SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

POSLIJEDIPLOMSKI DOKTORSKI STUDIJ ELEKTROTEHNIKE I INFORMACIJSKE TEHNOLOGIJE

KVALIFIKACIJSKI DOKTORSKI ISPIT

Autonomni mobilni roboti

STANKO KRUŽIĆ

Split, srpanj 2018.

Sadržaj

1. Uvod

Robot se može definirati kao stroj koji je sposoban izvršiti određeni niz akcija automatski, a izrađuju se s ciljem da obavljaju neke zadatke umjesto čovjeka. Općenito roboti se mogu podijeliti u više kategorija: mobilni roboti, industrijski roboti i manipulatori, edukacijski roboti, humanoidni roboti, itd. Mobilni roboti su, najopćenitije, roboti koji imaju sposobnost kretanja u prostoru, što uključuje kretanje po tlu, vodi i zraku, te kroz vodu. Mogu se dalje podijeliti na mobilne robote za čvrste podloge, robotska plovila, ronilice i bespilotne letjelice. Iako se u literaturi pod pojmom "mobilni robot" najčešće smatra mobilni robot za čvrste podloge, dok se ostale nabrojane vrste nazivaju njihovim pripadajućim nazivima. Za razliku od industrijskih robota (manipulatora) čije je kretanje ograničeno na usko definirani radni prostor, mobilni roboti se mogu slobodno kretati u većem prostoru, što ih čini pogodnim za razne primjene u strukturiranim i nestrukturiranim okolinama.

Fokus ovog rada je na autonomnim mobilnim robotima. Autonomija podrazumijeva snalaženje u nepoznatom okolišu te samostalno obavljanje određenih zadataka bez intervencije čovjeka. U današnje vrijeme, autonomni mobilni roboti postaju sveprisutni, te se razvijaju različiti inovativni scenariji njihove primjene. Posebna podvrsta mobilnih robota su tzv. servisni roboti kojima je cilj da zamjene čovjeka u obavljanju zadataka koji su opasni, štetni za zdravlje, ponavljajući ali i nepopularni (uključujući i kućanske poslove). Takve robote se već danas može naći u kućanstvima (robotski usisavači), različitim ustanovama i/ili poduzećima (autonomni perači podova, autonomne kosilice trave) te skladištima (autonomni paletari). Saznanja iz područja mobilne robotike se primjenjuju i u automobilskoj industriji, a koriste se ponajviše u razvoju autonomnih automobila.

U radu će se dati pregled područja mobilnih robota i njihovih primjena. U poglavlju 2 će biti dan uvod u mobilne robote, načine zapisa pozicije i orijentacije robota, njihovi modeli i algoritmi za upravljanje robotima. Autonomni roboti trebaju prikupljati informacije o sebi i okolini, pa će stoga biti dana podjela, pregled i opis principa funkcioniranja najčešće korištenih senzora u robotici. Lokalizacija robota u prostoru, bilo poznatom ili nepoznatom, od velike je važnosti za autonomnu navigaciju robota. Poznavanje precizne lokacije robota omogućava efikasno planiranje putanje te sigurnu navigaciju kroz prostor. Poglavlje 3 daje pregled algoritama za lokalizaciju i navigaciju robota te za izradu mapa prostora, te algoritama za planiranje putanje kroz poznatu okolinu. Da bi mobilni robot bio sposoban za autonoman rad u realnoj okolini, koja uključuje prepreke čiji položaji nisu unaprijed poznati i ne moraju biti nepomične, mora biti opremljen kvalitetnim algoritmima za izbjegavanje prepreka, kako bi se izbjegla šteta za robota i njegovu okolinu. Prepreke možemo podijeliti na statičke (kao npr. zidovi) i dinamičke (kao npr. ljudi ili neki drugi pomični objekti). U poglavlju 4 je dan pregled algoritama za detekciju i izbjegavanje statičkih i dinamičkih prepreka.

Mobilni roboti u autonomnom način rada, ovisno o zadatku, mogu koristiti i neke od metoda umjetne inteligencije. Poglavlje 5 donosi pregled metoda umjetne inteligencije koje se koriste u robotici, te koje uključuju različite klase neuronskih mreža i *reinforcement learning*. Predstavljena je inteligentna navigacija u nepoznatoj okolini, te je posebno dan naglasak na

pristupe inteligentnoj navigaciji koji su inspirirani navigacijom bioloških entiteta. Ovakvi pristupi se koriste kako bi se ostvario određeni vid ponašanja robota koje je što je moguće sličniji ponašanju živih organizama. Ponekad kod mobilnih robota u režimu autonomnog upravljanja, zbog nepredvidivosti i dinamičnosti okoline dogodi da algoritmi za autonomni rad robota zakažu. U tom slučaju, da bi se pomoglo robotu, potrebno imati mogućnost upravljanja robotom s udaljene lokacije. Poglavlju 6 daje pregled mogućnosti upravljanja mobilnim robotima s udaljene lokacije, interakcija čovjeka-operatora i robota, te analizira problem upravljanja s udaljene lokacije uz prisutnost latencije, koja je u ovakvim slučajevima neizbježna.

2. Mobilni roboti

Iako postoji više vrsta mobilnih robota, u radu će se promatrati samo mobilni roboti s kotačima za kretanje po čvrstim podlogama. Od ostalih vrsta mogu se izdvojiti mobilni roboti s nogama (npr. humanoidni roboti), bespilotne letjelice, ronilice i slično.

2.1. Oblici zapisa pozicije i orijentacije robota

Robot, kojeg promatramo kao kruto tijelo na kotačima, kreće se po ravnoj čvrstoj podlozi. Definira se i broj stupnjeva slobode, koji je jednak broju nezavisnih načina gibanja. Robot koji se kreće po čvrstoj podlozi, općenito se nalazi u ravnini na poziciji (x, y), te ima orijentaciju s obzirom na vertikalnu os θ. Broj stupnjeva slobode takvog robota je tri, iako može biti i manji ako nekom od ove tri veličine nije moguće direktno upravljati. Dodatni stupnjevi slobode mogu postojati ako se promatraju i interni dijelovi robota, kao što su kotači pojedinačno, pasivni kotačići, osi kotača i slično. Međutim, ako se promatra gibanje robota u cjelini kao krutog tijela, broj stupnjeva slobode je tri ili manji, kako je prethodno opisano.

Da bi se opisala pozicija robota u ravnini, uvode se pojmovi referentnog koordinatnog sustava i koordinatnog sustava robota. Referentni koordinatni sustav je nepomičan i vezan je za neku točku u prostoru, dok je koordinatni sustav robota vezan za robota i pomiče se zajedno s njim što za posljedicu ima da se robot, ako ga promatramo u njegovom vlastitom koordinatnom sustavu, nikada ne miče. Stoga, da bismo mogli odrediti položaj robota u odnosu na referentni koordinatni sustav je potrebno definirati transformacije između koordinatnih sustava.

Stanje svakog robota se opisuje njegovom konfiguracijom $\boldsymbol{q}$. Kada se govori o konfiguraciji mobilnog robota, tu podrazumjevamo poziciju i orijentaciju robota, koje se u referentnom koordinatnom sustavu može zapisati kao vektor s tri elementa, koordinatama x i y, koje predstavljaju udaljenost ishodišta koordinatnog sustava robota od ishodišta referentnog koordinatnog sustava u smjeru pripadajućih osi, i kutom θ koji predstavlja orijentaciju koordinatnog sustava robota u odnosu na referentni koordinatni sustav, tj. pokazuje koliko je koordinatni sustav robota zakrenut u odnosu na referentni koordinatni sustav:

$$\boldsymbol{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.1}$$

Orijentaciju nekog koordinatnog sustava u odnosu na drugim referentni koordinatni sustav može se opisati rotacijskom matricom, Eulerovim kutevima, te kvaternionima.

2.1.1. Rotacijska matrica

Rotacijska matrica se dobiva skalarnim produktima jediničnih vektora koji definiraju koordinatne osi promatranih sustava $\{i\}$ i $\{j\}$:

$$ {}^{j}\boldsymbol{R}_{i} = \begin{bmatrix} \hat{x}_j \cdot \hat{x}_i & \hat{y}_j \cdot \hat{x}_i & \hat{z}_j \cdot \hat{x}_i \\ \hat{x}_j \cdot \hat{y}_i & \hat{y}_j \cdot \hat{y}_i & \hat{z}_j \cdot \hat{y}_i \\ \hat{x}_j \cdot \hat{z}_i & \hat{y}_j \cdot \hat{z}_i & \hat{z}_j \cdot \hat{z}_i \end{bmatrix} \tag{2.2} $$

Kako je kod mobilnih robota jedina promatrana rotacija oko vertikalne osi robota, rotacijska matrica oko te osi je

$$ \boldsymbol{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

Transformaciju koja uključuje translaciju i rotaciju između dva koordinatna sustava ($\{i\}$ i $\{j\}$) se opisuje homogenom matricom transformacije (koja se može smatrati proširenjem rotacijske matrice tako da uključuje i translaciju)

$$ {}^{j}\boldsymbol{T}_{i} = \begin{bmatrix} {}^{j}\boldsymbol{R}_{i} & {}^{j}\boldsymbol{t}_{i} \\ \boldsymbol{0}^{T} & 1 \end{bmatrix} \tag{2.4} $$

gdje je vektor $\boldsymbol{t}$ pozicija koordinatnog sustava. Ovakav zapis se može koristiti za prikaz orijentacije i pozicije robota u odnosu na referentni koordinatni sustav.

Matrica rotacije i matrica homogene transformacije su ortonormalne, tj. za njih vrijedi

$$ {}^{j}\boldsymbol{R}_{i} = {}^{i}\boldsymbol{R}_{j}^{-1} = {}^{i}\boldsymbol{R}_{j}^{\top} \tag{2.5} $$

$$ {}^{j}\boldsymbol{T}_{i} = {}^{i}\boldsymbol{T}_{j}^{-1} = {}^{i}\boldsymbol{T}_{j}^{\top} \tag{2.6} $$

2.1.2. Eulerovi kutevi

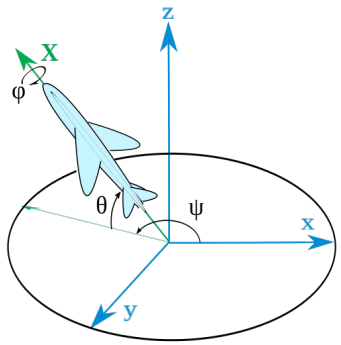
Eulerovim kutevima se opisuje orijentacija krutog tijela u odnosu na neki drugi, referentni koordinatni sustav. Svaku orijentaciju se može postići nizom elementarnih rotacija[1]. Eulerovi kutevi se mogu definirati kroz tri takve rotacije, napravljene po određenom redoslijedu. Te elementarne rotacije mogu biti ekstrinsične (rotacije oko *xyz* osi referentnog, nepomičnog koordinatnog sustava) i intrinsične (rotacije oko *XYZ* osi pomičnog sustava – krutog tijela). Postoji 12 nizova rotacija podijeljenih u dvije grupe:

- pravi Eulerovi kutevi: z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y
- Tait-Bryan kutevi: x-y-z, y-z-x, z-x-y, x-z-y, **z-y-x**, y-x-z

Najčešće korišten niz elementarnih rotacija je prvo oko z osi, potom y osi, te konačno x osi (Slika 2.1), a dobivaju se kutevi koji se u primjenama najčešće nazivaju i kutevi zakretanja ψ, poniranja θ i valjanja ϕ (engl. *yaw, pitch, roll*).

Ovakav zapis orijentacije se najčešće koristi u aeronautici, ali ima i svoju primjenu u robotici. Međutim, kako je kod mobilnih robota za čvrste podloge jedina promatrana rotacija oko osi z, opisivanje rotacije pomoću Eulerovih kuteva se svodi na samo jedan kut. Interesantan

[1] rotacija oko jedne od koordinatnih osi



Slika 2.1: Eulerovi kutevi

problem koji se javlja kod korištenja Eulerovih kuteva je tzv. *gimbal lock*, pojava kod koje sustav gubi jednu os rotacije. Kada se ravnina xy poklopi s ravninom XY, vrijedi da je $\theta = 0$, dok je moguće odrediti veličinu $(\phi + \psi)$, ali ne i pojedinačne kuteve (slično vrijedi i kad je os z u suprotnom smjeru od Z, kada se može odrediti $\phi - \psi$, ali ne i oba pojedinačno). Kod ostalih vrsta zapisa orijentacije robota, ovaj problem se ne javlja.

2.1.3. Kvaternion

Kvaternioni su općenito proširenje kompleksnih brojeva. Za prikaz rotacije krutog tijela u odnosu na referentni koordinatni sustav se koristi jedinični kvaternion (kvaternion čija je norma jednaka 1) definiran s

$$\hat{\boldsymbol{q}} = q_x \boldsymbol{i} + q_y \boldsymbol{j} + q_z \boldsymbol{k} + q_w = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} \tag{2.7}$$

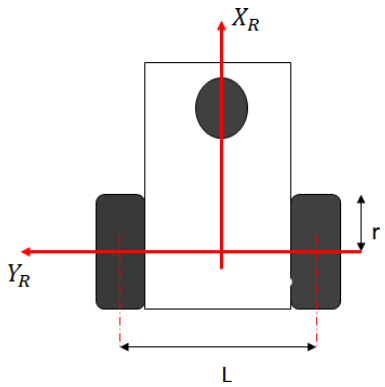
Prednost korištenja kvaterniona u odnosu na rotacijsku matricu je kompaktnost zapisa (sastoje se od 4 broja, dok se rotacijska matrica sastoji od 9), a posljedično i računska efikasnost. U nedostatke kvaterniona u odnosu na druge metod možemo ubrojiti težu interpretaciju od strane čovjeka.

Jednostavan prijelaz iz zapisa kvaterniona u zapis rotacijske matrice je moguć prema

$$\boldsymbol{R} = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2(q_x q_y - q_z q_w) & 2(q_x q_z + q_y q_w) \\ 2(q_x q_y + q_z q_w) & 1 - 2q_x^2 - 2q_z^2 & 2(q_j q_z - q_x q_w) \\ 2(q_x q_z - q_y q_w) & 2(q_x q_w + q_y q_z) & 1 - 2_x^2 - 2q_y^2 \end{bmatrix} \tag{2.8}$$

i obrnuto, iz rotacijske matrice u kvaternion

$$\boldsymbol{q} = \begin{bmatrix} \frac{1}{4q_w}(R_{32} - R_{23}) \\ \frac{1}{4q_w}(R_{13} - R_{31}) \\ \frac{1}{4q_w}(R_{21} - R_{12}) \\ \frac{1}{2}(\sqrt{1 + R_1 1 + R_2 2 + R_3 3}) \end{bmatrix} \tag{2.9}$$



Slika 2.2: Diferencijalni mobilni robot

2.2. Kinematički model diferencijalnog mobilnog robota

Izrada modela mobilnog robota je "bottom-up" proces [1]. Započinje se tako da se promatra svaki od kotača robota koji doprinosi gibanju, te se izvode izrazi za gibanje robota u cjelini.

Prema vrsti pogona koji koriste, postoje različiti modeli mobilnih robota, među kojima su najčešći oni s diferencijalnim pogonom, *bicycle* roboti, *unicycle* roboti, višesmjerni (engl. *omnidirectional*) roboti, itd. U radu će se koristiti model diferencijalnog robota.

Diferencijalne mobilne robote karakterizira pogon koji koristi dva kotača na istoj osovini, od kojih je svakom pojedinačno moguće zadati kutnu brzinu okretanja. Kako su duljina osovine i dimenzije kotača poznate i konstantne, lako se odredi transformacija koja povezuje gibanje robota u cjelini s gibanjem pojedinih kotača. Kod ovog modela, brzina v i kutna brzina ω robota u cjelini su povezani s kutnim brzinama okretanja pojedinih kotača izrazom:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_d \\ \omega_l \end{bmatrix} \tag{2.10}$$

gdje je r radijus kotača robota, L je duljina osovine na kojoj su kotači montirani, a ω_d i ω_l su kutne brzine okretanja desnog, odnosno lijevog kotača. Ove veličine i njihova povezanost su ilustrirani slikom 2.2.

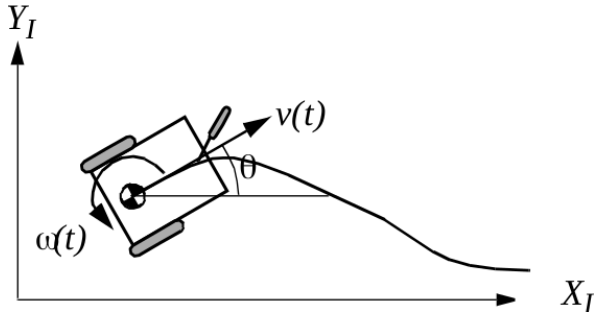
Kinematički model diferencijalnog mobilnog robota (slika 2.3) je dan s:

$$\dot{x} = v \cos\theta \tag{2.11}$$
$$\dot{y} = v \sin\theta \tag{2.12}$$
$$\dot{\theta} = \omega \tag{2.13}$$

a može se zapisati i u matričnom obliku kao

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.14}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{H}\boldsymbol{u} \tag{2.15}$$



Slika 2.3: Diferencijalni mobilni robot u odnosu na referentni koordinatni sustav

2.2.1. Kinematička ograničenja

Kod kinematičkog modela robota, potrebno je uvesti i ograničenja gibanja robota, koja ovise o vrsti pogona i vrsti kotača koje robot koristi, a koji imaju različita kinematička svojstva.

Ograničenja su oblika:

$$F(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = 0 \tag{2.16}$$

Ako ograničenje dano jednaždbom (2.16) može svesti na oblik koji uključuje samo osnovne varijable, bez njihovih derivacija

$$F(\boldsymbol{q}, t) = 0 \tag{2.17}$$

ograničenje je holonomno. [2].

Mobilni roboti su, općenito, neholonomni sustavi jer imaju manje aktuatora od broja stupnjeva slobode. Kod mobilnog robota s diferencijalnim pogonom, broj stupnjeva slobode je 3, dok broj kotača čijim se okretnim momentom može upravljati 2.

Kod standardnih kotača, koji nisu upravljivi i vezani su za robot, kakvi se koriste kod diferencijalnih mobilnih robota, ovo ograničenje je neholonomno. Izvodi se iz jednadžbe (2.14) eliminacijom linearne brzine v, a čijim integriranjem nije moguće dobiti odnos između x, y i θ.

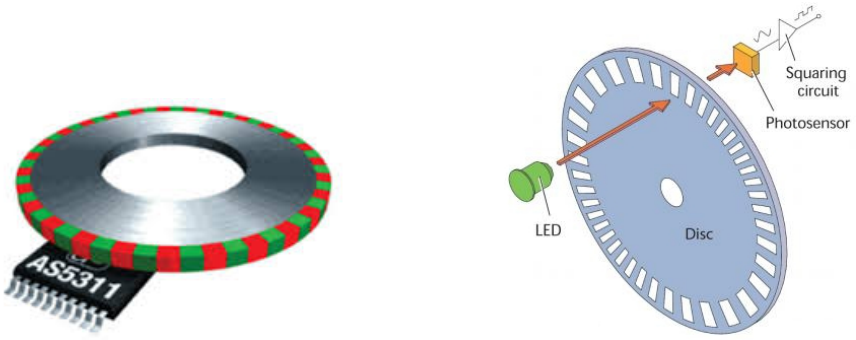
$$\dot{x}\sin\theta - \dot{y}\cos\theta \equiv 0 \tag{2.18}$$

2.2.2. Probabilistički model robota

Kinematički model robota opisan jednadžbom (2.14) je deterministički. Kako su u stvarnosti sustavi vrlo rijetko deterministički, kinematički model mobilnog robota ćemo opisati i probabilistički.

