StuCoSReC

Proceedings
of the 2018
5th Student
Computer
Science
Research
Conference

StuCoSReC

Proceedings of the 2018 5th Student Computer Science Research Conference

Edited by

Iztok Fister jr. and Andrej Brodnik

Reviewers and Programme Committee

Klemen Berkovič • University of Maribor, Slovenia
Lucija Brezočnik • University of Maribor, Slovenia
Patricio Bulić • University of Ljubljana, Slovenia
Zoran Bosnić • University of Ljubljana, Slovenia
Borko Bošković • University of Maribor, Slovenia
Janez Brest • University of Maribor, Slovenia
Andrej Brodnik • University of Primorska and University
of Ljubljana, Slovenia

Haruna Chiroma • Gombe State University, Nigeria
Mojca Ciglarich • University of Ljubljana, Slovenia
Jani Dugonik • University of Maribor, Slovenia
Iztok Fister • University of Maribor, Slovenia
Iztok Fister Jr. • University of Maribor, Slovenia
Matjaž Gams • Jozef Stefan Institute, Slovenia
Tao Gao • North China Electric Power University, China
Andres Iglesias • Universidad de Cantabria, Spain
Branko Kavšek • University of Primorska, Slovenia
Miklos Kresz • University of Szeged, Hungary
Niko Lukač • University of Maribor, Slovenia
Uroš Mlakar • University of Maribor, Slovenia
Marjan Mernik • University of Maribor, Slovenia
Eneko Osaba • University of Deusto, Spain
Vili Podgorelec • University of Maribor, Slovenia
Peter Rogelj • University of Primorska, Slovenia
Xin-She Yang • Middlesex University, United Kingdom
Damjan Vavpotič • University of Ljubljana, Slovenia
Grega Vrbančič • University of Maribor, Slovenia
Aleš Zamuda • University of Maribor, Slovenia
Nikolaj Zimic • University of Ljubljana, Slovenia
Borut Žalik • University of Maribor, Slovenia

Published by

University of Primorska Press
Titov trg 4, SI-6000 Koper

Editor-in-Chief

Jonatan Vinkler

Managing Editor

Alen Ježovnik

Koper, 2018

ISBN 978-961-7055-26-9 (pdf)

www.hippocampus.si/ISBN/978-961-7055-26-9.pdf

ISBN 978-961-7055-27-6 (html)

www.hippocampus.si/ISBN/978-961-7055-27-6/index.html

DOI: <https://doi.org/10.26493/978-961-7055-26-9>

© University of Primorska Press

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in
univerzitetni knjižnici v Ljubljani

COBISS.SI-ID=296769024

ISBN 978-961-7055-26-9 (pdf)

ISBN 978-961-7055-27-6 (html)

Preface

It is no longer necessary to emphasize that computer science is omnipresent today. Terms as digitalization, Industry 4.0 etc. are frequently heard in the mass media and are becoming a part of our everyday life. The rapid advances in computer science require a large investment in research and development. In particular, it is important to educate young scientists, which is one of the objectives of the StuCoSReC conference.

In front of you is the proceedings of the fifth StuCoSReC conference. The StuCoSReC is intended for masters & PhD students to show their research achievements. Socializing is also important and the conference is an excellent opportunity for students to get to know each other. The conference connects students of computer science and goes beyond the invisible limits of faculties. The uniqueness of StuCoSReC conference is that it is organized each year by different higher education institution. The University of Ljubljana - Faculty of Computer and Information Science is proud to host the conference this year.

Twelve papers addressed this conference, covering several topics of the computer science. All the papers were reviewed by two international reviewers and accepted for the oral presentation.

ref. prof. dr. Nikolaj Zimic

Preface

II

Papers

<i>Regulacija plovca v zračnem tunelu (zračna levitacija) na osnovi mehke logike</i> • Primož Bencak, Dominik Sedonja, Andrej Picej, Alen Kolman, Matjaž Bogša	5–13
<i>Razpoznavanje literature v dokumentih in določanje njihovih sklicev v besedilu</i> • Matej Moravec, David Letonja, Jan Tovornik, Borko Boškovič	15–19
<i>Klasifikacija samomorilskih pisem</i> • Dejan Rupnik, Denis Ekart, Gregor Kovačevič, Dejan Orter, Alen Verk, Borko Boškovič	21–25
<i>Načrtovanje poprocesorja za pretvorbo NC/ISO G-kode v translacije za 3/5 osnega robota ACMA ABB XR701</i> • Gregor Germadnik, Timi Karner, Klemen Berkovič, Iztok Fister	27–33
<i>Using constrained exhaustive search vs. greedy heuristic search for classification rule learning</i> • Jamolbek Mattiev, Branko Kavšek	35–38
<i>Real-time visualization of 3D digital Earth</i> • Aljaž Jeromel	39–42
<i>Towards an enhanced inertial sensor-based step length estimation model</i> • Melanija Vežočnik, Matjaž Branko Jurič	43–46
<i>Breast Histopathology Image Clustering using Cuckoo Search Algorithm</i> • Krishna Gopal Dhal, Iztok Fister Jr., Arunita Das, Swarnajit Ray, Sanjoy Das	47–54
<i>Time series classification with Bag-Of-Words approach post-quantum cryptography</i> • Domen Kavran	55–59
<i>The problem of quantum computers in cryptography and post-quantum cryptography</i> • Dino Vlahek	61–64
<i>Named Entity Recognition and Classification using Artificial Neural Network</i> • Luka Bašek, Borko Boškovič	65–70
<i>Context-dependent chaining of heterogeneous data</i> • Rok Gomišček, Tomaž Curk	71

