \cdot 0.1 + 0.08 \cdot 0.2} = 0.5738$$

Primer 9.3. *Robot ima senzor za detekcijo odprtosti vrat. Stanje, ki nas zanima je odprtost vrat* $X \in (odprta, zaprta)$. *Verjetnost, da so vrata v objektu odprta je* $P(X = odprta) = 0.4$. *Meritev senzorja* $Z \in (odprta, zaprta)$ *je pošumljena in jo lahko opišemo s sledečimi pogojnimi verjetnostmi*

$$P(Z = odprta|X = odprta) = 0.8$$
$$P(Z = zaprta|X = zaprta) = 0.9$$

Opazimo, da je verjetnost, da senzor izmeri narobe dokaj majhna. Če so vrata odprta je verjetnost napačne meritve 0.2, v primeru pa, da so vrata zaprta pa je verjetnost napačne meritve 0.1. Kolikšna je verjetnost, da so vrata odprta, če senzor zazna odprtost vrat?

Rešitev

Uporabimo sledeče oznake

$$P(x) = P(X = odprta) = 0.4$$
$$P(\bar{x}) = P(X = zaprta) = 1 - P(x) = 0.6$$
$$P(z|x) = P(Z = odprta|X = odprta) = 0.8$$
$$P(\bar{z}|\bar{x}) = P(Z = zaprta|X = zaprta) = 0.9$$

Zanima nas

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)}$$

s teoremom polne verjetnosti določimo verjetnost, da senzor zazna odprta vrata

$$P(z) = P(z|x)P(x) + P(z|\bar{x})P(\bar{x}) = 0.8 \cdot 0.4 + 0.1 \cdot 0.6 = 0.38$$

kjer je $P(z|\bar{x}) = 1 - P(\bar{z}|\bar{x}) = 0.1$. *Iskana pogojna verjetnost je*

$$P(x|z) = \frac{0.8 \cdot 0.4}{0.38} = 0.8421$$

dobljena verjetnost je visoka, a kljub temu obstaja 0.1579 verjetnost ($P(\bar{x}|z) = 1-P(x|z)$), da se na podlagi meritve odločimo narobe in z robotom zapeljemo v zaprta vrata.

9.2.2 Bayesov filter, primer opazovanja

Bayesov filter predstavlja najbolj splošen algoritem za izračun porazdelitev verjetnosti.

V primeru 9.3 smo videli, kako lahko ocenimo vrednost stanja ob znanem izidu meritve $p(x|z)$, če poznamo statistični model senzorja $p(z|x)$ (porazdelitev verjetnosti izda meritve ob poznanem stanju) in verjetnost izida meritve $p(z)$.

Poglejmo si še, kako združujemo informacijo večkratno zaporedno opravljene meritve z pri oceni verjetnosti stanja x. Zanima nas $p(x_k|z_1, \cdots z_k)$, torej porazdelitev verjetnosti stanja v trenutku k, ob zaporedno opravljenih meritvah. Napišemo lahko Bayesov izrek v rekurzivni obliki

$$p(x_k|z_1, \cdots, z_k) = \frac{p(z_k|x_k, z_1, \cdots, z_{k-1})p(x_k|z_1, \cdots, z_{k-1})}{p(z_k|z_1, \cdots, z_{k-1})}$$

kar lahko krajše zapišemo kot

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k, z_{1:k-1})p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{9.3}$$

kjer je

- $p(x_k|z_{1:k})$ ocena porazdelitve verjetnosti stanja v trenutku k, posodobljenega s poznanimi meritvami,
- $p(z_k|x_k, z_{1:k-1})$ porazdelitev verjetnosti meritve v trenutku k, če poznamo trenutno stanje x_k in pretekle meritve do trenutka $k-1$,
- $p(x_k|z_{1:k-1})$ predikcija ocene porazdelitve verjetnosti stanja na osnovi preteklih meritev,
- $p(z_k|z_{1:k-1})$ verjetnost opravljene meritve (zaupanje v opravljeno meritev) v trenutku k.

Nadalje velja, da je trenutna meritev z_k v relaciji 9.3 neodvisna od preteklih meritev (stanje je vsebovano, Markov Proces) $z_{1:k-1}$, če poznamo stanje sistema x_k

$$p(z_k|x_k, z_{1:k-1}) = p(z_k|x_k)$$

zato lahko zapišemo 9.3 kot

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{9.4}$$

Izpeljavo relacije 9.4 prikazuje primer 9.4.

Primer 9.4. *Izpeljava relacije 9.4*

$$\begin{aligned}
p(x_k|z_{1:k}) &= \frac{p(z_{1:k}|x_k)p(x_k)}{p(z_{1:k})} \\
&= \frac{p(z_k,z_{1:k-1}|x_k)p(x_k)}{p(z_k,z_{1:k-1})} \\
&= \frac{p(z_k|z_{1:k-1},x_k)p(z_{1:k-1}|x_k)p(x_k)}{p(z_k|z_{1:k-1})p(z_{1:k-1})} \\
&= \frac{p(z_k|z_{1:k-1},x_k)p(x_k|z_{1:k-1})p(z_{1:k-1})p(x_k)}{p(z_k|z_{1:k-1})p(z_{1:k-1})p(x_k)} \\
&= \frac{p(z_k|z_{1:k-1},x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \\
&= \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})}
\end{aligned}$$

Rekurzivno pravilo za posodobitev stanja (9.4) na podlagi preteklih meritev vsebuje tudi predikcijo $p(x_k|z_{1:k-1})$. Zato celotni postopek razdelimo na predikcijski in korekcijski korak.

Predikcijski korak

Predikcijo $p(x_k|z_{1:k-1})$ določimo kot

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1}, z_{1:k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1}$$

kar lahko poenostavimo s predpostavko, da je stanje vsebovano. Dobimo

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{9.5}$$

kjer $p(x_k|x_{k-1})$ podaja porazdelitev verjetnost za prehajanje med stanji in $p(x_{k-1}|z_{1:k-1})$ je korekcijska ocena porazdelitve verjetnosti za stanja iz prejšnjega trenutka.

Korekcijski korak

Oceno stanj po opravljeni meritvi v trenutku k in predhodno izračunani predikciji določimo z

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{9.6}$$

Verjetnost $p(z_k|z_{1:k-1})$, ki podaja zaupanje v opravljeno meritev, določimo s pomočjo

$$p(z_k|z_{1:k-1}) = \int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k$$

Izračun najbolj verjetne ocene stanj

Vprašajmo se, kako nam lahko ocenjena verjetnost porazdelitve gostote $p(x_k|z_{1:k})$ pomaga pri oceni vrednosti stanja x_k. Najbolj verjetno ocena (matematično upanje) $E[x_k]$, ki minimizira povprečni kvadratični pogrešek meritve je

$$E[x_k] = \int x_k p(x_k|z_{1:k})dx_k$$

Ocenimo pa lahko tudi vrednost $x_{k_{max}}$, ki maksimira posteriorno verjetnost $p(x_k|z_{1:k})$

$$x_{k_{max}} = \max_{x_k} p(x_k|z_{1:k})$$

dobljeno vrednost imenujemo tudi normirni koeficient.

Primer 9.5. *Kakšna je verjetnost, da so vrata iz primera 9.3 odprta, če senzor v naslednjem trenutku $k = 2$ ponovno zazna odprta vrata?*
Rešitev
V primeru 9.3 smo določili, da so vrata v trenutku $k = 1$ odprta ob meritvi $z_1 =$ odprta, s pogojno verjetnostjo

$$P(x_1|z_1) = \frac{0.8 \cdot 0.4}{0.38} = 0.8421$$

V naslednjem trenutku $k = 2$ izmerimo $z_2 =$ odprta z verjetnostno $P(z_2|x_2) = 0.8$ pravilno in z verjetnostjo $P(z_2|\bar{x}_2) = 0.1$ napačno (lastnosti senzorja so enake, kot v primeru 9.3).

Določimo najprej predikcijo $P(x_2|z_1)$, torej da so vrata odprta na podlagi pretekle meritve. Upoštevamo relacijo 9.5, kjer integral zamenjamo z sumatorjem, saj obravnavamo diskretni primer

$$P(x_k|z_{1:k-1}) = \sum_{x_{k-1}} P(x_k|x_{k-1})P(x_{k-1}|z_{1:k-1})$$

kjer v našem primeru vrata le zaznavamo in na njih nič ne vplivamo (jih zapiramo ali odpiramo) je verjetnost za prehajanje stanj kar $P(x_2|x_1) = 1$ in $P(x_2|\bar{x}_1) = 0$. Dobimo torej

$$P(x_2|z_{1:1}) = P(x_2|x_1)P(x_1|z_1)+P(x_2|\bar{x}_1)P(\bar{x}_1|z_1) = 1{\cdot}0.8421+0{\cdot}0.1579 = 0.8421$$

kar je logičen rezultat, saj če meritve z_2 še nimamo, so vrata v trenutku $k = 2$ enako verjetno odprta kot v trenutku $k = 1$, saj nimamo nove informacije.

Nato določimo še korekcijo, upoštevajoča relacijo (9.6), dobimo

$$P(x_2|z_{1:2}) = \frac{P(z_2|x_2)P(x_2|z_{1:1})}{P(z_2|z_{1:1})} = \frac{0.8 \cdot 0.8421}{P(z_2|z_{1:1})}$$

kjer moramo določiti še normirni faktor, to je verjetnost, da so vrata v trenutku $k = 2$ odprta. To določimo tako, da seštejemo verjetnosti vseh možnih kombinacij stanj, ki lahko rezultirajo v trenutno meritev z_k, upoštevajoč izide preteklih meritev $z_{1:k-1}$

$$P(z_k|z_{1:k-1}) = \sum_{x_k} P(z_k|x_k)P(x_k|z_{1:k-1})$$

V našem primeru je

$$P(z_2|z_1) = P(z_2|x_2)P(x_2|z_1)+P(z_2|\bar{x}_2)P(\bar{x}_2|z_1) = 0.8\cdot0.8421+0.1\cdot0.1579 = 0.6895$$

Končni rezultat, ki podaja verjetnost, da so vrata odprta, če dvakrat zapored izmerimo odprtost vrat, je

$$P(x_2|z_{1:2}) = 0.9771$$

Primer 9.6. *Za primer 9.3 določite, kako se spreminja verjetnost, da so vrata odprta, če izmerimo tri meritve $z_{1:3} = (odprta, odprta, zaprta)$*

Rešitev

Prvi dve pogojni verjetnosti smo že določili, določimo še tretjo

$$P(x_3|z_{1:3}) = \frac{P(z_3|x_3)P(x_3|z_{1:2})}{P(z_3|z_{1:2})} = \frac{0.2 \cdot 0.9771}{0.2 \cdot 0.9771 + 0.9 \cdot (1 - 0.9771)} = 0.9046$$

kjer je $P(Z = zaprta|X = odprta) = 1 - P(Z = odprta|X = odprta)$. Verjetnosti za odprtost vrat pri zaporednih meritvah ($i \in 1, 2, 3$) torej so

$$P(x_k|z_{1:i}) = (0.8421, 0.9771, 0.9046)$$

9.2.3 Bayesov filter, primer opazovanja in akcije

V poglavju 9.2.2 smo okolico le opazovali in na podlagi tega ocenjevali stanja. Večinoma pa lahko nek sistem na okolico tudi vpliva z akcijami, ki jih v tem okolju počne (npr. robot se premika, odpira ali zapira vrata,...). Pri vsaki izvedeni akciji, obstaja neka negotovost, zaradi česar izid akcije ni deterministično, ampak je podan z neko verjetnostjo. Ta verjetnost je $p(x_k|x_{k-1}, u_k)$ in podaja verjetnost prehoda iz prejšnjega stanja v naslednje stanje pri znani akciji v okolici. V splošnem akcije izvedene v neki okolici povečujejo negotovost našega znanja o okolici, opravljene meritve pa to negotovost zmanjšujejo.

Poglejmo si, kako vplivajo opravljene akcije in meritve v oklici na naše vedenje o stanjih. Zanima nas torej $p(x_k|z_{1:k}, u_{1:k})$, kjer so z meritve in u akcije.

Podobno kot v podpoglavju 9.2.2 napišimo Bayesov izrek

$$p(x_k|z_{1:k}, u_{1:k}) = \frac{p(z_k|x_k, z_{1:k-1}, u_{1:k})p(x_k|z_{1:k-1}, u_{1:k})}{p(z_k|z_{1:k-1}, u_{1:k})} \tag{9.7}$$

kjer je

- $p(x_k|z_{1:k}, u_{(1:k})$ ocena porazdelitve verjetnosti stanja v trenutku k, posodobljenega s poznanimi meritvami in opravljenimi akcijami,
- $p(z_k|x_k, z_{1:k-1}, u_{1:k})$ porazdelitev verjetnosti meritve v trenutku k, če poznamo trenutno stanje x_k, opravljene akcije in pretekle meritve do trenutka $k-1$,
- $p(x_k|z_{1:k-1}, u_{1:k})$ predikcija ocene porazdelitve verjetnosti stanja na osnovi preteklih meritev in opravljenih akcij do trenutka k,
- $p(z_k|z_{1:k-1}, u_{1:k})$ verjetnost opravljene meritve (zaupanje v opravljeno meritev) v trenutku k.

Nadalje velja, da trenutno meritev z_k v relaciji (9.7) lahko opišemo le s poznanim stanjem sistema x_k, saj pretekle meritve in akcije ne prinašajo dodatne informacije (stanje je vsebovano, Markov Proces).

$$p(z_k|x_k, z_{1:k-1}, u_{1:k}) = p(z_k|x_k)$$

zato lahko zapišemo 9.7 kot

$$p(x_k|z_{1:k}, u_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1}, u_{1:k})}{p(z_k|z_{1:k-1}, u_{1:k})} \tag{9.8}$$

Rekurzivno pravilo za posodobitev verjetnosti ocene stanja (9.8) na podlagi preteklih meritev in akcij vsebuje tudi predikcijo $p(x_k|z_{1:k-1}, u_{1:k})$, kjer

iz preteklih meritev $z_{1:k-1}$ in vseh akcij $u_{1:k}$ napovemo porazdelitev verjetnosti za oceno stanja. Zato celotni postopek razdelimo na predikcijski in korekcijski korak. V predikcijskem koraku trenutne meritve še ne poznamo, poznamo pa trenutno akcijo, zato lahko na podlagi modela sistema napovemo verjetnostno porazdelitev za stanje. Ko pa je trenutna meritev na voljo izvedemo še korak korekcije.

Predikcijski korak

Predikcijo $p(x_k|z_{1:k-1}, u_{1:k})$ določimo z uporabo polne verjetnosti kot

$$p(x_k|z_{1:k-1}, u_{1:k}) = \int p(x_k|x_{k-1}, z_{1:k-1}, u_{1:k})p(x_{k-1}|z_{1:k-1}, u_{1:k})dx_{k-1}$$

kjer zaradi vsebovanosti stanja velja

$$p(x_k|x_{k-1}, z_{1:k-1}, u_{1:k}) = p(x_k|x_{k-1}, u_k)$$

nadalje pa tudi ugotovimo, da za oceno stanja v prejšnjem trenutku, vrednost trenutne akcije ni pomembna

$$p(x_{k-1}|z_{1:k-1}, u_{1:k}) = p(x_{k-1}|z_{1:k-1}, u_{1:k-1})$$

Končni izraz za izračun predikcijske ocene je

$$p(x_k|z_{1:k-1}, u_{1:k}) = \int p(x_k|x_{k-1}, u_k)p(x_{k-1}|z_{1:k-1}, u_{1:k-1})dx_{k-1} \tag{9.9}$$

kjer $p(x_k|x_{k-1}, u_k)$ podaja razporeditev verjetnost za prehajanje med stanji in $p(x_{k-1}|z_{1:k-1}, u_{1:k-1}$ je korekcijska ocena porazdelitve verjetnosti za stanja iz prejšnjega trenutka.

Korekcijski korak

Oceno stanj po opravljeni meritvi v trenutku k in predhodno izračunani predikciji določimo z

$$p(x_k|z_{1:k}, u_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1}, u_{1:k})}{p(z_k|z_{1:k-1}, u_{1:k})} \tag{9.10}$$

Verjetnost $p(z_k|z_{1:k-1}, u_{1:k})$, ki podaja zaupanje v opravljeno meritev, določimo s pomočjo

$$p(z_k|z_{1:k-1}, u_{1:k}) = \int p(z_k|x_k)p(x_k|z_{1:k-1}, u_{1:k})dx_k$$

Bayes_filter$(bel(x_{k-1}), u_k, z_k)$:
  $\alpha = 0$
  za vse x_k naredi
    $bel_p(x_k) = \int p(x_k|x_{k-1}, u_k) bel(x_{k-1}) dx_{k-1}$
    $bel'(x_k) = p(z_k|x_k) bel_p(x_k)$
    $\alpha = \alpha + bel'(x_k)$
  za vse x_k naredi
    $bel(x_k) = \frac{1}{\alpha} bel'(x_k)$
  vrni $bel(x_k)$

Tabela 9.2: Algoritem za Bayesov filter

Splošni algoritem za Bayesov filter

Algoritem za izvedbo filtra je podan v tabeli 9.2. Pogojno verjetnost korekcijskega dela $p(x_k|z_{1:k}, u_{1:k})$, ki podaja oceno porazdelitve verjetnosti za stanje ob poznanih akcijah in meritvah imenujemo zaupanje (ang. belief), kar označimo z

$$bel(x_k) = p(x_k|z_{1:k}, u_{1:k})$$

Pogojno verjetnost predikcije $p(x_k|z_{1:k-1}, u_{1:k-1})$ pa označimo z

$$bel_p(x_k) = p(x_k|z_{1:k-1}, u_{1:k})$$

Bajesov filter ocenjuje verjetnostno porazdelitev ocene stanja. V primeru, ko imamo na voljo informacijo o akciji u_k, ki jo bomo izvedli izvajamo predikcijski korak, ko pa nam je znana še meritev izvajamo korekcijski korak. Vpeljemo še normirni faktor $\eta = \frac{1}{\alpha} = \frac{1}{p(z_k|z_{1:k-1}, u_{1:k})}$. S podanim algoritmom za Bayesov filter (tabela 9.2) najprej izvedemo predikcijo, nato določimo korekcijo ter jo ustrezno normiramo.

Kot lahko vidimo iz tabele 9.2, potrebujemo za izračun porazdelitve v predikcijskem koraku rešiti integral. Slednje omejuje praktično uporabo Bayesovega filtra na enostavne zvezne primere okolja, kjer je eksplicitna rešitev integrala možna ali pa za diskretne primere s končnim številom stanj, kjer integral zamenja vsota.

Primer 9.7. *Robot ima senzor za detekcijo odprtosti vrat ($Z \in (odprta, zaprta)$) in aktuator s pomočjo katerega lahko vrata potisne, da se odprejo oziroma aktuatorja ne uporabi, če meni, da so vrata že odprta ($U \in (potisni, miruj)$). Stanje, ki nas zanima je odprtost vrat $X \in (odprta, zaprta)$.*

Začetna verjetnost (zaupanje), da so vrata odprta je

$$bel(X_0 = open) = 0.5$$

veljavnost meritev senzorja je podana s statističnim modelom senzorja

$$P(Z_k = odprta|X_k = odprta) = 0.8 \quad P(Z_k = zaprta|X_k = odprta) = 0.2$$
$$P(Z_k = zaprta|X_k = zaprta) = 0.9 \quad P(Z_k = odprta|X_k = zaprta) = 0.1$$

Uspeh opravljene akcije potiska vrat pa

$$P(X_k = odprta|X_{k-1} = zaprta, U_k = potisni) = 0.8$$
$$P(X_k = zaprta|X_{k-1} = zaprta, U_k = potisni) = 0.2$$
$$P(X_k = odprta|X_{k-1} = odprta, U_k = potisni) = 1$$
$$P(X_k = zaprta|X_{k-1} = odprta, U_k = potisni) = 0$$

v kolikor pa aktuator ne uporabimo pa velja

$$P(X_k = odprta|X_{k-1} = zaprta, U_k = miruj) = 0$$
$$P(X_k = zaprta|X_{k-1} = zaprta, U_k = miruj) = 1$$
$$P(X_k = odprta|X_{k-1} = odprta, U_k = miruj) = 1$$
$$P(X_k = zaprta|X_{k-1} = odprta, U_k = miruj) = 0$$

Predpostavimo, da robot najprej opravi akcijo in nato prejme meritev. Določite zaupanja na osnovi opravljene akcije $bel_p(x_k)$ (predikcija) in zaupanje na osnovi meritve $bel(x_k)$ (korekcija) za sledeče sekvence opravljenih akcij in zaznanih meritev

	U_k	Z_k
$k = 1$	$miruj$	$zaprta$
$k = 2$	$potisni$	$odprta$
$k = 3$	$potisni$	$odprta$

Rešitev

Za boljšo preglednost označimo $X_k \in (odprta, zaprta)$ kot $X_k \in (x_k, \bar{x_k})$, $Z_k \in (odprta, zaprta)$ kot $Z_k \in (z_k, \bar{z_k})$ ter $U_k \in (potisni, miruj)$ kot $U_k \in (u_k, \bar{u_k})$.

