Vodenje kvadrokopterja na podlagi slike

Matevž BOŠNAK, Drago MATKO, Sašo BLAŽIČ

Izvleček: Določanje položaja plovila predstavlja enega izmed temeljnih problemov pri vodenju zračnih plovil. V članku bo predstavljen sistem za določanje položaja brezpilotnega zračnega plovila s sposobnostjo VTOL (angl. *vertical take-off and landing*) v zaprtem prostoru z namenom vodenja po položaju. Sistem temelji na računalniški obdelavi slike, zajeti s kamero na plovilu, in je sposoben ne le določanja položaja plovila glede na fiksne oznake na tleh prostora, temveč tudi gradnje zemljevida oznak. V predstavljeni rešitvi se tako določen položaj plovila skupaj s podatki o njegovem stanju prenese v okolje Matlab Simulink, kjer je načrtan sistem vodenja, rezultati sistema vodenja pa se nato iz okolja Simulink preko namenske aplikacije pošljejo brezpilotnemu zračnemu plovilu.

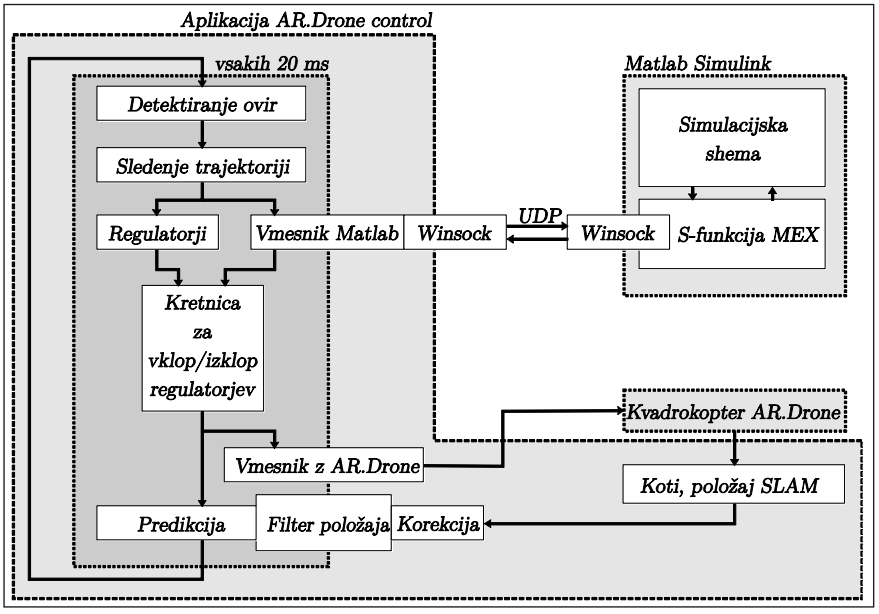
Ključne besede: kvadrokopter, avtonomni sistemi, računalniška analiza slike, določanje položaja in gradnja zemljevida (postopek SLAM)

