Uporaba sodelujočega robota in obogatene resničnosti na primeru aplikacije sestavljanja

Simon Erjavec, Matjaž Mihelj, Marko Munih, Sebastjan Šlajpah

Izvleček:
Robotske sodelovalne aplikacije omogočajo združevanje prednosti robota in operaterja, kar pripomore k hitrejšemu, enostavnejšemu in hkrati bolj učinkovitemu izvajanju nalog. Razvili smo sistem, ki temelji na robotu ABB YuMi in omogoča interaktivno sodelovanje robota in operaterja za izvedbo naloge sestavljanja. Pristop je nadgrajen z uporabo robotskega vida ter obogatene resničnosti za posredovanje informacij operaterju med izvajanjem naloge. Integrirani sistem smo preizkusili na primeru sestavljanja mehanizma za linearni pomik.

Ključne besede:
sodelujoči robot, sodelovalna aplikacija, sestavljanje, robotski vid, obogatena resničnost

1 Uvod

Z željo po razvoju, optimizaciji dela in višji produktivnosti se v industriji pristop k proizvodnji hitro spreminja. Veliko je poudarka na pametnih sistemih, ki so krmiljeni s pametnimi mrežami, preko pametnih programskih orodij, vse to pa lahko opredelimo kot pametno proizvodnjo [1].

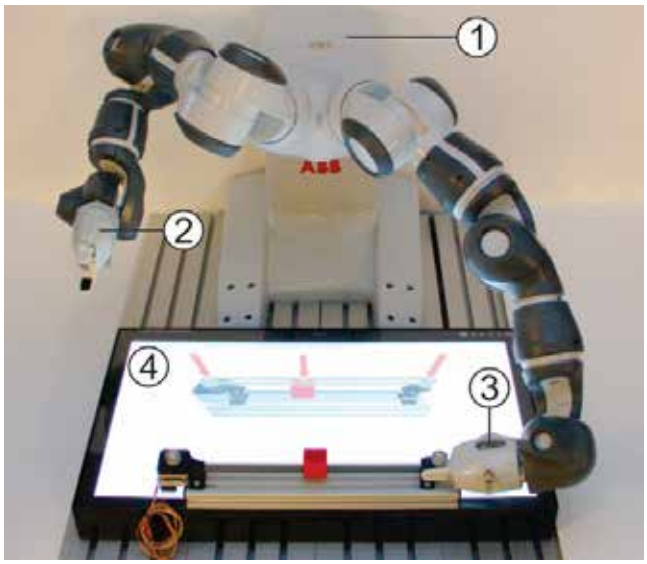
Industrija 4.0 predstavlja četrto razvojno stopnjo proizvodnje v industriji, ki se od prejšnjih razvojnih stopenj razlikuje predvsem v tem, da jo sestavljajo kibernetsko-fizični sistemi. Za razliko od avtomatizacije klasičnih industrijskih tovarn se kot ključne tehnologije napredne industrije uporabljajo strojno učenje, analiza velike količine podatkov, komunikacija z ostalimi napravami, napredna robotika, hibridno-aditivne proizvodnje, virtualne tovarne in digitalni dvojčki [1, 2]. Dva pomembna vidika industrije 4.0 sta tudi sodelovanje robota z operaterjem in obogatena resničnost.

V okviru Centra za sodelujočo robotiko (*www.cobotic.si*, Laboratorij za robotiko, Fakulteta za elektrotehniko, Univerza v Ljubljani) smo razvili testno sodelovalno aplikacijo za primer sestavljanja. Koncept sodelovalne aplikacije operaterju, ki sodeluje z robotom, olajša delo, saj ga vodi, zmanjša skrb za pravilno izvedbo naloge, poveča produktivnost ter izboljša sodelovalno izkušnjo. S tem namenom smo nadgradili vodenje robota, delovanje robotskega vida in izdelali sistem za prikaz obogatene resničnosti.