Uporabimo algoritem v tabeli 9.2 in določimo za $k = 1$ in opravljeno akcijo $\bar{u_1} = miruj$ predikcijo zaupanja, da so vrata odprta

$$bel_p(x_1) = \sum_{X_0} p(x_1|x_0, \bar{u_1})bel(x_0) =$$
$$= P(x_1|\bar{x_0}, \bar{u_1})bel(\bar{x_0}) + P(x_1|x_0, \bar{u_1})bel(x_0) = 0 \cdot 0.5 + 1 \cdot 0.5 = 0.5$$

in predikcijo zaupanja, da so vrata zaprta

$$bel_p(\bar{x_1}) = \sum_{X_0} p(\bar{x_1}|x_0, \bar{u_1})bel(x_0) =$$
$$= P(\bar{x_1}|\bar{x_0}, \bar{u_1})bel(\bar{x_0}) + P(\bar{x_1}|x_0, \bar{u_1})bel(x_0) = 1 \cdot 0.5 + 0 \cdot 0.5 = 0.5$$

Vrednost zaupanja se torej ne spremeni, saj nismo izvajali akcije. Na podlagi meritve $\bar{z_1} = zaprta$ *pa določimo korekcije za zaupanja*

$$bel(x_1) = \eta p(\bar{z_1}|x_1) bel_p(x_1) = \eta 0.2 \cdot 0.5 = \eta 0.1$$

in

$$bel(\bar{x_1}) = \eta p(\bar{z_1}|\bar{x_1}) bel_p(\bar{x_1}) = \eta 0.9 \cdot 0.5 = \eta 0.45$$

Določimo še normirno konstanto η

$$\eta = \frac{1}{0.1 + 0.45} = 1.82$$

in končne vrednosti za zaupanji

$$bel(x_1) = 0.182 \quad bel(\bar{x_1}) = 0.818$$

Nadaljujemo še za $k = 2$ *in* $u_2 = potisni$ $z_2 = odprta$, *dobimo*

$$bel_p(x_2) = 0.8364 \quad bel_p(\bar{x_2}) = 0.1636$$

$$bel(x_2) = 0.9761 \quad bel(\bar{x_2}) = 0.0239$$

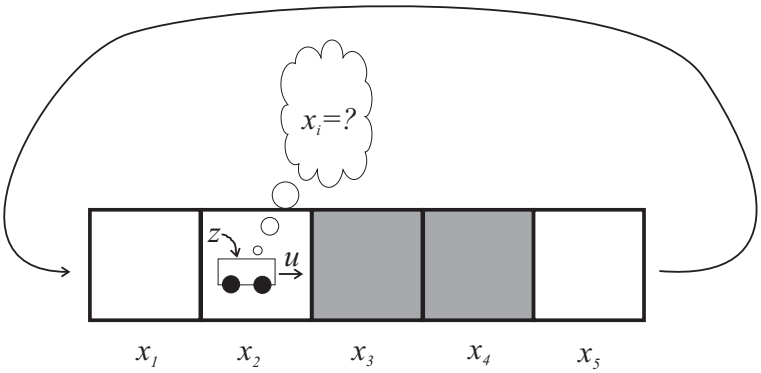
Nadaljujemo še za $k = 3$ *in* $u_3 = potisni$ $z_3 = odprta$, *dobimo*

$$bel_p(x_3) = 0.9952 \quad bel_p(\bar{x_3}) = 0.0048$$

$$bel(x_3) = 0.9994 \quad bel(\bar{x_3}) = 0.0006$$

9.3 Primer lokalizacije z uporabo Bayesovega filtra

Na preprostem primeru si poglejmo princip lokalizacije, ki predstavlja osnovno idejo Monte Carlo lokalizacijskega algoritma. Robot se premika v okolju in ga zaznava s senzorjem. Na podlagi informacij o opravljenih meritvah in premikih z algoritmom lokalizacije določimo lego robota. Pri tem uporabljamo teorem polne verjetnosti in Bayesovo pravilo, kar predstavlja osnovo za izvedbo Bayesovega filtra. V nadaljevanju najprej ločeno obravnavamo proces zaznavanja ob prisotnosti negotovosti senzorja ter proces izvajanja akcije ob prisotnosti negotovosti aktuatorja. Akcijo robot izvaja z namenom spreminjanja stanja v okolju (spreminjanje lege s premikanjem robota).



Slika 9.3: Okolje sestavljeno iz petih celic. Robot se lahko med celicami premika in zaznava njihovo barvo (svetlo, temno).

Primer 9.8. *Predpostavimo enodimenzionalno okolje sestavljeno iz petih svetlih ali temnih celic, v katerih se robot lahko nahaja, kot prikazuje slika 9.3. Gre torej za diskretno predstavitev okolja, kjer je osnovni element celica. Robot pozna zemljevid okolja, ne ve pa v kateri celici se nahaja. Začetno zaupanje v njegovo lokacijo je podano z uniformno porazdelitvijo verjetnosti, kjer je vsaka celica enako verjetna. Robot ima senzor s katerim lahko zazna ali se nahaja v svetli ali temni celici. Robot se lahko med celicami tudi premika, kjer se premik z neko verjetnostjo izvede pravilno oziroma nepravilno (preveč ali premalo). Okolje je ciklično, torej ko robot pride do konca se vrne na začetek in obratno. V nadaljevanju si na omenjenem okolju poglejmo ločeno proces zaznavanja okolja in gibanja v okolju.*

9.3.1 Zaznavanje okolja

S pomočjo meritve opravljene v okolici lahko izboljšamo oceno stanja X (npr. lokacijo) v tej okolici.

Zamislimo si, da ponoči tavamo po hiši, ker nas nosi luna. Ko se med tem prebudimo lahko ugotovimo kje se nahajamo s svojimi tipali (oči, otip,...).

Matematično je začetno vedenje o oklici, podano s porazdelitvijo $p(x)$ (prior). To porazdelitev lahko ob nastopu meritve Z (z zaupanjem $p(z|x)$) izboljšamo z izračunom porazdelitve verjetnosti $p(x|z)$ po opravljeni meritvi. Pri tem si pomagamo z Bayesovim izrekom $p(x|z) = bel(x) = \frac{p(z|x)p(x)}{p(z)}$. Kjer $p(z|x)$ predstavlja (statistični) model senzorja, $bel(x)$ pa je zaupanje v oceno stanja po opravljeni meritvi. Pri procesu zaznavanja torej izvajamo korekcijski del Bayesovega filtra.

Primer 9.9. *Za primer 9.8 predpostavimo, da robot zazna temno celico $Z = temna$. Robot pravilno zazna temno barvo z verjetnostjo 0.6, z verjetnostjo 0.2 pa naredi napako in svetlo celico razpozna kot temno, kar opišemo z*

$$\begin{aligned} p(Z = temna|X = x_t) &= 0.6 \quad r \in 3, 4 \\ p(Z = temna|X = x_s) &= 0.2 \quad g \in 1, 2, 5 \end{aligned}$$

kjer indeks t označuje temne celice, indeks s pa svetle celice. Robot ne ve, kje se nahaja, kar opišemo z uniformo porazdelitvijo verjetnosti $p(X = x_i) = bel(x_i) = 0.2$, $i \in 1 \cdots 5$. Kakšna je porazdelitev verjetnosti o lokaciji po opravljeni meritvi?

Rešitev

Zanima nas porazdelitev pogojne verjetnosti $p(X_1|Z = temna)$, torej zaupanje v oceno stanja po opravljeni meritvi. Določimo jo z Bayesovim izrekom, ki predstavlja korekcijski del Bayesovega filtra

$$\begin{aligned} p(X_1|Z = temna) &= \frac{p(Z=temna|X_1=x_i)p(X_1=x_i)}{P(Z=temna)} \\ &= \frac{[0.2,0.2,0.6,0.6,0.2]\star[0.2,0.2,0.2,0.2,0.2]}{P(Z=temna)} \\ &= \frac{[0.040.040.120.120.04]}{P(Z=temna)} \end{aligned}$$

kjer $\star$ predstavlja množenje istoležnih elementov vektorja.

Določiti moramo še verjetnost, da zaznamo temno barvo $P(Z = temna) = \eta$, kar predstavlja normirni člen η. Iščemo polno verjetnost, torej seštejemo verjetnosti zaznave temne barve pri vseh poljih

$$\begin{aligned} P(Z = temna) &= \textstyle\sum_i p(Z = temna|X_1 = x_i)p(X_1 = x_i) \\ &= [0.2, 0.2, 0.6, 0.6, 0.2][0.2, 0.2, 0.2, 0.2, 0.2]^T = 0.36 \end{aligned}$$

Iskana (posteriorna) porazdelitev torej je

$$p(X_1|Z = temna) = [0.11, 0.11, 0.33, 0.33, 0.11]$$

od koder lahko sklepamo, da se robot s trikrat večjo verjetnostjo nahaja v celici 3 ali 4, kot pa v celicah 1, 2 ali 5.

Primer 9.10. *Za primer 9.9 odgovorite:*

1. *Ali lahko večkratna zaporedna meritev izboljša vedenje o lokaciji robota (robot se med tem ne premika)?*
2. *Kakšna je porazdelitev verjetnosti za lego robota, če robot dvakrat zapored izmeri temno barvo?*

3. *Kakšna je porazdelitev verjetnosti za lego robota, če robot izmeri temno in nato svetlo?*

4. *Kakšna je porazdelitev verjetnosti za lego robota, če robot izmeri temno, svetlo in nato temno?*

Rešitev

1. *Večkratna zaporedna meritev izboljša vedenje o lokaciji robota, če je verjetnost pravilne meritve večja od verjetnosti napake pri meritvi.*

2. $$p(X_2|Z_1 = temna, Z_2 = temna) = p(X_2|z_1, z_2) = \frac{p(z_2|X_2)p(X_2|z_1)}{P(z_2|z_1)} = \\ = \frac{[0.2,0.2,0.6,0.6,0.2]\star[0.11,0.11,0.33,0.33,0.11]}{P(z_2|z_1)}$$

   kjer je $p(X_2|z_1) = \sum_{X_1} P(X_2|X_1)P(X_1|z_1) = p(X_1|z_1)$ *ker na stanja ne vplivamo (glej primer 9.5), ampak jih le merimo. Pogojno verjetnost v imenovalcu pa določimo z*

   $$P(z_2|z_1) = \sum_{X_2} p(z_2|X_2)p(X_2|z_1)) \\ = [0.2, 0.2, 0.6, 0.6, 0.2][0.11, 0.11, 0.33, 0.33, 0.11]^T = 0.462$$

   Torej končno imamo

   $$p(X_2|z_1, z_2) = \frac{[0.2,0.2,0.6,0.6,0.2]\star[0.11,0.11,0.33,0.33,0.11]}{[0.2,0.2,0.6,0.6,0.2][0.11,0.11,0.33,0.33,0.11]^T} = \\ = [0.0476, 0.0476, 0.4286, 0.4286, 0.0476]$$

3. *Svetlo barvo zaznamo pravilno z verjetnostjo* $p(Z = svetla|X = svetla) = 1 - p(Z = temna|X = svetla) = 0.8$ *in narobe z verjetnostjo* $p(Z = svetla|X = temna) = 1 - p(Z = temna|X = temna) = 0.4$. *Za drugo meritev izhajamo iz porazdelitve* $p(x_i|Z_1 = temna)$ *in določimo*

   $$p(X_2|Z_1 = temna, Z_2 = svetla) = p(X_2|z_1, z_2) = \\ = \frac{[0.8,0.8,0.4,0.4,0.8]\star[0.11,0.11,0.33,0.33,0.11]}{P(Z_2=svetla|Z_1=temna)}$$

   $$p(Z_2 = svetla|Z_1 = temna) = \\ = \sum_{X_2} p(Z_2 = svetla|X_2)p(X_2|Z_1 = temna) = \\ = [0.8, 0.8, 0.4, 0.4, 0.8][0.11, 0.11, 0.33, 0.33, 0.11]^T = 0.528$$

   $$p(X_2|Z_1 = temna, Z_2 = svetla) = [0.167, 0.167, 0.25, 0.25, 0.167]$$

4. $$p(X_3|Z_1 = temna, Z_2 = svetla, Z_3 = temna) = \\ = [0.083, 0.083, 0.375, 0.375, 0.083]$$

9.3.2 Gibanje v okolju

Gibanje v okolici je izvedeno s pomočjo aktuatorjev (npr. koles) in algoritma. Pri vsakem premiku je prisotna manjša ali večja negotovost, kar pomeni, da gibanje povečuje negotovost o našem stanju (legi) v okolici.

Zamislimo si sebe, ko se nahajamo v nekem znanem prostoru. Zapremo oči in naredimo nekaj korakov po prostoru. Približno vemo, kako velike korake smo delali in v katero smer in si zato predstavljamo, kje v prostoru se nahajamo. A dejstvo je, da naš korak nikoli ni točno tako dolg in v tisto smer, kot smo si zamislili, zato se s časom naše vedenje o lokaciji poslabšuje.

Relacijo 9.9 lahko za primer, ko se v okolici le gibljemo brez zaznavanja stanj, nekoliko preuredimo

$$p(x_k|, u_{1:k}) = \sum_{x_{k-1}} p(x_k|x_{k-1}, u_k)p(x_{k-1}|, u_{1:k-1})$$

torej je zaupanje v novo lokacijo $p(x_k|u_{1:k})$ odvisno od zaupanja v prejšnjem trenutku $p(x_{k-1}|u_{1:k-1})$ in pogojne verjetnosti za prehod med stanji $p(x_k|x_{k-1}, u_k)$. Porazdelitev verjetnost $p(x_k|, u_{1:k})$ določimo tako, da izvedemo integral oziroma sumacijo (za diskretni opis okolja) kjer seštejemo produkte vseh možnih prihodov $p(x_k|x_{k-1}, u_k)$ iz predhodnih stanj x_{k-1} v stanje x_k pri poznani akciji u_k.

Primer 9.11. *Za primer 9.8 predpostavimo, da se robot na začetku nahaja v prvi celici ($X_0 = 1$) kar je podano z porazdelitvijo $p(X_0) = [1, 0, 0, 0, 0]$. Robot se med celicami lahko premika, izid akcije premika je pravilen v 80%, v 10% se robot premakne premalo za eno celico in v 10% se robot premakne preveč za eno celico. Prehod med stanji je torej podan z*

$$\begin{aligned} p(X_k = i|X_{k-1} = j, U_k = u) &= 0.8 \quad za \quad i = j + u \\ p(X_k = i|X_{k-1} = j, U_k = u) &= 0.1 \quad za \quad i = j + u - 1 \\ p(X_k = i|X_{k-1} = j, U_k = u) &= 0.1 \quad za \quad i = j + u + 1 \end{aligned}$$

Robot naredi premik za dve celico v desno $U_1 = 2$. Določite zaupanje v lego robota po premiku.

Rešitev

Porazdelitev verjetnost (zaupanje) po premiku določimo tako, da za vsako celico izračunamo verjetnost, da se robot po premiku nahaja v njej (polna verjetnost). V prvo celico lahko očitno pridemo le iz celice 3 (premik preveč), 4 (pravi premik) in 5 (premik premalo), torej je porazdelitev verjetnosti za prehod v prvo celico podana z $p(X_1 = 1|X_0, U_1 = 2) = [0, 0, 0.1, 0.8, 0.1]$ in verjetnost, da se po premiku nahajamo v prvi celici

$$\begin{aligned} P(X_1 = 1|, U_1 = 2) = \textstyle\sum_{X_0} p(X_1 = 1|X_0, U_1 = 2)p(X_0) = \\ = [0, 0, 0.1, 0.8, 0.1][1, 0, 0, 0, 0]^T = 0 \end{aligned}$$

Verjetnost, da se po premiku nahajamo v drugi celici je

$$P(X_1 = 2|, U_1 = 2) = \sum_{X_0} p(X_1 = 2|X_0, U_1 = 2)p(X_0) = \\ = [0.1, 0, 0, 0.1, 0.8][1, 0, 0, 0, 0]^T = 0.1$$

podobno določimo verjetnosti še za ostale celice

$$P(X_1 = 3|, U_1 = 2) = [0.8, 0.1, 0, 0, 0.1][1, 0, 0, 0, 0]^T = 0.8$$
$$P(X_1 = 4|, U_1 = 2) = [0.1, 0.8, 0.1, 0, 0][1, 0, 0, 0, 0]^T = 0.1$$
$$P(X_1 = 5|, U_1 = 2) = [0, 0.1, 0.8, 0.1, 0][1, 0, 0, 0, 0]^T = 0$$

Zaupanje v lego po premiku torej je

$$P(X_1|, U_1 = 2) = [0, 0.1, 0.8, 0.1, 0]$$

Primer 9.12. *Kakšno je zaupanje v lego, če po premiku iz primera 9.11 izvedemo še premik za eno celico v desno $U_2 = 1$?*

Rešitev

Porazdelitev verjetnost (zaupanje) po premiku določimo tako, da za vsako celico izračunamo verjetnost, da se robot po premiku nahaja v njej (polna verjetnost). V prvo celico lahko očitno pridemo le iz celice 1 (premik premalo), 4 (premik preveč) in 5 (pravi premik preveč), v drugo celico lahko pridemo iz celic 1,2,5 in podobno za ostale celice. Izračunamo verjetnosti za nahajanje v celicah po premiku

$$P(X_2 = 1|, U_1 = 2, U_2 = 1) = [0.1, 0, 0, 0.1, 0.8][0, 0.1, 0.8, 0.1, 0]^T = 0.01$$
$$P(X_2 = 2|, U_1 = 2, U_2 = 1) = [0.8, 0.1, 0, 0, 0.1][0, 0.1, 0.8, 0.1, 0]^T = 0.01$$
$$P(X_2 = 3|, U_1 = 2, U_2 = 1) = [0.1, 0.8, 0.1, 0, 0][0, 0.1, 0.8, 0.1, 0]^T = 0.16$$
$$P(X_2 = 4|, U_1 = 2, U_2 = 1) = [0, 0.1, 0.8, 0.1, 0][0, 0.1, 0.8, 0.1, 0]^T = 0.66$$
$$P(X_2 = 5|, U_1 = 2, U_2 = 1) = [0, 0, 0.1, 0.8, 0.1][0, 0.1, 0.8, 0.1, 0]^T = 0.16$$

Zaupanje v lego po premiku torej je

$$P(X_2|, , U_1 = 2, U_2 = 1) = [0.01, 0.01, 0.16, 0.66, 0.16]$$

Opazimo, da se robot z največjo verjetnostjo nahaja v celici 4. Ugotovimo, da porazdelitev verjetnosti nima tako izrazitega maksimuma kot pred premikom (iz 80% se zmanjša na 66%). Vsak premik namreč povečuje negotovost o stanju v okolici.

Primer 9.13. *Robot iz primera 9.11 se na začetku nahaja v prvi celici* $p(X_0) = [1, 0, 0, 0, 0]$. *Nato se v vsakem časovnem trenutku premakne za eno celico v desno.*

1. *Kakšno je zaupanje v lego robota po desetem premiku?*
2. *Kakšna je limitna vrednost zaupanja v lego robota, ko robot izvede neskončno premikov?*

Rešitev

1. $$P(X_{10}|, U_{1:10}) = [0.29, 0.22, 0.13, 0.13, 0.22]$$
2. *Po neskončno premikih dobimo uniformno porazdelitev, kjer so vse celice enako verjetne*
$$P(X_{inf}|, U_{1:inf}) = [0.2, 0.2, 0.2, 0.2, 0.2]$$

9.3.3 Lokalizacija v okolju

Robot lahko ugotovi svojo lokacijo v okolju, katerega zemljevid ima poznan, ne pozna pa svoje začetne lege v okolju. Svojo lokacijo lahko ugotovi z neko verjetnostno porazdelitvijo natančno. Proces ugotavljanja lokacije imenujemo lokalizacija. Lokalizacija združuje proces zaznavanja in premikanja. Kot smo že ugotovili vsaka nova meritev okolice povečuje znanje o svoji lokaciji, s premikanjem pa se to znanje zmanjšuje.

Pri lokalizaciji mobilni robot stalno posodablja porazdelitev verjetnosti, ki predstavlja vedenje o svoji lokaciji v okolici. Maximum porazdelitve verjetnosti predstavlja najbolj verjetno lokacijo robota.

Pri procesu lokalizacije izvajamo Bayesov filter (tabela 9.2), kar združuje proces premikanja in zaznavanja.

Primer 9.14. *Robot se premika v oklici iz primera 9.8. najprej izvede premik, nato zaznava okolico. Začetna lega robota ni poznana, kar opiše uniformna porazdelitev* $p(X_0) = bel(x_0) = [0.2, 0.2, 0.2, 0.2, 0.2]$.

Premik za u_k *celic v desno je v 80% uspešen v 10% pa je za eno celico prekratek ali predolg, kar opiše*

$$\begin{aligned} p(X_k = i|X_{k-1} = j, U_k = u) &= 0.8 \quad za \quad i = j + u \\ p(X_k = i|X_{k-1} = j, U_k = u) &= 0.1 \quad za \quad i = j + u - 1 \\ p(X_k = i|X_{k-1} = j, U_k = u) &= 0.1 \quad za \quad i = j + u + 1 \end{aligned}$$

Robot pravilno zazna temno barvo z verjetnostjo 0.6, svetlo barvo pa zazna pravilno z 0.8 verjetnostjo, kar opišemo z

$$p(Z = temna|X = temna) = 0.6 \quad p(Z = svetla|X = temna) = 0.4$$
$$p(Z = svetla|X = svetla = 0.8 \quad p(Z = temna|X = svetla = 0.2$$

Robot v vsakem časovnem trenutku dobi ukaz za premik za eno celico v desno ($u_k = 1$). Izid meritev za prve tri trenutke so $z_{1:3} = [svetla, temna, temna]$.

1. *Kakšno je zaupanje v prvem koraku $k = 1$?*
2. *Kakšno je zaupanje v drugem koraku $k = 2$?*
3. *Kakšno je zaupanje v tretjem koraku $k = 3$?*
4. *V kateri celici se robot nahaja z največjo verjetnostjo po tretjem koraku?*

Rešitev

Pri vsakem premiku izvajamo predikcijski del Bayesovega filtra, pri zaznavi pa korekcijski del (tabela 9.2).