Na kretanje stvarnog robota utječe šum pa su stvarne brzine robota u određenoj mjeri različite od onih zadanih upravljačkim signalima. U tom slučaju stvarna brzina je jednaka zadanoj brzini uz dodani za mali iznos koji predstavlja (bijeli) šum:

$$\begin{bmatrix} \hat{v} \\ \hat{\omega} \end{bmatrix} = \begin{bmatrix} v \\ \omega \end{bmatrix} + \mathcal{N}(0, \sigma) \tag{2.19}$$



(a) Magnetski enkoder (b) Optički enkoder

Slika 2.4: Shematski prikazi magnetskih i optičkih enkodera

2.3. Senzori i obrada signala

Senzori su vrlo važan dio autonomnih mobilnih robota, jer mu omogućavaju prikupljanje podataka o sebi i okolini. Senzori koji se koriste su vrlo različiti i može ih se generalno podijeliti na dva načina [1]:

1. prema funkciji: na *proprioceptivne* (mjere unutarnja stanja robota, npr. brzinu i napon baterije) i *exteroceptivne* (podaci dolaze iz okoline, npr. senzori udaljenosti)
2. prema vrsti energije: na pasivne (mjere "energiju" koja dolazi iz okoliša, npr. senzori temperature, mikrofoni) i aktivne (emitiraju signal u okoliš, te mjere reakciju okoliša, npr. ultrazvučni i laserski senzori udaljenosti)

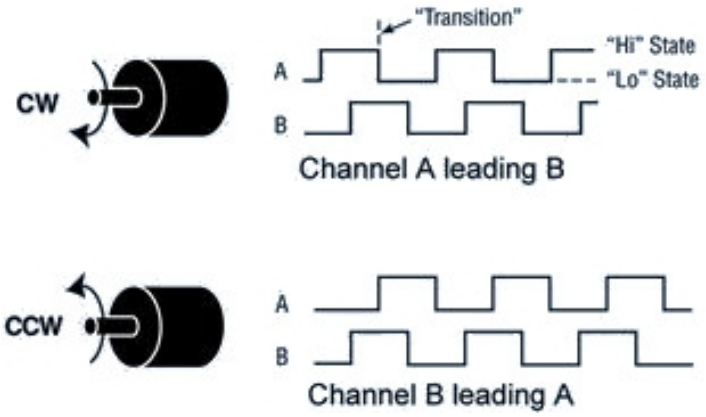
Računalnom obradom signala dobivenih sa senzora moguće je dobivene informacije iskoristiti za razne primjene (npr. podatke sa senzora udaljenosti se primjenjuje kod lokalizacije, autonomne navigacije i mapiranja). Ovisno o vrsti senzora, njihove izlazne signale je potrebno obraditi na različite načine kako bi se iz njih izvukle odgovarajuće informacije.

U nastavku se daje pregled najčešće korištenih senzora u mobilnoj robotici.

2.3.1. Enkoderi

Enkoder, u kontekstu mobilnih robota, je senzor koji služi za dobivanje položaja, a posljedično i kutne brzine kotača, na način da pretvara okretanje kotača u analogni ili digitalni signal. Općenito se dijele na diferencijalne (inkrementalne) i apsolutne. Postoje dvije osnovne vrste enkodera prema načinu rada: magnetski i optički enkoderi. Magnetski enkoder je prikazan na slici 2.4a, a sastoji se od metalnog diska koji je magnetiziran po svom opsegu s promjenjivim polovima, te senzora koji očitava promjenu u magnetskom polju te je pretvara u signal. Kod optičkih enkodera, koji je shematski prikazan na slici 2.4b, disk je napravljen od plastike, a po rubu se nalaze naizmjenično prozirne i neprozirne zone, te LED diode koja emitira svjetlo i fotodiode, koja detektira reflektirano svjetlo za vrijeme prozirne zone, dok ne reflektira ništa za vrijeme neprozirne zone, te tu informaciju pretvara u signal.

Enkoderi su senzori koji dolaze kao standardna oprema na gotovo svim ozbiljnijim mobilnim robota. Postavljaju se na osovinu motora koji pokreću kotače robota, a koriste se za mjerenje udaljenosti (kuta) koju je kotač prešao. S obzirom da su radijus kotača i broj magnetskih



Slika 2.5: Ilustracija dobivenih signala kada se enkoder okreće u smjeru kazaljke, odnosno u suprotnom od smjera kazaljke. Izvor: Dynapar Encoders

polova, odnosno prozirnih i neprozirnih zona poznati, moguće je dobiti udaljenost koju je kotač prešao, a posljedično i brzinu.

Enkoder na svom izlazu ima dva signala, A i B, koji proizvode (obično) pravokutne impulse kada se enkoder okreće, a A i B su postavljeni tako da su u fazi pomaknuti za 90°[2]. Taj pomak u fazi će biti ili pozitivan ili negativan, što nam omogućava da na temelju toga detektiramo okreće li se kotač unaprijed ili unazad. Frekvencija impulsa je proporcionalna brzini okretanja osovine kotača. Ako se signali ne mijenjaju kroz vrijeme, to znači da kotač miruje. Ovo je ilustrirano slikom 2.5.

Enkoderi se najčešće koriste za odometriju. Najopćenitije, odometrija (od gr. *οδός* = put) je mjerenje pređenog puta. U kontekstu mobilnih robota, odometrija je mjerenje prijeđenog puta koje se temelji na mjerenju i vremenskoj integraciji rotacije kotača robota, te uz poznate dimenzije kotača i njihovog razmaka, te upravljačkog signala.

Korištenjem kinematičkom modela robota opisanog jednadžbom (2.15), integriranjem se dobiva pozicija i orijentacija robota u trenutku t (upravljački signal $\boldsymbol{u}$ je vremenski promjenjiv):

$$\boldsymbol{q}_t = \boldsymbol{q}_0 + \int_0^t \boldsymbol{H}\boldsymbol{u}\,\mathrm{d}t \tag{2.20}$$

Jednadžba (2.20) nije pogodna za implementaciju zbog integrala. Kako je računalo diskretni uređaj, nije u stanju analitički riješiti integral, pa je jednadžbu (2.20) potrebno numerički aproksimirati kao sumu od početka gibanja do trenutnog trenutka t:

$$\boldsymbol{q}_t = \boldsymbol{q}_0 + \sum_{k=0}^{t} \boldsymbol{H}\boldsymbol{u}_k\,\Delta t \tag{2.21}$$

Ako se za Δt uzme dovoljno mali konstantni vremenski raspon (npr. 0.02 s), jednadžba (2.21) vrlo dobro aproksimira jednadžbu (2.20).

Iako je odometrija vrlo jednostavna za implementaciju u realnom vremenu, pogreška mjerenja se akumulira (zbog integrala/sume) te ju je potrebno korigirati. Općenito, pogreške u

[2]Enkoder s ovakvim izlazima se naziva engl. *quadrature encoder*

odometriji možemo podijeliti u nesistemske i sistemske. Nesistemske pogreške su one koje su uzrokovane vanjskim faktorima, npr. neravna podloga po kojoj robot vozi ili proklizavanje kotača po podlozi. Ove pogreške su u pravilu nepredvidive. Sistemske pogreške su uzrokovane kinematičkim nesavršenostima robota, najčešće zbog nejednakog radijusa kotača i zbog netočnog podatka o međuosovniskom razmaku [3]. Na sistemske pogreške se može utjecati matematički, modifikacijom algoritma za računanje odometrije na osnovu podataka s enkodera [3].

U literaturi se može pronaći različite pristupe kojima se nastoji utjecaj navedenih pogrešaka na točnost rezultata svesti na najmanju moguću mjeru. U [4] je predložen model koji kombinira sistemsku i nesistemsku pogrešku u jednu, zajedničku pogrešku te predstavlja statističku metodu za uklanjanje pogreške. U [5] je predstavljena metoda za detekciju i korekciju mjerenja odometrije kod proklizavanja kotača. Detekcija se temelji na mjerenju struje elektromotora koji pokreće kotač robota, a korekcija radi tako da se prilagođavaju očitanja enkodera.

Postoje i druge metode za korekcije pogrešaka odometrije, ali one se uglavnom oslanjaju na podatke dobivene iz okoline putem drugih senzora na robotu, te će biti obrađena u slijedećim poglavljima.

2.3.2. Senzori smjera

U senzore smjera koji se koriste u robotici ubrajamo žiroskope i magnetometre.

Žiroskop zadržava svoju orijentaciju u odnosu na fiksni referentni sustav, pa su stoga pogodni za mjerenje smjera pokretnog sustava. Može ih se podijeliti na mehaničke i optičke žiroskope. Mehanički žiroskop se zasniva principu očuvanja momenta sile koji je dan s

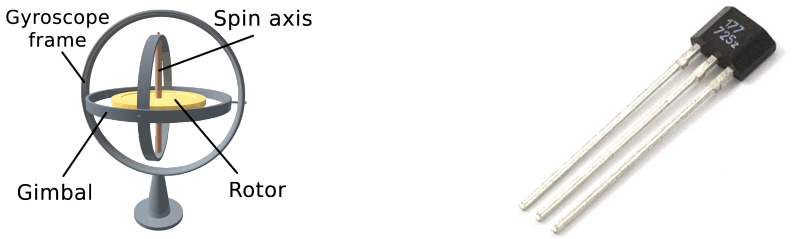
$$\boldsymbol{L} = I \times \boldsymbol{\omega} \tag{2.22}$$

Sastoji se od rotirajućeg diska koji je pričvršćen na osovinu tako da može mijenjati os vrtnje (slika 2.6a). Rotacija diska proizvodi inerciju koja os rotacije diska u nedostatku nekih vanjskih smetnji zadržava usmjerenu u fiksnom pravcu u prostoru (tj. u fiksnoj orijentaciji u odnosu na referentni koordinatni sustav). Osim mogućnosti okretanja oko svoje osi, žirokop ima barem još jedan stupanj slobode kretanja. Na taj način žiroskop dobiva svojstva velike stabilnosti i precesije. Stabilnost žiroskopa se ogleda u tome što se snažno suprotstavlja svim vanjskim utjecajima koji mu žele promijeniti položaj osi. Pod precesijom se podrazumijeva osobinu žiroskopa da pri promjeni položaja jedne njegove osi skreće oko druge, njoj okomite osi.

Optički žiroskopi su inovacije novijeg datuma, a zasnivaju se na mjerenju kutnih brzina na temelju emitiranja jednobojnih zraka svjetla (lasera) iz istog izvora, jednu u smjeru vrtnje, a jednu u suprotnom smjeru kroz optičko vlakno. Laserska svjetlost koja putuje u smjeru vrtnje će na odredište doći nešto ranije od one koja putuje u suprotnom smjeru zbog nešto kraće putanje, pa će zato imati višu frekvenciju (Sagnacov efekt). Razlika u frekvenciji je onda proporcionalna kutnoj brzini precesije žiroskopa.

Magnetometri su uređaji za mjerenje smjera magnetskog polja, a najčešći su temeljeni na Hallovom efektu, magnetskom otporu, flux gate senzori, te MEMS[3] magnetometri. Hallov

[3]Micro Electro-Mechanical Systems, tehnologija za izradu mikroskopskih uređaja, koji najčešće imaju pomične djelove. Karakterizira ih vrlo mala dimenzija, do 0.1 mm, a dimenzije uređaja koji sadrže MEMS do 1 mm.



(a) Mehanički žiroskop (b) Senzor Hallovog efekta

Slika 2.6: Senzori smjera

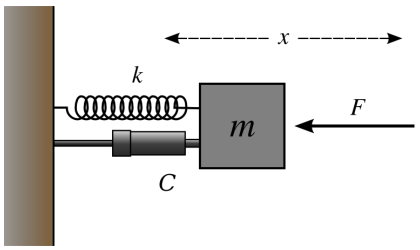
efekt opisuje ponašanje električnog potencijala unutar poluvodiča u prisutnosti magnetskog polja. Ako se na poluvodič dovede konstantna struja po cijeloj njegovoj dužini, inducira se napon u okomitom smjeru, po širini, u odnosu na relativnu orijentaciju poluvodiča i silnica magnetskog polja. Zato jedan poluvodič može mjeriti smjer u samo jednoj dimenziji, pa su stoga za primjene u robotici popularni magnetometri s dva poluvodiča postavljena tako da mogu pokazivati smjer u dvije dimenzije. Magnetometri koje rade na magnetootpornom principu su napraljeni od materijala koji s promjenom magnetskog polja mijenja svoj otpor. Osjetljivost im je usmjerena duž jedne od osi, a moguće je proizvesti i 3D varijante senzora. Rezolucija mu je oko 0.1°, a brzina odziva mu je oko 1μs. Kod flux gate magnetometra dvije zavojnice se namotaju oko feritne jezgre i fiksiraju jedna okomito na drugu. Kada se kroz njih propusti izmjenična struja, magnetsko polje uzrokuje pomake u fazi koji se mjere te na temelju njih računaju smjerovi magnetskog polja u dvije dimenzije. Konačno, MEMS magnetometri se javljaju u više izvedbi, od kojih je najčešća ona u kojoj se mjere efekti Lorenzove sile. Mjeri se mehanički pomak unutar strukture MEMS senzora koja je rezultat Lorenzove sile koja djeluje na vodič u magnetskom polju. Pomak se mjeri optičkim (laser ili LED) ili elektroničkim putem (piezootpor ili elektrostatička pretvorba).

2.3.3. Akcelerometri

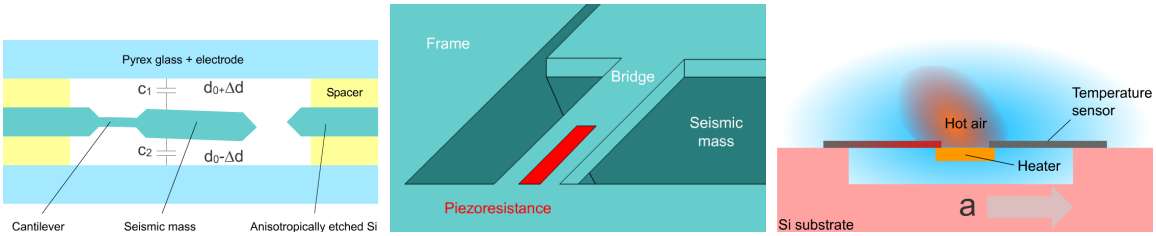
Akcelerometar je uređaj koji mjeri sve vanjske sile koje na njega djeluju (uključujući i gravitaciju). Konceptualno, akcelerometar se ponaša kao sustav mase obješene na oprugu s prigušnicom. Kada neka sila djeluje na sustav ilustriran slikom 2.7, ona djeluje na masu i razvlači oprugu prema:

$$\boldsymbol{F} = \boldsymbol{F}_{in} + \boldsymbol{F}_{prig} + \boldsymbol{F}_{opr} = m\ddot{\boldsymbol{x}} + c\dot{\boldsymbol{x}} + k\boldsymbol{x} \tag{2.23}$$

gdje je m masa, c je koeficijent prigušenja, a k je koeficijent rastezanja opruge. Ovakav



Slika 2.7: Princip rada akcelerometra



(a) Kapacitivni (b) Piezoelektrični (c) Termodinamički

Slika 2.8: MEMS akcelerometri u različitim izvedbama

akcelerometar se naziva mehanički akcelerometar, te je sile na ovaj način potrebno mjeriti direktno što nije pogodno za primjene u robotici. Postoje još i piezoelektrični akcelerometri koji funkcioniraju na temelju svojstava određenih kristala koji pod djelovanjem sile induciraju određeni napon koji je moguće mjeriti. Većina modernih akcelerometara u primjenama su MEMS akcelerometri, koji postoje u više izvedbi. Pri primjeni sile masa se pomiče iz svog neutralnog položaja. Pomak u odnosu na neutralni položaj se mjeri u ovisnosti o kojoj fizikalnoj izvedbi MEMS akcelerometra se radi, pa imamo kapacitivne akcelerometre kod kojih se mjeri kapacitet između fiksne strukture i mase koja se pomiče, drugi su piezoelektrični akcelerometri u MEMS izvedbi, te u termodinamičkoj izvedbi u kojoj se grijačem zagrijava balon zraka, koji se pomiče u suprotnom smjeru od smjera akceleracije, a na krajevima su postavljeni senzori temperature kako bi se dobio signal. Ove vrste senzora su prikazane na slici 2.8.

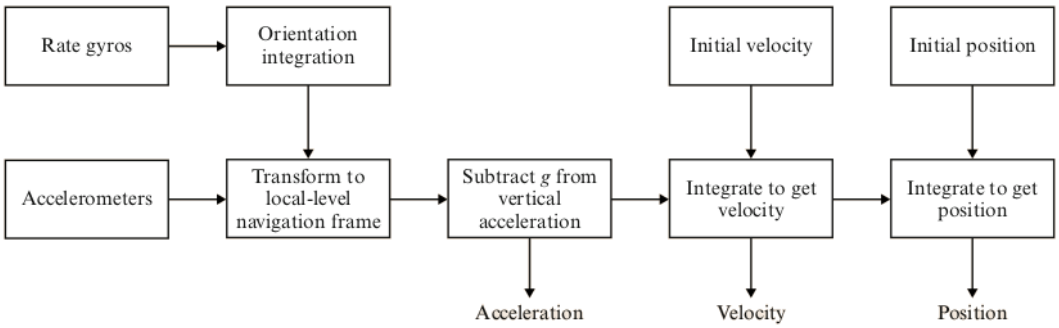
2.3.4. IMU

Jedinica za mjerenje inercije (engl. *inertial measurement unit*, IMU) je elektronički uređaj koji se sastoji od žiroskopa, akcelerometra i magnetometra (opcionalno), te mjeri kut za svaku od osi robota (poniranje, valjanje, zakretanje). Postoje izvedbe s po tri komada svakog od senzora (po jedan za svaki os), te izvedbe s jednim od svakog, ali tada je svaki senzor u 3D izvedbi (mjeri u sve tri dimenzije).

IMU služi prvenstveno kao senzor orijentacije. U izvedbama IMU u kojima je prisutan magnetometar, koristi ga se za popravljanje pogreške žiroskopa. Dobivene kutove se dalje koristi za kompenziranje utjecaja gravitacije na očitanja akcelerometra. Naime, zakretanjem IMU-a, gravitacija se projicira na dvije osi, pa je stoga za kompenzaciju potrebno poznavati kut između tih osi.

Ima šest stupnjeva slobode, pa može dati procjenu pozicije (x, y, z), te orijentacije (α, β, γ), te brzine i ubrzanja u smjeru svih osi krutog tijela. IMU mjeri akceleracije i Eulerove kuteve robota, a integriranjem (uz poznatu početnu brzinu) se može dobiti brzina, odnosno dvostrukim integriranjem (uz poznati početni položaj) se može dobiti pozicija robota. Ovo je ilustrirano na slici 2.9.

IMU su prilično osjetljivi na drift kod žiroskopa, što unosi pogrešku u procjenu orijentacije robota, što za posljedicu ima pogrešku kod kompenzacije gravitacije kod očitanja akcelerometra. Pogreške očitanja akcelerometra su, u kontekstu dobivanja pozicije, još više izražene jer se mjerenje akcelerometrom integrira dvaput, pa protekom vremena akumulirana pogreška samo raste. Za poništavanje utjecaja IMU pogreške, potrebno je koristiti neke od metoda koje se zasnivaju na nekim drugim senzorima, primjerice lokalizacije.



Slika 2.9: IMU blok dijagram. Izvor: [6]

2.3.5. Ultrazvučni senzori

Ultrazvučni senzori (ili sonari) su senzori udaljenosti. Oni mjere udaljenost tako da emitiraju zvučni signal kratkog trajanja na ultrazvučnoj frekvenciji, te mjere vrijeme od emisije signala dok se reflektirani val (jeka) ne vrati natrag u senzor. Izmjereno vrijeme je proporcionalno dvostrukoj udaljenosti do najbliže prepreke u rasponu senzora, a iz toga se lako dobija udaljenost do najbliže prepreke, prema

$$d = \frac{1}{2} v_z t_0 \tag{2.24}$$

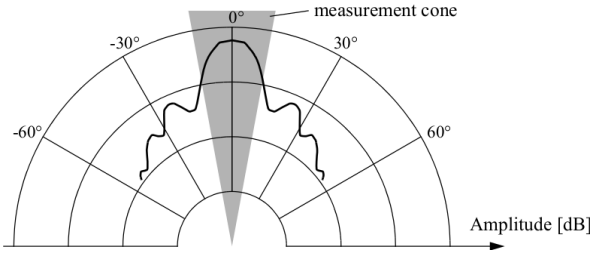
gdje je v_z brzina zvuka, a t_0 je izmjereno vrijeme. Ultrazvučni val se tipično emitira u rasponu frekvencija od 40 kHz do 180 kHz, a udaljenosti koje mogu ovakvi senzori detektirati se kreću od 12 cm do 5 m. Kod mjerenja udaljenosti ultrazvukom treba voditi računa da se zvučni valovi šire kružno, prema slici 2.10.

