UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Troha

Robotsko učenje in planiranje potiskanja predmetov

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Ivan Bratko

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01645/2010
Datum: 15.03.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MIHA TROHA**

Naslov: **ROBOTSKO UČENJE IN PLANIRANJE POTISKANJA PREDMETOV**
ROBOT LEARNING AND PLANNING FOR PUSHING OBJECTS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Naloga je uporabiti metode strojnega učenja in planiranja za to, da se mobilni robot sam nauči izvajanja opravil, ki zahtevajo premikanje predmetov s potiskanjem. Na začetku predpostavimo, da robot nima modela oz. nobenega znanja o fiziki potiskanja predmetov. Zato mora robot najprej z avtonomnim izvajanjem poskusov in uporabo algoritmov učenja sestaviti napovedni model potiskanja. Zatem naj ta model uporabi za planiranje akcij, s katerimi robot premakne dani predmet na novo lokacijo. Posebej preučite možnost uporabe kvalitativnega učenja in planiranja.

Mentor:

akad. prof. dr. Ivan Bratko

Fakulteta za računalništvo in informatiko
2.
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01645/2010
Datum: 15.03.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MIHA TROHA**

Naslov: **ROBOTSKO UČENJE IN PLANIRANJE POTISKANJA PREDMETOV**
ROBOT LEARNING AND PLANNING FOR PUSHING OBJECTS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Naloga je uporabiti metode strojnega učenja in planiranja za to, da se mobilni robot sam nauči izvajanja opravil, ki zahtevajo premikanje predmetov s potiskanjem. Na začetku predpostavimo, da robot nima modela oz. nobenega znanja o fiziki potiskanja predmetov. Zato mora robot najprej z avtonomnim izvajanjem poskusov in uporabo algoritmov učenja sestaviti napovedni model potiskanja. Zatem naj ta model uporabi za planiranje akcij, s katerimi robot premakne dani predmet na novo lokacijo. Posebej preučite možnost uporabe kvalitativnega učenja in planiranja.

Mentor:

akad. prof. dr. Ivan Bratko

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Miha Troha,

z vpisno številko 63050120,

sem avtor diplomskega dela z naslovom:

Robotsko učenje in planiranje potiskanja predmetov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom prof. dr. Ivana Bratka;
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela;
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 10.6.2010 Podpis avtorja:

Zahvala

Za strokovno vodenje, nasvete in pomoč pri izdelavi diplomskega dela se iskreno zahvaljujem mentorju dr. Ivanu Bratku. Ker sem skoraj celoten čas izdelave diplomskega dela preživel v tujini, bi se mu rad posebej zahvalil za čas in trud, ki ga je vložil zaradi otežene komunikacije.

Posebna zahvala tako za moralno kot tudi finančno podporo tekom celotnega študija gre staršem. Sestri Nini in prijateljem pa se zahvaljujem za marsikatero spodbudno besedo in zanimanje za robota, ki sem ga uporabljal pri izdelavi diplomskega dela.

Aljažu Košmerlju iz Laboratorija za umetno inteligenco bi se rad zahvalil za pomoč pri začetnih nastavitvah CMV knjižnice.

Kazalo

Povzetek **1**

Abstract **3**

1 Uvod **5**
1.1 Motivacija 5
1.2 Zgradba diplomskega dela 6

2 Algoritem QUIN **9**
2.1 Nadzorovano strojno učenje 9
2.2 Kvalitativni modeli 9
2.3 Kvalitativno omejene funkcije 11
2.4 Učenje kvalitativno omejenih funkcij 12
2.5 Kvalitativna drevesa 14

3 Kvalitativno zvesto učenje in algoritem Qfilter **17**
3.1 Izboljšava numeričnih napovedi z uporabo kvalitativnih omejitev 17
3.2 Algoritem Qfilter 18

4 Planiranje **23**
4.1 STRIPS 23
4.2 Razširitev jezika STRIPS 25
4.3 Mehko planiranje 26
4.4 Kvalitativno planiranje 29

5 Implementacija **39**
5.1 Opredelitev problema 39
5.2 Opredelitev glavnega problema in podproblemov 43

6 Upravljanje gibanja **45**
6.1 Problem navidezne poti 45
6.2 Premikanje robota 50
6.3 Zaznavanje trkov med robotom in predmetom 52

7 Učenje modela potiskanja predmetov **53**
7.1 Opredelitev modela potiskanja predmetov 53
7.2 Strategija učenja 58
7.3 Izdelava modela 62
7.4 Rezultati učenja 64

8 Planiranje in izvedba plana **71**
8.1 Splošne določitve problema 71
8.2 Kvantitativno planiranje 73
8.3 Kvalitativno planiranje 78
8.4 Rezultati planiranja 84

9 Zaključki in nadaljnje delo **95**
9.1 Zaključki 95
9.2 Napotki in izkušnje pri uporabi programa QUIN 96
9.3 Nadaljnje delo 97

Seznam slik **100**

Seznam tabel **101**

Literatura **101**

Seznam uporabljenih kratic in simbolov

QUIN - algoritem za učenje kvalitativnih modelov (angl. QUalitative INduction)

QCF - kvalitativno omejena funkcija, ki določa kvalitativne zakonitosti, ki veljajo v nekem sistemu (angl. Qualitatively constrained functions)

LUR - Lokalno Utežena Regresija je metoda numeričnega strojnega učenja (angl. locally weighted regression)

STRIPS - ima dvojni pomen (a) avtomatski planer STRIPS (angl. Stanford Research Institute Problem Solver) in (b) formalni jezik za opis problema planiranja, ki ga planer STRIPS uporablja kot vhod

PDDL - trenutno najpopolnejši jezik za opis problemov planiranja (angl. Planning Domain Definition Language)

CMV - knjižnica, uporabljena za zaznavanje lokacije robota in predmeta (angl. Computer Machine Vision)

PID kontroler - splošen kontrolni mehanizem, ki temelji na povratni zanki (angl. Proportional - Integral - Derivate Controller)

Povzetek

V tem delu se bomo najprej seznanili s paradigmo učenja z eksperimentiranjem. Nadalje si bomo podrobneje pogledali nekatere znane algoritme, ki omogočajo izvedbo učenja z eksperimentiranjem pri uporabi robota. Posebej nas bo zanimalo, če je pri tem mogoče smiselno uporabiti kvalitativne modele. Za začetek bo predstavljen Šucev algoritem QUIN, ki ga lahko uporabimo za izdelavo kvalitativnih modelov. Sledil bo še opis algoritma Qfilter, ki kvalitativne modele uporablja za izboljšavo numeričnih napovedi.

Nadalje se bomo posvetili splošnim konceptom planiranja. Za začetek bo predstavljen pogosto uporabljeni jezik STRIPS z nekaterimi razširitvami. Nadaljevali bomo z osnovnimi značilnostmi mehkega planiranja. Velika računska kompleksnost mehkega planiranja nam bo predstavljala motivacijo za razvoj novega načina planiranja, ki ga bomo poimenovali kvalitativno planiranje. Ideja o kvalitativnem planiranju je po našem vedenju novost na področju planiranja in tako predstavlja največjo dodano vrednost tega diplomskega dela.

V nadaljevanju bo sledila opredelitev konkretnega problema, na katerem bomo preizkusili predstavljene algoritme. Opis problema bo vseboval predmet in robota. Osnovna naloga bo določena kot premik predmeta iz trenutne na želeno lokacijo zgolj z robotovim potiskanjem. Reševanje naloge bo potekalo v dveh fazah. Prva faza bo identificirala zakonitosti, ki veljajo v problemski domeni potiskanja predmetov, druga faza pa bo pridobljeno znanje uporabila za izvedbo naloge.

Opis prve faze bomo pričeli s predstavitvijo učenja kvalitativnega in numeričnega modela potiskanja predmeta. Strategijo zbiranja učnih primerov sem razvil sam, za izdelavo modelov pa sem uporabili že znana algoritma QUIN in Qfilter. Oba naučena modela bosta tudi ovrednotena. Kvalitativni modeli bodo ocenjeni kot intuitivno preprosto razumljivi. Oba modela se bosta na podlagi uporabe pri planiranju izkazala kot pravilna.

Opisu učenja bo sledila predstavitev druge faze, ki jo imenujemo planiranje in izvedba plana. Faza bo realizirana na dva načina. Prvi bo kvalitativno planiranje, drugega pa bomo poimenovali kvantitativno planiranje, saj pri izdelavi

plana ne upošteva kvalitativnih modelov. Oba načina planiranja sem implementiral sam, pri čemer pa je bila teoretična osnova za kvantitativno planiranje že znana. Za konec bomo predstavili rezultate obeh načinov planiranja. Rezultati bodo pokazali, da je kvalitativno planiranje izrazito uspešnejše.

Ključne besede:

učenje z eksperimentiranjem, kvalitativno planiranje, mehko planiranje, kvalitativni modeli, inteligentni robotski sistemi

Abstract

In this work we first examine the paradigm of learning by experimentation. We then detail certain known algorithms which enable learning by experimentation with a view to their implementation in a intelligent robot system. We direct particular focus toward the applicability of qualitative models. We first review Šuc's QUIN algorithm, which can be used to induce quantitative models. We then proceed by examining the Qfilter algorithm, which utilizes qualitative models to enhance numerical predictions.

Further, we discuss the basic concepts of planning. We first review the widely used STRIPS language and certain extensions of it. We continue with an examination of planning under uncertainty. Its high computational complexity propels us to develop a new approach to planning, called qualitative planning. This idea is, according to our knowledge, new and can thus be considered as the most valuable part of this work.

We proceed with a definition of a real world problem which will serve as a base to test the previously described algorithms. The problem definition comprises a robot and an object. The main problem is defined as a shift of the object from its current to a desired location conducted by the robot. The solution consists of two steps. In the first step, models of pushing the box need to be constructed. The models are then applied to perform shifting tasks in the second step.

In a detailed description of the first step, we first present our strategy for learning qualitative and numerical models of pushing the box. The strategy utilizes the already known QUIN and Qfilter algorithms to induce the models. The evaluation of the learned models is also afforded. We consider the qualitative models as intuitively understandable. In view of the usage of the models in planning, both have proven correct.

We proceed with a detailed description of the second step, which is called planning and plan execution. This step is realized in two different ways. The first is qualitative planning and the second is called quantitative planning. The name of the latter stems from the fact that it does not consider qualitative

models in the process of planning. We implemented both of them on our own, however, the theory for quantitative planning is already known. Finally, we give the results of qualitative and quantitative planning. The results indicate that qualitative planning is distinctively better.

Key words:

learning by experimentation, qualitative planning, planning under uncertainty, qualitative models, intelligent robot systems

Poglavje 1

Uvod

1.1 Motivacija

Na področju umetne inteligence poznamo veliko različnih algoritmov, ki omogočajo pridobivanje novih znanj iz podatkov. Tudi načinov zbiranja podatkov, na podlagi katerih lahko nova znanja odkrijemo, je več. Podatke lahko priskrbi človek ali pa so zbrani avtomatsko. Eden izmed načinov avtomatskega zbiranja podatkov je zbiranje s pomočjo robota. Ker je področje zanimivo, vendar ne posebej dobro raziskano, se je ustanovil evropski projekt XPERO [1], v katerega je bila vključena tudi Univerza v Ljubljani. Osnovna ideja o avtomatskem zbiranju podatkov s pomočjo avtonomnih robotov, je pri tem projektu razvita na naslednji način.

Denimo, da robot o zunanjem svetu ne ve ničesar. Zgolj preko poskušanja lahko spoznava osnovne lastnosti in zakonitosti, ki veljajo v svetu, ki ga obkroža. Za smiselno delovanje v takem okolju mora robot stalno zbirati podatke o vplivu svojih dejanj na okolje. V podatkih potem poskuša odkriti zakonitosti, ki jih bo lahko uporabil pri nadaljnjem delovanju. Ta paradigma se imenuje učenje z eksperimentiranjem.

Pred nekaj leti je bil predstavljen algoritem za kvalitativno učenje, imenovan QUIN [2]. Opisan je bil v okviru doktorske disertacije, ki je prejela nagrado za najboljši doktorat s področja umetne inteligence v Evropi [3]. Algoritem je na podlagi vhodnih podatkov zmožen odkriti kvalitativne zakonitosti, ki v teh podatkih veljajo. Zakonitosti so preproste, vendar zelo pomembne, saj človek njihove kršitve hitro zazna. Izkaže se, da lahko kvalitativne zakonitosti s pomočjo algoritma Qfilter uporabimo tudi pri izboljšavah numeričnega učenja [4]. Ta pristop je bil že preizkušen na različnih problemskih domenah in rezultati so bili v večini primerov dobri [5].

Osnovna ideja tega diplomskega dela je povezava obeh predstavljenih področij. Uspešnost kvalitativnega učenja bi radi preizkusili pri učenju z eksperimentiranjem. Ker je upoštevanje celotne paradigme učenja z eksperimentiranjem za izdelavo diplomskega dela preobsežno, smo se omejili zgolj na enega izmed podproblemov. To je učenje z eksperimentiranjem pri potiskanju predmetov.

Najpomembnejša vprašanja, na katera bi radi odgovorili v tem diplomskem delu, so:

- Vprašanje 1: Ali je algoritem QUIN primeren za uporabo na domeni potiskanja predmetov?

  Primernost moramo oceniti na podlagi (a) pravilnosti in (b) intuitivne razumljivosti. Intuitivna razumljivost naj bi bila po [2] pomembna značilnost algoritma QUIN.

- Vprašanje 2: Ali je algoritem Qfilter primeren za uporabo na domeni potiskanja predmetov?

  Primernost moramo oceniti zgolj na podlagi pravilnosti. Intuitivna razumljivost nas pri numeričnih napovedih ne zanima.

- Vprašanje 3: Ali je mogoče kvalitativne zakonitosti uporabiti za zmanjšanje kompleksnosti planiranja in izboljšanje izvedbe plana?

  V primeru pritrdilnega odgovora moramo določiti splošen postopek uporabe kvalitativnih zakonitosti pri planiranju. Seveda se pri tem omejimo na tiste probleme planiranja, kjer je uporaba kvalitativnih modelov smiselna. Ustreznost postopka je potrebno tudi praktično dokazati na domeni potiskanja predmetov.

  V primeru negativnega odgovora moramo jasno pokazati, zakaj kvalitativne zakonitosti ne pripomorejo k zmanjšanju kompleksnosti planiranja in izboljšanju izvedbe plana.

1.2 Zgradba diplomskega dela

Zgradbo diplomskega dela lahko v grobem razdelimo na dva dela. V prvem opišemo teoretično osnovo in predstavimo algoritme, ki so za izvedbo diplomskega dela pomembni. V drugem pa predstavimo, kako smo vsakega izmed algoritmov uporabili pri učenju z eksperimentiranjem.

Prvi del pričnemo s poglavjem 2, kjer se posvetimo algoritmu za kvalitativno učenje, imenovanem QUIN. V poglavju 3 si ogledamo delovanje algoritma Qfilter. V poglavju 4 se posvetimo planiranju, ki predstavlja osnovo za uporabo naučenih zakonitosti pri nadaljnjem delovanju. Pri tem, poleg opisa nekaterih že obstoječih algoritmov planiranja, predlagamo tudi nov način planiranja, imenovan kvalitativno planiranje. Ideja o kvalitativnem planiranju je po našem vedenju novost na področju planiranja in tako predstavlja največjo dodano vrednost tega diplomskega dela.

V drugem delu diplomskega dela se posvetimo implementaciji in prilagoditvi posameznih algoritmov domeni potiskanja predmetov. Predstavitev pričnemo s poglavjem 5, kjer natančno opišemo problem, ki ga v tem diplomskem delu rešujemo. Opis problema vsebuje predmet in robota. Osnovna naloga je določena kot premik predmeta s trenutne na želeno lokacijo zgolj z robotovim potiskanjem. Izvedbo naloge razdelimo na več podproblemov. Vsakemu izmed njih se posvetimo v naslednjih poglavjih.

V poglavju 6 tako predstavimo uporabljene rešitve za upravljanje gibanja robota. Za učinkovito izvedbo upravljanja gibanja robota smo implementirali algoritem A*, PID kontroler in Pure Pursuit.

V poglavju 7 opišemo uporabljene pristope za učenje modela potiskanja predmetov. Za učenje kvalitativnih modelov smo uporabili algoritem QUIN. Numerični model pridobimo z algoritmom Qfilter. V poglavju 7 kvalitativne in numerične rezultate tudi ovrednotimo. Pri tem ugotovimo, da so kvalitativni rezultati intuitivno preprosto razumljivi. Ponujajo dober vpogled v zakonitosti, ki veljajo v domeni potiskanja predmetov. Kasneje, na podlagi uporabe pri planiranju, dokažemo pravilnost tako kvantitativnih kot numeričnih rezultatov.

V poglavju 8 prikažemo, kako smo pridobljene modele uporabili za izvedbo različnih nalog potiskanja predmeta. Uporabo modelov implementiramo na dva načina. Prvi je kvalitativno planiranje. Drugi način pa pri izdelavi plana ne upošteva kvalitativnih zakonitosti in ga zato imenujemo kvantitativno planiranje. Oba algoritma planiranja smo implementirali sami. Z vsakim načinom planiranja izvedemo in zabeležimo rezultate za tri kar najbolj različne naloge potiskanja predmetov. Ob primerjavi rezultatov ugotovimo, da je kvalitativno planiranje izrazito boljše. V dokaz ustreznosti kvalitativnega planiranja bralca tudi povabimo, da si pogleda videoposnetke izvedbe nekaterih nalog.[1]

Zaključke diplomskega dela predstavimo v poglavju 9, kjer predlagamo tudi možne smernice za nadaljnje delo.

[1]Videoposnetek je dostopen na: `http://www.youtube.com/watch?v=3xwwoIEDoQo`.

Poglavje 2

Algoritem QUIN

V tem poglavju bomo predstavili algoritem QUIN [7], ki se uporablja za izdelavo kvalitativnih modelov. Najprej bomo predstavili lastnosti kvalitativnih modelov, nato pa se bomo posvetili njihovi izdelavi.

2.1 Nadzorovano strojno učenje

Strojno učenje lahko po eni izmed delitev razdelimo na nadzorovano in nenadzorovano strojno učenje. Za nadzorovano strojno učenje je značilno, da kot vhod prejme množico učnih primerov. Vsak učni primer je podan kot vektor $n + 1$ spremenljivk, pri čemer je prvih n spremenljivk neodvisnih, $n + 1$ spremenljivka (imenovana tudi razredna spremenljivka) pa je določena z vrednostjo ostalih n spremenljivk. Na podlagi učnih primerov algoritmi strojnega učenja ugotovijo, kakšne povezave veljajo med neodvisnimi in odvisno (razredno) spremenljivko. Ugotovljene povezave oz. odvisnosti so lahko podane v obliki numeričnega ali kvalitativnega modela. Numerični modeli so na primer rezultat regresije. Algoritem QUIN pa izdela kvalitativni model. Tem se bomo bolj natančno posvetili v nadaljevanju.

2.2 Kvalitativni modeli

2.2.1 Abstrakcija

Kvalitativni modeli predstavljajo abstrakcijo numeričnih modelov. Poznamo več vrst abstrakcije, ki so podrobno opisane v [6]. Tukaj samo povzemamo glavno delitev in za lažje razumevanje navajamo primer iz splošno znane

Kvalitativna vrednost	Ustrezna numerična predstavitev
neg	-1
nič	0
poz	1

Tabela 2.1: Predstavitev kvalitativnih vrednosti s števili.

domene enakomernega gibanja.

- Abstrakcija števil v intervale

  Pri uporabi kvalitativnih modelov točne vrednosti števil nadomestimo z intervali. Število intervalov lahko prilagodimo problemu. V skrajnem primeru imamo lahko le tri intervale: neg, nič in poz. Primer: $s(2.3\,s) = 4m \rightarrow s(t_1) = poz$.

- Abstrakcija odvodov v smer spremembe

  Abstrakcijo števil v intervale lahko posplošimo tudi na odvode. Odvod v določeni točki je namreč ravno tako število, le pomen je nekoliko drugačen. Tako so trije intervali v skrajnem primeru naslednji: pada, konst in narašča. Primer: $\frac{\mathrm{d}}{\mathrm{d}t}s(2.3s) = 3m/s \rightarrow s(t1)narašča$.

- Abstrakcija funkcij v monotone relacije

  Namesto natančnega zapisa funkcije lahko uporabimo monotone relacije, ki določajo zgolj monotonost med izbranimi spremenljivkami. Primer: $s = t * 3m/s \rightarrow M^+(s,t)$, kar pomeni, da če se poveča t, se poveča tudi s in obratno.

- Abstrakcija naraščajočih časovnih intervalov

  Namesto tabele vrednosti določene funkcije pri različnih naraščajočih vrednostih neodvisne spremenljivke, lahko pri uporabi kvalitativnih modelov vrednosti zapišemo zgolj v eni vrstici. Primer: $s(zač...kon) = nič...maks/narašča$.

Kvalitativne vrednosti je potrebno predstaviti v računalniku. To je eden izmed razlogov, da bomo v nadaljevanju kvalitativne vrednosti predstavljali numerično. Preslikavo podaja Tabela 2.1.

2.2.2 Matematične operacije nad kvalitativnimi vrednostmi

Ker se kvalitativne vrednosti razlikujejo od kvantitativnih, jim je potrebno prilagoditi tudi matematične operacije. V tej točki bomo predstavili zgolj nasprotno vrednost in razliko, saj ostalih operacij pri implementaciji ne uporabljamo. V obeh primerih predpostavljamo numerično predstavitev kvalitativnih vrednosti.

- Nasprotna vrednost je določena z enačbo 2.1.

$$nasprotnaVrednost(x) = -x \tag{2.1}$$

- Razlika dveh kvalitativnih vrednosti je določena z enačbo 2.2.

$$x - y = \begin{cases} 1; & x > y \\ 0; & x = y \\ -1; & x < y \end{cases} \tag{2.2}$$

Obe operaciji sta definirani tudi nad vektorji kvalitativnih vrednosti. Nasprotna vrednost se izračuna za vsak element posebej. Razlika vektorjev pa je določena z razliko isto ležečih elementov. Smiselna je zgolj za vektorje enakih dolžin.

2.3 Kvalitativno omejene funkcije

Za razumevanje delovanja algoritma QUIN je pomembna predvsem abstrakcija števil v intervale in abstrakcija funkcij v monotone relacije, ki pa je v primeru QUINa nekoliko prilagojena. Namesto monotonih relacij v tem primeru nastopajo kvalitativno omejene funkcije (angl. qualitatively constrained functions). Kvalitativno omejena funkcija $M^{s_1,s_2,...,s_m} : \mathbb{R}^m \mapsto \mathbb{R}, s_i \in \{+,-\}$ predstavlja funkcijo z m zveznimi spremenljivkami, ki jim ustrezajo kvalitativne omejitve $s_1, s_2, ..., s_m$. Kvalitativna omejitev $s_i = +$ ($s_i = -$) določa, da funkcija $M^{s_1,s_2,...,s_m}$ v odvisnosti od i-te spremenljivke strogo narašča (pada), če vse ostale spremenljivke ohranijo konstantno vrednost. Če se spremeni vrednost več spremenljivk, potem lahko sklepamo na spremembo funkcije $M^{s_1,s_2,...,s_m}$, samo če so vse spremembe spremenljivk v smeri naraščanja (vrednost spremenljivk s pozitivno odvisnostjo $s = +$ se poveča, vrednost spremenljivk s negativno odvisnostjo $s = -$ pa zmanjša) ali padanja (vrednost

ΔT	ΔV	Δp
1	0	1
-1	0	-1
0	1	-1
0	-1	1
1	-1	1
-1	1	-1
1	1	/
-1	-1	/

Tabela 2.2: Prikaz vseh možnih kombinacij sprememb temperature T in volumna V. Za vsako spremembo je podana tudi sprememba tlaka p, tako da ustreza kvalitativno omejeni funkciji $p = M^{+,-}(T, V)$. Zadnji vrstici predstavljata spremembo, pri kateri je vrednost tlaka p nedefinirana.

spremenljivk z pozitivno odvisnostjo $s = -$ se poveča, vrednost spremenljivk z negativno odvisnostjo $s = +$ pa zmanjša) funkcije $M^{s_1,s_2,\ldots,s_m}$. V prvem primeru se vrednost funkcije $M^{s_1,s_2,\ldots,s_m}$ poveča, v drugem se zmanjša, v vseh ostalih primerih pa je nedefinirana.

Kvalitativno omejene funkcije si lahko pogledamo na domeni plinskega zakona, ki je določen z zvezo $\frac{p*V}{T} = const.$ Če tlak p izberemo kot odvisno spremenljivko, potem lahko zvezo zapišemo kot kvalitativno omejeno funkcijo v obliki $p = M^{+,-}(T, V)$. V Tabeli 2.2 so podane vse možne kombinacije sprememb prostornine V in temperature T. Tem spremembam je prirejena tudi sprememba tlaka p, ki ustreza podani kvalitativni omejeni funkciji.

Dodatno naj še omenimo, da plinski zakon ni konsistenten s kvalitativno omejeno funkcijo $p = M^{+}(T)$, ki trdi, da se ob povečanju temperature T vedno poveča tudi tlak p. V primeru, da se temperatura T malo poveča, volumen V pa zelo poveča, se bo tlak p zmanjšal. To pa ne ustreza napovedi kvalitativno omejene funkcije.

2.4 Učenje kvalitativno omejenih funkcij

Denimo, da imamo množico učnih primerov, kjer je vsak primer podan kot vektor $n+1$ spremenljivk, pri čemer je prvih n spremenljivk, $n+1$ spremenljivka pa predstavlja razredno spremenljivko. Učenje kvalitativnih omejenih funkcij iz tako podanih učnih primerov lahko v grobem razdelimo na tri zaporedne podnaloge:

1. Določitev množice P vseh možnih kvalitativno omejenih funkcij

   Kvalitativno omejene funkcije za n neodvisnih vhodnih spremenljivk se lahko razlikujejo tako v množici spremenljivk, ki jih vzamejo v obzir, kot tudi v vrsti odvisnosti (+, -) za vsako izmed spremenljivk. Množica spremenljivk, ki jih vzamejo v obzir so kombinacije m neodvisnih spremenljivk za $m = 1, 2, .., n$ iz celotne množice n neodvisnih spremenljivk. Če celotno množico vseh mogočih kombinacij spremenljivk označimo s S, potem je njena moč podana z enačbo 2.3.

   $$|S| = \sum_{i=1}^{n} \mathrm{C}_n^i \tag{2.3}$$

   Za vsak element iz množice S lahko nadalje ločimo 2^m kvalitativnih omejenih funkcij, ki se razlikujejo zgolj po vrsti odvisnosti.

   Primer: če učni primeri predstavljajo različne vrednosti plinskega zakona, je $P = \{p = M^+(T), p = M^-(T), p = M^+(V), p = M^-(V), p = M^{+,+}(T,V), p = M^{+,-}(T,V), p = M^{-,+}(T,V), p = M^{-,-}(T,V)\}$.

2. Določitev kvalitativnih sprememb za vsak par učnih primerov

   Ker kvalitativne omejene funkcije operirajo s spremembami spremenljivk in ne kar z njihovimi vrednostmi, je potrebno učne primere preurediti tako, da izražajo spremembe. Sprememba je po definiciji relativna, torej vezana na neko drugo vrednost. Do sprememb lahko tako pridemo z vektorskim odštevanjem dveh različnih učnih primerov. To lahko naredimo za vsak par učnih primerov. Tako iz n učnih primerov dobimo $\frac{n*(n-1)}{2}$ preurejenih učnih primerov, ki izražajo spremembe. Vsakega nato še abstrahiramo v kvalitativno vrednost. Rezultat je zelo podoben Tabeli 2.2, le da lahko kakšna izmed kombinacij vrednosti neodvisnih spremenljivk manjka, kakšna pa je lahko vsebovana večkrat.

3. Ugotoviti, katera iz množice kvalitativnih funkcij P najbolje ustreza podatkom, pridobljenim v točki 2

   Postopek pravi, da za vsako kvalitativno omejeno funkcijo iz množice P preštejemo vse primere v podatkih, pridobljenih v točki 2, kjer je napoved funkcije napačna (N_{nap}), nedefinirana (N_{nedef}) ali pravilno definirana (N_{prdef}). Ta števila lahko potem z enačbo 2.4 združimo v t. i. ceno $E(g)$ kvalitativno omejene funkcije. Največ k ceni prinese število napačnih napovedi N_{nap}, nato število nedefiniranih napovedi N_{nedef}, nekaj

pa pripomore tudi število pravilnih napovedi N_{prdef}, število vseh neodvisnih spremenljivk n in število neodvisnih spremenljivk, ki določajo kvalitativno omejeno funkcijo m. Najboljša je tista kvalitativna omejena funkcija, ki ima najnižjo ceno.

$$E(g) = \log_2 n + m(\log_2 n + 1) + \log_2 N_{prdef} + \\ N_{nap}(\log_2 N_{prdef}) + \log_2 N_{nedef} + N_{nedef} \tag{2.4}$$

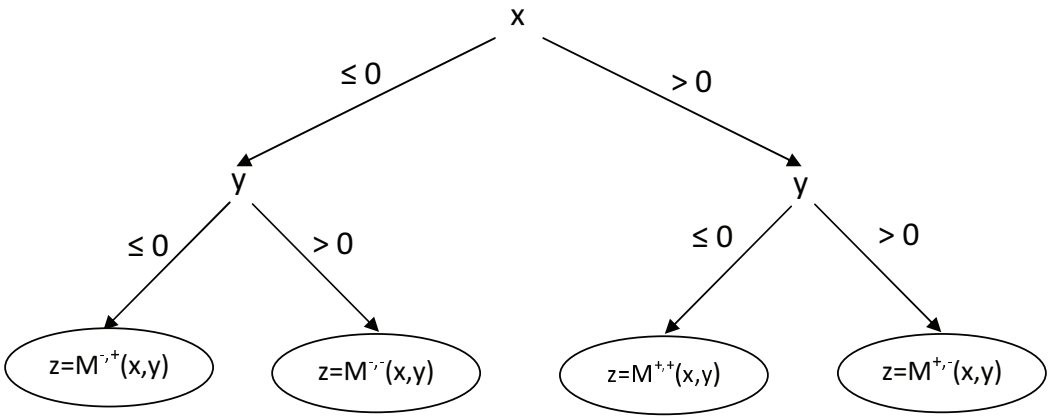
Enačba 2.4 temelji na principu najkrajše opisne dolžine (angl. minimum description length). Podrobnejša razlaga je dostopna v [7]. Namesto zgolj štetja različnih (N_{nap}, N_{nedef} in N_{prdef}) napovedi funkcije lahko pri izračunu cene upoštevamo tudi razdaljo med neodvisnimi spremenljivkami in t. i. konsistentnost spremembe razredne spremenljivke. Več o tem pristopu je opisano v [7].