1 Uvod

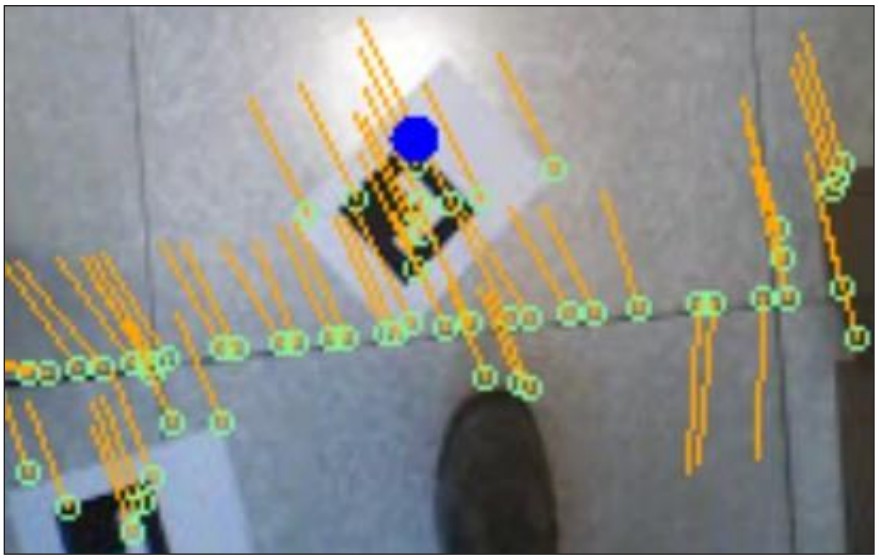
Vodenje brezpilotnih plovil po položaju zahteva natančno poznavanje trenutnega položaja plovila v vsakem časovnem trenutku, kar je v našem primeru predstavljalo osnovno raziskovalno nalogo. Rešitev se zgleduje po principih vodenja na osnovi slike [1], [2], [3], [4] in temelji na združevanju informacije iz rezultatov računalniške obdelave digitalne slike s kamere ter kopice senzorjev, kot so senzorji pospeška, kotne hitrosti in višine. Prikaz delovanja predlaganega sistema je bil izveden z implementacijo na kvadrokopterju AR.Drone [5]. Kvadrokopter AR.Drone je množični proizvod, namenjen za uporabo z mobilnim telefonom. Ogrodje je sestavljeno iz karbonskih in plastičnih struktur, nanj so pritrjeni štirje rotorji s pogonom z brezkrtačnimi motorji, senzorsko in nadzorno vezje, odstranljiva baterija in par barvnih videokamer. Za zaščito rotorjev in ogrodja sta na voljo večja zaščitna lupina za letenje v zaprtih prostorih in manjša lupina za letenje zunaj.

Senzorsko vezje v kvadrokopterju AR.Drone že vsebuje enoto IMU (angl. *inertial measurement unit*) s pospeškometri in žiroskopi ter ultrazvočni merilnik višine. Kvadrokopter je dodatno opremljen s kamero VGA, ki je usmerjena v smer leta in ima ločljivost *640* × *480* slikovnih točk, in kamero, usmerjeno proti tlom, ločljivosti *176* × *144* slikovnih točk. Za izvajanje nizkonivojske stabilizacije kvadrokopterja in komunikacije z nadzornim sistemom je kvadrokopter AR.Drone opremljen s procesorjem ARM9 s taktom *468* MHz, ki poganja vgrajeni sistem Linux z jedrom 2.6.27 [6]. Za komunikacijo kvadrokopter AR.Drone deluje kot brezžična do-

Dr. Matevž Bošnak, univ. dipl. inž., VESOLJE-SI, Ljubljana; prof. dr. Drago Matko, univ. dipl. inž., izr. prof. dr. Sašo Blažič, univ. dipl. inž.; oba Univerza v Ljubljani, Fakulteta za elektrotehniko in VESOLJE-SI, Ljubljana



Slika 1. *Bločni diagram predstavljene rešitve*



Slika 2. *Prikaz razpoznanih vektorjev optičnega toka*

stopna točka WiFi in tako omogoča vodenje kvadrokopterja tako preko mobilnih naprav s sistemoma Android in iOS kot tudi preko osebnega računalnika, saj je proizvajalec Parrot izdal specifikacije uporabljenega protokola [7] in s tem omogočil dostop do meritev senzorjev, slik kamer s kvadrokopterja ter vodenje.

Za vodenje kvadrokopterja smo razvili program v programskem jeziku C# (prikazan kot Aplikacija AR.Drone control na *sliki 1*), ki je namenjen:
- komunikaciji s kvadrokopterjem preko omenjenega protokola,
- izvajanju obdelave zajete slike (navpično navzdol postavljene) kamere, izvajanju prilagojenega algoritma za metodo SLAM (angl. *Simultaneous Localization And Mapping*) [8],
- izvajanju algoritmov vodenja kvadrokopterja po položaju [9],
- zaznavanju trkov z ovirami na podlagi modela kvadrokopterja,
- posredovanju podatkov meritev v simulacijsko shemo in iz nje v okolju Matlab Simulink.