1. *Predikcijski del na osnovi izvedenega premika, kjer z malim x_i, $i \in 1, \cdots, 5$ označimo nahajanje v celici i, veliki X_k pa označuje vektor vseh stanj v trenutku k.*

$$\begin{aligned}
bel_p(x_1) &= \textstyle\sum_i p(x_1|x_i, u_1)bel(x_i) = \\
&= P(x_1|x_1, u_1)bel(x_1) + P(x_1|x_2, u_1)bel(x_2) + P(x_1|x_3, u_1)bel(x_3) + \\
&\quad + P(x_1|x_4, u_1)bel(x_4) + P(x_1|x_5, u_1)bel(x_5) \\
&= [0.1, 0, 0, 0.1, 0.8][0.2, 0.2, 0.2, 0.2, 0.2]^T = 0.2 \\
bel_p(x_2) &= [0.8, 0.1, 0, 0, 0.1][0.2, 0.2, 0.2, 0.2, 0.2]^T = 0.2 \\
bel_p(x_3) &= [0.1, 0.8, 0.1, 0, 0][0.2, 0.2, 0.2, 0.2, 0.2]^T = 0.2 \\
bel_p(x_4) &= [0, 0.1, 0.8, 0.1, 0][0.2, 0.2, 0.2, 0.2, 0.2]^T = 0.2 \\
bel_p(x_5) &= [0, 0, 0.1, 0.8, 0.1][0.2, 0.2, 0.2, 0.2, 0.2]^T = 0.2
\end{aligned}$$

Torej celotna porazdelitev za predikcijski del

$$bel_p(X_1) = [0.2, 0.2, 0.2, 0.2, 0.2]$$

Na osnovi meritve meritve barve pa izvajamo korekcijo

$$\begin{aligned}
bel(x_1) &= \eta p(Z_1 = svetla|x_1)bel_p(x_1) = \eta 0.8 \cdot 0.2 = \eta 0.16 \\
bel(x_2) &= \eta p(Z_1 = svetla|x_2)bel_p(x_2) = \eta 0.8 \cdot 0.2 = \eta 0.16 \\
bel(x_3) &= \eta p(Z_1 = svetla|x_3)bel_p(x_3) = \eta 0.4 \cdot 0.2 = \eta 0.08 \\
bel(x_4) &= \eta p(Z_1 = svetla|x_4)bel_p(x_4) = \eta 0.4 \cdot 0.2 = \eta 0.08 \\
bel(x_5) &= \eta p(Z_1 = svetla|x_5)bel_p(x_5) = \eta 0.8 \cdot 0.2 = \eta 0.16
\end{aligned}$$

$$\eta = \frac{1}{0.16 + 0.16 + 0.08 + 0.08 + 0.16} = 1.56$$

vektor porazdelitve torej je

$$bel(X_1) = [0.25, 0.25, 0.125, 0.125, 0.25]$$

kar bi lahko kompaktneje določili tudi kot

$$\begin{aligned} bel(X_1) &= \frac{[0.8,0.8,0.4,0.4,0.8] \star [0.2,0.2,0.2,0.2,0.2]}{[0.8,0.8,0.4,0.4,0.8][0.2,0.2,0.2,0.2,0.2]^T} \\ &= [0.25, 0.25, 0.125, 0.125, 0.25] \end{aligned}$$

2.

$$\begin{aligned} bel_p(x_1) &= [0.1, 0, 0, 0.1, 0.8][0.25, 0.25, 0.125, 0.125, 0.25]^T = 0.237 \\ bel_p(x_2) &= [0.8, 0.1, 0, 0, 0.1][0.25, 0.25, 0.125, 0.125, 0.25]^T = 0.25 \\ bel_p(x_3) &= [0.1, 0.8, 0.1, 0, 0][0.25, 0.25, 0.125, 0.125, 0.25]^T = 0.237 \\ bel_p(x_4) &= [0, 0.1, 0.8, 0.1, 0][0.25, 0.25, 0.125, 0.125, 0.25]^T = 0.138 \\ bel_p(x_5) &= [0, 0, 0.1, 0.8, 0.1][0.25, 0.25, 0.125, 0.125, 0.25]^T = 0.138 \end{aligned}$$

Torej celotna porazdelitev za predikcijski del

$$bel_p(X_2) = [0.237, 0.25, 0.237, 0.138, 0.138]$$

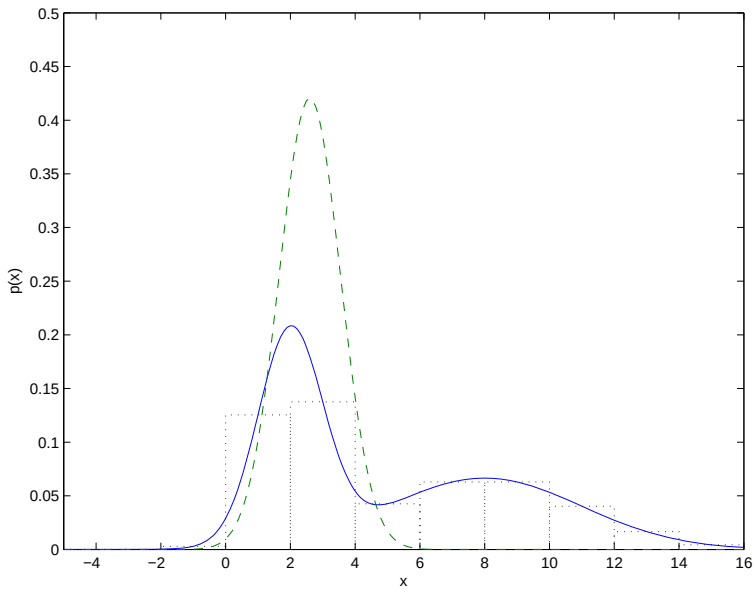
$$\begin{aligned} bel(X_2) &= \frac{[0.2,0.2,0.6,0.6,0.2] \star [0.237,0.25,0.237,0.138,0.138]}{[0.2,0.2,0.6,0.6,0.2][0.237,0.25,0.237,0.138,0.138]^T} \\ &= [0.136, 0.143, 0.407, 0.236, 0.079] \end{aligned}$$

3.

$$bel_p(X_3) = [0.1, 0.131, 0.167, 0.363, , 0.237]$$

$$bel(X_3) = [0.048, 0.063, 0.245, 0.528, 0.115]$$

4. *Po tretjem koraku se robot najbolj verjetno nahaja v četrti celici z verjetnostjo 52.8%. Druga najbolj verjetna celica je tretja celica, kjer se robot nahaja z verjetnostjo 24.5%.*



Slika 9.4: Primer porazdelitev verjetnosti spremenljivke x (polno), aproksimacija z Gaussovo funkcijo (črtkano) in aproksimacija s histogramom (pikčasto).

9.4 Kalmanov filter

Dejansko porazdelitev verjetnosti pri Kalmanovem filtru aproksimiramo z Gaussovo funkcijo

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} \tag{9.11}$$

kjer je μ srednja vrednost (matematično upanje) in σ^2 varianca. Gaussova funkcija je unimodalna (levo in desno od maksimuma funkcija pada proti nič) za razliko od splošnih porazdelitev, ki so lahko več modalne. Pri Kalmanovem filtru torej predpostavimo, da so porazdelitve verjetnosti zveznih spremenljivk unimodalne. V kolikor spremenljivke niso unimodalne, je ocena dobljenih stanj neoptimalna, vprašljiva pa je tudi konvergenca algoritma k pravi rešitvi. Bayesov filter teh problemov nima, je pa zaradi tega omejen le na enostavna zvezna okolja ali diskretna okolja s končnim številom stanj.

Slika 9.4 prikazuje primer porazdelitve verjetnosti (ki ni unimodalna) zvezne spremenljivke, ki jo aproksimiramo z Gaussovo funkcijo in s histogramom (prostor razdelimo na diskretna področja). Aproksimacija z Gaussovo funkcijo se uporablja pri Kalmanovem filtru, histogram pa pri Bayesovem filtru.

Bistvo korekcijskega koraka (glej Bayesov filter 9.10) je združevanje informacije dobljene iz dveh neodvisnih virov, to je meritev senzorja in predikcija stanja na osnovi pretekle ocene stanja. Na primeru 9.15 si poglejmo, kako optimalno združiti dve neodvisni oceni za isto veličino x (stanje), če imamo za vsak vir podano vrednost in varianco (zaupanje).

Primer 9.15. *Imamo dve neodvisni oceni spremenljivke x. Prva ocena je podana z vrednostjo x_1 in varianco σ_1^2, druga ocena pa z vrednostjo x_2 in varianco σ_2^2. Kakšna je optimalna linearna kombinacija teh dveh ocen, ki predstavlja oceno stanja $\hat{x}$?*

Rešitev

Zanima nas

$$\hat{x} = \omega_1 x_1 + \omega_2 x_2$$

kjer sta ω_1 in ω_2 iskani uteži, za kateri velja $\omega_1 + \omega_2 = 1$. Optimalni vrednosti uteži minimizirata varianco σ^2 ocene $\hat{x}$. Varianca ocene zapišemo kot

$$\begin{aligned}
\sigma^2 &= E\left[(\hat{x} - E[\hat{x}])^2\right] \\
&= E\left[(\omega_1 x_1 + \omega_2 x_2 - E[\omega_1 x_1 + \omega_2 x_2])^2\right] \\
&= E\left[(\omega_1 x_1 + \omega_2 x_2 - \omega_1 E[x_1] - \omega_2 E[x_2])^2\right] \\
&= E\left[\left(\omega_1 \left(x_1 - E[x_1]\right) + \omega_2 \left(x_2 - E[x_2]\right)\right)^2\right] \\
&= E\left[\omega_1^2 \left(x_1 - E[x_1]\right)^2 + \omega_2^2 \left(x_2 - E[x_2]\right)^2 + 2\omega_1\omega_2 (x_1 - E[x_1])(x_2 - E[x_2])\right] \\
&= \omega_1^2 E[(x_1 - E[x_1])^2] + \omega_2^2 E[(x_2 - E[x_2])^2] + 2\omega_1\omega_2 E[(x_1 - E[x_1])(x_2 - E[x_2])] \\
&= \omega_1^2\sigma_1^2 + \omega_2^2\sigma_2^2 + 2\omega_1\omega_2 E[(x_1 - E[x_1])(x_2 - E[x_2])]
\end{aligned}$$

ker sta x_1 in x_2 neodvisna, sta neodvisna tudi $x_1 - E[x_1]$ in $x_2 - E[x_2]$ in je zato $E[(x_1 - E[x_1])(x_2 - E[x_2]) = 0$. Imamo torej

$$\sigma^2 = \omega_1^2\sigma_1^2 + \omega_2^2\sigma_2^2$$

nadalje označimo $\omega_2 = \omega$ in $\omega_1 = 1 - \omega$

$$\sigma^2 = (1 - \omega)^2\sigma_1^2 + \omega^2\sigma_2^2$$

Iščemo vrednost uteži ω pri katerih je varianca minimalna

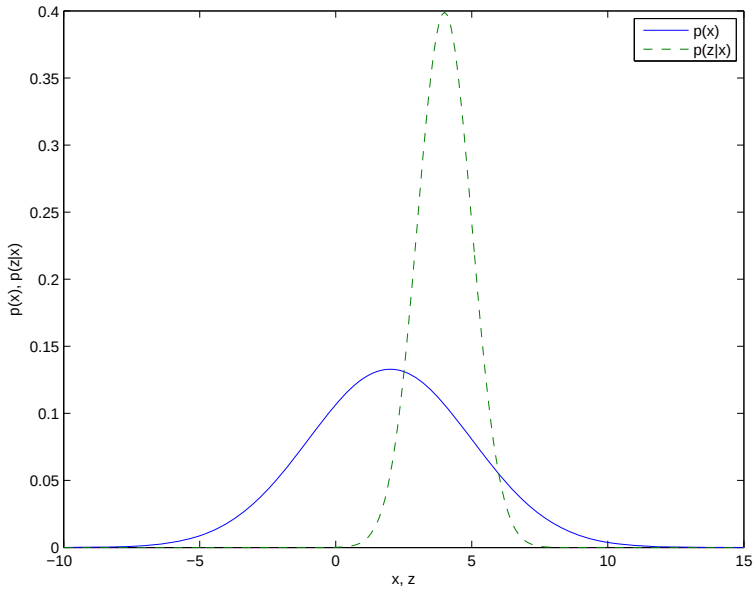
$$\frac{d}{d\omega}\sigma^2 = -2(1 - \omega)\sigma_1^2 + 2\omega\sigma_2^2 = 0$$

kjer je rešitev

$$\omega = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}$$

Končna ocena je

$$\hat{x} = \frac{\sigma_2^2 x_1 + \sigma_1^2 x_2}{\sigma_1^2 + \sigma_2^2}$$



Slika 9.5: Porazdelitev verjetnosti spremenljivke x (polno) in meritve (črtkano).

in varianca ocene

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} = \left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}\right)^{-1}$$

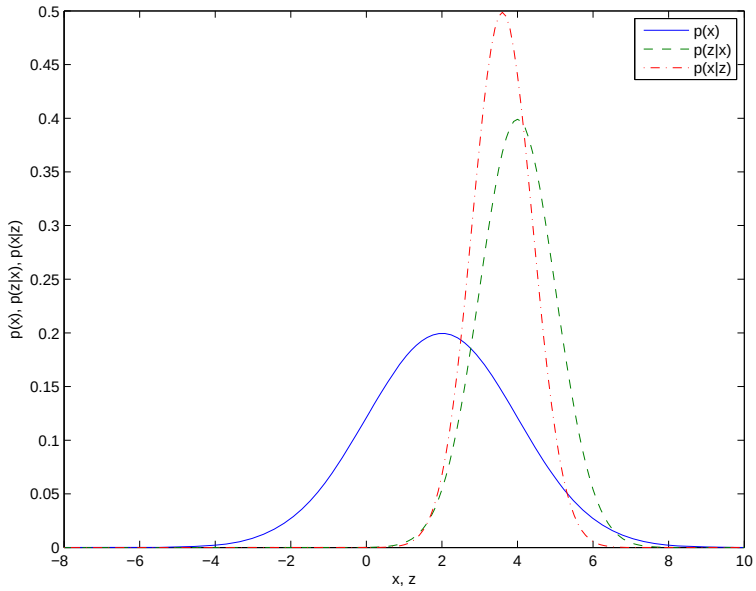
Iz dobljenih rezultatov vidimo, da vir, ki ima manjšo varianco (večje zaupanje) bolj upoštevamo pri končni oceni in obratno.

Primer 9.16. *V nekem trenutku imamo podano izhodiščno oceno stanja $x = 2$ z varianco $\sigma^2 = 4$. Nato s senzorjem izmerimo vrednost stanja, ki je podana z vrednostjo $z = 4$ in varianco $\sigma_z^2 = 1$. Gaussovi porazdelitvi stanja in meritve verjetnosti ilustrira slika 9.5.*

Kakšna nova optimalna ocena stanja, ki združuje informacijo predhodne ocene in meritve? Kakšna je porazdelitev verjetnosti nove optimalne ocene?
Rešitev

Iz slike 9.5 lahko ugotovimo, da bo nova srednja vrednost x' bližje meritvi, ker ima le-ta manjšo varianco (negotovost).

$$x' = \frac{\sigma_z^2 x + \sigma^2 z}{\sigma^2 + \sigma_z^2} = 3.6$$



Slika 9.6: Porazdelitev verjetnosti spremenljivke x (polno), meritve (črtkano) in posodobljene spremenljivke x' (črta pika).

Standardna deviacija nove ocene σ' pa je manjša od obeh predhodnih varianc, saj z integracijo obeh predhodnih informacij zmanjšujemo negotovost nove ocene. Varianca nove ocene je

$$\sigma^{2\prime} = \left(\frac{1}{\sigma^2} + \frac{1}{\sigma_z^2}\right)^{-1} = 0.8$$

oziroma standardna deviacija

$$\sigma' = \sqrt{\sigma^{2\prime}} = 0.894$$

Nova porazdelitev verjetnosti $p(x'|z)$ stanja po opravljeni korekciji na osnovi meritve je prikazana na sliki 9.6

Uporabimo ugotovitve primera 9.15 za rekurzivno določitev ocene nekega stanja. V vsakem trenutku prejmemo meritev stanja $z(k) = x(k) + n(k)$ s pomočjo senzorja, kjer je $n(k)$ šum meritve. Vsaka meritev ima podano varianco $\sigma_z(k)$. Nova optimalna ocena stanja je kombinacija prejšnje ocene stanja $\hat{x}(k)$ ter meritve $z(k)$ kot sledi

$$\hat{x}(k+1) = (1-\omega)\,\hat{x}(k) + \omega(k)z(k) = \hat{x}(k) + \omega(z(k) - \hat{x}(k))$$

varianca za novo oceno je

$$\sigma^2(k+1) = \frac{\sigma^2(k)\sigma_z^2(k)}{\sigma^2(k)+\sigma_z^2(k)} = (1-\omega)\sigma^2(k)$$

kjer je

$$\omega = \frac{\sigma^2(k)}{\sigma^2(k)+\sigma_z^2(k)}$$

Torej za podano začetno oceno stanja x, $\hat{x}(0)$ in njeno varianco $\sigma^2(0)$ lahko ocenimo optimalno združujemo meritve $z(1)$, z(2), ..., da ocenimo trenutno vrednost stanja in njegovo varianco. To predstavlja osnovno idejo korekcijskega dela Kalmanovega filtra.

Predikcijski del Kalmanovega filtra pa podaja predikcijo stanja ob znanem premiku stanja. Izhodiščno ocena stanje $\hat{x}(k)$ je podano z neko porazdelitvijo verjetnosti z varianco $\sigma^2(k)$. Ravno tako je akcija $u(k)$, ki izvede premik stanja iz $x(k)$ v $x(k+1)$, podana z neko porazdelitvijo verjetnosti (negotovost premika) $\sigma_u^2(k)$. Na primeru 9.17 si poglejmo kakšna je vrednost stanja in njegova varianca po izvedeni akciji (premiku stanja).

Primer 9.17. *Imamo izhodiščno oceno stanja $\hat{x}(k)$ z $\sigma^2(k)$. Nato izvedemo akcijo u(k), ki predstavlja neposreden premik stanja z negotovostjo $\sigma_u^2(k)$. Kakšna je vrednost stanja po premiku in njegova negotovost?*

Rešitev

Nova ocena stanje po premiku je

$$\hat{x}(k+1) = \hat{x}(k) + u(k)$$

negotovost te ocene pa je

$$\begin{aligned}
\sigma^2(k+1) &= E\left[(\hat{x}(k+1) - E[\hat{x}(k+1)])^2\right] \\
&= E\left[(\hat{x}(k) + u(k) - E[\hat{x}(k) + u(k)])^2\right] \\
&= E\left[((\hat{x}(k) - E[\hat{x}(k)]) + (u(k) - E[u(k)]))^2\right] \\
&= E[(\hat{x}(k) - E[\hat{x}(k)])^2 + (u(k) - E[u(k)])^2 \\
&\quad +2(\hat{x}(k) - E[\hat{x}(k)])(u(k) - E[u(k)])] \\
&= \sigma^2(k) + \sigma_u^2(k) + E\left[2\left(\hat{x}(k) - E[\hat{x}(k)]\right)(u(k) - E[u(k)])\right]
\end{aligned}$$

ker sta $\hat{x}$ in u neodvisna je $E\left[2\left(\hat{x}(k) - E[\hat{x}(k)]\right)(u(k) - E[u(k)])\right] = 0$ in imamo

$$\sigma^2(k+1) = \sigma^2(k) + \sigma_u^2(k)$$

Kalman_filter$(\hat{x}_{k-1|k-1}, u_k, z_k)$:

izračun predikcije:

$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + u_k$

$\sigma^2_{k|k-1} = \sigma^2_{k-1|k-1} + \sigma_{u_k}^2$

izračun korekcije:

$\omega_k = \frac{\sigma^2_{k|k-1}}{\sigma^2_{k|k-1} + \sigma_{z_k}^2}$

$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \omega(z_k - \hat{x}_{k|k-1})$

$\sigma^2_{k|k} = (1 - \omega_k)\sigma^2_{k|k-1}$

Tabela 9.3: Algoritem za Kalmanov filter z enim stanjem

Algoritem za poenostavljeno izvedbo Kalmanovega filtra

Kalmanov filter za enostavni primer z enim stanjem je podan v tabeli 9.3, kjer veličine z oznako $_{k|k-1}$ predstavljajo ocenjene vrednosti iz predikcijskega dela, veličine z oznako $_{k|k}$ pa iz korekcijskega dela. Zaradi boljše preglednosti označimo še $u(k) = u_k$ in $z(k) = z_k$.

Kalmanov filter izvajamo v dveh korakih, predikciji in korekciji. V predikcijskem koraku uporabimo znano akcijo in določimo vrednost stanja v naslednjem trenutku. Torej iz začetnega zaupanja določimo novo zaupanje, ki ima večjo negotovost, kot začetno nezaupanje. V korekcijskem koraku pa uporabimo meritev, da iz prejšnjega zaupanja določimo novo zaupanje (v stanje), ki ima manjšo negotovost kot prejšnje zaupanje. Pri obeh korakih potrebujemo le dve stvari. Pri predikciji potrebujemo vrednost predhodnega zaupanja in izvedene akcije. Pri korekciji pa potrebujemo vrednost predhodnega zaupanja in meritev.

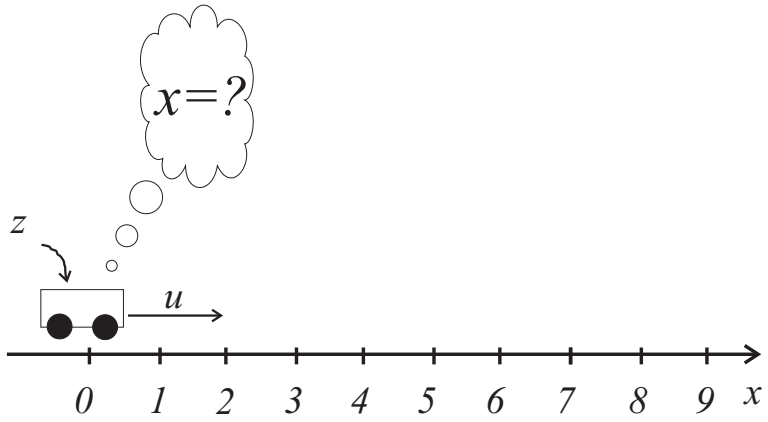
Primer 9.18. *Imamo robota, ki se premika v eni dimenziji. Njegova začetna lega nam je neznana, predvidevamo, da je* $\hat{x}_0 = 3$ *in ima veliko negotovost* $\sigma_0 = 100$ *(dejanska pozicija* $x_0 = 0$ *nam ni znana).*

Nato robot v vsakem trenutku $k = 1, \cdots 5$ *premaknemo za* $u_{1:5} = [2, 3, 2, 1, 1]$ *in nato izvedemo meritev lege* $z_{1:5} = [2, 5, 7, 8, 9]$. *Pomik in meritev sta motena s šumom z normalno porazdelitvijo, kar opišemo s konstantno negotovostjo za pomik* $\sigma_u^2 = 2$ *in negotovostjo meritve* $\sigma_z^2 = 4$.