2.5 Kvalitativna drevesa

Namesto zgolj učenja najboljše kvalitativne omejene funkcije za vse podatke skupaj, je včasih bolje podatke razdeliti na dva dela in se naučiti kvalitativno omejeno funkcijo za vsak del posebej. Enak razmislek pa velja tudi za vsakega izmed tako pridobljenih delov in naprej. Na ta način dobimo drevesno strukturo, kjer vozlišča predstavljajo napotke za delitev celotnega prostora podatkov, listi pa kvalitativne omejene funkcije, ki najbolje ustrezajo določenemu delu podatkov. Primer kvalitativnega drevesa za funkcijo $z = x^2 + y^2$ prikazuje Slika 2.1.

Pri izdelavi drevesa se pojavi vprašanje, katera delitev je v nekem vozlišču najboljša, torej taka, da bodo kvalitativno omejene funkcije v listih kar najbolje opisovale učne primere. Za odgovor na to vprašanje je potrebno pojem cene, ki je opisan v razdelku o kvalitativno omejenih funkcijah, nekoliko posplošiti. Poleg cene posamezne kvalitativne funkcije je potrebno definirati tudi ceno vozlišča v drevesu. Pri algoritmu QUIN je ta določena s ceno levega poddrevesa E_{levi}, desnega poddrevesa E_{desni} in ceno razdelitve. Natančneje ceno določata enačbi 2.5, kjer $stRazdelitev_i$ predstavlja število vseh možnih razdelitev pri dani neodvisni spremenljivki X_i (kar je enako tudi številu vseh različnih vrednosti neodvisne spremenljivke X_i), n pa število vseh neodvisnih spremenljivk.

$$E_{vozlisca} = E_{leva} + E_{desna} + cenaRazdelitve \\ cenaRazdelitve = \log_2 n + \log_2(stRazdelitev_i - 1) \tag{2.5}$$



Slika 2.1: Primer kvalitativnega drevesa za funkcijo $z = x^2 + y^2$.

QUIN za izdelavo drevesa uporablja požrešni algoritem. Celoten postopek se začne z množico, ki vsebuje vse učne primere. Na množici se preizkusi vse možne delitve in za vsako se poskuša izračunati ceno. Ker cene posameznih poddreves v trenutku delitve še niso znane, jih je potrebno za vsako izmed delitev rekurzivno izračunati. Rekurzija oz. delitev se konča, ko je za izbrani del učnih primerov cena nadaljnje najboljše delitve večja kot cena najboljše kvalitativne omejene funkcije. Cene posameznih kvalitativno omejenih funkcij se potem vrnejo očetom, ki z uporabo enačbe 2.5 izračunajo ceno vozlišča. Ta cena se ponovno poda očetom vozlišč. Tako poteka vse do korena drevesa. Drevo z najnižjo ceno v korenu predstavlja končni rezultat učenja.

Kvalitativna drevesa lahko uporabimo za reševanje različnih nalog. V tem diplomskem delu sta to izboljševanje numerične predikcije in kvalitativno planiranje. Nekateri izmed ostalih področij uporabe so dostopni v [2].

Poglavje 3

Kvalitativno zvesto učenje in algoritem Qfilter

Pri uporabi numeričnih metod za strojno učenje niso redki primeri, ko so rezultati teh metod očitno napačni. V [4] je opisan primer, ki ga bomo tukaj povzeli.

Denimo, da imamo posodo, ki je napolnjena z vodo. V spodnjem delu posode zvrtamo luknjico, tako da lahko voda iz posode odteka. Podatke o nivoju vode v posodi beležimo pri različnih časih. Poskus nato večkrat ponovimo pri različnih začetnih gladinah. Iz tako pridobljenih podatkov želimo napovedati gibanje nivoja vode za poljubno začetno gladino. Pri pregledu rezultatov lahko vidimo, da obstajajo časovno zaporedne napovedi, ko se nivo vode v posodi poveča ali pa ima celo negativno vrednost. Povsem očitno je, da so napovedi v teh primerih napačne. Tako očitno je zgolj iz razloga, ker napovedi niso samo kvantitativno ampak tudi kvalitativno napačne. Take napake pa človeško oko zazna zelo hitro. V tem poglavju je opisan postopek, ki lahko te napake odpravi.

3.1 Izboljšava numeričnih napovedi z uporabo kvalitativnih omejitev

V prejšnjem poglavju smo si ogledali algoritem QUIN, katerega rezultat je kvalitativno drevo, ki se kar najbolj prilega učnim primerom. Če bi lahko tako pridobljen kvalitativni model uporabili nad numeričnimi napovedmi, bi lahko odkrili mesta, kjer so napovedi kvalitativno napačne. Te napovedi bi potem popravili. Bolj natančno lahko ta proces opišemo z naslednjimi koraki (glej

Sliko 3.1):

1. pridobiti učne primere, kjer je vsak primer podan kot vektor $n + 1$ spremenljivk, pri čemer je prvih n spremenljivk neodvisnih, $n + 1$ spremenljivka pa predstavlja razredno spremenljivko;

2. z uporabo algoritma QUIN inducirati kvalitativni model;

3. z uporabo numeričnih metod za strojno učenje (lokalno utežena regresija, regresijska drevesa ...) podati numerične napovedi za razredno spremenljivko v želenih točkah;

4. za napovedi, pridobljene v točki 3, pregledati, ali se ujemajo s kvalitativnim modelom, pridobljenim v točki 2, in morebitne anomalije popraviti.

Podnalogi, določeni v točki 3 in 4, sta rešeni z algoritmom Qfilter[4].

3.2 Algoritem Qfilter

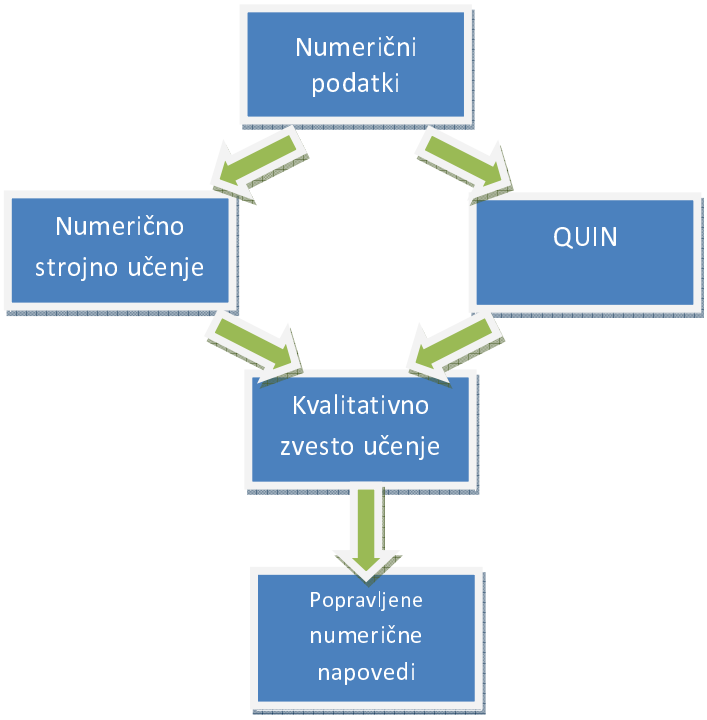
Kot že omenjeno lahko delovanje algoritma Qfilter razdelimo na dva dela. Prvi del je izdelava numeričnih napovedi razredne spremenljivke. Za to je uporabljena metoda strojnega učenja, imenovana lokalno utežena regresija (angl. Locally weighted regression). Drugi del pa je popravljanje numeričnih napovedi, tako da ustrezajo kvalitativnemu modelu. Za to pa je uporabljena metoda, imenovana kvalitativno zvesto učenje. Obe omenjeni metodi sta razloženi v nadaljevanju.

3.2.1 Lokalno utežena regresija

Lokalno utežena regresija (LUR) je metoda strojnega učenja za numerično predikcijo. Za njeno razumevanje je najbolje, če si najprej pogledamo pomen vsake izmed besed od zadaj naprej.

- Regresija

  Regresija je numerična metoda, pri kateri poskušamo najti funkcijo, ki se kar najbolje prilega učnim primerom. Ker je problem računsko zahteven, se večina implementacij omeji zgolj na ugotavljanje najprimernejše funkcije iz množice linearnih funkcij. V tem primeru lahko problem definiramo na naslednji način.



Slika 3.1: Postopek izboljšave numeričnih napovedi z uporabo algoritma QUIN in Qfilter.

Denimo, da imamo n neodvisnih spremenljivk. Vrednost odvisne spremenljivke je tako določena z enačbo 3.1, kjer w_0 predstavlja prosti člen, $w_1, ..., w_n$ pa so uteži, ki določajo pomembnost posamezne neodvisne spremenljivke. Uteži se izračuna tako, da se napovedi kar najbolj prilegajo učnim primerom. Podrobnosti izračuna uteži za več spremenljivk so dostopne v [12].

$$f(x_1, x_2, ..., x_n) = w_0 + x_1 * w_1 + x_2 * w_2 + ... + x_n * w_n \tag{3.1}$$

- Utežena

  Že pri opisu regresije smo spoznali, da se uteži lahko uporablja, da različnim faktorjem priredimo različno pomembnost. V okviru LUR ta termin predstavlja različno pomembnost posameznih učnih primerov. Namesto iskanja linearne kombinacije (regresijskih uteži), kjer so vsi učni primeri enako pomembni, lahko v primeru vnaprej podanega testnega

primera, učnim primerom, ki so bližje testnega, dodelimo večjo pomembnost. Regresijske uteži bodo v tem primeru izračunane tako, da se bo napoved bolje prilegala učnim primerom, ki so blizu testnemu. Napovedi za posamezen testni primer so tako boljše. Negativna stran tega pristopa pa je, da je vrednost odvisna od testnega primera. To pomeni, da je potrebno celoten postopek ponoviti za vsak testni primer. Pri uporabi regresije brez uteži se celotni postopek učenja izvede samo enkrat.

- Lokalna

  Termin v kontekstu LUR pomeni, da se za izračun najboljše funkcije upošteva samo tiste učne primere, ki so blizu testnemu. To je zgolj nekoliko rigoroznejša omejitev od uteži. Lahko si namreč predstavljamo, da se učnim primerom, ki so od testnega preveč oddaljeni, enostavno dodeli vrednost uteži 0. Ta dodatna omejitev izračun napovedi odvisne spremenljivke precej pohitri.

Lokalno utežena regresija torej deluje tako, da najprej omeji množico učnih primerov zgolj na tiste, ki so dovolj blizu testnemu. Vsakemu izmed preostalih učnih primerov potem dodeli še utež, ki je določena z oddaljenostjo od testnega primera. Na tako uteženih učnih primerih se potem uporabi regresijo.

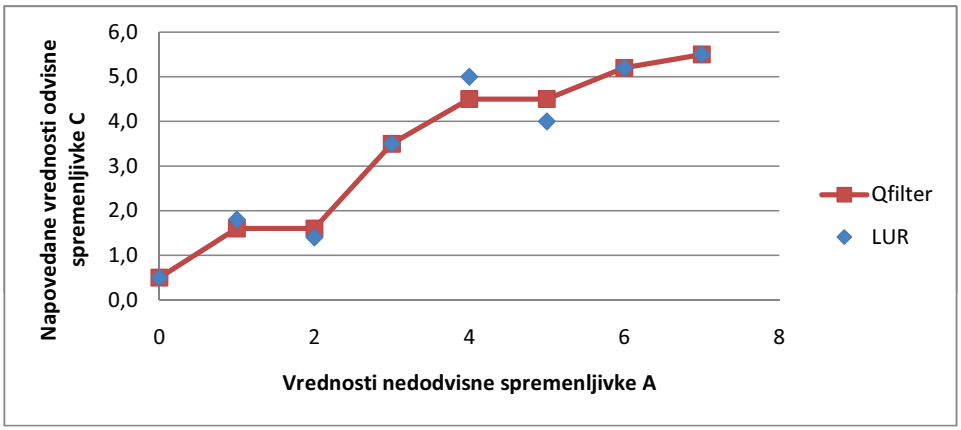
Opis LUR, podan v tem oddelku ni posebej podroben, vendar pa zaobjema vse koncepte, ki so pomembni za razumevanje tega diplomskega dela. Bolj podroben in tudi matematično popoln opis pa je podan v [13].

3.2.2 Kvalitativno zvesto učenje

V okviru algoritma Qfilter je implementirana tudi metoda, imenovana kvalitativno zvesto učenje. Osnovna ideja te metode je spremeniti napovedane numerične vrednosti razredne spremenljivke tako, da le-te ustrezajo kvalitativnim omejitvam. Ker lahko to naredimo na mnogo načinov, je potrebno definirati mero, ki bo omogočala razlikovanje uspešnosti posameznih načinov. Želja je, da so spremembe numerične napovedi razredne spremenljivke, ki jih izvedemo pri popravljanju, kar najmanjše. Smiselna mera je tako kar vsota kvadratov vseh sprememb. Najboljši način je tisti, ki ima najmanjšo vrednost tako določene mere.

Za lažje razumevanje si delovanje kvalitativno zvestega učenja poglejmo na primeru, ki je povzet po [4]. Podanih imamo osem testnih primerov, ki so določeni z $(a_i, c_i), i = 0, 1, 2, ..., 7$, kjer a_i predstavlja neodvisno c_i pa razredno spremenljivko (glej Sliko 3.2). Kvalitativna omejena funkcija, ki bi ji morali testni primeri ustrezati, je $C = M^+(A)$. To lahko v primeru naraščajočih

vrednosti a_i opišemo tudi z zahtevo $c_{i+1} > c_i$. Vidimo, da je neenakost kršena pri $i = 1$ in $i = 4$ (glej Sliko 3.2). Da bo neenakost vedno dosežena, je potrebno nekatere točke premakniti. Če d_i označuje velikost premika i-te točke, potem mora veljati $c_{i+1} + d_{i+1} > c_i + d_i$ za $i = 0, 1, ..., 6$. Naloga pa je najti take d_i, da bo vrednost funkcije $\sum_{i=0}^{7} d_i^2$ minimalna.



Slika 3.2: Prikaz delovanja Qfilter algoritma: Modre točke predstavljajo numerične napovedi, pridobljene z lokalno uteženo regresijo, rdeče točke pa predstavljajo popravljene napovedi, ki ustrezajo kvalitativno omejeni funkciji $C = M^+(A)$.

Rešitev te naloge spada v domeno optimizacijskih problemov, ki se jih da uspešno rešiti s kvadratičnim programiranjem. Podrobnosti o prevedbi problema v kvadratično programiranje so dosegljivi v [4]. Primer, ki smo si ga ogledali, opisuje delovanje algoritma Qfilter, ko imamo samo eno neodvisno spremenljivko. Posplošitev delovanja na več neodvisnih spremenljivk se ne razlikuje veliko od delovanja pri eni neodvisni spremenljivki. Dodatno je potrebno le še upoštevati nekatere dodatne lastnosti kvalitativnih omejenih funkcij. Tudi podrobnosti o posplošitvi so dostopne na [4].

Predstavljen pristop se izkaže za zelo uporabnega. Popravljeni rezultati v večini problemskih domen bolje opisujejo resnično dogajanje kot nepopravljeni. Podrobni rezultati so predstavljeni v [5].

Poglavje 4

Planiranje

Planiranje je področje umetne inteligence, ki se ukvarja z načrtovanjem zaporedja akcij, ki spremenijo trenutno stanje v želeno stanje. V tem poglavju bomo predstavili različne probleme planiranja in za preprostejše predstavili tudi načine reševanja. Za enega izmed težjih primerov bomo na koncu predstavili tudi nov pristop planiranja, ki pri planiranju uporablja kvalitativne modele.

4.1 STRIPS

Eden izmed prvih uspešnejših pristopov za reševanje problemov planiranja je STRIPS. Ime ima danes dvojni pomen in se lahko uporablja kot:

- avtomatski planer STRIPS (Stanford Research Institute Problem Solver) ali
- formalni jezik za opis problema planiranja, ki ga planer STRIPS uporablja kot vhod.

V tej diplomskem delu se uporaba kratice vedno nanaša na drugi pomen.

4.1.1 Opredelitev problema

Za predstavitev problema si oglejmo preprost problem planiranja pri menjavi žarnice. Celotno opravilo je sestavljeno iz točno določenega zaporedja akcij, ki jih moramo izvesti v pravilnem vrstnem redu, da je opravilo končano. Pri menjavi žarnice moramo na primer prinesti stol iz kuhinje v dnevno sobo, stopiti na stol, odviti pregorelo žarnico, stopiti s stola, iti po novo žarnico,

stopiti na stol, priviti novo žarnico, stopiti s stola in na koncu še pospraviti stol nazaj v kuhinjo. Pri tem primeru vidimo, da med akcijami veljajo nekatere zakonitosti, saj ne moremo stola odnesti nazaj iz dnevne sobe v kuhinjo, če ga nismo najprej prinesli iz kuhinje v dnevno sobo. Podobno ne moremo priviti ali odviti žarnice, če nismo pred tem stopili na stol. Iz primera menjave žarnice lahko ugotovimo, da akcijo določa tako sprememba, ki jo akcija povzroči, kot tudi predpogoji, ki morajo biti izpolnjeni, da se akcija lahko izvede. Vsako akcijo A lahko tako bolj formalno definiramo kot trojček $\langle P, V, R\rangle$, kjer imajo posamezni elementi naslednji pomen:

- P predstavlja množico predpogojev, ki morajo biti izpolnjeni, da se akcija lahko izvede (npr. za odvijanje pregorele žarnice moramo biti na stolu);
- V predstavlja množico relacij, ki se z izvedbo akcije vzpostavijo (npr. po stopanju na stol smo potem na stolu);
- R predstavlja množico relacij, ki se z izvedbo akcije porušijo (npr. po stopanju na stol nismo več na tleh).

Poleg akcij je problem planiranja določen še z začetnim stanjem S (npr. stol je v kuhinji, pokvarjena žarnica je v lestencu, nova žarnica je v kuhinji, človek stoji v kuhinji) in želenim stanjem G (npr. stol je v kuhinji, nova žarnica je v lestencu, stara žarnica je v kuhinji).

4.1.2 Izdelava plana

Ob definiranih akcijah, začetnem in želenem stanju se pojavi vprašanje, katero zaporedje akcij povzroči spremembo iz začetnega v želeno stanje. Načinov reševanja tega problema je več. Tukaj bomo opisali le najpreprostejšega. Zaradi specifik planiranja v okviru tega diplomskega dela smo mnenja, da tudi kompleksnejši načini reševanja (v okviru predstavitve STRIPS) ne bi bili uspešnejši. Vsi nam znani algoritmi so namreč boljši zgolj pri reševanju določenih problemov, ki pa v okviru tega diplomskega dela ne nastopijo. To je tudi razlog, da se v njihove podrobnosti ne spuščamo, so pa dostopni v [6].

Najbolj preprost način reševanja problema planiranja v okviru jezika STRIPS se imenuje analiza sredstev in ciljev (angl. means-ends analysis). Pristop je v strnjeni obliki prikazan z naslednjimi tremi točkami:

1. izvajanje se prične s primerjavo želenega in trenutnega stanja. Razlika med želenim in trenutnim stanjem predstavlja cilje, ki jih je še potrebno doseči. Enega izmed teh ciljev se izbere za nadaljnjo obdelavo;

2. poišče se vse akcije, ki vzpostavijo izbrani cilj. Za nadaljnjo obdelavo se izbere eno izmed teh akcij;

3. pregleda se množico predpogojev, ki morajo biti izpolnjeni za izvedbo izbrane akcije. Če so v trenutnem stanju vsi predpogoji izpolnjeni, smo našli rešitev za izbrani cilj. V tem primeru nadaljujemo postopek v točki 1, kjer izberemo nov cilj. V nasprotnem primeru pa še neizpolnjeni predpogoji predstavljajo cilje, ki jih je potrebno doseči. Z njimi se vrnemo v točko 1. Namesto izvedbe točke 1 z razširjenimi cilji se lahko tudi vrnemo nazaj v točko 2 in se lotimo izvedbe točke 3 z drugo izbrano akcijo. Podobno se lahko vrnemo nazaj tudi v točko 1 in se lotimo izvedbe z drugim ciljem.

Očitno je, da je iskanje zaporedja akcij lahko zelo kompleksen in računsko zahteven problem. Očitno je tudi, da bi lahko z uporabo hevristike, ki bi ocenila, katere cilje v točki 1 ali pa katere akcije v točki 2 je potrebno pregledati najprej, problem precej zmanjšali. Ker je uporaba hevristike odvisna od problemske domene, na tem mestu možnih hevristik ne bomo opisovali. Se pa na to vprašanje vrnemo v poglavju 9.

Jezik STRIPS se je pri nekaterih problemih izkazal kot omejen. Moteča je lahko na primer predpostavka, da se vse akcije izvedejo v trenutku ali pa omejitev na uporabo binarnih vrednosti za opisovanje akcij. To sta zgolj dva izmed razlogov, ki sta pripomogla k razvoju drugih jezikov za opis problema planiranja, ki predstavljajo razširitev jezika STRIPS.

4.2 Razširitev jezika STRIPS

Ena izmed najpopolnejših razširitev jezika STRIPS je jezik PDDL (Planning Domain Definition Language), katerega preprost pregled je podan v [8]. Ta jezik je omogočil opis akcij z numeričnimi vrednostmi, obravnavo trajajočih akcij ... Žal pa je jezik zaradi splošnosti precej kompleksen in tako preveč obsežen za uporabo v okviru tega diplomskega dela. To je razlog, da smo se želene razširitve STRIPS lotili sami. Za vsako izmed identificiranih, za izvedbo diplomskega dela pomembnih pomanjkljivosti, je v nadaljevanju opisana razširitev.

- Opis akcij z numeričnimi vrednostmi

  Razširitev jezika STRIPS, ki omogoča opis akcij z numeričnimi vrednostmi, je podana v [9]. Numerične vrednosti lahko nastopajo v predpogojih

akcije in v spremembah, ki jih akcija povzroči. Za predpogoje vpeljemo opis $R[n..m]$, ki pomeni $R \geq n$ in $R \leq m$. Spremembe pa opišemo z $R := n$, $R := R + n$ in $R := R - n$. Neomejene intervale predstavimo z $[-\infty, n]$ ali z $[n, \infty]$. Pri tem opisu naj samo dodamo, da lahko na tak način predstavimo tudi binarne vrednosti. Interval [1..1] lahko interpretiramo kot resničen, medtem ko interval [0..0] lahko interpretiramo kot neresničen.

- Trajajoče akcije

  Razširitev jezika STRIPS, ki omogoča opis trajajočih akcij, je podan v [10]. Pri vpeljavi trajajočih akcij vsaki akciji A priredimo vrednost $trajanje(A) > 0$. Trajanje lahko tako zavzame vrednosti na intervalu $\mathbb{R}^+$. Za vsako akcijo A, ki se izvede v časovnem intervalu $[t, t + trajanje(A)]$, predpostavljamo:

  - množica predpogojev P akcije A mora veljati ob pričetku izvedbe akcije in ves čas izvedbe akcije;
  - spremembe V in R se zgodijo v poljubnem trenutku med izvajanjem akcije, vendar pa jih lahko uporabimo šele po koncu akcije.

- Popoln opis stanja

  Pri razširitvi jezika STRIPS, ki dopušča uporabo numeričnih vrednosti, je smiselna vpeljava popolnega opisa stanja. Popoln opis stanja določa, da so vrednosti numeričnih spremenljivk stalno definirane. Posledica tega je, da nastopajo v vsakem stanju. V tem primeru je opis akcij s parametroma V in R nesmiseln. Ob vsaki spremembi vrednosti določene spremenljivke se namreč trenutna relacija poruši, nova pa se vzpostavi. Namesto parametra V in R lahko vpeljemo spremembo SP, ki določa, koliko se vsaka izmed numeričnih spremenljivk spremeni ob izvedbi akcije.

4.3 Mehko planiranje

Problemi planiranja, ki smo jih obravnavali do sedaj, predpostavljajo, da je naše vedenje o sistemu, v katerem se plan izvaja, popolno. Vsaka akcija povzroči točno določeno spremembo stanja, ki se v celoti izvrši vedno, ko izvedemo akcijo. V realnem svetu pa ta predpostavka pogosto ne drži. Izvedba določene akcije navadno lahko pripelje do več različnih stanj, od katerih so ena bolj zaželena od drugih. V tem primeru moramo pri planiranju upoštevati

možnosti, da lahko nekateri plani z določeno verjetnostjo pripeljejo do nezaželenih stanj. Tak način planiranja imenujemo mehko planiranje (angl. planning under uncertainty).

Dobro ogrodje za pričetek obravnave takih problemov nam ponuja odločitvena teorija (angl. decision theory). Če poznamo porazdelitveno funkcijo, ki določa verjetnost prehoda v vsakega izmed mogočih stanj, lahko zaporedje akcij predstavimo z odločitvenim drevesom. Listi le-tega predstavljajo vsa mogoča stanja, ki jih z izvedbo zaporedja akcij lahko dosežemo (glej Sliko 4.1). Denimo, da obstaja še kriterijska funkcija, ki vsakemu izmed mogočih končnih stanj priredi t. i. koristnost (angl. utility). Le-ta nam pove, kako zaželeno je določeno končno stanje. V tem primeru je določitev ustreznosti posameznega zaporedja akcij povsem preprosta. Žal pa lahko na tak način zgolj ocenimo ustreznost določenega zaporedja akcij, ne moremo pa hitro najti najbolj ustreznega zaporedja akcij. Pregledovanje vseh mogočih zaporedij je računsko izjemno zahtevno in tako za praktično uporabo nemogoče.

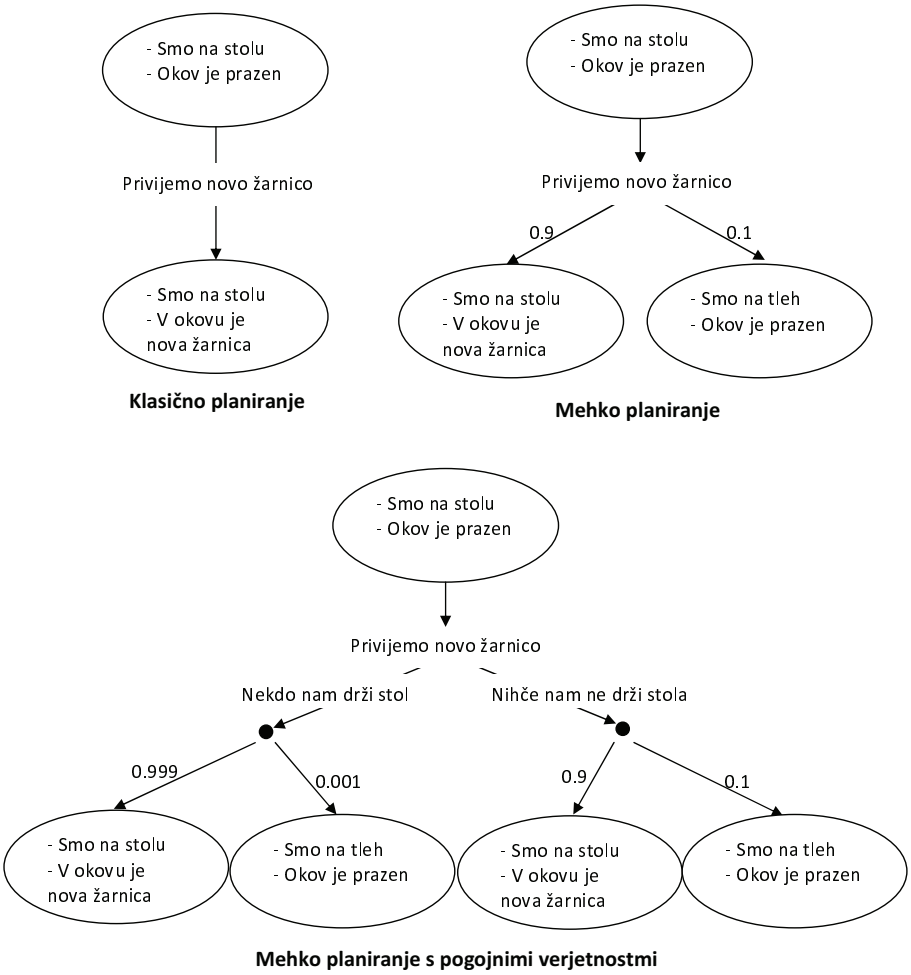
Za učinkovito izvedbo mehkega planiranja je potrebno rešiti naslednje štiri podprobleme:

- Predstavitev akcij

  Pri mehkem planiranju lahko določena akcija povzroči več različnih sprememb. Vsaka izmed sprememb se pri izvedbi akcije zgodi z določeno verjetnostjo. Verjetnost je lahko odvisna tudi od trenutnega stanja (glej tretji primer na Sliki 4.1). Dobra predstavitev akcij je preprosto razumljiva, jedrnata in hkrati popolna (z njo opišemo vse želene akcije). Za primer povejmo, da opis akcije, kot je prikazan na tretjem primeru na Sliki 4.1, ni jedrnat. Kljub temu, da lahko akcija povzroči le dve različni spremembi, je predstavljena s štirimi spremembami, od katerih sta dve in dve enaki.

