Izvedba sistemov vodenja na osnovi modelov za avtonomno vožnjo miniaturnih robotskih vozil

Marko Tavčar, Urban Skalar, Jernej Perko, Andrej Zdešar

Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana

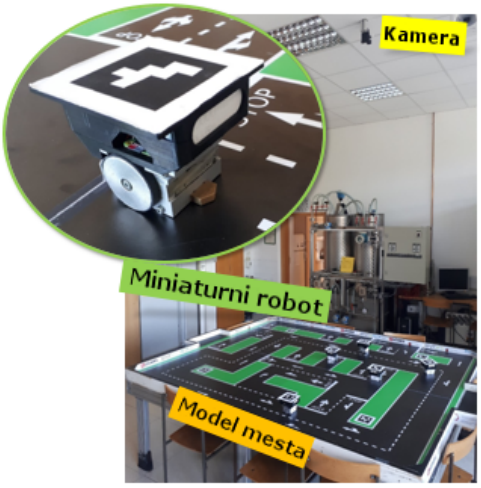
E-pošta: mt6480@student.uni-lj.si, us1673@student.uni-lj.si, jp2352@student.uni-lj.si, andrej.zdesar@fe.uni-lj.si

Implementation of model-based control systems for autonomous driving of miniature robotic vehicles

In this work we focus on implementation of algorithms for localisation, path planning and path tracking on a cyber-physical system of miniature mobile robots. We implemented Kalman filter to combine the data from two different sensor sources. An A algorithm was used to find the optimal path in the map. Robot was controlled in such a manner that it is capable of following the reference path, which leads to the desired goal.*

I. Uvod

V svetu zanimanje za avtonomne mobilne sisteme narašča. V tem delu predstavljamo, kako smo implementirali algoritme, ki so ključni za avtonomno delovanje robotskih vozil. V ta namen smo uporabili poenostavljen fizični model mesta z miniaturnimi mobilnimi roboti, ki je prikazan na sliki 1. Na fizičnem modelu smo implementirali algoritme za lokalizacijo, načrtovanje poti in vodenje po poti [1] tako, da se mobilni robot lahko avtonomno vozi po cestah med poljubnimi cilji in pri tem vozi varno ter upošteva cestno-prometne predpise.



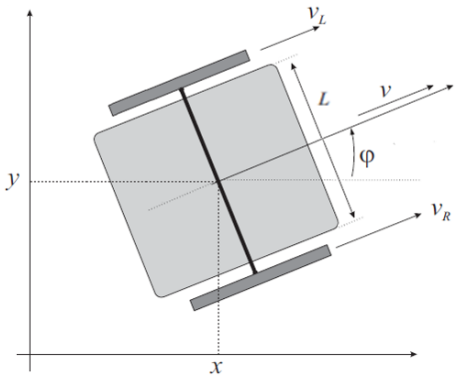
Slika 1: Fizični model mesta s cestami in miniaturni mobilni roboti z značkami ter kamera

Delo, predstavljeno v tem članku, je bilo opravljeno v okviru projekta InMotion, ki ga sofinancira program Evropske unije Erasmus+, št. 573751-EPP-1-2016-1-DE-EPPKA2-CBHE-JP.

II. Predstavitev sistema

A. Kolesni mobilni robot

V našem sistemu smo uporabili 5 kolesnih mobilnih robotov z diferencialnim pogonom. Na vsakem mobilnem robotu se nahaja mini računalnik *Raspberry Pi Zero W*, ki omogoča izvedbo algoritmov za avtonomno vožnjo, in mikrokrmilnk *Arduino Mini*, ki se uporablja za krmiljenje motorjev in branje podatkov iz enkoderjev. Maksimalna hitrost gibanja robota znaša $0{,}3\,\mathrm{m/s}$. Na vsakem motorju kolesa ima robot nameščen en enkoder za merjenje zasuka kolesa s frekvenco vzorčenja $100\,\mathrm{Hz}$.