Ovo za posljedicu ima da se korištenjem ultrazvučnog senzora ne dobivaju točke različite dubine, već regije konstantne dubine, što znači da je prisutan objekt na određenoj udaljenosti, a unutar mjernog konusa.

Neki od ultrazvučnih senzora su prikazani na slikama 2.11a i 2.11b. Uređaj na slici 2.11b osim ultrazvuka sadrži i infracrveni senzor udaljenosti, te za izračun udaljenosti koristi mjerenja dobivena od oba senzora.

2.3.6. Laserski senzori udaljenosti

Laserski senzori udaljenosti (LiDAR senzori, od engl. *Light Detection And Ranging*) mjere udaljenost na temelju emitirane i reflektirane svjetlosti, na sličan način kao i sonari. Ovi



Slika 2.10: Distribucija intenziteta tipičnog ultrazvučnog senzora

(a) HC-SR04 (b) Teraranger Duo (c) RPLIDAR

Slika 2.11: Senzori udaljenosti

senzori se sastoje od predajnika koji osvjetljuje objekt laserskom zrakom i prijamnika koji prima reflektiranu zraku. Ovi senzori su sposobni pokriti svih 360° u ravnini jer se sastoje od rotirajućeg sustava koje usmjeruje svjetlost tako da pokrije cijelu ravninu. Primjer LiDAR senzora je dan na slici 2.11c.

Mjerenje udaljenosti se zasniva na mjerenju faznog otklona reflektiranog svjetla prema shemi prikazanoj na slici 2.12.

Iz izvora se emitira (skoro) infracrveni val koji se, kada naleti na većinu prepreka (osim onih vrlo fino ispoloranih ili prozirnih) reflektira. Komponenta reflektirane zrake koja se vrati u senzor će biti skoro paralelna emitiranoj zraci za objekte koji su relativno daleko. Kako senzor emitira val poznate frekvencije i amplitude, moguće je mjeriti fazni otklon između emitiranog i reflektiranog signala. Duljina koju svijetlost pređe je, prema slici 2.12

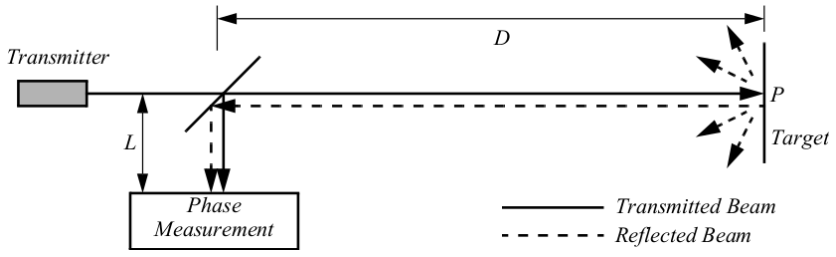
$$D' = L + 2D = L + \frac{\theta}{2\pi}\lambda \tag{2.25}$$

gdje je θ izmjereni kut faznog otklona između emitirane i reflektirane zrake.

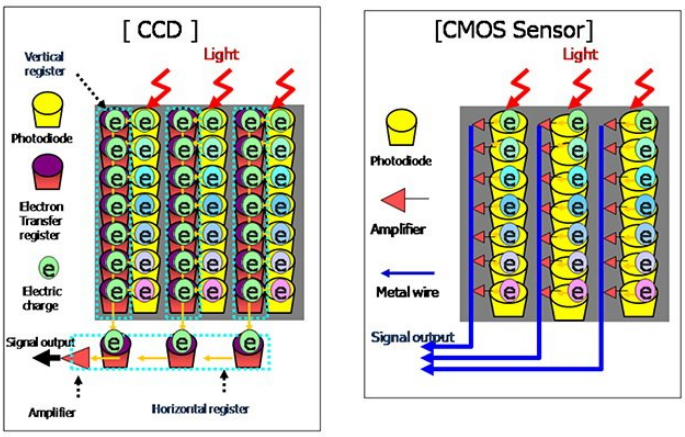
Postoje i 3D laserski senzori udaljenosti, koji funkcioniraju na sličnim principima, ali svoje podatke dobavljaju u više od jedne ravnine.

2.3.7. Senzori vida

Vid je vjerojatno "najmoćnije" ljudsko osjetilo koje obiluje informacijama o vanjskom svijetu, pa su zbog toga razvijeni odgovarajući senzori koji bi imitirali ljudski vid na strojevima. Senzori vida su digitalne i video kamere, koje se, osim kod mobilnih robota koristi i u raznim drugim, svakodnevnim primjenama. Interpretacija toga što robot vidi korištenjem kamera, tj. izvlačenje informacije iz slike koju kamera snimi se dobiva obradom i analizom slika. Međutim, osim svojih prednosti, mane kamera se mogu očitovati u kvaliteti snimaka ako



Slika 2.12: Shematski prikaz mjerenja udaljenosti pomoću faznog otklona. Izvor: [1]



Slika 2.13: Shematski prikaz CCD i CMOS čipova. Izvor: Emergent Vision Technologies

su snimljene pri neodgovarajućem osvjetljenju ili ako su loše fokusirane, te u tome da je moguće pogrešno interpretirati snimljenu scenu.

Današnje digitalne i video kamere se razlikuju po čipovima koji se u njima koriste:

CCD (*Charged coupled device*) čipovi su matrice piksela osjetljivih na svjetlo, a svaki piksel se obično modelira kao kondenzator koji je inicijalno napunjen a koji se prazni kada je osvjetljen. Za vrijeme osvjetljenosti, fotoni padaju na piksele i oslobađaju elektrone koje hvataju električna polja i zadržavaju na tom pikselu. Po završetku osvjetljavanja, naponi na svakom od piksela se "čitaju", te se ta informacija digitalizira i pretvara u sliku.

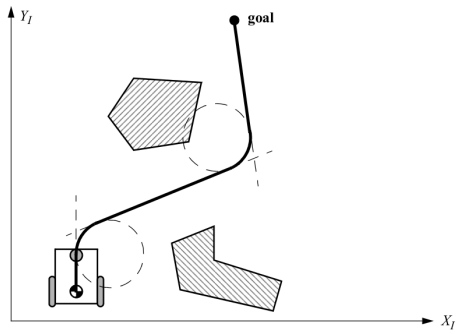
CMOS (*Complementary metal oxide on silicon*) su čipovi koji također imaju matrice piksela, ali ovdje je uz svaki piksel smješteno nekoliko tranzistora koji su specifični za taj piksel. I ovdje se skuplja osvjetljenje na pikselima, ali su tranzistori ti koji su zaduženi za pojačavanje i pretvaranje električnog signala u sliku, što se događa paralelno za svaki piksel u matrici.

2.4. Upravljanje mobilnim robotom

Upravljanje mobilnim robotom je problem koji se rješava određivanjem brzina i kutnih brzina (ili sila i zakretnih momenata u slučaju korištenja dinamičkog modela robota) koji će robot dovesti u zadanu konfiguraciju, omogućiti mu da prati zadanu trajektoriju ili općenitije, da izvrši zadani zadatak s određenim performansama.

Robotom se može upravljati vođenjem (open-loop) i regulacijom. Vođenje se može upotrijebiti za praćenje zadane trajektorije tako da ju se podijeli na segmente, obično na ravne linije i kružne segmente. Problem vođenja se rješava tako da se unaprijed izračuna glatka putanja od početne do zadane krajnje konfiguracije (primjer na slici 2.14). Ovaj način upravljanja ima brojne nedostatke. Najveći je taj da je često teško odrediti izvedivu putanju do zadane konfiguracije uzimajući u obzir kinematička i dinamička ograničenja robota, te nemogućnost robota da se adaptira ako se u okolini dogodi neka dinamička promjena.

Prikladnije metode upravljanja robotom se zasnivaju na povratnoj vezi [7]. U ovom slučaju zadatak regulatora je zadavanje međucilja koji se nalazi na putanji do zadanog cilja. Ako se uzme da je pogreška robotove pozicije i orijentacije u odnosu na zadani cilj (izražena u koordinatnom sustavu robota) jednaka $\boldsymbol{e} = {}^{R}[\, x \quad y \quad \theta \,]^{T}$, zadatak je izvesti regulator koji će



Slika 2.14: Vođenje mobilnog robota. Putanja do cilja je podijeljena na ravne linije i segmente kruga.

dati upravljački signal $\boldsymbol{u}$ takav da $\boldsymbol{e}$ teži prema nuli. Koordinatni sustavi i oznake korištene za ovaj primjer su prikazani na slici 2.15.

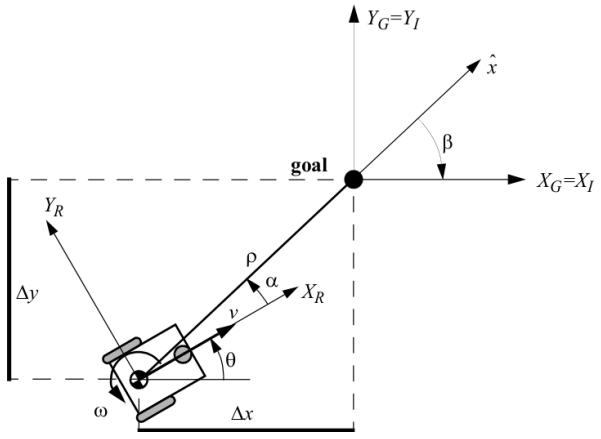
Ako se kinematički model robota opisan jednadžbom (2.15) prikaže u polarnom koordinatnom sustavu sa ishodištem u zadanom cilju, onda imamo:

$$\begin{aligned} \rho &= \sqrt{\Delta x^2 + \Delta y^2} \\ \alpha &= -\theta + \arctan \frac{\Delta y}{\Delta x} \\ \beta &= -\theta - \alpha \end{aligned} \tag{2.26}$$

Korištenjem jednadžbe (2.26) izvodi se zapis kinematike robota u polarnim koordinatama

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.27}$$

gdje su ρ udaljenost od robota do cilja, a θ je orijentacija robota (kut koji robot zatvara s referentnim koordinatnim sustavom). Ovdje je polarni koordinatni sustav uveden kako bi se olakšalo pronalaženje odgovarajućih upravljačkih signala te analiza stabilnosti. Ako upravljački signal ima oblik



Slika 2.15: Opis robota u polarnom koordinatnom sustavu s ishodištem u zadanom cilju

$$\boldsymbol{u} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} k_\rho \rho \\ k_\alpha \alpha + k_\beta \beta \end{bmatrix} \tag{2.28}$$

iz jednadžbe (2.26) dobiva se opis sustava zatvorene petlje

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -k_\rho \rho \cos \alpha \\ k_\rho \sin \alpha - k_\alpha \alpha - k_\beta \beta \\ -k_\rho \sin \alpha \end{bmatrix} \tag{2.29}$$

Iz analize stabilnosti sustava opisanog matričnom jednadžbom (2.29) slijedi se da je sustav stabilan dok su je zadovoljeno $k_\rho > 0$, $k_\beta < 0$ i $k_\alpha - k_\rho > 0$.

3. Lokalizacija i navigacija

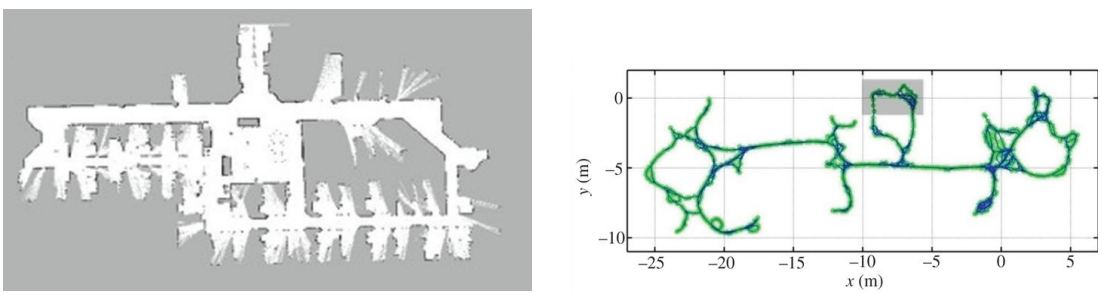
3.1. Zapisi okoline - mape

Za opisivanje prostora (okoliša) u kojima se roboti kreću se koriste mape. Njima opisujemo svojstva i lokacije pojedinih objekata u prostoru.

U robotici razlikujemo dvije glavne vrste mapa: metričke i topološke. Metričke mape se temelje na poznavanju dvodimenzionalnog prostora u kojem su smješteni objekti na određene koordinate. Prednost im je ta što su jednostavnije i ljudima i njihovom načinu razmišljanja bliže, dok im je mana osjetljivost na šum i teškoće u preciznom računanju udaljenosti koje su potrebne za izradu kvalitetnih mapa. Topološke mape u obzir uzimaju samo određena mjesta i relacije među njima, pa takve mape prikazujemo kao grafove kojima su ta mjesta vrhovi, a udaljenosti među njima su stranice grafa. Ove dvije vrste mapa su usporedno prikazane na slici 3.1.

Najpoznatiji model za prikaz mapa koji se koristi u robotici su tzv. *occupancy grid* mape. One spadaju pod metričke mape i ima široku primjenu u mobilnoj robotici. Kod njih se za svaku (x, y) koordinatu dodjeljuje binarna vrijednost koja opisuje je li se na toj lokaciji nalazi objekt, te se stoga lako može koristiti za pronaći putanju kroz prostor koji je slobodan. Binarnim 1 se opisuje slobodan prostor, dok se s binarnom 0 opisuje prostor kojeg zauzima prepreka. Kod nekih implementacija *occupancy grid* mape se pojavljuje i treća moguća vrijednost koja je negativna (najčešće -1), a njom se opisuju točke na mapi koje nisu definirane (tj. ne zna se je li taj prostor slobodan ili ne, budući da ga robot kojim mapiramo nije posjetio).

U 2015. godini je donesen standard IEEE 1873-2015 [9] za prikaz dvodimenzionalnih mapa za primjenu u robotici. Definiran je XML zapis lokalnih mapa koje mogu biti topološke, mrežne (engl. *grid-based*) i geometrijske. Iako zapis mape u XML formatu zauzima nešto više memorijskog prostora od nekih drugih zapisa, glavna prednost mu je što je "čitljiv" te ga je lako debuggirati i otkloniti pogreške, ako ih ima. Globalna mapa se onda sastoji od više lokalnih mapa koje ne moraju biti iste vrste, što omogućava kombiniranje mapa izrađenih



(a) *Occupancy grid* mapa (b) *Topološka mapa*

Slika 3.1: Primjeri *occupancy grid* i topološke mape. Izvor: [8]

korištenjem više različitih robota. Ovakvim standardom se nastoji postići interoperabilnost između različitih robotskih sustava.

3.2. Procjena položaja i orijentacije

Jedan od najosnovnijih postupaka u robotici je procjena stanja robota (ili robotskog manipulatora) i njegove okoline iz dostupnih senzorskih podataka. Za ovo se u literaturi najčešće koristi naziv lokalizacija. Pod stanjem robota se mogu podrazumijevati položaj i orijentacija, te brzina. Senzori uglavnom ne mjere direktno veličine koje nam trebaju, već se iz senzorskih podataka te veličine moraju rekonstruirati i dobiti procjenu stanja robota. Taj postupak je probabilistički, zbog dinamičnosti okoline, te algoritmi za probabilističku procjenu stanja robota zapravo daju distribucije vjerojatnosti da je robot u nekom stanju.

Među najvažnijim i najčešće korištenjim stanjima robota je njegova konfiguracija (koja uključuje položaj i orijentaciju) u odnosu na neki fiksni koordinatni sustav. Postupak lokalizacije robota uključuje određivanje konfiguracije robota u odnosu prethodno napravljenu mapu okoline u kojoj se robot kreće.

Prema [10], problem lokalizacije robota je moguće podijeliti na tri različite vrste s obzirom na to koji podaci su poznati na početku i trenutno. Tako postoje:

Praćenje pozicije Ovo je najjednostavniji slučaj problema lokalizacije. Inicijalna pozicija i orijentacija unutar okoliša moraju biti poznate. Ovi postupci uzimaju u obzir odometriju tijekom gibanja robota te daju procjenu lokacije robota (uz pretpostavku da je pogreška pozicioniranja zbog šuma malena). Ovaj problem se smatra lokalnim jer je nesigurnost ograničena na prostor vrlo blizu robotove stvarne lokacije.

Globalna lokalizacija Ovdje početna konfiguracija nije poznata. Kod globalne lokalizacije nije moguće pretpostaviti ograničenost pogreške pozicioniranja na ograničeni prostor oko robotove stvarne pozicije, pa je stoga ovaj problem teži od problema praćenja pozicije.

Problem kidnapiranog robota Ovaj problem je inačica gornjeg problema. Tijekom rada, robota se "otme" i "prenese" na neku drugu lokaciju unutar istog okoliša, bez da je robot svjestan nagle promjene lokacije. Rješavanje ovog problema je vrlo važno da se testira može li algoritam za lokalizaciju ispravno raditi kada globalna lokalizacija ne funkcionira.

3.2.1. Lokalizacija temeljena na Kalmanovom filteru

Kalmanov filter (KF) [11] je metoda za spajanje podataka i predviđanje odziva linearnih sustava uz pretpostavku bijelog šuma u sustavu. S obzirom da je robot, čak i u najjednostavnijim slučajevima općenito nelinearan sustav, Kalmanov filter u svom osnovnom obliku nije moguće koristiti.

Stoga, uvodi se Extended Kalman filter (EKF) koji rješava problem primjene KF za nelinearne sustave, uz pretpostavku da je funkcija koja opisuje sustav derivabilna. EKF se temelji na linearizaciji sustava razvojem u Taylorov red. Za radnu točku se uzima najvjerojatnije stanje sustava (robota), te se u okolini radne točke obavlja linearizacija.

Općenito, EKF na temelju stanja u trenutku $t-1$ i pripadajuće srednje vrijednosti poze robota μ_{t-1}, kovarijance Σ_{t-1}, upravljanja u_{t-1} i mape (bazirane na svojstvima) m na izlazu daje novu procjenu stanja u trenutku t sa srednjom vrijednosti μ_t i kovarijancom Σ_t.

EKF je u širokoj primjeni za lokalizaciju u mobilnim robotima i jedna je od danas najčešće korištenih metoda za tu namjenu. Prvi put se spominje 1991. u [12] gdje se metoda temeljena za EKF koristi za lokalizaciju i navigaciju u poznatoj okolini korištenjem koncepta "geometrijskih svjetionika" (prema autorima, to su objekti koje se može konzistentno opisati ponovljenim mjerenjima pomoću senzora ili, pojednostavljeno, nekakva svojstva koja se pojavljuju u okolišu i korisni su za navigaciju). U 1999. je razvijen adaptivni EKF za lokalizaciju mobilnih robota kod kojeg se on-line prilagođavaju kovarijance šumova senzorskih (ulaznih) i mjerenih (izlaznih) podataka [13].

Jedna od poznatijih implementacija ove metode je [14]. Karakterizira je mogućnost korištenja proizvoljnog broja senzorskih podataka na ulazu u filter, a korištena je u Robot Operating System-u (ROS), vrlo popularnoj modernoj programskoj platformi za razvoj robotskih prototipova. EKF se koristi kao jedan dio metoda za (djelomično) druge namjene, kao što je Simpltaneous Localisation and Mapping (SLAM) [15], metode koja istovremeno mapira nepoznati prostor i lokalizira robota unutar tog prostora. U poglavlju 3.3. metoda će biti detaljnije prezentirana.