V tem prispevku bomo podali kratek opis sistema za razpoznavanje slike, sistema SLAM in vmesnika z okoljem Matlab Simulink.

2 Delovanje sistema za razpoznavanje slike

Računalniška obdelava slike je proces, s katerim dvodimenzionalno polje vrednosti, ki ustrezajo izmerjenim jakostim svetlobe za posamezno slikovno točko, pretvorimo v novo sliko ali v niz številskih podatkov, s katerimi opišemo vsebino slike, ki nas zanima. V našem primeru iz vsake zajete slike (in kopije predhodno zajete slike) izluščimo podatek o položaju točno določenih umetnih značilnih točk v sliki in podatek o slikovnem toku.

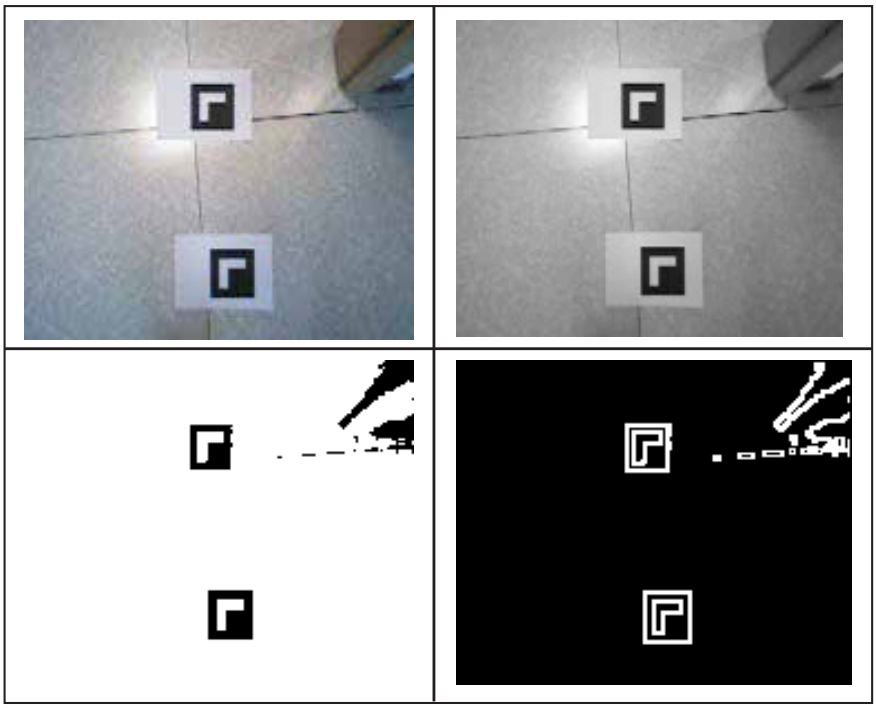
Sistem za obdelavo slike, ki smo ga realizirali z uporabo ene videokamere, ki je fiksno pritrjena na ohišje kvadrokopterja, smo ločili v sistem za:
- merjenje optičnega toka v sliki z značilnimi točkami FAST in metodo Lukas-Kanade,
- razpoznavo značilk v sliki in
- določevanje položaja kamere na podlagi razpoznanih značilnih točk.

Polje gibanja je vektorsko polje, sestavljeno iz vektorjev, ki kažejo premike posameznih točk med zajetjem dveh zaporednih slik in ga potrebujemo za ocenjevanje linearne hitrosti premikanja kamere nad površino. V večini situacij (problemi nastanejo pri spreminjanju osvetlitve predmeta) lahko polje gibanja aproksimiramo z uporabo optičnega toka [10] (angl. *optical flow*), ki predstavlja vektorsko polje (prikaz na *sliki 2*), katerega komponente so vektorji, ki kažejo navidezne premike izbranih točk slike. Za izračun optičnega toka v sliki smo izbrali diferenčno metodo Lucas-Kanade z izvajanjem po piramidnih slojih [11], ki predpostavi konstanten optični tok v bližnji okolici preiskovane slikovne točke in reši sistem enačb za optični tok za vse slikovne točke v bližnji okolici z uporabo metode najmanjših kvadratov. Zaradi pristopa s piramidnimi sloji se postopek najprej izvede na nivoju nizke ločljivosti in se nato ponavlja na slojih z višjimi ločljivostmi. Značilne točke (značilke), ki smo jih uporabili za določitev vektorjev optičnega toka, smo določili z uporabo algoritma FAST [12].

