StuCoSReC

# Proceedings of the 2018 5th Student Computer Science Research Conference

University of Primorska Press

StuCoSReC
Proceedings of the 2018 5th Student Computer Science Research Conference

*Edited by*
Iztok Fister jr. and Andrej Brodnik

*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

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

## Contents

# Regulacija plovca v zračnem tunelu (zračna levitacija) na osnovi mehke logike

Primož Bencak
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
primoz.bencak@student.um.si

Dominik Sedonja
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
dominik.sedonja@student.um.si

Andrej Picej
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
andrej.picej@student.um.si

Alen Kolman
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
alen.kolman@student.um.si

Matjaž Bogša
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
matjaz.bogsa@student.um.si

## POVZETEK

Teorija mehke logike je prisotna že dobrih 40 let, razvita pa je bila z namenom preprostega in intuitivnega vodenja sistemov. Mehka logika namreč posnema človeško sklepanje ter sprejemanje odločitev na podlagi dejstev. V preteklih desetletjih se je uveljavila kot učinkovita in procesorsko nezahtevna alternativa klasičnim tehnikam vodenja. Čeprav se pojavljajo pomisleki o njeni uporabnosti na zahtevnejših aplikacijah, je vendarle našla svoj prostor v mnogih segmentih našega življenja.

Mehka logika nam omogoča, da lahko sami določimo karakteristiko regulatorja, napram linearnemu PID regulatorju, kjer je le-ta vedno linearna in zato v večini primerov neprimerna za vodenje nelinearnih sistemov, razen v omejenih območjih delovanja. V članku je tako predstavljena regulacija valjastega plovca znotraj zračnega tunela na osnovi mehke logike z dolgim mrtvim časom (angl. *dead time*). Opravljene so bile simulacije omenjenega sistema v MATLAB/Simulinku, izdelano je bilo preizkuševališče za vodenje plovca ter izvedena primerjava med klasično tehniko vodenja in tehniko vodenja z mehko logiko. Preizkuševališče je namenjeno učenju sistemov daljinskega vodenja in preizkušanju različnih tipov regulatorjev.

## Kjučne besede

zračna levitacija, mehka logika, mikrokrmilniki, LabVIEW

## 1. UVOD

*Zračna levitacija* predstavlja mirovanje (v nadaljevanju: lebdenje) telesa v neki točki dvignjeni od tal, zaradi umetno ustvarjene sile zračnega toka, ki je nasprotno enaka sili teže telesa [3]. Lebdenje objekta (plovca) v cevi je s stališča regulacije zahtevnejši problem, saj gre za nelinearen dinamični sistem drugega reda. Za vodenje takega sistema moramo zelo dobro nastaviti parametre PID regulatorja ali se poslužiti drugačnega načina vodenja, kot je mehka logika. Sistem je bil najprej zgrajen in preizkušen v simulacijskem okolju, nato pa je bil prenešen v realno okolje in za dane razmere optimiziran.

Ideja za izdelavo sistema lebdenja izhaja iz članka [3], v katerem avtorji s PID regulatorjem uspešno vodijo takšen sistem. Izrazito nelinearen sistem avtorji linearizirajo v okolici delovne točke, sledi identifikacija modela in nato iz dobljenih formul izpeljejo prenosno funkcijo, ki je vstavljena v uporabniški vmesnik, preko katerega vodijo sistem.

Dušan Fister pa predlaga uporabo naprednih optimizacijskih algoritmov po vzoru iz narave za nastavljanje parametrov PID regulatorja, da dosežemo najboljše razmerje med vzponskim časom, prenihajem in statičnim pogreškom, ter ga preizkusi na dvoosnem SCARA robotu [4].

V članku je predstavljena rešitev, kjer postopka linearizacije in določanja parametrov PID regulatorja ni potrebno izvesti, saj ni potrebno poznati niti funkcije sistema. Uporabili smo mehki regulator, pri čemer kot vhodna podatka uporabimo le oddaljenost od točke *ekvilibriuma* oz. referenčne točke (točka v prostoru stanj sistema, kjer je doseženo ravnotežno stanje sistema) in odvod te razdalje, torej hitrost premikanja objekta.

Mehka logika se uporablja na veliko različnih področjih kot so: aeronavtična industrija [10], avtomobilska industrija (prestavljanje menjalnika [5], strategija nadzora električnega vozila [9]), medicina (večnamenska kontrola anestezije, nadzorovanje arterijskega tlaka med anestezijo) [17], nadzorovanje

DOI: https://doi.org/10.26493/978-961-7055-26-9.5-13

polnjenja in praznjenja baterij [12], procesiranje slik v industriji [1] in še na mnogih drugih.

Struktura članka v nadaljevanju je naslednja: drugo poglavje opisuje teoretično ozadje lebdenja v zraku in mehke logike, tretje poglavje govori o simulaciji sistema v MATLAB/-Simulink-u, v četrtem poglavju je predstavljen postopek načrtovanja fizičnega dela sistema (angl. *hardware*), v petem poglavju je govora o programskem delu (angl. *software*), sledijo še rezultati in na koncu sklep s podanimi možnostmi za izboljšavo obstoječega sistema.

## 2. LEBDENJE V ZRAKU

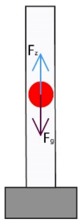
Lebdenje v zraku je v osnovi preprosto opisati, saj mora veljati le: $\sum F = 0$. To pomeni, da je potrebno izpolniti pogoj $F_g = F_z$. Vendar, ko želimo za izbrani sistem na ta način izračunati vrednost $F_z$, ki predstavlja silo ustvarjeno s pomočjo pretoka zraka v cevi, naletimo na težavo, saj moramo za optimalno vodenje poznati razne koeficiente, ki niso konstante (težavo predstavljajo koeficient potiska – odvisen od Reynoldsovega števila; gostota zraka – odvisna od temperature, nadmorske višine, ... in hitrost zraka v cevi). Iz drugega Newtonovega zakona lahko zapišemo ravnotežno enačbo lebdenja v zraku (1) in nato razčlenimo posamezno silo (2).

Enačbe:

$$F = F_z - F_g, \tag{1}$$

$$F = \frac{1}{2} \cdot C_d \cdot \rho \cdot A \cdot (v_w - z')^2 - m \cdot g, \tag{2}$$

pri čemer je $C_d$ koeficient potiska, $\rho$ gostota zraka, $A$ presek objekta na katerega deluje zrak, $v_w$ hitrost zraka v cevi, $z$ višina na kateri se nahaja objekt v cevi, $m$ masa objekta in $g$ težnostni pospešek.