Osim ovdje opisane varijante EKF-a, postoje i druge koje koriste naprednije i preciznije tehnike linearizacije neliarnog sustava. Ovim se postiže manja pogreška linearizacije za sustave koji su visoko nelinearni. Te metode su iterativni EKF, EKF drugog reda, te EKF viših redova, te su opisani u [16]. Kod prethodno opisane varijante EKF-a, spomenuto je kako se linearizacija obavlja razvojem u Taylorov red oko najvjerojatnijeg stanja robota. Kod iterativnog EKF-a, nakon prethodno opisane linearizacije se obavlja relinearizacija kako bi se dobio što točniji linearizirani model. Ovaj postupak relinearizacije je moguće ponoviti više puta, ali u praksi je to dovoljno napraviti jednom. Kod EKF-a drugog reda, postupak je ekvivalentan iterativnom EKF-u, ali se kod razvoja u Taylorov red uzimaju dva elementa (u odnosu na jedan u osnovnoj i interativnoj varijanti).

Osim EKF, razvijena je još jedna metoda za rješavanje problema primjene KF za nelinearne sustave, i to za visoko nelinearne sustave za koje primjena EKF-a ne pokazuje dobre performanse. Metoda se naziva Unscented Kalman Filter (UKF), a temelji se na deterministčkom metodi uzorkovanja (*unscented* transformaciji), koja se koristi za odabir minimalnog skupa točaka oko stvarne srednje vrijednosti, te se točke propagiraju kroz nelinearnost da se dobije nova procjena srednje vrijednosti i kovarijance [17].

Od ostalih pristupa može se izdvojiti metoda Gaussovih filtera sume, koja razlaže svaku ne-Gaussovu funkciju gustoće vjerojatnosti na konačnu sumu Gaussovih funkcija gustoće vjerojatnosti, na svaku od kojih se onda može primjeniti obični Kalmanov filter [16].

3.2.2. Monte Carlo lokalizacija

Monte Carlo metode su, općenito, metode koji se koriste za rješavanje bilo kojeg problema koji je probabilistički. Zasnivaju se na opetovanom slučajnom uzorkovanju na temelju pretpostavljene distribucije uzoraka, te zakona velikih brojeva, a daju očekivanu vrijednost neke slučajne varijable.

Monte Carlo metoda za lokalizaciju (MCL) robota koristi filter čestica (engl. *particle filter*) [10, 18] koji funkcionira kako slijedi. Procjena trenutnog stanja robota X_t (engl. *belief*)

je funkcija gustoće vjerojatnosti, s obzirom da točnu lokaciju robota nije nikada moguće odrediti. Procjena stanja robota u trenutku t je skup M čestica $X_t = \{\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)}, \cdots, \boldsymbol{x}_t^{(M)}\}$, od kojih je svaka jedno hipoteza o stanju robota. Stanja u čijoj se okolini nalazi veći broj čestica odgovaraju većoj vjerojatnosti da se robot nalazi baš u tim stanjima. Jedina pretpostavka potrebna je da je zadovoljeno Markovljevo svojstvo (da distribucija vjerojatnosti trenutnog stanja ovisi samo o prethodnom stanju, tj. da X_t ovisi samo o X_{t-1}).

Osnovna MCL metoda je opisana u algoritmu 3.1, a značenje korištenih oznake slijedi. $\bar{X}_t$ je privremeni skup čestica koji se koristi u izračunavanju stanja robota X_t, $w_t^{(m)}$ je faktor važnosti (engl. *importance factor*) m-te čestice koji se konstruira iz senzorskih podataka $\boldsymbol{z}_t$ i trenutno procjenjenog stanja robota s obzirom na gibanje $\boldsymbol{x}_t^{(m)}$.

Algoritam 3.1. Monte Carlo lokalizacija

Ulaz: $X_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t, m$
  $\bar{X}_t = X_t = \emptyset$
  for all $m = 1$ to M **do**
    $\boldsymbol{x}_t^{(m)} =$ **motion_update**$(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}^{(m)})$
    $\boldsymbol{w}_t^{(m)} =$ **sensor_update**$(\boldsymbol{z}_t, \boldsymbol{x}_t^{(m)})$
    $\bar{X}_t \leftarrow \bar{X}_t \cup \{\langle \boldsymbol{x}_t^{(m)}, \boldsymbol{w}_t^{(m)} \rangle\}$
  for all $m = 1$ to M **do**
    $\boldsymbol{x}_t^{(m)} \sim \bar{X}_t,\ p(\boldsymbol{x}_t^{(m)}) \propto \boldsymbol{w}_t^{(m)}$
    $X_t \leftarrow X_t \cup \{\boldsymbol{x}_t^{(m)}\}$
Izlaz: X_t

Osnovne prednosti MCL metode u odnosu na metode temeljene na Kalmanovom filteru je globalna lokalizacija [18], te mogućnost modeliranja šumova po distribucijama koje nisu normalne, što je slučaj kod Kalmanovog filtera. Nju omogućava korištenje multimodalnih distribucija vjerojatnosti što kod Kalmanovog filtera nije moguće, što rezultira mogućnošću lokalizacije robota "od nule", tj. bez poznavanja približne početne pozicije i orijentacije.

Jedna od varijanti MCL algoritma je i KLD-sampling MCL ili Adaptive MCL (AMCL). Karakterizira ga metoda za poboljšanje efikasnosti filtera čestica što se realizira promjenjivom veličinom skupa čestica M koja se mijenja u realnom vremenu [19, 20] i čijom primjenom se ostvaruju značajno poboljšane performanse i značajna ušteda računalnih resursa u odnosu na osnovni MCL.

3.3. Simultaneous Localisation and Mapping (SLAM)

SLAM je proces kojim robot stvara mapu okoliša i istovremeno koristi tu mapu za dobivanje svoje lokacije. Trajektorija robotske platforme i lokacije objekata (prepreka) u prostoru nisu unaprijed poznate [15, 21].

Postoje dvije formulacije SLAM problema. Prva je *online* SLAM problem, kod kojega se procjenjuje trenutna *a posteriori* vjerojatnost

$$p(\boldsymbol{x}_t, m \mid \mathrm{Z}_{0:t}, \mathrm{U}_{0:t}, \boldsymbol{x}_0) \tag{3.1}$$

gdje je $\boldsymbol{x}_t$ konfiguracija robota u trenutku t, m je mapa, $\mathrm{Z}_{0:t}$ je skup senzorska očitanja, a $\mathrm{U}_{0:t}$ su upravljački signali.

Druga formulacija je kompletni (*full*) SLAM problem koji se od prethodnog razlikuje po tome što se traži procjena *a posteriori* vjerojatnosti za konfiguracije robota tijekom cijele putanje $\boldsymbol{x}_{1:t}$:

$$p(\boldsymbol{x}_{1:t}, m \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) \tag{3.2}$$

Razlika između ove dvije formulacije je u tome koji algoritmi se mogu koristiti za rješavanje problema. Online SLAM problem je rezultat integriranja prošlih poza robota iz kompletnog SLAM problema. U probabilističkom obliku [15], SLAM problem zahtjeva da se izračuna distribucija vjerojatnosti (3.1) za svaki vremenski trenutak t, uz zadanu početnu pozu robota, upravljačke signale i senzorska očitanja od početka sve do vremenskog trenutka t.

SLAM algoritam je rekurzivan, pa se za izračunavanje vjerojatnosti iz jednadžbe (3.1) u trenutku t koriste vrijednosti dobivene u prošlom trenutku $t-1$ uz korištenje Bayesovog teorema, te upravljačkog signala $\boldsymbol{u}_t$ i senzorskih očitanja $\boldsymbol{z}_t$. Ova operacija zahtjeva da budu poznati modeli promatranja i gibanja. Model promatranja opisuje vjerojatnost za dobivanje očitanja $\boldsymbol{z}_t$ kada su poza robota i stanje orijentira (mapa) poznati

$$P(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{m}) \tag{3.3}$$

Model gibanja robota opisuje distribuciju vjerojatnosti za promjene stanja (tj. konfiguracije) robota

$$P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \tag{3.4}$$

Iz jednadžbe (3.4) je vidljivo da je promjena stanja robota Markovljev proces[1] i da novo stanje $\boldsymbol{x}_t$ ovisi samo o prethodnom stanju $\boldsymbol{x}_{t-1}$ i upravljačkom signalu $\boldsymbol{u}_t$.

Kada je prethodno navedeno poznato, SLAM algoritam je dan u obliku dva koraka koji uključuju rekurzivno sekvencijalno ažuriranje (engl. update) po vremenu (gibanje) i korekcije senzorskih mjerenja.

$$P(\boldsymbol{x}_t, \boldsymbol{m} \mid \mathrm{Z}_{0:t-1}, \mathrm{U}_{0:t-1}, \boldsymbol{x}_0) = \int P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) P(\boldsymbol{x}_{t-1}, \boldsymbol{m} \mid \mathrm{Z}_{0:t-1}, \mathrm{U}_{0:t-1}, {}_x 0) \mathrm{d}\boldsymbol{x}_{t-1} \tag{3.5}$$

$$P(\boldsymbol{x}_t, \boldsymbol{m} \mid \mathrm{Z}_{0:t}, \mathrm{U}_{0:t}, \boldsymbol{x}_0) = \frac{P(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{m}) P(\boldsymbol{x}_t, \boldsymbol{m} \mid \mathrm{Z}_{0:t-1}, \mathrm{U}_{0:t}, \boldsymbol{x}_0)}{P(\boldsymbol{z}_t \mid \mathrm{Z}_{0:t-1}, \mathrm{U}_{0:t})} \tag{3.6}$$

Jednadžbe (3.5) i (3.6) se koriste za rekurzivno izračunavanje vjerojatnosti definirane s (3.1).

Rješavanje SLAM problema se postiže pronalaženjem prikladnih rješenja za model promatranja (3.3) i model gibanja (3.4) s kojim je moće lako i dosljedno izračunati vjerojatnosti dane jednadžbama (3.5) i (3.6).

[1] Markovljev proces je stohastički proces u kojem je za predviđanje budućeg stanja dovoljno znanje samo trenutnog stanja.

3.3.1. EKF SLAM

SLAM algoritam baziran na Extended Kalman filteru (EKF SLAM) je prvi razvijeni algoritam za SLAM.

Ovaj algoritam je u mogućnosti koristiti jedino mape temeljene na značajkama (orijentirima). EKF SLAM može obrađivati jedino pozitivne rezultate kada robot primjeti orijentir, tj. ne može koristiti negativne informacije o odsutnosti orijentira u senzorskim očitanjima. Također, EKF SLAM pretpostavlja bijeli šum i za kretanje robota i percepciju (senzorska očitanja) [10].

EKF-SLAM opisuje model gibanja dan jednadžbom (3.4) s

$$\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{w}_t \tag{3.7}$$

gdje je $\boldsymbol{f}(\cdot)$ kinematički model robota, a $\boldsymbol{w}_t$ smetnje (engl. *disturbances*) u gibanju koje nisu u korelaciji, modelirane kao bijeli šum $\mathcal{N}(\boldsymbol{x}_t; \boldsymbol{0}, \boldsymbol{Q}_t)$. Model promatranja iz jednadžbe (3.3) je dan s

$$\boldsymbol{z}_t = \boldsymbol{h}(\boldsymbol{x}_t, \boldsymbol{m}) + \boldsymbol{v}_t \tag{3.8}$$

gdje $\boldsymbol{h}(\cdot)$ opisuje geometriju senzorskih očitanja, a $\boldsymbol{v}_t$ je aditivni šum u senzorskim očitanjima modeliran s $\mathcal{N}(\boldsymbol{z}_t; \boldsymbol{0}, \boldsymbol{R}_t)$.

Sada se primjenjuje EKF metoda opisana u poglavlju 3.2.1. za izračunavanje srednje vrijednosti i kovarijance združene a posteriori distribucije dane jednažbom (3.1) [22].

$$\begin{bmatrix} \hat{\boldsymbol{x}}_{t|t} \\ \hat{\boldsymbol{m}}_t \end{bmatrix} = \mathrm{E}\left[\begin{array}{c|c} \boldsymbol{x}_t & \\ \boldsymbol{m} & \boldsymbol{Z}_{0:t} \end{array} \right] \tag{3.9}$$

$$P_{t|t} = \begin{bmatrix} P_{xx} & P_{xm} \\ P_{xm}^T & P_{mm} \end{bmatrix}_{t|t} = \mathrm{E}\left[\begin{array}{c|c} \begin{pmatrix} \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \\ \boldsymbol{m} - \hat{\boldsymbol{m}}_t \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \\ \boldsymbol{m} - \hat{\boldsymbol{m}}_t \end{pmatrix}^T & \boldsymbol{Z}_{0:t} \end{array} \right] \tag{3.10}$$

Sada se računaju novi položaj robota prema jednadžbi (3.5) kao

$$\hat{\boldsymbol{x}}_{t|t-1} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1|t-1}, \boldsymbol{u}_t) \tag{3.11}$$

$$P_{xx,k|t-1} = \nabla \boldsymbol{f} P_{xx,t-1|t-1} \nabla \boldsymbol{f}^T + \boldsymbol{Q}_t \tag{3.12}$$

gdje je $\nabla \boldsymbol{f}$ vrijednost Jakobijana od $\boldsymbol{f}$ za $\hat{\boldsymbol{x}}_{k-1|k-1}$, te korekcija senzorskih mjerenja iz (3.6) prema

$$\begin{bmatrix} \hat{\boldsymbol{x}}_{t|t} \\ \hat{\boldsymbol{m}}_t \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{x}}_{t|t-1} & \hat{\boldsymbol{m}}_{t-1} \end{bmatrix} + \boldsymbol{W}_t \left[\boldsymbol{z}_t - \boldsymbol{h}(\hat{\boldsymbol{x}}_{t|t-1}, \hat{\boldsymbol{m}}_{t-1}) \right] \tag{3.13}$$

$$P_{t|t} = P_{t|t-1} - \boldsymbol{W}_t \boldsymbol{S}_t \boldsymbol{W}_t^T \tag{3.14}$$

gdje su $\boldsymbol{W}_t$ i $\boldsymbol{S}_t$ matrice ovisne o Jakobijanu od $\boldsymbol{h}$, te kovarijanci (3.10) i šumu $\boldsymbol{R}_t$.

U ovoj osnovnoj varijanti EKF-SLAM algoritma sve orijentire i matricu kovarijance je potrebno osvježiti pri svakom novom senzorskom očitanju, što može stvarati probleme u vidu zagušenja računala, ako imamo mape s po nekoliko tisuća orijentira. To je "popravljeno" razvojem novih rješenja SLAM problema temeljenih na EKF-u koji učinkovitije koriste resurse pa mogu raditi u skoro realnom vremenu [23, 24].

3.3.2. FastSLAM

FastSLAM algoritam [25,26] se, za razliku od EKF-SLAM-a, temelji na rekurzivnom Monte Carlo uzorkovanju (filter čestica) kako bi se doskočilo nelinearnosti modela i ne-normalnim distribucijama poza robota.

Direktna primjena filtera čestica nije isplativa zbog visoke dimenzionalnosti SLAM problema, pa se stoga primjenjuje Rao-Blackwellizacija. Općenito, združeni prostor je prema pravilu produkta

$$P(\mathrm{a}, \mathrm{b}) = P(\mathrm{b} \mid \mathrm{a})P(\mathrm{a}) \tag{3.15}$$

pa kada se $P(\mathrm{b} \mid \mathrm{a})$ može iskazati analitički, potrebno je uzorkovati samo $\mathrm{a}^{(i)} \sim P(\mathrm{a})$. Združena distribucija je predstavljena sa skupom $\{\mathrm{a}^{(i)}, P(\mathrm{b} \mid \mathrm{a}^{(i)}\}_i^N$ i statistikom

$$P(\mathrm{b}) \approx \frac{1}{N} \sum_{i=1}^{N} P(\mathrm{b}|\mathrm{a}^i) \tag{3.16}$$

koju je moguće dobiti s većom točnošću od uzorkovanja na združenom skupu.

Kod FastSLAM-a se promatra distribucija tokom čitave trajektorije od početka gibanja do sadašnjeg trenutka t, a prema pravilu produkta, opisanog jednadžbom (3.15), se može podijeliti na dva dijela, trajektoriju $\mathrm{X}_{0:t}$ i mapu $\boldsymbol{m}$ koja je nezavisna od trajektorije:

$$P(\mathrm{X}_{0:t}, \boldsymbol{m} \mid \mathrm{Z}_{0:t}, \mathrm{U}_{0:t}, \boldsymbol{x}_0) = P(\boldsymbol{m} \mid \mathrm{X}_{0:t}, \mathrm{Z}_{0:t})P(\mathrm{X}_{0:t} \mid \mathrm{Z}_{0:t}, \mathrm{U}_{0:t}, \boldsymbol{x}_0) \tag{3.17}$$

Ključna značajka FastSLAM-a je u tome da se, kako je prikazano u jednadžbi (3.17), mapa se prikazuje kao skup nezavisnih normalno distribuiranih značajki. FastSLAM se sastoji u tome da se trajektorija robota računa pomoću Rao-Blackwell filtera čestica i prikazuje otežanimuzorcima, a mapa se računa analitički, korištenjem EKF metode čime se ostvaruje znatno veća brzina u odnosu na EKF-SLAM.

3.4. Navigacija

Navigacijom se, u kontekstu mobilnih robota, može smatrati kombinacija tri temeljne operacije mobilnih robota: lokalizacija, planiranje putanje i izrada, odnosno interpretacija mape [2]. Kako su lokalizacija i mapiranje već prethodno obrađeni, u ovom dijelu će se dati pregled algoritama za planiranje putanje.

Postoje dvije "vrste" navigacije: globalna i lokalna. Globalnom navigacijom se smatra navigacija u poznatom prostoru (tj. prostoru za koji postoji izrađena mapa). Kod takve navigacije

robot može, korištenjem algoritama za planiranje putanje (npr. Dijkstra, A*), unaprijed isplanirati put do cilja bez da se pritom sudari s preprekama. S druge strane, lokalna navigacija nema saznanja o prostoru u kojem se robot giba i stoga je ograničena na mali prostor oko robota. Korištenjem lokalne navigacije robot obično samo izbjegava prepreke koje uočava korištenjem senzora. U većini primjena, globalna i lokalna navigacija se koriste zajedno, gdje algoritam za globalnu navigaciju odredi optimalnu putanju kojom će robot stići do odredišta, a lokalna navigacija ga vodi tamo, usput izbjegavajući prepreke koje se pojave a nisu vidljive na mapi.

U nastavku slijedi pregled algoritama za globalnu navigaciju. Većina algoritama se svodi na prikaz mape kao grafa, te se korištenjem algoritama za najkraći put traži optimalne putanje na tom grafu. Algoritmi za lokalnu navigaciju će biti obrađeni u poglavlju 4..

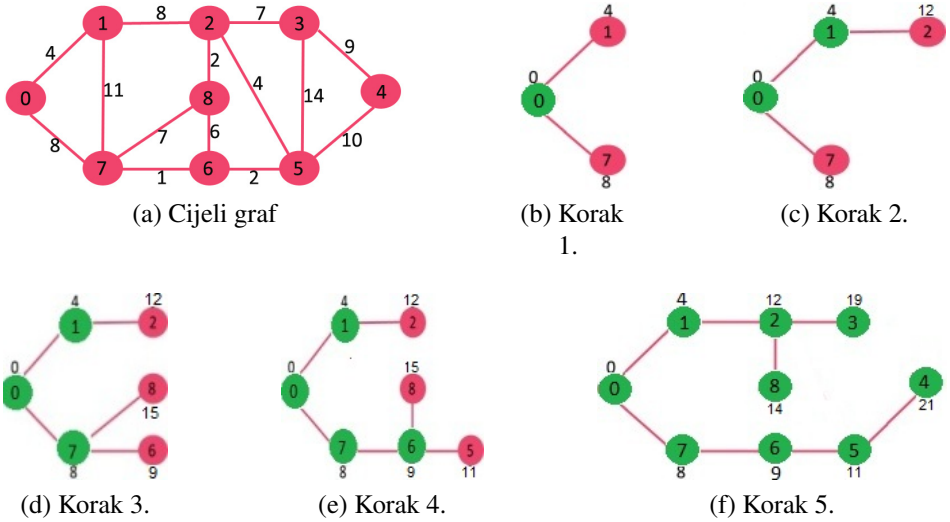
3.4.1. Dijkstra

Dijkstra algoritam [27] je algoritam koji za zadani početni vrh u usmjerenom grafu pronalazi najkraći put od tog vrha do svakog drugog vrha u grafu, a može ga se koristiti i za pronalazak najkraćeg puta do nekog specifičnog vrha u slučaju čega se algoritam zaustavlja kada je najkraći put pronađen. Osnovna varijanta ovog algoritma se izvršava s $O(|V|^2)$, gdje je $|V|$ broj vrhova grafa. Nešto kasnije je objavljena optimizirana verzija koja koristi Fibonnaccijevu gomilu (engl. *heap*), te koja se izvršava s $O(|E| + |V| \log |V|)$, gdje je $|E|$ broj stranica grafa. Temeljni algoritam je ilustriran primjerom na slici 3.2, te je korak po korak opisan tablicom 3.1.