- Določitev koristnosti stanja

  Za boljšo izvedbo planiranja je potrebno določiti koristnost posameznega stanja. Bolj koristna stanja lahko pregledamo prej, saj bolj verjetno vodijo do dobre rešitve. Najpreprostejša kriterijska funkcija priredi stanju, ki se ujema z želenim stanjem, koristnost 1, vsem ostalim stanjem pa koristnost 0. Za določevanje stanj, ki jih je potrebno pregledati najprej, je neuporabna. Nekoliko jo lahko izboljšamo, če upoštevamo tudi delno ujemanje z želenim stanjem. Denimo, da želeno stanje vsebuje dve relaciji. Če v tem primeru v nekem stanju ena izmed relacij drži, druga pa ne, se takšnemu stanju priredi koristnost 0.5. Žal pa takšna kriterijska funkcija ne odraža vedno smiselne koristnosti. Če smo na primer



Slika 4.1: Prva slika predstavlja opis akcij pri navadnem planiranju. Druga in tretja pa prikazujeta možnosti za opis akcij pri mehkem planiranju. Akcije so lahko predstavljene v obliki odločitvenega drevesa.

pri menjavi žarnice pospravili stol nazaj v kuhinjo, preden smo stopili nanj in zamenjali žarnico, je to slabo. Kljub temu pa bi imelo stanje s stolom v kuhinji večjo koristnost kot stanje s stolom pod žarnico, saj izpolnjuje enega izmed ciljev. Problem bi lahko delno rešili z določitvijo uteži. Njihova določitev je odvisna od problemske domene, zato je na tem mestu ne bomo opisovali.

- Določitev ustreznosti plana

  Ustreznost plana nam pove povprečno doseženo koristnost končnega stanja ob izvedbi plana. V primeru uporabe najpreprostejše kriterijske funkcije (glej prejšnji odstavek) je ustreznost plana kar enaka verjetnosti, da pri izvedbi plana pridemo do stanja s koristnostjo 1. V primeru bolj kompleksne kriterijske funkcije je izračun nekoliko bolj zapleten. Še vedno pa ne presega razmisleka o odločitvenih drevesih, predstavljenega v začetku te točke. Izračun ustreznosti plana se precej zakomplicira, če vsa mogoča končna stanja in njihove koristnosti niso znana. Pri mehkem planiranju je to precej pogost pojav, saj se poskušamo izogniti preiskovanju nepomembnih področij.

- Pogojno planiranje (angl. conditional planning)

  Pri mehkem planiranju stanje po izvedbi akcije ni določeno enolično. Dober plan mora vsebovati ustrezne nadaljnje akcije za vsako izmed mogočih stanj. Nadaljnja izvedba plana je tako odvisna od že izvedenega dela plana. Če smo na primer pri menjavi žarnice padli s stola, so ustrezne prihodnje akcije popolnoma drugačne kot v primeru, ko smo žarnico uspešno zamenjali. Plan mora vsebovati ustrezne akcije za vsako izmed nastalih stanj.

Opis algoritmov, ki rešujejo probleme mehkega planiranja presega okvir tega diplomskega dela. Bralec si lahko preprost opis osnovnih idej pogleda v [11].

Ker je mehko planiranje izjemno kompleksno, nas je zanimalo, če bi ga bilo mogoče uspešno reševati s pomočjo kvalitativnih modelov. Ideja o kvalitativnem planiranju je nova in tako predstavlja največjo dodano vrednost tega diplomskega dela. Splošno je predstavljena v naslednji točki.

4.4 Kvalitativno planiranje

Besedna zveza kvalitativno planiranje ima dvojni pomen:

- lahko opisuje množico vseh načinov planiranja, ki pri izdelavi plana upoštevajo kvalitativne modele ali
- lahko se nanaša konkretno na enega izmed načinov planiranja, ki pri izdelavi plana upošteva kvalitativne modele.

V tem diplomskem delu se besedna zveza vedno uporablja kot določa drugi pomen. Konkreten način planiranja, ki ga imenujemo kvalitativno planiranje, je predstavljen v nadaljevanju.

4.4.1 Opredelitev problema kvalitativnega planiranja

Osnovo kvalitativnega planiranja predstavljata jezik STRIPS in mehko planiranje z razširitvijo, ki omogoča numerične vrednosti akcij. Predpostavlja se tudi popoln opis stanja.

Problem planiranja v okviru STRIPS je predstavljen z začetnim stanjem S, želenim stanjem G in množico akcij A. Tak način predstavitve se ne ukvarja z načinom izvedbe akcij. Pri kvalitativnem planiranju je ločnica med planiranjem in izvedbo precej manj jasna, zato moramo napotke za izvedbo plana vključiti že v sam problem planiranja. Napotke določata vektorja U in V. U določa numerične vrednosti spremenljivk, preko katerih upravljamo robota ob začetku izvajanja plana.V pa določa parametre, ki jih uporabljamo pri pretvorbi med numeričnimi vrednostmi, ki jih uporablja robot, ki izvaja plan, in vrednostmi, ki se uporabljajo pri kvalitativnem planiranju. Vrednosti obeh vektorjev so odvisne od uporabljenega robota in jih je zato potrebno določiti empirično.

Na tem mestu naj samo dodamo, da vektorja U in V ne nasprotujeta splošnosti ideje, predstavljene v nadaljevanju. Pri klasičnem planiranju sta fazi planiranja in izvedbe plana strogo ločeni in se zato pri planiranju z izvedbo ne ukvarjamo. Izvedba pa kljub temu vključuje določene vrednosti, ki so lastne robotu in njihova določitev ni splošna. Pri kvalitativnem planiranju sta planiranje in izvedba združeni, zato se uporabi robotu lastnih vrednosti ne moremo izogniti.

Poleg uvedbe parametrov U in V moramo pri problemu kvalitativnega planiranja definirati še parameter N. Ta predstavlja kvalitativni model, ki velja pri izvedbi plana. V okviru tega diplomskega dela smo uporabili kvalitativni model, določen z algoritmom QUIN, vendar pa kvalitativno planiranje ni omejeno z načinom predstavitve kvalitativnega modela.

Če nekoliko povzamemo, lahko rečemo, da je problem kvalitativnega planiranja definiran kot šestorček $\langle S, G, A, U, V, N\rangle$. Pri tem je vsaka akcija

določena z dvojčkom $\langle P, SP \rangle$. Vsi elementi z izjemo N so sprva določeni numerično. Takšna predstavitev problema planiranja se uporabi kot vhod v kvalitativno planiranje.

Na podlagi dosedanjega razmisleka lahko vidimo, da je uporaba kvalitativnega planiranja smiselna za vse numerično opredeljene probleme mehkega planiranja, za katere lahko izdelamo kvalitativni model in katerih rešitve želimo udejanjiti z uporabo določenega robota.

Reševanje problema kvalitativnega planiranja razdelimo v dve fazi:

1. izdelava kvalitativnega plana;

2. izvedba kvalitativnega plana in uporaba kvalitativnega modela za izboljšavo izvedbe.

Glavna razlika med opisanima fazama je predvsem nivo podrobnosti, ki ga upoštevamo pri izdelavi plana, in čas izvedbe faze. Vsaki fazi se bomo v nadaljevanju podrobneje posvetili. Pri opisu posamezne faze bomo predstavili odgovore na vsakega izmed štirih podproblemov mehkega planiranja.

4.4.2 Izdelava kvalitativnega plana

Osnoven namen te faze je poenostaviti problem planiranja in izdelati plan za poenostavljen problem. Problem mora biti poenostavljen do te mere, da je hiter izračun plana mogoč tudi za kompleksnejše sisteme. Posamezne podprobleme mehkega planiranja smo v tej fazi rešili na naslednji način:

- Predstavitev akcij

  Denimo, da je stanje pri problemu planiranja podano s k numeričnimi spremenljivkami. Zaradi predpostavke o polnem opisu stanja je tudi sprememba, ki jo določena akcija povzroči, opredeljena s k numeričnimi spremenljivkami. Pri kvalitativnem planiranju najprej nad vsako spremembo vsake akcije uporabimo abstrakcijo števil v intervale. Predpogoje akcije pustimo v numerični obliki. Glavni razlog za to odločitev je, da predpogojev zgolj z uporabo kvalitativnih vrednosti ne moremo dovolj natančno izraziti. Na enako ugotovitev lahko sklepamo tudi iz algoritma QUIN, kjer so omejitve, kjer veljajo posamezne kvalitativno omejene funkcije, določene numerično.

  Iz problema mehkega planiranja izhaja, da lahko akcija povzroči več različnih numeričnih in posledično tudi kvalitativnih sprememb. Vse

kvalitativne spremembe lahko opišemo z matriko 4.3. Vsaka vrstica predstavlja eno izmed m kvalitativnih sprememb, ki jih lahko akcija povzroči.

$$SP = \begin{bmatrix} n_{11}n_{12} & ... n_{1k} \\ n_{21}n_{22} & ... n_{2k} \\ \vdots\vdots & \ddots\vdots \\ n_{m1}n_{m2} & ... n_{mk} \end{bmatrix} \tag{4.1}$$

Poleg sprememb, ki jih lahko akcija povzroči poznamo tudi verjetnosti, da se določena sprememba zgodi. Do verjetnosti kvalitativnih sprememb pridemo preprosto z vsoto verjetnosti vseh numeričnih sprememb, ki smo jih abstrahirali v isto kvalitativno spremembo. Verjetnosti posameznih kvalitativnih sprememb lahko predstavimo z vektorjem $w = [w_1, w_2, ..., w_m]$. Če za kvalitativne vrednosti n_{ij} iz matrike 4.3 uporabimo numerično predstavitev, potem lahko izračunamo povprečno kvalitativno spremembo, ki jo določena akcija povzroča. Izračun podaja enačba 4.2.

$$\overline{SP} = w * SP \tag{4.2}$$

Poljubna akcija A je v prvi fazi kvalitativnega planiranja definirana kot dvojček $\langle P, \overline{SP} \rangle$.

Primer: Denimo, da opazujemo spreminjanje temperature in količine toplogrednih plinov v ozračju. Zanima nas primerjava med mesecem majem leta 2000 in mesecem majem leta 2010. V vsakem mesecu smo izvedli pet meritev. Vrednosti leta 2010 so podane relativno glede na vrednosti v letu 2000. Numerične meritve so naslednje (prva vrednost predstavlja spremembo temperature, druga vrednost pa spremembo količine toplogrednih plinov): $SP_1 = [6, 230]$, $SP_2 = [-3, 180]$, $SP_3 = [0, -220]$, $SP_4 = [-1, 30]$ in $SP_5 = [-8, 430]$. Pri pretvorbi numeričnih vrednosti v kvalitativne se odločimo za naslednjo abstrakcijo števil v intervale. Temperaturno spremembo obravnavamo kot pozitivno (negativno), če je večja (manjša) od 1 (-1). Podobno količino toplogrednih plinov obravnavamo kot pozitivno (negativno), če je večja (manjša) od 200 (-200). Vrednost, ki ni niti negativna niti pozitivna je 0. Ustrezno kvalitativno matriko SP v tem primeru prikazuje enačba 4.3. Vektor w je v tem

primeru $[1, 1, 2, 1]$, povprečna sprememba $\overline{SP}$ pa je $[-1/5, 0]$.

$$SP = \begin{bmatrix} 1, & 1 \\ -1, & 0 \\ 0, & 0 \\ -1, & 1 \end{bmatrix} \tag{4.3}$$

- Določitev koristnosti stanja

  Opis stanja je pri kvalitativnem planiranju podan kvalitativno. Iz numeričnega opisa pridemo do kvalitativnega opisa z uporabo abstrakcije števil v intervale. Denimo, da je trenutno stanje podano s kvalitativnim vektorjem $S = [s_1, s_2, ..., s_k]$ in želeno stanje s kvalitativnim vektorjem $G = [g_1, g_2, ..., g_k]$. Koristnost trenutnega stanja izračunamo z enačbo 4.4.

$$c = \sum_{i=1}^{k} |s_i - g_i| \tag{4.4}$$

- Določitev ustreznosti plana

  Kot je bilo že omenjeno, je stanje pri kvalitativnem planiranju določeno kvalitativno. To predstavlja izjemno zmanjšanje kompleksnosti problema planiranja. Žal pa kvalitativna stanja niso primerna za uporabo pri planiranju več kot ene akcije vnaprej. Glavni razlog za to je, da v poljubnem kvalitativno določenem stanju ne moremo vedeti, kateri predpogoji za izvedbo akcij so resnični. Predpogoji so namreč določeni numerično. Planiranje, kjer v vsakem stanju upoštevamo vse mogoče akcije ne glede na njihove predpogoje, pa je nesmiselno. Omenjena omejitev se pri uporabi kvalitativnega planiranja na realnem primeru ni izkazala kot moteča.

  Na podlagi ugotovitve v prejšnjem odstavku se določitev ustreznosti plana prevede v določitev ustreznosti akcije. Ustreznost posamezne akcije izračunamo z enačbo 4.5. Vektor kvalitativnih vrednosti D predstavlja želene spremembe in je določen z razliko želenega in trenutnega stanja ($D = G - S$). d_i predstavlja element na i-tem mestu kvalitativnega vektorja D. Podobno sp_i predstavlja element na i-tem mestu vektorja $\overline{SP}$, ki določa povprečno kvalitativno spremembo, ki jo opazovana akcija povzroči. w_i predstavlja pomembnost posamezne spremenljivke v trenutnem stanju. V splošnem primeru lahko vsem spremenljivkam priredimo

pomembnost 1. Planiranje lahko izboljšamo, če priredimo višjo pomembnost spremenljivkam, katerih numerična napaka med želenim in trenutnim stanjem je večja.

$$u = \sum_{i=1}^{k} w_i * |sp_i - d_i| \tag{4.5}$$

- Pogojno planiranje

  Zaradi poenostavitve problema planiranja se v fazi določitve kvalitativnega plana s pogojnim planiranjem ne ukvarjamo. Izvedbo pogojnega planiranja prepustimo drugi fazi.

Rezultat prve faze kvalitativnega planiranja je najustreznejša akcija A_Z. Postopek določitve je naslednji:

1. določimo množico A_K kvalitativnih akcij, katerih predpogoji ustrezajo trenutnemu numerično določenemu stanju (ta korak se izvede pred pretvorbo v kvalitativno predstavitev stanja, saj se ob pretvorbi ta informacija izgubi);
2. za vsako akcijo iz množice A_K se z enačbo 4.5 izračuna ustreznost;
3. akcija z največjo ustreznostjo predstavlja najustreznejšo akcijo A_Z in tako rezultat prve faze kvalitativnega planiranja.

4.4.3 Izvedba kvalitativnega plana in uporaba kvalitativnega modela za izboljšavo izvedbe

Rezultat prve faze kvalitativnega planiranja je najustreznejša akcija A_Z. V drugi fazi akcijo A_Z izvedemo. Izvajanje, s pomočjo določenih podakcij, stalno prilagajamo trenutnim razmeram.

- Predstavitev akcij

  Akcije, ki jih uporabljamo v drugi fazi planiranja, imenujemo podakcije. Podrobno so predstavljene v točki Določitev in ustreznost plana.

- Določitev koristnosti stanja

  Denimo, da je trenutno stanje podano z vektorjem $S = [s_1, s_2, ..., s_k]$ in želeno stanje z vektorjem $G = [g_1, g_2, ..., g_k]$. Vrednosti, ki določajo stanje, so numerične. Kriterijska funkcija, ki določa koristnost stanja, je

podana kot $f(S, G) \rightarrow x$, kjer je $x \in \mathbb{R}$. Določitev konkretne kriterijske funkcije je odvisna od problemske domene. Če na primer stanje vsebuje zgolj spremenljivke, ki predstavljajo prostorske koordinate, lahko za kriterijsko funkcijo uporabimo Evklidsko razdaljo.

Določitev koristnosti stanja je pri drugi fazi kvalitativnega planiranja zelo pomembna. Akcija A_Z se namreč izvaja, dokler se koristnost stanja izboljšuje (včasih je smiselno dopustiti tudi manjše poslabšanje koristnosti stanja), potem pa se izvajanje prekine.

- Določitev ustreznosti plana

  Ustreznosti plana je v drugi fazi kvalitativnega planiranja enaka ustreznosti podakcije. Množica podakcij je določena z robotom, ki ga uporabljamo za izvedbo plana. Spremenljivke s katerimi upravljamo robota, imenujemo neodvisne spremenljivke. Vse ostale spremenljivke so odvisne spremenljivke. Postopek izračuna ustreznosti posamezne podakcije je opisan v naslednjih točkah. Na koncu opišemo tudi določitev najustreznejše podakcije.

  1. Določimo množico M mogočih kvalitativnih sprememb neodvisnih spremenljivk. Vsako izmed spremenljivk lahko povečamo, zmanjšamo ali pa ne spremenimo. Posamezno spremembo neodvisnih spremenljivk iz množice M označimo z i.
  2. Vsaki izmed odvisnih spremenljivk j določimo pomembnost p_j in želeno kvalitativno spremembo z_j. Pri določitvi si lahko pomagamo s kriterijsko funkcijo za določevanje koristnosti stanja. Spremenljivkam, katerih razlika do želenega stanja je velika, je smiselno prirediti veliko pomembnost. Problem določitve pomembnosti in želene kvalitativne spremembe je odvisen od problemske domene. Za boljše razumevanje si ga bomo pogledali na preprostem primeru. Zahtevnejši primer je predstavljen v poglavju 8.

     Primer: Denimo, da želimo z motornim čolnom potovati z otoka, na katerem se trenutno nahajamo, do sosednjega otoka. Na podlagi prve faze kvalitativnega planiranja smo že določili ustrezno smer plovbe, zato lahko s plovbo kar začnemo. Do sosednjega otoka bi radi prišli hitro, vendar moramo paziti, da nam pri tem ne zmanjka goriva. Prevožena pot in količina goriva predstavljata odvisni količini, hitrost pa je neodvisna količina. Želene vrednosti so preprosto določljive. Za prevoženo pot želimo, da se povečuje kar najhitreje (odvod je čim večji). Za porabo goriva želimo, da je čim

manjša. Pomembnost pa lahko določimo z naslednjim razmislekom. Pomembnost odvoda prevožene poti je ves čas poti konstantna, če predpostavimo, da se naša želja, da pot končamo hitro, tekom poti ne spreminja. Pomembnost porabe goriva pa je odvisna od količine goriva v rezervoarju. Dokler je goriva veliko, poraba ni posebej pomembna. Če pa je goriva malo, je poraba izjemnega pomena.

3. Za vsako mogočo spremembo neodvisnih spremenljivk m_i iz množice M s pomočjo kvalitativnega modela N določimo kvalitativno spremembo odvisnih spremenljivk o_i. Spremembe primerjamo z želenimi kvalitativnimi spremembami z in z uporabo enačbe 8.4 določimo parameter l_{ij}. Parameter določa skladnost med želenimi spremembami odvisnih spremenljivk in spremembami odvisnih spremenljivk, ki jih povzroči sprememba neodvisnih spremenljivk m_i.

$$l_{ij} \quad = \quad \begin{cases} |o_{ij} - z_j|; & sprememba \quad o_{ij} \quad je \quad določena \\ 0.5; & sprememba \quad o_{ij} \quad ni \quad določena \end{cases} \tag{4.6}$$

4. Za vsako spremembo neodvisnih spremenljivk m_i iz množice M z enačbo 4.7 izračunamo uteženo vsoto d_i.

$$d_i = \sum_j p_j * l_{ij} \tag{4.7}$$

5. Izberemo spremembo m_{min} z najnižjo vrednostjo d_i, saj predstavlja najustreznejšo kvalitativno spremembo. Le-ta bo naslednjih Δt sekund določala izvajanje.

6. Robota, ki izvaja plan, lahko nadzorujemo preko neodvisnih spremenljivk, ki so določene numerično. Tako moramo določiti vpliv izbrane kvalitativne spremembe na numerično vrednost neodvisnih spremenljivk. Preden lahko to storimo, moramo definirati pomembnost posamezne neodvisne spremenljivke. Pomembnost p-te neodvisne spremenljivke označimo s t_p. Izračunamo jo kot razliko med parametrom d_i, ki je izračunan na podlagi kvalitativne spremembe m_{min}, ki ima spremembo p-te neodvisne spremenljivke 0 in parametrom d_i, izračunanem na podlagi nespremenjene kvalitativne spremembe m_{min}. Za določitev numeričnih vrednosti neodvisnih spremenljivk pa potrebujemo tudi vektorja U in V, ki smo ju že opredelili v problemu planiranja. Z u_p in v_p označimo vrednost p-te neodvisne spremenljivke. Podobno z m_{min_p} označimo spremembo

p-te neodvisne spremenljivke v m_{min}. Z r_p označimo numerično vrednost p-te neodvisne spremenljivke, ki se bo izvajala naslednjih Δt sekund. Izračunamo jo z enačbo 4.8.

$$r_p = u_p + m_{min_p} * t_p * v_p \tag{4.8}$$

- Pogojno planiranje

  Celotno izvedbo druge faze kvalitativnega planiranja bi lahko uvrstili med pogojno planiranje, saj je izbira ustrezne podakcije odvisna od trenutnega stanja.

Kvalitativno planiranje v drugi fazi poteka na naslednji način:

1. začnemo z izvedbo akcije A_Z, ki je rezultat prve faze planiranja. Izvajanje pričnemo tako kot ga določa vektor U;
2. vsakih Δt sekund izvedbe plana preverimo:
   (a) spremembo koristnosti stanja.
      i. če je koristnost stanja v okviru želene končne koristnosti izvedbo prekinemo, saj se je naloga uspešno izvedla;
      ii. drugače:
         A. če se je koristnost stanja izboljšala, potem z izvedbo nadaljujemo;
         B. če se je koristnost stanja poslabšala, izvedbo prekinemo. V tem primeru ponovno pričnemo s prvo fazo kvalitativnega planiranja;
   (b) najustreznejšo podakcijo, ki jo določimo na podlagi opisa v Določitev ustreznosti plana. Podakcijo izvajamo Δt sekund, potem pa izvedbo kvalitativnega planiranja nadaljujemo na začetku točke 2.

Poglavje 5

Implementacija

To poglavje predstavlja uvod v predstavitev uporabe že predstavljenih algoritmov in metod na konkretnem primeru. Poglavje se prične z opisom problemske domene in nadaljuje z globalnim pregledom posameznih rešitev. Vsaki izmed njih se v naslednjih poglavjih tudi podrobno posvetimo.

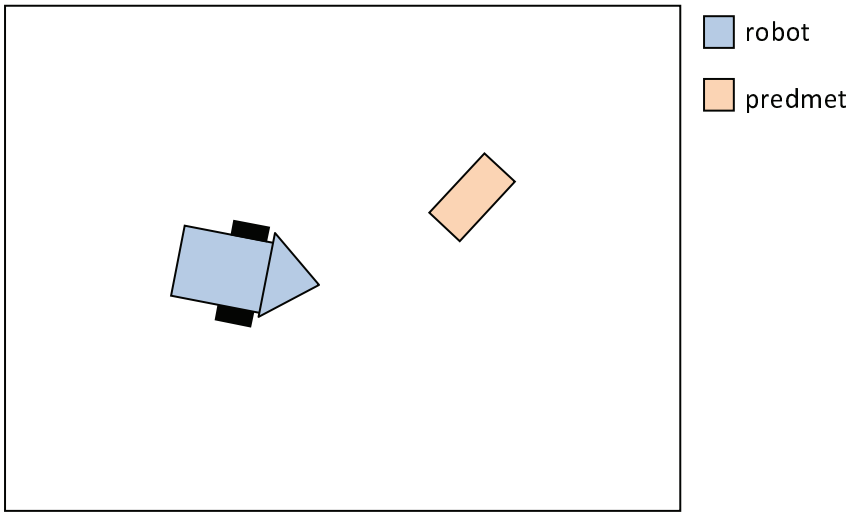
5.1 Opredelitev problema

Podan imamo pravokoten poligon Z, kot ga prikazuje Slika 5.1. Znotraj poligona se nahajata robot in predmet.

Robot je izdelan iz gradnikov Lego Mindstorms [24]. Njegov izgled prikazuje Slika 5.2. Sprednji del robota predstavlja odbijač, ki je namenjen potiskanju predmeta. Robota lahko upravljamo s spreminjanjem hitrosti levega in desnega kolesa, ki jo lahko nadziramo preko Bluetooth povezave. Lokacijo in orientacijo robota ter predmeta zaznavamo s pomočjo kamere, ki se nahaja nad poligonom Z. Njen zorni kot je skladen s Sliko 5.1. Za avtomatsko določitev lokacije in orientacije robota ter predmeta iz zajete slike smo uporabili knjižnico, ki je podrobno opisana v [25].

5.1.1 Predstavitev stanja in gibanja robota

Pri upravljanju robota je prvi problem, ki ga moramo rešiti, problem razumevanja gibanja. Robota upravljamo zgolj preko hitrosti levega v_L in hitrosti desnega v_D kolesa, zato je potrebno preučiti njihovo povezavo s spremembami lokacije in orientacije robota. Za lažje razumevanje in manipulacijo enačb na začetku vpeljemo dve novi količini:



Slika 5.1: Prikaz tlorisa poligona Z z robotom in predmetom za učenje in premikanje.

- hitrost težišča robota v, ki jo določa enačba 5.1

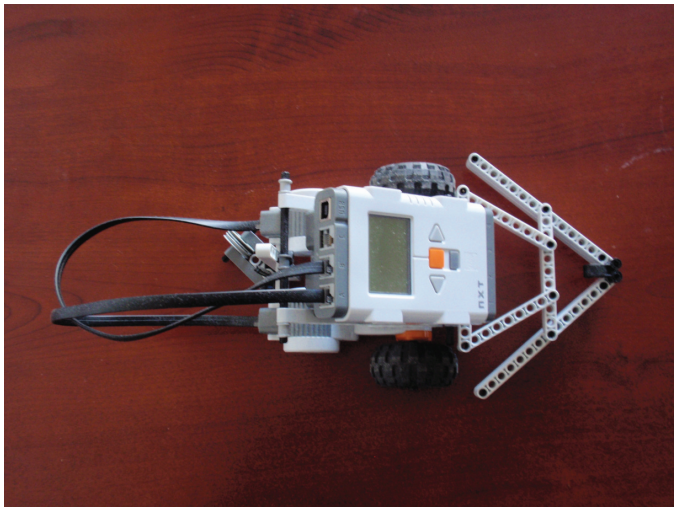
$$v(t) = \frac{v_L(t) + v_D(t)}{2} \tag{5.1}$$

- kotno hitrost robota ω, ki jo določa enačba 5.2. Količina d predstavlja medosno razdaljo.

$$\omega(t) = \frac{v_L(t) - v_D(t)}{d} \tag{5.2}$$

Z upoštevanjem podanih enačb lahko stanje robota v času t opišemo z vektorjem $q(t) = (x_r(t), y_r(t), \theta_r(t))$, gibanje pa z vektorjem $u(t) = (v(t), \omega(t))$. Vrednosti (x_r, y_r) predstavljata lokacijo težišča robota, θ_r pa je kot med osjo y in orientacijo robota (glej Sliko 5.3).

Ob tako definiranem stanju in vektorju gibanja robota,se pojavi vprašanje, kakšna je sprememba stanja, ki jo povzroči določen vektor gibanja, če obstaja določen čas Δt. Na podlagi Slike 5.4, lahko identificiramo enačbi 5.3 in 5.4, kjer R predstavlja velikost polmera navidezne krožnice, po kateri se giblje robot pri določenem vektorju gibanja. $\Delta x'_r$ in $\Delta y'_r$ pa predstavljata spremembi koordinat robota v koordinatnem sistemu, v katerem je $\theta_r = 0$.