Načrtovanje poprocesorja za pretvorbo NC/ISO G-kode v translacije za 3/5 osnega robota ACMA ABB XR701

Gregor GERMADNIK^{*}
Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenia
gregor.germadnik
@student.um.si

Timi KARNER[†]
Fakulteta za strojništvo
Smetanova ulica 17
2000 Maribor, Slovenia
timi.karner@um.si

Klemen BERKOVIČ
Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenia
klemen.berkovic1@um.si

Iztok FISTER
Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenia
iztok.fister@um.si

POVZETEK

V moderni dobi imamo na voljo večje število robotov, ki prevzema vedno večje število opravil v industriji, saj so poceni in preprosti za montažo. Po navadi je potrebno z dobavo robota kupiti tudi programske pakete oz. programsko opremo, ki omogoča programiranje robotovih funkcij, ter predstavlja največji strošek nakupa. V tem članku zato prikazujemo, kako z implementacijo poprocesorja za pretvorbe NC/ISO G-kode v translacije 3/5 osnega robota ACMA proizvajalca ABB model XR701 ta strošek zmanjšamo. Cilji doseženi v članku so: razvoj poprocesorja, ki z linearno algebro in z razčlenjevanjem pretvori program za rezkanje kompleksnejših oblik večjih dimenzij, pohitritev nastavitve in pretvorbo kode z uporabo uporabnikom prijaznega vmesnika izdelanega v programski opremi Visual Studio 2017 CLI (angl. *Combined Language Infrastructure*) in programskim jezikom C++. Glavni dosežen cilj članka je varčevanje pri nakupu licence programske opreme za pretvarjanje ISO G-kode v robotsko kodo.

Ključne besede

stroj CNC, programski jezik C++, ISO G-koda, linearna algebra, program NC, poprocesor, robot

^{*}Dopisni avtor

[†]Pomoč pri upravljanju robota ACMA XR701

1. UVOD

V preteklosti se roboti v industriji niso tako številno uporabljali kot pa se v sedanjem svetu industrije, saj so jih prekašali t.i. računalniško-numerično krmiljeni stroji (angl. *Computer Numerical Control*, krajše CNC), ki že imajo točno določeno natančnost in zagotavljajo dolgoročno ponovljivost. Z razvojem krmilnikov PID (angl. *Proportional-Integral-Derivative controller*) in harmoničnih gonil, ki zagotavljajo natančnejše določanje položajev osi robota, je robotizacija zacvetela. Danes se lahko nekateri roboti že približajo enostavnejšim strojem CNC, vendar jih zaradi reguliranja več prostorskih stopenj težko presegajo. Zato se roboti večinoma uporabljajo za manj natančne ponovljive, nevarne, težke in velikokrat monotone operacije, ki zmanjšujejo poklicne bolezni pri človeku. Zaradi eksponentne rasti novih robotov in njihove pripadajoče programske opreme je postala uporaba starejših modelov robotov različnih proizvajalcev vse zahtevnejša. Nova programska oprema se ne ujema več z mehanskimi lastnostmi robota, kar omejuje nakup robota istega proizvajalca.

Za vodenje robotov oz. strojev CNC uporabljamo program NC pridobljen s poprocesiranjem položajev orodja (angl. *Cutter Location*, krajše CL). Poprocesor je opredeljen kot vmesno orodje in omogoča obdelavo podatkov med programsko opremo za računalniško podprto načrtovanje (angl. *Computer Aided Design*, krajše CAD) in računalniško podprto proizvodnjo (angl. *Computer Aided Manufacturing*, krajše CAM), ki pripravi program za proizvodne stroje. Z njim zagotavljamo vodenje proizvodnega stroja glede na načrtovan izdelek in je unikaten glede na mehanske lastnosti stroja.

Na področju našega raziskovalnega dela lahko najdemo naslednja sorodna dela. Sekirnik [11] v svoji diplomski nalogi opisuje, kako s programsko opremo Siemens NX 8.5 poprocesira podatke s sistemov CAD in CAM v G-kodo s pomočjo programa, narejenega v programski opremi Microsoft Office Excel, in kot izhod generira numerični program za dodajanje točk. Nastavitev izhodnega programa in robota je zahtevna

naloga, saj je potrebno imeti predznanje tako o robotu kakor tudi o formatu numeričnega krmilnega programa.

V magistrskem delu Filipič [2] opisuje uporabo programske opreme RobotStudio, kjer se v virtualnem okolju ustvari model robota ACMA XR701, ki omogoča učenje robota brez poseganja v industrijske procese. Izhodni program je zapisan s programsko kodo ABB, iz katere avtor poprocesira vrednosti za robotsko kodo ACMA iz ukazov *MoveJ* in *MoveL*. Magistrsko delo Nilssona [8] in članek [10] po drugi strani opisujeta, kako lahko z ukazov *G2* in *G3* v G-kodi generiramo ukaz *MoveC* za krožni gib robota ABB.