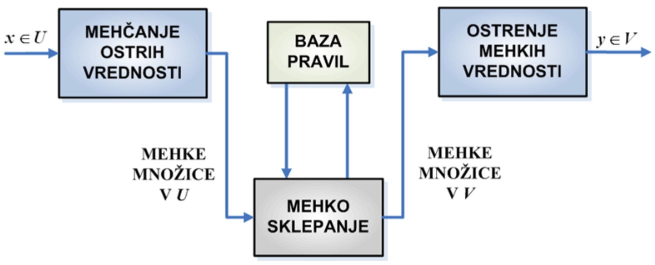


Slika 1: Delovanje sil na objekt v cevi. Vir [3].

### 2.1 Mehka logika

Začetki mehke logike segajo v leto 1985, ko sta to metodo predstavila Takagi in Sugeno. Pri mehki logiki gre za manj matematično zapletene zapise relacij med vhodnimi in izhodnimi spremenljivkami, saj so spremenljivke lahko lingvistične (podane z besedami naravnega jezika – velik, majhen, hiter, počasen, ...). Operacije nad mehkimi spremenljivkami izvajamo z mehkimi operatorji (unija, presek in komplement). Sestaviti pa je potrebno tudi bazo pravil, ki je v osnovi pri mehkih sistemih zgrajena iz "če-potem"(angl. *"if-then"*) mehkih pravil, ki opisujejo vzročno – posledično dogajanje v sistemu. Glede na izhodno spremenljivko ločimo več tipov mehkih pravil, in sicer Mamdani [6] sisteme (vhodi in izhodi so lingvistični), Takagi-Sugeno (mehka pravila, kjer je izhod ostra vrednost) [11] in Singleton (izhod je namesto funkcije kar število). Pisanje baze pravil mehke logike za posamezen sistem ni trivialno kakor nakazuje metoda, saj je potrebno dobro poznavanje sistema, zato pravila pišejo strokovnjaki (angl. *expert*) dotičnega področja. Strokovnjak mora določiti tudi vhodne in izhodne pripadnostne funkcije. V kompleksnejših sistemih lahko število funkcij in pravil preseže zmožnost človeškega uma in strokovnjak ne more zapisati ustreznega števila pravil [13].

V sisteme regulacije s pomočjo mehke logike velikokrat vstopa t.i. ostra vrednost, v obliki podatka iz senzorja (npr. temperatura zraka). To informacijo je potrebno mehčati (angl. *fuzzification*), kar pomeni, da informacijo spremenimo v mehko vrednost (besedno oz. številčno). Ta postopek poteka s pomočjo pripadnostnih funkcij (funkcije, ki določajo pripadnost elementov iz senzorja posamezni mehki množici in lahko zavzamejo vrednost med 0 in 1). Poznamo več vrst pripadnostnih funkcij, in sicer Z, S , $\pi$, sigmoidno, Gaussovo, zvončasto, odsekoma linearno, trikotno in trapezoidno pripadnostno funkcijo. Od sistema in strokovnjaka je odvisno katere pripadnostne funkcije bodo uporabljene v posamezni situaciji. Mehko sklepanje se izvede na podlagi baze pravil. Na ta način se vhodni podatki pretvorijo v izhodne, ki pa so mehke vrednosti. Z mehko vrednostjo računalnik oz. krmilnik ne more operirati, zato je mehko vrednost potrebno ponovno ostriti (angl. *defuzzification*), da dobimo numerično vrednost, s katero lahko računalnik računa oz. na podlagi le-te proži določeno akcijo na aktuatorju. Poznamo več metod ostrenja, in sicer metodo največje vrednosti, težiščno metodo in poenostavljeno težiščno metodo. V praksi se pogosto uporablja poenostavljena težiščna metoda, ki je računsko dokaj preprosta in ne zahteva prevelike procesorske moči [13].



Slika 2: Shematski prikaz mehkega inferenčnega stroja. Vir [13].

Ugotovimo lahko, da se na ta način elegantno izognemo zapletenemu matematičnemu zapisu in reševanju diferencialnih enačb zgolj z dobrim poznavanjem procesa. Kljub temu se pojavljajo pomisleki, ki odvračajo od uporabe mehke logike. Argumenti proti se nanašajo predvsem na to, da ni sistematičnega pristopa k načrtovanju tovrstnih sistemov, pri kompleksnejših sistemih lahko mehka logika postane ovira in ogroža stabilnost sistema [8], ter da predstavlja manjvreden inženirski pristop k vodenju [13].

## 3. SIMULACIJA LEBDENJA V ZRAKU

Za uspešno simulacijo modela je potrebno implementirati naslednje sestavne dele:

- plovec, ki predstavlja reguliran objekt

- regulacijo (regulator in Smithov prediktor)

Prenosna funkcija reguliranega sistema je sestavljena iz dveh sklopov - ventilatorja in reguliranega objekta (plovca). Prenosno funkcijo ventilatorja zapišemo s členom prvega reda, plovca pa s členom drugega reda [3]. Slednjo razvijemo na podlagi ravnotežnih enačb, ki smo jih zapisali zgoraj (enačbi (1) in (2)). Enačbo (2) preuredimo tako, da lahko iz rezultirajoče sile $F$ izrazimo pospešek $z''$. Enačbo lineariziramo v okolici delovne točke in nad njo izvedemo Laplaceovo transformacijo. Izrazimo izhod ($\Delta z(s)$ - prirastek višine od ravnovesne točke) proti vhodu ($\Delta v(s)$ - prirastek hitrosti zraka od ravnovesne lege) ter tako dobimo prenosno funkcijo $Z(s)$. Ventilator modeliramo kot člen prvega reda, z ojačenjem $k_v$ in vzponskim časom $\tau_s$. Celotna izpeljava funkcije je podana v članku [3], zaradi nazornosti pa so tukaj podani samo bistveni poudarki.

Prenosna funkcija ventilatorja je tako podana kot:

$$V(s) = \frac{v(s)}{u(s)} = \frac{k_v}{\tau_s + 1}, \tag{3}$$

kjer je $k_v$ konstanta ventilatorja, podana v $\frac{m/s}{V}$ in $\tau_s$ vzponski čas sistema (čas, da ventilator doseže nazivno hitrost pretoka zraka).