Slika 5.2: Robot, uporabljen pri izdelavi diplomskega dela.

$$\Delta y'_r = \sin \Delta\theta_r * R \tag{5.3}$$

$$\Delta x'_r = \cos \Delta\theta_r * (1 - R) \tag{5.4}$$

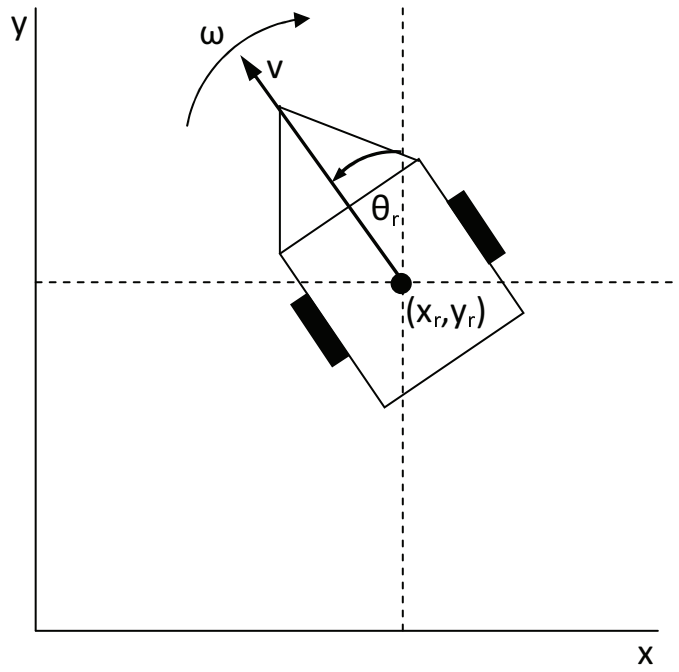
Za izračun sprememb Δx_r in Δy_r je potrebno uporabiti enačbi 5.5 za vrtež koordinatnega sistema, ki so vzete iz [14]. Ker se bomo na enačbi za vrtež koordinatnega sistema sklicevali večkrat, sta navedeni splošno. Osnovni koordinatni sistem je xy, zavrten koordinatni sistem pa $x'y'$. Prilagoditev konkretnemu primeru je trivialna.

$$\begin{aligned} x &= x' * cos\theta - y' * sin\theta \\ y &= x' * sin\theta + y' * cos\theta \end{aligned} \tag{5.5}$$

Za popoln opis modela moramo dodati še enačbi 5.6 in 5.7, ki splošno veljata za enakomerno kroženje. V njuno razlago se tako ne bomo spuščali, saj predpostavljamo, da je bralec s tem področjem seznanjen. V nasprotnem primeru si lahko podrobnosti prebere v [15].

$$\Delta\theta_r = \omega * \Delta t \tag{5.6}$$

$$v = \omega * R \tag{5.7}$$



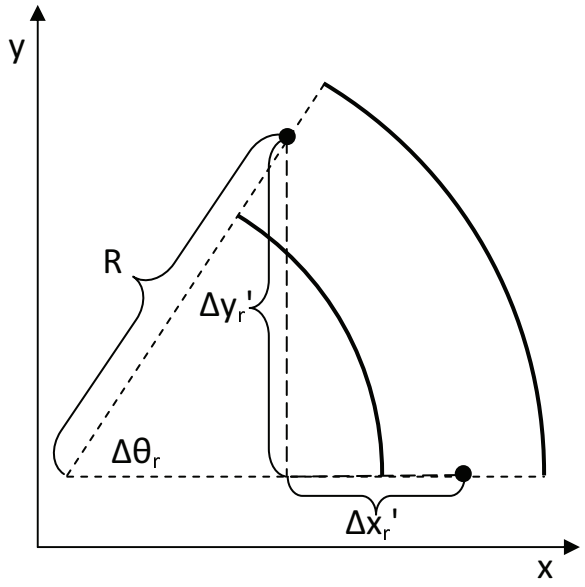
Slika 5.3: Prikaz količin, ki določajo stanje in gibanje robota. Puščice prikazujejo smer naraščanja izbranih količin.

Tako podan model gibanja je za naše potrebe popoln, saj omogoča napovedovanje prihodnjega stanja robota, če poznamo trenutno stanje in vektor gibanja. Podobno omogoča tudi izračun ustreznega vektorja gibanja ob podanem trenutnem in želenem stanju.

5.1.2 Predstavitev stanja predmeta

Predmet predstavlja element poligona Z, ki je namenjen potiskanju. Robot lahko s potiskanjem spreminja stanje predmeta. V času je stanje predmeta določeno na dva načina:

- kot poligon $O(t) = (T_1(t), T_2(t), ..., T_n(t))$, kjer posamezne točke predstavljajo oglišča predmeta (glej Sliko 5.5) in
- kot vektor $o(t) = (x_o(t), y_o(t), \theta_o(t))$, kjer vrednosti (x_o, y_o) predstavljata težišče predmeta, θ_o pa je kot med osjo y in orientacijo predmeta (glej Sliko 5.5).



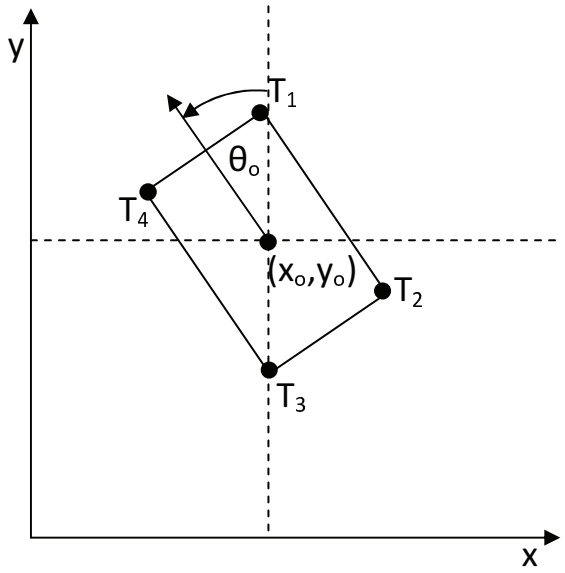
Slika 5.4: Model gibanja robota, kjer odebeljeni črti predstavljata odtis vsakega izmed koles v zavoju.

Dvojni način določanja stanja je nujen. Prvi način omogoča izračun oddaljenosti med robotom in predmetom, ki ga potrebujemo za zaznavanje trkov. Drugi način pa določa stanje predmeta, kot ga dobimo z uporabo knjižnice za analizo slike. Zaradi želje po enolični preslikavi poligona O v stanje o in obratno, je vsak predmet s strani uporabnika podan s poligonom O, ki ustreza stanju $o = (0, 0, 0)$. Na ta način lahko med predstavitvama preprosto pretvarjamo.

5.2 Opredelitev glavnega problema in podproblemov

Ob definiranih posameznih elementih poligona Z lahko opredelimo tudi glavni problem tega diplomskega dela. To je sprememba trenutnega stanja predmeta v želeno stanje zgolj z robotovim potiskanjem.

Zaradi lažje obvladljivosti je dobro, če glavni problem razdelimo na več podproblemov, ki jih lahko obravnavamo kar najbolj ločeno. Smiselna razdelitev je naslednja:



Slika 5.5: Prikaz količin, ki določajo stanje predmeta.

1. Upravljanje gibanja

   Rešitev tega podproblema mora biti knjižnica, ki bo omogočala manipulacijo z robotom.

2. Učenje modela potiskanja predmetov

   Rešitev tega podproblema mora biti knjižnica, ki izdela kvalitativni model potiskanja za poljuben predmet znotraj poligona Z. Hkrati mora ponujati tudi numerične napovedi, ki so skladne s kvalitativnim modelom.

3. Planiranje in izvedba plana

   Rešitev tega podproblema mora biti knjižnica, ki bo z uporabo kvalitativnega modela in numeričnih napovedi, določila zaporedje akcij, ki jih mora izvesti robot, da bo premaknil predmet iz trenutnega stanja v želeno stanje. Robota mora znati tudi usmerjati tako, da bo izvedel predviden plan. V primeru neuspešnega izvajanja plana mora to tudi ugotoviti in zahtevati ponovno planiranje.

V nadaljevanju se bomo natančno posvetili vsakemu izmed identificiranih podproblemov.

Poglavje 6

Upravljanje gibanja

V tem poglavju so najprej opisani problemi, ki jih je bilo potrebno rešiti za učinkovito upravljanje robota. Sledi razlaga rešitev posameznih problemov.

6.1 Problem navidezne poti

Pri problemu navidezne poti imamo v ravnini podano pot P, ki je določena kot zaporedje točk $P = P_1, P_2, ..., P_n$. Robot se nahaja v točki P_1, njegova naloga pa je, da pride do točke P_n tako, da obišče vse vmesne točke v naraščajočem vrstnem redu. Pri tem se ne sme od navidezne poti, ki je določena z linearnimi povezavami med točkami na poti P oddaljiti za več kot določeno razdaljo $dist$.

Celoten problem je sestavljen iz dveh med seboj povezanih podproblemov:

1. Kako povezati zaporedje točk P, da bo sledenje pridobljeni navidezni poti najlažje?
2. Kako kar najbolj točno slediti navidezni poti, pridobljeni v prejšnji točki?

V nadaljevanju si bomo pogledali vsakega izmed podproblemov.

6.1.1 Način povezave točk navidezne poti

Glede reševanja prvega podproblema obstaja kar nekaj rešitev. V matematične podrobnosti vsake izmed njih se bomo spuščali, saj so dostopne v [16]. Tukaj navajamo zgolj nekaj različnih možnosti:

- Uporaba numerične interpolacije:
  - Linearna interpolacija

  - Kvadratna, kubna in še višje interpolacije
  - Lagrangeova formula
  - Newtonova formula
  - Zlepki polinomov

- Uporaba Bezierova krivulj ali njihovih zlepkov (podrobnosti v [17])

6.1.2 Sledenje navidezni poti

V literaturi se za sledenje navidezni poti pojavlja precej različnih pristopov. V okviru tega diplomskega dela smo se osredotočili zgolj na dva izmed njih, ker sta pri podobnih problemih pogosto uporabljena in dajeta dobre rezultate. Hkrati pa sta tudi precej preprosta za implementacijo. To sta:

- Pure Pursuit;
- PID (proportional-integral-derivate) kontroler.

V nadaljevanju si bomo vsakega izmed njih nekoliko bolj podrobno pogledali.

Pure Pursuit

Metoda spada med starejše metode na področju reševanja problema sledenja navidezne poti. Ker je preprosta za implementacijo, kalibracijo in kljub temu daje dobre rezultate, je še vedno precej razširjena [18]. Osnovna ideja je sestavljena iz treh korakov:

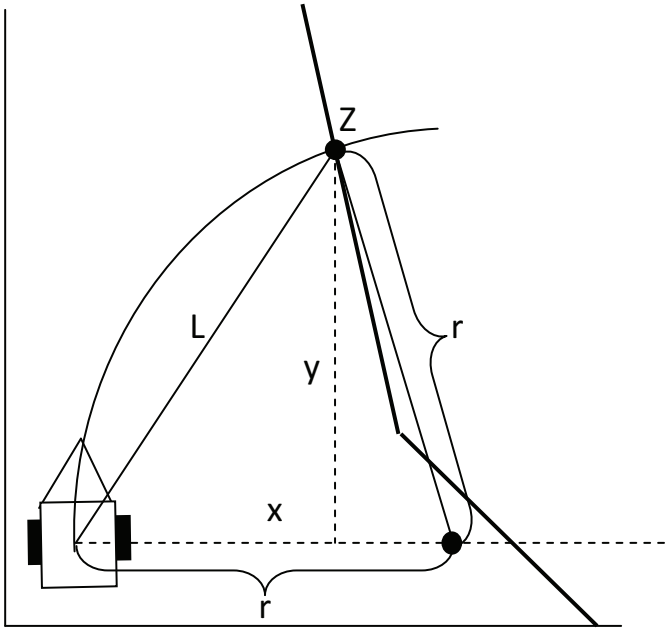
1. določiti želeno točko Z na navidezni poti, ki je od trenutne lokacije robota oddaljena določeno razdaljo L.

   Določevanje lahko prevedemo na izračun presečišča med premico, ki je določena s točkami navidezne poti in krožnico s polmerom L in središčem v trenutni lokaciji robota;

2. določiti polmer kroga, ki povezuje trenutno lokacijo robota in želeno točko Z.

   Z upoštevanjem povezav med količinami na Sliki 6.1, izpeljemo enačbo 6.1. Ta določa polmer kroga r, ki povezuje trenutno lokacijo robota in želeno točko Z.

$$r = \frac{L^2}{2x} \tag{6.1}$$



Slika 6.1: Prikaz količin, ki nastopajo pri metodi Pure Pursuit. Odebeljeno je prikazana navidezna pot.

Izpeljava je preprosta in dostopna v [18], zato jo bomo na tem mestu izpustil. Tukaj naj samo dodamo, da slika 6.1 in iz nje izpeljane enačbe predpostavljajo kot $\theta_r = 0°$. Za drugačne vrednosti kota θ_r moramo uporabiti enačbo 5.5 za vrtež koordinatnega sistema;

3. izračunati vektor gibanja robota, ki bo omogočal sledenje polmeru kroga.

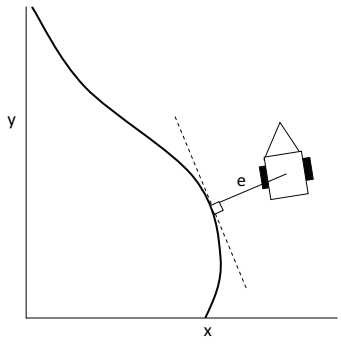
   Izračun vektorja gibanja iz polmera r je preprost, saj je edina zveza, ki jo moramo upoštevati, enačba 5.7. Kot že omenjeno povezava predstavlja eno izmed osnovnih zakonitosti enakomernega kroženja. Ker imamo zgolj eno enačbo in dve neznanki, je ena izmed neznank lahko določena poljubno. V okviru tega diplomskega dela je to vedno hitrost težišča robota v, tej izbiri pa se potem prilagodi izračun kotne hitrosti ω;

4. pričeti z izvajanjem vektorja gibanja iz točke 3.

Postopek se ponovi ob vsakem zajemu kamere. Ker je razdalja L konstantna, to pomeni, da želena točka Z dokaj gladko drsi pred robotom po navidezni poti.

PID kontroler

To je ena izmed trenutno najbolj aktualnih metod za reševanje problema sledenja navidezni poti [19]. Ta pristop je sestavljen iz treh delov (Proportional, Integral, Dervate), od katerih ima vsak določen prispevek k izboljšanju sledenja navidezni poti. Osnovna ideja je, da se ob vsakem zajetju kamere ugotovi trenutno lokacijo robota in izračuna njeno oddaljenost e od idealne točke na navidezni poti (glej Sliko 6.2).



Slika 6.2: Napaka, ki jo pri izračunu upošteva PID kontroler.

Na podlagi napake e se potem določi, kako velika bo kotna hitrost robota ω v naslednjem trenutku gibanja. Povsem očitno je, da večja napaka pomeni večjo kotno hitrost. To načelo upošteva proporcionalni del PID kontrolerja, ki je določen z enačbo 6.2.

$$P_{izhod} = K_p * e(t) \tag{6.2}$$

P_{izhod} predstavlja prispevek proporcionalnega dela k celotnemu izhodu. K_p predstavlja utež, ki določa pomembnost proporcionalnega dela. Spremenljivka e je napaka, ki je prikazana na sliki 6.2; t predstavlja trenutni čas.

Tak način uravnavanja gibanja je mogoče še izboljšati. Zgolj z uporabo proporcionalnega dela se robot na napako odziva počasi, hkrati pa ne izkorišča podatkov o zgodovini. To lahko odpravi integralni del, ki je odvisen od napak v preteklosti, saj upošteva tako velikost napake kot tudi njeno trajanje. Integralni del opisuje enačba 6.3.

$$I_{izhod} = K_i \int_0^t e(\tau)\mathrm{d}\tau \tag{6.3}$$

I_{izhod} predstavlja prispevek integralnega dela k celotnemu izhodu. K_i predstavlja utež, ki določa pomembnost integralnega dela. Spremenljivka e je napaka, ki je prikazana na Sliki 6.2; t je trenutni čas in τ integracijska spremenljivka.

Zgolj z uporabo proporcionalnega in integralnega dela se robot na idealni poti nikoli ne ustali, ampak zgolj niha okrog idealne lege. Težavo lahko znatno izboljša upoštevanje odvoda napake. Ta poskrbi, da se idealni točki ne približujemo prehitro, saj bi to pomenilo, da se bo ob prečkanju idealne točke napaka začela hitro povečevati. Del z upoštevanjem odvoda je podan z enačbo 6.4.

$$D_{izhod} = K_d \frac{\mathrm{d}}{\mathrm{d}t} e(t) \tag{6.4}$$

D_{izhod} predstavlja prispevek dela z odvodom k celotnemu izhodu. K_d predstavlja utež, ki določa pomembnost dela z odvodom. Spremenljivka e je napaka, ki je prikazana na Sliki 6.2; t predstavlja trenutni čas

Vsi deli se nato seštejejo v skupen izhod, kot določa enačba 6.5.

$$izhod(t) = P_{izhod} + I_{izhod} + D_{izhod} \tag{6.5}$$

Predstavljen opis velja za zvezne spremenljivke, čas v primeru zajemanja s kamero pa je diskreten. Enačbo 6.5 moramo preurediti v diskretno obliko in jo prilagodi našemu problemu kotne hitrosti. Rezultat je enačba 6.6.

$$\omega(t_k) = K_p * e(t_k) + K_i * \sum_{i=1}^{k} (e(t_i) * \Delta t) + K_d \frac{e(t_k) - e(t_{k-1})}{\Delta t} \tag{6.6}$$

Zadnji korak pri uporabi PID kontrolerja je določitev konstant K_p, K_i in K_d. V literaturi se pojavlja več metod za njihovo določitev, ki so nekoliko bolj natančno opisane v [20]. Pri izdelavi tega diplomskega dela smo uporabili ročno določitev konstant.

Kombinacija PID kotrolerja in Pure Pursuit pristopa

Namesto uporabe zgolj PID kontrolerja ali Pure Pursuit pristopa so se pojavili tudi poskusi združitve obeh pristopov. Predlagana združitev je preprosta, saj je določena kar z aritmetično sredino rezultatov vsakega izmed pristopov. To podaja enačba 6.7.

$$\omega = \frac{\omega_{PID} + \omega_{PurePursuit}}{2} \tag{6.7}$$

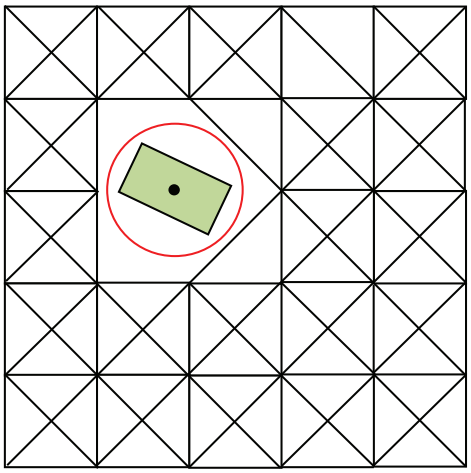
Rezultati pri uporabi kombiniranega pristopa so boljši kot pri uporabi zgolj enega izmed pristopov [21].

6.2 Premikanje robota

Osnovna naloga premikanja robota je spremeniti trenutno stanje robota v želeno stanje zgolj z upravljanjem hitrosti levega v_L in hitrosti desnega v_D kolesa. Pri opravljanju naloge premikanja robot ne sme spremeniti stanja predmeta. Izvedba poteka v več korakih:

1. Izdelava neusmerjenega grafa G

   Iz celotne notranjosti poligona Z se ustvari neusmerjen graf $G = \langle V, E \rangle$. Vozlišča V grafa so vse koordinate v notranjosti poligona Z, ki so deljive z 10 (npr. (0, 0), (10, 0), (50, 60)) in niso preblizu predmeta (glej rdeče območje na Sliki 6.3). Povezave E grafa so vse mogoče povezave med sosednjimi vozlišči v vertikalni, horizontalni in diagonalni smeri (glej Sliko 6.3). Cene diagonalnih povezav so 14, vertikalnih in horizontalnih pa 10. Cene povezav tako približno ustrezajo dejanski razdalji med vozlišči.



Slika 6.3: Graf G, ki predstavlja vse mogoče poti v notranjosti poligona Z.

2. Iskanje najkrajše poti v neusmerjenem grafu G

   Vrednosti x_r in y_r trenutnega stanja robota se zaokrožita na vrednost najbližjega vozlišča v grafu G. Enako se naredi tudi z želeno končno

točko. Med tako pridobljenima vozliščema se z uporabo grafa G in A* algoritma izračuna najkrajšo pot. Pri uporabi A* algoritma smo kot hevrsitiko uporabili Evklidsko razdaljo. Na tem mestu predpostavljamo, da je bralec seznanjen s problemom najkrajše poti in algoritmom A*. V nasprotnem primeru si lahko splošen opis problema najkrajše poti pogleda v [22], opis algoritma A* pa v [23].

3. Izbris nepotrebnih vozlišč

   Začetno vozlišče najkrajše poti se nadomesti z vrednostma x_r in y_r trenutnega stanja robota. Podobno se končno vozlišče najkrajše poti nadomesti z želeno končno lokacijo. Nato se izbrišejo vsa nepotrebna vmesna vozlišča. Vozlišče je nepotrebno, če povezava, ki povezuje neposrednega predhodnika in neposrednega naslednika vozlišča na svoji poti ni nikoli preblizu predmeta (glej rdeče območje na Sliki 6.3).

4. Premik robota po navidezni poti, pridobljeni v točki 3

   Ta korak predstavlja reševanje problema sledenja navidezni poti, zato smo se poslužili pristopov, opisanih v začetku tega poglavja. Implementirali smo naslednje:

   (a) Način povezave točk navidezne poti
   - Linearna interpolacija

   (b) Način sledenja navidezni poti
   - PID kontroler
   - Kombinacija PID kontrolerja in Pure Pursuit pristopa

   Pri implementaciji se izkaže, da ostri robovi sledenju z uporabo PID kontrolerja povzročajo velike težave, saj robot večino ostrih robov ne izpelje pravilno. Kombinacija PID kontrolerja in Pure Pursuit pristopa rezultate nekoliko izboljša, saj robot več zavojev izpelje pravilno, vendar pa se v ostrih zavojih od navidezne poti preveč oddalji. Ta problem lahko zmanjšamo z uporabo manjše vrednosti L pri Pure Pursuit pristopu, vendar pa to na drugi strani poveča verjetnost neizpeljanega zavoja. Problem lahko povsem rešimo, če hitrost težišča robota v ostrih zavojih navidezne poti zmanjšamo. V skrajnem primeru se robot v vozliščih povsem ustavi, zavrti in šele potem nadaljuje pot. To omogoča izjemno dobro sledenje navidezni poti. Tudi vsi zavoji so odpeljani pravilno. Cena za dobro zanesljivost je nekoliko počasnejša izvedba. Zmanjševanje hitrosti v zavojih se izkaže kot ustrezno za oba načina sledenja navidezni poti.

Pri izbiri najboljšega načina smo v prvi vrsti upoštevali zanesljivost, v drugi vrsti pa še preprostost in čas izvedbe. Zaradi zanesljivosti je zmanjševanje hitrosti nujno. Čas izvedbe se pri implementiranih načinih praktično ne razlikuje. Zaradi preprostosti kot najprimernejši način sledenja navidezni poti izberemo PID kontroler.

6.3 Zaznavanje trkov med robotom in predmetom

Natančno zaznavanje trka med robotom in predmetom je nujno potrebno za natančno potiskanje predmeta. Pristopa, ki ju lahko uporabimo za izvedbo te naloge sta dva:

- Ugotavljanje trka na podlagi razdalje med robotom in predmetom

  Odbijač robota lahko opišemo s trikotnikom, ki ga predstavimo kot poligon P. Točke, ki določajo poligon P, lahko izračunamo iz informacije o stanju robota in razdalje med težiščem robota in konico odbijača. Stanje predmeta je predstavljeno s poligonom O. Za izračun razdalje med dvema poligonoma (P in O) lahko uporabimo več postopkov. Večina njih je preprosta, zato podrobnosti ne bomo opisovali. Eden izmed postopkov je predstavljen v [26]. Če je na tak način določena razdalja manjša od nekaj milimetrov, lahko to obravnavamo kot trk robotovega odbijača in predmeta.

- Ugotavljanje dotika na podlagi spremembe stanja predmeta

  Pri tem pristopu ob vsakem zajemu kamere primerjamo trenutno stanje predmeta s stanjem predmeta pri predhodnem zajemu. Če se je stanje predmeta spremenilo, lahko sklepamo na dotik robota in predmeta. Robot je namreč edini, ki lahko spreminja stanje predmeta.

Prvi izmed opisanih pristopov je nenatančen, saj morajo biti vse razdalje s strani uporabnika določene do milimetra natančno. Tudi kalibracija kamere mora biti izjemna. To je razlog, da smo uporabili drugega izmed opisanih pristopov, ki je za uporabo nekoliko bolj preprost. Zaznavanje trka je v tem primeru natančnejše.

Poglavje 7

Učenje modela potiskanja predmetov

Model potiskanja predmetov mora identificirati odvisnosti, ki veljajo med posameznimi, za reševanje problema pomembnimi, spremenljivkami v sistemu robot - predmet. Tem odvisnostim se bomo v poglavju 7 bolj natančno posvetili. Opis učenja modela pričnemo z identifikacijo pomembnih spremenljivk. V naslednjem koraku se posvetimo strategiji zbiranja učnih primerov, ki bodo pomagali pri določitvi odvisnosti. Na koncu predstavimo tudi rezultate učenja, ki jih razdelimo na kvalitativne in kvantitativne.

7.1 Opredelitev modela potiskanja predmetov

Preden začnemo z učenjem odvisnosti med spremenljivkami, moramo identificirati, katere so pomembne odvisne in neodvisne spremenljivke pri modelu potiskanja predmetov. Vemo, da imamo moč nadzorovati gibanje robota in da lahko stanje predmeta spreminjamo zgolj s pomočjo robota. Tako gibanje robota zagotovo spada med neodvisne spremenljivke, medtem ko stanje predmeta spada med odvisne spremenljivke. Na spremembo stanja predmeta pa vplivajo še določene spremenljivke, ki povezujejo stanje robota in stanje predmeta. Precej pomembno je na primer v kateri točki robot potiska predmet in pod kakšnim kotom. Pomembno je tudi, koliko časa potiskanje traja. V nadaljevanju te točke si bomo vsako skupino spremenljivk nekoliko bolj natančno ogledali.

- Gibanje robota

  Vektor gibanja robota je definiran s hitrostjo težišča robota v in kotno

hitrostjo robota ω. Obe spremenljivki sta v modelu potiskanja predmetov pomembni, saj njuno spreminjanje povzroči različne premike predmeta.

- Stanje sistema

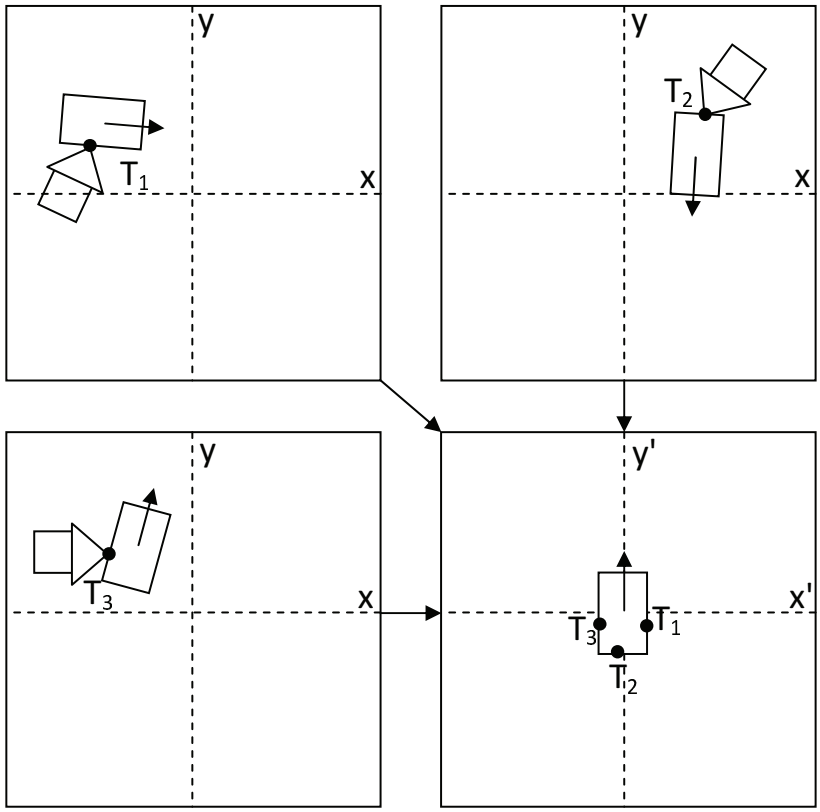
  Izbira spremenljivk, ki določajo stanje sistema, je nekoliko zahtevnejša, saj moramo z njimi zaobjeti povezavo med stanjem robota in stanjem predmeta. Pri tem je stanje predmeta podano na dva načina: kot poligon O ali kot vektor o.

  Prva ideja za izbiro ustrezne spremenljivke je točka dotikališča med robotom in predmetom, saj je njen pomen intuitivno razumljiv. Vpeljava točke dotikališča je smiselna zgolj v primeru, če je neodvisna od stanja predmeta. Vemo namreč, da trenutno stanje predmeta na premik predmeta ne vpliva, če znotraj celotnega poligona Z veljajo enake zakonitosti. Tej predpostavki je v našem primeru zadoščeno. Tako si moramo izbrati osnovni koordinatni sistem, v katerem bomo točko dotikališča podajali. Najpreprostejša izbira zanj je kar $o = (0, 0, 0)$. V tem koordinatnem sistemu je predmet v središču in usmerjen navpično navzgor. Koordinatni sistem označimo z x'y'. Pretvorbo iz osnovnega koordinatnega sistema xy v sistem x'y' prikazuje Slika 7.1. Matematično je prikazana transformacija sestavljena iz vrteža in premika koordinatnega sistema. Vrtež je opisan z enačbama 5.5, premik pa z enačbama 7.1, kjer (a, b) predstavlja vektor premika.

$$\begin{aligned} x &= x' + a \\ y &= y' + b \end{aligned} \tag{7.1}$$

  Za zapis točke dotikališča robota in predmeta v koordinatnem sistemu x'y' je primernejši zapis v polarnih koordinatah $T = (\alpha, r)$. Zaradi dvojne opredelitve stanja predmeta je namreč očitno, da je pri tem zapisu koordinata r odveč. V primeru dotikanja robota in predmeta jo namreč lahko enolično izračunamo iz informacije o stanju predmeta v obliki poligona O in kota α. Primer, ko se robot in predmet ne dotikata, za nas ni pomemben, saj ne omogoča spreminjanja stanja predmeta. S tem razmislekom smo definirali in utemeljili prvo izmed spremenljivk, ki določa stanje sistema. Zgolj s središčnim kotom α v koordinatnem sistemu x'y' pa stanje robota še ni enolično določeno.