V našem članku opisujemo razvoj dodatnega poprocesorja, ki s pridobljeno kodo (tj. programom NC ali ISO G-kodo) izvede ponovno pretvorbo v numerični jezik razumljiv programu za dodajanje točk. Vhod predlaganega poprocesorja je program v NC oz. ISO G-kodi za 3/5 osnega robota in podatki obdelave (npr. lokacija polizdelka in postavitev mehanizma) pridobljenih s pomočjo grafičnega uporabniškega vmesnika (angl. *Graphical User Interface*, krajše GUI). Ta s pomočjo razčlenitve izlušči s programa translacije in iz podatkov GUI strukturira numerični program za učenje točk robota. Obdelava kode 5 osnega robota je zelo preprosta, saj podatke izlušči in jih ustrezno oblikuje glede na *G* ukaze in podatke obdelave. Težava se pojavi pri kartezični G-kodi za 3 osnega robota, kjer je krožna giba ukazov *G2* in *G3* potrebno preoblikovati v dovolj majhne translacije oz. interpolacije, ki ustrezajo natančnosti robota. Ta problem smo rešili z izračunom ustreznih vektorjev glede na ukaz krožnega giba. Določitev kota med dvema vektorjema smo izračunali s funkcijo *atan2* (inverzni tangens 2) izpeljano s pomočjo trigonometričnih funkcij in določanjem točk krožnega izseka z rotacijsko matriko. Po poprocesiranju je treba ponovno obdelati podatke s programom, ki iz programa za učenje točk pretvori točke v numerični jezik robota.

Struktura članka je v nadaljevanju naslednja. V poglavju 2 predstavimo robota ACMA XR701. Opis predlagane rešitve je vsebina poglavja 3. Rezultati so predstavljeni v poglavju 4. Članek zaključimo s poglavjem 5, kjer povzamemo opravljeno delo in napovemo možnosti za nadaljnje delo.

2. ROBOT ACMA XR701

Robot ACMA XR701 je bil zasnovan leta 1994 v podjetju Renault Automation. To je antropomorfní industrijski robot s šestimi prostostnimi stopnjami in velikim delovnim prostorom. Razvili so ga za manipulacijo, strego in sestavo, zaradi njegove zanesljivosti, hitrosti in natančnosti pa ga uporabljamo predvsem za uporovno varjenje. Omogočal je uspešne avtomatizirane rešitve tudi v kosovni proizvodnji. [3]

Robot je v obliki paralelograma, ki omogoča večjo nosilnost zaradi večje vztrajnosti in nižjega težišča. [1, 6] Premešča mase na šesti osi od 125 kg do 270 kg. Zgornja roka je lahko obremenjena z največjo maso do 160 kg. Vse osi so opremljene s trifaznim sinhronim električnim brezkrtačnim motorjem Brushless, ki omogočajo hitrosti do 3 m/s. Motorji so opremljeni z resolverji, ki v vsakem trenutku daje informacijo o položaju osi. Za doseganje večje natančnosti ima vsak servomotor vgrajeno dodatno elektromagnetno zavoro in reduktor [3]. Zaradi starosti robota in obrabe mehanskih elementov je natančnost robota 0,3 mm.

Njegovi glavni členi so prikazani na Sliki 1.

Slika 1: Prikaz robota ACMA ABB XR701 [3]

Legenda:

1. Podstavek robotskega mehanizma	8. Tretji člen robota (3. os)
2. Motor z gonilom (1. os)	9. Motor z gonilom (4. os)
3. Vrtljivo podnožje	10. Sklep drugega in tretjega člena (5. os)
4. Motor z gonilom (2. os)	11. Pritrdilna prirobnica (6. os)
5. Protiutež	12. Prenosna konzola
6. Ravnotežna vzmet	13. Krmilna omara
7. Drugi člen robota	

2.1 Primerjava CNC stroja z robotom

Z vidika avtomatizacije je obdelovalni stroj CNC avtomatiziran sistem, ki samostojno obdeluje polizdelek po zapisani ISO G-kodi ali programu NC. Najpreprostejši obdelovalni stroji omogočajo pomikanje v smeri *x*, *y* in *z* (3 osni stroji). Imenujemo jih kartezični, saj so brez rotacijskih osi in težko obdelujejo kompleksnejše oblike, razen v primeru uporabe kroglastega rezkarja. Z dodajanjem dodatnih osi oz. rotacij lahko obdelujemo kompleksnejše oblike. Zaradi njihove togosti dosegajo večjo natančnost in ponovljivost ter manjšo obrabo kot roboti [5]. Njihova slabost je majhen delovni prostor za večji delovni prostor je treba povečati konstrukcijo stroja in izbrati material in obliko elementov konstrukcije, ki so odporni na upogib za doseganje dobre natančnosti. Zaradi njegovega pravokotnega delovnega prostora in konstrukcije prav tako težko opravlja zahtevnejše obdelovalne položaje, s čimer pa robot nima težav, saj je delovni prostor po katerem ga lahko poljubno vodimo sferičen in dostopen s poljubno postavitev robota.