Prenosno funkcijo objekta v vetrovniku po izpeljavi zapišemo kot:

$$Z(s) = \frac{\Delta z(s)}{\Delta v(s)} = \frac{1}{s} \cdot \frac{a}{s(s+a)}, \tag{4}$$

Parameter $a$, je podan kot:

$$a = \frac{2 \cdot g}{v_{eq}}, \tag{5}$$

$v_{eq}$ pa je potrebna hitrost zraka, da vzpostavimo ravnovesno lego objekta v vetrovniku (dosežemo *ekvilibrium*):

$$v_{eq} = \sqrt{\frac{g \cdot m}{\frac{1}{2} \cdot C_d \cdot \rho \cdot A}} \tag{6}$$

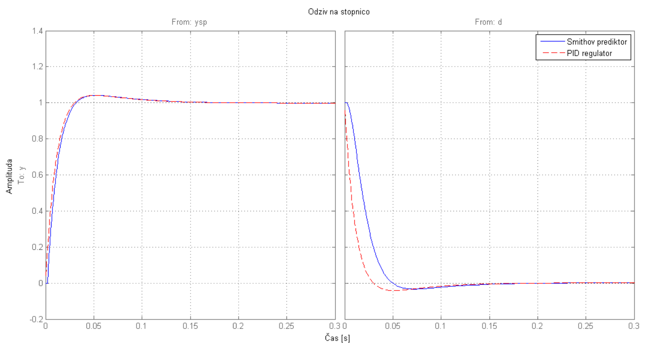
Skupna prenosna funkcija sistema $H_s$ je podana kot:

$$H(s) = V(s) \cdot Z(s) \tag{7}$$

in je tretjega reda. Integratorski člen v prenosni funkciji $Z(s)$ nakazuje na potencialno nestabilnost sistema. Objekt v vetrovniku ima namreč tendenco, da odleti iz cevi, v kolikor se ne poslužimo pravilnega pristopa pri načrtovanju regulatorja. Prav tako zaradi integralskega člena v prenosni funkciji le-tega ne potrebujemo v regulatorju, zato je teoretično dovolj le regulacija s PD regulatorjem (v okolici delovne točke). Kljub temu, se v praksi doda zelo mala vrednost integralnega (I) člena z namenom izboljšanja odziva.

Simulacije v MATLAB-u so bile narejene še preden so bili znani vsi parametri sistema ($C_d$, hitrost zraka, hitrost zajema podatkov,...), da je bilo okvirno preverjeno delovanje sistema. Tudi če bi uspelo natančno izmeriti vse potrebne parametre, se modelirani sistem zaradi raznih motenj iz okolice, ki jih je prezahtevno modelirati, ne bi odzval enako kot realni sistem. Treba je poudariti tudi to, da naj bi izračunano delovanje regulatorja zaradi linearizacije, veljalo le v območju okoli delovne točke. Za nastavitev parametrov regulatorja je bilo uporabljeno MATLAB-ovo orodje *pidtool*. Regulacija oziroma zaprto zančno vodenje vsebuje povratno informacijo izhoda s pomočjo senzorjev in povratne zanke sistema.

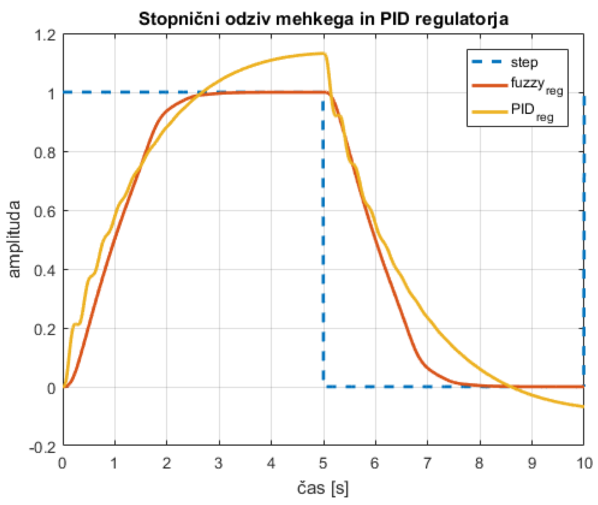
V osnovi je bila ideja vodenje na daljavo, kar pomeni, da se algoritem regulacije ne izvaja na istem krmilniku, ki krmili ventilator. Posledično so podaljšani časi do prihoda podatka v regulator. V kolikor se sistem obnaša zelo dinamično (časovne konstante sistema so majhne), lahko pride do težav pri regulaciji. Zaradi je PID regulatorju bil dodan Smithov prediktor, ki glede na matematični model regulirane proge predvidi obnašanje regulatorja med časom, ko ni na voljo novega podatka. V vseh realnih sistemih prihaja v povratni zanki do zakasnitev, ki jih je potrebno za uspešno regulacijo upoštevati. Smithov prediktor uporablja model predikcije zakasnitve izhoda procesa, primerja razliko med predvidenim in željenim izhodom ter določi potrebno korekcijo [7]. Prednost dodanega Smithovega prediktorja v primerjavi s samim PID regulatorjem je hitrejši odziv in manjši prenihaj pri odzivu na vrednost nič, kot prikazuje slika 3.



Slika 3: Stopnični odziv PID regulatorja brez in z uporabo Smithovega prediktorja.

Pri dejanskem nastavljanju parametrov regulatorja simulacije niso bile v veliko pomoč, kar je tudi eden izmed glavnih razlogov, da je bilo uporabljeno vodenje z mehko logiko. Z enakimi pripadnostnimi funkcijami in enakimi pravili, ki so bila uporabljena na realnem sistemu, je bil simuliran še odziv na stopnico z regulatorjem na osnovi mehke logike. Rezultati so napram pridobljenim s PID regulatorjem mnogo boljši. Vzponski čas je padel za okoli 15%, mehki regulator pa je prav tako izničil prenihaj, ki se pojavlja pri PID regulatorju, kar je vidno na sliki 4.

Pri načrtovanju mehkega regulatorja se tako nismo ukvarjali z matematičnim modelom in konstantami sistema, ampak so bile pripadnostne funkcije postavljene glede na dejanski odziv sistema, hitrost $v_{eq}$ pa določena empirično.



Slika 4: Stopnični odziv mehkega in PID regulatorja.

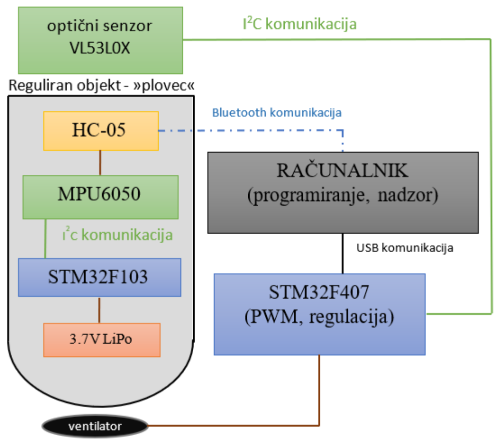
## 4. IZVEDBA REALNEGA SISTEMA

V nadaljevanju bo predstavljena priprava strojne in programske opreme za preizkuševališče in reguliran objekt, tj. plovec.