  Smiselna se zdi še vpeljava kota ϕ, ki opisuje kot med pravokotnico na

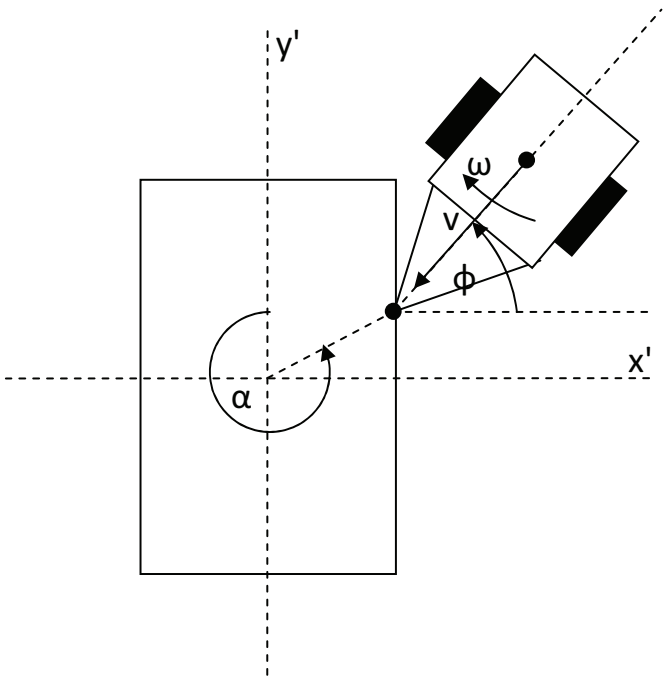


Slika 7.1: Trije primeri transformacije osnovnega koordinatnega sistema xy v koordinatni sistem x'y', ki je uporabljen za obravnavo točke dotikališča.

dotikajočo se stranico predmeta in kotom θ'_r v koordinatnem sistemu x'y' (glej Sliko 7.2). S parametroma α in ϕ je stanje robota določeno enolično.

- Premik predmeta

  Vsako spremembo stanja predmeta lahko opišemo s stanjem predmeta pred in po spremembi. Zdi se nesmiselno za vsak učni primer podati začetno in končno stanje predmeta, saj vemo, da je premik neodvisen od začetnega stanja predmeta. Tako namesto končnega in začetnega stanja predmeta vpeljemo spremembo stanja, ki jo določa vektor $\Delta o = (\Delta x_o, \Delta y_o, \Delta\theta_o)$. Takšna definicija še ni neodvisna od začetnega stanja predmeta, saj za isto spremembo pri različnih začetnih vrednostih θ_o dobimo različne vektorje Δo. Problem lahko rešimo z zahtevo, da se vektor Δo vedno podaja v koordinatnem sistemu x'y'. Pretvorbe lahko



Slika 7.2: Prikaz pomembnih neodvisnih spremenljivk pri izdelavi modela. Puščice prikazujejo smer naraščanja posameznih količin.

dosežemo z vrtežem, ki ga določata enačbi 5.5.

- Sprememba stanja sistema

  Sprememba stanja sistema je določena s spremenljivkama $\Delta\alpha$ in $\Delta\phi$. Izbira koordinatnega sistema, v katerem ju podajamo, ni pomembna.

- Čas

  Vse spremembe se dogajajo določen čas, zato je pomembno vedeti, v kolikšnem času je bila določena sprememba povzročena. Ker želimo, da je model kar najbolj preprost, se splača razmisliti o možnosti, da izvajanje vsake akcije opazujemo kar konstanten čas. Tako bi lahko čas kot spremenljivko v modelu kar izpustili. Z razmislekom ugotovimo, da pri tej poenostavitvi ne izgubimo skoraj nobene informacije. Vsako dlje trajajočo akcijo lahko namreč zapišemo kot zaporedje krajših akcij. Nekaj težav lahko predstavljajo le daljše akcije, katerih trajanje ni večkratnik izbrane časovne konstante. Takrat lahko pride do manjših napak, ki pa so v primerjavi z napakami učenja zanemarljive. Negativna posledica

tega pristopa pa je, da se z zaporedjem akcij napaka akcij sešteva. Tako je lahko za določeno dlje trajajočo akcijo precej večja, kot bi bila, če bi akcijo opazovali neposredno. Kakorkoli, poenostavitev modela je na ta način precejšnja in zato se eliminacija časa zdi smiselna.

Po analizi vseh skupin spremenljivk nam ostane zgolj še vprašanje, katere izmed identificiranih spremenljivk so neodvisne in katere odvisne. Robota upravljamo preko spremenljivk v in ω, zato ti dve spremenljivki zagotovo spadata med neodvisne spremenljivke. Tudi izbira spremenljivk α in ϕ ob začetku potiskanja predmeta je poljubna in zato lahko spremenljivki uvrstimo med neodvisne spremenljivke. Opis sprememb, ki jih potisk povzroči, pa je določen z odvisnimi spremenljivkami, saj teh vrednosti ne moremo neposredno nadzorovati. Spremenljivke Δx_o, Δy_o, $\Delta\theta_o$, $\Delta\alpha$ in $\Delta\phi$ tako uvrstimo med odvisne spremenljivke.

Če razmislek povzamemo, lahko določimo naslednje skupine spremenljivk:

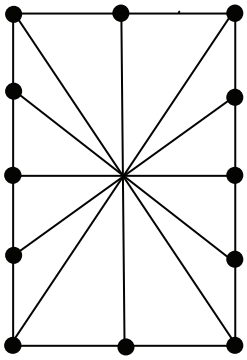
- neodvisne spremenljivke: v, ω, α in ϕ;
- odvisne spremenljivke: Δx_o, Δy_o, $\Delta\theta_o$, $\Delta\alpha$ in $\Delta\phi$;
- konstanta: t.

Učenje modela potiskanja predmetov predstavlja določitev funkcij f_x, f_y, f_θ, f_α in f_ϕ tako da velja:

- $\Delta x_o = f_x(\alpha, \phi, v, \omega)$;
- $\Delta y_o = f_y(\alpha, \phi, v, \omega)$;
- $\Delta\theta_o = f_\theta(\alpha, \phi, v, \omega)$;
- $\Delta\alpha = f_\alpha(\alpha, \phi, v, \omega)$;
- $\Delta\phi = f_\phi(\alpha, \phi, v, \omega)$.

Zgolj zaradi uporabe v nadaljevanje definirajmo še strnjen zapis vseh funkcij modela, ki je $(\Delta x_o, \Delta y_o, \Delta\theta_o) = f(\alpha, \phi, v, \omega)$.

Za časovno konstanto smo izbrali $\Delta t = 0.5\,s$. Večja konstanta je boljša zaradi manjšega vpliva napake kamere, medtem ko je slabša za točen zapis dlje trajajoče akcije kot zaporedja krajših akcij. Izbrana vrednost predstavlja smiseln kompromis med obema problemoma.



Slika 7.3: Primer razredov za spremenljivko α. Točke določajo meje posameznih razredov.

7.2 Strategija učenja

V prejšnji točki smo ugotovili, katere zakonitosti med spremenljivkami moramo identificirati. V tej točki pa se bomo posvetiti učnim primerom, iz katerih bomo poskušali te zakonitosti odkriti. Učni primeri morajo biti dovolj raznoliki, da se da zakonitosti iz njih razbrati.

7.2.1 Določitev razredov

Zaradi želje po čim večji raznolikosti učnih primerov smo se odločili izdelati razrede za vsako izmed neodvisnih spremenljivk. V procesu zbiranja učnih primerov za vsako kombinacijo razredov zberemo k primerov.

- Določitev razredov za kot α

  Vsako stranico predmeta z dolžino l centimetrov razdelimo na $n' = \lfloor \frac{l}{5} \rfloor$ enakih delov. Vsak izmed delov predstavlja en razred. Teh n' razredov je definiranih z $n' + 1$ točkami (glej Sliko 7.3). Za vsako izmed točk se izračuna ustrezen središčni kot α. Razredi so določeni z vrednostmi središčnih kotov α točk, ki jih omejujejo. Zaradi lažjega zapisa v nadaljevanju vpeljimo še število n, ki predstavlja vsoto n' po vseh stranicah predmeta.

- Določitev razredov za kot ϕ

  Vrednosti spremenljivke ϕ so omejene z obliko odbijača robota. V primeru velikih ali malih vrednosti se robot predmeta ne dotika več s kon-

ico odbijača, ampak kar s celotno stranico. Takim učnim primerom bi se radi izognili. Smiselna določitev razredov za kot ϕ je tako $(-30°, -10°)$, $[-10, 10°]$ in $(10°, 30°)$.

- Določitev vrednosti za hitrost težišča v

  Ker lahko vrednosti hitrosti težišča določamo neposredno, zanje ni potrebno definirati razredov, ampak kar točne vrednosti, pri katerih bomo zbirali učne primere. Izbrani vrednosti za hitrost težišča v sta 5 cm/s in 10 cm/s.

- Določitev vrednosti za kotno hitrost ω

  Podobno kot hitrost težišča lahko tudi kotno hitrost spreminjamo neposredno. Tako tudi v tem primeru namesto razredov določimo kar točne vrednosti. Vrednosti, uporabljene za učenje, so $-0.5/s$, 0 in $0.5/s$. Podane so v radianih.

Če s k označimo število učnih primerov, ki jih zberemo za posamezno kombinacijo zgoraj opisanih razredov, potem celotno število učnih primerov N določa enačba 7.2.

$$N = n * 3 * 2 * 3 * k = 18 * n * k \tag{7.2}$$

Pri implementaciji smo uporabili $k = 2$. Posamezne vrednosti n so odvisne od velikosti predmeta, zato si bomo konkretne vrednosti pogledali v nadaljevanju. Če predpostavimo vrednost $n = 12$, dobimo vrednost $N = 432$. Denimo, da za vsak učni primer porabimo približno 8 sekund. V tem primeru zbiranje vseh učnih primerov traja skoraj eno uro. V naslednji točki si bomo pogledali, kako je mogoče čas učenja skrajšati.

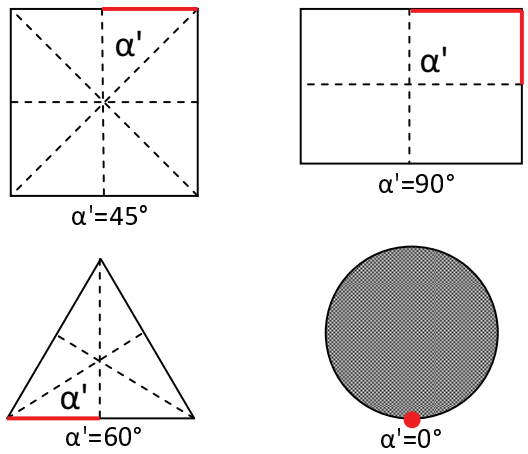
7.2.2 Pohitritev zbiranja učnih primerov z uporabo simetrije

Časovno komponento učenja lahko precej zmanjšamo z upoštevanjem simetrije predmeta. Ta omogoča, da lahko ob vsakem zajetem učnem primeru sklepamo še na nekatere druge učne primere.

Simetrijo predmeta lahko ugotovimo, če je predmet predstavljen kot poligon O. Iz gimnazijske matematike je znano, da mora simetrijska premica bodisi sovpadati s simetralo stranice bodisi s simetralo kota predmeta. To je nujen, ni pa še zadosten pogoj za simetrijo. Vsako potencialno simetrijsko premico moramo preveriti, če resnično razdeli predmet na dva simetrična dela.

Na tak način lahko določimo število simetrijskih premic, ki pa ga moramo še razpoloviti, saj je pri opisanem postopku vsaka šteta dvakrat. Tako dobimo število s simetral določenega predmeta.

Če je število simetral večje od 0, potem lahko izračunamo kot $\alpha' = \frac{180°}{s}$. Ta kot predstavlja središčni kot dela predmeta, ki ga je potrebno uporabiti za učenje (glej Sliko 7.4, kjer je ta del prikazan z rdečo barvo). Na podlagi določenega učnega primera iz označenega dela lahko sklepamo še na $2 * s - 1$ učnih primerov na drugih mestih predmeta. To pomeni, da je število učnih primerov, ki jih moramo zbrati ob upoštevanju simetrije $N' = \frac{N}{2*s}$.

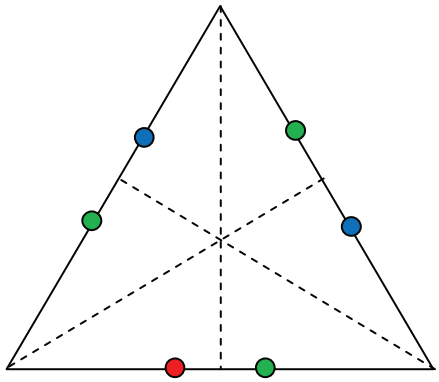


Slika 7.4: Prikaz, kako lahko z uporabo simetrije zmanjšamo čas učenja. Z rdečo je označen del predmeta, ki je pomemben za učenje. Iz učnih primerov iz označenega dela lahko sklepamo na učne primere vseh ostalih delov predmeta.

Način sklepanja iz enega učnega primera na ostale učne primere je v splošnem določen z vrtežem koordinatnega sistema x'y'z'. Enačbe, ki veljajo pri tem, so precej kompleksne [14], zato se jih za naš primer splača kar najbolj poenostaviti. Z razmislekom ugotovimo, da je poljubno točko predmeta mogoče preslikati v vse njene simetrijske točke zgolj z uporabo dveh transformacij:

- vrtež koordinatnega sistema x'y' za poljuben kot β in
- vrtež koordinatnega sistema x'y'z' za 180°okrog osi y'.

Primer prikazuje Slika 7.5.



Slika 7.5: Prikaz točk, ki so si zaradi simetrije ekvivalentne. Rdeča točka predstavlja zajet učni primer, modre so tiste točke, do katerih lahko pridemo zgolj z vrtežem koordinatnega sistema x'y', zelene pa so točke, do katerih lahko pridemo še z vrtežem koordinatnega sistema x'y'z'.

Če je kot, za katerega moramo zavrteti koordinatni sistem x'y' β, potem velja povezava 7.3, kjer $(\Delta x_o(\beta), \Delta y_o(\beta))$ predstavljata zavrtene koordinate, skladno z enačbo 5.5.

$$(\Delta x_o, \Delta y_o, \Delta\theta_o) = f(\alpha, \phi, v, \omega) \Longrightarrow (\Delta x_o(\beta), \Delta y_o(\beta), \Delta\theta_o) = f(\alpha + \beta, \phi, v, \omega) \tag{7.3}$$

Za tiste simetrijske točke, do katerih ne moremo priti zgolj z vrtežem koordinatnega sistema x'y', je potreben še vrtež za 180° okrog osi y'. Ta vrtež pa je določen s povezavo 7.4.

$$(\Delta x_o, \Delta y_o, \Delta\theta_o) = f(\alpha, \phi, v, \omega) \Longrightarrow (-\Delta x_o, \Delta y_o, -\Delta\theta_o) = f(-\alpha, -\phi, v, -\omega) \tag{7.4}$$

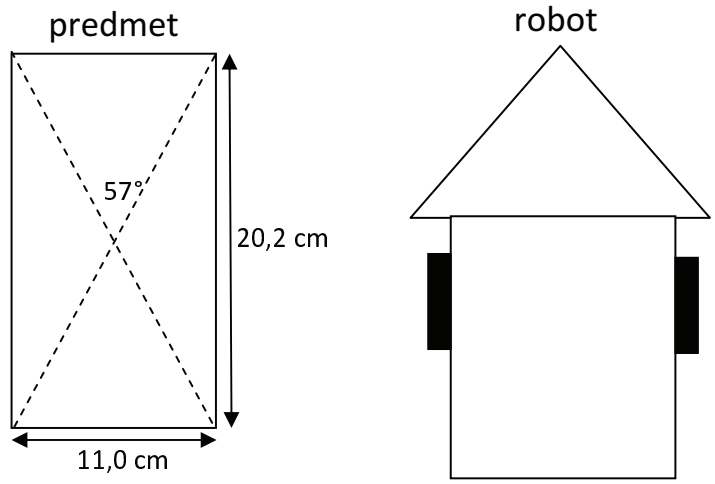
Postopek, pri katerem iz zajetega učnega primera določimo vsakega izmed ostalih $2 * s - 1$ učnih primerov, je pri uporabi pravkar izpeljanih enačb naslednji:

- uporaba povezave 7.3 za vsak $\beta = 2j * \alpha'$, kjer je $j \in \mathbb{N}_0$, tako da velja $2j * \alpha' \in [0, 360)$ in
- uporaba povezave 7.4 za vsak primer, pridobljen v prejšnji točki.

Uporaba simetrije precej pohitri zbiranje učnih primerov, saj lahko v krajšem času zberemo več učnih primerov. Število učnih primerov, ki jih uporabimo za izdelavo kvalitativnega modela, pri tem ostane enako.

7.2.3 Opis predmeta za potiskanje

Učenje je bilo preizkušeno na predmetu prikazanem na Sliki 7.6.



Slika 7.6: Prikazane so točne dimenzije uporabljenega predmeta. Zgolj za primerjavo je zraven podana še velikost robota.

Za prikazan predmet lahko določimo vrednost $n = 2 + 4 + 2 + 4 = 12$.

7.3 Izdelava modela

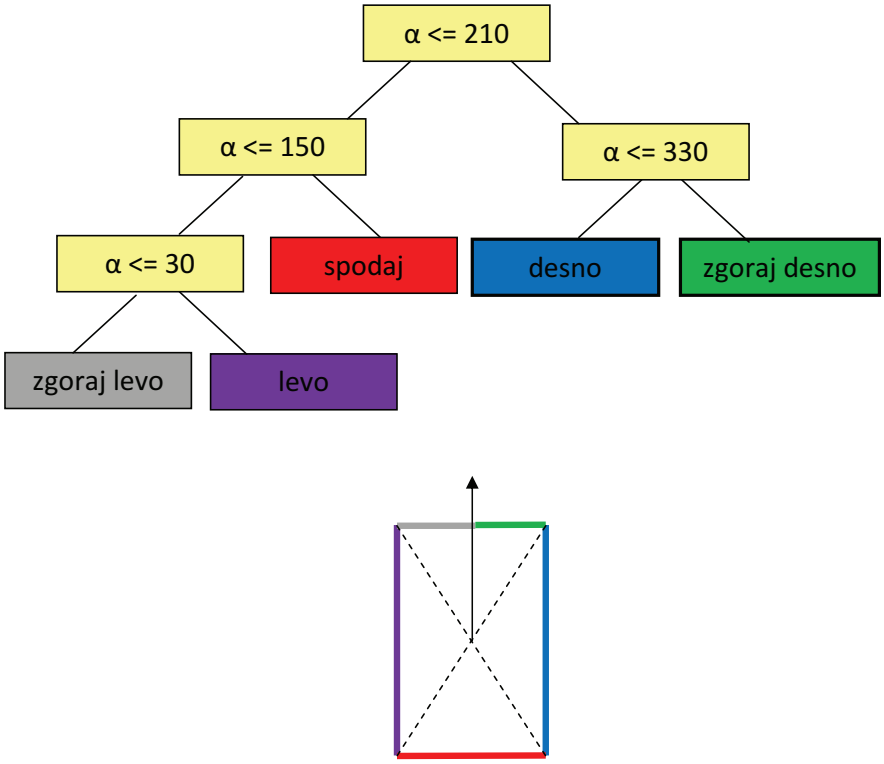
Na podlagi zbranih učnih primerov se lahko lotimo izdelave modelov. Najprej si bomo pogledali izdelavo kvalitativnega in nato še kvantitativnega modela.

7.3.1 Izdelava kvalitativnega modela

Na podlagi enačb 7.2 smo videli, da lahko število učnih primerov presega nekaj sto. Tako veliko število pa začne algoritmu QUIN povzročati težave [7], saj modela ni sposoben inducirati v razumnem času. Na ta problem je Šuc opozoril tudi v svojem predavanju, ki je v video obliki dostopno na [28].

Precejšnjo poenostavitev učenja lahko dosežemo z upoštevanjem specifičnih lastnosti predmeta. Namesto učenja kvalitativnega modela za celoten predmet naenkrat, lahko najprej izdelamo kvalitativne modele za potiskanje vsake izmed stranic predmeta. Te nato združimo v en, skupen, kvalitativni model. To povsem preprosto naredimo na podlagi neodvisne spremenljivke

α (glej Sliko 7.7). Vsaki stranici tako ustreza določeno poddrevo. Postopek izdelave drevesa lahko avtomatiziramo, saj so stranice predmeta razvidne iz predstavitve predmeta v obliki poligona O.



Slika 7.7: Prikaz, kako lahko uporabimo informacije o predmetu in namesto indukcije zgolj enega kvalitativnega drevesa iz vseh učnih podatkov, induciramo pet manjših induciranih dreves, ki jih potem združimo.

Ob upoštevanju opisane poenostavitve se čas izdelave kvalitativnega modela za posamezno stranico skrajša pod 5 sekund. Tako lahko za učenje brez težav uporabimo tudi več sto učnih primerov. To našim potrebam zadostuje.

Izdelavo modela bi lahko nadalje pohitrili še z upoštevanjem simetrije. V tem primeru bi izdelali kvalitativni model zgolj za del, ki je na Sliki 7.4 označen z rdečo barvo. Na podlagi kvalitativnega modela za omenjen del bi sklepali na celoten kvalitativni model. Število učnih primerov v našem primeru ni tako veliko, da bi bila uporaba simetrije pri izdelavi kvalitativnega modela smiselna. To je razlog, da te izboljšave nismo implementirali. Smo pa njeno implementacijo predlagali v smernicah za nadaljnje delo.

7.3.2 Izdelava numeričnega modela

Izdelava kvantitativnega modela poteka v več korakih. Za izbrane želene vrednosti neodvisnih spremenljivk se najprej izračuna numerično napoved zgolj z uporabo LUR. Te napovedi se nadalje spremeni tako, da ustrezajo kvalitativnemu modelu. Izbrane želene vrednosti neodvisnih spremenljivk bomo imenovali testni primeri.

Pri določanju testnih primerov moramo upoštevati, da večje število primerov pomeni natančneje določene odvisnosti med spremenljivkami, hkrati pa je izračun časovno bolj zahteven. Med obema faktorjema je potrebno najti smiseln kompromis. Testni primeri morajo biti kar najbolj raznoliki. Pri implementaciji tega diplomskega dela smo se odločili za naslednje:

- α zavzame ustrezne vrednosti za vsako izmed točk določenih na Sliki 7.3. Točke, ki predstavljajo ogliščе predmeta, izpustimo, saj je obnašanje okrog oglišč zelo težko napovedati. Moč take množice je $|\alpha|$;
- $\phi \in \{-15, 0, 15\}$. Moč množice je $|\phi|$;
- $v \in \{10, 20\}$. Moč množice je $|v|$;
- $\omega \in \{-0.5, 0, 0.5\}$. Moč množice je $|\omega|$.

Vsaka kombinacija vrednosti neodvisnih spremenljivk predstavlja testni primer.

7.4 Rezultati učenja

Za določitev odvisnosti med neodvisnimi in odvisnimi spremenljivkami smo uporabili pristop prikazan s sliko 3.1. Iz učnih primerov smo s pomočjo algoritma QUIN izdelali kvalitativni model, ki smo ga potem s pomočjo algoritma Qfilter uporabili za izboljšanje numerične predikcije. Tako kvalitativni model kot rezultati numerične predikcije so predstavljeni v nadaljevanju.

7.4.1 Kvalitativni rezultati učenja

V skladu s poenostavitvijo opisano v točki 7.3.1 bomo predstavili rezultate učenja zgolj za eno izmed stranic predmeta. Izbrali smo si levo stranico. Njeno umestitev v popoln kvalitativni model prikazuje Slika 7.7. Inducirani modeli za ostale stranice so zelo podobni, saj smo pri določanju učnih primerov za ostale stranice upoštevali simetrijo predmeta.

Pri pridobivanju učnih primerov za izbrano stranico namenoma nismo upoštevali simetrije med zgornjim in spodnjim delom stranice. Na ta način bomo lahko preverili, če je simetrijo sposoben odkriti tudi algoritem QUIN sam.

Pri učenju kvalitativnih modelov smo izdelali pet kvalitativnih dreves, po enega za vsako izmed odvisnih spremenljivk. V nadaljevanju si bomo natančneje pogledali vsako izmed njih.

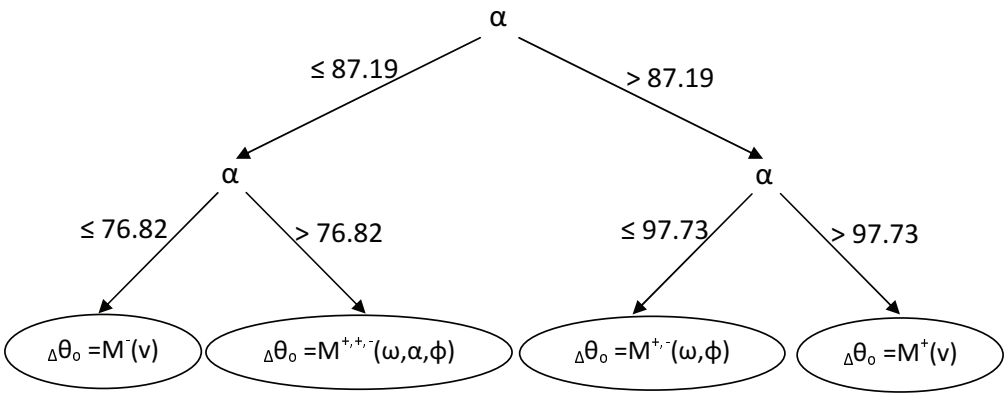
- Spremenljivka $\Delta\theta_o$ (Slika 7.8)

  Stranica je razdeljena na podlagi spremenljivke α na štiri dele. Precej lepo je vidna simetrija. V zgornjem delu velja kvalitativno omejena funkcija $\Delta\theta_o = M^-(v)$. Odvisnost pravi, da pri potiskanju z večjo hitrostjo povzročimo manjšo spremembo $\Delta\theta_o$ (na tem mestu naj samo spomnimo, da ne gledamo absolutnih vrednosti sprememb in je tako na primer sprememba -3 manjša kot sprememba -2). Za povsem spodnji del velja kvalitativno omejena funkcija $\Delta\theta_o = M^+(v)$, kar lepo kaže na simetrijo. V spodnjem delu je sprememba $\Delta\theta_o$ ob povečanju hitrosti večja. Nekoliko drugačno obnašanje pa velja v srednjem delu (v okolici $\alpha = 90°$). Tam na spremembo vplivata predvsem spremenljivki ω in ϕ. Odvisnost pri ω določa, da se z zavijanjem v desno sprememba $\Delta\theta_o$ veča. Odvisnost pri ϕ pa določa, da se spremembe $\Delta\theta_o$ povečujejo, če se kot ϕ povečuje. Odvisnost od kota α je bila najdena zgolj v zgornjem izmed osrednjih delov. To nekoliko pokvari simetrijo, vendar očitno njen vpliv ni posebno velik in je bil zato opažen zgolj v enem delu. Odvisnost določa, da so spremembe $\Delta\theta_o$ večje, ko se pomikamo navzdol po stranici (povečujemo α). Vse kvalitativne odvisnosti se zdijo intuitivno smiselne.

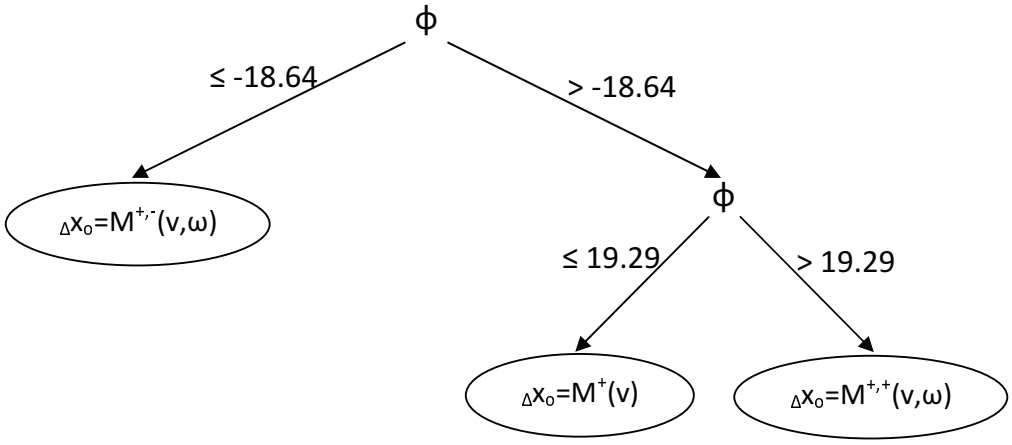
- Spremenljivka Δx_o (Slika 7.9)

  Stranico se v tem primeru razdeli na podlagi kota ϕ, pod katerim se robot dotika stranice. Vidimo, da v vseh delih velja, da večja hitrost povzroči večjo spremembo Δx_o. Posebej zanimive so odvisnosti od kotne hitrosti ω pri velikih ali majhnih kotih ϕ. Odvisnost določa, da se v primeru, ko se robot dotika predmeta zelo z leve (zelo negativna vrednost kota ϕ), sprememba Δx_o povečuje, če robot zavija levo. Podobno se v primeru, ko se robot dotika predmeta zelo z desne (zelo pozitivna vrednost kota ϕ), sprememba Δx_o povečuje, če robot zavija desno. Ugotovitev se zdi intuitivno smiselna.