Značilke FAST smo uporabili le za določanje vektorja globalnega optičnega toka in niso uporabljene za določanje položaja. Za določanje položaja in gradnjo zemljevida smo uporabili umetno postavljene značke (angl. *glyph*), podprte z zbirko GRATF za razpoznavanje in sledenje značk (angl. *Glyph Recognition And Tracking Framework*). Značke, podprte z zbirko GRATF, imajo obliko črnega kvadrata na beli podlagi, ki je razdeljen v enako število vrstic in stolpcev znotraj črne obrobe (*slika 3*).

V knjižnici GRATF [13] se za razpoznavo značk izvorna slika (*slika 3* – levo zgoraj) pretvori v sivinske nivoje (slika 3 – desno zgoraj) in upragovi s fiksnim pragom (*slika 3* – levo spodaj). Upragovljena slika vsebuje le črne in bele slikovne točke, nad katerimi se izvede iskanje robov (*slika 3* – desno spodaj), ki točke na robovih (torej točke, ki imajo vsaj eno izmed sosednjih štirih točk drugačne vrednosti) obarva z belo barvo in vse ostale s črno barvo. Povezane bele slikovne točke na robovih (torej bele slikovne točke, ki mejijo na vsaj še eno belo slikovno točko) se nato združijo v skupke, ki se filtrirajo po velikosti tako, da se izločijo vse, ki vsebujejo manj kot *10* slikovnih točk po višini ali širini.

Obdelava značk se nato nadaljuje na vsakem posameznem razpoznanem skupku. Najprej se poiščejo obrobne točke in se preveri, če te točke predstavljajo obliko štirikotnika. Nato se preveri, če je kontrast med točkami zunaj štirikotnika in točkami robu v sivinski sliki nad fiksnim pragom. Če skupek zadosti vsem pogojem, se nad njim s pomočjo obrobnih točk izvede transformacija, ki slikovne



Slika 3. *Prikaz korakov pri razpoznavanju umetno postavljenih značk*

točke znotraj štirikotnika z linearno transformacijo preslika v normiran kvadrat. Notranjost kvadrata se nato razdeli v enako število vrstic in stolpcev (v našem primeru *5* vrstic in *5* stolpcev) in določi povprečna vrednost svetlosti slikovnih točk znotraj posamezne celice (preseka vrstice in stolpca). Povprečna svetlost celic se nato upragovi in se uporabi kot binarni zapis, ki ustreza identifikaciji značke ter zasuka. Identifikacija značke se primerja z zapisi v bazi znanih značk in v primeru najdene korelacije se značka označi in vpiše v bazo najdenih značk v sliki. Baza najdenih značk vsebuje zapis o položaju značke v sliki ter njeni orientaciji.

Za določitev položaja kamere v referenčnem koordinatnem sistemu na podlagi razpoznanih značilk v našem primeru predpostavljamo, da sta koordinatna sistema kvadrokopterja in kamere poravnana in da v koordinatnem sistemu kvadrokopterja poznamo orientacijo »gor-vektorja« (angl. *up-vector*), ki je pravokoten na ravnino *x-y* referenčnega koordinatnega sistema in ga izmerimo direktno z enoto IMU. Ker je ta vektor kolinearen z vektorjem sile gravitacije, orientacijo gor--vektorja ocenimo iz komponent projekcije vektorja gravitacije na ravnino *x-y* koordinatnega sistema kamere. Rešitev v obliki položaja in orientacije kamere smo nato določili na podlagi pristopov, predstavljenih v [14].