Simon Erjavec, mag. inž. el., **prof. dr. Matjaž Mihelj**, univ. dipl. inž. el., **prof. dr. Marko Munih**, univ. dipl. inž. el., **dr. Sebastjan Šlajpah**, univ. dipl. inž. el.; vsi Fakulteta za elektrotehniko, Univerza v Ljubljani

2 Robotska celica

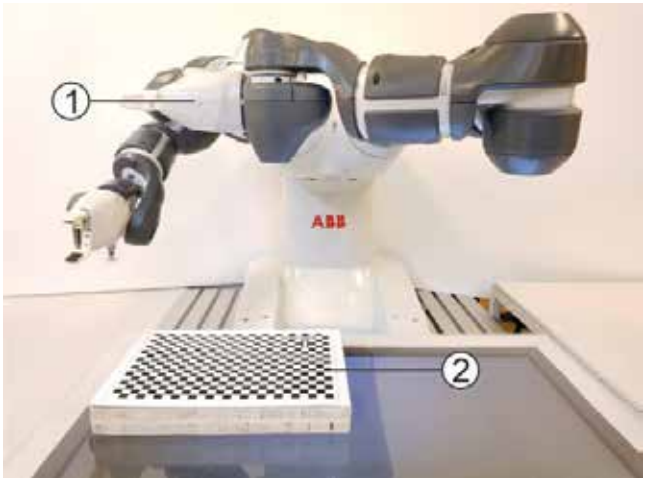
Robotsko celico sestavlja dvoročni sodelujoči robot ABB IRB 14000 – YuMi, ki je prikazan na *sliki 1*. Vsaka roka ima 7 prostostnih stopenj, ponovljivost 0,02 mm, maksimalno hitrost vrha 1,5 m/s in nosilnost 0,5 kg. Robot ima implementirano detekcijo trka na osnovi identificiranega dinamičnega modela, za dodatno varnost pa poskrbi zasnova brez možnih točk uklešćitve prstov ter dodana penasta



Slika 1 : *Prikaz uporabljenih naprav: 1 – robot ABB YuMi, 2 – prijemalo SmartGripper, 3 – kamera Cognex, 4 – računalniški zaslon*

zaščita segmentov. Robotski roki sta opremljeni s prijemali SmartGripper (servo prijemalo in pnevmatski sesek), levo prijemalo pa ima integrirano še kamero proizvajalca Cognex. V sistem smo vključili še računalniški zaslon, ki smo ga uporabili za implementacijo obogatene resničnosti.

Robot ABB YuMi je že v osnovi opremljen s programskim okoljem RobotStudio, ki omogoča tako sprotno (ang. online) kot predhodno (ang. offline) programiranje robota. Na voljo je tudi razširitev In-Sight Vision za načrtovanje programa z uporabo robotskega vida. Kot dodatno programsko okolje, ki je služilo za nadzor robota, smo uporabili programski paket Matlab, grafiko za obogateno resničnost pa smo realizirali v programu Unity.



Slika 2 : *Prikaz kalibracije kamere: 1 – prijemalo SmartGripper z integrirano kamero, 2 – kalibracijska mreža na višini 3 cm*

3 Gradniki sistema

Sodelovalna aplikacija temelji na treh gradnikih sistema, ki skrbijo za vodenje robota, robotski vid in obogateno resničnost.

3.1 Vodenje robota

Uporaba ročne učne enote v kombinaciji s programskim okoljem RobotStudio omogoča uporabniku enostavno in intuitivno programiranje robota po principu od točke do točke. Osnovno možnost programiranja smo razširili z uporabo dodatnega zunanjega programskega okolja Matlab, ki nam je omogočilo izgradnjo fleksibilnega sistema in kompleksnega vodenja. Okolje Matlab je bilo uporabljeno kot nadzorno okolje, znotraj katerega je glavni program oziroma logika vodenja sistema za izvedbo nalog. Povezava med okoljem Matlab in okoljem RobotStudio je potekala preko komunikacije TCP/IP. Uporabnik preko nadzornega okolja pošilja sporočila v programsko okolje RobotStudio, kjer so definirani primeri odziva robota glede na vsebino sporočila. Program, ki teče na robotu, tako predstavlja vnaprej sprogramirano fiksno programsko strukturo, ki omogoča posredno vodenje robota in ne potrebuje dodatnega urejanja in dograjevanja.