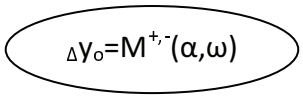
- Spremenljivka Δy_o (Slika 7.10)



Slika 7.8: Prikaz kvalitativnega drevesa za spremenljivko $\Delta\theta_o$.



Slika 7.9: Prikaz kvalitativnega drevesa za spremenljivko Δx_o.



Slika 7.10: Prikaz kvalitativnega drevesa za spremenljivko Δy_o.

Kvalitativno drevo, ki prikazuje kvalitativne odvisnosti, je v tem primeru zreducirano zgolj na eno kvalitativno omejeno funkcijo. Preprostost je razumljiva, saj pri potiskanju v levo stranico predmeta težko povzročimo spremembe spremenljivke Δy_o. Inducirana kvalitativno omejena funkcija določa, da se sprememba Δy_o veča, če se pomikamo po stranici navzdol (povečujemo kot α). Dodatno pa kvalitativna omejena funkcija določa še, da se z zavijanjem v desno vrednost Δy_o zmanjšuje. Tudi v tem primeru se zdijo ugotovljene odvisnosti intuitivno smiselne.

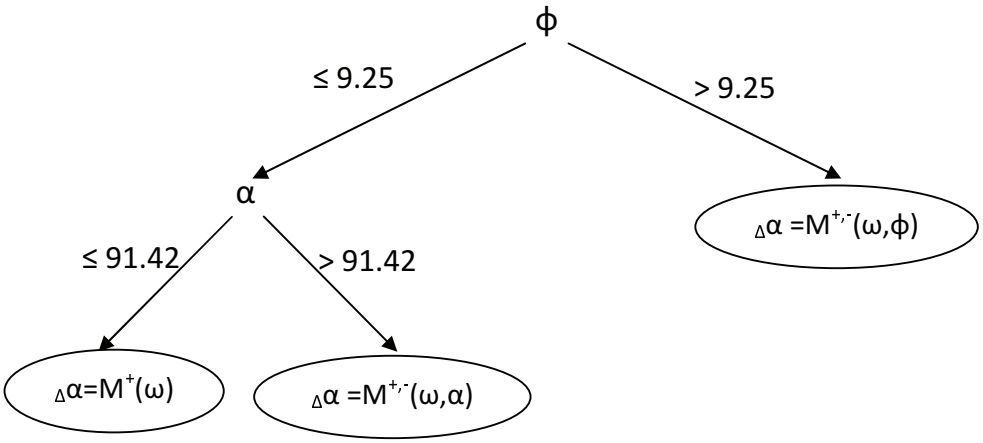
- Spremenljivka $\Delta\alpha$ (Slika 7.11)

  V primeru spremenljivke $\Delta\alpha$ algoritem QUIN simetrije ni odkril. Prva delitev na podlagi spremenljivke ϕ ni smiselna. Nadaljnja delitev levega poddrevesa na podlagi spremenljivke α je precej boljša. V vseh kvalitativno omejenih funkcijah vidimo, da zavijanje v desno povečuje $\Delta\alpha$. QUIN je v določenih predelih identificiral še odvisnosti od α ali ϕ. Odvisnosti se zdijo intuitivno pravilne, nerazumljiva nam je zgolj določitev predelov, na katerih te zakonitosti veljajo.

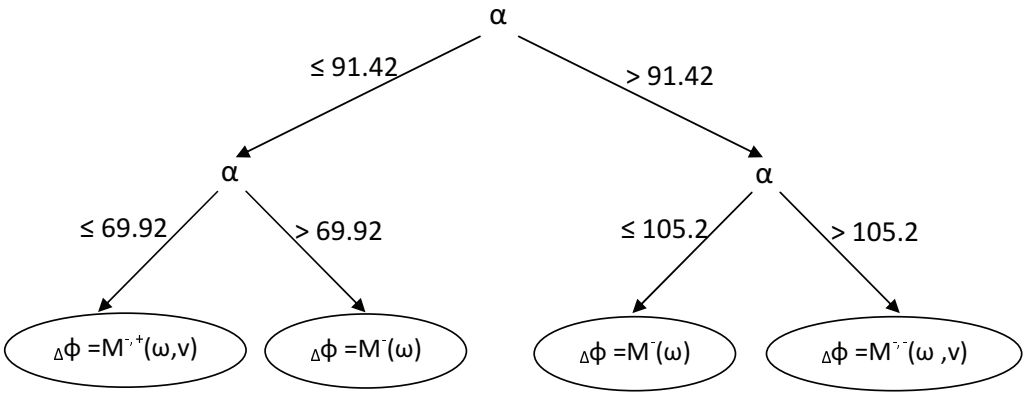
- Spremenljivka $\Delta\phi$ (Slika 7.12)

  V tem primeru je simetrija lepo vidna. Stranica je na podlagi spremenljivke α razdeljena na štiri dokaj simetrične dele. Tudi inducirane kvalitativno omejene funkcije lepo kažejo na simetrijo predmeta. V vseh predelih je odkrita zakonitost, ki pravi, da zavijanje v desno zmanjšuje $\Delta\phi$. Na robovih je izrazita še odvisnost od hitrosti težišča robota v. Odvisnost v zgornjem delu pravi, da večanje hitrosti povečuje $\Delta\phi$. Simetrično pa se v spodnjem delu $\Delta\phi$ z večanjem hitrosti zmanjšuje. Vse induciranje odvisnosti se zdijo intuitivno pravilne.

Za konec naj zgolj omenimo, da je pri uporabi algoritma QUIN potrebno nastaviti parameter, ki ocenjuje napako posameznih učnih primerov. Napaka je podana sorazmerno z razliko med maksimalnim in minimalnim elementom v



Slika 7.11: Prikaz kvalitativnega drevesa za spremenljivko $\Delta\alpha$.



Slika 7.12: Prikaz kvalitativnega drevesa za spremenljivko $\Delta\phi$.

Spremenljivka	Predvidena napaka
$\Delta\theta_o$	2 %
Δx_o	2 %
Δy_o	40 %
$\Delta\alpha$	20 %
$\Delta\phi$	4 %

Tabela 7.1: Določitev napake učnih primerov.

učnih primerih. Pri problemski domeni potiskanja predmetov je glavni vir napake zaznavanje točne lokacije in orientacije robota ter predmeta. Absolutno napako posamezne koordinate lokacije smo ocenili na 0.5 cm, absolutno napako orientacije pa na 4^o. Te vrednosti so nam predstavljale osnovo za določitev relativne napake. Konkretne vrednosti relativne napake pri potiskanju leve stranice predmeta prikazuje Tabela 7.1. Pri uporabi drugačnih vrednosti se inducirana drevesa hitro spremenijo in postanjejo težko intuitivno razumljiva. Za izdelavo kvalitativnih modelov smo uporabili algoritem QUIN z uporabniškim vmesnikom, ki je dostopen na [27].

Na podlagi prikazanih rezultatov lahko zaključimo, da je QUIN induciral intuitivno razumljiva drevesa in ponudil zanimiv vpogled v zakonitosti pri potiskanju predmetov. Nekoliko težje razumljivo se zdi zgolj kvalitativno drevo za spremenljivko $\Delta\alpha$. Vendar pa se tudi v tem primeru zakonitosti zdijo pravilne. Pravilnost kvalitativnih modelov bomo točno preverili kasneje s pomočjo kvalitativnega planiranja.

7.4.2 Numerični rezultati učenja

Vsaka kombinacija neodvisnih spremenljivk iz množic, opisanih v 8.3.2, se nahaja v tabeli, ki predstavlja rezultat numeričnega učenja. Manjša podmnožica rezultatov je predstavljena v Tabeli 7.2. Vsaka vrstica določa vrednost funkcije f pri izbranih vrednostih neodvisnih spremenljivk. Napovedane vrednosti funkcije f so kvalitativno pravilne.

Za lažje razumevanje numeričnega učenja smo v Tabeli 7.3 predstavili nekaj primerov nepopravljenih in popravljenih rezultatov. Nepopravljeni rezultati so pridobljeni zgolj z uporabo LUR. Popravljeni rezultati pa so z uporabo kvalitativnega zvestega učenja spremenjeni tako, da ustrezajo kvalitativnim modelom.

Pravilnost numeričnih napovedi bomo preverili kasneje z uporabo kvantitativnega planiranja.

α	ω	ϕ	v	$\Delta\theta_o$	Δx_o	Δy_o	$\Delta\alpha$	$\Delta\phi$
0.0	0.0	0.0	10	-0.28	0.04	-2.17	0.49	0.18
0.0	0.0	0.0	20	-1.72	-0.02	-4.94	1.6	1.25
0.0	0.0	-15.0	10	0.24	-0.24	-1.97	2.45	0.94
0.0	0.0	-15.0	20	1.17	-0.14	-3.74	5.23	-1.88
0.0	0.0	15.0	10	-1.72	0.14	-2.17	-1.09	2.76
0.0	0.0	15.0	20	-5.88	0.45	-4.74	0.23	5.28

Tabela 7.2: Primer rezultatov numeričnega učenja.

α	ω	ϕ	v	$\Delta\theta_o$- LUR	$\Delta\theta_o$- Qfilter
270.0	0.0	0.0	20.0	1.79	1.79
270.0	0.5	0.0	20.0	5.05	5.05
270.0	-0.5	0.0	20.0	-3.4	-3.4
270.0	0.0	-15.0	10.0	5.72	5.13
270.0	0.5	-15.0	10.0	4.22	5.13
270.0	-0.5	-15.0	10.0	-0.27	-0.27
270.0	0.0	-15.0	20.0	5.21	5.13
270.0	0.5	-15.0	20.0	9.46	9.46
270.0	-0.5	-15.0	20.0	0.99	0.99
270.0	0.0	15.0	10.0	-1.56	-1.56
270.0	0.5	15.0	10.0	2.63	4.61

Tabela 7.3: Primer popravljenih napovedi.

Poglavje 8

Planiranje in izvedba plana

V poglavju 5 smo si pogledali teoretično osnovo, ki smo jo uporabili za izvedbo planiranja. V tej točki bomo predstavili uporabo kvalitativnega planiranja na problemski domeni potiskanja predmetov. Za primerjavo smo implementirali tudi planiranje, ki pri določitvi plana ne uporablja kvalitativnih modelov. Poimenovali smo ga kvantitativno planiranje. Podrobnosti obeh implementacij skupaj z rezultati so predstavljeni v nadaljevanju.

8.1 Splošne določitve problema

Problema kvantitativnega in kvalitativnega planiranja pri potiskanju predmetov imata nekaj skupnih lastnosti. Da bi se izognili ponavljanju, jih bomo navedli v tej točki. Pogledali si bomo določitev stanja sistema in določitev akcij.

8.1.1 Določitev stanja sistema

Pri obeh načinih planiranja moramo definirati stanje sistema. Smiselno stanje vsebuje informacijo o stanju predmeta in stanju robota.

- Stanje predmeta

  Uporabljena predstavitev stanja predmeta (x_g, y_g, θ_g) je definirana v koordinatnem sistemu, kjer je želeno stanje določeno z $(0, 0, 0)$. Za izračun stanja uporabimo enačbi 5.5 in 7.1, ki določata vrtež in premik koordinatnega sistema. Uporabljena predstavitev stanja je smiselna, ker je informacija o razliki med trenutnim in želenim stanjem shranjena kar v stanju predmeta.

- Stanje robota

  Stanje robota je v primeru dotikanja med robotom in predmetom lahko enolično predstavljeno s kotom α in ϕ. V primeru, da se predmet in robot ne dotikata, pa točno stanje robota ni pomembno. Vsa taka stanja so si namreč enakovredna v smislu, da bo za vsako izmed njih potrebna akcija, ki bo povzročila spremembo, da se bosta robot in predmet dotikala. Tako je dovolj, če poleg α in ϕ uvedemo novo neznanko $seDotikata$, ki zavzame vrednost 1, če se robot in predmet dotikata. Drugače ima vrednost 0.

Če združimo razmišljanje o stanju predmeta in stanju robota, lahko ugotovimo, da trenutno stanje sistema opisuje vektor $S = (x_g, y_g, \theta_g, \alpha, \phi, seDotikata)$, želeno stanje pa vektor $G = (0, 0, 0, \_, \_, \_)$, kjer _ predstavlja poljubno vrednost spremenljivke.

Pri zbiranju učnih primerov smo se omejili zgolj na določene vrednosti posameznih spremenljivk. Kot ϕ se je na primer vedno gibal v mejah od približno -30^o do približno 30^o. Zakonitosti, ki veljajo pri potiskanju predmeta izven tega intervala (na primer pri ritenskem potiskanju, ko je kot $\phi = 180^o$), so neznane. To je razlog, da se želimo takim stanjem izogniti. V nadaljevanju pri opisu teh stanj uporabljamo izraz nezaželena stanja. Nezaželena stanja so določena z maksimalno in minimalno vrednostjo posamezne spremenljivke v učnih primerih.

8.1.2 Določitev akcij

Ob definiranem trenutnem in želenem stanju sistema moramo za izvedbo planiranja definirati še množico akcij. Definicija akcije, ki jo uporabljamo pri kvantitativnem in kvalitativnem planiranju, sta si zelo podobni. Pri kvantitativnem planiranju je vsaka akcija določena s četvorčkom $\langle P, SP, I, T\rangle$, medtem ko je pri kvalitativnem planiranju vsaka akcija določena z dvojčkom $\langle P, SP\rangle$. Na tem mestu bomo opredelili zgolj skupne elemente.

- Množica predpogojev P

  Množica predpogojev P določene akcije je definirana z vrednostmi tistih spremenljivk, ki nastopajo kot omejitve v kvalitativnih drevesih induciranih z algoritmom QUIN. Taki spremenljivki sta α in ϕ. Dodati moramo še spremenljivko $seDotikata$. Določeno akcijo potiskanja predmeta lahko izvedemo samo pod pogojem, če se robot dotika predmeta na določenem mestu in pod določenim kotom.

- Množica sprememb SP

Spremembo stanja, ki ga akcija povzroči, lahko definiramo z vektorjem (pri kvantitativnem planiranju) oziroma matriko (pri kvalitativnem planiranju) $SP = (\Delta x_o, \Delta y_o, \Delta\theta_o, \alpha_K, \phi_K, seDotikata_K)$. Vrednosti $(\Delta x_o, \Delta y_o, \Delta\theta_o)$ predstavljajo spremembo stanja predmeta, vrednosti $(\alpha_K, \phi_K, seDotikata_K)$ pa stanje robota po izvedbi akcije.

Vsebinsko lahko v okviru problemske domene potiskanja predmetov razlikujemo med dvema tipoma akcij:

- Akcije, ki predstavljajo zgolj premikanje robota

  Te so definirane z $A_1 = \langle(\_, \_, \_), (0, 0, 0, \alpha_K, \phi_K, 1)\rangle$. Ta vrsta akcij je lahko določena zgolj kvantitativno, saj za premikanje robota nimamo izdelanega kvalitativnega modela, ki je nujen za izvedbo kvalitativnega planiranja;

- Akcije, ki predstavljajo premikanje robota in predmeta

  Te so definirane z $A_2 = \langle(\alpha, \phi, 1), (\Delta x_o, \Delta y_o, \Delta\theta_o, \alpha_K, \phi_K, seDotikata)\rangle$. Konkretna določitev akcij je odvisna od uporabljenega načina planiranja, zato se k temu vprašanju vrnemo pri opisu vsakega izmed načinov planiranja.

8.2 Kvantitativno planiranje

V tej točki je podrobneje predstavljena opredelitev in implementacija kvantitativnega planiranja. Predstavitev stanja smo definirali v prejšnji točki, zato ga bomo v tej točki izpustili. Pogledali si bomo določitev kvantitativnih akcij, kvantitativno planiranje in izvedbo kvantitativnega plana.

8.2.1 Določitev akcij

V točki 8.1.2 smo predstavili elemente, ki nastopajo tako pri kvalitativnih kot pri kvantitativnih akcijah. Pri predstavitvi smo tako izpustili elementa I in T, ki nastopata zgolj pri kvantitativnem planiranju.

I določa način izvedbe kvantitativne akcije. Predstavljen je kot vektor z dolžino 2. Prvi element določa hitrost težišča robota v, drugi element pa kotno hitrost robota ω. T pa predstavlja čas izvedbe akcije. Določen je skladno z opisom v točki 4.2. Vsako kvantitativno akcijo lahko tako predstavimo kot četvorček $\langle P, SP, I, T\rangle$.

α	ϕ	v	ω	$\Delta\theta_o$	Δx_o	Δy_o	$\Delta\alpha$	$\Delta\phi$
49.28	-15.0	10	0	-2.71	2.2	0.24	1.23	5.88

Tabela 8.1: Primer določitve elementov kvantitativne akcije na podlagi tabele, ki je rezultat numeričnega učenja. Za prikazano vrstico so vrednosti naslednje: $\alpha = [47.44, 90.0]$, $\phi = [-30, -10]$, $v = 10$, $\omega = 0$, $\Delta\theta_o = -2.71$, $\Delta x_o = 2.2$ in $\Delta y_o = 0.24$.

V nadaljevanju se bomo podrobneje posvetili vsakemu tipu akcij (upoštevajoč vsebinsko delitev) posebej. Predstavili bomo tudi postopek določitve konkretnih kvantitativnih akcij iz podatkov, ki jih imamo na voljo.

- Akcije, ki predstavljajo zgolj premikanje robota

  Te akcije so definirane z $A_1 = \langle(\_, \_, \_), (0, 0, 0, \alpha_K, \phi_K, 1), \emptyset, 15\rangle$, kjer α_K in ϕ_K zavzameta vse različne vrednosti, ki nastopajo v tabeli, ki je rezultat numeričnega učenja. Takih akcij je $N_{A_1} = |\alpha| * |\phi|$. Čas trajanja smo določili po dokončani implementaciji in nekaj poskusih izvedbe kvantitativnega planiranja. Povprečni čas trajanja akcije A_1 smo ocenili na 15 sekund;

- Akcije, ki predstavljajo premikanje robota in predmeta

  Te akcije so definirane z $A_2 = \langle(\alpha, \phi, 1), (\Delta x_o, \Delta y_o, \Delta\theta_o, \alpha_K, \phi_K, seDotikata_K), (v, \omega), 1\rangle$. Vsaka vrstica iz tabele, ki je rezultat numeričnega učenja, ustreza natanko eni izmed tako definiranih akcij. Vseh takih akcij je $N_{A_2} = |\alpha| * |\phi| * |v| * |\omega|$. Intervale za spremenljivki α in ϕ smo določili enako kot v pri zbiranju učnih primerov (glej točko 7.2.1). Spremenljivke Δx_o, Δy_o, $\Delta\theta_o$, v in ω so vzete neposredno iz tabele, ki je rezultat numeričnega učenja (za primer glej Tabelo 8.1).

  Določitev vrednosti spremenljivk α_K, ϕ_K in $seDotikata_K$ je nekoliko bolj zapletena. Ker sta α in ϕ določena zgolj na podlagi intervala, je točno vrednost α_K in ϕ_K v trenutku definiranja akcij nemogoče napovedati. Vrednosti v tabeli, ki je rezultat numeričnega učenja, so bile pridobljene pri konkretni vrednosti α in ϕ ter zato niso povsem ustrezne. Najbolje je, če namesto točnih vrednosti zapišemo funkcijo, ki bo te vrednosti določila med planiranjem. Takrat bosta α in ϕ določena natančno in ne zgolj kot intervala. Smernice določitev funkcije za izračun α_K in ϕ_K so podane z naslednjim postopkom:

  1. s pomočjo kota α v trenutnem stanju in predstavitve predmeta kot

poligon O izračunamo točko dotikališča P med robotom in predmetom;

2. s pomočjo točke dotikališča P, kota ϕ v trenutnem stanju in predstavitve predmeta kot poligon O izračunamo lokacijo robota (x_r, y_r, θ_r);
3. s pomočjo lokacije robota (x_r, y_r, θ_r) in uporabo enačb za opis gibanja robota izračunamo novo lokacijo robota (x'_r, y'_r, θ'_r) za vektor gibanja $q = (v, \omega)$ in čas izvedbe $\Delta t = 0.5$ s;
4. lokacija robota mora biti določena v koordinatnem sistemu, v katerem je stanje predmeta določeno z $(0, 0, 0)$. Pri izvedbi akcije se je stanje predmeta spremenilo za $(\Delta x_o, \Delta y_o, \Delta\theta_o)$. Če želimo stanje predmeta transformirati v $(0, 0, 0)$, moramo novo lokacijo robota (x'_r, y'_r, θ'_r) premakniti za vektor $(\Delta x_o, \Delta y_o)$ in zavrteti za kot $\Delta\theta_o$. To storimo z uporabo enačb 5.5 in 7.1. Transformirano lokacijo robota označimo z $(x''_r, y''_r, \theta''_r)$;
5. s pomočjo transformirane lokacije robota $(x''_r, y''_r, \theta''_r)$ in predstavitve predmeta kot poligon O izračunamo α_K in ϕ_K.

Ob definirani množici akcij in stanju sistema se lahko lotimo izdelave kvantitativnega plana.

8.2.2 Določitev kvantitativnega plana

Osnovo za izdelavo plana predstavlja planiranje z uporabo analize sredstev in ciljev. Pri opisu postopka v točki 4.1 smo ugotovili, da postopek omogoča dve mesti za optimizacijo:

- Izbira cilja, ki ga bomo poskušali uresničiti najprej

  Akcije, ki jih uporabljamo pri kvantitativnem planiranju, spreminjajo vse elemente stanja. Tako nam cilja, ki ga bomo poskušali uresničiti najprej, ni potrebno izbrati. Uresničujejo se lahko vsi cilji skupaj.

- Izbira akcije, katere izvedbo bomo preizkusili najprej

  Vsaka akcija spremeni stanje sistema. Na podlagi spremembe stanja lahko preprosto ugotovimo, ali je akcija ustrezna ali ne. Ustrezne akcije bodo zmanjšale vrednost x_g in y_g. Vrednost θ_g bodo, upoštevajoč modul 360, približale vrednosti 0. Podrobnosti implementacije te ugotovitve v postopek planiranja si bomo pogledali v nadaljevanju.

Postopek kvantitativnega planiranja pričnemo z določitvijo akcij, ki jih lahko na podlagi predpogojev P izvedemo v trenutnem stanju. Preizkusimo vse izmed njih. Vsaka akcija spremeni stanje sistema. V vsakem novo pridobljenem stanju lahko spet preizkusimo vse akcije, katerih predpogoj P se ujema s stanjem sistema. Postopek se lahko nadaljuje, dokler eno izmed doseženih stanj ni dovolj blizu želenemu stanju sistema $(0, 0, 0)$. Pri vsakem stanju moramo še preveriti, če stanje predstavlja nezaželeno stanje. V tem primeru ga iz nadaljnje obravnave izpustimo.

Navidezno izvajanje akcij in prehajanje v nova stanja lahko opišemo z grafom, kjer stanja predstavljajo vozlišča, akcije pa povezave. Problem planiranja lahko tako prevedemo na iskanje najcenejše poti med trenutnim stanjem in želenim stanjem. Akcije imajo določeno trajanje. Ugotovili smo tudi, da obstaja hevristika, ki določa, katera stanja so bolj zaželena od drugih. Tako določen problem lahko rešujemo z uporabo algoritma A*.

Pri trenutni definiciji problema vidimo, da stanja, ki imajo manjše vrednosti x_g in y_g predstavljajo boljša stanja. Pri kotu θ_g je ocena zaradi računanja po modulu 360 nekoliko bolj zapletena. Ugotovimo lahko, da stanje z manjšo vrednostjo $min(\theta_g, 360-\theta_g)$ predstavlja boljše stanje. Na podlagi tega razmisleka lahko definiramo hevristiko, ki je določena z enačbo 8.1, kjer z predstavlja parameter za uravnavanje velikosti funkcije h^*. Prevelika vrednosti h^* lahko povzroči, da algoritem A* ni dostopen. Pri implementaciji je uporabljena vrednost $z = 2$.

$$h^* = \frac{\sqrt{x_g^2 + y_g^2} + min(\theta_g, 360 - \theta_g)/2}{z} \tag{8.1}$$

Kvantitativno planiranje pa se kljub uporabi hevristike in A* algoritma ne izkaže za posebej uspešnega. Ocena časovne zahtevnosti je zapletena, vendar pa nam že preprost razmislek pokaže, da se število stanj v grafu eksponentno povečuje.

Razmislek začnemo v trenutnem stanju. Od tam lahko navidezno izvedemo vsaj $|\alpha| * |\phi|$ akcij (akcije, ki predstavljajo zgolj gibanje robota in zato nimajo nobenega predpogoja). Tako pridemo do vsaj $|\alpha|*|\phi|$ novih stanj, kjer se robot dotika predmeta. Iz vsakega izmed novih stanj lahko izvedemo $|\alpha|*|\phi|+|v|*|\omega|$ akcij. Nekaj izmed tako pridobljenih stanj predstavlja že obiskana stanja, večina novih stanj pa je še ne obiskanih. Pri konkretni izdelavi plana se izkaže, da sta vsaj dva izmed novih stanj vedno ne obiskana. Graf lahko v tem primeru poenostavljeno predstavimo kot binarno drevo. Število stanj na n-tem nivoju je 2^n, kar kaže na eksponentno povečevanje. S hevristiko lahko iskanje izboljšamo, vendar pa problem še vedno ostane neobvladljiv.

Vse, kar lahko storimo je, da namesto zaporedja akcij, ki pripeljejo predmet do želenega stanja, iščemo zaporedje akcij, ki stanje zgolj nekoliko približajo želenemu stanju. Po izvedbi teh akcij se postopek planiranja ponovi. Tak način je uporabljen pri končni implementaciji. Plani vsebujejo od dve do štiri akcije.

8.2.3 Izvedba plana

Izvedba kvantitativnega plana je preprosta. Robot izvaja predvideno zaporedje akcij in pri tem beleži trenutno stanje. Po izvedbi vsake akcije se:

- na podlagi hevristične funkcije 8.1 ugotovi, če se je stanje z izvedbo akcije izboljšalo. Če se je, izvedbo nadaljujemo z naslednjo akcijo plana. V nasprotnem primeru izvedbo prekinemo in ponovno pričnemo s planiranjem;
- preveri, če trenutno stanje predstavlja nezaželeno stanje. Če ga, potem izvedbo prekinemo in ponovno pričnemo s planiranjem. V nasprotnem primeru izvedbo nadaljujemo z naslednjo akcijo plana.

Opisan postopek se ponavlja, dokler trenutno stanje ne sovpada z želenim stanjem.

8.2.4 Ugotovitve

Pri uporabi kvantitativnega planiranja smo identificirali dve glavni pomanjkljivosti:

- velika kompleksnost planiranja in zato nezmožnost izdelave celotnega plana;
- nezmožnost mehkega planiranja. Akcije v večini primerov ne pripeljejo točno v predvideno stanje. Smiselno bi bilo, če bi pri nadaljnji izvedbi akcij določenega plana upoštevali napako, ki so jo povzročile že izvedene akcije. To bi lahko rešili z uporabo algoritmov mehkega planiranja, vendar bi to še povečalo kompleksnost planiranja. Tudi določitev porazdelitvene funkcije, ki določa verjetnost prehoda v vsakega izmed mogočih stanj, bi bila precej zapletena. To sta glavna razloga, da se reševanja problema z uporabo mehkega planiranja nismo lotili.

Predstavljeni pomanjkljivosti nam je uspelo uspešno rešiti z uporabo kvalitativnega planiranja, ki je podrobneje opisano v nadaljevanju.

8.3 Kvalitativno planiranje

Kvalitativno planiranje smo splošno predstavili v poglavju 4. V začetku poglavja 8 smo predstavili tudi splošne ugotovitve, ki veljajo pri planiranju v domeni potiskanja predmetov. V tej točki so predstavljene preostale podrobnosti implementacije kvalitativnega planiranja na domeni potiskanja predmetov.

8.3.1 Priprava vhoda v kvalitativno planiranje

Problem kvalitativnega planiranja je definiran kot šestorček $\langle S, G, A, U, V, N\rangle$. Pri tem je vsaka akcija določena z dvojčkom $\langle P, SP\rangle$. Vsi elementi z izjemo N so pri vhodu določeni numerično. V začetku poglavja 8 smo že predstavili, kako je definirano začetno stanje S, želeno stanje G in posamezna akcija. V tej točki si bomo pogledali določitev elementov U, V in N. Opisali bomo tudi konkretno določitev akcij.

- Element U

  U določa numerične vrednosti spremenljivk, preko katerih upravljamo robota ob začetku izvajanja plana. Robota, ki je uporabljen pri izdelavi tega diplomskega dela, lahko upravljamo preko hitrosti težišča v in kotne hitrosti ω. Element U je tako določen z vektorjem $U = (v, \omega)$ in vrednostjo $v = 10$ in $\omega = 0$.

- Element V

  V določa parametre, ki jih uporabljamo pri konverziji med numeričnimi vrednostmi, ki jih uporablja robot, ki izvaja plan in vrednostmi, ki se uporabljajo pri kvalitativnem planiranju. Pri robotu, ki je uporabljen v tem diplomskem delu, sta dva parametra: hitrost težišča v in kotna hitrost ω. Element V nima človeku razumljive vsebine. Posamezni elementi vektorja V uravnavajo velikost podakcij, ki jih robot izvaja v drugi fazi kvalitativnega planiranja. Za učinkovito izvedbo planiranja jih je potrebno določiti empirično. Vrednost V, uporabljena pri implementaciji kvalitativnega planiranja v okviru tega diplomskega dela, je $(3, 1)$. Zaradi preprostejše izvedbe kvalitativnega planiranja bi bilo smiselno za določitev vektorja V uporabiti metode strojnega učenja. Te bi vektor določile avtomatsko na podlagi uspešnosti izvedbe. Idejo smo zapisali v smernicah za nadaljnje delo.