2.2 Poprocesor za pretvorbo G-kode

Obdelovalni stroji postajo vedno bolj zahtevni, saj se v industriji pogosto uporabljajo površine poljubnih oblik. Ročno programiranje kompleksnejših površin je v tem primeru nesmiselno. Zato se v ta namen uporablja programska oprema za pretvorbo modela CAD v ISO G-kodo ali program NC 3/5 osnega obdelovalnega stroja. Pri tem mesto rezila CL pridobivamo neposredno z modela CAD, ki smo ga poprocesirali v programski opremi CAM.

Pri komunikaciji med sistemom CAD/CAM in strojem NC lahko večkrat pride do težav zaradi nekompatibilnosti, zato je potrebno za vsako orodje oz. stroj ustvariti unikaten poprocesor oz. vmesnik za pretvorbo podatkov CL v ustrezen

numerični strojni jezik [4]. V našem primer smo to izvedli z implementacijo poprocesorja, ki pretvori G-kodo oz. program NC v točke za učenje robota ACMA in z obsoječim programom pretvori pridobljene točke v numerični strojni jezik ACMA. Proces pretvorbe prikazuje Slika 2.

Slika 2: Prikaz pretok podatkov

Zaradi enostavnosti uporabljamo programsko opremo CAD/CAM, ki že pripravi G-kodo oz. program NC glede na uporabljeno orodje. Za pretvorbo G-kode v jezik za učenje robota ACMA moramo vhodno kodo razčleniti in po potrebi s pomočjo interpolacije izračunati dodatne točke gibov. Izhodno kodo, ki predstavlja program za učenje točk robota, mora biti tudi primerno strukturirana. Po obdelavi, program za učenje točk ponovno pretvori kodo v numerično robotsko kodo ACMA.

3. RAZVOJ POPROCESORJA

Razvoj poprocesorja za pretvorbo G-kode v robotski jezik 3/5 osnega robota ACMA je sestavljen iz treh faz:

- kreiranje uporabniškega vmesnika,
- pretvorbe G-kode v program robota ACMA
- evaluacija.

V nadaljevanju opisujemo omenjene faze podrobneje.

3.1 Uporabniški vmesnik

Težave v modernem svetu so vidne pri kompleksnosti naprav in programov, saj jih praviloma uporabljajo manj izobraženi kadri. Zato se za pohitritev in zmanjševanje napak pri krmljenju strojev CNC in robotov ACMA uporablja namenski uporabniški vmesnik, v našem primeru predstavljeni kot grafični uporabniški vmesnik, ki natančno opredeli nastavljanje parametrov stroja in s tem olajša njegovo uporabo.

V našem primeru smo razvili GUI, preko katerega lahko nastavimo položaj polizdelka, konfiguracijo robota in druge

manj specifične nastavitve, kot sta, na primer, natančnost robota in nastavitve natančnosti števil s plavajočo vejico, kjer nastavimo število decimalnih mest koordinat. Z GUI smo z aplikacijskimi vmesniki (angl. *Application Programming Interface*, krajše API) izvajali pretvorbo, pridobivali vhodno G-kodo in shranjevali pridobljen program ACMA za učenje točk in razčlenjeno G-kodo s klikom na gumb. Za razvoj vmesnika smo uporabili programsko okolje Visual studio 2017 in izdelali GUI s pomočjo programskega paketa Windows Forms C++/CLI, ki ga ponuja uporabljeno razvojno okolje. C++/CLI je jezikovna specifikacija, ki jo je razvil Microsoft je namenjena nadomeščanju upravljalnih (angl. *Managed*) razširitev za C++ [7, 9], katere namen je poenostaviti starejšo upravljalno sintakso C++, ki je zdaj opuščena. C++/CLI je standardizirana v standard ECMA-372. Trenutno je na voljo, kot programski paket za Visual Studio 2005 - 2017, vključno z izdajami Express.

3.2 Pretvorba G-kode v program robota ACMA

Pretvorba G-kode v program 3/5 osnega robota ACMA temelji na uporabi standardnih knjižnic, ki nam olajšajo delo s pisanjem funkcij. To je zbirka razredov in funkcij napisanih v baznem jeziku C++. Osnovo pretvorbe predstavlja razred *Pridobitev_Koordinat*, ki je opremljen z javnimi metodami za arhiviranje, upravljanje s spremenljivkami pridobljenih iz vhodne G-kode oz. programa NC in podatkov iz GUI. Vsebuje tudi zasebne metode za pridobivanje vrednosti iz vhodnih podatkov, zasebne metode za preračunavanje krožnih gibov v ustrezne interpolacije. Delovanje glavnega aplikacijskega vmesnika "Pretvori vhodno G-kodo" je predstavljeno s Pseudokodom 1. Kjer spremenljivke z velikimi tiskanimi črkami predstavljajo sistemski nizi pridobljeni iz vmesnika GUI. Pseudokod 1 vsebuje tudi klice javnih metod, katerih ime je ločen s podčrtaji.