### 4.1 Priprava strojne opreme

Teoretično delovanje mehkega regulatorja je bilo potrebno preveriti tudi v praksi, zato smo zbrali oz. izdelali vse potrebne komponente za regulacijo. Sistem sestavlja:

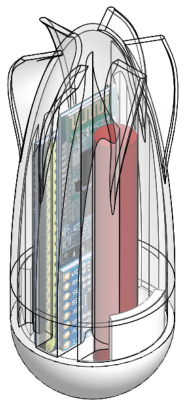
- plovec, ki predstavlja reguliran objekt (vsebuje MPU-6050, mikrokrmilnik STM32F103, Bluetooth modul HC-05, 3.7 V LiPo baterijo)
- tunel oz. regulacijska proga (cev iz pleksi stekla, usmernik, ventilator Dr. Mad Thrust KV3300)
- zunanje vezje za regulacijo (STM32F407 Discovery Board, optični senzor VL53L0X, krmilnik za ventilator ESC 20A)



Slika 5: Shematski prikaz regulacijske proge.

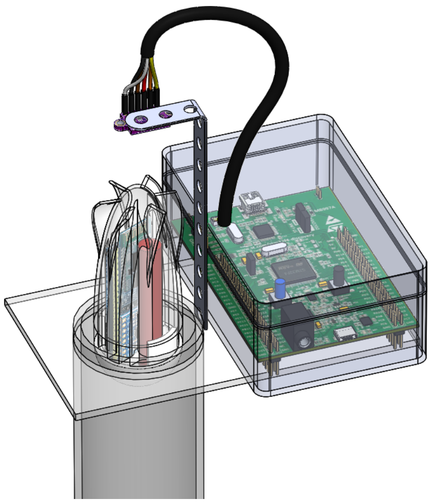
Prvotna ideja regulacije plovca v cevi je zajemala regulacijo preko kotnega zasuka oz. pospeška. Plovec, tj. telo z aerodinamično obliko, naj bi zato vseboval vse komponente potrebne za zajem, obdelavo in posredovanje podatkov gostujočemu sistemu. V ta namen je bila v plovec vgrajena mikrokrmilniška plošča STM32F103C8T6 (t.i. *Blue Pill*), kombinirani senzor MPU-6050, baterijsko napajanje ter vmesnik Bluetooth HC-05. Senzor MPU-6050 vsebuje integrirani MEMS (Mikro Elektro Mehanski Sistem) pospeškometer in žiroskop. Ta sočasno zajema podatke v X, Y in Z osi, ter je preko komunikacije $I^2C$ povezan s krmilnikom. Prikaz, nadzor in regulacija potekajo na računalniku, kamor se obdelani podatki pošiljajo preko Bluetooth komunikacije. Zaradi potrebe po napajanju plovec vsebuje 3.7 V LiPo baterijo s kapaciteto 270 mAh, ki naj bi zagotavljala energijo celotnemu sistemu do 4 ure.

Plovec je zaradi potrebe po čim bolj stabilnem gibanju in vrtenju okoli svoje osi, izdelan v obliki kaplje z dodanimi lopaticami. Ohišje plovca je bilo načrtovano v programu SolidWorks in natisnjeno s 3D tiskalnikom. Potrebnih je bilo nekaj iteracij, da so bile zagotovljene dimenzijske tolerance in usklajene velikosti delov. V programu SolidWorks je bila prav tako opravljena simulacija pretoka zraka po cevi in izračun težišča plovca z vsemi komponentami (slika 6).



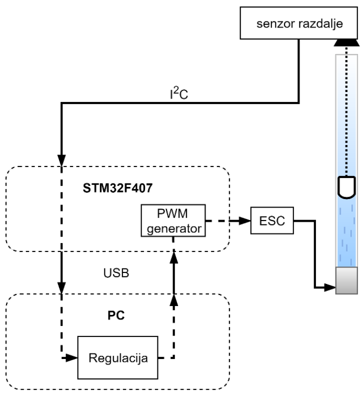
Slika 6: 3D model plovca z vgrajenimi komponentami.

Regulacijsko progo predstavlja votla cev, skozi katero ventilator piha zrak (slika 18). Za ventilatorjem je nameščen usmernik, ki skrbi za enakomeren laminarni pretok zraka, s tem pa je preprečeno nastajanje vrtincev, ki neugodno vplivajo na gibanje in regulacijo plovca. Za ventilator je bil izbran Dr. Mad Thrust KV3300, ki ga poganja brezkrtačni elektromotor z nazivno močjo 400 W in ga krmili ESC z maksimalnim tokom 20 A. Regulacija hitrosti se izvaja preko pulzno širinske modulacije, pri čemer mora biti frekvenca preklapljanja 50 Hz. Zaradi težav s pridobivanjem koristnih informacij za regulacijo s senzorja MPU-6050, je bil sistemu dodan optični senzor za merjenje razdalje. Širina plastične cevi ni omogočala povečanja plovca, ki je že bil na meji svoje velikosti, zato je bil senzor nameščen izven plovca na nosilec, pritrjen na vrhu cevi. Izvajanju regulacije je tako namenjeno zunanje vezje, ki ga sestavljata krmilnik STM32F407 Discovery Board in senzor za merjenje razdalje. Krmilnik je namenjen generiranju pulzno širinske modulacije za krmiljenje ventilatorja in posledično pretoka zraka skozi cev, kar za regulacijo predstavlja vhodno spremenljivko. Sprva je bil kot merilnik razdalje uporabljen infrardeči senzor Sharp GP2Y0A21YK0F (10 – 80 cm), ki je bil zaradi prevelikih nihanj v izmerjeni razdalji, zamenjan z optičnim senzorjem STM VL53L0X. Slednji ima popolnoma linearno karakteristiko ter višji domet (do 2 m), vendar komunicira preko protokola $I^2C$. Regulacija se izvaja na računalniku, ki je s krmilnikom povezan preko USB komunikacije.


Slika 7: 3D model zgornjega dela preizkuševališča.