- Element N

N predstavlja kvalitativni model, ki velja pri izvedbi plana. Uporabljen model mora biti kar najbolj popoln, v smislu, da vsebuje vse zakonitosti, ki v sistemu veljajo. Na ta način se izognemo nepotrebnim nedefiniranim stanjem. Pri implementaciji smo uporabili kvalitativni model, določen z algoritmom QUIN. Popoln model v tem primeru določa kvalitativno drevo za vsako odvisno spremenljivko. Element N je tako definiran z vektorjem $N = (f_x, f_y, f_\theta, f_\alpha, f_\phi)$.

- Določitev akcij iz učnih primerov

  Posamezna kvalitativna akcija A je določena z dvojčkom $\langle P, SP \rangle$. P predstavlja predpogoje za izvedbo akcije, SP pa kvantitativne spremembe, ki jih akcija pri teh predpogojih lahko povzroči. SP je predstavljena kot matrika, določena skladno z enačbo 4.3. Matriko sprememb dobimo neposredno iz učnih primerov. Intervale za predpogoje P pa smo določili podobno kot razrede α in ϕ pri zbiranju učnih primerov (glej točko 7.2.1). Edina razlika je pri intervalu α, kjer smo vsako stranico predmeta z dolžino l centimetrov razdeli na $n' = \lfloor \frac{l}{4} \rfloor$ enakih delov.

8.3.2 Prva faza kvalitativnega planiranja

Splošne ugotovitve izvedbe prve faze kvalitativnega planiranja so predstavljene v točki 4.4.2. Za implementacijo splošnega postopka na konkretni domeni potiskanja predmetov moramo definirati preslikave numeričnih vrednosti v kvalitativne za vsako izmed spremenljivk in funkcijo, ki bo določila vrednosti uteži, uporabljenih v enačbi 4.5. V nadaljevanju so ločeno predstavljene preslikave za spremenljivke, ki nastopajo v stanju sistema, in spremenljivke, ki nastopajo v opisu akcij. Sledi še določitev funkcije za izračun uteži.

Stanje sistema

Trenutno stanje sistema opisuje vektor $S = (x_g, y_g, \theta_g, \alpha, \phi, seDotikata)$, želeno stanje pa vektor $G = (0, 0, 0, \_, \_, \_)$, kjer _ predstavlja poljubno vrednost spremenljivke. Pri kvalitativnem planiranju so posamezne spremenljivke, ki določajo stanje sistema, določene kvalitativno. Posvetimo se transformaciji numeričnih vrednosti v kvalitativne vrednosti za vsako skupino spremenljivk posebej:

- Spremenljivki x_g in y_g

  Za spremenljivki x_g in y_g uporabimo abstrakcijo števil v intervale. Preslikava je določena s Tabelo 8.2.

Numerična vrednost x_g oz. y_g	Kvalitativna vrednost x_g oz. y_g
$v \geq 2.0; v \in \{x_g, y_g\}$	1
$v \leq -2.0; v \in \{x_g, y_g\}$	-1
$v < 2.0 \wedge v > -2.0; v \in \{x_g, y_g\}$	0

Tabela 8.2: Preslikava numeričnih vrednosti v kvalitativne za spremenljivki x_g in y_g.

Numerična vrednost θ_g	Kvalitativna vrednost θ_g
$\theta_g \geq 5.0 \wedge \theta_g < 180.0$	1
$\theta_g \geq 180.0 \wedge \theta_g \leq 355.0$	-1
$\theta_g > 0.0 \wedge \theta_g < 5.0 \vee \theta_g > 355.0 \wedge \theta_g < 360.0$	0

Tabela 8.3: Preslikava numeričnih vrednosti v kvalitativne za spremenljivko θ_g.

- Spremenljivka θ_g

  Za vrednost θ_g uporabimo abstrakcijo števil v intervale. Abstrakcija je v tem primeru malenkost bolj zapletena, saj je kvantitativna vrednost določena po modulu 360. Preslikava je določena s Tabelo 8.3. Kvalitativna vrednost 1 pove, da je potrebno kot θ_g zmanjševati, medtem ko kvalitativna vrednost -1 pove, da je potrebno kot θ_g povečevati. Vrednost 0 pove, da je kot θ_g blizu želene vrednosti.

- Spremenljivke α, ϕ in *seDotikata*

  Za spremenljivke α, ϕ in *seDotikata* pretvorba v kvalitativne vrednosti ni potrebna, saj lahko v ciljnem stanju zavzamejo poljubno vrednost. To je razlog, da jih v prvi fazi kvalitativnega planiranja ne obravnavamo.

Ob tako opredeljenem stanju sistema se lahko bolj natančno posvetimo še kvalitativnim akcijam.

Akcije

Vsako akcijo A lahko na podlagi točke 9.1.2 definiramo kot $A = \langle(\alpha, \phi, seDotikata), (\Delta x_o, \Delta y_o, \Delta\theta_o, \alpha_K, \phi_K, seDotikata_K)\rangle$. Osnovna značilnost kvantitativnih akcij je, da so nekatere izmed spremenljivk določene kvalitativno. V nadaljevanju si bomo pogledali transformacijo numeričnih vrednosti v kvalitativne za vsako skupino spremenljivk.

Numerična vrednost Δx_o oz. Δy_o	Kvalitativna vrednost Δx_o oz. Δy_o
$v \geq 0.5; v \in \{\Delta x_o, \Delta y_o\}$	1
$v \leq -0.5; v \in \{\Delta x_o, \Delta y_o\}$	-1
$v < 0.5 \wedge v > -0.5; v \in \{x_g, y_g\}$	0

Tabela 8.4: Preslikava numeričnih vrednosti v kvalitativne za spremenljivki Δx_o in Δy_o.

Numerična vrednost $\Delta\theta_o$	Kvalitativna vrednost $\Delta\theta_o$
$\Delta\theta_o \geq 2.0$	1
$\Delta\theta_o \geq -2.0$	-1
$\Delta\theta_o < 2.0 \wedge \Delta\theta_o > -2.0$	0

Tabela 8.5: Preslikava numeričnih vrednosti v kvalitativne za spremenljivko $\Delta\theta_o$.

- Spremenljivke α, ϕ in *seDotikata*

  Spremenljivke, ki določajo predpogoje za izvedbo akcije, pustimo v kvantitativni obliki.

- Spremenljivke Δx_o, Δy_o, in $\Delta\theta$

  Za spremenljivkeΔx_o, Δy_o, in $\Delta\theta$ je smiselna abstrakcija v intervale. Za spremenljivki Δx_o in Δy_o smo jo implementirali kot prikazuje Tabela 8.4. Za spremenljivko $\Delta\theta_o$ pa abstrakcijo prikazuje Tabela 8.5.

- Spremenljivke α_K, ϕ_K in $seDotikata_K$

  Stanje robota po izvedbi kvalitativnih akcij je nedoločeno in hkrati nepomembno, saj spremenljivke α, ϕ in *seDotikata* v želenem stanju niso določene.

Določitev uteži

V točki 4.4.2 smo ugotovili, da imajo nekateri elementi stanja večjo pomembnost kot drugi. Stanje je definirano numerično s $S = (x_g, y_g, \theta_g, \alpha, \phi, seDotikata)$. Želeno stanje pa je določeno z $G = (0, 0, 0, \_, \_, \_)$. Elementom, katerih vrednost v želenem stanju ni pomembna, priredimo pomembnost 0. Za ostale elemente pa se pomembnost spreminja dinamično, v skladu z naslednjim postopkom:

1. Izračunamo vrednost funkcije h_g, ki je določena z enačbo 8.3.

$$h_g = |x_g| + |y_g| + min(\theta_g, 360 - \theta_g)/2 \tag{8.2}$$

2. Izračunamo vrednost uteži za vsako izmed spremenljivk $w_1 = \frac{|x_g|}{h_g}$, $w_2 = \frac{|y_g|}{h_g}$ in $w_3 = \frac{min(\theta_g, 360-\theta_g)}{2*h_g}$.

Celoten vektor uteži je tako določen z $W = [w_1, w_2, w_3, 0, 0, 0]$.

8.3.3 Druga faza kvalitativnega planiranja

Rezultat prve faze kvalitativnega planiranja je akcija A_Z, s katero pričnemo izvajanje potiskanja predmeta. Če je predpogoj za izvedbo akcije v trenutne stanju že izpolnjen, začnemo z izvedbo akcije takoj. V nasprotnem primeru pa najprej izvedemo akcijo za gibanje robota, ki vzpostavi predpogoje za izvedbo akcije. Izvedbo akcije pričnemo tako, kot določa vektor U.

V nadaljevanju je predstavljena prilagoditev druge faze kvalitativnega planiranja domeni potiskanja predmetov. Določiti moramo zgolj funkcijo za izračun koristnosti stanja in določiti pomembnost posameznih odvisnih spremenljivk.

Izračun koristnosti stanja

Koristnost trenutnega stanja izračunamo s pomočjo enačbe 8.3. Vrednosti vseh spremenljivk so določene numerično.

$$f = \sqrt{x_g^2 + y_g^2} + min(\theta_g, 360 - \theta_g)/2 \tag{8.3}$$

Izračun pomembnosti odvisnih spremenljivk

Odvisne spremenljivke pri domeni potiskanja predmetov so Δx, Δy, $\Delta\theta$, $\Delta\alpha$ in $\Delta\phi$. V nadaljevanju je predstavljeno, kako se izračuna pomembnost p za vsako izmed njih.

- Pomembnost spremenljivk Δx, Δy in $\Delta\theta$.

  Izračuna se vrednost funkcije 8.2. Sledi še izračun $p_{\Delta x} = \frac{|x_g|}{h_g}$, $p_{\Delta y} = \frac{|y_g|}{h_g}$ in $p_{\Delta\theta} = \frac{min(\theta_g, 360-\theta_g)}{h_g}$. Vrednosti predstavljajo pomembnost posameznega dela napake. Najbolj pomembna je spremenljivka z največjo napako.

- Ocenimo pomembnost spremenljivk α in ϕ.

  Pomembnost je določena na podlagi bližine nezaželenih stanj in bližine robov, ki so določeni v predpogojih P izvajane akcije. Za določanje pomembnosti posameznega robu iz predpogojev P se upošteva lastnosti akcije, do katere bi prišli v primeru kršenja omejitve. Bolj, ko je ta akcija primerna za zmanjševanje trenutne napake, manj pomemben je rob. V primeru, ko akcija povzroča enake kvalitativne spremembe kot trenutno izvajana akcija, robu sploh ne upoštevamo. Podroben postopek določitve pomembnosti je naslednji:

  1. pregledamo mejne vrednosti spremenljivk α in ϕ, ki omejujejo trenutno izvajano akcijo. Za akcijo velja $\alpha \in (\alpha_{min}, \alpha_{max}]$ in $\phi \in (\phi_{min}, \phi_{max}]$. Omejitve lahko določajo predpogoji P trenutno izvajane akcije ali nezaželena stanja;
  2. identificiramo množico $A_{sosednji}$ sosednjih akcij. V našem primeru so to:
     (a) akcija, če vrednost α pade pod α_{min};
     (b) akcija, če vrednost α zraste nad α_{max};
     (c) akcija, če vrednost ϕ pade pod ϕ_{min};
     (d) akcija, če vrednost ϕ zraste nad ϕ_{max};
  3. za vsako izmed akcij iz množice sosednjih akcij $A_{sosednji}$ pregledamo, kakšne povprečne kvalitativne spremembe Δx_o, Δy_o in $\Delta\theta_o$ povzročajo. Povprečno kvalitativno spremembo j-te sosednje akcije označimo s $\overline{SP_j}$. Te spremembe primerjamo z želenimi spremembami v trenutnem stanju (želene vrednosti so določene z nasprotno vrednostjo kvalitativno opredeljenega trenutnega stanja S). Nato z enačbo 8.4 določimo še vrednosti $k_{i,j}$, kjer je $i \in \Delta x, \Delta y, \Delta\theta$ in $j \in A_{sosednji}$;

     $$k_{i,j} = (sp_{j,i} - s_i) + 1 \tag{8.4}$$

  4. za tiste izmed omejitev, ki so določene z množico predpogojev P, izračunamo ustrezne izmed naslednjih točk:
     - $T_{\alpha_{min}} = (\alpha_{min}, \sum_i \quad p_{i*} k_{i,A_{\alpha_{min}}})$;
     - $T_{\alpha_{max}} = (\alpha_{max}, \sum_i \quad p_{i*} k_{i,A_{\alpha_{max}}})$;
     - $T_{\phi_{min}} = (\phi_{min}, \sum_i \quad p_{i*} k_{i,A_{\alpha_{min}}})$;
     - $T_{\alpha_{max}} = (\alpha_{max}, \sum_i \quad p_{i*} k_{i,A_{\phi_{max}}})$;

5. za tiste izmed omejitev, ki so določene z nezaželenimi stanji uporabimo ustrezne izmed naslednjih točk:
   - $T_{\alpha_{min}} = (\alpha_{min}, 5)$;
   - $T_{\alpha_{max}} = (\alpha_{max}, 5)$;
   - $T_{\phi_{min}} = (\phi_{min}, 5)$;
   - $T_{\phi_{max}} = (\phi_{max}, 5)$;
6. s točkami določimo funkciji $y_{\Delta\alpha}(\alpha) = a_1 * (\alpha - b_1)^2$ in $y_{\Delta\phi}(\phi) = a_2 * (\phi - b_2)^2$ (glej Sliko 8.1). Ker ima funkcija dva prosta parametra, ju lahko z dvema točkama enolično določimo. Pomembnost spremenljivke $\Delta\alpha$ je določena s $p_{\Delta\alpha} = y_{\Delta\alpha}(\alpha_{trenutni})$, kjer $\alpha_{trenutni}$ predstavlja trenutno vrednost kota α. Podobno je pomembnost spremenljivke $\Delta\phi$ določena s $p_{\Delta\phi} = y_{\Delta\phi}(\phi_{trenutni})$, kjer $\phi_{trenutni}$ predstavlja trenutno vrednost kota ϕ.

8.3.4 Ugotovitve

Kvalitativno planiranje uspešno odpravi pomanjkljivost, identificirane pri kvantitativnem planiranju:

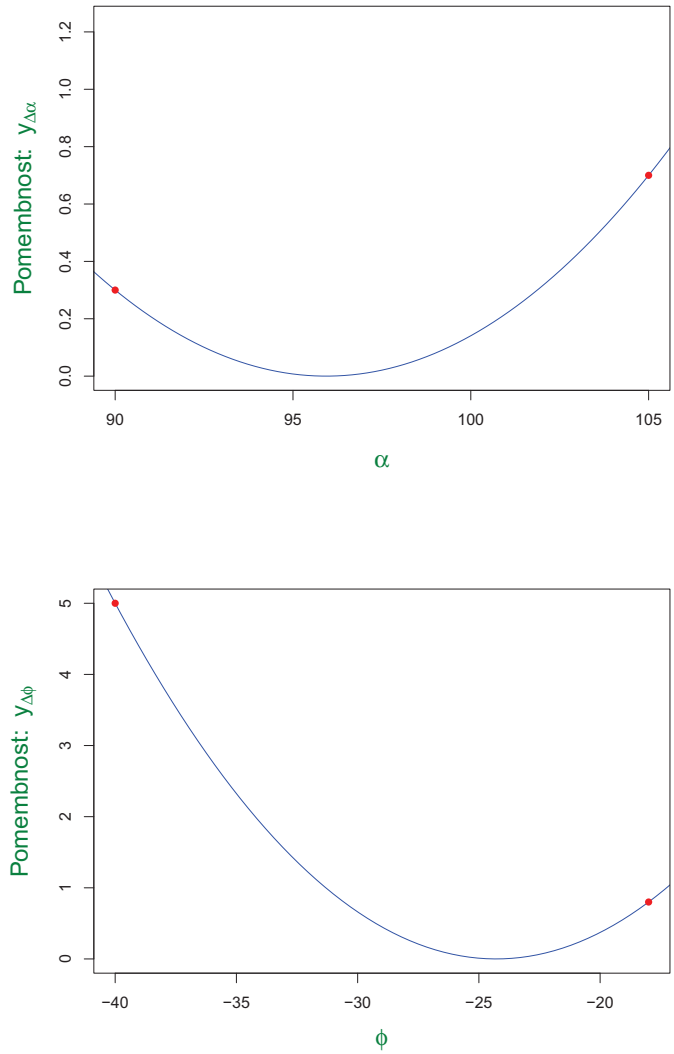
- kljub kompleksnosti problemske domene se kvalitativno planiranje izvrši v trenutku, saj je določitev najustreznejše akcije računsko preprosta;
- pri izvedbi nadaljnjih podakcij določenega plana se upošteva napaka, ki so jo povzročile predhodne podakcije. To izvedemo s pomočjo druge faze kvalitativnega planiranja.

8.4 Rezultati planiranja

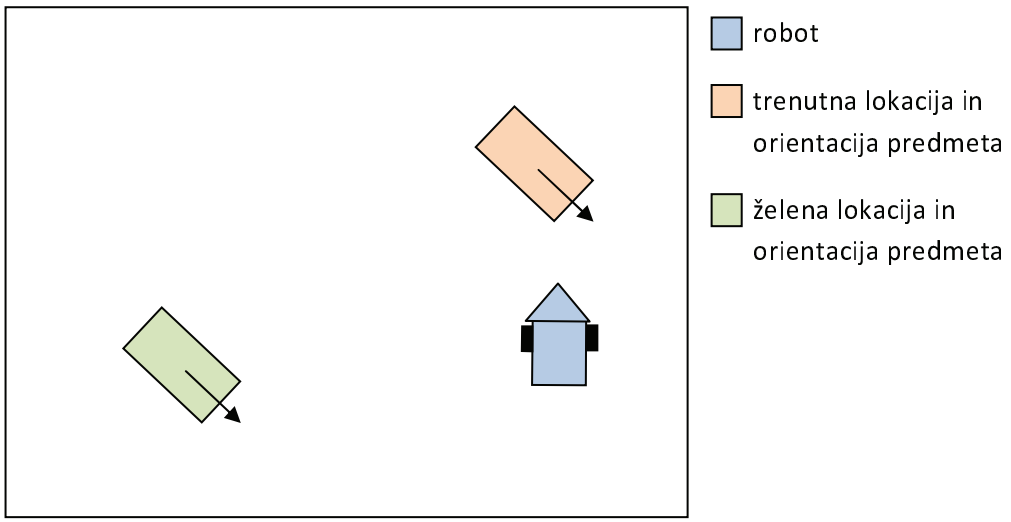
Predstavitev rezultatov planiranja je razdeljena na štiri dele. V prvem delu predstavimo način predstavitve rezultatov. V drugem delu predstavimo rezultate kvantitativnega planiranja in v tretjem rezultate kvalitativnega planiranja. V četrtem delu rezultate obeh načinov planiranja med seboj primerjamo.

8.4.1 Način predstavitve rezultatov

Rezultati so predstavljeni preko opazovanja izvedbe treh kar najbolj raznolikih nalog. Izbrane naloge so naslednje:



Slika 8.1: Primer funkciji $y_{\Delta\alpha}$ in $y_{\Delta\phi}$. $y_{\Delta\alpha}$ je določena s točkama $T_{\alpha_{min}} = (90, 0.3)$ in $T_{\alpha_{max}} = (105, 0.7)$. Podobno je $y_{\Delta\phi}$ določena s točkama $T_{\phi_{min}} = (-40, 5)$ in $T_{\phi_{max}} = (-18, 0.8)$. Točka $T_{\phi_{min}}$ je določena z nezaželenim stanjem.



Slika 8.2: Grafična predstavitev 1. naloge.

- 1. naloga

  Pri tej nalogi preizkusimo delovanje planiranja, ko se trenutna lokacija predmeta veliko razlikuje od želene lokacije. Trenutna orientacija pa je zelo podobna želeni. Nalogo prikazuje Slika 8.2.

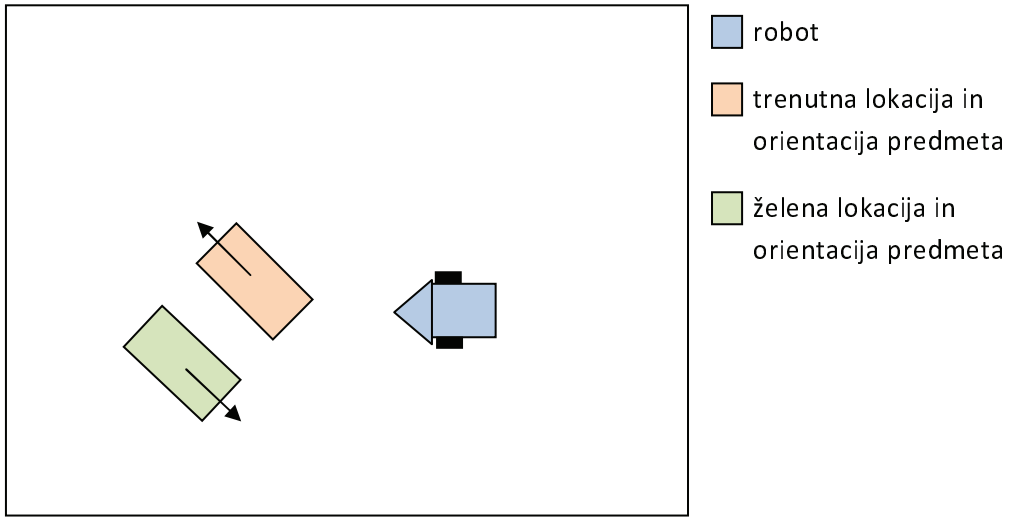
- 2. naloga

  Pri tej nalogi preizkusimo delovanje planiranja, ko se trenutna orientacija predmeta veliko razlikuje od želene orientacije. Trenutna lokacija pa je podobna želeni. Nalogo prikazuje Slika 8.3.

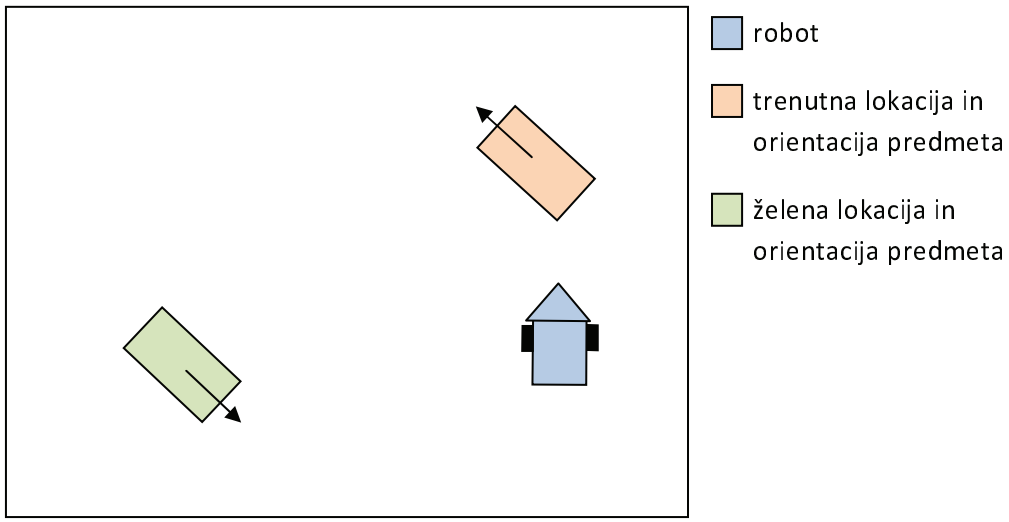
- 3. naloga

  Pri tej nalogi preizkusimo delovanje planiranja, ko se trenutna lokacija kot tudi trenutna orientacija predmeta veliko razlikujta od želene lokacije in orientacije predmeta. Nalogo prikazuje Slika 8.4.

Za vsako kombinacijo naloge in načina planiranja smo izvedli dve meritvi. Pri vsaki meritvi smo beležili razdaljo med težiščem trenutne in želene lokacije predmeta v času. Razdaljo smo poimenovali napaka lokacije. Izračunamo jo kot $\sqrt{x_g^2 + y_g^2}$. Poleg napake lokacije smo pri vsaki meritvi beležili še napako orientacije v času. Napako orientacije izračunamo kot $min(\theta_g, 360 - \theta_g)$ za $\theta_g \in [0, 360)$. Pove nam, za koliko se razlikujeta trenutna in želena orientacija predmeta. Ker je predstavitev z uporabo napake lokacije in napake orientacije



Slika 8.3: Grafična predstavitev 2. naloge.



Slika 8.4: Grafična predstavitev 3. naloge.

včasih zapletena, smo uvedli še napako, ki združuje obe količini. Določena je z enačbo 8.1, kjer je vrednost parametra $z = 1$.

Izvedba naloge je končana, če je napaka lokacije manjša od 5 cm in napaka orientacije manjša od 8^o. Pri uporabi manjših napak se povprečen čas izvedbe in standardni odklon časa izvedbe posamezne naloge povečata, saj mora robot navadno izvesti dodatne akcije, ki predmet natančno približajo želeni lokaciji. Izvedba vsake dodatne akcije traja povprečno 15 sekund. Število dodatnih akcij ni odvisno od uporabljenega načina planiranja. Uporaba manjših napak tako ne pripomore k razlikovanju uspešnosti posameznega načina planiranja. Hkrati pa bi zaradi povečanega standardnega odklona časa izvedbe naloge za korekten prikaz rezultatov potrebovali precej več kot zgolj dva poskusa za vsako nalogo. S tem bi predstavitev rezultatov zapletli, vendar pa ne bi prišli do nobenih pomembnejših ugotovitev, ki niso vidne tudi pri večjih vrednostih napak.

Nekateri izmed grafov v nadaljevanju prikazujejo skupaj tako napako lokacije kot tudi napako orientacije v času. Pri tem naj samo omenimo, da se zavedamo, da je uporaba dveh skal na istem grafu lahko zavajajoča, vendar pa je določene zakonitosti in ugotovitve (npr. izvedba balansiranja pri kvalitativnem planiranju) težko jasno predstaviti na drugačen način.

8.4.2 Rezultati kvantitativnega planiranja

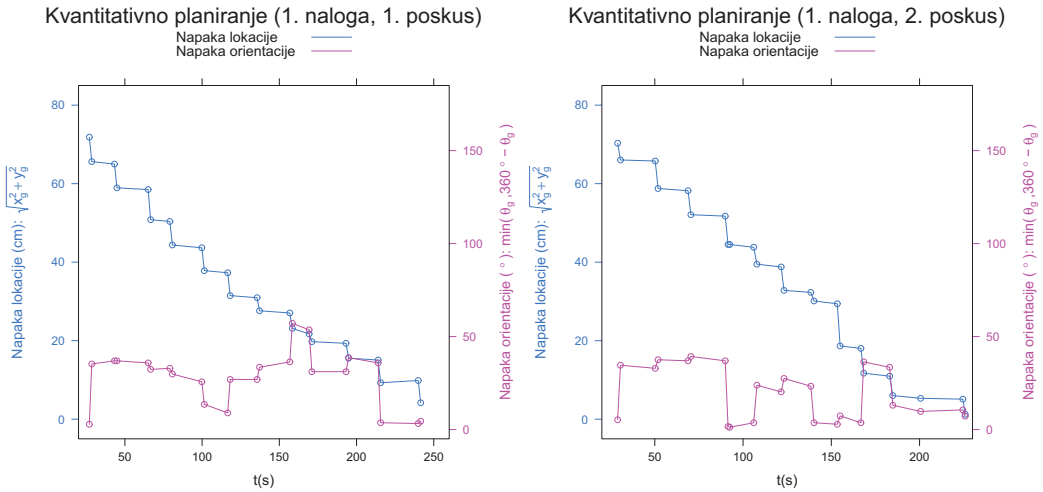
V tej točki so predstavljeni rezultati kvantitativnega planiranja. Slike 8.5, 8.6 in 8.7 grafično prikazujejo izvajanje vsake izmed nalog. Vsak graf prikazuje napako lokacije in napako orientacije v času.