Cijeli a lgoritam ilustriran gornjim primjerom se daje kako slijedi. Prvo se inicijalizira prazni skup S koji će sadržavati popis vrhova grafa koji su posjećeni. Nadalje, svim vrhovima grafa se incijaliziraju početne vrijednosti udaljenosti od zadanog početnog vrha D i to kao tako da se svima pridruži vrijednost beskonačno, osim vrha koji je odabran kao početni, kojemu se pridruži vrijednost udaljenosti 0. Sada se iterativno, sve dok svi vrhovi ne budu u skupu S, uzima jedan od vrhova koji nije u skupu S, a ima najmanju udaljenost. Potom se taj vrh dodaje u skup S, te se ažuriraju udaljenosti susjednih vrhova (ako su manji od trenutnih).

Iteracija	**Vrh**	S	D
0.	–	$\emptyset$	$[0 \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty]$
1.	0	$\{0\}$	$[0 \ 4 \ \infty \ \infty \ \infty \ \infty \ \infty \ 8 \ \infty]$
2.	1	$\{0, 1\}$	$[0 \ 4 \ 12 \ \infty \ \infty \ \infty \ \infty \ 8 \ \infty]$
3.	7	$\{0, 1, 7\}$	$[0 \ 4 \ 12 \ \infty \ \infty \ \infty \ 9 \ 8 \ 15]$
4.	6	$\{0, 1, 7, 6\}$	$[0 \ 4 \ 12 \ \infty \ \infty \ 11 \ 9 \ 8 \ 15]$
5.	5	$\{0, 1, 7, 6, 5\}$	$[0 \ 4 \ 12 \ \infty \ 21 \ 11 \ 9 \ 8 \ 15]$
6.	2	$\{0, 1, 7, 6, 5, 2\}$	$[0 \ 4 \ 12 \ 19 \ 21 \ 11 \ 9 \ 8 \ 15]$
7.	3	$\{0, 1, 7, 6, 5, 2, 3\}$	$[0 \ 4 \ 12 \ 19 \ 21 \ 11 \ 9 \ 8 \ 15]$
8.	4	$\{0, 1, 7, 6, 5, 2, 3, 4\}$	$[0 \ 4 \ 12 \ 19 \ 21 \ 11 \ 9 \ 8 \ 15]$

Tablica 3.1: Koraci Dijkstra algoritma iz primjera sa slike 3.2



(a) Cijeli graf (b) Korak 1. (c) Korak 2.

(d) Korak 3. (e) Korak 4. (f) Korak 5.

Slika 3.2: Ilustracija Dijkstra algoritma. Pretraživanje počinje u vrhu 0, te se iterativno pronalazi najkraći put do svakog pojedinačnog vrha, dok se putevi koji se pokažu duži od najkraćeg ne razmatraju. Izvor: GeeksforGeeks

3.4.2. A*

A* algoritam [28] je nadogradnja Dijkstra algoritma, te služi za istu namjenu on. Glavna prednost u odnosu na Dijkstru su bolje performanse koje se ostvaruju korištenjem heuristike za pretraživanje grafa. Izvršava se s $O(|E|)$, gdje je $|E|$ broj stranica grafa.

A* algoritam odabire put koji minimizira funkciju

$$f(n) = g(n) + h(n) \tag{3.18}$$

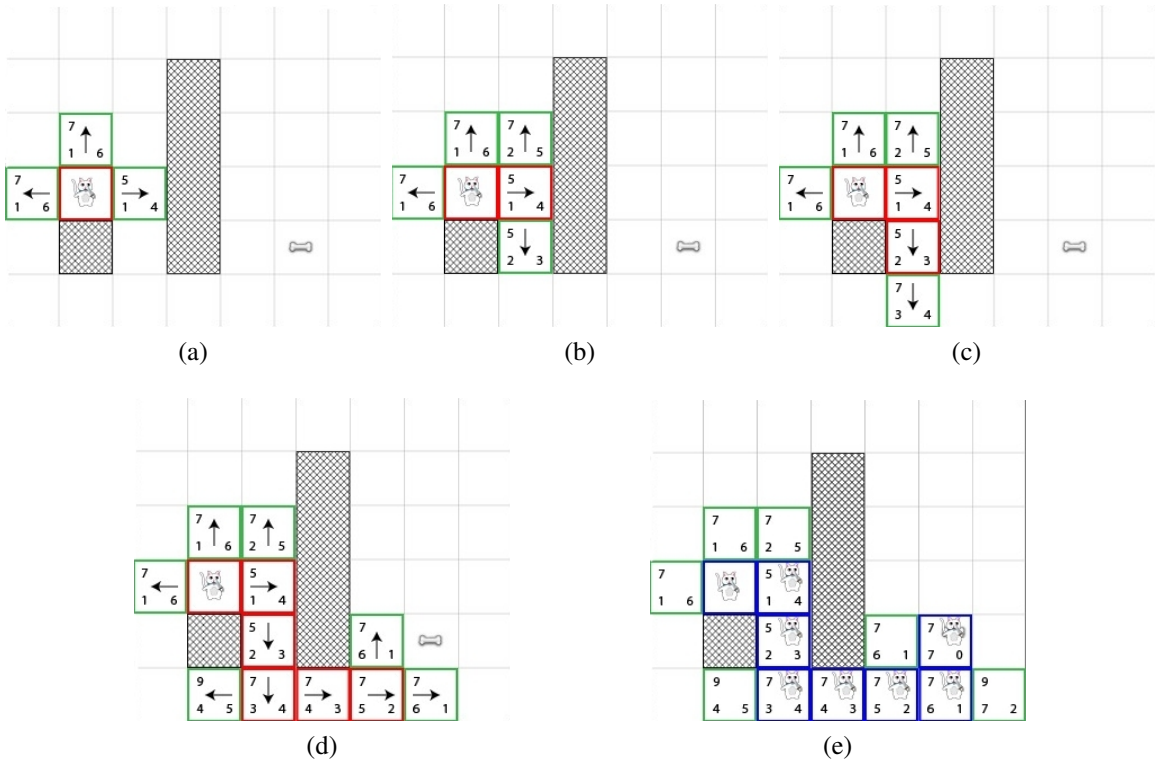
gdje je $g(\cdot)$ "cijena" gibanja od početne točke do trenutne, dok je $h(\cdot)$ procjena "cijene" gibanja od trenutne točke do odredišta, nazvana i heuristika. U svakoj iteraciji, algoritam za sljedeću točku u koju će krenuti odabire onu koja minimizira funkciju $f(\cdot)$.

Heuristiku se odabire, pritom vodeći računa da daje dobru procjenu, tj. da ne precjenjuje "cijenu" gibanja do odredišta. Procjena koju se daje mora biti manja od stvarne "cijene" ili u najgorem slučaju jednaka stvarnoj udaljenosti, a nikako ne smije biti veća. Jedna od najčešće korištenih heuristika je euklidska udaljenost, budući da je to fizikalno najmanja moguća udaljenost između dvije točke. Ovo je ilustrirano primjerom sa slike 3.3.

3.4.3. Probabilističko planiranje puta

Probabilističko planiranje puta (engl. *probabilistic roadmap*, PRM) [29] je algoritam za planiranje putanje robota u statičkom okolišu i primjenjiv je, osim mobilnih, i na ostale vrste robota. Planiranje putanje između početne i krajnje konfiguracije robota se sastoji od dvije faze: faza učenja i faza traženja.

U fazi učenja, konstruira se probabilistička struktura u obliku praznog neusmjerenog grafa $\mathcal{R} = (N, E)$ (N je skup vrhova grafa, a E je skup stranica grafa) u koji se potom dodaju vrhovi koji predstavljaju slučajno odabrane dozvoljene konfiguracije. Za svaki novi vrh $\boldsymbol{c}$ koji se



(a) (b) (c) (d) (e)

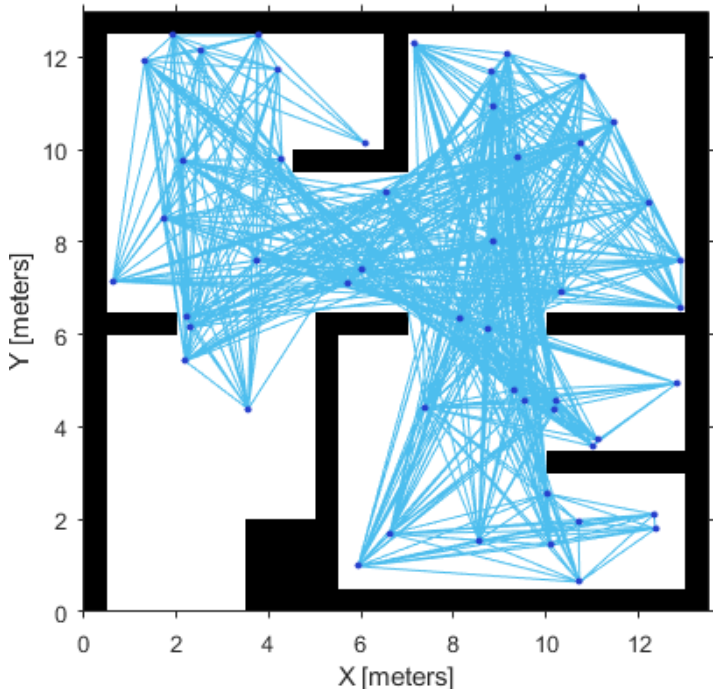
Slika 3.3: Primjer funkcioniranja A* algoritma, uz korištenje euklidske udaljenosti kao heuristike (nisu dane slike svih koraka). U svakoj od ćelija koje se razmatraju, broj u gornjem lijevom kutu je iznos funkcije f, u donjem lijevom funkcije g, a u donjem desnom je heuristika h. U svakom koraku kreće u ćeliju koja ima najmanju vrijednost funkcije f.

dodaje, ispituje se povezivost s već postojećim vrhovima u grafu korištenjem lokalnog planera. Ako se uspije pronaći veza (direktni spoj između vrhova bez prepreka), taj novi vrh se dodaje u graf i spaja s tim vrhove s kojim je poveziv za svaki vrh s kojim je pronađena moguća povezivost. Ne testira se povezivost sa svim vrhovima u grafu, već samo s određenim podskupom vrhova, čija je udaljenost od $\boldsymbol{c}$ do neke određene granične vrijednosti (koristeći neku unaprijed poznatu metriku D, primjerice Euklidsku udaljenost). Cijela faza učenja je prikazana algoritmom 3.2, a $D(\cdot)$ je funkcija metrike s vrijednostima u $\mathbb{R}^+ \cup \{0\}$, a $\Delta(\cdot)$ je funkcija koja vraća $\{0, 1\}$, ovisno može li lokalni planer pronaći putanju između vrhova. Opisani postupak je iterativan i u algoritmu 3.2 nije naznačeno koliko puta se ponavlja, što ovisi o odabranom broju komponenti grafa. Primjer grafa izrađenog na opisani način je dan na slici 3.4. U fazi traženja, u $\mathcal{R}$ se dodaju početna i krajnja konfiguracija, te se nekim od poznatih algoritama za traženje najkraćeg puta pronalazi optimalna putanja između početne i krajnje konfiguracije.

Opisani postupak planiranja putanje je, iako probabilistički, vrlo jednostavan i pogodan za primjenu na razne probleme u robotici. Jednostavno ga se može prilagoditi specifičnom problemu, primjeri traženju putanje za mobilne robote i to s odabirom prikladne funkcija D i lokalnog planera Δ. Slijedeći osnovni algoritam, izrađene su i razne varijante koje daju poboljšanja u određenim aspektima, te razne implementacije za primjene u određenim problemima. Posebno su za ovaj rad važne primjene na neholonomne mobilne robote [30, 31] te višerobotske sustave [32].

Algoritam 3.2. Probabilistic roadmap, faza učenja

$\mathcal{R} = (N, E)$
$N \leftarrow \emptyset$
$E \leftarrow \emptyset$
Ponavljaj:
  $\boldsymbol{c} \leftarrow$ slučajno odabrana slobodna konfiguracija robota
  $N_c \leftarrow$ skup kandidata za povezivanje s $\boldsymbol{c}$
  $N \leftarrow N \cup \{\boldsymbol{c}\}$
  Za svaki: $n \in N_c$ sortirano po redu rastućeg $D(\boldsymbol{c}, n)$
    Ako je $\neg komponente\_povezane(c, n) \wedge \Delta(\boldsymbol{c}, n)$ **onda**
      $E \leftarrow E \cup \{(c, n)\}$
      osvježi čvorove u grafu i njihove međusobne veze



Slika 3.4: Vizualizacija grafa dobivenog algoritmom 3.2. Tamnoplave točke su vrhovi grafa, dok svijetloplave linije predstavljaju stranice grafa. Izvor: MathWorks

4. Detekcija i izbjegavanje prepreka

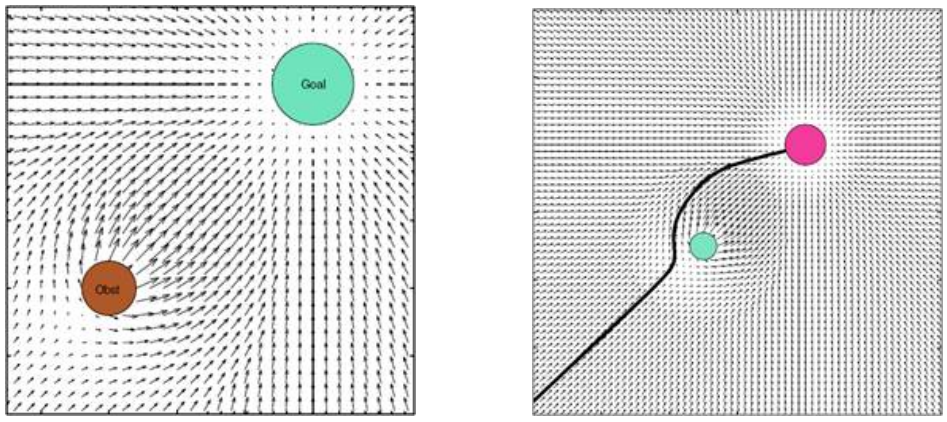
Iako je povezana s navigacijom robota, detekcija i izbjegavanje prepreka se obrađuje odvojeno od navigacije. Pod preprekama se ovdje podrazumjevaju objekti koji se ne nalaze na mapi temeljem koje se izrađuje plan putanje, ili se mapa ne koristi. Oni objekti koji se nalaze na mapi se uzimaju u obzir prilikom planiranja putanje, dok algoritmi za izbjegavanje prepreka se brinu da robot obiđe prepreke koje mu se nađu na putu prema zadanom cilju.

4.1. Statičke prepreke

4.1.1. Artificial Potential Fields

Pristup nazvan Umjetna potencijalna polja, (engl. *Artificial Potential Fields*, AFP) [33, 34, 35, 36] tretira robot kao elektron u zamišljenom umjetnom električnom polju, u kojem cilj "privlači" robota dok ga prepreke "odbijaju". Ova metoda se često koristi zbog svoje računske jednostavnosti.

Metoda funkcionira tako da se u svakom trenutku, na mjestu robota računa potencijal uzimajući u obzir da cilj privlači robota dok ga prepreke odbijaju, na način da, kako se robot približava prepreci tako odbojna sila raste. Stoga, robot će se u svakom trenutku gibati u smjeru inducirane sile (koja je vektorski zbroj privlačnih i odbojnih sila u cijelom prostoru). Ilustracija ove metode je prikazana na slici 4.1.



(a) (b)

Slika 4.1: Ilustracija metode potencijalnih polja. Slika (a) pokazuje kombinaciju privlačnog i odbojnog polja, dok slika (b) ilustrira putanju robota u prisutnosti odbojne i privlačne sile koje su rezultat potencijalnih polja. Izvor: McGill University, School of Computer Science

Sila koja djeluje na robot je dana s

$$\boldsymbol{F}(\boldsymbol{q}) = -\nabla(U_p(\boldsymbol{q}) + U_o(\boldsymbol{q})) \tag{4.1}$$

gdje je $\boldsymbol{q}$ pozicija i orijentacija robota u odnosu na globalni koordinatni sustav dana jednadžbom (2.1), $U_p(\boldsymbol{q})$ i $U_o(\boldsymbol{q})$ su privlačni odnosno odbojni potencijal koji djeluju na robota i za posljedicu imaju silu $\boldsymbol{F}(\boldsymbol{q})$. Potencijali su ovdje funkcije položaja i orijentacije robota.

Za $U_p(\boldsymbol{q})$ i $U_o(\boldsymbol{q})$ obično se uzimaju

$$U_p(\boldsymbol{q}) = \frac{1}{2}\|\boldsymbol{q} - \boldsymbol{q}_{\text{cilj}}\|^2 \tag{4.2}$$

$$U_o(\boldsymbol{q}) = \frac{1}{\|\boldsymbol{q} - \boldsymbol{q}^*\|} \tag{4.3}$$

gdje je $\boldsymbol{q}_{\text{cilj}}$ pozicija cilja, a $\boldsymbol{q}^*$ je pozicija najbliže prepreke.

Iako jednostavan, algoritam je često korišten u svom osnovnom obliku. Ipak, postoje situacije kada može zapeti i lokalnom minimumum, a može imati problema s uskim prolazima, pa su stoga predložena poboljšanja koja bi to izbjegla. Tako se u [37] za pronalaženje optimalne putanje koristi *simulated annealing*[1], dok se u [38] za istu namjenu koriste evolucijski algoritmi, te proširuju mogućnost korištenja na izbjegavanje pomičnih prepreka. Nadalje, autori rada [34] ga zbog gore navedenih problema napuštaju, te razvijaju novu metodu Vector Field Histogram, opisanu u nastavku.

4.1.2. Obstacle Restriction Method

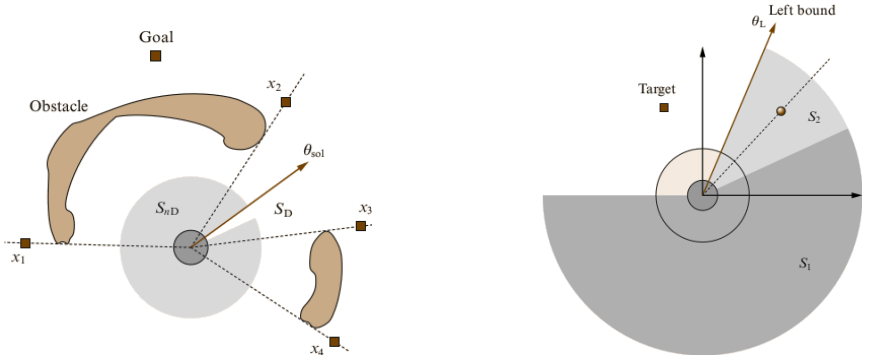
Obstacle Restiction Method (ORM) [39] metoda se odvija u tri koraka. U prvom se izračunava međucilj, ako je potrebno gibanje usmjeriti prema nekoj drugoj zoni osim cilja. Međuciljevi x_i se, prema slici 4.2a, nalaze između prepreka ili na krajevima prepreka. Nadalje, provjerava se je li moguće doći direktno do cilja od trenutne lokacije robota. Ako nije, odabire se najbliži međucilj. U drugom koraku se pridjeljuju ograničenja na gibanje s obzirom na prepreke i računa se skup kandidata za željeni smjer gibanja, prema slici 4.2b. U zadnjem koraku se od kandidata odabire jedan koji je najpovoljniji s obzirom na korištenu strategiju odabira. Kao rezultat se dobiva kutna brzina ω za ostvarivanje odabranog smjera gibanja, dok je linearna brzina v obrnutno proporcionalna udaljenosti od najbliže prepreke.