Slika 2: Prikaz mobilnega robota od zgoraj z označenimi osnovnimi vektorji gibanja

Model kolesnega mobilnega sistema z diferencialnim pogonom je prikazan na sliki 2. Vpliv vhodne tangencialne hitrosti $v(t)$ in kotne hitrosti $\omega(t)$ na spremembo lege mobilnega sistema podaja kinematični model (1).

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

B. Kamera

Nad poligonom se nahaja kamera tako, da je celotni poligon v njenem vidnem polju. Kamera preko WiFi-povezave sporoča vsem sistemom v lokalnem omrežju izmerjene lege robotov na poligonu glede na globalni koordinatni sistem, ki se nahaja v enem izmed robov poligona. Sistem za vizualno sledenje značk omogoča obdelavo slike s frekvenco $10\,\mathrm{Hz}$.

III. Lokalizacija mobilnega robota

Za merjenje lege $\boldsymbol{x}(k) = [x(k)\ y(k)\ \varphi(k)]^T$ mobilnega robota imamo na voljo dva nepovezana senzorska vira: enkoderje na kolesih in kamero. Na podlagi enkoderjev na kolesih lahko določimo lego mobilnega sistema s postopkom odometrije. Merilni sistem s kamero podaja lego mobilnega robota glede na globalni koordinatni sistem, in v tem smislu deluje kot globalni satelitski sistem za določanje položaja.

A. Odometrija

Kinematični model (1) lahko z Eulerjevo integracijsko metodo pretvorimo v diskretno obliko (2). Za kratke čase vzorčenja T je napaka, ki jo dobimo zaradi integracijske metode zanemarljiva glede na ostale vire napak (npr. zdrsovanje koles).

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \begin{bmatrix} \cos\varphi(k) & 0 \\ \sin\varphi(k) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(k)T \\ \omega(k)T \end{bmatrix} \tag{2}$$

Vpeljemo relativni tangencialni premik $\Delta d(k) = v(k)T$ in relativni kotni zasuk $\Delta\varphi(k) = \omega(k)T$ mobilnega robota. Relaciji med relativnimi premiki mobilnega sistema in relativnimi premiki levega in desnega kolesa, $\Delta d_L(k)$ in $\Delta d_D(k)$, sta:

$$\begin{aligned} \Delta d(k) &= \frac{\Delta d_D(k) + \Delta d_L(k)}{2}\,, \\ \Delta\varphi(k) &= \frac{\Delta d_D(k) - \Delta d_L(k)}{L}\,, \end{aligned} \tag{3}$$

kjer je L razdalja med kolesoma. Relativne premike koles lahko merimo z relativnimi spremembami vrednosti enkoderjev na kolesih ($\Delta\alpha_D(k)$ in $\Delta\alpha_L(k)$):

$$\boldsymbol{u}(k) = \begin{bmatrix} \Delta d_L(k) \\ \Delta d_D(k) \end{bmatrix} = \begin{bmatrix} K_L\Delta\alpha_L(k) \\ K_D\Delta\alpha_D(k) \end{bmatrix} \tag{4}$$

Konstante $K_L = K_D = 9{,}537\,\mu\mathrm{m}$ in $L = 0{,}075\,\mathrm{m}$ smo določili s postopkom kalibracije. Po vstavitvi (3) in (4) v (2), dobimo model, ki omogoča spremljanje lege mobilnega sistema na podlagi merjenja relativnih premikov koles z enkoderji.

B. Nedeterministični model kolesnega robota

Za združevanja informacije o legi na podlagi odometrije in kamere smo uporabili razširjeni Kalmanov filter, ki je pogosta izbira pri ocenjevanju lege v mobilni robotiki [2]. Za ta namen smo definirali nedeterministični model mobilnega robota, ki je predstavljen v (5) in (6).