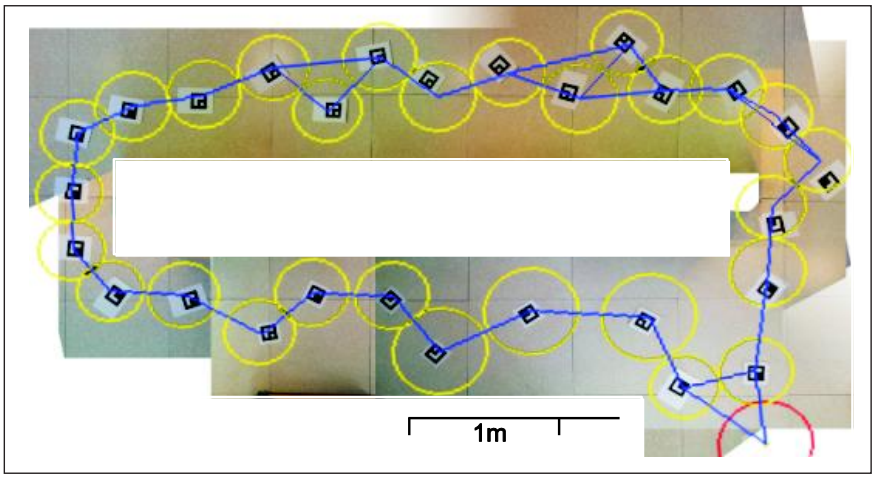
3 Sistem SLAM

Sočasna gradnja zemljevida in določanje položaja v prostoru SLAM [15] prav tako predstavlja enega izmed temeljnih problemov v mobilni robotiki. Če pojmujemo uporabo podatkov s senzorjev kot zaznavanje okolja, lahko pojmujemo gradnjo in uporabo zemljevida kot zavedanje robota o okolju in njegovega položaja v njem. Zemljevid okolja robotu omogoča, da se v njem premika samostojno, torej v okolju poišče primerno pot za premik od trenutnega do zahtevanega položaja. Algoritem SLAM omogoča, da robot med premikanjem po okolju sistematično zbira znanje o okolju v obliki zemljevida. Postopek temelji na dveh korakih – določanju trenutnega položaja v okolju na podlagi izmerkov senzorjev in do tedaj poznanega zemljevida ter izpopolnjevanju zemljevida okolja na podlagi istih izmerkov senzorjev. Spremenjeni postopek SLAM, ki smo ga uporabili v obravnavanem sistemu, smo zasnovali predvsem za uporabo z majhnimi brezpilotnimi zračnimi plovili v zaprtih prostorih in brez zunanjega sistema za določevanje položaja. Postopek temelji na algoritmu FastSLAM [8], pri katerem so se avtorji lotili problema SLAM s statističnega pogleda z uporabo verjetnosti. Ugotovili so, da je za problem SLAM značilna pogojna neodvisnost pri posameznih korakih problema, torej poznavanje položajev in orientacij kamere povzroča, da so posamezne meritve značilk v okolju pogojno neodvisne. Na podlagi te ugotovitve so avtorji problem SLAM razstavili na ločene korake ocenjevanja položaja kamere ter ocenjevanja položajev značk v okolju, pogojenih z oceno položaja kamere. Pri tem pristopu so uporabili prilagojen filter delcev za ocenjevanje a posteriori ocene položajev kamere, hkrati pa v vsakem od delcev filtra uporabili po en Kalmanov filter za vsako značko v zemljevidu.

Razvita rešitev za problem SLAM je rekurziven postopek, ki se izvede po vsaki analizirani sliki s kamere.

Sestavljen je iz naslednjih korakov:

- Za vsakega od *M* - delcev v filtru delcev:
  - izvedi predikcijo položaja delca na podlagi optičnega toka in spremembe kota sukanja,
  - pretvori parametre razpoznanih značk iz lokalnega koordinatnega sistema kvadrokopterja v referenčni koordinatni sistem,
  - poišči pare med razpoznanimi značkami in značkami v shranjenem zemljevidu značilk,
  - izračunaj oceno položaja kamere na podlagi meritev in najdenih parov ter ponovno pretvori parametre razpoznanih značk v globalni koordinatni sistem na podlagi izračunane ocene položaja kamere,
  - oceni verjetnost meritve,
  - izvedi predikcijo in korekcijo nad značkami v zemljevidu, za katere so bili najdeni pari,
  - posodobi graf povezav med značkami,
  - poišči pare med razpoznanimi značkami brez parov in začasnim zemljevidom značk, dodaj značke brez parov v obeh zemljevidih v začasni zemljevid, razpoznane značke s parom v začasnem zemljevidu pa prestavi v zemljevid značk,
  - izvedi premik ocene položaja, če nobeni razpoznani znački ni bil najden par,
  - v primeru dodane nove značke



Slika 4. *Kolaž slik, ki prikazujejo postavitev značk v laboratoriju s prikazanim zgrajenim zemljevidom*

v zemljevid značk z verjetnostjo $p = 0,2$ izvedi postopek iskanja in zapiranja zank.
- Normiraj uteži delcev in izračunaj efektivno število delcev N_{ef}.
- Izvedi ponovno vzorčenje z algoritmom *izbira z zamenjavo*, če je $N_{ef} < 0,5\ M$ ali če se ponovno vzorčenje ni izvedlo v zadnjih *100* iteracijah.
- Poišči delec z največjo utežjo in ga obravnavaj kot najbolj verjetno hipotezo o položaju in zemljevidu.

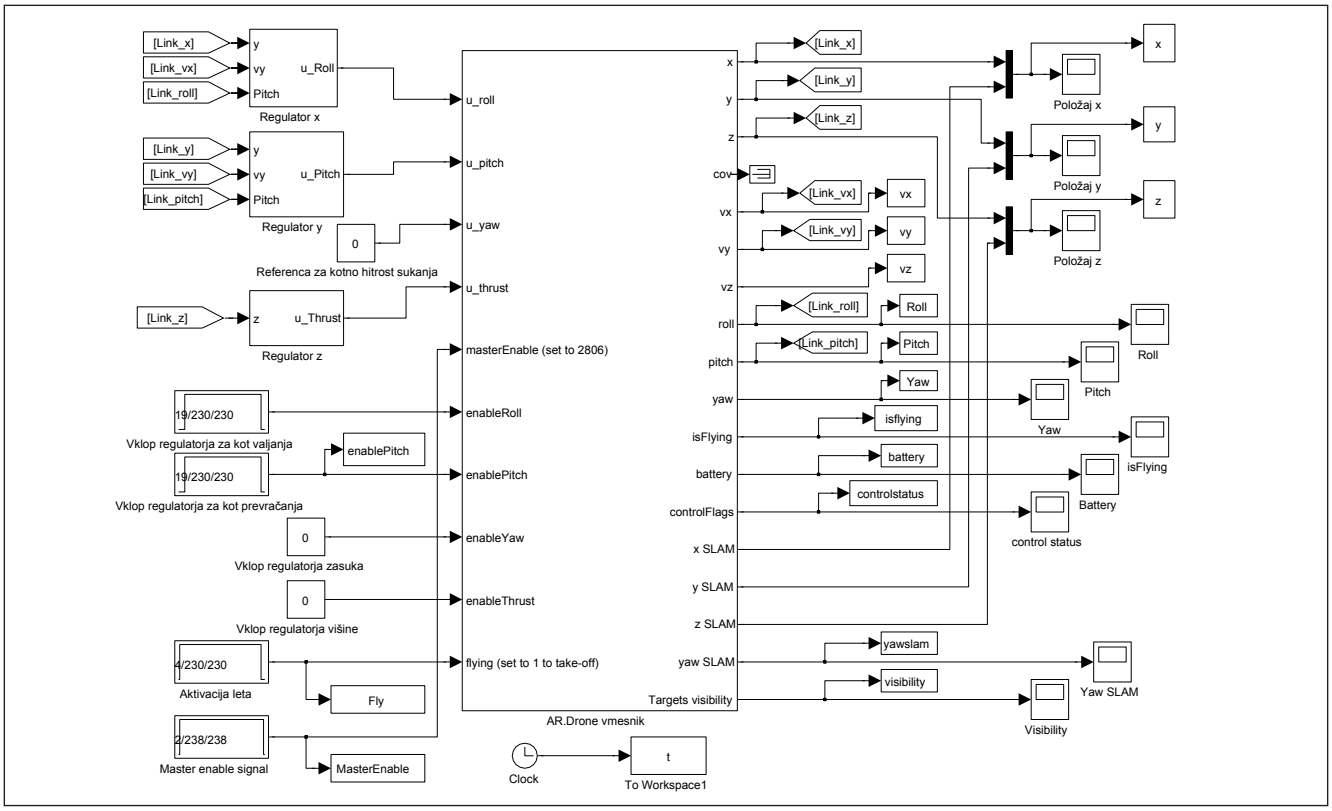
Rezultat izvajanja algoritma SLAM v laboratoriju je prikazan na *sliki 4*.