Algoritem 1 Aplikacijski vmesnik za pretvorbo G-kode

```

1: String^ AcmaIzpis, GkodaIzpis;
2: Pridobitev_Koordinat Koord;
3: Koord.Vhodni.Podatki(VHGKODA, NATANC, RELX,
  RELY, RELZ, RELA, RELB, RELC, STDEC, V, A,
  RAMA, KOMOLEC, ZAPESTJE);
4: OUTACMA->Clear();
5: OUTGKODA->Clear();
6: AcmaIzpis = Koord.Izpis_Glave_Prog_ACMA();
7: OUTACMA->AppendText(AcmaIzpis);
8: Koord.Clear_Data();
9: while Koord.Zakljucni_Znak() is not ture do
10:   Koord.Pridobitev_Vrstice();
11:   if Koord.Prazna_Vrstica() is not true then
12:     AcmaIzpis = Koord.Izpis_Koord_Prog_ACMA();
13:     GkodaIzpis = Koord.Izpis_Vrstice_Gkode();
14:     Koord.Prepis_Podatkov();
15:     OUTGKODA->AppendText(GkodaIzpis);
16:     OUTAMCA->AppendText(AcmaIzpis);
17:     Koord.Clear_Data();
18:   end if
19: end while
20: AcmaIzpis = Koord.Izpis_Noge_Prog_ACMA();
21: OUTACMA->AppendText(AcmaIzpis);
22: Koord.Clear_Data();
  
```


Omenjeni vmesnik GUI se sestoji iz treh glavnih delov:

- zajem podatkov in strukturiranje numeričnega programa ACMA za učenje točk,
- razčlenjevanje podatkov in preverjanje zapolnjenost vrstice in
- pretvorba krožnih izsekov v translacije, kjer je to potrebno.

V nadaljevanju članka opisujemo omenjene glavne dele podrobneje.

3.2.1 Zajem podatkov in struktura numeričnega programa za učenje točk za robota ACMA XR701

Robota ACMA programiramo prek medija (magnetno zapisana zgoščenka) s sintakso za dodajanje točk. Za programiranje moramo zapisati v niz določeno zaporedje znakov za dodajanje točk linearnih gibov oz. translacij. V ta namen strukturiramo glavo, vsebino točk in nogo programa in je zgrajena z zaporedja klikov, ki so opredeljeni v Tabeli 1.

Tabela 1: Vrednosti tipk konzole za programiranje robota ACMA

Izbira menija	'x'	F1	@59
Vrednost	"x"	F6	@64
Tab	09	F9	@67
Enter	13	F10	@68
Esc	27	Tipka desno	@77

Ničelno točko polizdelka, postavitev robota, točke gibov oz. translacij in druge manj pomembne podatke pridobimo z javno metodo *Vhodni.Podatki*. Te podatke pridobimo prek GUI in jih arhiviramo v razred *Pridobitev_Koordinat*. Pred uporabo je nize potrebno ustrezno pretvoriti glede na uporabo zasebnih metod. Z metodami *Izpis_Glave_Programa_ACMA*, *Izpis_Koordinat_Programa_ACMA*, *Izpis_Vrstice_Gkode* in *Izpis_Noge_Programa_ACMA* polnimo niz znakov, glede na strukturo opredeljeno s Sliko 3 in Sliko 4 ter z Algoritmom 2, ki vrnejo sistemski niz za izpis v bogato tekstovno polje *RichTextBox*. Za ponoven izpis smo z javno metodo *Clear_Data* počistili spremenljivki za izpisovanje izhodne G-kode in programa ACMA za učenje točk ter priredili spremenljivkam znotraj razreda vrednost 0, ki določajo prazno vrstico in pridobivajo vrednosti posamezne koordinate, ki opisuje podglavje 3.2.2.

```
27 27 +0 27 +0 '3' +0 '1' +0
@80 @80 @80 @80 @80 @80 @80 @80 +0 13
13 13 13 27
@64 "x"09"x"09"x"09"x"09"x"09"x"09"x" +@68
```

Slika 3: Sekvenca za nastavitve ničelne točke

```
'8' @77 13 +0 13 +0 @77 @77 @77 13 +0
```

Slika 4: Sekvenca za nastavitve načina dodajanje koordinat

Algoritem 2 Pseudokoda za določanje sekvence za nastavitve konfiguracije robota

```
1: if Rama is Levo then
2:   AcmaIzhod.append("13 +0 13 +0\n");
3: else if Rama is Spredej then
4:   AcmaIzhod.append("13 +0\n");
5: end if
6: if Komolec is Zgoraj then
7:   AcmaIzhod.append("@77 13 +0\n");
8: end if
9: if Zapestje is Nizko then
10:  AcmaIzhod.append("@77 @77 13 +0\n");
11: end if
```

Pri določitvi konfiguracije je potrebno biti pozoren na transformacijo koordinatnega sistema, saj se pri normalni konfiguraciji (Rama:levo, Komolec:zgoraj in Zapestje:nizko) koordinatni sistem robotske celice transformira os x za 180° in os z za 180° v koordinatni sistem orodja, prikazano na Sliki 5.

Slika 5: Koordinatni sistemi robota [3]

Vsebina programa ACMA je sestavljena z vpisovanjem točk in poteka na enak način, kot dodajanje ničelne točke. Razlikuje se v tem, da še dodamo hitrost v in pospešek a prikazano na Sliki 6.

```
@64 "X"09"Y"09"Z"09"A"09"B"09"C" +@68
13 +0 "v" 13 +0 "a" 13 +0
```

Slika 6: Sekvenca za določanje točk, hitrosti in pospeška

Noga zaključi program s sekvenco prikazano na Sliki 7.

```
27 27 27 27 27 +0 @59
```

Slika 7: Sekvenca noge programa ACMA za dodajanje točk

3.2.2 Razčlenjevanje podatkov in preverjanje zapolnjenosti vrstice

Za dodajanje novega izdelka v industriji operater brez predznanja težko uporablja G-kodo različnih struktur. V ta namen smo strukturirali metodo za branje ključnih podatkov s poljubnih G-kod.