Različne vplive motenj, ki nastopajo pri merjenju položaja na podlagi odometrije, smo v modelu prehajanja stanj (5) modelirali z normalno porazdeljenim šumom $\boldsymbol{w}$ s srednjo vrednostjo nič in varianco $\boldsymbol{Q}$.

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{w}(k)) = \\ &= \boldsymbol{x}(k) + \begin{bmatrix} \frac{\cos\varphi(k)}{2} & \frac{\cos\varphi(k)}{2} \\ \frac{\sin\varphi(k)}{2} & \frac{\sin\varphi(k)}{2} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} (\boldsymbol{u}(k) + \boldsymbol{w}(k)) \end{aligned} \tag{5}$$

$$\boldsymbol{w}(k) = \begin{bmatrix} w_L(k) \\ w_D(k) \end{bmatrix}\,, \quad \boldsymbol{Q} = \begin{bmatrix} \sigma_L^2 & \sigma_L\sigma_D \\ \sigma_L\sigma_D & \sigma_D^2 \end{bmatrix}$$

Predpostavljamo, da meritve s kamere $\boldsymbol{z}(k)$ vsebujejo normalno porazdeljen šum $\boldsymbol{v}(k)$ s srednjo vrednostjo nič in varianco $\boldsymbol{R}$, kot to opisuje (6).

$$\boldsymbol{z}(k) = \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{v}(k)) = \boldsymbol{x}(k) + \boldsymbol{v}(k) \tag{6}$$

$$\boldsymbol{v}(k) = \begin{bmatrix} v_x(k) \\ v_y(k) \\ v_\varphi(k) \end{bmatrix}\,, \quad \boldsymbol{R} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\varphi^2 \end{bmatrix}$$

C. Razširjeni Kalmanov filter

Razširjeni Kalmanov filter smo izpeljali na podlagi nedeterminističnega modela mobilnega robota, ki je predstavljen v poglavju III-B. Algoritem vsebuje predikcijski in korekcijski korak, ki ju izvajamo zaporedno s frekvenco vzorčenja podatkov z odometrije.

V predikcijskem koraku s (7) napovemo trenutno lego mobilnega sistema $\boldsymbol{x}(k)$ na podlagi meritev iz enkoderjev $\boldsymbol{u}(k-1)$ in zadnje ocene lege $\boldsymbol{x}(k-1)$. Z (8) določimo še varianco napovedi.

$$\hat{\boldsymbol{x}}(k) = \boldsymbol{f}(\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), \boldsymbol{0}) \tag{7}$$

$$\hat{\boldsymbol{P}}(k) = \boldsymbol{A}_{k-1}\boldsymbol{P}(k-1)\boldsymbol{A}_{k-1}^T + \boldsymbol{F}_{k-1}\boldsymbol{Q}\boldsymbol{F}_{k-1}^T \tag{8}$$

Če imamo na voljo novo meritev s kamere, izračunamo ojačenje $\boldsymbol{K}_k$:

$$\boldsymbol{K}_k = \hat{\boldsymbol{P}}(k)\boldsymbol{C}_k^T(\boldsymbol{C}_k\hat{\boldsymbol{P}}(k)\boldsymbol{C}_k^T + \boldsymbol{R})^{-1}\,, \tag{9}$$

sicer pa postavimo $\boldsymbol{K}_k = \boldsymbol{0}$. V korekcijskem koraku posodobimo oceno lege mobilnega robota $\boldsymbol{x}(k)$ na podlagi napovedi $\hat{\boldsymbol{x}}(k)$ in ob upoštevanju meritve s kamere $\boldsymbol{z}(k)$, kot to opisuje (10). Z (11) posodobimo tudi varianco ocene.

$$\boldsymbol{x}(k) = \hat{\boldsymbol{x}}(k) + \boldsymbol{K}_k(\boldsymbol{z}(k) - \boldsymbol{h}(\hat{\boldsymbol{x}}(k), \boldsymbol{0})) \tag{10}$$

$$\boldsymbol{P}(k) = \hat{\boldsymbol{P}}(k) - \boldsymbol{K}_k\boldsymbol{C}_k\hat{\boldsymbol{P}}(k) \tag{11}$$