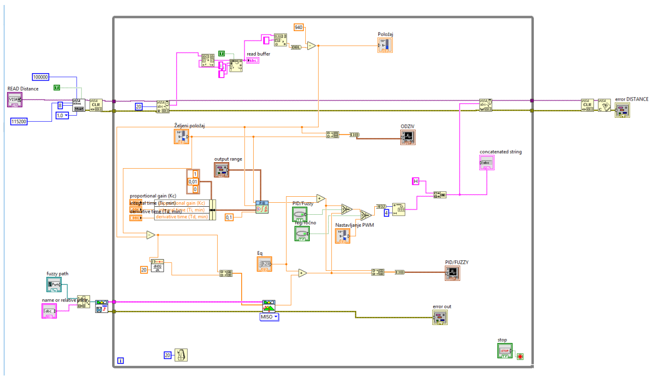
## 4.2 Priprava programske opreme

Za namene projektne naloge so bili napisani programi, ki so se izvajali na uporabljenih krmilnikih, kot tudi program, ki se je izvajal na osebnem računalniku in je skrbel za regulacijo. Za mikrokrmilnik z oznako STM32F103C8T6, ki se je nahajal v plovcu, ter za mikrokrmilnik serije STM32F407VG Discovery Board, je bil uporabljen programski jezik C. Pri programiranju je bil v pomoč namenski, brezplačni program CubeMX, ki omogoča lažjo konfiguracijo vseh enot mikrokrmilnika, pripravi vse potrebne knjižnice in ustvari grobo zasnovo vseh programov [16]. Pisanje programa za oba mikrokrmilnika pa je bilo opravljeno v integriranem razvojnem okolju (IDE) $\mu$Vision programskega paketa Keil. Oba omenjena programa sta namenjena delu in programiranju mikrokrmilnikov z Arm Cortex procesorji [2]. Pri programiranju krmilnikov so bile uporabljene tudi posebne visokonivojske knjižnice, t.i. knjižnice HAL (angl. *Hardware Abstraction Layer*), ki so posebej namenjene programiranju krmilnikov serije STM32. HAL knjižnice olajšajo in pospešijo programiranje, saj omogočajo delo z lažje razumljivimi ukazi [15]. Za mikrokrmilnik v plovcu (STM32F103C8T6) je bil napisan program, ki je omogočal branje pospeškov in kotnih hitrosti iz pospeškometra. Za komunikacijo med senzorjem in krmilnikom je skrbel protokol $I^2C$. Komunikacija je bila implementirana s pomočjo že napisanih prosto dostopnih knjižnic. Vsi podatki so bili ustrezno obdelani in s pomočjo Bluetooth modula HC-05 poslani na računalnik. Podatki so bili sprva prikazani s pomočjo programa LabVIEW, kasneje pa je bilo načrtovano te podatke uporabiti v Kalmanovem filtru pri izračunu višine plovca v vetrovniku. Ta naloga se je izkazalo za zelo zahtevno, saj se za pravilno delovanje Kalmanovega filtra priporoča zelo natančen model sistema, poleg tega so podatki iz senzorja MPU-6050 vsebovali preveč motenj in šuma, da bi se iz njih dalo razbrati uporabne meritve, kar bi povzročalo težave pri obdelavi podatkov in regulaciji. Kotna hitrost se je spreminjala naključno zaradi trkov v stene plastične cevi, zato ni bilo mogoče pridobiti uporabnih podatkov, pospeškometer pa vrača zelo šumne meritve, ki so same po sebi neprimerne za dvojno integracijo. Pri mikrokrmilniku (STM32F407VG), ki je skrbel za branje podatkov iz senzorja časa preleta (VL53L0X), je bila prav tako uporabljena komunikacija $I^2C$. Tudi ti podatki so bili poslani na računalnik, tokrat preko USB povezave. Na računalniku so bili podatki nato uporabljeni v regulaciji. Rezultat regulacije je bil podatek za hitrost vrtenja ventilatorja, ki je bil nato poslan nazaj na isti mikrokrmilnik. Na mikrokrmilniku je bil napisan tudi program za generiranje pulzno-širinskega signala, ki je bil namenjen krmilniku hitrosti vrtenja (ESC) motorja, glej sliko 8.



Slika 8: Shematski prikaz regulacijske proge.

Kot orodje za prejemanje, pošiljanje in obdelavo podatkov na računalniku je bil uporabljen program LabVIEW. Za prejemanje in pošiljanje podatkov je bilo v njem uporabljeno orodje VISA, ki omogoča branje in pisanje podatkov iz/na računalniški komunikacijski vmesnik. To orodje je bilo uporabljeno pri prejemanju podatkov iz Bluetooth povezave, kot tudi pri prejemanju in pošiljanju podatkov preko USB povezave s krmilnikom STM32F407. Sprva je bil program za regulacijo sestavljen s PID regulatorjem, ki ga je enostavno sestaviti z ustreznim blokom v programu LabVIEW, nato pa je bil zaradi težav implementacije ter pravilnega delovanja na vetrovniku ustvarjen mehki regulator tipa Mamdani.

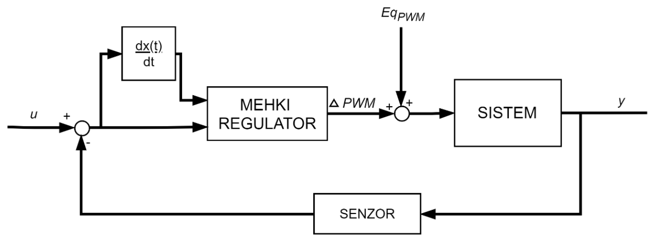


Slika 9: Blok diagram regulacijske zanke v LabVIEW-u.

V programu LabVIEW, glej sliko 9, je bil takšen regulator ustvarjen s pomočjo orodja imenovanega *Fuzzy System*

*Designer*. S tem orodjem se ustvari mehki regulator, kateremu določimo število in oblike pripadnostnih funkcij ter napišemo bazo pravil. Kot je razvidno iz slike 10, vhodni množici v mehki regulator predstavljata pogrešek med dejansko in željeno višino plovca v vetrovniku, ter njegov odvod.

Mehki regulator spreminja vrednost pulzno-širinskega signala v malem območju okoli vrednosti, pri kateri plovec lebdi v vetrovniku. Hitrost $v_{eq}$ potrebna za levitacijo pa je odvisna od mase plovca in zmožnosti premikanja po cevi – če se plovec zatika ob cev, je potrebna višja hitrost in obratno.

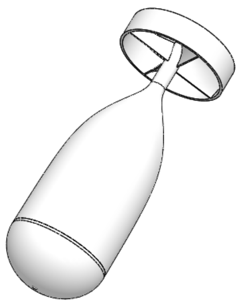


Slika 10: Regulacijska proga z vključenim mehkih regulatorjem. Veličina $y$ predstavlja višino, na kateri se nahaja plovec.