Za razčlenjevanje podatkov niza smo napisali javno metodo *Pridobitev_Vrstice*, ki preverja znake vrstice do ukaza *G*, nato se preverjanje nadaljuje do parametrov *X*, *Y*, *Z*, *A*, *B*, *C*, *I*, *J* in *K* kar je prikazano v Algoritmu 3. Za vsak parameter se z zasebno metodo *Pridobi_Vrednost* zapisujejo znaki v medpomnilnik, dokler so enaki številom, piki ali minusom. Metoda vrne medpomnilnik, ki se pretvori v tip (*long double*), s pomočjo funkcije *stof* (angl. *string to float*), prikazano v Algoritmu 4.

Algoritem 3 Razčlenjevanje vhodne G-kode

```

1: while TrenutniZnak is not \n do
2:   if TrenutniZnak is G
3:     and (NaslednjiZnak is Stevilo 0 – 3) then
4:       Nastavi na naslednji znak;
5:       Vrstica[0] = Trenutna vrednost;
6:       UkazIniciliziran = true;
7:       UkazVpisan[0] = true;
8:       Zmanjšaj StPraznihPolj;
9:   end if
10:  if TrenutniZnak is X
11:    and NaslednjiZnak is Stevilo
12:    and UkazVpisan[0] is true
13:    and UkazIniciliziran is true then
14:      Nastavi na naslednji znak;
15:      Vrstica[1] = Pridobi_Vrednost();
16:      UkazVpisan[1] = true;
17:      Zmanjšaj StPraznihPolj;
18:    end if
19:  //Ponovitev za parametre Y, Z, A, B, C, I, J, K
20:  Nastavi na naslednji znak;
21: end while

```

Algoritem 4 Pridobivanje vrednosti parametrov

```

1: string Medpomnilnik;
2: while TrenutniZnak is Stevilo do
3:   Medpomnilnik += TrenutniZnak;
4:   Nastavi na naslednji znak;
5: end while
6: Nastavi na prejšnji znak;
7: return stof(Medpomnilnik);

```

Za določanje prazne vrstice preverimo polje binarnih vrednosti *UkazVpisan* in *StPraznihPolj* pridobljenih v Algoritmu 3. Če pogoji, ki so predstavljeni v Algoritmu 5 veljajo, vrstico preskočimo oz. je ne izpišemo. To je prikazano v Algoritmu 1 z javno metodo *Prazna_Vrstica*, ki vrne dvojisko vrednost.

Algoritem 5 Preverjanje zapolnjenosti vrstice

```

1: if StPraznihPolj is 10 or (VpisanPodatek[0] is true
2:   and StPraznihPolj is 9) then
3:   PraznaVrstica = true;
4: else
5:   PraznaVrstica = false;
6:   for i ∈ 0, ..., 6 do
7:     if VpisanPodatek[i] is false then
8:       Vrstica[i] = PredhodnaVrstica[i];
9:     end if
10:  end for
11: end if

```

3.2.3 Pretvorba krožnih izsekov *G2* in *G3*

Kot je bilo že omenjeno, v GUI dodajamo program NC za 3/5 osnega robota. Program za 5 osnega robota preprosto vrednosti polja *Vrstica* prišteje nastavitve robota in strukturira izhodni program ACMA za dodajanje linearnih točk z določenimi rotacijami. Določanje koordinat gibov v programu NC za 3 osnega robota je zahtevnejše, saj je potrebno za ukaza *G2* oz. *G3*, ki predstavljata krožni izsek, določiti ustrezne interpolacije oz. odseke in jih definiranti glede na natančnost robotovih gibov. V ta namen smo napisali zasebni metodi *G2_Algebra* in *G3_Algebra* za izračun giba *G2* ter *G3*.

Za izbiranje izpisa se je uporabila izjava (switch case), ki glede na vhodno celo število *G* ukaza (0, 1, 2, 3) izvede določen primer. *G0* in *G1* izpišeta vhodne podatke v izhoden niz z ustrezno strukturo in ukaza *G2* in *G3* se pred izpisom pretvorita v ustrezne translacijske gibe.

Za lažjo predstavbo smo pretvorbo *G2* in *G3* krožnega giba predstavili z realnim primerom, prikazanim na Sliki 8 in Sliki 9.