3.2 Robotski vid

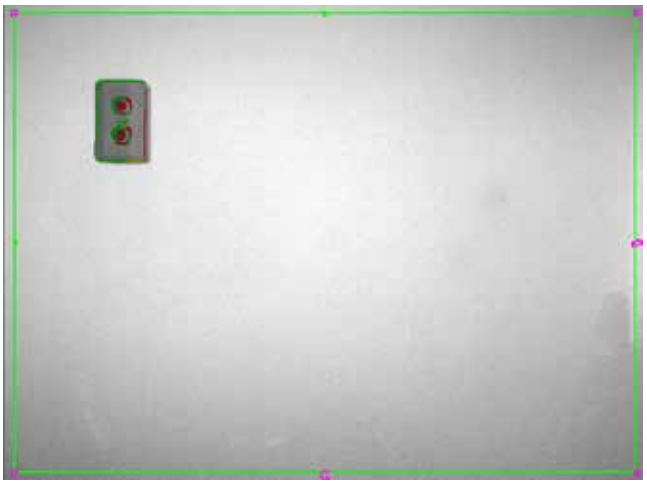
Največji poudarek pri izvedbi sodelujoče aplikacije je bil na robotskem vidu. Želeli smo namreč implementirati sistem, ki bi omogočal visoko stopnjo prilagodljivosti, zanesljivost iskanja objektov skozi celotno delovno površino robota (pri različnih položajih kamere) ter enostavno definiranje objektov zanimanja. Po navodilih proizvajalca je potrebno za vsako lego kamere narediti novo kalibracijo, območje iskanja pa je pogojeno z vidnim poljem kamere. Razširitve področja iskanja objektov skozi celoten delovni prostor robota smo se lotili s transformacijo osnovnega vidnega polja kamere, znotraj katerega bi moralo v osnovi potekati iskanje. Z iskanjem objektov v razširjenem delovnem prostoru se je pojavila napaka pri prijemu objekta. Identificirali smo tri stvari, ki prispevajo k tej napaki. To so bile napake zaradi globine, pogleda s strani in odstopanja lege robotske roke.

Napaka zaradi globine

Najprej smo opazili napako zaradi napačnega določanja oddaljenosti ravnine pri kalibraciji. Kalibracijo smo namreč izvajali s kalibracijsko mrežo, ki smo jo položili na delovno mizo. S kalibracijsko mrežo smo določili vrednost pretvorbe slikovnih pik v milimetre, ki se spreminja z razdaljo med ravnino in objektivom. Ker je višina sestavnih delov okoli 3 cm, se je napaka prijema z oddaljenostjo objekta od središča spreminjala. Napako zaradi globine smo zmanjšali s ponovno kalibracijo kamere, pri čemer smo kalibracijsko mrežo dvignili za 3 cm, kot je prikazano na *sliki 2*. Na ta način smo zagotovili ustrezno pretvorbo slikovnih elementov v metrične enote. Ker pa so bili objekti različnih višin, s to metodo nismo popolnoma odstranili napake prijemanja.

Napaka zaradi pogleda s strani

Kot drugo napako smo identificirali napako zaradi pogleda s strani; opazen primer napake je prikazan na *sliki 3*. S slike je razvidno, da se ujemanje vzorca, ki smo ga določili kot naš model objekta znotraj okolja In-Sight, zmanjšuje z oddaljenostjo od središča. Zaradi pogleda s strani se namreč vzorec objekta spremeni, s tem pa se zgubijo določene značilke, na katerih temelji prepoznava modela. V primeru, da je pogled s strani preveč spremenil vzorec, zaznava objekta odpove. S slike

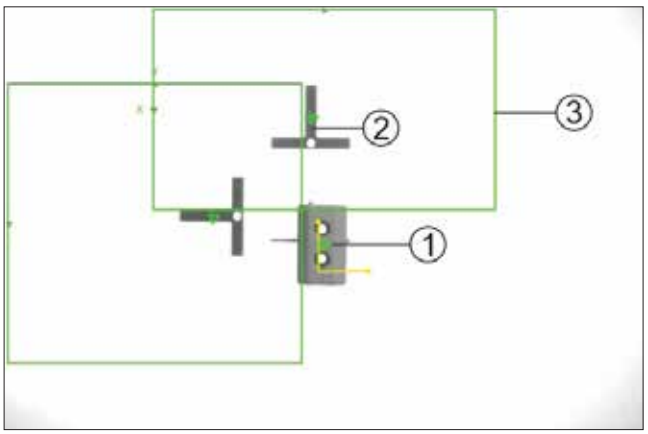


Slika 3 : *Prikaz objekta, ki ga kamera zajame s strani; z rdečimi linijami so označene značilke, ki so bile določene med učenjem objekta, na iskanem objektu pa ti robovi niso jasni zaradi pogleda s strani.*

je razvidna tudi sprememba vzorca zaradi višine objekta: višji objekti pri pogledu s strani postanejo širši, kot so v resnici. Napako zaradi pogleda s strani smo izničili z avtomatsko poravnavo kamere na robotski roki glede na iskani objekt. Lega prepoznanega objekta Ta je po končani poravnavi v središču zajete slike (direktno pod kamero), kot je razvidno s *slike 4*.