Matrike $\boldsymbol{A}_k$, $\boldsymbol{F}_k$ in $\boldsymbol{C}_k$, ki se pojavljajo v enačbah (7) do (11), določimo na podlagi linearizacije nelinearnega modela mobilnega sistema v okolici ocene v trenutku k:

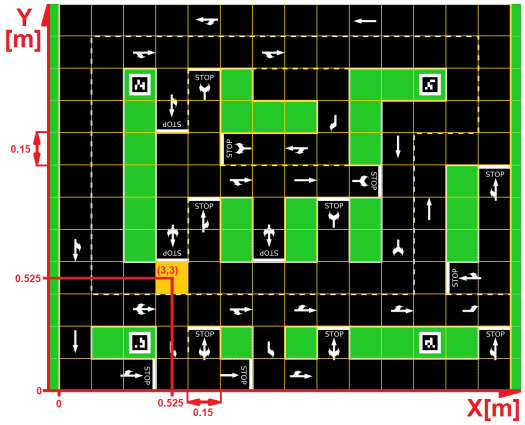
$$\begin{aligned} \boldsymbol{A}_k &= \left.\frac{\partial\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{w})}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}(k),\boldsymbol{u}=\boldsymbol{u}(k),\boldsymbol{w}=\boldsymbol{0}} \\ \boldsymbol{F}_k &= \left.\frac{\partial\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{w})}{\partial\boldsymbol{w}}\right|_{\boldsymbol{x}=\boldsymbol{x}(k),\boldsymbol{u}=\boldsymbol{u}(k),\boldsymbol{w}=\boldsymbol{0}} \\ \boldsymbol{C}_k &= \left.\frac{\partial\boldsymbol{h}(\boldsymbol{x},\boldsymbol{v})}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}(k),\boldsymbol{v}=\boldsymbol{0}} \end{aligned} \tag{12}$$

IV. Načrtovanje poti

Avtonomni mobilni roboti morajo biti sposobni načrtovanja optimalne poti od točke A do točke B. Pri implementaciji načrtovanja poti so bila upoštevana določena cestno-prometna pravila. Mobilni robot se lahko vozi le po desnem pasu ceste, vožnja po zelenici ni dovoljena. Pri vožnji mora upoštevati tudi stop znake, na križiščih pa so mu dovoljeni U-zavoji.

A. Modeliranje okolja

Da lahko mobilni robot najde optimalno pot, se mora zavedati okolja v katerem se nahaja. Okolje smo z mrežo razdelili na 14×12 celic (slika 3) in vsaki celici smo



Slika 3: Razdelitev poligona na celice z označenimi koordinatnimi sistemi in dimenzijami

priredili unikatno oznako v obliki dveh števil $(i,\ j)$. Preslikavo med položajem v globalnem koordinatnem sistemu in oznako pripadajoče celice podaja (13), kjer operator $\lfloor\cdot\rfloor$ predstavlja celoštevilsko zaokroževanje navzdol.

$$i = \left\lfloor \frac{x}{0{,}15} \right\rfloor, \quad j = \left\lfloor \frac{y}{0{,}15} \right\rfloor \tag{13}$$

Obratna transformacija (14) priredi vsaki celici položaj v globalnem koordinatnem sistemu, ki se nahaja v središču te celice.

$$x = 0{,}15i + 0{,}075\,, \quad y = 0{,}15j + 0{,}075 \tag{14}$$

Slika 3 ilustrira primer transformacije za eno celico.