N0	G1	X0	Y0		
N1	G2	X5	Y5	I0	J5
N2	G3	X10	Y10	I5	J0

Slika 8: Primer NC programa

Slika 9: Primer G-koDE oz. NC programa krožnega izseka *G2* in *G3*

Izračun interpolacij oz. odsekov bomo izvedli z linearno algebro in trigonometričnimi funkcijami iz študijske litera-

ture [12]. Iz podatkov določimo vektorja, ki izhajata s središča krožnega izseka do predhodnih koordinat ZV (začetni vektor) ter do trenutnih koordinat KV (končni vektor) prikazana na Sliki 9. Začetni vektor je definiran kot nasprotna smer vektorja.

$$ZV = \begin{bmatrix} x_{zv} \\ y_{zv} \\ z_{zv} \end{bmatrix} = - \begin{bmatrix} i \\ j \\ k \end{bmatrix} \quad (1)$$

Končni vektor, ki je definiran kot razlika začetne oz. predhodne koordinate in vektorja središča krožnega izseka od trenutne koordinate.

$$KV = \begin{bmatrix} x_{kv} \\ y_{kv} \\ z_{kv} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{pred} \\ y_{pred} \\ z_{pred} \end{bmatrix} - \begin{bmatrix} i \\ j \\ k \end{bmatrix} \quad (2)$$

Radij krožnega izseka se izračuna s pomočjo skalarne produkta in sicer dolžina vektorja $\vec{a}$ z vrednostmi i , j in k razvidno s Slike 9.

$$r = |\vec{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2} = \sqrt{i^2 + j^2 + k^2} \quad (3)$$

Kot med vektorjema θ je razlika od inverznega tangensa začetnega vektorja in končnega vektorja. Kot vektorja je izražen vadianih, med $-\pi$ in π prikazano s Enačbo (4).

$$\theta = \arctan \frac{y}{x} \quad (4)$$

Pri tem so se pojavile težave, če je bila vrednost imenovalca x enaka 0, saj takrat ulomek ni definiran.

Za rešitev težave s Slike 10 predpostavimo, da velja $y = \sin \theta$ in $x = r + \cos \theta$ in določimo novo Enačbo (5) ter jo preuredimo v Enačbo (6). S pomočjo skalarne produkta in Pitagorovega izreka izračunamo polmer krožnega izseka in s tem dobimo končno Enačbo (7).

$$\frac{\theta}{2} = \arctan \frac{y}{r + x} \quad (5)$$

$$\arctan(x, y) = \theta = 2 \frac{\theta}{2} \quad (6)$$

$$\theta = 2 \arctan \frac{y}{\sqrt{x^2 + y^2} + x} \quad (7)$$

Z Enačbo (8) lahko iz obsega kroga določimo lok l oz. krožni izsek kota θ .

Slika 10: Prikaz izpeljave inverznega tangensa 2

$$l = 2 \cdot \pi \cdot r \cdot \frac{\theta}{360} \quad (8)$$

S tem izračunamo število interpolacij z dolžino loka l v odvisnosti od natančnosti robota in določimo kot interpolacije s kotom med vektorjema v odvisnosti od števila interpolacij. Preostane nam še izračun posamezne interpolacije, ki se izvede z rotacijsko matriko. Pri tem je treba paziti, po kateri ravnini potuje krožni gib razvidno z G-kode in Algoritma 6.

Algoritem 6 Določanje delovne površine

```

1: if I is NotDefined then
2:   Delovna površina YZ
3: else
4:   if J is NotDefined then
5:     Delovna površina XZ
6:   else
7:     Delovna površina XY
8:   end if
9: end if

```

Za naš primer sta pri $G2$, krožni gib v smeri urnega kazalca, podana parametra I in J, za katerega sledi Enačba (9).

$$\begin{bmatrix} x_{inter} \\ y_{inter} \\ z_{inter} \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{zv} \\ y_{zv} \\ z_{zv} \end{bmatrix} \quad (9)$$

Za gib $G3$ proti smeri urinega kazalca preoblikujemo rotacijsko matriko po Enačbi (10).

$$\begin{bmatrix} x_{inter} \\ y_{inter} \\ z_{inter} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{zv} \\ y_{zv} \\ z_{zv} \end{bmatrix} \quad (10)$$

V ravninah XZ in YZ uporabljamo $G2$ in $G3$ za gibanje v obliki vijačnice (angl. *Helix*). Program ACMA se lahko v

teh ravninah uporabi za ukaza $G2$ in $G3$ v obliki krožnega giba, ki ni standardiziran in giba potujeta v smeri osi z . S tem je treba prilagoditi tudi rotacije robota. Če imamo krožni gib v ravnini XZ , prilagajamo rotacijo okoli osi x (rotacija R_ψ), če imamo gib v ravnini YZ , prilagajamo rotacijo okoli osi y (rotacija R_θ). To izvedemo z izračunom kota interpolacije po Enačbi (10) in tega prištejemo oz. odštejemo od ustrezne rotacije robota. S tem dosežemo, da je rezkar vedno pravokoten na obdelovalno površino in se enakomerno obrablja orodje ter odvzema material, kar posledično vpliva na kakovost obdelave.