4 Filtriranje položaja

Rezultate sistema za računalniški vid smo uporabili za izvajanje predikcijskih (na podlagi izmerjenega optičnega toka) in korekcijskih (na podlagi razpoznanih značk) korakov sistema SLAM. Zaradi zakasnitev (med *100* in *200* ms) pri prenosu in računalniški obdelavi slike s kamere smo oceno položaja s strani sistema SLAM uporabili za izvajanje korakov korekcije filtra položaja, korake predikcije tega filtra pa smo izvajali na podlagi matematičnega modela kvadrokopterja ter regulacijskih signalov, poslanih kvadrokopterju [9] – kotu valjanja, kotu prevračanja, kotni hitrosti sukanja ter spremembi višine. Izhodi filtra položaja so nato bili uporabljeni kot ocena trenutnega stanja sistema v sklopu regulatorja. Izvajanje opisanega filtra položaja kvadrokopterja je bilo izvedeno pri časovnem koraku $T = 20$ ms. V isti zanki smo poleg samega filtra položaja kvadrokopterja izvedli tudi zaznavanje trkov z ovirami in sledenje trajektoriji, vendar se bomo v nadaljevanju omejili na vmesnik za vodenje kvadrokopterja preko okolja Matlab Simulink.

5 Vmesnik Matlab Simulink

Simulacijsko okolje Matlab Simulink temelji na grafičnem programiranju in je pogosto uporabljeno za simulacije in vodenje. Načrtali smo maskiran funkcijski blok, ki vsebuje vmesnik za vodenje in zaznavanje kvadrokopterja AR.Drone in omogoča preizkušanje različnih algoritmov vodenja (primer sheme prikazan na *sliki 5*). Pod masko funkcijskega bloka »AR.Drone vmesnik« je zakrita Matlab S-funkcija tipa MEX (angl. *Matlab Executable*), ki s po-



Slika 5. *Primer realizacije sistema vodenja v okolju Matlab Simulink*

močjo vtičnic Winsock preko protokola UDP komunicira z aplikacijo *AR.Drone control* (ta skrbi za komunikacijo s kvadrokopterjem, izvajanje sistema računalniškega vida, sistema SLAM, filtriranja položaja in vodenje). Časovna sinhronizacija simulacijske sheme v okolju Matlab Simulink in ostalih sistemov v aplikaciji *AR.Drone control* temelji na podlagi izvajanja zanke, opisane v prejšnjem poglavju.

Poleg signalov o stanju, položaju, hitrosti in orientaciji kvadrokopterja prenaša vmesnik signale tudi v obratni smeri. Poleg krmilnih vhodov (*u_roll*, *u_pitch*, *u_yaw* ter *u_thrust*) dodatni vhodi (označeni z *masterEnable*, *enableRoll*, *enablePitch*, *enableYaw* in *enableThrust*) omogočajo vklapljanje in izklapljanje posameznih regulatorjev v aplikaciji *AR.Drone control*, vhod *flying* pa ima funkcijo aktivacije izvedbe samodejnega vzleta ali pristanka.