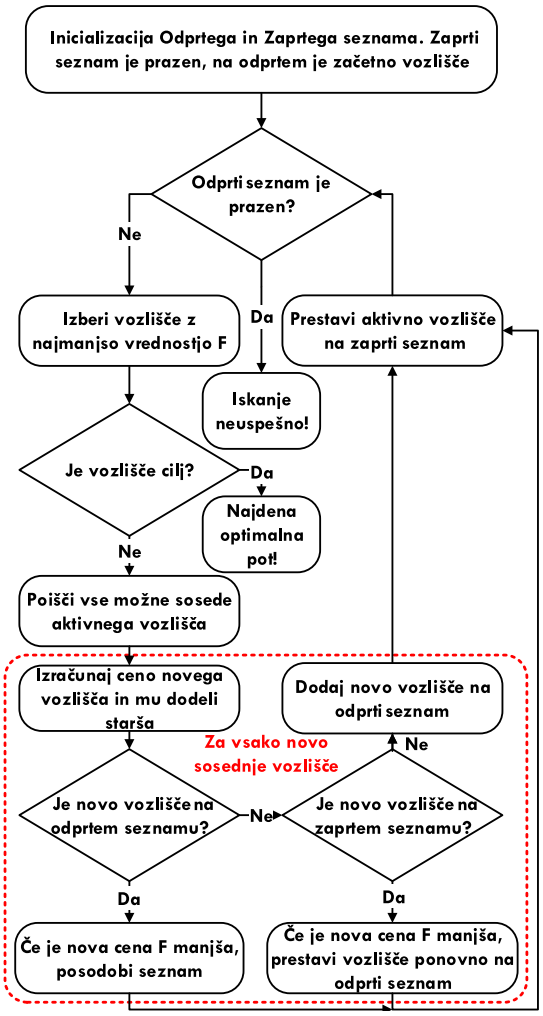
B. Algoritem iskanja poti

Cilj algoritma za iskanje poti je najti optimalno pot od starta do cilja. Pri tem vhod v algoritem predstavljata točki start (začetna lega robota) in cilj (želena lega robota). Izhod algoritma predstavlja urejen niz celic, ki sestavljajo pot od start do cilja. Optimalna pot ni nujno najkrajša pot. V naši implementaciji, algoritem posebej upošteva stop celice. Vsaka stop celica vzame mobilnemu robotu več časa kot pa ostale celice, saj se mora mobilni robot pred stop znakom ustaviti. Posledično imajo takšne celice manjši potencial, da se znajdejo na optimalni poti.

*1) Algoritem A**

V našem sistemu smo implementirali algoritem A* [3], ki je zaradi svoje optimalnosti in hitrosti izračuna, pogosta izbira za iskanje optimalne poti. Z našo implementacijo algoritma smo dosegli zadovoljivo hitro izvajanje algoritma, zato nismo iskali alternativnih algoritmov. A* je informiran algoritem iskanja poti, ki pri svojem delovanju uporablja hevristično oceno *cene do cilja*. S tem se iskanje po vozliščih omeji na celice, ki bolj verjetno sestavljajo optimalno pot, kar posledično poveča hitrost izračuna.

V algoritmu nastopa cena F, ki določa vrstni red odkrivanja celic in je definirana kot vsota dveh cen: $F = G + H$. G predstavlja ceno poti, ki vodi od začetne do trenutno izbrane celice. Določimo jo na podlagi vsote uteženih prehodov med celicami. V našem primeru smo uporabili različne uteži za prehode čez stop znake (utež 2) kot za običajne prehode med celicami (utež 1). Tako postanejo celice s stop znakom slabši kandidati za nadaljevanje poti, saj se mora tam mobilni robot ustaviti. H podaja hevristično oceno razdalje od trenutne celice do cilja. V našem primeru smo uporabili Manhattansko razdaljo.