## 5. SESTAVLJANJE KOMPONENT

Sestava komponent vetrovnika je bila sorazmerno preprosta, vendar je bilo kljub temu med gradnjo sistema potrebno rešiti nekaj težav. Optični senzor VL53L0X komunicira s krmilnikom STM32F407 preko $I^2C$ komunikacije, zato je bila omejena dolžina kabla, ki povezuje senzor s krmilnikom. Krmilnik je bilo zato potrebno namestiti v neposredno bližino senzorja. V ta namen je bil izdelan preprost nosilec, ki je bil z vročim lepilom pritrjen na cev vetrovnika, kot prikazuje slika 17. Na spodnji strani je bila cev skupaj z usmernikom zraka vstavljena v osnovo, natisnjeno s 3D tiskalnikom. Iz spodnje strani je bil vanjo nameščen še ventilator ter v škatlo ob osnovi krmilnik motorja, ki je z dvema žicama povezan na mikrokrmilnik (slika 16). Prvi testi so pokazali, da se v sistemu pojavi precej vibracij, ki neugodno vplivajo na premikanje plovca po cevi. Oblika plovca, predvsem širina, je omejena s komponentami, ki se nahajajo v njem, poleg tega pa še z notranjim premerom cevi. Preširok plovec se zaradi preslabega obtoka zraka ni vrtel in se je zatikal ob cev. Ožji plovec je bil s stališča vrtenja ob zagonu boljša izbira, vendar zaradi notranjih komponent in načina izdelave, njegovega težišča ni bilo lahko spraviti na os vrtenja, zato je kmalu po začetku vrtenja postal nestabilen. Plovec je bil še dodatno obtežen, saj so se na ta način povečale časovne konstante sistema in spustile zahtevo po visokem vzorčevalnem času. Pri končni izvedbi so združene dobre lastnosti obeh prej omenjenih izvedb. Na začetku je plovec ožji in zagotavlja zadosten obtok zraka, zadnji del plovca pa je širši. Opletanje plovca je rešeno s tem, da je razlika med širino plastične cevi in zadnjega dela plovca le 1 mm, istočasno pa je s krilci omogočeno njegovo vrtenje.

Čeprav je bila prvotna ideja o regulaciji preko podatkov žiroskopa in pospeškometra opuščena, pa je bilo vrtenje plovca pomembno, saj se je plovec ob konstantnem vrtenju lažje premikal ter lažje dosegel želeno vrednost višine. Vibracije,



Slika 11: Plovec aerodinamične oblike.

ki jih povzroča ventilator, so odpravljene tako, da je plastična osnova privita na debelejši in težji kos podlage. Za pravilno delovanje sistema, je zelo pomemben tudi nemoten, predvsem pa zadosten dotok zraka, ki služi tudi hlajenju krmilnika ESC. Ta se lahko zaradi visokih nazivnih tokov pri krmiljenju ventilatorja temu primerno segreje. Napajanje sistema je možno izvesti na več načinov, z uporabo laboratorijskega napajalnika (0-50V, 20 A), računalniškega napajalnika ali s pomočjo LiPo baterije 3S, ki je pogosto uporabljena v modelarstvu. Na ta način je zagotovljena tudi prenosljivost preizkuševališča, ki je primerno za demonstracijsko rabo v pedagoškem procesu.

## 6. REZULTATI

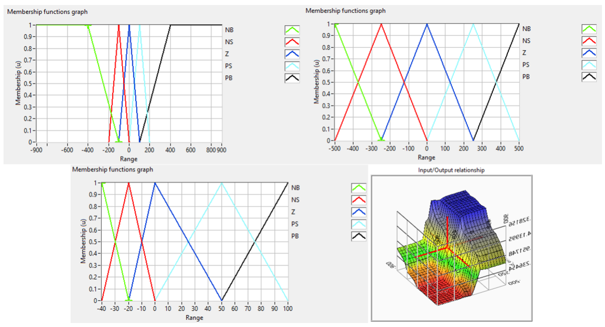
V končnem izdelku nam PID regulatorja ni uspelo implementirati predvsem zaradi:

- predolgega časa zajema položaja plovca. Čas potreben za pridobitev novega podatka o višini je trajal preko 100 ms, kar je bilo za regulacijo sistema občutno preveč. Dodatno je potrebno upoštevati še čas do nove nastavitve hitrosti, ki ga zaradi USB komunikacije v obe smeri, ter obdelave algoritma ocenjujemo na vsaj 10 ms. Težava s posredovanjem novega podatka sicer ni bila v samem senzorju, saj le-ta zmore zagotoviti novo vrednost vsakih 20 ms v režimu hitrega zajema podatkov, temveč v izvajanju prekinitvene rutine. Najnižji, še mogoč čas proženja prekinitev preko časovnika je znašal 100 ms. Natančnega razloga, zaradi katerega nismo mogli doseči hitrejšega izvajanja prekinitev, ne poznamo, predvidevamo pa, da je bil procesor v tem času preobremenjen.

- nezmožnosti PID regulatorja, da mu nastavimo karakteristiko, saj se v celotnem področju regulator obnaša enako. PID regulator, ki je bil sposoben držati plovec v bližini referenčne točke smo sicer izdelali, vendar z rezultatom nismo bili zadovoljni.

V obeh omenjenih pomanjkljivostih PID regulatorja pa ima uporaba mehkega regulatorja prednost. Uporaben je pri počasnejših časih vzorčenja, njegova največja prednost pa je ta, da omogoča bolj transparentno nastavljanje njegovih parametrov [13]. Strokovnjak, ki načrtuje regulator ima tako izboljšano predstavo o relaciji med vhodnimi in izhodnimi podatki, kar pa je za PID težko trditi. Glavni parametri, od katerih je odvisno delovanje mehkega regulatorja so število in položaj pripadnostnih funkcij ter pravila, ki povežejo vhodne in izhodne pripadnostne funkcije.

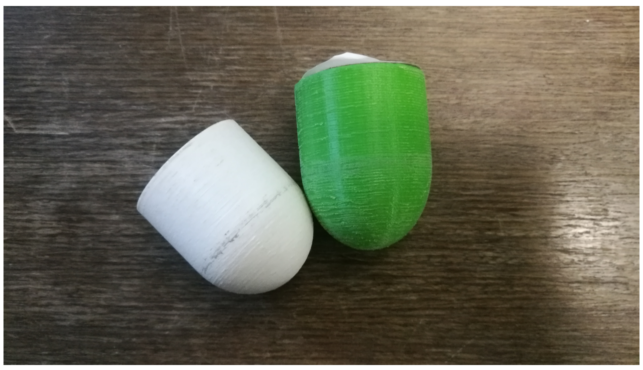
Končni regulator je imel 5 pripadnostnih funkcij za vsako vhodno in izhodno množico, med sabo pa jih je povezovalo 25 pravil. Vse pripadnostne funkcije so trikotne nezvezne oblike, kakor so prikazane na sliki 12.