Ugotovitve:

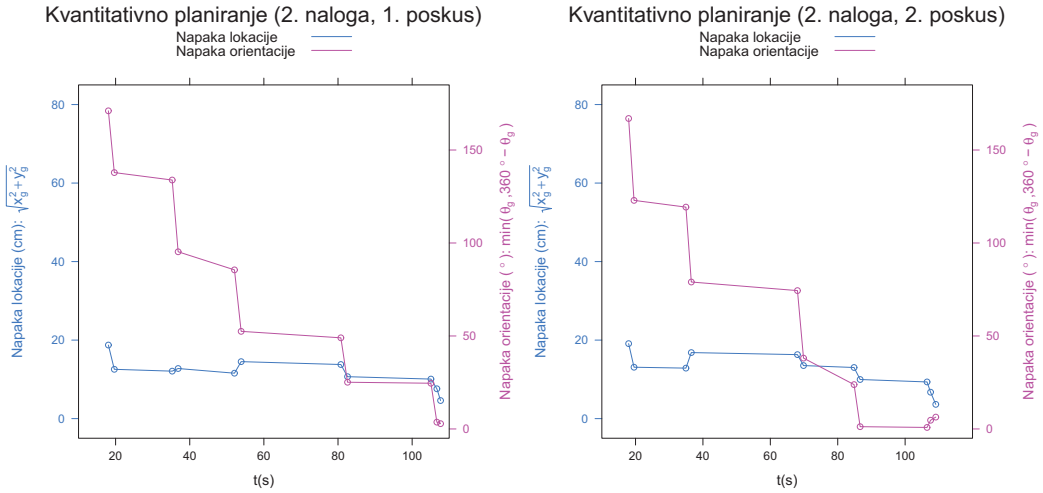
- vsi poskusi izvedbe nalog so končani;

- grafi so stopničasti. Glavni razlog za to je kompleksnost kvantitativnega planiranja, ki nam omogoča izračunavanje planov, ki vsebujejo le nekaj (navadno dva do štiri) akcije. Po izvedbi teh akcij se planiranje ponovi. Vsaka stopnička predstavlja izvedbo enega plana;

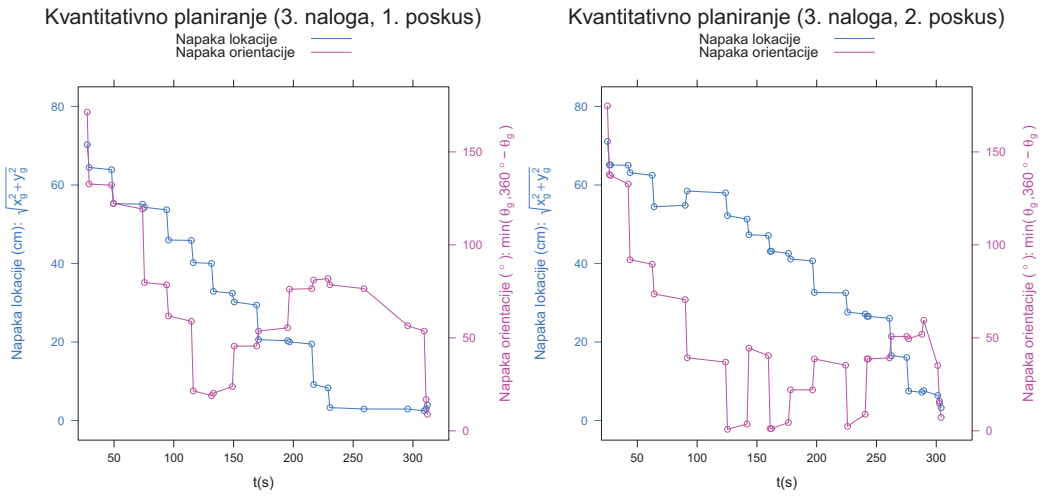
- grafi 8.11, 8.12 in 8.13 prikazujejo, da so izvedeni plani v veliki večini ustrezni (vrednost napake pada). Neustrezni plani se pojavijo zaradi napak pri zaznavanju lokacije predmeta in robota ter tako niso posledica napačnega planiranja.



Slika 8.5: Izvajanje 1. naloge z uporabo kvantitativnega planiranja.



Slika 8.6: Izvajanje 2. naloge z uporabo kvantitativnega planiranja.



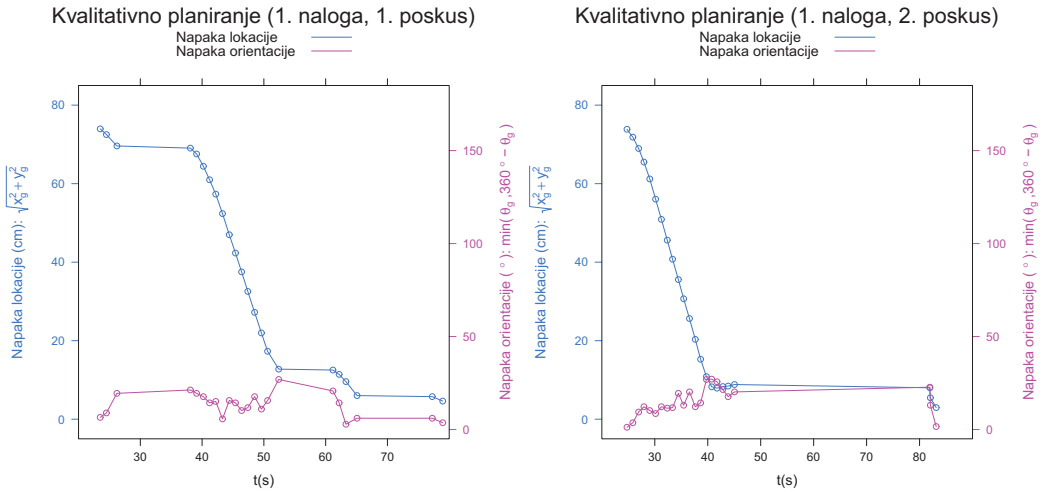
Slika 8.7: Izvajanje 3. naloge z uporabo kvantitativnega planiranja.

8.4.3 Rezultati kvalitativnega planiranja

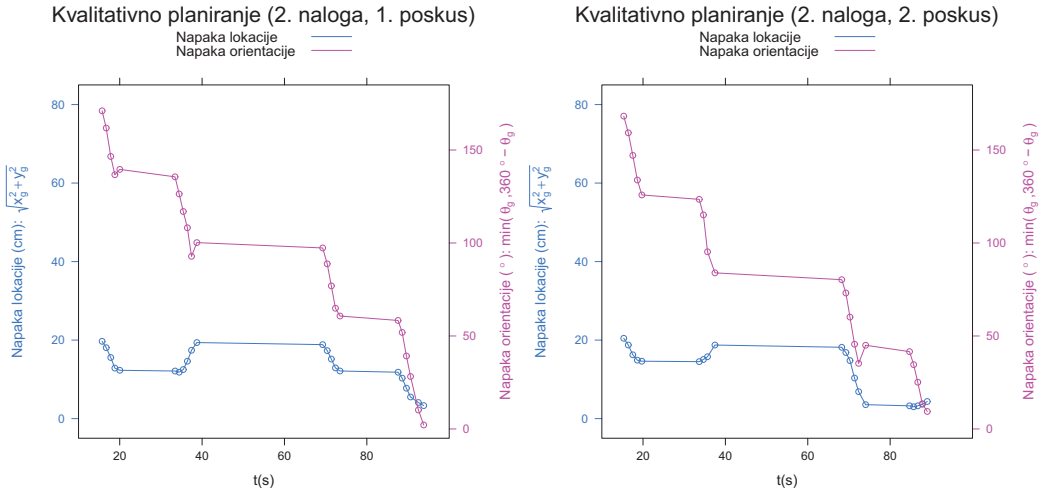
V tej točki so predstavljeni rezultati kvalitativnega planiranja. Slike 8.8, 8.9 in 8.10 grafično prikazujejo izvajanje vsake izmed nalog. Vsak graf prikazuje napako lokacije in napako orientacije v času.

Ugotovitve:

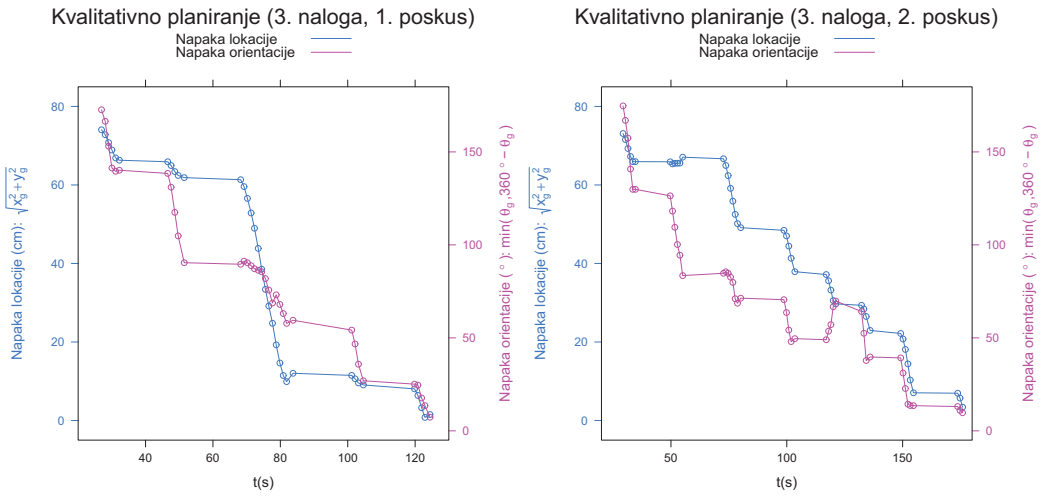
- vsi poskusi izvedbe nalog so končani;
- pri obeh poskusih izvedbe 1. naloge in pri 1. poskusu izvedbe 3. naloge je lepo viden postopek balansiranja predmeta (podobna dejavnost, kot jo izvaja človek, ki ima na prstu palico, za katero ne želi, da bi padla na tla). Pri balansiranju se napaka lokacije stalno zmanjšuje, odvod napake orientacije pa se hitro spreminja. Hitro spreminjanje odvoda napake orientacije je posledica izvajanja podakcij, ki omogočajo balansiranje. S pomočjo balansiranja lahko potiskanje predmeta brez prekinitev traja dlje časa. Pri 2. poskusu izvedbe 1. naloge izvajanje brez prekinitev traja 30 sekund. Napaka lokacije se v tem času zmanjša za 65 cm.
- grafi 8.11, 8.12 in 8.13 prikazujejo, da so izvedeni plani v veliki večini ustrezni (vrednost napake pada). Neustrezni plani se pojavijo zaradi napak pri zaznavanju lokacije predmeta in robota ter tako niso posledica napačnega planiranja. Takih napak je zanemarljivo malo.



Slika 8.8: Izvajanje 1. naloge z uporabo kvalitativnega planiranja.



Slika 8.9: Izvajanje 2. naloge z uporabo kvalitativnega planiranja.



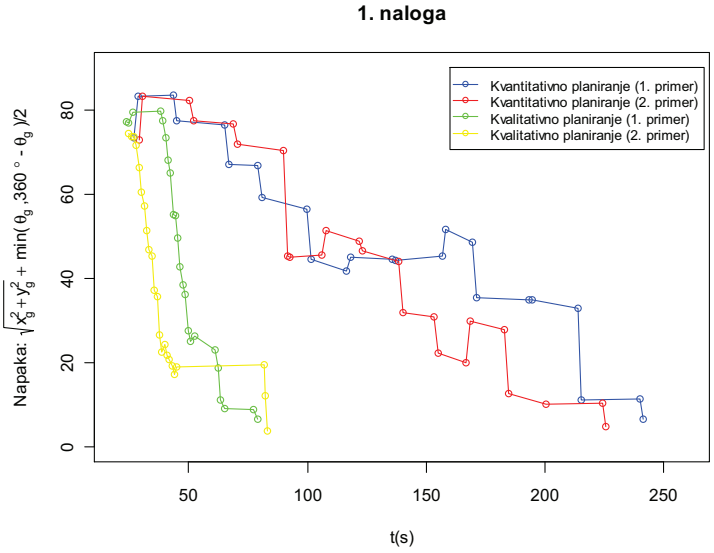
Slika 8.10: Izvajanje 3. naloge z uporabo kvalitativnega planiranja.

8.4.4 Primerjava rezultatov

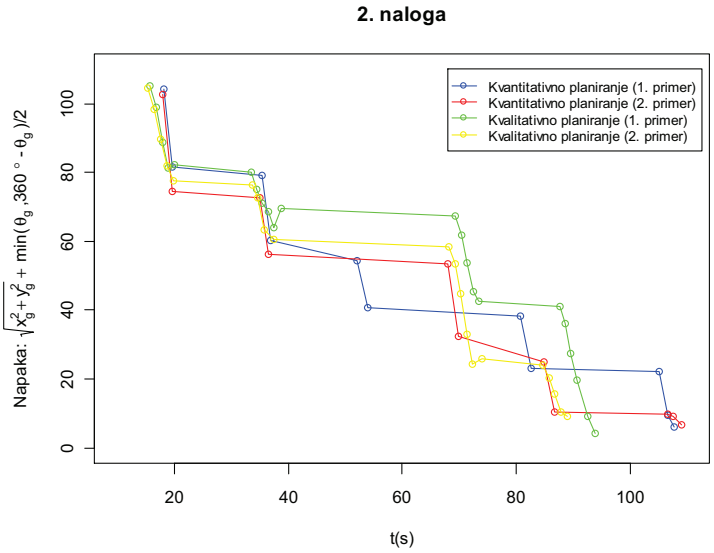
V tej točki so predstavljeni združeni rezultati kvantitativnega in kvalitativnega planiranja. Namesto predstavitve napake lokacije in napake orientacije smo se zaradi preglednosti odločili za združitev omenjenih količin v skupno napako. Grafi 8.11, 8.12 in 8.13 prikazujejo napako pri izvajanju vsake izmed nalog.

Ugotovitve:

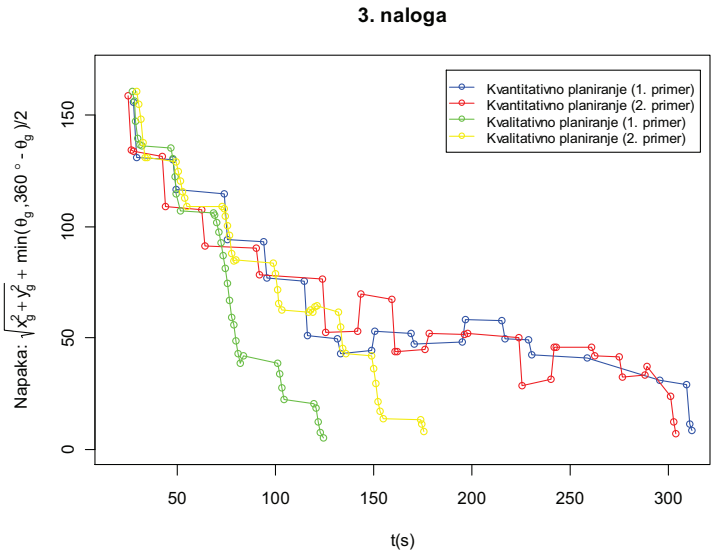
- kvalitativno planiranje je pri nalogah, ki vsebujejo veliko napako lokacije, izrazito učinkovitejše (z izvedbo naloge konča v krajšem času);
- pri nalogah, ki vsebujejo zgolj napako orientacije, je kvalitativno planiranje učinkovitejše, vendar je razlika majhna;
- pri kvalitativnem planiranju je manj neustreznih planov. Zaradi zmožnosti balansiranja se pomen napake pri določevanju lokacije robota in predmeta zmanjša, saj lahko nekatere sprva nekoliko neustrezne plane s pravilnimi podakcijami spremenimo v ustrezne.



Slika 8.11: Primerjava izvajanj 1. naloge.



Slika 8.12: Primerjava izvajanj 2. naloge.



Slika 8.13: Primerjava izvajanj 3. naloge.

Poglavje 9

Zaključki in nadaljnje delo

9.1 Zaključki

V preteklih poglavjih diplomskega dela smo si najprej pogledali teoretično ozadje algoritmov in pristopov, uporabljenih za realizacijo učenja z eksperimentiranjem pri uporabi avtonomnega robota. Pričeli smo s Šucevim algoritmom QUIN, ki ga uporabljamo za izdelavo kvalitativnih modelov. V nadaljevanju smo se posvetili kvalitativnemu zvestemu učenju in znanemu algoritmu Qfilter, ki se uporablja za izboljšavo numeričnih napovedi z upoštevanjem kvalitativnih modelov.

Kot zadnje v teoretičnem delu diplomskega dela si pogledamo še problem planiranja. Tam smo najprej predstavili pogosto uporabljeni jezik STRIPS, ustrezne razširitve jezika STRIPS in mehko planiranje. Poglavje smo zaključili s splošnim opisom kvalitativnega planiranja, ki predstavlja novost za reševanje nekaterih problemov planiranja. Pri tem ugotovimo, da je "uporaba kvalitativnega planiranja smiselna za vse numerično opredeljene probleme mehkega planiranja, za katere lahko izdelamo kvalitativni model in katerih rešitve želimo udejanjiti z uporabo določenega robota". Ideje o kvalitativnem planiranju v literaturi nismo zasledili, zato jo štejemo kot najpomembnejšo ugotovitev tega diplomskega dela.

Diplomsko delo nadaljujemo s podrobnim opisom implementacije predstavljenih algoritmov na domeni potiskanja predmetov. Opis implementacije pričnemo s predstavitvijo problemske domene in nadaljujemo s formalno opredelitvijo problema. Nadaljujemo s predstavitvijo algoritmov in pristopov, uporabljenih za upravljanje gibanja robota. Algoritme A*, PID kontroler in Pure Pursuit smo implementirali sami.

Sledi opis učenja modela potiskanja predmetov. Predstavili smo našo

strategijo zbiranja učnih primerov in rezultate kvalitativnega ter kvantitativnega učenja. Za izračun kvalitativnih rezultatov smo uporabili algoritem QUIN, za izračun kvantitativnih rezultatov pa algoritem Qfilter. Rezultati kvalitativnega učenja so intuitivno preprosto razumljivi. Omogočijo nam razumljiv vpogled v zakonitosti, ki veljajo v domeni potiskanja predmetov. Pravilnost kvantitativnih in kvalitativnih rezultatov smo preverili kasneje, z uporabo planiranja.

S planiranjem tudi nadaljujemo predstavitev diplomskega dela. Planiranje smo implementirali na dva načina. Prvi je kvalitativno planiranje, drugega pa smo poimenovali kvantitativno planiranje. Oba smo implementirali sami. Podrobnosti implementacije tudi predstavimo. Sledi predstavitev rezultatov obeh načinov planiranja. Vsak način preizkusimo pri izvedbi treh kar najbolj različnih nalog. Rezultate med seboj tudi primerjamo. Pri tem ugotovimo, da je kvalitativno planiranje izrazito uspešnejše. Z uporabo kvalitativnega planiranja smo uspešno dosegli celo izvedbo balansiranja pri potiskanju predmeta (podobno delo, kot ga izvajamo, ko imamo na prstu daljšo palico in želimo, da nam ne pade na tla). V dokaz uspešnosti kvalitativnega planiranja bralca tudi povabimo, da si ogleda videoposnetke robotovega izvajanja nekaterih nalog.[1]

9.2 Napotki in izkušnje pri uporabi programa QUIN

Pri izdelavi diplomskega dela smo uporabili program QUIN. Na tem mestu bomo povzeli naše izkušnje pri njegovi uporabi.

Kot eno izmed negativnih lastnosti programa QUIN smo navedli časovno zahtevnost. Tekom izdelave diplomskega dela smo opazovali zgolj časovno zahtevnost v odvisnosti od števila učnih primerov. Pri tem smo izvedli nekaj meritev, ki jih prikazuje Tabela 9.1. Preprosta analiza nam pokaže, da je časovna zahtevnost v odvisnosti od števila učnih primerov blizu kvadratni. Za natančnejše ugotovitve bi se morali analizi časovne zahtevnosti podrobneje posvetiti, kar pa ni bil cilj tega diplomskega dela.

Kvalitativni modeli, ki jih je induciral QUIN, so se nam zdeli zelo dobri, saj so ponudili preprost in točen vpogled v dogajanje v domeni potiskanja predmetov. Zbiranje učnih primerov smo izvedli večkrat. Pri tem so si bila inducirana kvalitativna drevesa precej podobna. Razlik v induciranih kvalitativno omejenih funkcijah skoraj ni bilo, nekoliko pa so se med seboj razlikovale

[1] Dostopno na: `http://www.youtube.com/watch?v=3xwwoIEDoQo`.

Število učnih primerov	Povprečen čas izvedbe (s)
108	4
216	58
432	503
648	1491

Tabela 9.1: Časovna zahtevnost algoritma QUIN v odvisnosti od števila učnih primerov. Tabela prikazuje povprečen čas izvedbe na podlagi petih poskusov.

delitve. Nekatere delitve so bile preprosto, druge pa nekoliko težje, intuitivno razumljive.

Število učnih primerov po presegu določene kritične meje (v primeru tega diplomskega dela je bila določena s parametrom $k = 2$ - glej točko 7.2.1) na inducirana kvalitativna drevesa ni veliko vplivalo.

Parameter, ki ocenjuje napako posameznih učnih primerov, ima na inducirana kvalitativna drevesa precejšen vpliv. Na podlagi poznavanja problemske domene lahko njegovo vrednost dokaj natančno določimo z razmislekom. Okrog te vrednosti je navadno potrebno nekaj poskušanja, da pridemo do najlažje intuitivno razumljivega kvalitativnega drevesa.

9.3 Nadaljnje delo

Tekom izdelave diplomskega dela smo identificirali tri možne smernice za nadaljnje delo. Prva spada na področje izdelave kvalitativnih modelov, druga in tretja pa na področje kvalitativnega planiranja.

Program QUIN je neuporaben za večje količine podatkov, saj kvalitativnih modelov ni zmožen inducirati v sprejemljivem času. Na področju učenja kvalitativnih modelov na domeni potiskanja predmetov je mogoče ta problem nekoliko omejiti z upoštevanjem simetrije predmeta. V okviru tega diplomskega dela smo simetrijo predmeta upoštevali zgolj pri zbiranju učnih primerov. Smiselno bi bilo ugotoviti, kako lahko simetrijo upoštevamo tudi pri izdelavi kvalitativnih modelov. Pri tem bi z uporabo algoritma QUIN izdelali kvalitativni model zgolj za določen del predmeta. Na podlagi tega modela bi znali sklepati na kvalitativni model celotnega predmeta.

Pri konkretni uporabi kvalitativnega planiranja moramo empirično določiti vrednosti vektorjev U in V. Pri tem vektor V nima človeku preprosto razumljivega pomena. Določevanje takih parametrov je lahko včasih zapleteno, zato se nam zdi za njihovo določitev smiselna uporaba strojnega učenja. Na

ta način bi lahko konkretne vrednosti vektorjev U in V določili avtomatsko.

Kvalitativno planiranje smo v diplomskem delu opredelili splošno. Ustreznost smo preverili zgolj na eni problemski domeni z enim robotom. Za dokaz splošne uporabnosti algoritma bi ga bilo potrebno preizkusiti tudi na drugih problemskih domenah z različnimi roboti.

Slike

2.1 Primer kvalitativnega drevesa. 15

3.1 Postopek izboljšave numeričnih napovedi. 19
3.2 Prikaz delovanja algoritma Qfilter. 21

4.1 Primer akcij pri mehkem planiranju. 28

5.1 Prikaz poligona z robotom. 40
5.2 Robot. 41
5.3 Stanje robota. 42
5.4 Model gibanja robota. 43
5.5 Stanje predmeta. 44

6.1 Pure Pursuit. 47
6.2 Napaka pri PID kontrolerju. 48
6.3 Neusmerjen graf mogočih poti. 50

7.1 Transformacija koordinatnega sistema xy v koordinatni sistem x'y'. 55
7.2 Neodvisne učne spremenljivke. 56
7.3 Primer razredov za spremenljivko α. 58
7.4 Uporaba simetrije za skrajšanje časa učenja. 60
7.5 Prikaz množice točk, ki so si zaradi simetrije ekvivalentne. . . . 61
7.6 Dimenzije predmeta. 62
7.7 Uporaba znanja o problemski domeni za poenostavitev učenja. . 63
7.8 Prikaz kvalitativnega drevesa za spremenljivko $\Delta\theta_o$. 66
7.9 Prikaz kvalitativnega drevesa za spremenljivko Δx_o. 66
7.10 Prikaz kvalitativnega drevesa za spremenljivko Δy_o. 67
7.11 Prikaz kvalitativnega drevesa za spremenljivko $\Delta\alpha$. 68
7.12 Prikaz kvalitativnega drevesa za spremenljivko $\Delta\phi$. 68

8.1 Primer funkciji $y_{\Delta\alpha}$ in $y_{\Delta\phi}$. 85
8.2 Grafična predstavitev 1. naloge. 86
8.3 Grafična predstavitev 2. naloge. 87
8.4 Grafična predstavitev 3. naloge. 87
8.5 Izvajanje 1. naloge z uporabo kvantitativnega planiranja. 89
8.6 Izvajanje 2. naloge z uporabo kvantitativnega planiranja. 89
8.7 Izvajanje 3. naloge z uporabo kvantitativnega planiranja. 90
8.8 Izvajanje 1. naloge z uporabo kvalitativnega planiranja. 91
8.9 Izvajanje 2. naloge z uporabo kvalitativnega planiranja. 91
8.10 Izvajanje 3. naloge z uporabo kvalitativnega planiranja. 92
8.11 Primerjava izvajanj 1. naloge. 93
8.12 Primerjava izvajanj 2. naloge. 93
8.13 Primerjava izvajanj 3. naloge. 94

Tabele

2.1 Predstavitev kvalitativnih vrednosti s števili. 10
2.2 Prikaz vrednosti kvalitativno omejene funkcije 12

7.1 Določitev napake. 69
7.2 Primer rezultatov numeričnega učenja. 70
7.3 Primer popravljenih napovedi. 70

8.1 Primer določitve elementov kvantitativne akcije. 74
8.2 Preslikava numeričnih vrednosti v kvalitativne za spremenljivki x_g in y_g. 80
8.3 Preslikava numeričnih vrednosti v kvalitativne za spremenljivko θ_g. 80
8.4 Preslikava numeričnih vrednosti v kvalitativne za spremenljivki Δx_o in Δy_o. 81
8.5 Preslikava numeričnih vrednosti v kvalitativne za spremenljivko $\Delta\theta_o$. 81

9.1 Časovna zahtevnost algoritma QUIN v odvisnosti od števila učnih primerov. 97

Literatura

[1] XPERO. Dostopno na: http://www.xpero.org

[2] D. Šuc, *Machine reconstruction of human control strategies : doctoral dissertation*, Ljubljana, 2001.

[3] D. Šuc, *Machine Reconstruction of Human Control Strategies, volume 99 of Frontiers in Artificial Intelligence and Applications*, Nizozemska: IOS Press, 2003.

[4] D. Šuc in I. Bratko, "Improving numerical prediction with qualitative constraints", v *Machine learning / 12th European Conference on Machine Learning*, Cavtat-Dubrovnik, Hrvaška, sept. 2003, str. 385 - 396.

[5] D. Šuc in I. Bratko, "Combining learning constraints and numerical regression", v *International Joint Conference on Artificial Intelligence*, Edinburgh, Škotska, avg. 2005, str. 596 - 602.

[6] I. Bratko, *Prolog Programming for Artificial Intelligence*, 3. izd., Addison Wesley, 2001, pogl. 17, 19 in 20.

[7] D. Šuc in I. Bratko, "Induction of Qualitative trees", v *Machine learning / 12th European Conference on Machine Learning*, Freiburg, Nemčija, sept. 2001, str. 442 - 453.

[8] H. Geffner, "PDDL 2.1: Representation vs. Computation", v *Journal of Artificial Intelligence*, dec. 2003, str. 139 - 144.

[9] J. Rintanen in H. Jungholt, "Numeric State Variables in Constraint-Based Planning", v *Proceedings of ECP-99*, 1999, str. 109 - 121.

[10] P. Haslum in H. Geffner, "Heuristic Planning with Time and Resources", v *Proceedings of ECP-01*, 2001, str. 682 - 696.

[11] J. Blythe, "An Overview of Planning Under Uncertainty", v *AI Magazine*, 1999, str 37 - 54.

[12] Multiple regression. Dostopno na: http://www.statsoft.com/textbook/multiple-regression/

[13] T. M. Mitchell, *Machine Learning*, 1. izd., McGraw-Hill Science, 1997, pogl. 8.

[14] I. N. Bronštejn, K. A. Semendjajev, G. Musiol in H. Muhlig, *Matematični priročnik*, 2. izd., Tehniška založba Slovenije, 1997, str. 159.

[15] J. Strnad, *Fizika 1. del: Mehanika, Toplota*, 1. izd., Ljubljana: DZS, 1977, pogl. 3.

[16] B. Orel, *Osnove numerične matematike*, 2. izd., Ljubljana: Fakulteta za računalništvo in informatiko, 1999.

[17] Smoothing algorithm using Bezier Curves. Dostopno na: http://www.efg2.com/Lab/Graphics/Jean-YvesQueinecBezierCurves.htm

[18] J. Morales, J. L. Martinez, M. A. Martinez in A. Mandow. (2009, jan.). Pure-Pursuit Reactive Path Tracking for Nonholonomic Mobile Robots with a 2D Laser Scanner, *EURASIP Journal on Advances in Signal Processing*, št. 2009. Dostopno na: http://www.hindawi.com/journals/asp/2009/935237.html

[19] (2006) AVR221: Discrete PID controller. Dostopno na: http://www.atmel.com/dyn/resources/prod_documents/doc2558.pdf

[20] PID tutorial. Dostopno na: http://www.jashaw.com/pid/tutorial/

[21] Algorithms and Sensors for Small Robot Path Following. Dostopno na: http://www-users.cs.umn.edu/~stergios/papers/PathFollowingICRA02.pdf

[22] B. Vilfan, *Osnovni algoritmi*, 2. izd., Ljubljana: Fakulteta za računalništvo in informatiko, 2002, pogl. 6.

[23] C. Thornton, B. Boulay, *Artificial Intelligence - Strategies, Applications, and Models Through Search*, 2. izd., AMACOM, 1998, pogl. 4.

[24] Lego Mindstorms. Dostopno na: http://mindstorms.lego.com/en-us/default.aspx

[25] Color Machine Vision. Dotopno na: http://www.cs.cmu.edu/ jbruce/cmvision/

[26] (1998) The minimum distance between two convex polygons. Dostopno na: http://cgm.cs.mcgill.ca/ orm/mind2p.html

[27] (2010) Algoritma QUIN in Qfilter z uporabniškim grafičnim vmesnikom. Dostopno na: http://www.ailab.si/dorian/q2/quin.htm

[28] (2001) Učenje kvalitativnih dreves. Dostopno na: http://videolectures.net/solomon_suc_kzuui/