Slika 4: Diagram izvajanja algoritma A*

Koraki algoritma A* so razvidni na sliki 4. V inicializaciji algoritma definiramo prazen odprti in zaprti seznam ter uvrstimo začetno celico na odprti seznam. Nato vstopimo v neskončno zanko, kjer v vsakem ciklu izvedemo naslednje korake. Iz odprtega seznama odstranimo celico, ki ima najmanjšo vrednost F. Nato poiščemo vse njene sosednje celice in jih uvrstimo na odprti seznam Za vsako novo odkrito celico izračunamo vrednosti G, H in F ter ji dodelimo starševsko (trenutno) celico. En cikel zanke končamo tako, da uvrstimo trenutno celico na zaprti seznam.

Ko po tem postopku pridemo do ciljne celice, lahko algoritem prekinemo, saj to pomeni, da obstaja pot do cilja. Optimalno pot poiščemo tako, da sledimo staršev-skim celicam, dokler ne prispemo do začetne celice. Izvajanje algoritma zaključimo tudi, če je odprti seznam prazen. To pomeni, da pot do želenega cilja ne obstaja.

V. Vodenje mobilnega robota po poti

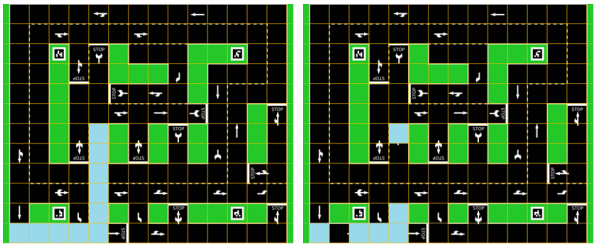
Lego mobilnega robota glede na podane vhodne hitrosti opisuje direktna kinematika (1). Pri načrtovanju vodenja po poti se ukvarjamo s problemom izračunavanja tangencialne in kotne hitrosti, ki sta potrebni, da se mobilni robot vozi po želeni poti. Sistem (1) ima neholonomično omejitev, zato je izračun invezne kinematike v splošnem zahtevna naloga. Preprosta rešitev inverzne kinematike obstaja za primere, ko se mobilni robot vozi le naravnost in obrača na mestu, a takšno gibanje robota običajno ni optimalno. V nadaljevanju je predstavljen princip vodenja mobilnega robota po referenčni poti, ki smo jo dobili s postopkom načrtovanja poti, predstavljenim v poglavju IV-A.

A. Poenostavljen zapis poti

Mobilni robot dobi navodila za pot v obliki urejenega seznama celic, ki jih mora obiskati, da doseže cilj. Ali so vse celice potrebne za popoln opis poti, ki jo želimo opraviti? Sosednje celice v urejenem seznamu celic nimajo dodane vrednosti, če le-te predstavljajo ravno pot. Takšne celice lahko s seznama odstranimo. Obdržati moramo torej le začetno in končno celico ter celice, kjer pride do spremembe smeri. Poleg tega moramo na seznamu obdržati še celice s stop znakom, saj se moramo na teh mestih ustaviti.

Za zaznavanje celic, kjer pride do spremembe smeri poti, smo uporabili naslednji algoritem. Sprehodimo se skozi urejen seznam celic, ki predstavljajo pot. Za vsako celico izvedemo preprost test, tako da primerjamo oznako trenutne celice (oznaka celice je definirana v poglavju IV-A) z oznako celice s seznama, ki je oddaljena za dve mesti. Če se obe vrednosti oznake razlikujeta, pomeni, da je na tem mestu zavoj, zato vse celice, ki tvorijo zavoj, obdržimo na seznamu.

Primer poenostavitve poti je prikazan na sliki 5.