Napaka zaradi odstopanja lege robotske roke

V nekaterih konfiguracijah robotske roke smo pri zajemanju slike opazili, da prihaja do napačnega prijemanja zaradi napačne orientacije kamere. Vzrok je bila napačna interpretacija direktne kinematike robota, predvsem okoli singularnih leg. Da bi to napako izničili, smo ob detekciji objekta okoli objekta na zaslonu izrisali dva kalibracijska markerja, kot je prikazano na *sliki 4*. Nato smo ponovno zajeli sliko z namenom, da prepoznamo in določimo lego teh markerjev. Na podlagi primerjave dejanske lege s prepoznano lego markerjev smo določili napako lege, ki je bila vzrok napačne orientacije kamere. S kompenzacijo te napake pri prepoznavi iskanega objekta smo zmanjšali napako določevanja referenčne pozicije za prijemanje ter tako izboljšali samo izvedbo prijemanja.

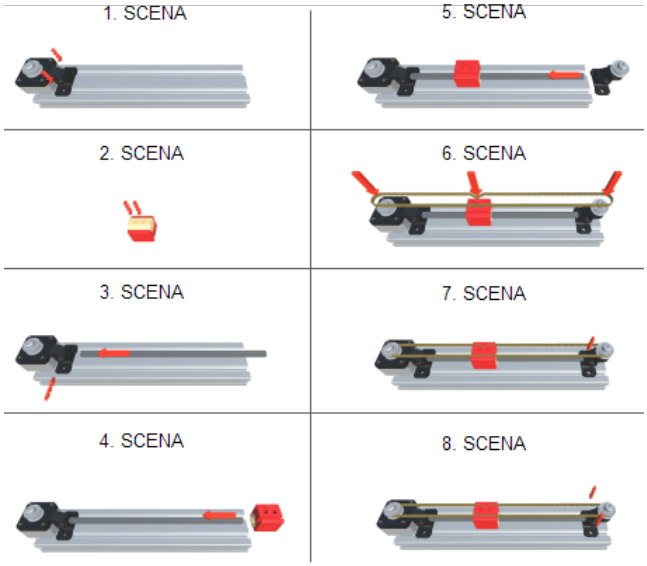


Slika 4 : *Prikaz novega pristopa k zaznavi lege objekta s pomočjo markerjev: 1 – iskani objekt, 2 – kalibracijski marker, 3 – območje iskanja kalibracijskega markerja*

3.3 Obogatena resničnost

Za zagotavljanje ustrezne sekvence sestavljanja smo operaterju med izvedbo naloge prikazovali trenutno stanje naloge. Pri fazah sestavljanja, kjer je bilo zahtevano sodelovanje operaterja, smo operaterju z obogateno resničnostjo jasno prikazali navodila na zaslonu ter ga tako vodili skozi postopek izvedbe naloge. Sodelovanje operaterja je zajemalo naloge, za katere nismo imeli namenske opreme robota (npr. vijačenje elementov), naloge vstavljanje, kjer zaradi variabilnosti sestavnih delov in visokih toleranc izdelave robot ni bil sposoben sestavljanja, ne da bi sprožil varnostne funkcije ustavitve zaradi prekoračenih obremenitev, ter manipulacijo z upogljivimi elementi (jermenica).

Vizualna navodila znotraj okolja Unity so razdeljena na več različnih scen, ki s premiki 3D objektov ponazarjajo potrebne korake za izvedbo naloge. Vizualizacijo v obogateni resničnosti sestavlja osem različnih scen, ki se posamezno prikazujejo, ko je potreba po operaterjevem sodelovanju. Na vsaki sceni je naloga operaterja jasno ponazorjena z animiranimi rdečimi puščicami. Vseh osem scen je prikazanih na *sliki 5*.