Slika 12: Pripadnostne funkcije mehkega regulatorja in karakteristika - nelinearna ravnina.

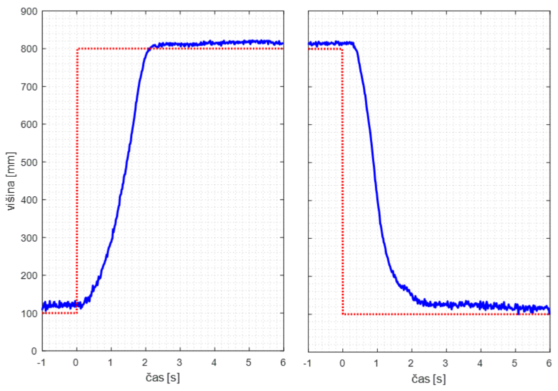
Z uporabo mehkega regulatorja, smo se prav tako izognili uporabi Smithovega prediktorja, ki sicer v nekoliko spremenjeni obliki obstaja tudi za mehki regulator [14].

Težave z dolgimi časi zajema ($T_s$) smo rešili tudi na način, da smo v končni regulaciji uporabili težji plovec, zaradi česar je bil odziv sistema počasnejši in posledično je mehki regulator lažje in bolje reguliral sistem. Ugotovili smo tudi, da je mehki regulator manj občutljiv na spremembe reguliranega objekta – plovca, je pa potrebno poudariti, da telesa z nizko maso (žogica za namizni tenis) na ta način ni mogoče regulirati, ker so časovne konstante tega objekta precej nižje od plovca, uporabljenega v naši nalogi.



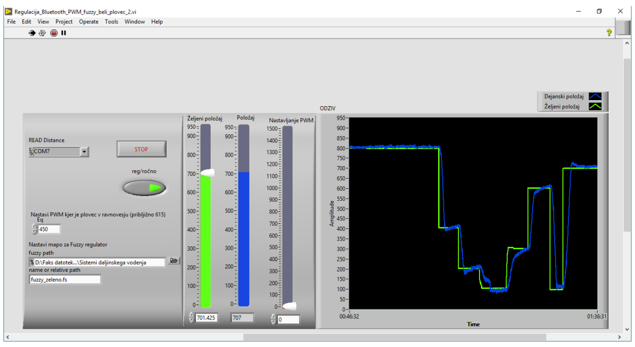
Slika 13: Uporabljeni plovci za testiranje preizkuševališča.

Pred začetkom regulacije je bilo potrebno poiskati le vrednost hitrosti zraka v cevi, pri kateri plovec v vetrovniku lebdi ($v_{eq}$), kar naredimo z enostavnim povečevanjem hitrosti vrtenja motorja z drsnikom na nadzorni plošči v programu LabVIEW (slika 15). Takšen način zagona vetrovnika je bil uporabljen zlasti zaradi uporabljenih različnih plovcev (slika 13), pri katerih so se ravnovesne vrednosti vrtenja motorja med seboj. Ko je ta vrednost zabeležena in vpisana, lahko regulator ustrezno regulira višino plovcev različnih oblik in tež.



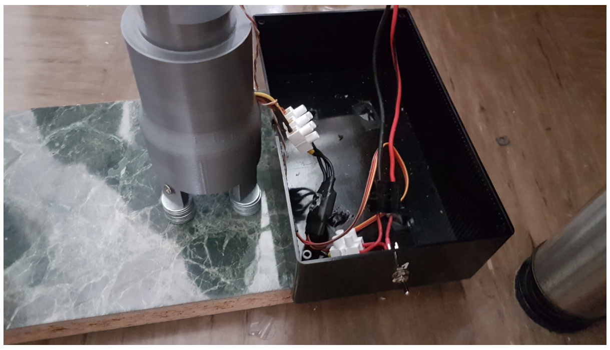
Slika 14: Stopnični odziv mehkega regulatorja na realnem sistemu. Uporabljen je bil zeleni plovec z maso 25 g.

Kot je razvidno iz stopničnega odziva na sliki 14, se plovec med skrajnima višinama premakne v približno dveh sekundah. Ob željenem manjšem nastavitvenem času pride do večjih prenihajev, kar lahko pomeni, da plovec odleti iz vetrovnika in zadane senzor, ki je nameščen nad vetrovnikom ter ga poškoduje. Podobno se zgodi pri spuščanju plovca, kjer lahko plovec pri večjih prenihajih poškoduje usmernik zračnega pretoka. Iz teh razlogov je bil narejen kompromis med prenihajem in nastavitvenim časom pri odzivu mehkega regulatorja.


Regulacija_Bluetooth_PWM_fuzzy_beli_plovec_2.vi
File Edit View Project Operate Tools Window Help
READ Distance
COM7
STOP
reg/ročno
Nastavi PWM kjer je plovec v ravnovesju (približno 615)
Eq
450
Nastavi mapo za Fuzzy regulator
fuzzy path
D:\Faks datoteke\...\Sistemi daljinskega vodenja
name or relative path
fuzzy_zeleno.fs
Željeni položaj
Položaj
Nastavljanje PWM
701.425
707
0
ODZIV
Dejanski položaj
Željeni položaj
Amplitude
Time
00:46:32
01:36:31

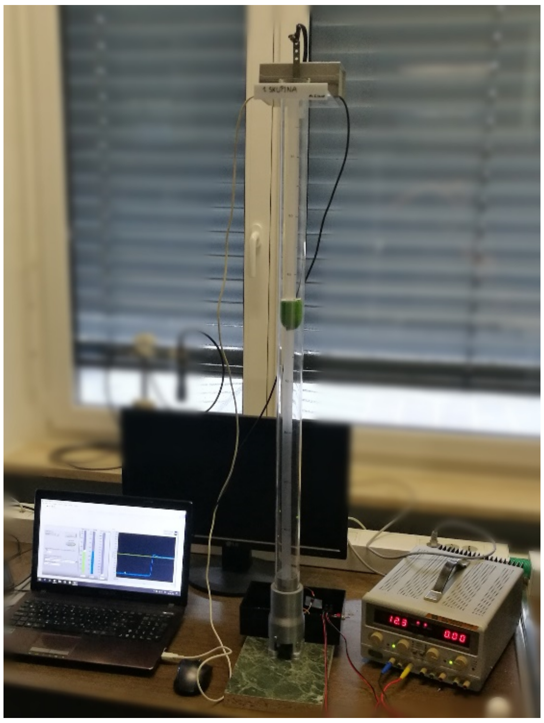
Slika 15: LabVIEW kontrolna plošča.