4.1.3. Dynamic Window Approach

Dynamic Window Approach (DWA) [40] koristi dinamiku robota na način da upravljačke signale kojima se kontrolira brzina i kutna brzina robota traži samo unutar manjeg okvira (engl. *dynamic window*), prostora koji je robotu dostupan u kratkom vremenskom intervalu koji slijedi nakon sadašnjeg trenutka. Na ovaj način se kao upravljački signali razmatraju samo brzine i kutne brzine koje daju trajektoriju koja omogućava sigurno i pravovremeno zaustavljanje.

DWA postupak se ponavlja iterativno, a jedna iteracija se sastoji od slijedećih koraka (koji imaju svoje podfaze). U prvom, u "prostoru" brzina se, od svih brzina (v, ω) izdvajaju se

[1] probabilistički algoritam za pronalaženje globalnog optimuma funkcije



(a) Međuciljevi sa željenim S_D i neželjenim S_{nD} smjerovima gibanja

(b) Ilustracija dva skupa neželjenih smjerova gibanja S_1 i S_2 s obzirom na zadani cilj

Slika 4.2: Ilustracija ORM metode. Izvor: [6]

one koje je moguće postići. Razmatraju samo one brzine koje robot može postići na temelju svojih fizikalnih i dinamičkih svojstava, te se odbacuju sve one koje ne garantiraju sigurno zaustavljanje prije najbliže prepreke na trenutnoj putanji. Drugi korak je optimizacija, u kojem se traži maksimum funkcije cilja

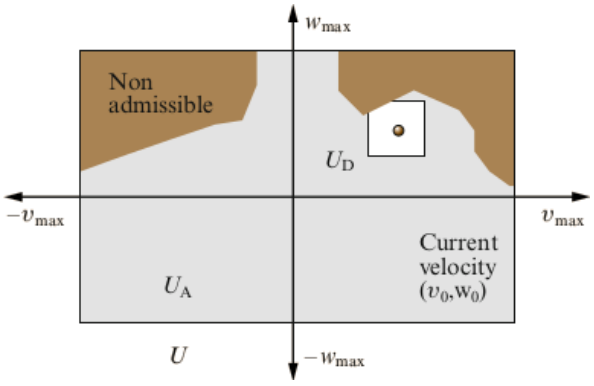
$$G(v, \omega) = \sigma(\alpha \cdot h(v, \omega) + \beta \cdot d(v, \omega) + \gamma \cdot V(v, \omega)) \tag{4.4}$$

gdje je $h(\cdot)$ mjera napredovanja prema cilju i poprima maksimalnu vrijednost ako se robot giba direktno prema cilju, $d(\cdot)$ je mjera udaljenosti od najbliže prepreke na trenutnoj trajektoriji i veća je što je robot bliže prepreci (tj. predstavlja "želju" robota da ide okolo prepreke), dok je $V(\cdot)$ mjera brzine prema naprijed. α, β i γ su konstante, a $\sigma(\cdot)$ je funkcija za izglađivanje ponderirane sume.

Skup brzina koje su dozvoljene V_d (iz prvog koraku) je dan s

$$V_d = \left\{(v, \omega) \mid v \leq \sqrt{2d(v, \omega) + \dot{v}_k} \ \wedge \ \omega \leq \sqrt{2d(v, \omega) + \dot{\omega}_k}\right\} \tag{4.5}$$

gdje su $\dot{v}_k$ i $\dot{\omega}_k$ akceleracije robota pri kočenju. Ovo je shematski prikazano na slici 4.3, pa je vidljivo koje kombinacije (v, ω) su dozvoljene (svjetlija zona na slici), a koje nisu dozvoljene jer rezultiraju sudarom (tamnjije zone slike).



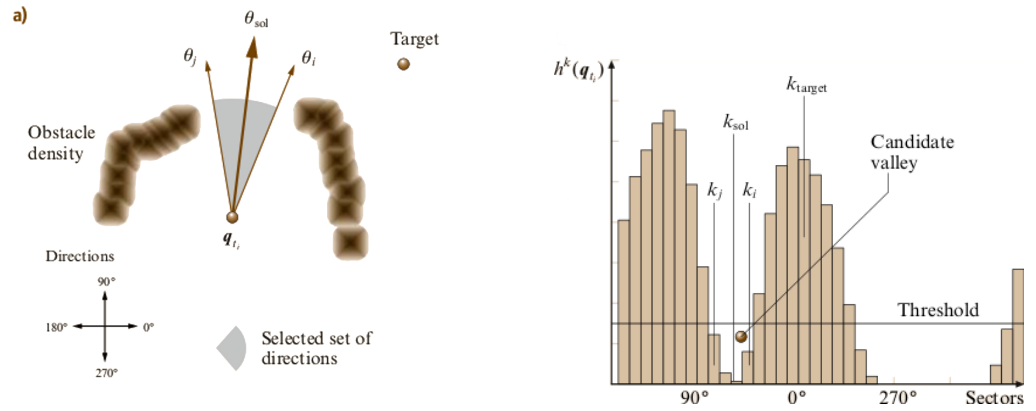
Slika 4.3: Prikaz dozvoljenih brzina i dinamičkog prozora u DWA. Izvor: [6]

Potencijalni nedostatak ovog algoritma je da u nekim slučajevima robot zapne u lokalnom minimumu gdje nema dohvatljivih trajektorija koje robotu dozvoljavaju translacijsko gibanje. Ovaj problem je rješen tako što je tada robotu dozvoljeno rotiranje u mjestu sve dok ne mu ne bude moguće translacijsko gibanje. Ovo rezultira suboptimalnim trajektorijama, ali se događa dovoljno rijetko da ne utječe bitno na performanse algoritma u cjelini.

4.1.4. Vector Field Histogram

Histogram vektorskih polja (engl. *Vector Field Histogram*, VFH) [41] je algoritam za izbjegavanje nepoznatih prepreka (tj. onih kojih nema na mapi) u realnom vremenu koji istovremeno i vodi robot prema zadanom cilju. Razvijen je kao odgovor na nedostatke algoritma umjetnih potencijalnih polja, a sastoji se od dva koraka.

U prvom se oko trenutne pozicije robota, a na temelju podataka prikupljenih pomoću senzora udaljenosti, konstruira polarni histogram, koji je podjeljen na određeni broj sektora fiksne širine (recimo, 72 sektora k, svaki širok po 5°), te se svakom od sektora pridjeljuje vrijednost koja predstavlja gustoću prepreka (engl. *polar obstacle density*, POD). Dobiveni histogram obično ima ekstreme (maksimumi predstavljaju smjerove s visokim POD, dok minimumi predstavljaju niski POD). Uzima se skup smjerova-kandidata za nastavak gibanja kojima je gustoća manja od zadane granične vrijednosti a koji je najbliže zadanom smjeru gibanja (na slici 4.4b je to predstavljeno minimumom histograma). U drugom koraku se odabire najprikladniji sektor za smjer gibanja (prikazano na slici 4.4a).



(a) Izračunavanje smjera gibanja korištenjem VFH

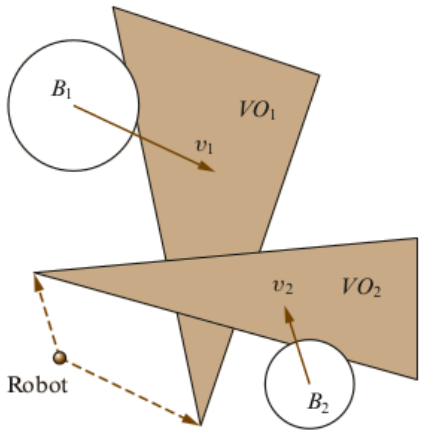
(b) Histogram gustoće prepreka s označenim sektorom k_{sol} koji sadrži rješenje

Slika 4.4: Ilustracija VFH metode. Izvor: [6]

VFH je metoda koja je prilagođena obilaženju prepreka koje su definirane probabilistički, što ga čini pogodnim za korištenje u senzorima s nesigurnošću (engl. *uncertainity*). Jedno unaprijeđenje VFH algoritma je opisano u [42] i nazvano VFH*, a temelji se na tome da verificira je li planirana predloženja putanja vodi robot oko prepreke, a ta verifikacija se obavlja pomoću A* algoritma za planiranje puta.

4.2. Dinamičke prepreke

U kontekstu izbjegavanja prepreka, dinamičkim preprekama se smatraju one prepreke kojima je njihova pozicija (i orijentacija) vremenski promjenjiva (tj. prepreke koje se kreću).



Slika 4.5: VO skup nesigurnih trajektorija uz prisutnost dvije prepreke. Upravljački signal izvan granica skupova VO_1 i VO_2 rezultira gibanjem bez sudara s preprekama. Izvor: [6]

Načelno, sve metode za izbjegavanje statičkih prepreka se mogu koristiti i za izbjegavanje dinamičkih prepreka, ali njihova uspješnost obično ovisi o tome koliko često se osvježavaju senzorske informacije i izračuni, te gibaju li se pomični objekti brzo ili sporo. Međutim, postoje i metode koje uzimaju u obzir i kretanje prepreka, koje će biti obrađene u nastavku.

4.2.1. Velocity Obstacle

Velocity Obstacle (VO) [43] je jedna od najpoznatijih i najčešće korištenih metoda za izbjegavanje dinamičkih prepreka je vrlo slična metodi DWA, uz razliku da se za računanje sigurnih trajektorija (onih koje ne rezultiraju sudarima) uzimaju u obzir i brzine kretanja prepreka. Konstruira se "konus sudara" (engl. *collision cone*) koji je skup definiran s

$$CC_i = \left\{ \boldsymbol{u}_i \mid \lambda_i \bigcap B_i \neq \emptyset \right\} \tag{4.6}$$

gdje je $\boldsymbol{u}$ upravljački signal robota, $\boldsymbol{v}_i$ je brzina kretanja prepreke, λ_i je smjer jediničnog vektora $\boldsymbol{u}_i = \boldsymbol{u_i} - \boldsymbol{v}_i$, a B_i je prostor kojeg zauzima prepreka i. Velocity obstacle se onda dobija prema

$$VO_i = CC_i \oplus \boldsymbol{v}_i \tag{4.7}$$

Skup svih nesigurnih trajektorija je ilustriran slikom 4.5, a dan je s

$$\bar{\mathcal{U}} = \cup_i VO_i \tag{4.8}$$

5. Umjetna inteligencija i učenje u mobilnoj robotici

Roboti su inicijalno bili korišteni za obavljanje jednostavnih zadataka. Međutim, razvojem umjetne inteligencije, omogućeno im je da "uče" nova ponašanja ili akcije, kako bi kada se jednom nađu u nepoznatoj situaciji mogli reagirati na prikladan način. Umjetna inteligencija omogućava robotima da samostalno obavljaju i složenije zadatke uz minimalnu ili nikakvu intervenciju čovjeka.

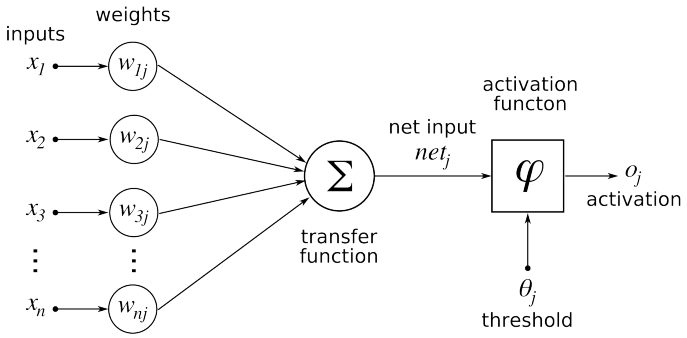
U zadnje vrijeme ta disciplina dobiva na popularnosti, zbog velike mogućnosti primjene u različitim scenarijima korištenja, prvenstveno u raznim aspektima upravljanja autonomnim vozilima ili autonomnom obavljanju zadataka kod servisnih robota (čistači, paletari, kosilice i sl.).

5.1. Metode umjetne inteligencije

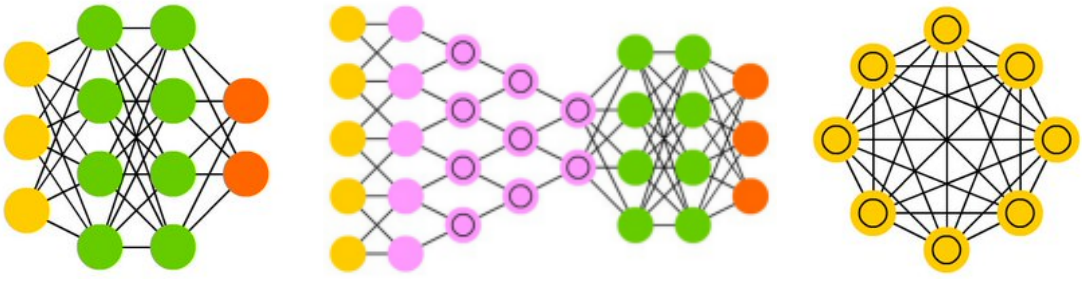
5.1.1. Neuronske mreže

Neuronske mreže su model strojnog učenja koji je inspiriran biološkim neuronskim mrežama (veze između neurona u mozgu životinja). Općenito, neuronske mreže su dizajnirane tako da rade na način sličan mozgu, a implementirane su na digitalnim računalima. Pomoću njih je moguće modelirati određene nelinearne funkcije/zadatke kroz proces učenja.

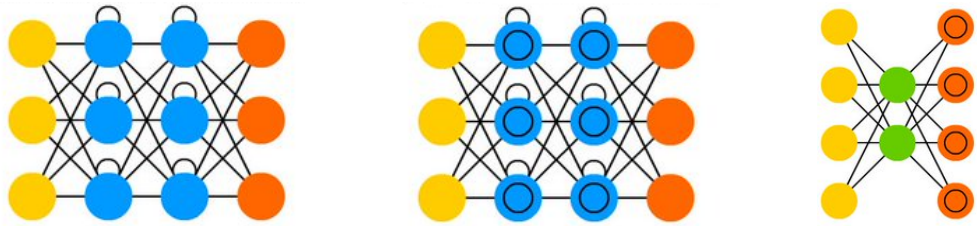
Neuronska mreža se sastoji od umjetnih neurona koji su međusobno povezani vezama koje imaju određenu „težinu" koja se može mijenjati/ugađati, što neuronskim mrežama daje sposobnost učenja i čini ih prilagodljivima ulaznim signalima. Ta težina veza kojima su neuroni međusobno povezani im daje sposobnost učenja. Shematski prikaz neurona je dan na slici 5.1.



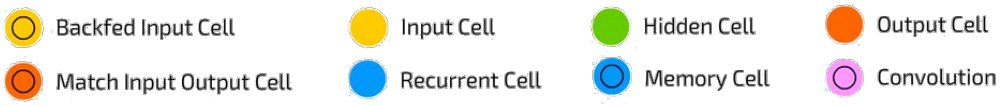
Slika 5.1: Shematski prikaz umjetnog neurona



(a) Višeslojni perceptron (b) Konvolucijska neuronska mreža (c) Hopfield mreža



(d) Mreža s povratnom vezom (e) Mreža s povratnom vezom i memorijom (f) Autoenkoder



(g) Legenda

Slika 5.2: Neke od arhitektura neuronskih mreža

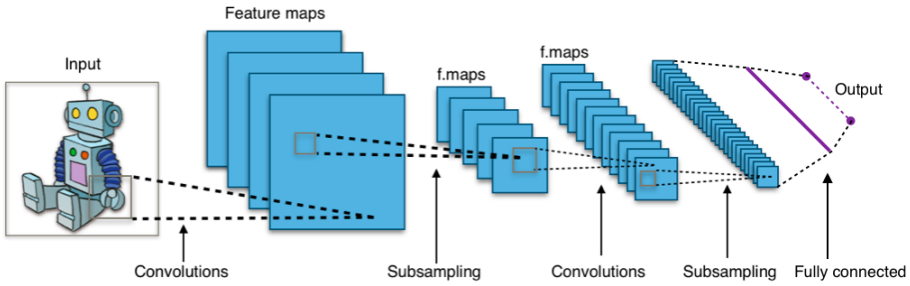
Neuron, koji je osnovni građevni element neuronske mreže, je matematička funkcija koja na osnovu vrijednosti ulaza daje vrijednost na izlazu. Vrijednost na izlazu određena je vrijednostima na ulazima, te težinama veza θ_i na koje se primjenjuje funkcija aktivacije. Kao funkcije aktivacije $\varphi(\cdot)$ se najčešće koristi logistički sigmoid i hiberbolni tangens. Izlaz se daje prema

$$y(\boldsymbol{x}) = \varphi(\Theta^T x) = \varphi\left(\sum_{i=0}^{n} \theta_i x_i\right) \tag{5.1}$$

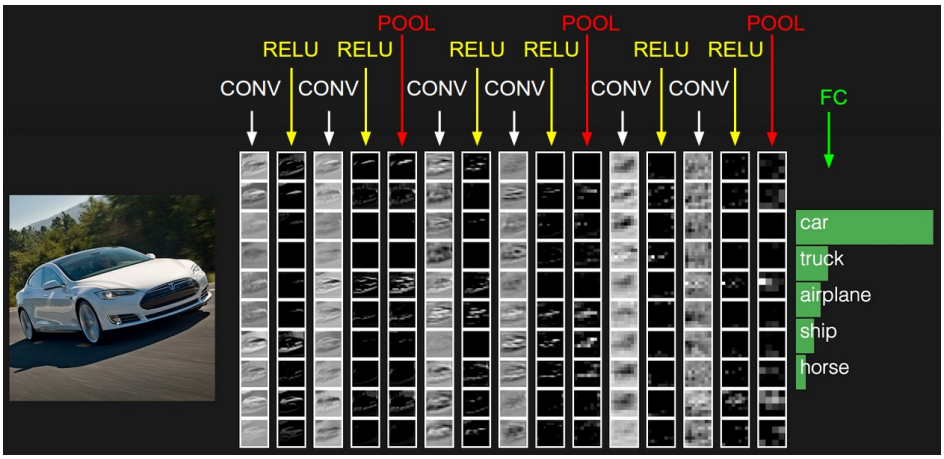
Osnovna i najčešće korištena klasa neuronskih mreža je tzv. višeslojni perceptron (engl. *multilayer perceptron*, MLP), koji je prikazan na slici 5.2a. Sastoji se od ulaznog sloja, jednog ili više skrivenih slojeva te izlaznog sloja. U svakom od slojeva se nalaze neuroni, a svaki neuron u skrivenom je povezan s drugima na način da je izlaz iz neurona povezan sa svim neuronima u slijedećem sloju. Općenito, neuronska mreža koja ima više od jednog skrivenog sloja (bilo kojeg tipa) se naziva duboka neuronska mreža (engl. *deep neural network*, DNN), dok se mreža s jednim skrivenim slojem naziva plitka neuronska mreža (engl. *shallow neural network*).

Osim opisanog osnovne arhitekture neuronske mreže, u robotici se još koriste i druge arhitekture, primjerice konvolucijske neuronske mreže i neuronske mreže s povratnom vezom, te još mnogo drugih. Važno je naglasiti da različite arhitekture neuronskih mreža ne konkuriraju jedni drugima, već se koriste za rješavanje različitih klasa problema. Neke od češće korišćenih arhitektura su prikazane na slici 5.2.

Konvolucijske neuronske mreže (engl. *convolutional neural networks*, CNN) [44, 45, 46] su klasa dubokih neuronskih mreža koje se koriste uglavnom za obradu i klasifikaciju vizualnih podataka (tj. slika). One se sastoje od niza konvolucijskih slojeva i slojeva udruživanja (engl. *pooling layer*) koji za cilj imaju smanjivanje ulazne slike i izvlačenje ključnih svojstava iz vizualnih podataka koji se mogu koristiti za učenje. Ti podaci onda ulaze u potpuno povezani sloj (skriveni sloj korišten kod višeslojnih klasičnih neuronskih mreža kod kojih je svaki neuron povezan sa svim neuronima u slijedećem sloju). Shematski prikaz je dan na slici 5.3.