Kakšna je ocena lege robota in negotovost te ocene?

Rešitev

Izvajamo algoritem v tabeli 9.3 in za prvi korak ($k = 1$*) dobimo predikcijo*



Slika 9.7: Lokalizacija robota, ki se premika v eni dimenziji in ne pozna začetne lege.

stanja in negotovosti

$$\hat{x}_{1|0} = \hat{x}_{0|0} + u_1 = 3 + 2 = 5$$
$$\sigma^2_{1|0} = \sigma^2_{0|0} + {\sigma_u}^2 = 100 + 2 = 102$$

za korekcijo pri $k = 1$ *pa dobimo*

$$\omega_1 = \frac{\sigma^2_{1|0}}{\sigma^2_{1|0} + {\sigma_z}^2} = \frac{102}{102+4} = 0.962$$
$$\hat{x}_{1|1} = \hat{x}_{1|0} + \omega(z_1 - \hat{x}_{1|0}) = 5 + 0.962(2-5) = 2.113$$
$$\sigma^2_{1|1} = (1-\omega_1)\sigma^2_{1|0} = (1-0.962)102 = 3.849$$

Podobno izvajamo še za ostale korake in dobimo za predikcijo

$$\hat{x}_{1:5|0:4} = [5.00, 5.11, 7.05, 8.02, 9.01]$$
$$\sigma^2_{1:5|0:4} = [102, 5.85, 4.38, 4.09, 4.02]$$

ter za korekcijske korake

$$\hat{x}_{1:5|1:5} = [2.11, 5.05, 7.02, 8.01, 9.01]$$
$$\sigma^2_{1:5|1:5} = [3.85, 2.38, 2.09, 2.02, 2.01]$$

vidimo, da robot po parih korakih ugotovi svojo lego v okolici z negotovostjo 2, kar ustreza negotovosti iz predikcijskega dela in negotovosti meritve $\left(\frac{1}{\sigma^2_{5|4}} + \frac{1}{\sigma^2_z}\right)^{-1} =$ 2.01. *Negotovost predikcijske ocene lege pa konvergira k 4, kar ustreza vsoti negotovosti korekcije iz prejšnjega koraka in meritve* $\sigma^2_{4|4} + \sigma^2_u = 4.02$

9.4.1 Kalmanov filter podan z matričnim zapisom

Sistem z več vhodi, stanji in izhodi lahko podamo v matričnem zapisu, kar omogoča večjo preglednost. Linearni proces, opisan s sistemom stanj se glasi

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{F}\mathbf{w}(k) \\ \mathbf{z}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{v}(k) \end{aligned} \tag{9.12}$$

kjer je $\mathbf{x}$ vektor stanj, $\mathbf{A}$ matrika prehajanja stanj, $\mathbf{B}$ vhodna matrika, $\mathbf{F}$ vhodna matrika šuma, $\mathbf{C}$ izhodna matrika, $\mathbf{w}(k)$ vhodni Gaussov šum, $\mathbf{z}$ izhod oziroma meritev in $\mathbf{v}$ merilni šum. V kolikor šum $\mathbf{w}$ nastopa na vhodu sistema $\mathbf{u}$ velja $\mathbf{F} = \mathbf{B}$. Procesni šum $\mathbf{w}(k)$ in merilni šum $\mathbf{v}(k)$ sta medsebojno neodvisna (nekorelirana) bela šuma z ničelno srednjo vrednostjo, katerih kovariančni matriki sta $\mathbf{Q}_k = E\left[\mathbf{w}(k)\mathbf{w}^T(k)\right]$ in $\mathbf{R}_k = E\left[\mathbf{v}(k)\mathbf{v}^T(k)\right]$.

Porazdelitev verjetnosti stanj $\mathbf{x}$, ki so motena z Gaussovim šumom podamo v matrični obliki kot

$$p(\mathbf{x}) = \det\left(2\pi\mathbf{P}\right)^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{P}^{-1}(\mathbf{x}-\mu)}$$

kjer je $\mathbf{P}$ kovariančna matrika napake ocene stanj.

Kalmanov filter je pristop za filtriranje in predikcijo linearnih sistemov z zveznim prostorom stanj, ki je moten z normalnim šumom. Šum opišemo z Gaussovo funkcijo (Gaussov šum). Šum, ki nastopa na vhodih in šum, ki nastopa na meritvah (izhodih sistema) se preko modela sistema prenaša na stanja sistema, ki jih želimo oceniti. V kolikor je model linearen je tudi preneseni šum (iz vhodov in izhodov) na stanjih Gaussov šum. Sistem torej mora biti linearen, saj nam to zagotavlja Gaussovo porazdelitev šuma na stanjih, kar je izhodišče pri izpeljavi Kalmanovega filtra. Kalmanov filter bo konvergiral k pravi oceni stanj le v primeru linearnih sistemov, ki so moteni z Gaussovim šumom.

Kalmanov filter za linearni sistem (9.12) je podan s predikcijski korakom

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{A}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}\mathbf{u}_k \\ \mathbf{P}_{k|k-1} &= \mathbf{A}\mathbf{P}_{k-1|k-1}\mathbf{A}^T + \mathbf{F}\mathbf{Q}_k\mathbf{F}^T \end{aligned} \tag{9.13}$$

in korekcijskim korakom

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{C}^T\left(\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^T + \mathbf{R}_k\right)^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1} \end{aligned} \tag{9.14}$$

V predikcijskem koraku določimo napovedno oceno $\hat{\mathbf{x}}_{k|k-1}$ (s tujko a priori), ki temelji na pretekli oceni $\hat{\mathbf{x}}_{k-1|k-1}$ dobljene iz meritev do trenutka $(k-1)$ in

trenutnega vhoda $\mathbf{u}(k)$. V korekcijskem koraku pa ocenimo trenutno oceno $\hat{\mathbf{x}}_{k|k}$ (s tujko a posteriori), ki temelji na meritvah do trenutka k. Predikcijski korak si lahko pripravimo v naprej, ko čakamo na meritve v trenutku k. Opazimo lahko podobnost matričnega zapisa (9.13) in (9.14) z zapisom podanim v Tabeli 9.3.

Izpeljimo izraz za kovariančno matriko napake ocene stanj v predikcijskem koraku

$$\begin{aligned}
\mathbf{P}_{k|k-1} &= E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^T\right] \\
&= \mathrm{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}) \\
&= \mathrm{cov}(\mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{F}\mathbf{w}_k - \mathbf{A}\hat{\mathbf{x}}_{k-1|k-1} - \mathbf{B}\mathbf{u}_k) \\
&= \mathrm{cov}(\mathbf{A}\mathbf{x}_{k-1} + \mathbf{F}\mathbf{w}_k - \mathbf{A}\hat{\mathbf{x}}_{k-1|k-1}) \\
&= \mathrm{cov}(\mathbf{A}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}) + \mathbf{F}\mathbf{w}_k) \\
&= \mathrm{cov}(\mathbf{A}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})) + \mathrm{cov}(\mathbf{F}\mathbf{w}_k) \\
&= E\left[(\mathbf{A}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}))(\mathbf{A}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}))^T\right] + \\
&\quad + E\left[(\mathbf{F}\mathbf{w}_k)(\mathbf{F}\mathbf{w}_k)^T\right] \\
&= E\left[(\mathbf{A}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})^T\mathbf{A}^T\right] + \\
&\quad + E\left[\mathbf{F}\mathbf{w}_k\mathbf{w}_k^T\mathbf{F}^T\right] \\
&= \mathbf{A}\mathbf{P}_{k-1|k-1}\mathbf{A}^T + \mathbf{F}\mathbf{Q}_k\mathbf{F}^T
\end{aligned}$$

kjer smo v šesti vrstici izpeljave upoštevali, da je vhodni šum $\mathbf{w}_k$ v trenutku k neodvisen od napake ocene stanj v prejšnjem trenutku $(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})$.

Izpeljimo še izraz za za kovariančno matriko napake ocene stanj v korekcijskem koraku

$$\begin{aligned}
\mathbf{P}_{k|k} &= E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^T\right] \\
&= \mathrm{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}) \\
&= \mathrm{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k(\mathbf{z}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1})) \\
&= \mathrm{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k(\mathbf{C}\mathbf{x}_k + \mathbf{v}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1})) \\
&= \mathrm{cov}((\mathbf{I} - \mathbf{K}_k\mathbf{C})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}) - \mathbf{K}_k\mathbf{v}_k) \\
&= \mathrm{cov}((\mathbf{I} - \mathbf{K}_k\mathbf{C})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})) + \mathrm{cov}(\mathbf{K}_k\mathbf{v}_k) \\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{C})\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{C})^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T
\end{aligned}$$

kjer smo v šesti vrstici izpeljave upoštevali, da je izhodni šum $\mathbf{v}_k$ nekoreliran z ostalimi členi. Dobljeni izraz za kovariančno matriko $\mathbf{P}_{k|k}$ je splošen in velja za katerokoli ojačenje $\mathbf{K}_k$. Izraz za $\mathbf{P}_{k|k}$ v enačbi (9.14) pa velja le za optimalno ojačenje (Kalmanovo ojačenje), torej za ojačenje, ki minimizira povprečni kvadratični pogrešek napake korekcije $E\left[\left|\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}\right|^2\right]$, kar je ekvivalentno minimizaciji vsote členov v diagonali kovariančne matrike korekcije $\mathbf{P}_{k|k}$.

Splošni izraz za $\mathbf{P}_{k|k}$ razširimo in preuredimo

$$\begin{aligned}\mathbf{P}_{k|k}=&(\mathbf{I}-\mathbf{K}_k\mathbf{C})\mathbf{P}_{k|k-1}(\mathbf{I}-\mathbf{K}_k\mathbf{C})^T+\mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T\\ =&\mathbf{P}_{k|k-1}-\mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1}-\mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{K}_k^T+\mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{K}_k^T+\mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T\\ =&\mathbf{P}_{k|k-1}-\mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1}-\mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{K}_k^T+\mathbf{K}_k(\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^T+\mathbf{R}_k)\mathbf{K}_k^T\\ =&\mathbf{P}_{k|k-1}-\mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1}-\mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{K}_k^T+\mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T\end{aligned}$$

kjer $\mathbf{S}_k=\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^T+\mathbf{R}_k$ predstavlja kovariančno matriko inovacije ($\mathbf{S}_k=\mathrm{cov}(\mathbf{z}_k-\mathbf{C}\hat{\mathbf{x}}_{k|k-1})$). Vsota diagonalnih členov $\mathbf{P}_{k|k}$ bo minimalna, ko bo odvod $\mathbf{P}_{k|k}$ po $\mathbf{K}_k$ enak 0

$$\frac{\partial\mathbf{P}_{k|k}}{\partial\mathbf{K}_k}=-2(\mathbf{C}\mathbf{P}_{k|k-1})^T+2\mathbf{K}_k\mathbf{S}_k=0$$

od koder sledi optimalno ojačenje iz enačbe (9.14)

$$\mathbf{K}_k=\mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{S}_k^{-1}=\mathbf{P}_{k|k-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^T+\mathbf{R}_k)^{-1}$$

Kovariančno matriko korekcije pri optimalnem ojačenju pa lahko izpeljemo, če optimalno ojačanje z desne strani pomnožimo z $\mathbf{S}_k\mathbf{K}_k^T$ in vstavimo v izraz za $\mathbf{P}_{k|k}$

$$\begin{aligned}\mathbf{P}_{k|k}&=\mathbf{P}_{k|k-1}-\mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1}-\mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{K}_k^T+\mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T\\ &=\mathbf{P}_{k|k-1}-\mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1}-\mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{K}_k^T+\mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{K}_k^T\\ &=\mathbf{P}_{k|k-1}-\mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1}\end{aligned}$$

Primer 9.19. *Mobilni robot se premika v ravnini in svojo lego meri z GPS desetkrat v sekundi (čas vzorčenja $T_S=0.1$ s). Meritev je motena z Gausovim šumom, ki ima varianco $10m^2$. Robot se premika s hitrostjo 1 $\frac{m}{s}$ v smeri x in s hitrostjo 0 $\frac{m}{s}$ v smeri y. Varianca Gausovega šuma hitrosti pomika je $1\frac{m^2}{s^2}$. Robot se nahaja v izhodišču $\mathbf{x}=[0,0]^T$, naša ocena začetne lege pa je $\hat{\mathbf{x}}=[3,3]^T$ in je podana z začetno varianco*

$$\mathbf{P}_0=\begin{pmatrix}10&0\\0&10\end{pmatrix}$$

Kakšen je časovni potek ocene lege robota in varianca te ocene?

Rešitev

Do rešitve lahko pridemo s simulacijo v okolju Matlab. Določimo model premikanja robota, kjer ocena stanja predstavlja lego robota v ravnini $\hat{\mathbf{x}}_{k|k-1}=[x_k,y_k]^T$ in vhod $\mathbf{u}=[v_x,v_y]^T$ predstavlja hitrost sistema v smeri x in v smeri y. Model za predikcijo stanja sistema torej je

$$\hat{\mathbf{x}}_{k|k-1}=\begin{bmatrix}1&0\\0&1\end{bmatrix}\hat{\mathbf{x}}_{k-1|k-1}+\begin{bmatrix}T_S&0\\0&T_S\end{bmatrix}\mathbf{u}$$

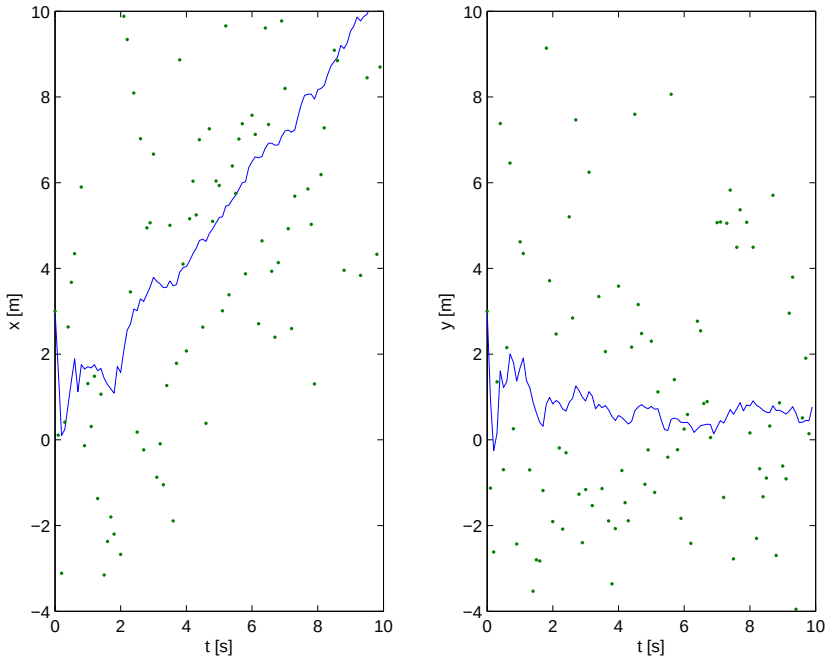
Model meritve pozicije z GPS pa je

$$\hat{\mathbf{z}}_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k|k-1}$$

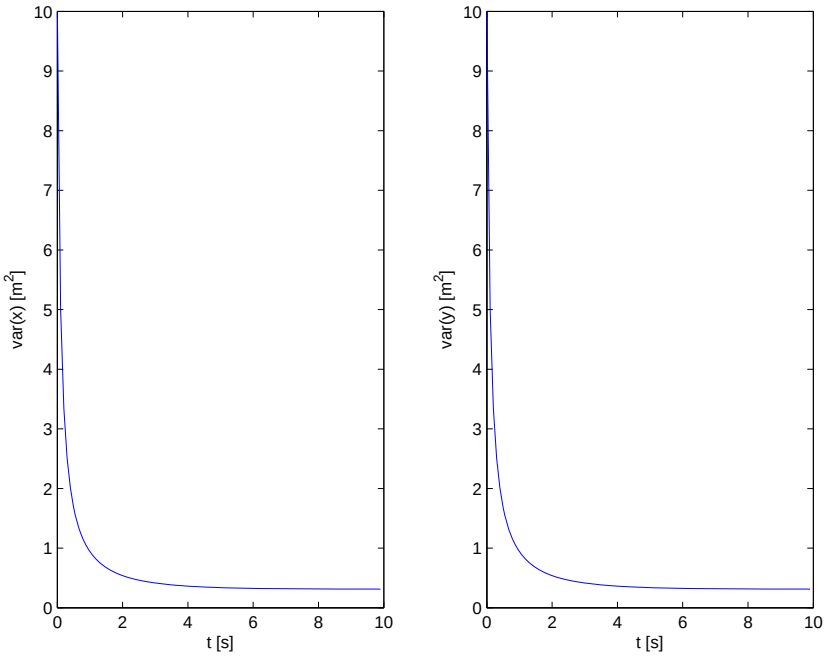
V nadaljevanju je podana koda in grafični prikaz rešitve.

```
function Kalman_Matricno

% linarni sistem z modelom v prostoru stanj
Ts=0.1; % cas vzorčenja
A=[1 0;0 1]; B=[Ts 0;0 Ts]; C=[1 0;0 1];
F=B;   %šum nastopa kar na vhodu

% Dejansko začetno stanje
xTrue=[0;0];
% Ocenjeno zacetno stanje
x=[3;3]; P=diag([10,10]);
% varianca šum aktuatorja za pomik
Q =diag([1,1]);
%varianca šuma GPS senzorja razdalje
R=diag([10,10]);

%shranjujemo rezultate za prikaz
XStore=[]; PStore=[]; XTrueStore=[]; ZStore=[]; XStore=[XStore,x];
PStore=[PStore,diag(P)];       % shranimo le diagonalna člena
XTrueStore=[XTrueStore,xTrue]; ZStore=[ZStore;x];

%naredimo zanko
N=100; for k = 1:N-1
    u=[1;0];    % ukaz za premik

    % simulacija stanja dejanskega robota in opravljene meritve
    xTrue=A*xTrue+B*u+F*sqrt(Q)*randn(2,1) ;
    zTrue = C*xTrue+sqrt(R)*randn(2,1);

    % ocena lege iz poznanih vhodov in meritev
    %%% Predikcija
    xPred=A*x+B*u;
    PPred=A*P*A'+F*Q*F';


    %%% Korekcija
    K=PPred*C'*(C*PPred*C'+R)^(-1);
    x=xPred+K*(z-C*xPred);
    P=PPred-K*C*PPred;

    % shranjujemo rezultate za prikaz
    XStore=[XStore,x];
    PStore=[PStore,diag(P)];
    XTrueStore=[XTrueStore,xTrue];
    ZStore=[ZStore,zTrue];
end

t=(0:N-1)*Ts;
figure,subplot(1,2,1),plot(t,XStore(1,:),t,ZStore(1,:),'.')
axis([0,100*Ts,-4,10]), ylabel ('x [m]'), xlabel('t [s]')
subplot(1,2,2),plot(t,XStore(2,:),t,ZStore(2,:),'.'),
axis([0,100*Ts,-4,10]), xlabel('t [s]'), ylabel ('y [m]')

figure,subplot(1,2,1),plot(t,PStore(1,:)) ylabel ('var(x) [m^2]'),
xlabel('t [s]') subplot(1,2,2),plot(t,PStore(2,:)) xlabel('t
[s]'), ylabel ('var(y) [m^2]')
```

Rezultati simulacije primera 9.19 so podani v slikah 9.8 in 9.9.



Slika 9.8: Ocena lege robota, ki začne v izhodišču, začetna ocena filtra pa je inicializirana na $\hat{\mathbf{x}} = [3, 3]$. Robot se premika s hitrostjo $\mathbf{u} = [1, 0]\frac{m}{s}$, meritev lege je označena s točkami.



Slika 9.9: Varianca ocene lege robota.

9.4.2 Razširjeni Kalmanov filter

Kalmanov filter je razvit za linearne sisteme, vse motnje in šumi pa morajo biti opisljivi s šumom normalne porazdelitve. Če Gaussov šum transformiramo skozi linearno funkcijo je dobljeni šum še vedno Gaussov z drugimi parametri, katere pa lahko eksplicitno izračunamo iz poznane linearne funkcije. Ravno zaradi slednjega je Kalmanov filter računsko učinkovit algoritem. V primeru, da vhodni Gaussov šum transformiramo skozi neko nelinearno funkcijo, pa dobljeni izhodni šum ni več Gaussov, čeprav ga lahko še vedno aproksimiramo z Gaussovo funkcijo.

V primeru nelinearne funkcije, ki opisuje prehajanje stanj sistema ali proces meritve uporabimo razširjeni Kalmanov filter Extended Kalman filter - EKF), kjer se nealinearnosti aproksimira z linarnim modelom. Linearni model se oceni s pomočjo linearizacije (razvoj v Taylorjevo vrsto prvega reda) v okolici trenutne ocene stanj. Z linearizacijo dobimo občutljivostne matrike (Jacobijeve matrike) za trenutne vrednosti ocenjenih stanj in meritev. Tako dobljeni linearni model omogoča izračun približka Gaussove porazdelitve gostote šuma za nek resnični šum, ki ni nujno Gaussov.