(a) (b)

Slika 5: (a) Pot iz celice $(0,\ 0)$ v celico $(4,\ 5)$ in (b) poenostavljena pot

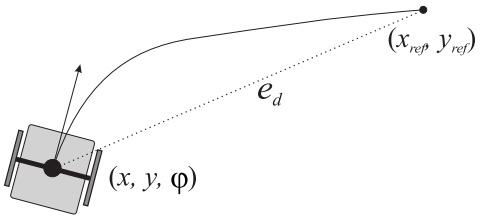
B. Algoritem za vodenje po poti

Oglejmo si algoritem za vodenje do točke. Definiramo pogrešek po razdalji (evklidsko mera) in kotu (slika 6):

$$e_d(k) = \sqrt{(x_{ref} - x(k))^2 + (y_{ref} - y(k))^2}\,, \qquad (15)$$

$$e_\varphi(k) = \arctan \frac{y_{ref} - y(k)}{x_{ref} - x(k)} - \varphi(k) + (n + 2m)\pi\,.$$

Pri izračunu pogreška po kotu moramo s pravilno izbiro parametra $n \in \{0, 1\}$ zagotoviti, da dobimo kot rezultat



Slika 6: Prikaz vodenja do referenčne točke

funkcije $\arctan$ kot, ki je v pravilnem kvadrantu. S parametrom $m \in \mathbb{Z}$ pa moramo zagotoviti, da je zaloga vrednosti vedno znotraj območja $e_\varphi(k) \in [-\pi, \pi)$.

Za izračun translatorne hitrosti $v(k)$ uporabimo enostaven P-regulator z ojačenjem K_v:

$$v(k) = \begin{cases} K_v e_d(k) \cos^2 e_\varphi(k)\,; & |e_\varphi(k)| < \frac{\pi}{2} \wedge e_d(k) > \xi\,, \\ 0\,; & \text{sicer}\,. \end{cases} \qquad (16)$$

Razdalja ξ je radij, ki predstavlja krog okoli referenčne točke, ki ga smatramo za cilj. Za izračun kotne hitrosti $\omega(k)$ uporabimo naslednji regulator z ojačenjem K_ω:

$$\omega(k) = \begin{cases} K_\omega e_\varphi(k)\,; & e_d(k) > \xi\,, \\ 0\,; & \text{sicer}\,. \end{cases} \qquad (17)$$

Ojačanje P-regulatorja predstavljenega v (16) je 1. Ojačanje regulatorja v (17) pa se spreminja, glede na oddaljenost od točke, v katero se robot pelje. Tako zagotovimo večjo stabilnost regulatorja. Vrednosti ojačanja sta bili določeni eksperimentalno in znašata $1{,}65$, ko je robot blizu cilja, in 70, ko je daleč od njega.

Vodenje po poti lahko izvedemo s prilagoditvijo algoritma za vodenje mobilnega robota do točke tako, da izvedemo primerno preklapljanje med točkami, ki sestavljajo pot. Ko je vrednost pogreška $e_d(k)$ manjša od ξ, za referenčno lego vzamemo naslednjo točko s seznama, ki predstavlja referenčno pot. Če referenčna točka pripada stop celici, se na njej ustavimo za določen čas in nato nadaljujemo z izvajanjem algoritma.

VI. Zaključek

Razvoj algoritmov za avtonomno vožnjo je zahtevna naloga. Fizični model mesta z avtonomnimi mobilnimi sistemi nam je omogočil enostavno implementacijo in študijo osnovnih algoritmov, ki so potrebni za avtonomno vožnjo. Tekom implementacije smo se srečali s številnimi izzivi, ki nastopajo pri prenosu teorije v prakso. V prihodnosti je pričakovati, da se bo področje avtonomnih vozil vedno bolj razvijalo.

Literatura

[1] G. Klančar et al., *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*, Elsevier, 2017.

[2] G. Klančar, L. Teslić in I. Škrjanc, "Mobile-robot pose estimation and environment mapping using an extended Kalman filter,"*International Journal of Systems Science*, zv. 45, št. 12, str. 2603–2618, 2014.

[3] Z. Zhang in Z. Zhao, "A Multiple Mobile Robots Path planning Algorithm Based on A-star and Dijkstra Algorithm", *International Journal of Smart Home*, zv. 8, št. 3, 75–86, 2014.