(a) Arhitektura konvolucijske mreže



(b) Primjer CNN za klasifikaciju s izgledima filtera u konvolucijskim slojevima mreže

Slika 5.3: Arhitektura i primjer klasifikacije za CNN

Neuronske mreže s povratnom vezom (engl. *recurrent neural networks*, RNN) su klasa neuronskih mreža u kojima veze među neuronima tvore usmjereni graf, što omogućuje da koristeći njih modeliramo vremenske nizove. Te mreže mogu koristiti svoje interno stanje (memorija) za obradu ulaznih nizova podataka, tj. da izlaz iz mreže ovisi o trenutnom stanju ulaza kao i o nekom unaprijed odabranom broju stanja ulaza i izlaza iz prethodnih trenutaka (za razliku od višeslojnog perceptrona, čiji izlaz ovisi samo o trenutnom stanju ulaza), što ih čini pogodnim za rješavanje određenih vrsta problema koji su u svojoj prirodi vremenski sljedovi, poput prepoznavanja rukopisa [47] ili govora [48].

5.1.2. Reinforcement learning

Reinforcement learning je računalni pristup razumijevanju i automatizaciji učenja usmjerenog na cilj (engl. *goal-directed learning*) i donošenja odluka [49], te je trenutno najpopularnija metoda za učenje novog ponašanja agenta (robota). Sastoji se u tome da agent uči kako određene situacije preslikati u akcije tako da dobije maksimalnu numeričku nagradu,

ali s pristupom koji je drukčiji od tradicionalnih metoda strojnog učenja. Ovdje agent sam mora otkriti koje akcije donose najveću nagradu u određenim situacijama, a otkriva na način da sam proba tu akciju (metoda pokušaja i pogreške). Nagrada koju agenti dobivaju je kumulativna, tako da je čest slučaj da se put do veće nagrade sastoji od više akcija koje agent poduzima u vremenskom slijedu.

Osim agenta i okoliša, osnovni elementi reinforcement learning pristupa su smjernice (engl. *policy*), funkcija nagrade (engl. *reward function*), funkcija vrijednosti (engl. *value function*) i model okoliša [49]. Smjernicama se definira ponašanje agenta u određenom stanju, tj. koje akcije može poduzeti u tom stanju. Funkcija nagrade definira cilj reinforcement learning problema, tj. svakom paru stanja i akcije dodjeljuje određenu numeričku vrijednost koja opisuje "poželjnost" tog stanja, a agentov zadatak je da dugoročno maksimizira nagradu. Funkcija vrijednosti definira što je dobro i poželjno polazeći od sadašnjeg trenutka tako da analizira vrijednost trenutnog stanja što se tiče nagrade za koju se očekuje da će je agent prikupiti u budućnosti polazeći iz tog stanja. Za razliku od funkcije nagrade ova funkcija pokazuje dugoročnu poželjnost pojedinog stanja uzimajući u obzir i stanja koja će vjerojatno slijediti ubuduće kao i njihove nagrade.

Reinforcement learning je u [50] definiran kao metoda koja u robotiku donosi "alate za dizajn sofisticiranih i teško programabilnih ponašanja". U ovom preglednom radu se daje pregled state-of-the-art *reinforcement learning* metoda korištenih u robotici, te se navode otvorena pitanja i izazovi koji tek trebaju biti riješeni.

Dobar primjer primjene *reinforcement learning* metode je [51], u kojem je razvijen umjetni inteligentni agent koji korištenjem *reinforcement learning* metode može naučiti ponašanje i upravljanje slično čovjekovom direktno iz senzorskih podataka. Pristup je testiran na računalnim igrama te su performanse razvijenog agenta nadišle performanse profesionalnog testera računalnih igara.

5.2. Inteligentna navigacija

Pod inteligentnom navigacijom u mobilnoj robotici se smatra mogućnost prepoznavanja i razumijevanja nepoznate i nestrukturirane okoline, te sposobnost samostalnog izvršavanja navigacijskih zadataka prema željenom cilju unutar te okoline.

Neuronske mreže se već duže vrijeme koriste za različite vidove navigacije u robotici. U novije vrijeme, s ponovnim rastom popularnosti neuronskih mreža, razvijaju se novi i inovativni pristupi navigaciji koristeći razne varijante neuronskih mreža (duboke unaprijedne neuronske mreže, konvolucijske neuronske mreže, neuronske mreže s povratnom vezom - engl. *recurrent neural networks*).

Među prvim primjenama neuronskih mreža za navigaciju mobilnog robota je praćenje ceste robotiziranim automobilom [52]. Na ulaz se dovodi smanjena slika s kamere na vozilu (30×32 piksela) što rezultira s 960 neurona u ulaznom sloju. Koristi se samo jedan skriveni sloj s 5 neurona koji su svi povezani sa svim neuronima u ulaznom i izlaznom sloju. Izlazni sloj ima 30 neurona od kojih oni u sredini su zaduženi za kretanje ravno, dok oni lijevo i desno od toga su zaduženi za skretanje, što je neuron više lijevo ili desno, skretanje je oštrije. Mreža je trenirana tako da se umjesto aktivirana jednog neurona u izlaznom sloju, aktivira više neurona oko onog smjera gibanja koji će održati vozilo na sredini ceste, prema

normalnoj razdiobi. Ovakav pristup je objašnjen s činjenicom da korištenjem obične klasifikacije, gdje se aktivira samo 1 izlaz može rezultirati drugačijim izlazom za malene promjene u ulazu, pa se koristi ovaj pristup da bi se takvo ponašanje izbjeglo. Mreža je trenirana korištenjem *backpropagation* algoritma, a korišteni su primjeri za treniranje mreže iz dva izvora. U prvom, koriste se umjetno napravljene slike s pripadajućim izlaznim vektorima, dok se u drugom koriste stvarne slike zabilježene dok čovjek-vozač upravlja vozilom, što za posljedicu ima da neuronska mreža "imitira" ponašanje čovjeka-vozača.

Također je istražena i navigacija s izbjegavanjem prepreka uz samostalan povratak mobilnog robota na stanicu za punjenje baterije [53]. Autori koriste neuronske mreže s povratnom vezom za upravljanje fizičkim mobilnim robotom, te genetički algoritam za dobivanje optimalne arhitekture spomenute neuronske mreže.

Iako su razvijeni metode navigacije temeljene na različitim senzorima, u novije vrijeme vizualna navigacija (ona koja se temelji na visualnim senzorima, prvenstveno kameri montiranoj na robotu) dobija na popularnosti, budući da vizualni senzori prikupljaju velike količine podataka koji obiluju informacijama, pa ih je stoga moguće koristiti za veliki broj različitih primjena u sferi navigacije robota. Za obradu i analizu vizualnih informacija i izvlačenje određenih podataka iz njih pogodne su konvolucijske neuronske mreže [44, 46]. Primjena ima jako puno, pogotovo u novije vrijeme kada su konvolucijske mreže postale state-of-the-art metoda za klasifikaciju i prepoznavanje predložaka u slikovnim podacima.

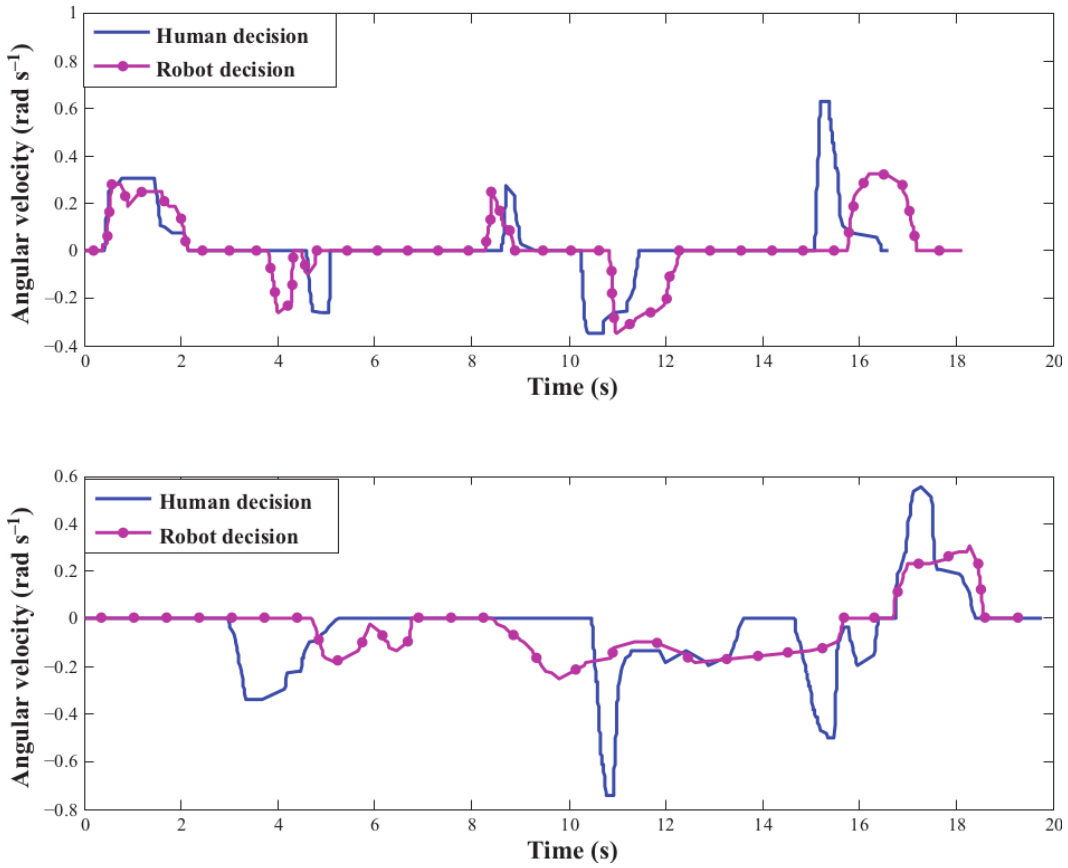
Konvolucijske mreže su korištene u [54] za autonomno reaktivno izbjegavanje prepreka kod *off-road* robota. Mreža na ulazu dobiva "sirove" slike s dvije kamere montirane na robotu, te na izlazu daje kuteve skretanja, a trenirana je s podacima koji su prikupljeni dok je čovjek upravljao robotom i to na različitim vrstama terena, osvjetljenja i prepreka, te po različitim vremenskim uvjetima. Rezultati testiranja dobivene mreže su pokazali 35.8% pogrešno klasificiranih uzoraka, što izgleda puno, ali kada se u obzir uzme da je istu prepreku moguće izbjeći na više načina (iako mreža kaže da su ti drugi pogrešni), te iako sustav ne obavi skretanje u identičnom trenutku od onoga kada bi to napravio čovjek, stvarni rezultati su puno bolji nego što ova brojka sugerira. S druge strane, u [55] je predstavljena metoda za daljinsku klasifikaciju terena korištenjem stereo vida, također korištenjem konvolucijskih mreža, kako bi bilo unaprijed odrediti je li neki teren prikladan za planiranje puta preko njega. Mreža klasificira teren u pet kategorija (super prohodan, prohodan, put (footline), prepreka i super prepreka). Mreža je trenirana na samonadzirući način (self-supervised).

5.2.1. Biološki inspirirana navigacija

U posljednje vrijeme se razvijaju pristupi navigaciji koji pokušavaju imitirati procese u mozgu bioloških entiteta kako bi se ostvarilo ponašanje robota koje je što sličnije ponašanju živih organizama. Većina radova u području biomimetičke navigacije se temelji na oponašanju lokalizacijskih i navigacijskih neurona u hipokamplanom kompleksu glodavaca, a sastoji se od više vrsta neurona koji imaju različiti zadaće i okidaju pod različitim uvjetima. Neuroni za lokalizaciju (engl. *place cells*) okidaju kada je glodavac na određenoj lokaciji unutar svog prostora u kojem luta i ne ovise o orijentaciji tijela. Orijentacijski neuroni (engl. *head direction cells*) su invarijantni od pozicije i svaki ima preferirani smjer u odnosu na trenutnu orijentaciju štakora. Granični neuroni (engl. *border cells*) okidaju u slučaju nekakvih granica ili barijera, dok su mrežni neuroni (engl. *grid cells*) [56] koji u principu navigacijski neuroni.

Jedan od algoritama razvijenih na temelju računalnog modela hipokampalnog kompleksa glodavaca je RatSLAM [57]. Računalni model hipokampalnog kompleksa je predstavljen u [58, 59, 60]. Temelji se na privlačnim mrežama (engl. *attractor networks*, koje se temelje na RNN, a karakterizira ih ustaljeno stanje koje je stabilno. Glavna prednost korištenja ovakvog sustava za lokalizaciju i mapiranje u konzistetnim reprezentacijama okoline. Iako ovaj sustav mapiranja nije kartezijski, prikaz prostore se može nazivati mapom zbog konzistentnosti reprezentacije. Ovo istraživanje su slijedile razrade kompleksniji slučajeva korištenja RatSLAM algoritma. Tako je u [61] korišten RatSLAM sa smanjenim brojem lokalizacijskih neurona. Kako smanjenjem broja neurona padaju performanse, uvodi se komponenta nazvana mapa iskustva (engl. *experience map*) koja daje kartezijske reprezentacije okoline, kombinirana s informacijama sa senzora te omogućava navigaciju prema cilju čak i uz velik broj sudara na originalno planiranoj putanji. Ovakav pristup je također pokazao bolje performanse u velikim prostorima, u odnosu na originalni pristup. Naknadno je u [62] prezentirana metoda koja umjesto RatSLAM jezgre koristi novodizajnirani filter koji ubrzava operaciju zadržavajući točnost i robustnost RatSLAM-a.

Također, postoje i pristupi koji su inspirirani načinom na koji čovjek donosi odluke, pa je u [63] prikazan pristup autonomnom pretraživanju prostora korištenjem dubokih neuronskih mreža. Pristup koristi konvolucijsku mrežu (konvolucijski sloj+*pooling* sloj u tri iteracije, te dva potpuno povezana sloja) koja je zadužena za klasifikaciju različitih scena (okoliša), prepoznavanje objekata i donošenje odluka. Izlazi iz mreže su upravljački signali. Ovo predstavlja višu razinu inteligencije od jednostavne klasifikacije (koja je osnovna namjena konvolucijskih mreža), jer u procesu donošenja odluka se negdje u mreži nalazi prepoznavanje objekata (klasifikacija). Za donošenje odluka i generiranje upravljačkog signala su korištena dva pristupa. U prvom, izlaze generira softmax klasifikator. Izlazi su diskretizirani u 5 koraka (skroz lijevo, polu-lijevo, ravno, polu-desno, desno). Drugi pristup karakterizira donošenje odluka na temelju pouzdanosti. Za svaki od 5 izlaza se određuje koeficijent pouzdanosti, a za izlaze za koje je pouzdanost veća, robot će težiti tome da ide u smjeru koji sugerira taj izlaz, istovremeno poštujući kompromis između više različitih izlaza. Pristupi su testirani na stvarnom robotu. Predstavljene metode su vrlo slične načinu na koji čovjek pretražuje prostor, što je pokazanu i u eksperimentu u kojem su uspoređene odluke koje donosi čovjek s onima robota i koje su pokazale visok stupanj sličnosti, kao što je ilustrirano slikom 5.4.



Slika 5.4: Usporedba odluka koje donosi robot s čovjekovima. Gornja slika prikazuje pristup sa softmax klasifikatorom, dok donja pokazuje pristup temeljen na pouzdanosti.

6. Upravljanje mobilnim robotom s udaljene lokacije

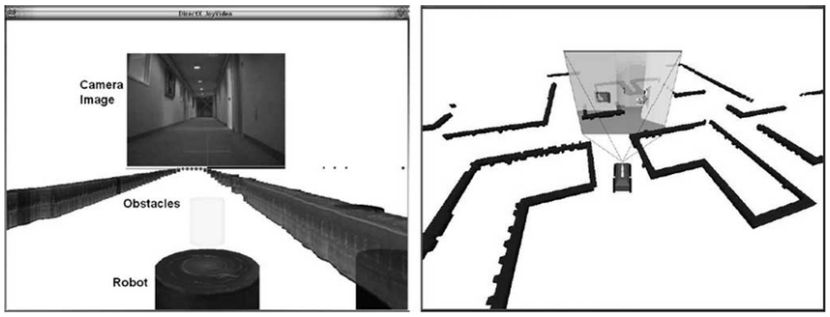
Ponekad je potrebno da čovjek preuzme kontrolu nad mobilnim robotom u autonomnom načinu rada, bilo da obavi zadatak koji robot ne može obaviti u autonomnom načinu ili kada algoritmi za autonomno upravljanje zakažu. Upravljanje se može ostvariti s udaljene lokacije, što se obično u literaturi naziva teleoperacija ili teleupravljanje, a obično se ostvaruje putem internet veze. Jedan od ključnih parametara za uspješnu i sigurnu teleoperaciju je svjesnost operatora o stanju robota i okoline u kojoj se robot kraće, kao i posljedica akcija robota na samog robota i na okolinu, što se u literaturi naziva svjesnost o situaciji (engl. *situational awareness*) [64, 65]. Poboljšanjem ovog parametra se ostvaruju bolje performanse u teleoperaciji, a to se postiže korištenjem odgovarajućih senzora i teleoperacijskih sučelja.

Teleoperaciju se, prema načinu upravljanja robotom može podijeliti na teleoperaciju niske razine (engl. *low-level*) i teleoperaciju visoke razine (engl. *high-level*). U teleoperaciju niske razine spada upravljanje robotom na daljinu u klasičnom smislu, gdje operater direktno upravlja robotom putem te percipira posljedice akcija putem teleoperacijskog sučelja. Najčešće se u literaturi pod pojmom teleoperacije podrazumjeva ovakva vrsta upravljanja robotom s udaljene lokacije.

Teleoperacijom visoke razine se može smatrati kada operater ne upravlja robotom direktno, već robotu zadaje zadatke visoke razine, a robotovi algoritmi su zaduženi za razumijevanje zadatka, te za autonomno izvršavanje akcija u skladu s time. Prednost ovakvog pristupa je što zahtjeva mnogo manje čovjekovog rada u odnosu na klasični pristup, a utjecaj latencije na performanse je bitno smanjen, jer se čak i u prisutnosti visoke latencije robot može nastaviti sa svojim zadatkom (jer se algoritmi za autonomnu operaciju izvršavaju na strani robota). Načina na koji se kod ovakvog pristupa mogu zadavati zadaci robotu su razni. Tako se u [66] koriste verbalne komande robotu, koji je opremljen algoritmima za prepoznavanje govora, koji mora razumjeti i poduzeti određene akcije. U [67] robotu se zadaci mogu zadati putem jednostavnog sučelja na tabletu ili računalu, te u slučaju zakazivanja autonomije ili nepostojanja funkcionalnosti za autonomno obavljanje određenog zadatka, može se preći na nižu razinu teleoperacije i preuzeti direktno upravljanje robotom.

6.1. Senzori i sučelja

Za dobivanje informacija o stanju robota i njegovoj okolini se koriste senzori montirani na robotu. Prvenstveno se tu koristi video kamera, budući da informacija u vidu slike korisniku najbolje opisuje okolinu robota, ali se mogu koristiti i drugi senzori, poput ultrazvučnih, LiDAR i sl. kao dodatna informacija operateru kako bi mu se olakšalo upravljanje robotom. S druge strane su tu teleoperacijska sučelja, koja se koriste za interakciju između robota i operatera, a koja imaju dva zadatka. Prvi je da podatke prikupljene senzorima prikazuju



(a) (b)

Slika 6.1: Primjeri 'ekoloških' sučelja koja kombiniraju više informacija integriranih u jednu sliku. Izvor: [70]

operateru i to tako da su prikazani na način koji će korisniku maksimalno olakšati njihovu interpretaciju i korištenje, dok je drugi da osigura način na koji će korisnik moći upravljati aktivnostima robota. Stoga se posebna pažnja posvećuje efikasno dizajniranim sučeljima i razrađuju se nove metode izrade sučelja te načini i rasporedi prikaza podataka na njima.

U [68] je dan detaljan pregled koje sve vrste sučelja za upravljanje vozilima s udaljene lokacije postoje. Najpoznatija i najjednostavnija je tzv. direktna metoda, u kojem operater upravlja robotom putem upravljača (joystick), a povratnu informaciju dobiva putem kamere montirane na robotu. Multimodalna i multisenzorska sučelja osiguravaju više od jednog načina upravljanja, odnosno fuziju podataka sa više različitih senzora u cilju dobivanja šire slike u odnosu na korištenje samo jednog senzora. Nadalje, kod nadziranog upravljanja (engl. *supervisory control*) operater razloži kompleksniji zadatak na niz jednostavnijih zadataka koje robot onda obavlja autonomno.