Z uporabo linearizacije pri modeliranju šuma je razširjeni Kalmanov filter še zmeraj računsko učinkovit algoritem in zato v praksi zelo pogosto upo-

rabljen. Točnost, ki jo z linearno aproksimacijo prenosa šuma dosežemo je odvisna od velikosti variance šuma (pri velikih negotovostih oziroma amplitudah šuma je linarni približek slabši, saj smo izven zadovoljivega linearnega območja) ter od stopnje nelinearnosti. Zaradi napake, ki jo v sistem vnaša linearizacija se lahko zgodi, da filter slabše konvergira ali sploh ne konvergira k pravi rešitvi.

Splošen zapis nelinearnega sistema je podan z

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k\right) \\ \mathbf{z}_k &= \mathbf{h}\left(\mathbf{x}_k\right) + \mathbf{v}_k \end{aligned} \tag{9.15}$$

kjer šum $\mathbf{w}_k$) lahko nastopa na vhodu sistema ali pa vpliva direktno na stanja sistema.

Kalmanov filter za nelinearni sistem (9.15) je podan s predikcijski korakom

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k\right) \\ \mathbf{P}_{k|k-1} &= \mathbf{A}\mathbf{P}_{k-1|k-1}\mathbf{A}^T + \mathbf{F}\mathbf{Q}_k\mathbf{F}^T \end{aligned} \tag{9.16}$$

in korekcijskim korakom

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{C}^T\left(\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^T + \mathbf{R}_k\right)^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{C}\mathbf{P}_{k|k-1} \end{aligned} \tag{9.17}$$

kjer je za oceno predikcije stanj (9.16) uporabimo nelinearni model, za izračun kovariančne matrike šuma pa določimo Jacobijevo matriko $\mathbf{A}$ za prehajanje šuma iz prejšnjih vrednosti stanj na stanja in matriko $\mathbf{F}$ za prehajanje šuma iz vhodov na stanja .

$$\mathbf{A} = \left.\frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}\right|_{\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k\right)} \tag{9.18}$$

$$\mathbf{F} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}}\right|_{\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k\right)} \tag{9.19}$$

V korekcijskem delu določimo oceno meritve na podlagi predikcijske ocene stanja $\hat{\mathbf{z}}_k = \mathbf{h}\left(\hat{\mathbf{x}}_{k|k-1}\right)$. Nato linearizirамo še prehod šuma iz stanj na meritve, kar opisuje matrika $\mathbf{C}$

$$\mathbf{C} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\left(\hat{\mathbf{x}}_{k|k-1}\right)} \tag{9.20}$$

Kovariančni matriki šumov pa sta podani z $\mathbf{Q}_k = E\left[\mathbf{w}(k)\mathbf{w}^T(k)\right]$ in $\mathbf{R}_k = E\left[\mathbf{v}(k)\mathbf{v}^T(k)\right]$.

Primer 9.20. *Mobilni robot z diferencialnim pogonom se premika v ravnini. Vhodni ukaz robota je translatorna hitrost v_k in kotna hitrost ω_k, ki sta motena z Gaussovim šumom z varianco* $\mathrm{var}(v_k) = 0.1\frac{m^2}{s^2}$ *in* $\mathrm{var}(\omega_k) = 0.1\frac{rad^2}{s^2}$.

Robot ima senzor s pomočjo katerega lahko izmeri razdaljo do markerja, ki se nahaja v koordinatnem izhodišču. Za merjenje svoje orientacije ima robot kompas. Meritev razdalje je motena z Gausovim šumom, ki ima varianco $0.5m^2$ *meritev kota pa z Gausovim šumom* $0.3rad^2$.

Robot se na začetku nahaja v izhodišču z orientacijo $\theta_0 = 0$. Njegova začetna lega tore je $\mathbf{x}_0 = [0,0,0]^T$, naša ocena začetne lege pa je $\hat{\mathbf{x}}_0 = [3,3,\frac{\pi}{4}]^T$ in je podana z začetno varianco

$$\mathbf{P}_0 = \begin{pmatrix} 9 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0.6 \end{pmatrix}$$

Kakšen je časovni potek ocene lege robota ($\hat{\mathbf{x}}_{k|k-1} = [x_k, y_k, \theta_k]^T$) in varianca te ocene, če robotu ob vsakem času vzorčenja $T_S = 0.1s$ pošljemo ukaz $\mathbf{u} = [v_k, \omega_k]^T = [0.5, 0.5]^T$?

Rešitev

Do rešitve lahko pridemo s simulacijo v okolju Matlab. Določimo model premikanja robota. Nelinearni model (kinematični model) za predikcijo stanja sistema torej je

$$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + \begin{bmatrix} T_s v_k \cos(\theta_{k-1}) \\ T_s v_k \sin(\theta_{k-1}) \\ T_s \omega_k \end{bmatrix}$$

Model meritve razdalje in kota za korekcijski korak pa je

$$\hat{\mathbf{z}}_k = \begin{bmatrix} \sqrt{{x_k}^2 + {y_k}^2} \\ \theta_k \end{bmatrix}$$

V nadaljevanju je podana koda in grafični prikaz rešitve.

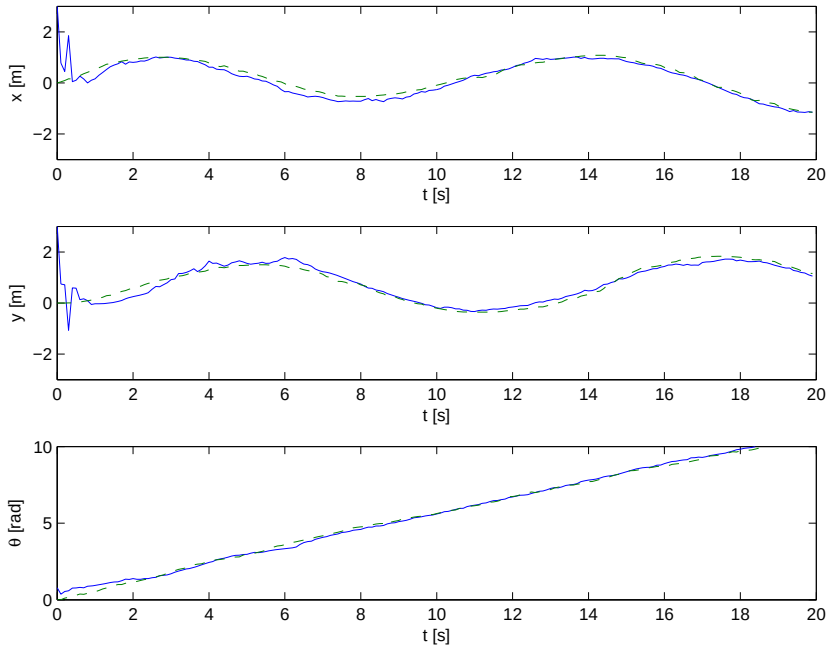
```
function KalmanEKF
close all, clear all,

Ts=0.1; % cas vzorčenja
xTrue=[0;0;0];  % Dejansko začetno stanje
x=[3;3;pi/4]; P=diag([9,9,0.6]); % Ocenjeno začetno stanje in kovarianca
Q =diag([0.1,0.1]); % Varianca šum aktuatorja za pomik
R=diag([0.5,0.3]);   %Varianca šuma meritve razdalje in orientacije


%shranjujemo rezultate za prikaz
XStore=[]; PStore=[]; XTrueStore=[]; ZStore=[]; XStore=[XStore,x];
PStore=[PStore,diag(P)];       % shranimo le diagonalna člena
XTrueStore=[XTrueStore,xTrue];
ZStore=[ZStore,[0;0]];

%naredimo zanko
N=200;
```

```
for k = 1:N-1
    u=[0.5;0.5];    % ukaz za premik (translatorna in kotna hitrost)
    u_sum=u+sqrt(Q)*randn(2,1) ;

    % simulacija dejanskega stanja (lege) robota
    xTrue = xTrue + Ts*[ u_sum(1)*cos(xTrue(3)); ...
                         u_sum(1)*sin(xTrue(3)); ...
                         u_sum(2)   ];

    % simuliramo dejansko pošumljeno meritev
    zTrue = [sqrt(xTrue(1)^2 + xTrue(2)^2 );...
             xTrue(3)]  + sqrt(R)*randn(2,1);

    %%% Predikcija (ocena lege in hitrosti iz poznanih vhodov)
    xPred = x + Ts*[ u(1)*cos(x(3)); ...
                     u(1)*sin(x(3)); ...
                     u(2)         ];

    % ocenimo matrike za prenos šuma
    A=[1 0 -Ts*u(1)*sin(x(3)); ...
       0 1  Ts*u(1)*cos(x(3)); ...
       0 0  1];
    F=[Ts*cos(x(3)) 0; ...
       Ts*sin(x(3)) 0; ...
       0           Ts];
    PPred=A*P*A'+F*Q*F';

    % ocenjena meritev
    z= [sqrt(xPred(1)^2 + xPred(2)^2 );...
        xPred(3)] ;

    %%% Korekcija
    d=sqrt(xPred(1)^2 + xPred(2)^2);
    C=[xPred(1)/d xPred(2)/d 0;...
        0         0          1];
    K=PPred*C'*(C*PPred*C'+R)^(-1);
    inov=zTrue-z;
    x=xPred+K*(inov);
    P=PPred-K*C*PPred;

    % shranjujemo rezultate za prikaz
    XStore=[XStore,x];
    PStore=[PStore,diag(P)];
    XTrueStore=[XTrueStore,xTrue];
    ZStore=[ZStore,zTrue];
end

t=(0:N-1)*Ts;
figure,subplot(3,1,1),plot(t,XStore(1,:),t,XTrueStore(1,:),'--'),axis([0,N*Ts,-3,3]), ylabel ('x [m]'), xlabel('t [s]')
subplot(3,1,2),plot(t,XStore(2,:),t,XTrueStore(2,:),'--'), axis([0,N*Ts,-3,3]), xlabel('t [s]'), ylabel ('y [m]')
subplot(3,1,3),plot(t,XStore(3,:),t,XTrueStore(3,:),'--'), axis([0,N*Ts,0,10]), xlabel('t [s]'), ylabel ('\theta [rad]')

figure,subplot(1,3,1),plot(t,PStore(1,:)),ylabel ('var(x) [m^2]'), xlabel('t [s]')
subplot(1,3,2),plot(t,PStore(2,:)), xlabel('t [s]'), ylabel ('var(y) [m^2]')
subplot(1,3,3),plot(t,PStore(3,:)), xlabel('t [s]'), ylabel ('\theta [rad^2]')

figure,subplot(1,2,1),plot(t,ZStore(1,:)), ylabel ('d(x) [m]'), xlabel('t [s]')
subplot(1,2,2), plot(t,ZStore(2,:)), xlabel('t [s]'), ylabel ('\alpha [rad]')
```

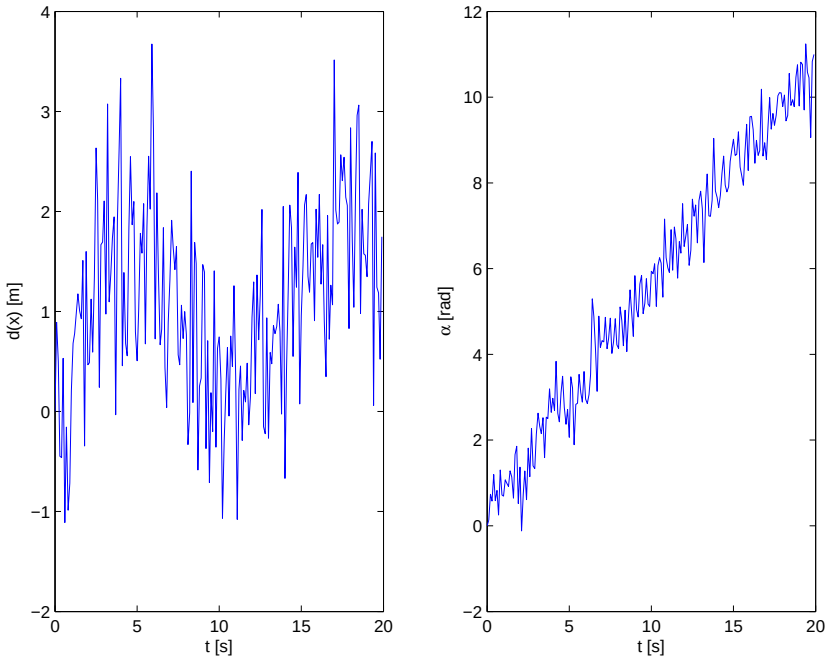
Rezultati simulacije primera 9.20 so podani v slikah 9.10, 9.11 in 9.12.

V primeru 9.21 si poglejmo še nekoliko spremenjen primer 9.20, kjer smo predpostavili, da ima robot senzor, ki izmeri razdaljo in kot do markerja v izhodišču.

Primer 9.21. *Mobilnemu robotu iz primera 9.20 spremenimo le model meritve. Robot ima senzor s pomočjo katerega lahko izmeri razdaljo in kot do markerja, ki se nahaja v koordinatnem izhodišču. Meritev kota je podana v območju* $\alpha \in [-\pi, \pi]$. *Meritev razdalje je motena z Gausovim šumom, ki ima varianco* $0.5m^2$ *meritev kota pa z Gausovim šumom* $0.3rad^2$.



Slika 9.10: Ocena lege robota, ki začne v izhodišču, začetna ocena filtra pa je inicializirana na $\hat{\mathbf{x}} = [3, 3, \frac{\pi}{4}]$. Ocenjena lega robota je označena s polno črto, dejanska lega robota pa črtkano.



Slika 9.11: Meritve razdalje in kota.

Kakšen je časovni potek ocene lege robota ($\hat{\mathbf{x}}_{k|k-1} = [x_k, y_k, \theta_k]^T$) in varianca te ocene, če robotu ob vsakem času vzorčenja $T_S = 0.1s$ pošljemo ukaz $\mathbf{u} = [v_k, \omega_k]^T = [0.5, 0.5]^T$?

Rešitev

Do rešitve lahko pridemo s simulacijo v okolju Matlab. Določimo model premikanja robota. Nelinearni model (kineamtični model) za predikcijo stanja sistema torej je

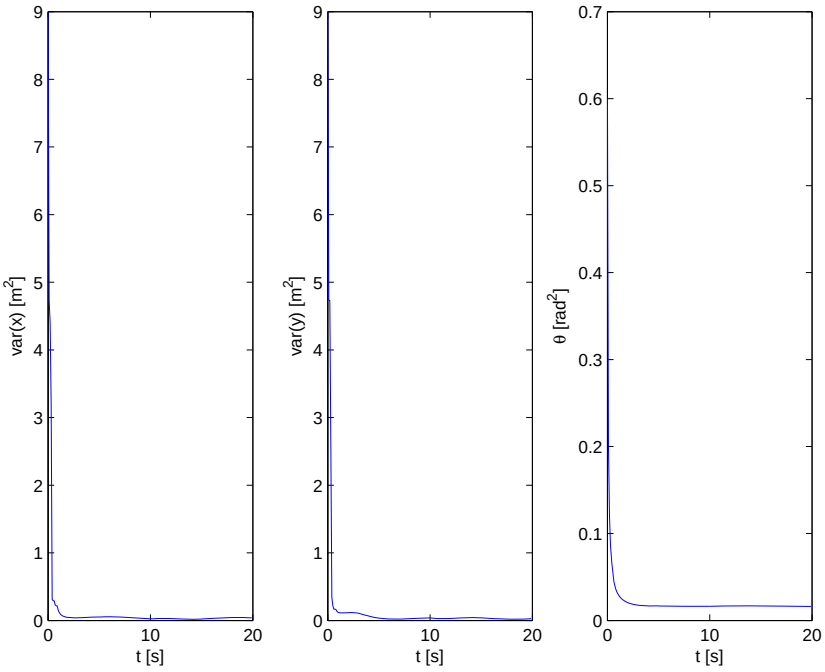
$$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + \begin{bmatrix} T_s v_k \cos(\theta_{k-1}) \\ T_s v_k \sin(\theta_{k-1}) \\ T_s \omega_k \end{bmatrix}$$

Model meritve razdalje in kota za korekcijski korak pa je

$$\hat{\mathbf{z}}_k = \begin{bmatrix} \sqrt{{x_k}^2 + {y_k}^2} \\ \arctan \frac{0-y_k}{0-x_k} - \theta_k \end{bmatrix}$$

V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.

```
function KalmanEKF
close all, clear all,
Ts=0.1; % cas vzorčenja
xTrue=[0;0;0];  % Dejansko začetno stanje
```



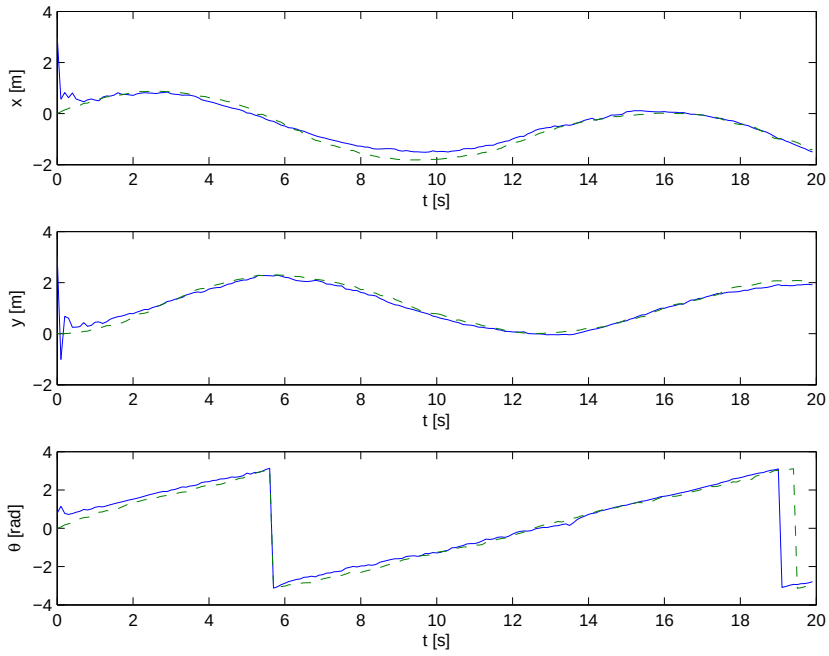
Slika 9.12: Varianca ocene lege robota.

```
x=[3;3;pi/4];  % Ocenjeno zacetno stanje

P=diag([9,9,0.6]);   % začetna varianca stanj
Q =diag([0.1,0.1]);  % varianca šum aktuatorja za pomik
R=diag([0.5,0.3]);   % varianca šuma meritve razdalje in kota

% shranjujemo rezultate za prikaz
XStore=[]; PStore=[]; XTrueStore=[]; ZStore=[];InovStore=[];
XStore=[XStore,x];
PStore=[PStore,diag(P)];       % shranimo le diagonalna člena
XTrueStore=[XTrueStore,xTrue]; ZStore=[ZStore,[0;0]];
InovStore=[InovStore,[0;0]];


%naredimo zanko
N=200; for k = 1:N-1
    u=[0.5;0.5];    % ukaz za premik (translatorna in kotna hitrost)
    u_sum=u + sqrt(Q)*randn(2,1) ;

    % simulacija dejanskega stanja (lege) robota
    xTrue = xTrue + Ts*[ u_sum(1)*cos(xTrue(3)); ...
                         u_sum(1)*sin(xTrue(3)); ...
                         u_sum(2)   ];
    xTrue(3)=PopraviCiklicnostKota(xTrue(3));

    % simuliramo dejansko pošumljeno meritev (razdalja in orientacija)
    zTrue = [sqrt(xTrue(1)^2 + xTrue(2)^2 );...
             atan2(0-xTrue(2),0-xTrue(1)) - xTrue(3)]  + sqrt(R)*randn(2,1);
    zTrue(2)=PopraviCiklicnostKota(zTrue(2));

    %%% Predikcija (ocena lege in hitrosti iz poznanih vhodov)
    xPred = x + Ts*[ u(1)*cos(x(3)); ...
                     u(1)*sin(x(3)); ...
                     u(2)         ];
    xPred(3)=PopraviCiklicnostKota(xPred(3));


    % ocenimo matrike za prenos šuma
    A=[1 0 -Ts*u(1)*sin(x(3)); ...
```