6 Zaključek

Predstavljeni sistem poenostavlja preizkušanje in razvoj algoritmov vodenja kvadrokopterja. Okolje Matlab Simulink je bilo izbrano predvsem zato, ker ga študentje že sedaj uporabljajo pri opravljanju praktičnega dela v sklopu laboratorijskih vaj. Tako je predstavljena rešitev uporabna tudi kot atraktivna popestritev vaj pri predmetih, ki se ukvarjajo z identifikacijo, modeliranjem in vodenjem sistemov.

Literatura

[1] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel in L. Eck: "Image-Based Visual Servo Control of the Translation Kinematics of a Quadrotor Aerial Vehicle," *IEEE Transactions on Robotics*, no. C, str. 2005–2008, 2009.

[2] F. Bonin-Font, A. Ortiz in G. Oliver: "Visual Navigation for Mobile Robots: A Survey," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, str. 263–296, May 2008.

[3] E. Altug, J. P. Ostrowski in R. Mahony: "Control of a quadrotor helicopter using visual feedback," v *Proceedings 2002 IEEE International Conference on Robotics and Automation*, 2002, str. 72–77.

[4] N. Guenard, T. Hamel in R. Mahony: "A Practical Visual Servo Control for an Unmanned Aerial Vehicle," *IEEE Transactions on Robotics*, vol. 24, no. 2, str. 331–340, Apr. 2008.

[5] Parrot: "AR.Drone Parrot," 2011. Spletni naslov: http://ardrone.parrot.com/. [Dostopano: 11-Jul-2011].

[6] T. Krajník, V. Vonásek, D. Fišer in J. Faigl: "AR-Drone as a Platform for Robotic Research and Education," v *Research and Education in Robotics: EU-Robot 2011*, 2011, p. (in press).

[7] S. Piskorski, N. Brulez in P. Eline: "AR.Drone Developer Guide (SDK 1.7)," 2011.

[8] M. Montemerlo, S. Thrun, D. Koller in B. Wegbreit: "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," v *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002, str. 593–598.

[9] M. Bošnak, D. Matko in S. Blažič: "Quadrocopter Hovering Using Position-estimation Information from Inertial Sensors and a High-delay Video System," *Journal of Intelligent & Robotic Systems (in press)*, 2012.

[10] D. J. Fleet in Y. Weiss: "Optical Flow Estimation," in *Mathematical models for Computer Vision: The Handbook*, N. Paragios, Y. Chen, in O. Faugeras, Eds. Springer, 2005, str. 239–258.

[11] J. Bouguet: "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm," *Intel Corporation Microprocessor Research Labs*, vol. 1, no. 2, str. 1–9, 2000.

[12] E. Rosten in T. Drummond: "Machine learning for high-speed corner detection," v *European Conference on Computer Vision, 2006*, str. 430–443.

[13] A. Kirillov: "Glyphs' recognition," *AForge.NET*, 2010. Spletni naslov: http://www.aforgenet.com/articles/glyph_recognition/.

[14] Z. Kukelova, M. Bujnak in T. Pajdla: "Closed-form solutions to the minimal absolute pose problems with known vertical direction," v *ACCV'10 Proceedings of the 10th Asian conference on Computer vision - Volume Part II*, 2010, str. 216–229.

[15] H. Durrant-whyte in T. Bailey: "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," str. 1–9, 2006.

Vision-based quadrotor control

Abstract: In recent years, Unmanned Aerial Vehicles (UAVs) are employed in many applications, from military operations to civilian tasks. The requirement that mobile robots become independent of external sensors, such as GPS, and are able to navigate in an environment by themselves means that designers have few alternative techniques available. An increasingly popular approach is to use computer vision as the source of information about the surroundings. This article presents an overview of controlling the quadrotor's position from Matlab Simulink, using low-cost, off-the-shelf components with the computer vision system and artificial markers placed in the environment. Especially small quadrotors are attracting a lot of attention, a fact that led to the desire to use the quadrotors as part of the laboratory practice for the undergraduate students. For this purpose, a Matlab Simulink interface was designed that allows easy implementation and testing of various control algorithms.

Key words: quadrotor, autonomous system, computer vision, simultaneous localization and mapping