U zadnje vrijeme se najviše radi na pristupima koji koriste tzv. proširenu stvarnost (engl. *augmented reality*, AR), pristup u kojem se informacije iz više izvora prikazuju u istom okviru. U [69] predstavljeno jednostavno sučelje koje na temelju fuziji senzora poboljšava performanse u teleoperaciji. Sustav se sastoji od odometrijskog senzora, stereo kamere i niza ultrazvučnih senzora čijim kombiniranjem se dobiva odgovarajuća slika koja prenosi više podataka o okolišu nego kada bi se ti podaci prenosili svaki zasebno, jedan pokraj drugoga te olakšava procjenu relativne udaljenosti robota od prepreka. U [70] predstavljena 'ekološka' sučelja koja se temelje na 'ekološkoj' teoriji vizualne percepcije koja kaže da za ispravnu percepciju okoline, operator ne mora razlučiti stvarne od virtulanih okolina, jer je ispravna percepcija samo ona koja rezultira uspješnim akcijama [71]. Stoga je implementirano 3D sučelje korištenjem proširene virtualnosti[1], prikazano na slici 6.1. Korištenjem opisanih sučelja su provedeni eksperimenti koji se tiču upravljanja robotom s udaljene lokacije, navigacije i pokrivanja prostora (mapiranje) i zadatak potrage za vrijeme navigacije. Rezultati pokazuju bolje performanse te manji broj udara u prepreke u odnosu na isto sučelje kada se robotom upravlja korištenjem 3D sučelja u odnosu na standardno 2D sučelje.

[1] Autori proširenu virtualnost (engl. *augmented virtuality*) definiraju kao virtualni okoliš koji je dograđen sa slikama iz stvarnog svijeta

Tablica 6.1: Prikaz performansi kod teleoperacije uz prisutnost latencije u eksperimentu provedenom u [70]

	Delay Condition		
	0-seconds	0.5-seconds	1-second
2D Interface	302s	422s	578s
3D Interface	221s	311s	466s
% Change	-27%	-26%	-19%
p-values	5.0×10^{-5}	2.4×10^{-4}	2.3×10^{-3}

(a) Vrijeme potrebno za izvršenje zadatka

	Delay Condition		
	0-seconds	0.5-seconds	1-second
2D Interface	10.6	22.7	38.6
3D Interface	1.7	8.1	28.4
% Change	-84%	-65%	-27%
p-values	9.9×10^{-4}	6.8×10^{-3}	1.2×10^{-1}

(b) Broj sudara

6.2. Latencija

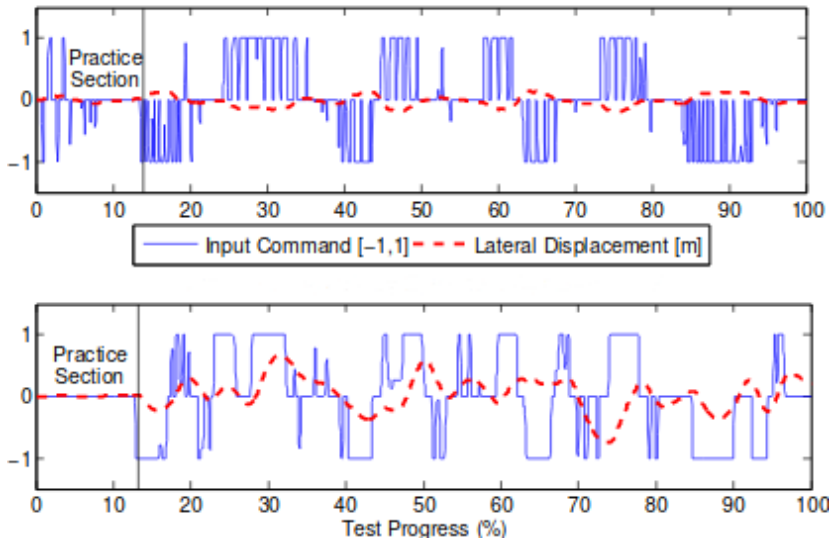
Budući da se upravljanje robotom s udaljene lokacije odvija putem nekog od komunikacijskih kanala, najčešće preko interneta, kašnjenje komandi operatora, kao i kašnjenje povratne veze, je u realnim uvjetima neizbježno. U ovisnosti o tome je li latencija konstantna i koliko je veliko, te je li vremenski promjenjiva, može doći do degradacije performansi u teleoperaciji.

Problem latencije se rješava na različite načine. U [72] su analizirane performanse u više različitih scenarija upravaljnja robotom u teleoperaciji (bez i sa senzorima, jednostavni i složeniji okoliši, ...). Operateri su imali zadatke za obaviti, a slika s kamere i eventualno senzorski podaci su im prikazivani na računalu na udaljenoj lokaciji. Rezultati su pokazali da operatori efikasnije upravljaju robotom u jednostavnim okolinama i bez latencije ako im se ne prikazuju dodatni senzorski podaci. Uvođenjem latencije (samo kašnjenje slike s kamere), kao i u kompleksnijim okolinama, operatori su se bolje snalazili uz prikazane dodatne senzorske podatke, i to su senzorski podaci bili potrebniji što je latencija bila veća.

U [70] je, među ostalim, proveden i ekspreriment s upravljanjem robotom korištenjem implementiranih teleoperacijskih sučelja sa i bez pristunosti latencije. Zadatak je bio provesti robot kroz labirint sa što manje sudara sa zidovima, a mjerenja su pokazala da s rastom latencije performanse drastično padaju (vrijeme potrebno za izvršenje zadatka je skoro udvostručeno uz latenciju od 1 sekunde, dok je rast prosječnog broj sudara eksponencijalan, što je vidljivo iz tablice 6.1).

Prethodni radovi demonstriraju velik utjecaj latencije na teleoperaciju, dok u nastavku slijedi pregled kako smanjiti utjecaj latencije na teleoperaciju. Tako u [73] implementirana teleoperacija između (simuliranog) mobilnog robota i haptičkog upravljača korištenjem komunikacijskog kanala s konstantnom latencijom. Upravljanje robotom je implementirano na sličan način kao što se upravlja automobilom, a zbog pasivnosti sustava osigurana je sigurna i stabilna interakcija s operatorom preko komunikacijskog kanala i uz prisutnost latencije.

Nešto drugačiji pristup je primijenjen u [74]. Tu su autori na temelju studije na korisnicima izradili model čovjeka-operatera koji ga imitira. Model je izrađen pod različitim latencijama i može se koristiti za više primjena, jedna od kojih je u situacijama velike latencije kao



Slika 6.2: Prikaz upravljačkih signala i pogrešku praćenja trajektorije za slučaj bez latencije (gornja slika) i za slučaj uz latenciju od 750 ms (donja slika). Izvor: [74]

zamjena za operatera. Model je izrađen tako da su ispitanici imali za zadatak pratiti unaprijed zadanu trajektoriju. Rezultati su pokazali da su odstupanja veća u slučaju više latencije (slika 6.2) i to se koristilo pri izradi modela, koji je implementiran kao PD regulator.

7. Zaključak

U ovom radu se daje pregled područja autonomnih mobilnih robota. To je područje koje je se u zadnje vrijeme razvija vrlo brzo su svakim danom postaju dostupni novi i inovativni pristupi rješavanju različitih problema u području.

Autonomni mobilni roboti u klasičnom smislu se svode na rješavanje problema navigacije (što uključuje i lokalizaciju), te izbjegavanja prepreka. Da bi to bilo moguće, robot mora biti opremljen odgovarajućim senzorima udaljenosti. U radu je dan pregled klasičnih algoritama za lokalizaciju u vidu EKF lokalizacije i Monte Carlo lokalizacije, koje su, iako relativno stare, najčešće korištene u brojnim primjenama, te naprednijih metoda lokalizacije koje su izvedene iz prethodno navedenih. Predstavljen je i SLAM algoritam, koji rješava problem istovremene navigacije i mapiranja prostora u kojem se robot kreće.

Navigacija robota do zadanog cilja u poznatom prostoru podrazumjeva se planiranje putanje kroz taj poznati prostor, a svodi se na prikazivanje prostora kao grafa, te traženjem najkraćeg puta do zadane točke korištenjem poznatih metoda (Dijkstra, A*) koje nisu razvijene specijalno za korištenje u robotici, već su generički i koriste se u raznim primjenama. Predstavljen je i algoritam Probabilistic roadmap za navigaciju koji u obzir uzima i probabilističku prirodu robota i okoliša.

Izbjegavanje prepreka kod autonomnih mobilnih robota podrazumjeva reaktivno izbjegavanje. Prepreke koje su vidljive na mapi su uzete u obzir kod planiranja putanje (problem navigacije), tako da se u kontekstu izbjegavanja prepreka govori o preprekama kojih na mapi nema, a mogu biti statičke i dinamičke. Metode izbjegavanja prepreka je također moguće primjeniti u slučaju kada je robot u nepoznatom prostoru (tj. kada nema mape).

U novije vrijeme sve više na popularnosti dobijaju pristupi navigaciji i izbjegavanju prepreka koji se temelje na metodama umjetne inteligencije. Umjetna inteligencija se uvelike koristi za navigaciju robota, a najčešća od metoda umjetne inteligencije korištenih za tu namjenu su neuronske mreže. Većina metoda za navigaciju koristi vizualne podatke s kamere kao ulaz, te donosi nekakvu odluku na temelju prepoznavanja i razumijevanja sadržaja slike.

U kontekstu umjetne inteligencije vrlo bitnu ulogu igra i biološki inspirirana navigacija, koja pokušava metodama umjetne inteligencije koja u sličnim situacijama nastoji oponašati ponašanje živih bića. Većina pristupa u ovom području se temelji na oponašanju funkcioniranja hipokampusa glodavaca, te je u radu je dan njihov pregled. RatSLAM je jedan od pristupa biološki inspiriranoj navigaciji koja rješava SLAM problem.

Konačno, dan je pregled metoda za upravljanje mobilnim robotom s udaljene lokacije, koje može povremeno zatrebati, najčešće u slučaju kada algoritmi za autonomno upravljanje robotom zakažu. Za upravljenje robotom s udaljene lokacije je potrebno odgovarajuće upravljačko sučelje koje će operatoru prikazati sve relevantne informacije o stanju robota i okoline i omogućiti mu sigurno upravljanje robotom. Rad donosi pregled različitih pristupa dizajnu upravljačkih sučelja, te daje pregled utjecaja latencije na upravljanje s udaljene lokacije.

Literatura

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[2] S. G. Tzafestas, *Introduction to mobile robot control*. Elsevier, 2013.

[3] J. Borenstein and L. Feng, "Correction of systematic odometry errors in mobile robots," in *International Conference on Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ*, vol. 3, pp. 569–574, IEEE, 1995.

[4] P. Mächler, *Robot Positioning by Supervised and Unsupervised Odometry Correction*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 1998.

[5] G. Reina, G. Ishigami, K. Nagatani, and K. Yoshida, "Odometry correction using visual slip angle estimation for planetary exploration rovers," *Advanced Robotics*, vol. 24, no. 3, pp. 359–385, 2010.

[6] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer Science & Business Media, 2008.

[7] A. Astolfi, "Exponential stabilization of a wheeled mobile robot via discontinuous control," *Journal of dynamic systems, measurement, and control*, vol. 121, no. 1, pp. 121–126, 1999.

[8] M. Milford and R. Schulz, "Principles of goal-directed spatial robot navigation in biomimetic models," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 369, no. 1655, 2014.

[9] F. Amigoni, W. Yu, T. Andre, D. Holz, M. Magnusson, M. Matteucci, H. Moon, M. Yokotsuka, G. Biggs, and R. Madhavan, "A standard for map data representation: Ieee 1873-2015 facilitates interoperability between robots," *IEEE Robotics Automation Magazine*, vol. 25, pp. 65–76, March 2018.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[11] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[12] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 376–382, Jun 1991.

[13] L. Jetto, S. Longhi, and G. Venturini, "Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 219–229, Apr 1999.

[14] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, pp. 335–348, Springer, July 2014.

[15] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[16] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[17] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068, pp. 182–194, International Society for Optics and Photonics, 1997.

[18] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, pp. 1322–1328 vol.2, 1999.

[19] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, pp. 713–720, 2002.

[20] C. Kwok, D. Fox, and M. Meila, "Adaptive real-time particle filters for robot localization," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, pp. 2836–2841 vol.2, Sept 2003.

[21] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pp. 1442–1447 vol.3, Nov 1991.

[22] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 229–241, Jun 2001.

[23] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 242–257, Jun 2001.

[24] J. J. Leonard and H. J. S. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," in *Robotics Research*, pp. 169–176, Springer, 2000.

[25] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[26] M. Michael, T. Sebastian, K. Daphne, and W. Ben, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2, 2003.

[27] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, Dec 1959.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.

[29] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, Aug 1996.

[30] S. Sekhavat, P. Švestka, J.-P. Laumond, and M. H. Overmars, "Multilevel path planning for nonholonomic robots using semiholonomic subsystems," *The International Journal of Robotics Research*, vol. 17, no. 8, pp. 840–857, 1998.

[31] P. Švestka and M. H. Overmars, "Motion planning for carlike robots using a probabilistic learning approach," *The International Journal of Robotics Research*, vol. 16, no. 2, pp. 119–143, 1997.

[32] P. Švestka and M. H. Overmars, "Coordinated motion planning for multiple car-like robots using probabilistic roadmaps," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1631–1636 vol.2, May 1995.

[33] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[34] J. Borenstein and Y. Koren, "High-speed obstacle avoidance for mobile robots," in *Proceedings IEEE International Symposium on Intelligent Control 1988*, pp. 382–384, Aug 1988.

[35] C. W. Warren, "Global path planning using artificial potential fields," in *Proceedings, 1989 International Conference on Robotics and Automation*, pp. 316–321 vol.1, May 1989.

[36] L. C. A. Pimenta, A. R. Fonseca, G. A. S. Pereira, R. C. Mesquita, E. J. Silva, W. M. Caminhas, and M. F. M. Campos, "Robot navigation based on electrostatic field computation," *IEEE Transactions on Magnetics*, vol. 42, pp. 1459–1462, April 2006.

[37] M. G. Park, J. H. Jeon, and M. C. Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," in *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570)*, vol. 3, pp. 1530–1535 vol.3, 2001.

[38] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1, pp. 256–263 vol.1, 2000.

[39] J. Minguez, "The obstacle-restriction method for robot obstacle avoidance in difficult environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2284–2290, Aug 2005.

[40] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, pp. 23–33, Mar 1997.

[41] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 278–288, Jun 1991.

[42] I. Ulrich and J. Borenstein, "Vfh*: local obstacle avoidance with look-ahead verification," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 3, pp. 2505–2511 vol.3, 2000.

[43] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[44] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[46] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 253–256, May 2010.

[47] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 855–868, May 2009.

[48] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[49] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning series)*. A Bradford Book, 1998.

[50] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[51] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[52] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[53] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, pp. 396–407, Jun 1996.

[54] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. LeCun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, pp. 739–746, 2006.

[55] H. Raia, S. Pierre, B. Jan, E. Ayse, S. Marco, K. Koray, M. Urs, and L. Yann, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[56] P. J. Zeno, S. Patel, and T. M. Sobh, "Review of neurobiologically based mobile robot navigation system research performed since 2000," *Journal of Robotics*, vol. 2016, pp. 1–17, 2016.

[57] M. J. Milford, G. F. Wyeth, and D. Prasser, "Ratslam: a hippocampal model for simultaneous localization and mapping," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, pp. 403–408 Vol.1, April 2004.

[58] A. Arleo, *Spatial Learning and Navigation in Neuro Mimetic Systems: Modeling the Rat Hippocampus*. Dissertation. de, 2000.

[59] A. D. Redish, A. N. Elga, and D. S. Touretzky, "A coupled attractor model of the rodent head direction system," *Network: Computation in Neural Systems*, vol. 7, no. 4, pp. 671–685, 1996.

[60] S. M. Stringer, E. T. Rolls, T. P. Trappenberg, and I. E. T. de Araujo, "Self-organizing continuous attractor networks and path integration: two-dimensional models of place cells," *Network: Computation in Neural Systems*, vol. 13, no. 4, pp. 429–446, 2002. PMID: 12463338.

[61] M. Milford, G. Wyeth, and D. Prasser, "Ratslam on the edge: Revealing a coherent representation from an overloaded rat brain," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4060–4065, Oct 2006.

[62] N. Sünderhauf and P. Protzel, "Beyond ratslam: Improvements to a biologically inspired slam system," in *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pp. 1–8, Sept 2010.

[63] L. Tai, S. Li, and M. Liu, "Autonomous exploration of mobile robots through deep neural networks," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, p. 1729881417703571, 2017.

[64] J. M. Riley, D. B. Kaber, and J. V. Draper, "Situation awareness and attention allocation measures for quantifying telepresence experiences in teleoperation," *Human Factors and Ergonomics in Manufacturing & Service Industries*, vol. 14, no. 1, pp. 51–67, 2004.

[65] M. R. Endsley, "Design and evaluation for situation awareness enhancement," *Proceedings of the Human Factors Society Annual Meeting*, vol. 32, no. 2, pp. 97–101, 1988.

[66] A. Poncela and L. Gallardo-Estrella, "Command-based voice teleoperation of a mobile robot via a human-robot interface," *Robotica*, vol. 33, no. 1, p. 1–18, 2015.

[67] S. Muszynski, J. Stückler, and S. Behnke, "Adjustable autonomy for mobile teleoperation of personal service robots," in *RO-MAN, 2012 IEEE*, pp. 933–940, IEEE, 2012.

[68] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Autonomous Robots*, vol. 11, pp. 9–18, Jul 2001.

[69] R. Meier, T. Fong, C. Thorpe, and C. Baur, "Sensor fusion based user interface for vehicle teleoperation," in *Field and Service Robotics*, no. LSRO2-CONF-1999-002, 1999.

[70] C. W. Nielsen, M. A. Goodrich, and R. W. Ricks, "Ecological interfaces for improving mobile robot teleoperation," *IEEE Transactions on Robotics*, vol. 23, pp. 927–941, Oct 2007.

[71] J. J. Gibson, *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[72] D. Sanders, "Analysis of the effects of time delays on the teleoperation of a mobile robot in various modes of operation," *Industrial Robot: the international journal of robotics research and application*, vol. 36, no. 6, pp. 570–584, 2009.

[73] D. Lee, O. Martinez-Palafox, and M. W. Spong, "Bilateral teleoperation of a wheeled mobile robot over delayed communication network," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 3298–3303, May 2006.

[74] S. Vozar and D. M. Tilbury, "Driver modeling for teleoperation with time delay," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3551 – 3556, 2014. 19th IFAC World Congress.

Konvencije označavanja

Brojevi, vektori, matrice i skupovi

a	Skalar
$\boldsymbol{a}$	Vektor
$\hat{\boldsymbol{a}}$	Jedinični vektor
$\boldsymbol{A}$	Matrica
A	Skup
a	Slučajna skalarna varijabla
$\mathbf{a}$	Slučajni vektor
$\mathbf{A}$	Slučajna matrica

Indeksiranje

a_i	Element na mjestu i u vektoru $\boldsymbol{a}$
A_{ij}	Element na mjestu (i, j) u matrici $\boldsymbol{A}$
$\boldsymbol{a}_t$	Vektor $\boldsymbol{a}$ u trenutku t
a_i	Element na mjestu i u slučajnog vektoru $\mathbf{a}$

Vjerojatnosti i teorija informacija

$P(\mathrm{a})$	Distribucija vjerojatnosti za diskretnu varijablu
$p(\mathrm{a})$	Distribucija vjerojatnosti za kontinuiranu varijablu
$\mathrm{a} \sim P$	a ima distribuciju P
$\boldsymbol{\Sigma}$	Matrica kovarijance
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normalna distribucija od $\boldsymbol{x}$ sa srednjom vrijednošću $\boldsymbol{\mu}$ i kovarijancom $\boldsymbol{\Sigma}$