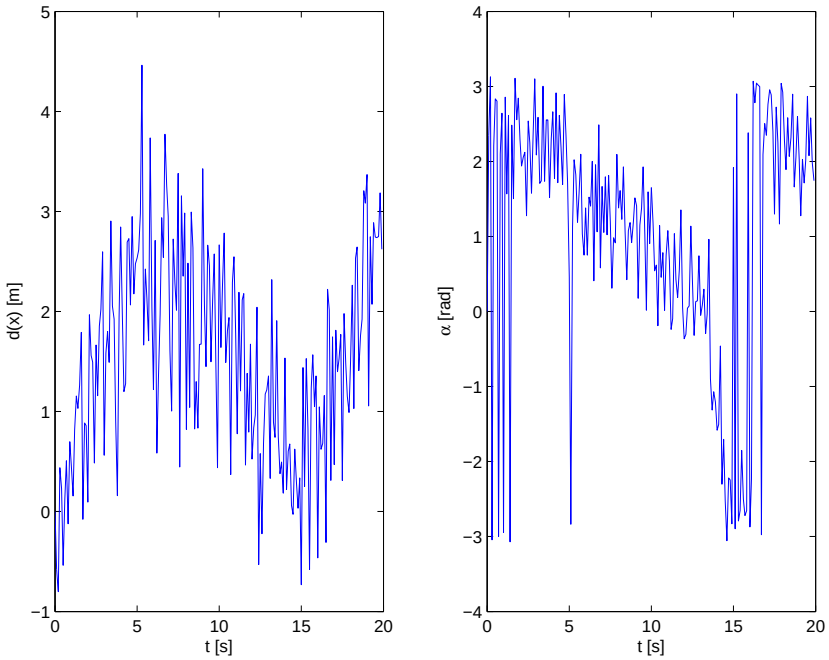
```
      0 1  Ts*u(1)*cos(x(3)); ...
      0 0  1];
   F=[Ts*cos(x(3)) 0; ...
      Ts*sin(x(3)) 0; ...
      0           Ts];
   PPred=A*P*A'+F*Q*F';


   % ocenjena meritev
   z= [sqrt(xPred(1)^2 + xPred(2)^2 );...
       atan2(0-xPred(2),0-xPred(1)) - xPred(3)] ;
   z(2)=PopraviCiklicnostKota(z(2));

   %%% Korekcija
   d=sqrt(xPred(1)^2 + xPred(2)^2 );
   C=[xPred(1)/d     xPred(2)/d  0;...
     -xPred(2)/d^2   xPred(1)/d^2    -1];
   K=PPred*C'*(C*PPred*C'+R)^(-1);
   inov=zTrue-z;
   % izberemo primerno inovacijo za kot zaradi šuma in cikličnosti kota
   iii=zTrue(2) - (z(2) + [0;2*pi;-2*pi]);
   [tmp,index]=min(abs(iii));
   inov(2)=iii(index);

   x=xPred+K*(inov);
   P=PPred-K*C*PPred;

   % shranjujemo rezultate za prikaz
   XStore=[XStore,x];
   PStore=[PStore,diag(P)];
   XTrueStore=[XTrueStore,xTrue];
   ZStore=[ZStore,zTrue];
   InovStore=[InovStore,inov];

end

t=(0:N-1)*Ts;
figure,subplot(3,1,1),plot(t,XStore(1,:),t,XTrueStore(1,:),'--'), ylabel ('x [m]'), xlabel('t [s]')
subplot(3,1,2),plot(t,XStore(2,:),t,XTrueStore(2,:),'--'), xlabel('t [s]'), ylabel ('y [m]')
subplot(3,1,3),plot(t,XStore(3,:),t,XTrueStore(3,:),'--'), xlabel('t [s]'), ylabel ('\theta [rad]')

figure,subplot(1,3,1),plot(t,PStore(1,:)) ylabel ('var(x) [m^2]'),xlabel('t [s]')
subplot(1,3,2),plot(t,PStore(2,:)), xlabel('t [s]'), ylabel ('var(y) [m^2]')
subplot(1,3,3),plot(t,PStore(3,:)), xlabel('t [s]'), ylabel ('\theta [rad^2]')

figure,subplot(1,2,1),plot(t,ZStore(1,:)) ylabel ('d(x) [m]'), xlabel('t [s]')
subplot(1,2,2),plot(t,ZStore(2,:)) xlabel('t[s]'), ylabel ('\alpha [rad]')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = PopraviCiklicnostKota(a)
    a=atan2(sin(a),cos(a));   % prevedemo kot na območje [-pi,pi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati simulacije primera 9.21 so podani v slikah 9.13, 9.14, 9.15 in 9.16.

Čeprav je ta primer je zelo podoben primeru 9.20, je konvergenca ocenjenih stanj filtra slabša. To se zgodi zaradi dodatne nelinearnosti pri meritvi kota in preskokov kota zaradi cikličnosti kota, ki jih je potrebno ustrezno upoštevati. Ker sta obe meritvi (razdalja in kot) relativni se lahko zgodi, da filter konvergira k napačni oceni stanj, kjer je inovacija (razlika meritve in predikcije meritve) še zmeraj blizu ničnе vrednosti. Primer konvergence k napačni oceni stanj prikazujejo slike 9.17 – 9.20. To bi lahko rešili, če bi opazovali več markerjev hkrati s čemer bi bila informacija dobljena iz meritev zadostna za pravilno oceno stanj, kar nakazuje primer 9.22.



Slika 9.13: Ocena lege robota, ki začne v izhodišču, začetna ocena filtra pa je inicializirana na $\hat{\mathbf{x}} = [3, 3, \frac{\pi}{4}]$. Ocenjena lega robota je označena s polno črto, dejanska lega robota pa črtkano.



Slika 9.14: Meritve razdalje in kota.

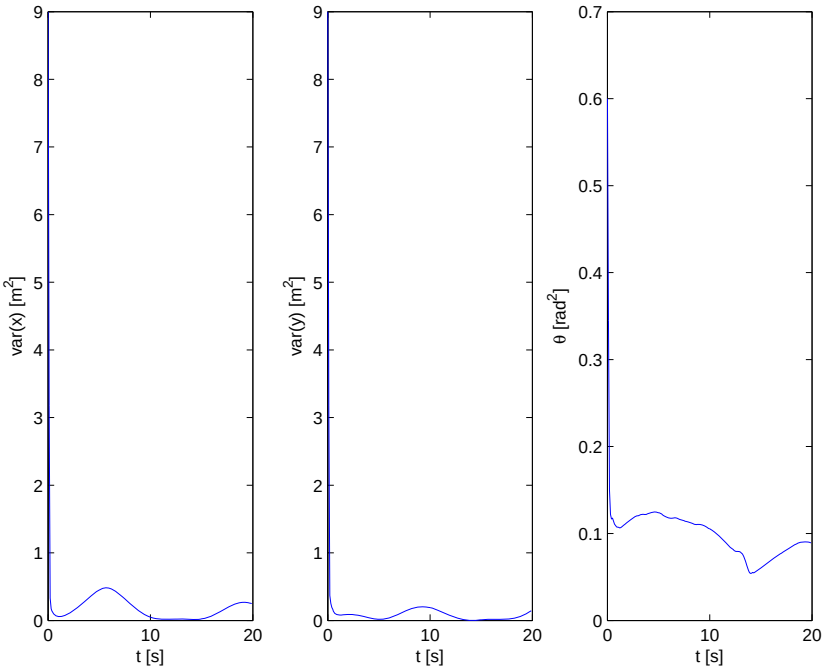
Primer 9.22. *V tem primeru nadgradimo primer 9.21 tako, da lahko opazujemo razdaljo in kot do dveh markerjev hkrati, ki se nahajta pri* $x_{M1} = 0$, $y_{M1} = 0$ *in* $x_{M2} = 5$, $y_{M2} = 5$. *Vsi ostali podatki pa so enaki, kot v primeru 9.21.*

Rešitev

Prilagoditi moramo le del kode, ki se nanaša na korekcijski korak. Vektor meritve sedaj vsebuje štiri podatke, razdaljo in kot do prvega markerja ter razdaljo in kot do drugega markerja.

$$\hat{\mathbf{z}}_k = \begin{bmatrix} \sqrt{(x_{M1} - x_k)^2 + (y_{M1} - y_k)^2} \\ \arctan \frac{y_{M1} - y_k}{x_{M1} - x_k} - \theta_k \\ \sqrt{(x_{M2} - x_k)^2 + (y_{M2} - y_k)^2} \\ \arctan \frac{y_{M2} - y_k}{x_{M2} - x_k} - \theta_k \end{bmatrix}$$

Razširimo še izhodno matriko C *(glej 9.20), ki jo lineariziramo okoli trenutne*



Slika 9.15: Varianca ocene lege robota.

predikcijske ocene (x_k, y_k)

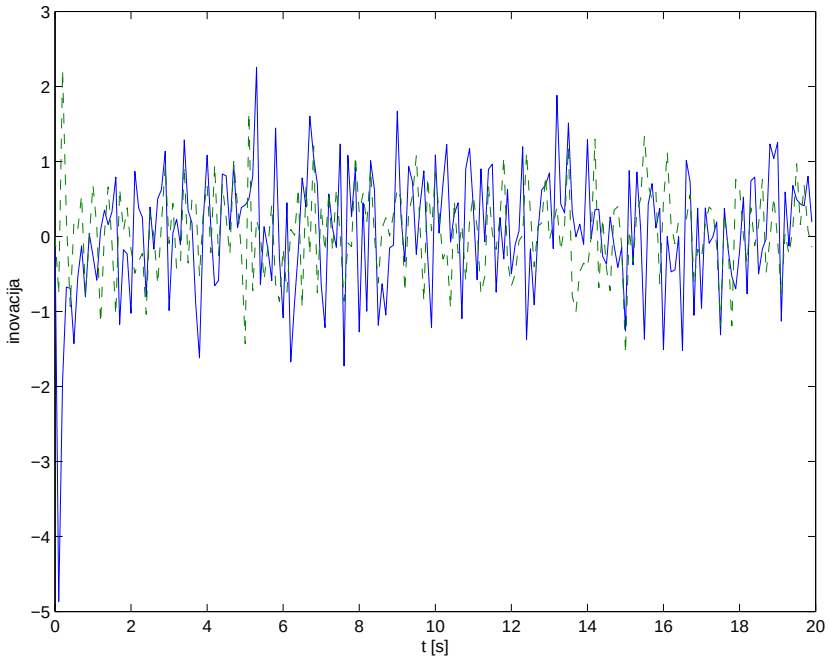
$$\mathbf{C} = \begin{bmatrix} \frac{x_k}{d_1} & \frac{y_k}{d_1} & 0 \\ -\frac{y_k}{d_1^2} & \frac{x_k}{d_1^2} & -1 \\ \frac{x_k}{d_2} & \frac{y_k}{d_2} & 0 \\ -\frac{y_k}{d_2^2} & \frac{x_k}{d_2^2} & -1 \end{bmatrix}$$

kjer je $d_1 = \sqrt{(x_{M1} - x_k)^2 + (y_{M1} - y_k)^2}$ *in* $d_2 = \sqrt{(x_{M2} - x_k)^2 + (y_{M2} - y_k)^2}$.
Razširimo tudi kovariančno matriko šuma meritve, ki je

$$\mathbf{R} = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix}$$

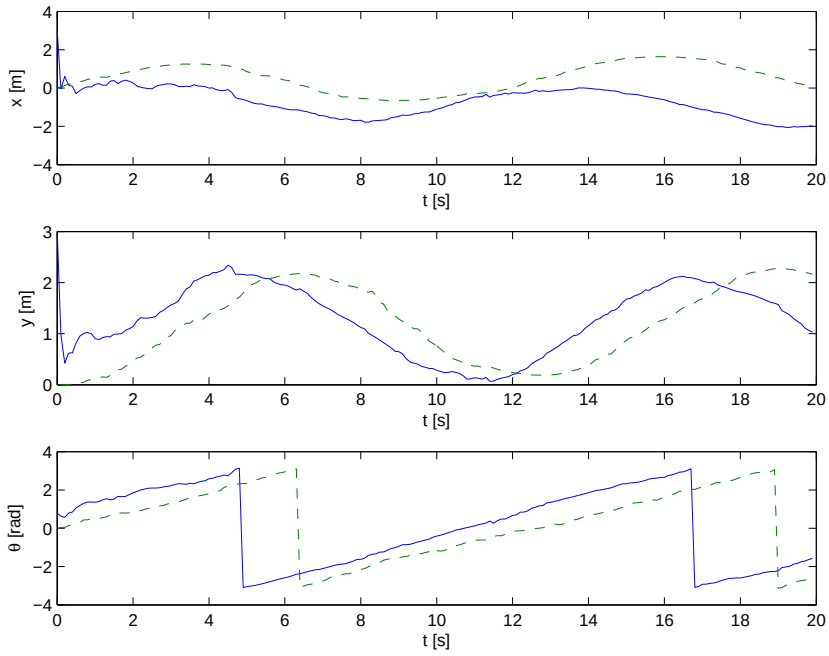
V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.

```
function KalmanEKF

close all, clear all,
Ts=0.1; % cas vzorčenja
xTrue=[0;0;0];  % Dejansko začetno stanje
x=[3;3;pi/4];   % Ocenjeno zacetno stanje
```

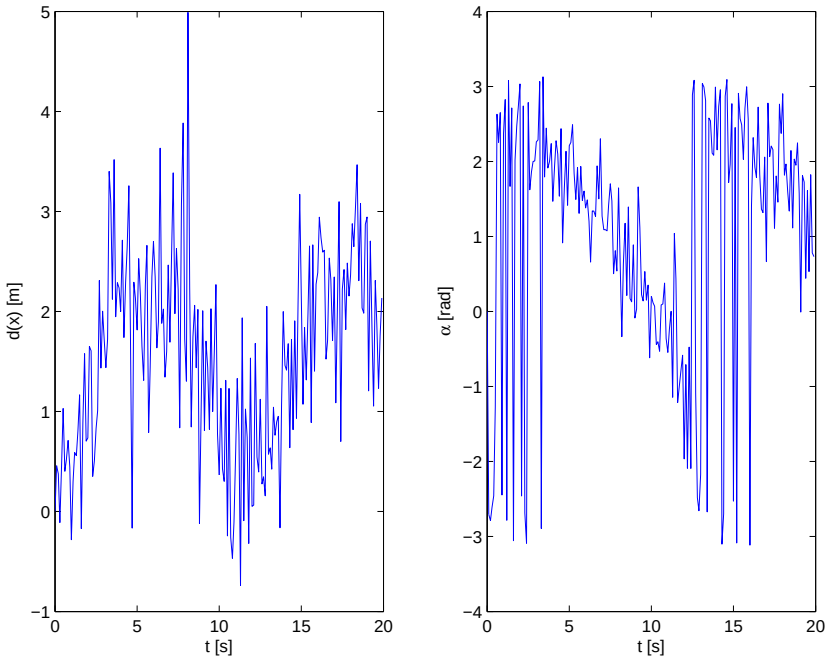


Slika 9.16: Potek inovacije.

```
Marker=[0 0; 5 5];

P=diag([9,9,0.6]);  % začetna kovariančna matrika stanj
Q =diag([0.1,0.1]); % varianca šum aktuatorja za pomik
R=diag([0.5,0.3]);  % varianca šuma meritve razdalje in kota

% shranjujemo rezultate za prikaz
XStore=[]; PStore=[]; XTrueStore=[]; ZStore=[];InovStore=[];

%naredimo zanko
N=300; for k = 1:N
    u=[0.5;0.5];    % ukaz za premik (translatorna in kotna hitrost)
    u_sum=u + sqrt(Q)*randn(2,1) ;

    % izračun dejanskega stanja (lege) robota
    xTrue = xTrue + Ts*[ u_sum(1)*cos(xTrue(3)); ...
                         u_sum(1)*sin(xTrue(3)); ...
                         u_sum(2)   ];
    xTrue(3)=PopraviCiklicnostKota(xTrue(3));

    % dejanska pošumljeno meritev (razdalja in orientacija)
    zTrue=[];
    for m=1:size(Marker,1)
        dist=sqrt((Marker(m,1)-xTrue(1))^2 + (Marker(m,2)-xTrue(2))^2 );
        kot=atan2(Marker(m,2)-xTrue(2),Marker(m,1)-xTrue(1)) - xTrue(3);
        zz=[dist;kot]  + sqrt(R)*randn(2,1);
        zz(2)=PopraviCiklicnostKota(zz(2));
        zTrue =[zTrue; zz ];
    end

    %%% Predikcija (ocena lege in hitrosti iz poznanih vhodov)
    xPred = x + Ts*[ u(1)*cos(x(3)); ...
                     u(1)*sin(x(3)); ...
                     u(2)         ];
    xPred(3)=PopraviCiklicnostKota(xPred(3));

    % ocenimo matrike za prenos šuma
    A=[1 0 -Ts*u(1)*sin(x(3)); ...
```

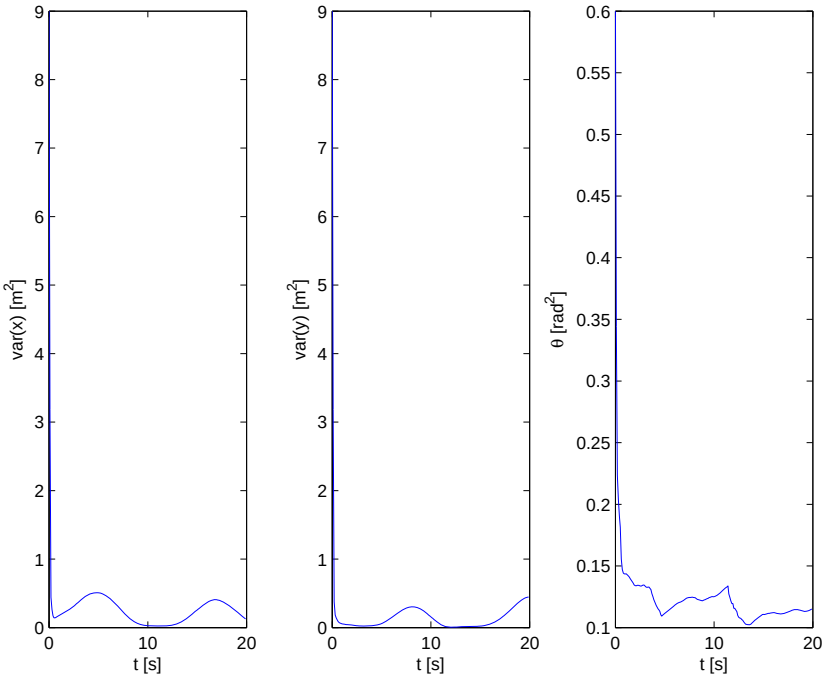


Slika 9.17: Ocena lege robota, ki začne v izhodišču, začetna ocena filtra pa je inicializirana na $\hat{\mathbf{x}} = [3, 3, \frac{\pi}{4}]$. Ocenjena lega robota je označena s polno črto, dejanska lega robota pa črtkano.

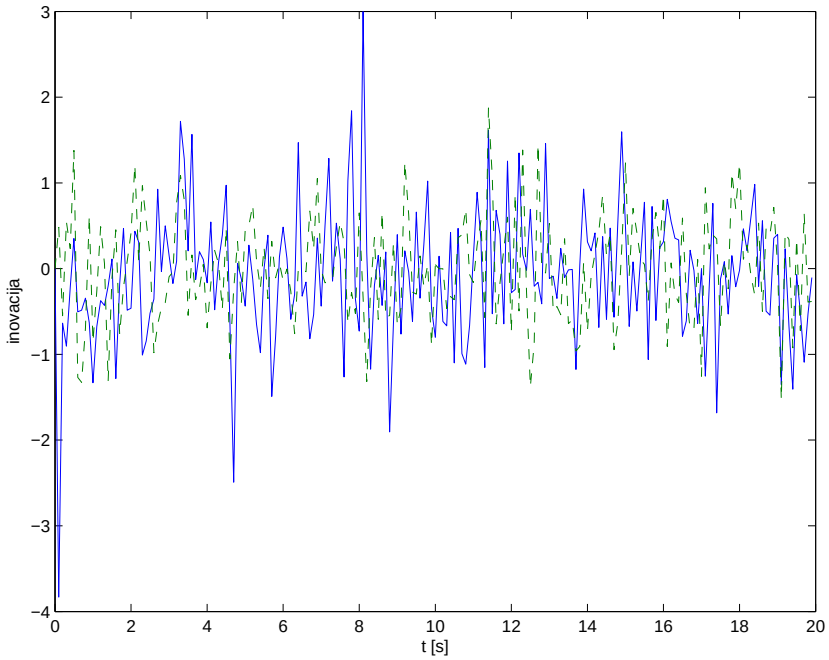


Slika 9.18: Meritve razdalje in kota.

```
   0 1  Ts*u(1)*cos(x(3)); ...
   0 0  1];
F=[Ts*cos(x(3)) 0; ...
   Ts*sin(x(3)) 0; ...
   0           Ts];
PPred=A*P*A'+F*Q*F';

%%% Korekcijski korak, ocenjena meritev
z=[]; C=[];
for m=1:size(Marker,1)
    dist=sqrt((Marker(m,1)-xPred(1))^2 + (Marker(m,2)-xPred(2))^2 );
    kot=atan2(Marker(m,2)-xPred(2),Marker(m,1)-xPred(1)) - xPred(3);
    zz=[dist;kot];
    zz(2)=PopraviCiklicnostKota(zz(2));
    z =[z; zz ];

    % sestavimo matriko C za korekcijo
    c=[xPred(1)/dist     xPred(2)/dist  0;...
      -xPred(2)/dist^2   xPred(1)/dist^2    -1];
    C=[C;c];
end

RR=diag(repmat([R(1,1) R(2,2)],1,size(Marker,1))); % kovariančna matrika meritve
K=PPred*C'*(C*PPred*C'+RR)^(-1);

inov=zTrue-z;
% izberemo pravo inovacijo za kot zaradi šuma in cikličnosti kota
for m=1:size(Marker,1)
    iii=zTrue(2*m) - (z(2*m) + [0;2*pi;-2*pi]);
    [tmp,index]=min(abs(iii));
    inov(2*m)=iii(index);
end

x=xPred+K*(inov);
P=PPred-K*C*PPred;

% shranjujemo rezultate za prikaz
XStore=[XStore,x];
PStore=[PStore,diag(P)];
```



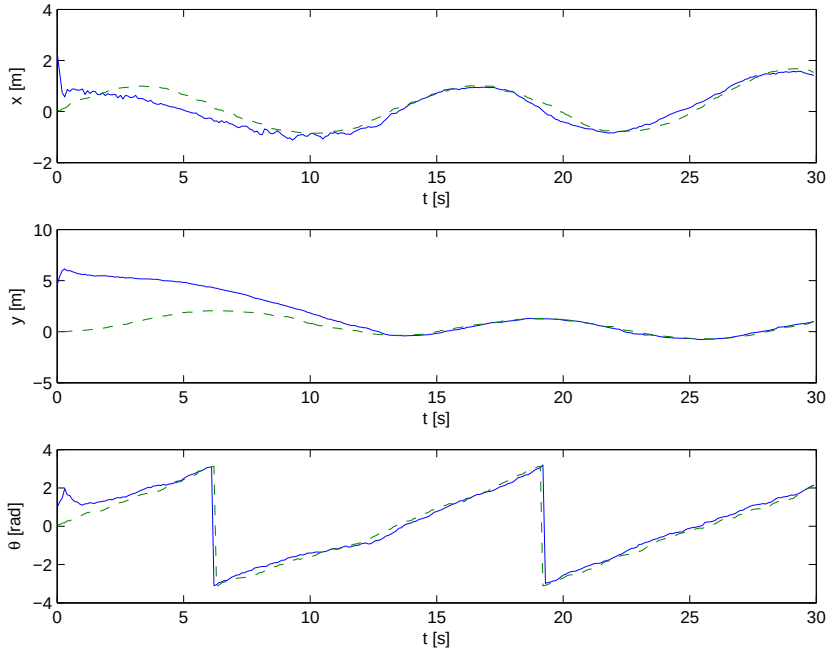
Slika 9.19: Varianca ocene lege robota.



Slika 9.20: Potek inovacije.