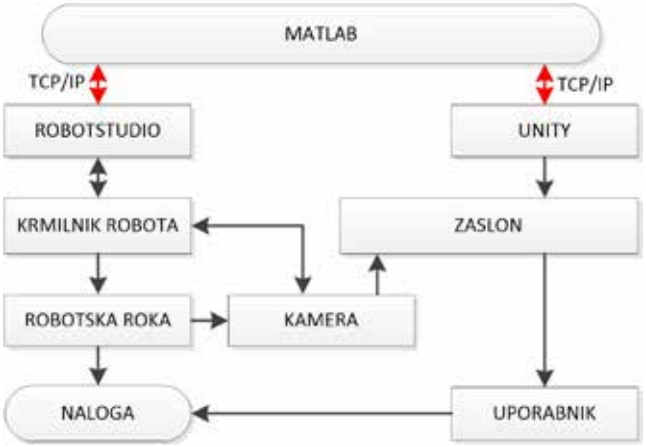


Slika 5 : *Navodila za izvedbo naloge; naloge operaterja so jasno označene z animiranimi rdečimi puščicami.*

4 Integracija sistema

Princip delovanja sistema je zajet na *sliki 6*. Nadzor gibanja robota in sistema za obogateno resničnost potekata znotraj nadzornega okolja Matlab, ki preko komunikacije TCP/IP upravlja z ostalimi programskimi okolji. V sklopu robotskega programa glavna rutina skrbi za sprejem in interpretacijo ukaza, prejetega iz Matlab okolja, ter poskrbi za ustrezen odziv robotskega sistema s klicem vnaprej pripravljenih podrutin.

Nadzorno okolje Matlab poskrbi tako za ustrezno iskanje objektov kot tudi za odpravljanje napak pri iskanju in prijemanju objektov. Pri tem poteka komunikacija tudi z okoljem Unity, ki poskrbi za ustrezen izris kalibracijskih markerjev na zaslonu. Na podlagi povratne informacije iz robotskega krmilnika nadzorno okolje Matlab proži ustrezne elemente v okolju Unity za vizualno komunikacijo z uporabnikom za obveščanje o stanju izvedbe naloge ter podajanje ustreznih postopkov sestavljanja.



Slika 6 : *Prikaz delovanja sistema*

5 Zaključek

Industrijski robotski krmilniki so največkrat omejeni na nabor osnovnih ukazov, ki ne omogočajo optimalnega razvoja naloge, pri kateri je potrebno aktivno sodelovanje človeka. Predstavljena nadgradnja sistema je vključevala dodatno programsko opremo kot tudi ustrezne naprave za večjo fleksibilnost sistema in možnost dodatne komunikacije z operaterjem. Sistem poleg osnovnih funkcionalnosti omogoča tudi implementacijo kompleksnejših principov vodenja. Dodaten zaslon je hkrati predstavljal delovno površino in pripomoček za podajanje informacij operaterju. Uporabili smo ga tudi za zmanjševanje napak pri zaznavi objektov, kar je še dodatno izboljšalo delovanje sistema. Predstavljene rešitve prinašajo nov pogled k reševanju problematike stika človeka in robota pri izvajanju skupne naloge, kjer se daje velik poudarek na pozitivni izkušnji uporabnika pri sodelovanju z robotom.

Viri

[1] N. Herakovič, »Nekateri tehnološki izzivi industrije 4.0«, Fakulteta za strojništvo, LASIM, Ventil, vol. 22, no. 1, str. 10–16, 2016.
[2] D. Bourne, »My boss the robot«, Scientific American, vol. 308, no. 5, str. 38–41, 2013.
[3] Ž. Majdič, »Uporaba in primeri uporabe simulacijskega okolja ABB RobotStudio«, strokovni članek, Fakulteta za strojništvo, LASIM, št. 1, letn. 15, str. 76–79, 2009.

Demonstration of collaborative assembly based on collaborative robot with augmented reality

Abstract:
Collaborative application merges the best properties of the robot and the human operator. It this way tasks are performed faster, more efficiently and with less cognitive effort. A collaborative assembly task based on an ABB YuMi robot system was developed. The system was upgraded with robot vision and augmented reality to provide enhanced feedback to the operator. The overall system functionality was demonstrated by assembling a one- degree-of-freedom linear mechanism.

Keywords:
collaborative robot, collaborative application, assembly, robot vision, augmented reality

Zahvala

Raziskovalni program št. P2-0228 je sofinancirala Javna agencija za raziskovalno dejavnost Republike Slovenije iz državnega proračuna. Gradniki, orodja in sistemi za tovarne prihodnosti / GOSTOP, OP20.00361, št. pogodbe C3330-16-529000, 2016-20, naložbo sofinancirata Evropska unija iz Evropskega sklada za regionalni razvoj in Republika Slovenija.