4. POSKUSI IN REZULTATI

Poprocesor za pretvorbo G-kode je kompleksna aplikacija, saj njegova izdelava traja dlje časa in zahteva popolno poznavanje stroja, za katerega program načrtujemo. V simulacijah deluje idealno, vendar teorija in praksa vedno ne gresta z roko v roki. V praksi običajno hitro naletimo na nepredvidene težave. Iz rezultatov poskusov lahko izluščimo naslednje ugotovitve:

1. Za izdelavo poprocesorja in vmesnika GUI je potrebno obsežno znanje programiranja v jeziku C++/CLI ter dobro poznavanje programskega okolja Visual Studio 2017.
2. Prvotno je bil poprocesor zapisan tako, da je celotno vhodno in izhodno G-kodo skupaj z generiranim programom ACMA shranjeval v pomnilnik. To je zahtevalo 3-krat več pomnilnika od izboljšane verzije, ki obdeluje vrstico po vrstico in jo sproti izpisuje. To se pravi, da pri vsaki vrstici prepiše predhodne podatke in jih med delovanjem izpisuje na izhodno mesto (bogato tekstovno polje), kar privarčuje pomnilnik.
3. S programiranjem poprocesorja smo imeli težave z pridobivanjem ključnih podatkov z vhodne G-kode, saj je ponavadi G-koda bogata z dodatnimi nastavitvami, ki pa v našem primeru niso nujne. Da bi se izognili shranjevanjem neuporabnih podatkov, smo dopisali if stavkom pogoje za zagotavljanje iskanja ključnih podatkov. Za našo pridobljeno G-kodo z CAM programske opreme Siemens NX 8.5 je program deloval brezhibno, G-kode drugih programskih oprem bi lahko povzročale težave.
4. Težave smo imeli tudi z doseganjem natančnih gibov, saj robotu ni uspelo izvesti majhnih oz. kratkih translacij. Ta problem smo odpravili z zmanjšanjem hitrosti in pospeška gibov.
5. Težave pri računanju interpolacij smo imeli ker določenih kotov nismo mogli izračunati zaradi nedefiniranega ulomka. Odpravili smo jo z implementacijo inverznega tangensa 2.
6. Uporaba programa na starejših sistemih lahko predstavlja težave, saj je program preveden za 64-bitno arhitekturo procesorjev. Program smo prevedli tudi za 32-bitno arhitekturo procesorjev, katerem bi morali preveriti združljivost z operacijskim sistemom Windows 98. V nasprotnem primeru je potrebno program prevesti z uporabo programa Microsoft Visual C++ 2005, saj je zapis podatkov v našem primeru v Unicode. Druga

možnost je uporaba MSLU (angl. *Microsoft Layer for Unicode*) za operacijske sistemom Windows 95, 98 in Me, ki bi omogočala delovanje zgrajenega programa.

5. ZAKLJUČEK

Članek opisuje razvoj poprocesorja za pretvorbo G-kode, ki omogoča translacije za 3/5 osnega robota ACMA. Omenjeni poprocesor izpolnjuje vse predpostavke, ki smo si jih zadali na začetku raziskovalnega dela.

Seveda bi ga lahko izboljšali tako, da bi združili poprocesor G-kode in pretvorbo v numerični programski jezik ACMA. Tako bi lahko robotu ACMA implementiral neposredno povezavo preko serijske komunikacije, če robot to omogoča, in ga poljubno programirali oz. vodili. S tem bi omogočili hitrejše prilagajanje programa in zmanjšali čas priprave, ki najbolj vpliva na ekonomičnost, saj med tem časom robot stoji.

Pri implementaciji ukazov $G2$ in $G3$ bi lahko kodo skrajšali tako, da bi prek serijske komunikacije neposredno programirali z ABB programskim jezikom, če robot to omogoča. Za dodajanje giba *MoveC* bi lahko pri določanju vmesne tretje točke krožnega giba uporabili enačbe s poglavja 3.2.3.

6. VIRI IN LITERATURA

- [1] T. Bajd, M. Mihelj, J. Lenarčič, A. Stanovnik, and M. Munih. *Robotika*. Fakulteta za elektrotehniko, 2008.
- [2] M. Filipič. *Posredno programiranje robota ACMA XR701*. Magistrsko delo, Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2014.
- [3] F. Klobučar and J. Zgonc. *Robotika: Vodenje in programiranje robota ABB XR701*.
- [4] R. S. Lee and C. H. She. Developing a postprocessor for three types of five-axis machine tools. *The International Journal of Advanced Manufacturing Technology*, 13(9):658–665, Sep 1997.
- [5] W. Lei and Y. Hsu. Accuracy test of five-axis cnc machine tool with 3d probe-ball. part i: design and modeling. *International Journal of Machine Tools and Manufacture*, 42(10):1153 – 1162, 2002.
- [6] A. Lesnika. *Robotski sistemi : interno gradivo za program mehatronika*. Višja strokovna šola, 2012.
- [7] M. D. Network. *.NET Programming with C++/CLI (Visual C++)*.
- [8] D. Nilsson. *G-Code to RAPID translator for Robot-Studio*. Master of science, University West, Department of Engineering Science, 2016.
- [9] V. Ragunathan. *C++/CLI Primer*. Springer, 2016.
- [10] F. Ribeiro. 3d printing with metals. *Computing Control Engineering Journal*, 9(1):31–38, Feb 1998.
- [11] M. Sekirnik. *Priprava robotiziranega sistema ACMA XR701 in obračalne mize za postopke 3D frezanja*. Diplomsko delo, Univerza v Mariboru, Fakulteta za strojništvo, 2015.
- [12] B. Z. Sovič Tina, Simon Špacapan and R. Erveš. *Matematika 1 : skripta*. Technical report, Univerzitetna založba, 2018.

University of Primorska Press
www.hippocampus.si
ISBN 978-961-7055-26-9