```
    XTrueStore=[XTrueStore,xTrue];
    ZStore=[ZStore,zTrue];
    InovStore=[InovStore,inov];

end

t=(0:N-1)*Ts;
figure,subplot(3,1,1),plot(t,XStore(1,:),t,XTrueStore(1,:),'--')
ylabel ('x [m]'), xlabel('t [s]')
subplot(3,1,2),plot(t,XStore(2,:),t,XTrueStore(2,:),'--'),
xlabel('t [s]'), ylabel ('y [m]')
subplot(3,1,3),plot(t,XStore(3,:),t,XTrueStore(3,:),'--'),
xlabel('t [s]'), ylabel ('\theta [rad]')

figure,subplot(1,3,1),plot(t,PStore(1,:)) ylabel ('var(x) [m^2]'),
xlabel('t [s]') subplot(1,3,2),plot(t,PStore(2,:)) xlabel('t
[s]'), ylabel ('var(y) [m^2]') subplot(1,3,3),plot(t,PStore(3,:))
xlabel('t [s]'), ylabel ('\theta [rad^2]')

figure,subplot(1,2,1),plot(t,ZStore(1,:)) ylabel ('d(x) [m]'),
xlabel('t [s]') subplot(1,2,2),plot(t,ZStore(2,:)) xlabel('t
[s]'), ylabel ('\alpha [rad]')

figure, plot(t,InovStore(1,:),t,InovStore(2,:),'--'),xlabel('t
[s]'), ylabel ('inovacija')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = PopraviCiklicnostKota(a)
    a=atan2(sin(a),cos(a));   % prevedemo kot na območje [-pi,pi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati simulacije primera 9.22 so podani v slikah 9.21, 9.22, 9.23 in 9.24. Iz slik vidimo, da sedaj stanja konvergirajo k pravim vrednostim.



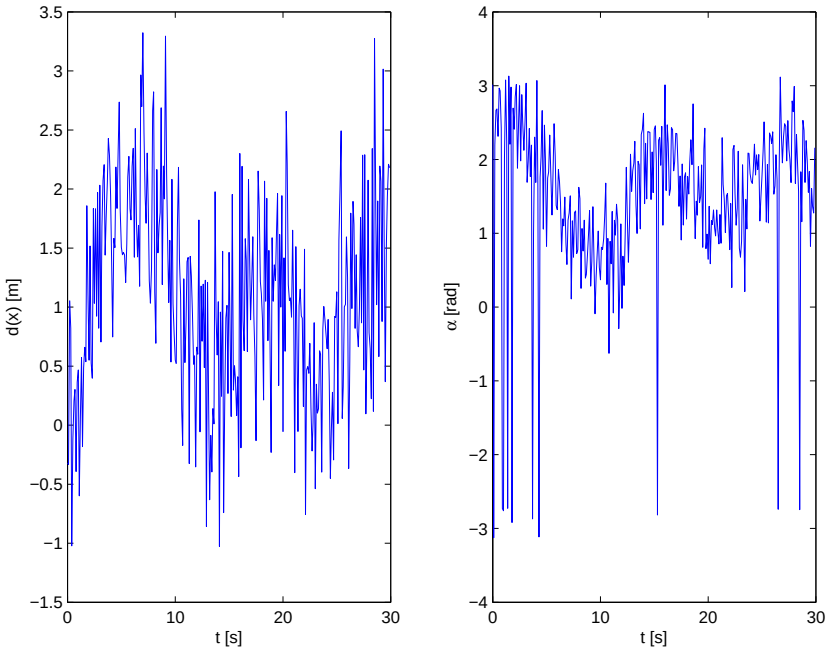
Slika 9.21: Ocena lege robota, ki začne v izhodišču, začetna ocena filtra pa je inicializirana na $\hat{\mathbf{x}} = [3, 3, \frac{\pi}{4}]$. Ocenjena lega robota je označena s polno črto, dejanska lega robota pa črtkano.

9.4.3 Nekatere različice Kalmanovega filtra

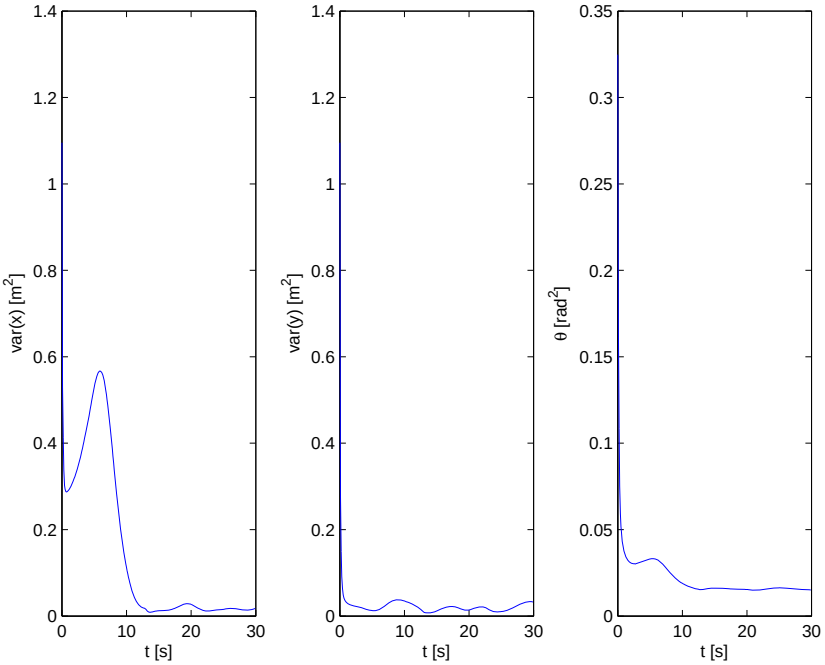
Poleg opisane osnovne verzije Kalmanovega filtra in njegove razširitve na nelinerane sisteme (EKF) obstaja še vrsta izpeljank, od katerih bomo omenili le nekaj najpogostejših.

Unscented Kalman filter se uporablja pri sistemih z bolj izrazito nelinearnostjo, kjer uporaba razširjenega Kalmanovega filtra da slabše rezultate. Kovariančne matrike šuma se tu oceni statistično s pomočjo manjše množice vhodnih točk, ki se preslikajo preko nelinearne funkcije in se nato oceni srednja vrednost in kovariančna matrika. Te točke imenujemo sigma točke, ki so razpršene okoli ocenjene vrednosti z nekim algoritmom (imamo $2n + 1$ točk za dimenzijo n).

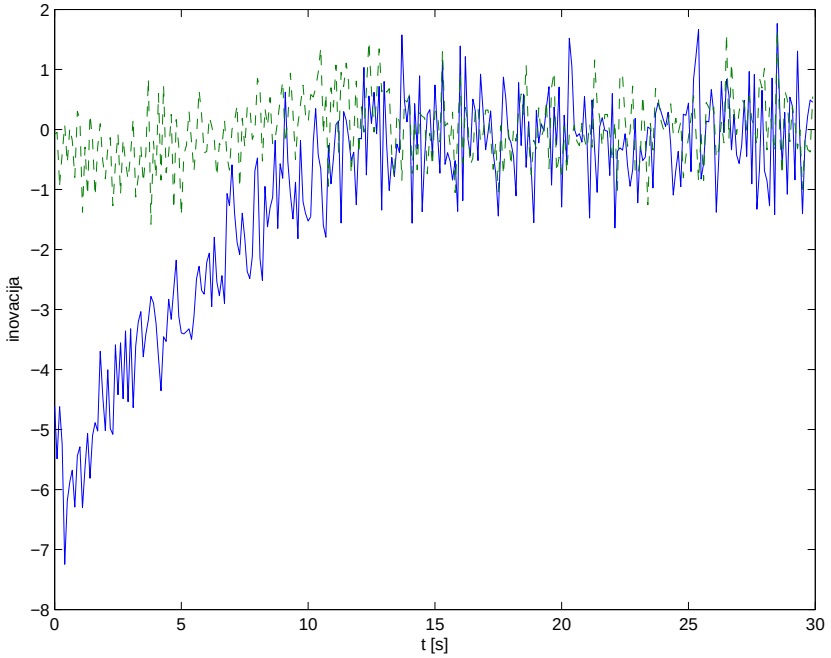
Informacijski filter (Information filter) se ocena kovariančne matrike in ocena stanj zamenjata z informacijsko matriko in informacijskim vektorjem. Informacijska matrika predstavlja inverz kovariančne matrike, informacijski vektor pa je produkt informacijske matrike in ocene vektorja stanj. Gre za dualen pristop Kalmanovemu filtru, kjer se bistveno poenostavi korekcijski korak (le seštevanje brez inverzije matrike). Bolj pa postane je kompleksen



Slika 9.22: Meritve razdalje in kota.



Slika 9.23: Varianca ocene lege robota.



Slika 9.24: Potek inovacije.

predikcijski korak.

Kalman-Bucy filter je Kalmanov filter, ki je podan za zvezni opis sistema.

9.5 Filter delcev

Do sedaj smo obravnavali Bayesov filter, ki se večinoma uporablja za diskreten prostor stanj s končnim številom vrednosti, ki jih spremenljivke stanj lahko zavzamejo. V kolikor želimo uporabiti Bayesov filter za zvezne spremenljivke, lahko te spremenljivke kvantiziramo na končno število področij oziroma vrednosti. Taka uporaba Bayesovega filtra za zvezne spremenljivke stanj se pogosto označuje za histogramski filter (angleško: histogram filter).

Za zvezne spremenljivke stanj bi namreč potrebovali eksplicitno rešitev Bayesovega filtra (9.10)

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}, \mathbf{u}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k})} \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k-1}) \mathbf{d}x_{k-1} \tag{9.21}$$

kar velja ob predpostavki, da so stanja vsebovana ter da obravnavamo Mar-

kov proces (glej poglavje 9.2). Trenutno oceno porazdelitve verjetnosti (9.21) potrebujemo za določitev najbolj verjetne ocene stanj (matematično upanje)

$$E[\hat{\mathbf{x}}_{k|k}] = \int \mathbf{x}_{k|k} \cdot p(\mathbf{x}_k|\mathbf{z}_{1:k}, \mathbf{u}_{1:k}) d\mathbf{x}_{k|k}$$

Eksplicitna rešitev (9.21) pa je možna le za omejen nabor primerov, kjer predpostavimo Gaussove porazdelitve verjetnosti in linearnost sistema, kar rezultira v Kalmanov filter. V primeru nelinearnih sistemov pa lahko nelinearnosti sistema, ki je v modelu gibanja (oz. aktuator) in/ali v modelu meritve linearizirano, kar nas privede do razširjenega Kalmanovega filtra.

Filter delcev (ang. particle filter) pa je bolj splošen pristop, kjer ne zahtevamo Gaussove porazdelitve (Gaussov šum) in linearnosti sistema. Osnovna ideja je, da trenutno oceno (a posteriori) porazdelitve verjetnosti (9.10) po opravljeni meritvi aproksimiramo z množico delcev $\mathbf{x}_k^i$, $i \in 1 \cdots N$, kjer je i indeks delca v množici N delcev. Vsak delec predstavlja svojo vrednost ocenjenega stanja $\mathbf{x}_k^i$, ki je naključno določena (vzorčena) iz porazdelitve verjetnosti (Monte Carlo simulacija). Vsak delec torej podaja svojo hipotezo o dejanskem stanju sistema. Opis porazdelitve verjetnosti s pomočjo naključno generiranih delcev predstavlja neparametričen opis porazdelitve verjetnosti, ki ni omejena le na Gaussovo porazdelitev. Opis porazdelitve z delci nadalje omogoča modeliranje nelinearnih transformacij šuma (model aktuacije in/ali meritev). Z delci torej lahko opišemo porazdelitev šuma, ki se iz vhodov ali meritev prenaša preko nelinearnih transformacij sistema na stanja.

V tabeli 9.4 je podan osnovni algoritem filtra delcev. V algoritmu filtra delcev moramo na začetku določiti začetno populacijo delcev $\hat{\mathbf{x}}_0^i$, $i \in 1 \cdots N$, katere raztros je odvisen od zaupanja (porazdelitve verjetnosti) $p(\mathbf{x}_0)$ v začetno stanje sistema. V kolikor začetnega stanja ne poznamo, je porazdelitev uniforma in so delci porazdeljeni po celotnem prostoru stanj enakoverjetno.

V koraku predikcije izračunamo novo stanje za vsak delec, glede na podan vhod v sistem. Dobljenim novim ocenam stanj za vsak delec dodamo še naključno vrednost šuma, kot ga pričakujemo na vhodnem signalu. Dobimo predikcijo stanja za vsak delec $\hat{\mathbf{x}}_{k|k-1}^i$. Dodani šum nam zagotavlja, da se delci razpršijo in s tem omogočijo sledenje dejanskemu stanju ob prisotnosti različnih motenj.

V korekcijskem koraku ovrednotimo pomembnost delcev, tako da za vsak delec izračunamo odstopanje dejanske meritve $\mathbf{z}_k$ od ocenjene meritve delca $\hat{\mathbf{z}}_k^i$ na podlagi njegove ocene stanja sistema. Odstopanje dejanske in ocenjene meritve je v literaturi pogosto imenovano inovacija ali residual meritve (ang. innovation) in je za vsak delec i podano kot

$$innov_k^i = \mathbf{z}_k - \hat{\mathbf{z}}_k^i$$

Filter_delcev($\hat{\mathbf{x}}^i_{k-1|k-1}, \mathbf{u}_k, \mathbf{z}_k$):

Inicializacija: če $k > 0$ potem:
množico N delcev $\mathbf{x}^i_k$ postavi na naključne
začetne vrednosti stanj glede na porazdelitev $p(\mathbf{x}_0)$.

Predikcija:
za vsak delec $\hat{\mathbf{x}}^i_{k-1|k-1}$ izvedi premik z modelom
premika in znanim vhodom $\mathbf{u}_k$, kateremu dodaj naključno vrednost
glede na šumne lastnosti, ki so del modela premika. Model premika
podaja $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$. Dobljena predikcija delcev je podana z naborom $\hat{\mathbf{x}}^i_{k|k-1}$.

Korekcija:
Za vsak delec $\hat{\mathbf{x}}^i_{k|k-1}$ **oceni vrednost meritve**, ki bi jo sistem izmeril, če bi njegovo stanje ustrezalo stanju delca.

Glede na dejansko izmerjeno meritev in primerjave z ocenjenimi meritvami delcev **oceni pomembnost delcev**.

Določi nov nabor delcev glede na njihovo pomembnost (ang. importance sampling), kjer iz nabora delcev naključno izberete delce z verjetnostjo, ki je proporcionalna njihovi pomembnosti torej $p(\mathbf{z}_k|\hat{\mathbf{x}}^i_{k|k-1})$. Bolj verjetni delci so izbrani večkrat, manj verjetni delci pa manjkrat.

Ocena stanja filtra $\hat{\mathbf{x}}_{k|k}$ je enaka povprečni vrednosti stanj delcev.

Tabela 9.4: Algoritem za filter delcev.

Bolj verjetni delci imajo to odstopanje manjše.

Na podlagi residuala meritve za vsak delec ocenimo njegovo pomembnost oziroma verjetnostno $p(\mathbf{z}_k|\hat{\mathbf{x}}^i_{k|k-1})$, kar predstavlja utež w^i_k za delec i. Utež lahko določimo z Gaussovo porazdelitvijo verjetnosti kot

$$w^i_k = \det(2\pi\mathbf{R})^{-\frac{1}{2}} e^{-\frac{1}{2}(innov^i_k)^T\mathbf{R}^{-1}(\mathbf{i}nnov^i_k)}$$

kjer je $\mathbf{R}$ kovariančna matrika meritve.

Zelo pomemben korak filtra delcev je določitev novega nabora delcev (ang. importance sampling) glede na uteži w^i_k. Iz nabora delcev se naključno izbere N delcev, kjer je verjetnost izbora posameznega delca proporcionalna njegovi uteži w^i_k. Torej se delci z večjo utežjo lahko izberejo večkrat, manj verjetni delca pa manjkrat oziroma nikoli. Postopek izbora nove generacije delcev lahko izvedemo na več načinov, v nadaljevanju podajamo eno od možnosti:

- uteži delcev w^i_k normiramo z vsoto uteži w^i_k, dobimo nove uteži $wn^i_k = \frac{w^i_k}{\sum_{i=1}^N w^i_k}$,
- kumulativno seštejemo normirane uteži, da dobimo komutativne uteži $wc^i_k = \sum_{j=1}^i wn^i_k$, kot nakazuje slika 9.25,
- naključno izberemo N števil od 0 do 1 in pogledamo katerim delcem ustrezajo naključno izbrane vrednosti. Primerjamo torej kumulativne uteži wc^i_k in naključno generirana števila. Glede na sliko 9.25 imajo delci z večjimi utežni večjo verjetnost (na sliki 9.25 zavzemajo več prostora), da jih izberemo.
- izbrane delce uporabimo v korekcijskem koraku za oceno trenutne vrednosti stanja $\hat{\mathbf{x}}_{k|k} = \sum_{i=1}^N w^i_k \hat{\mathbf{x}}^i_{k|k-1}$

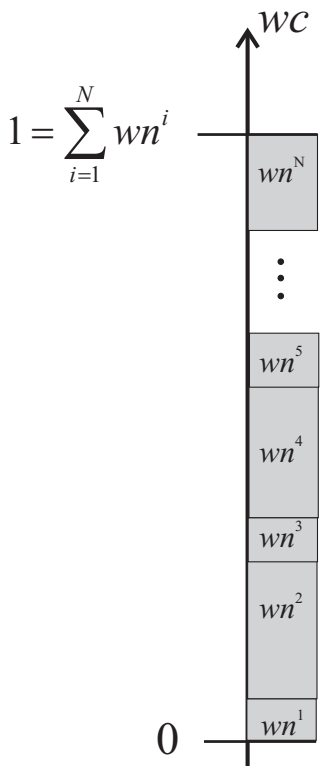
V kolikor sistem miruje (trenutno stanje je enako preteklemu) je priporočeno, da ne določamo novega nabora delcev ampak je bolje, da obdržimo kar stari nabor delcev in le prilagodimo nove uteži, kot je prikazano v [9].

V primeru 9.23 je prikazan primer uporabe filtra delcev.

Primer 9.23. *Z uporabo filtra delcev ocenjujemo najbolj verjetno stanje iz primera 9.22. V implementaciji uporabimo $N = 300$ delcev. Vsi ostali podatki pa so enaki, kot v primeru 9.22.*

Rešitev

V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.



Slika 9.25: Določitev nove populacije delcev v korekcijskem koraku (ang. importance sampling). Delce (oz. njihove uteži) nanizamo enega za drugim, in normiramo, tako da je vsota vseh uteži enaka 1. Bolj verjetni delci zavzamejo večje področje, manj verjetni delci pa manjše področje na intervalu od 0 do 1. Novo populacijo delcev določimo tako, da naključno izberemo N števil in pogledamo, katerim delcem pripadajo.

```
function FilterDelcev

close all, clear all,
Ts=0.1; % cas vzorčenja
Marker=[0 0; 5 5];
xTrue=[0;0;0]; % Dejansko začetno stanje
x=[3;3;pi/4]*1; % Ocenjeno zacetno stanje

P=diag([9,9,0.6]);
% varianca šum aktuatorja za pomik
Q =diag([0.1,0.1]);
% varianca šuma meritve razdalje in kota
R=diag([0.5,0.3]);

% incializacija delcev
nParticles=300;
xP=repmat(xTrue,1,nParticles)+diag([4,4,1])*randn(3,nParticles);

%vsi delci so enako verjetni
W = ones(nParticles,1)/nParticles;

% shranjujemo rezultate za prikaz
XStore=[]; PStore=[]; XTrueStore=[]; ZStore=[];InovStore=[];

%naredimo zanko
N=200; for k = 1:N
    u=[0.5;0.5];    % ukaz za premik (translatorna in kotna hitrost)
    u_sum=u + sqrt(Q)*randn(2,1) ;

    % simulacija dejanskega stanja (lege) robota
    xTrue = xTrue + Ts*[ u_sum(1)*cos(xTrue(3)); ...
                         u_sum(1)*sin(xTrue(3)); ...
                         u_sum(2)   ];
    xTrue(3)=PopraviCiklicnostKota(xTrue(3));

    % simuliramo dejansko pošumljeno meritev (razdalja in orientacija do markerja)
    zTrue=[];
    for m=1:size(Marker,1)
        dist=sqrt((Marker(m,1)-xTrue(1))^2 + (Marker(m,2)-xTrue(2))^2 );
        kot=atan2(Marker(m,2)-xTrue(2),Marker(m,1)-xTrue(1)) - xTrue(3);
        zz=[dist;kot]  + sqrt(R)*randn(2,1);
        zz(2)=PopraviCiklicnostKota(zz(2));
        zTrue =[zTrue; zz ];
    end

    % Predikcija delcev
    for(p = 1:nParticles)
        un=u + sqrt(Q)*randn(2,1)*1 ;    % delce premaknemo s šumom modela
        xP(:,p) = xP(:,p) + Ts*[ un(1)*cos(xP(3,p)); ...
                    un(1)*sin(xP(3,p)); ...
                    un(2)          ]   ;
        xP(3,p) = PopraviCiklicnostKota(xP(3,p));
    end;

     % Korekcija delcev
     for(p = 1:nParticles)
        % ocenjena meritev za vsak delec
        z=[];
        for m=1:size(Marker,1)
            dist=sqrt((Marker(m,1)-xP(1,p))^2 + (Marker(m,2)-xP(2,p))^2 );
            kot=atan2(Marker(m,2)-xP(2,p),Marker(m,1)-xP(1,p)) - xP(3,p);
            zz=[dist;kot];
            zz(2)=PopraviCiklicnostKota(zz(2));
            z =[z; zz ];
        end

        Innov = zTrue-z; %določimo inovacijo

        % izberemo pravo inovacijo za kot zaradi šuma in cikličnosti kota
        for m=1:size(Marker,1)
            iii=zTrue(2*m) - (z(2*m) + [0;2*pi;-2*pi]);
            [tmp,index]=min(abs(iii));
            Innov(2*m)=iii(index);
        end

        % določimo uteži delcev (njihovo verjetnost)
        RR=diag(repmat([R(1,1) R(2,2)],1,size(Marker,1))); % kovariančna matrika meritve
        W(p) = exp(-0.5*Innov'*inv(RR)*Innov)+0.0001;
     end;

     iNextGeneration=dolociNovoGeneracijoDelcev(W,nParticles);
     xP = xP(:,iNextGeneration);
```

```
    % ocena stanja je povprečje delcev
    x = mean(xP,2);
    x(3) = PopraviCiklicnostKota(x(3));
    [gg,ggi]=max(W);  % za kot pa ne vzameš kar povprečje, ampak najbolj verjeten delec
    x(3)=xP(3,ggi);

    figure(1)
    plot(xP(1,:),xP(2,:),'.',xTrue(1),xTrue(2),'x'), axis([-10,10,-10,10]), drawnow

    % shranjujemo rezultate za prikaz
    XStore=[XStore,x];
    PStore=[PStore,diag(P)];
    XTrueStore=[XTrueStore,xTrue];
    ZStore=[ZStore,zTrue];
    InovStore=[InovStore,Innov];
end

t=(0:N-1)*Ts;
figure,subplot(3,1,1),plot(t,XStore(1,:),t,XTrueStore(1,:),'--')
ylabel ('x [m]'), xlabel('t [s]')
subplot(3,1,2),plot(t,XStore(2,:),t,XTrueStore(2,:),'--'),
xlabel('t [s]'), ylabel ('y [m]')
subplot(3,1,3),plot(t,XStore(3,:),t,XTrueStore(3,:),'--'),
xlabel('t [s]'), ylabel ('\theta [rad]')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function iNextGeneration =dolociNovoGeneracijoDelcev(W,nParticles)
    %izberemo glede na uteži delcev
    CDF = cumsum(W)/sum(W);
    iSelect  = rand(nParticles,1); % naključno izbrana števila
    % indeksi novih delcev
    CDFg=[0;CDF];
    indg=[1;(1:nParticles)'];
    iNextGeneration_float = interp1(CDFg,indg,iSelect,'linear');
    iNextGeneration=round(iNextGeneration_float+0.5);   % zaokrožimo indeks navzgor na celo število
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = PopraviCiklicnostKota(a)
    a=atan2(sin(a),cos(a));   % prevedemo kot na območje [-pi,pi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati simulacije primera 9.23 so podani v slikah 9.26 in 9.27.

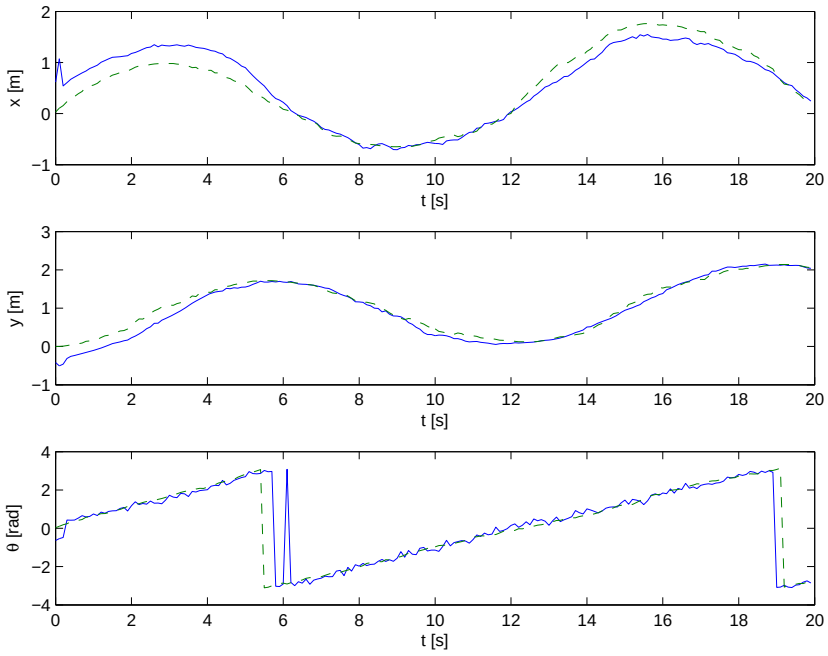
V primeru 9.23 smo poleg razdalje do markerjev merili tudi kot, kjer je potrebno paziti na preskoke kota zaradi cikličnosti. Primer lokalizacije kjer merimo le razdaljo do markerjev in od tod ocenjujemo lego robota prikazuje primer 9.24.

Primer 9.24. *Z uporabo filtra delcev ocenjujemo najbolj verjetno stanje iz primera 9.23, kjer spremenimo meritve, tako da merimo le razdaljo do markerjev. V implementaciji uporabimo* $N = 500$ *delcev. Vsi ostali podatki pa so enaki, kot v primeru 9.23.*

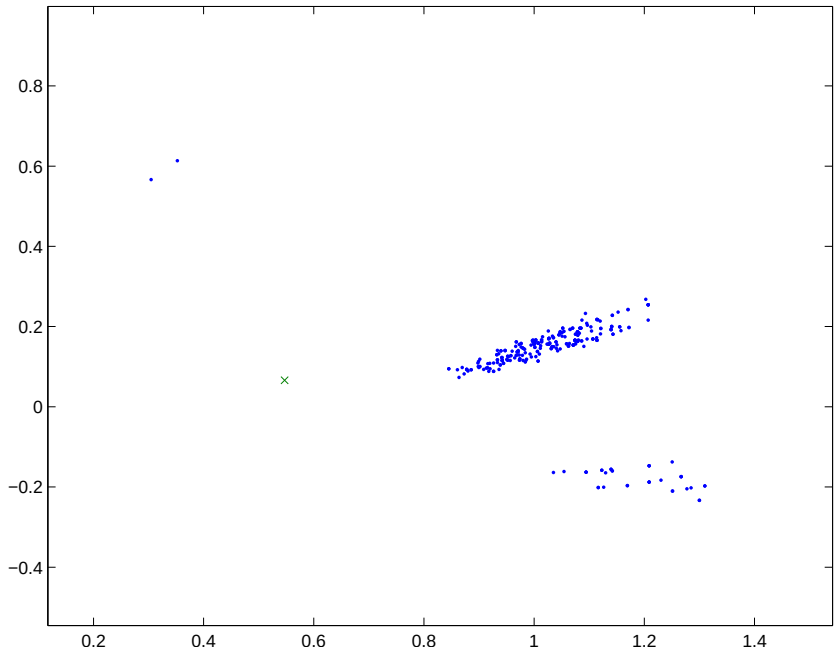
Rešitev

V nadaljevanju je podana koda, komentar in grafični prikaz rešitve.

```
function FilterDelcev close all, clear all,


Ts=0.1; % cas vzorčenja
```



Slika 9.26: Ocena lege robota, ki začne v izhodišču, začetna ocena filtra pa je inicializirana na $\hat{\mathbf{x}} = [3, 3, \frac{\pi}{4}]$. Ocenjena lega robota je označena s polno črto, dejanska lega robota pa črtkano.

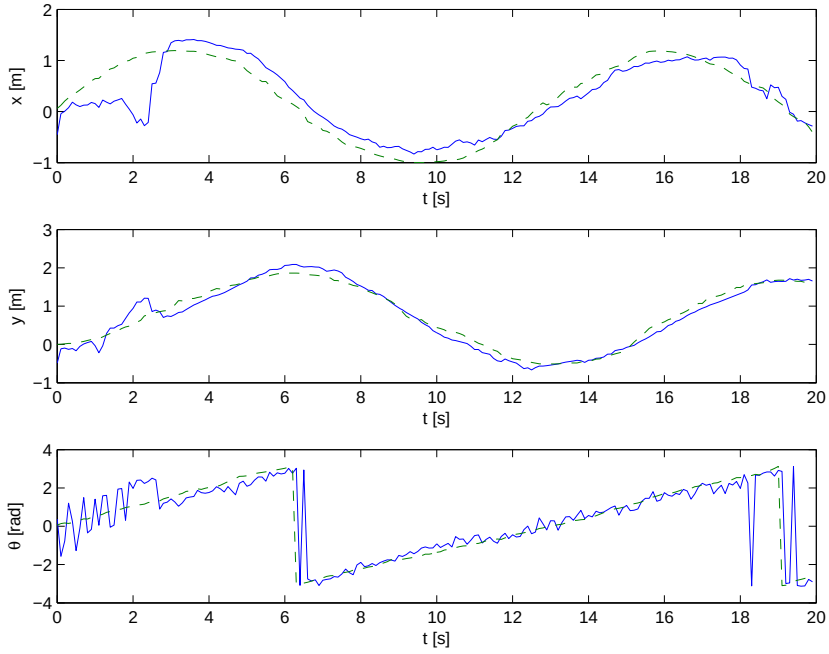


Slika 9.27: Prikaz dejanskega stanja sistema $\times$ in generacijo delcev v koraku $k = 10$.

```
Marker=[0 0; 5 5];

xTrue=[0;0;0];  % Dejansko začetno stanje
x=[3;3;pi/4]*1;  % Ocenjeno zacetno stanje

P=diag([9,9,0.6]);
Q =diag([0.1,0.1]);  % varianca šum aktuatorja za pomik
R=diag([0.5]); %varianca šuma meritve razdalje

% incializacija delcev
nParticles=500;
xP=repmat(xTrue,1,nParticles)+diag([4,4,1])*randn(3,nParticles);

% vsi delci so enako verjetni
W = ones(nParticles,1)/nParticles;

% shranjujemo rezultate za prikaz
XStore=[]; PStore=[]; XTrueStore=[]; ZStore=[];InovStore=[];

%naredimo zanko
N=200;
for k = 1:N
    u=[0.5;0.5];    % ukaz za premik (translatorna in kotna hitrost)
    u_sum=u + sqrt(Q)*randn(2,1) ;

    % simulacija dejanskega stanja (lege) robota
    xTrue = xTrue + Ts*[ u_sum(1)*cos(xTrue(3)); ...
                         u_sum(1)*sin(xTrue(3)); ...
                         u_sum(2)   ];
    xTrue(3)=PopraviCiklicnostKota(xTrue(3));

    % simuliramo dejansko pošumljeno meritev (razdalja in orientacija)
    zTrue=[];
    for m=1:size(Marker,1)
        dist=sqrt((Marker(m,1)-xTrue(1))^2 + (Marker(m,2)-xTrue(2))^2 );
        zz=[dist]  + sqrt(R)*randn(1,1);
        zTrue =[zTrue; zz ];
    end
```

```
   % predikcija
   for(p = 1:nParticles)
        un=u + sqrt(Q)*randn(2,1)*1 ;    % delce premaknemo s šumom modela
        xP(:,p) = xP(:,p) + Ts*[ un(1)*cos(xP(3,p)); ...
                     un(1)*sin(xP(3,p)); ...
                     un(2)         ]   ;
        xP(3,p) = PopraviCiklicnostKota(xP(3,p));
    end;

 %korekcija
 for(p = 1:nParticles)   % ocenjena meritev za vsak delec
    z=[];
    for m=1:size(Marker,1)
        dist=sqrt((Marker(m,1)-xP(1,p))^2 + (Marker(m,2)-xP(2,p))^2 );
        zz=[dist];
        z =[z; zz ];
    end

    Innov = zTrue-z;     %določimo inovacijo

    % določimo uteži delcev (njihovo verjetnost)
    RR=diag(repmat([R(1,1)],1,size(Marker,1))); % kovariančna matrika meritve
    W(p) = exp(-0.5*Innov'*inv(RR)*Innov)+0.0001;
 end;

    iNextGeneration=dolociNovoGeneracijoDelcev(W,nParticles);
    xP = xP(:,iNextGeneration);

    % ocena stanja je povprečje delcev
    x = mean(xP,2);
    x(3) = PopraviCiklicnostKota(x(3));
    [gg,ggi]=max(W);  % za kot pa ne vzameš kar povprečje, ampak najbolj verjeten delec
    x(3)=xP(3,ggi);

    figure(4),plot(xP(1,:),xP(2,:),'.',xTrue(1),xTrue(2),'x'), axis([-10,10,-10,10]), drawnow

    % shranjujemo rezultate za prikaz
    XStore=[XStore,x];
    PStore=[PStore,diag(P)];
    XTrueStore=[XTrueStore,xTrue];
    ZStore=[ZStore,zTrue];
    InovStore=[InovStore,Innov];

end

t=(0:N-1)*Ts;
figure,subplot(3,1,1),plot(t,XStore(1,:),t,XTrueStore(1,:),'--')
ylabel ('x [m]'), xlabel('t [s]')
subplot(3,1,2),plot(t,XStore(2,:),t,XTrueStore(2,:),'--'),
xlabel('t [s]'), ylabel ('y [m]')
subplot(3,1,3),plot(t,XStore(3,:),t,XTrueStore(3,:),'--'),
xlabel('t [s]'), ylabel ('\theta [rad]')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function iNextGeneration =
dolociNovoGeneracijoDelcev(W,nParticles)

    %izberemo glede na uteži delcev
    CDF = cumsum(W)/sum(W);

    iSelect  = rand(nParticles,1); % naključno izbrana števila

    % indeksi novih delcev
    CDFg=[0;CDF];
    indg=[1;(1:nParticles)'];
    iNextGeneration_float = interp1(CDFg,indg,iSelect,'linear');
    iNextGeneration=round(iNextGeneration_float+0.5);   % zaokrožimo indeks navzgor na celo število
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = PopraviCiklicnostKota(a)
    a=atan2(sin(a),cos(a));   % prevedemo kot na območje [-pi,pi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati simulacije primera 9.24 so podani v sliki 9.28, kjer vidimo, da filter konvergira k pravi vrednosti podobno kot v primeru 9.23.



Slika 9.28: Ocena lege robota, ki začne v izhodišču, začetna ocena filtra pa je inicializirana na $\hat{\mathbf{x}} = [3, 3, \frac{\pi}{4}]$. Ocenjena lega robota je označena s polno črto, dejanska lega robota pa črtkano.

Filter delcev je implementacija za zvezne sisteme (zvezni prostor stanj), omogoča opis nelinearnih sistemov in poljubne porazdelitve šuma. Uporaba filtra delcev postane računsko precej zahtevna za večje dimenzije, kjer je za ustrezno konvergenco potrebno veliko število delcev. Število potrebnih delcev narašča eksponentno z dimenzijo prostora stanj.

Dobra lastnost filtra delcev je robustnost in zmožnost rešitve problema globalne lokalizacije in problema ugrabljenega robota. Pri problema globalne lokalizacije je začetna lega (vrednost stanj) neznana, robot se lahko nahaja kjerkoli. V problem ugrabljenega robota, pa robota v določenem trenutku prestavimo (ugrabimo) na poljubno lokacijo in v kolikor je algoritem lokalizacije robusten se lahko prilagoditi nastali negotovosti (pogosto uporabljeno pri testiranju algoritmov za simulacijo večjih napak v lokalizaciji).

Literatura

[1] G. Klančar and I. Škrjanc, Tracking-error model-based predictive control for mobile robots in real time, Robotics and Autonomous Systems, 55(6) (2007) 460-469.

[2] Karba, R.: Modeliranje procesov, Fakulteta za elektrotehniko, Univerza v Ljubljani, 1999.

[3] Matko, D., Karba. R., Zupančič, B.: Simulation and modelling of continuous systems, A case study approach, Prentice Hall International Series in Systems and Control Engineering, Series Editor M. J. Grimble, 1992.

[4] Zupančič, B.: Zvezni regulacijski sistemi - I. del, Fakulteta za elektrotehniko, Univerza v Ljubljani, 2010. (http://msc.fe.uni-lj.si/Download/Zupancic/zrs1.pdf)

[5] Zupančič, B.: Simulacija dinamičnih sistemov, Fakulteta za elektrotehniko, Univerza v Ljubljani, 2010. (http://msc.fe.uni-lj.si/Download/Zupancic/sim.pdf)

[6] Ogata, K.: System dynamics, Third edition, Prentice Hall, 1998.

[7] Franklin, G. F., Powell, J.D., Emami-Naeini, A.: Feedback control of dynamic systems, Third edition, Addison-Wesley Publishing Company, Inc., 1994.

[8] Kuo, B. C.: Automatic control systems, Sixth edition, Prentice Hall, 1991.

Aktualni:

[9] Thrun, S., Burgard, W., Fox, D.: Probalistic Robotics, The Mit Press ,2006.

[10] Bowling, M., Veloso, M.: Motion Control in Dynamic Multi-Robot Environments) , IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA '99. Proceedings, 1999.

[11] C. Canudas de Wit, O. J. Sordalen: Exponential Stabilization of Mobile Robots with Nonholonomic Constraints, IEEE Transactions on Automatic Control, Vol. 37, No. 11, str. 1791-1797, 1992.

[12] J. A. Reeds, L. A. Shepp: Optimal paths for a car that goes both forwards and backwards, Pacific Journal of Mathematics, Vol. 145, No. 2, str. 367-393, 1990.

[13] A. Luca, G. Oriolo, M. Vendittelli: Control of wheeled mobile robots: An experimental overview, RAMSETE - Articulated and Mobile Robotics for Services and Technologies, S. Nicosia, B. Siciliano, A. Bicchi, P. Valigi, Eds., Springer-Verlag, 2001.

[14] Kanayama, Y., Kimura, Y., Miyazaki, F., Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. Proceedings of the 1990 IEEE International Conference on Robotics and Automation, Cincinnati, OH, 1, 384–389.

[15] Samson, C. Time-varying Feedback Stabilization of Car-like Wheeled Mobile Robots. International Journal of Robotics Research, Vol. 12, No. 1, str. 55-64, 1993.

[16] Zdešar, A. Simulacijsko okolje za analizo in sintezo algoritmov vodenja na osnovi digitalne slike, Diplomsko delo, Univerza v Ljubljani, Fakulteta za elektrotehniko, 2010.

[17] R.W. Brockett, Asymptotic stability and feedback stabilization, in: R.W. Brockett, R.S. Millman, H.J. Sussmann (Eds.), Differential Geometric Control Theory, Birkh¨auser, Boston, MA, 1983, pp. 181–191.

[18] Keplerjevi zakoni, http://sl.wikipedia.org/wiki/Keplerjevi_zakoni.

[19] Grm, K., Matematicni model in simulacija sistema za dolocanje orientacije satelita v nizki zemljini orbiti, Diplomsko delo, Univerza v Ljubljani, Fakulteta za elktrotehniko, 2013.

[20] Klančar, G., Stabilizacijski in orientacijski podsistem: delovno poročilo. Fakulteta za elektrotehniko, Univerza v Ljubljani, 2008.

[21] Matko, D., Blažič, S., Mušič, G., Klančar, G., Kinematika, dinamika in vodenje satelitov. Ljubljana: Vesolje-si, 2013.

[22] Wertz, J. Spacecraft Attitude Determination and Control. D. Reidel Publishing Company, Dordrecht, 1978.

[23] Vidmar, M., Tirnice umetnih satelitov, Uporaba vesoljskih tehnologij, Matko Drago, ur., Didakta, Radovljica, 1996.

[24] Klančar, G., Blažič, S., Matko, D., Mušič, G., Image-based attitude control of a remote sensing satellite. Journal of Intelligent & Robotic Systems, Volume 66, Number 3 (2012), 343-357, DOI: 10.1007/s10846-011-9621-1, 2012.

[25] Wikipedija: http://sl.wikipedia.org/wiki/Kvaternion

[26] Lepetić, M., Prediktivno vodenje nestabilnih in neholonomičnih sistemov, Doktorska disertacija, Ljubljana 2005

[27] R. M. Blenden, Regenerative Power Optimal Reaction Wheel Attitude Control, University of Colorado, 2010.

[28] T. Tomažič, Analiza in uglaševanje avtopilotskih sistemov, Diploma, Univerza v Ljubljani, 2007.

[29] B.L. Stevens, F.L. Lewis, Aircraft control and simulation, John Wiley & Sons, 2003.

[30] R. A Brooks, Intelligence Without Representation, Artificial Intelligence, Volume 47, pp. 139-159, 1991.

[31] R. C. Arkin, Motor schema-based mobile robot navigation, The International Journal of Robotics Research, Volume 8, Number 4, pp. 92–112, 1989.

[32] J. Ferber, Multi-agent Systems, An introduction to distributed artifical intelligence, Addison-Wesley, 1999.

[33] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, Principles of Robot motion, Theory, Algorithms and Implementation, MIT Press, 2004.

[34] A. Luca, G. Oriolo, Modelling and control of nonholonomic mechanical systems, Kinematics and Dynamics of Multi-Body Systems, J. Angeles, A. Kecskemethy Eds., Springer-Verlag, Wien, 1995.

[35] X. Yun, State Space Representation of Holonomic and Nonholonomic Constraints Resulting from Rolling Contacts, IEEE International Conference on Robotics and Automation, Nagoya, Japan, str. 2690-2694, 1995.

[36] N. Kneževič, Analiza načrtovanja gibanja znotraj znanega okolja, Dilomsko delo, Univerza v Ljubljani, Fakulteta za elektrotehniko, 2012.