Sistem v taki obliki je tudi stabilen in odporen na motnje. Preizkuševališče (slike 16, 17, 18) lahko dvignemo ali mu spremenimo naklon, pri čemer regulator še vedno odlično opravlja svoje delo.



Slika 16: Spodnji del preizkuševališča.

Slika 17: Zgornji del preizkuševališča.



Slika 18: Celotni sistem.

## 7. SKLEP

Članek zajema izvedbo regulirane levitacije plovca v zračnem tunelu, pri čemer gre za sistem daljinskega vodenja. Regulacija se izvaja na oddaljenem sistemu, v našem primeru na osebnem računalniku. Višino plovca v cevi nam je uspelo regulirati ne glede na visoke mrtve čase, kar nam sicer s PID regulatorjem ni povsem uspelo. Aplikacija na osebnem računalniku (LabVIEW) nam omogoča, da nastavimo želeno višino, sistem pa se samodejno odzove. Poleg tega lahko v njej istočasno spremljamo realne vrednosti pospeškov in kotnih hitrosti plovca, v kolikor uporabimo omenjeno izvedbo z vgrajenim krmilnikom. Tekom dela smo naleteli na več težav, največ težav je predstavljala strojna oprema in implementacija $I^2C$ komunikacije med optičnim senzorjem (VL53L0X) in mikrokrmilnikom (STM32F407). Dodatne težave, ki jih nismo povsem odpravili so vibracije cevi, ki vplivajo na vrtenje plovca in ga občasno popolnoma zaustavijo, saj se ta namesto da bi se vrtel, prične odbijati od stene. Problem je tudi trenje med plovcem in steno vetrovnika, ki prav tako zavira vrtenje, ter pa opletanje plovca, ker ta ni povsem centriran. Vsekakor pa so nekatere izmed teh težav privedle do tega, da smo se odločili za uporabo mehke logike.

Težave, ki so ostale, predstavljajo možnosti nadaljevanja dela na projektu in dodatnih izboljšav sistema. Boljše rezultate obeta uporaba širše cevi, tako bi bili nekoliko manj omejeni z oblikami in dimenzijami plovcev. Regulator, ki trenutno nadzira sistem je bil v primerjavi s PID, izdelan v zelo kratkem času, vendar je že v začetni fazi testiranja dajal spodbudne rezultate, vendar pa smo zaradi pomanjkanja časa morali izpustiti napredno optimizacijo procesa. Dokazali smo, da lahko z mehko logiko vodimo kompleksnejši sistem, ki ima dolge čase zajema in bi ga sicer s PID regulatorjem ob enakih pogojih zelo težko vodili brez dodanega Smithovega prediktorja ali nasploh. Za nastavitev regulatorja smo porabili veliko manj časa, kot smo ga sicer za poskus implementacije PID regulatorja, ki nam ni dajal pravih rezultatov. Ob predpostavki, da se mase plovca, na katerem je bil mehki regulator nastavljen ne spremenijo bistveno, regulator ohranja solidne odzive, bolj kot sama masa pa na rezultate bolj vpliva hrapavost in stabilnost plovca v cevi.

## ZAHVALA

Iskreno se zahvaljujemo rednemu profesorju dr. Dušanu Gleichu ter asistentu mag. Blažu Pongracu s Fakultete za elektrotehniko, računalništvo in informatiko v Mariboru za pomoč pri izvedbi simulacij in načrtovanju programske opreme.

## REFERENCES

[1] C. G. Amza and D. T. Cicic. Industrial image processing using fuzzy-logic. *Procedia Engineering*, 100:492–498, 2015.

[2] armKEIL. MDK Microcontroller Development Kit. Dostopno na: `http://www2.keil.com/mdk5/.Dostopano:16.9.2018.`

[3] J. Chacon, J. Saenz, L. d. l. Torre, J. M. Diaz, and F. Esquembre. Design of a Low-Cost Air Levitation System for Teaching Control Engineering. *Sensors*, 17(10):2321, 2017.

[4] D. Fister, R. Šafaric, and I. Fister. Nastavljanje parametrov regulatorja z optimizacijskimi algoritmi. In *StuCoSReC: Proceedings of the 2015 2nd Student Computer Science Research Conference*, pages 13–17,

2015.

[5] A. M. Gavgani, A. Sorniotti, J. Doherty, and C. Cavallino. Optimal gearshift control for a novel hybrid electric drivetrain. *Mechanism and Machine Theory*, 105:352 – 368, 2016.

[6] I. Iancu. A mamdani type fuzzy logic controller. In *Fuzzy Logic-Controls, Concepts, Theories and Applications*. InTech, 2012.

[7] A. Ingimundarson and T. Hägglund. Robust tuning procedures of dead-time compensating controllers. *Control Engineering Practice*, 9(11):1195–1208, 2001.

[8] K. Ngorex. Slideshare. Dostopno na: `https://www.slideshare.net/poerslide/lecture7-ml-machines-that-can-learn.Dostopano:7.8.2018.`

[9] O. Kraa, A. Aboubou, M. Becherif, M. Ayad, R. Saadi, M. Bahri, and H. Ghodbane. Fuzzy logic maximum structure and state feedback control strategies of the electrical car. *Energy Procedia*, 50:178 – 185, 2014. Technologies and Materials for Renewable Energy, Environment and Sustainability (TMREES14 – EUMISD).

[10] R. Li, Y. Guo, S. K. Nguang, and Y. Chen. Takagi-Sugeno fuzzy model identification for turbofan aero-engines with guaranteed stability. *Chinese Journal of Aeronautics*, 31(6):1206–1214, June 2018.

[11] K. Mehran. Takagi-sugeno fuzzy modeling for process control. *Industrial Automation, Robotics and Artificial Intelligence (EEE8005)*, 262, 2008.

[12] N. Rai and B. Rai. Control of fuzzy logic based PV-battery hybrid system for stand-alone DC applications. *Journal of Electrical Systems and Information Technology*, 2018.

[13] R. Šafarič and A. Rojko. *Inteligentne regulacijske tehnike v mehatroniki*. Fakulteta za elektrotehniko, računalništvo in informatiko, 2007.

[14] A. Sakr, A. M. El-Nagar, M. El-Bardini, and M. Sharaf. Fuzzy smith predictor for networked control systems. In *Computer Engineering & Systems (ICCES), 2016 11th International Conference on*, pages 437–443. IEEE, 2016.

[15] ST. Description of STM32F4 HAL and LL drivers. Dostopno na: `https://www.st.com/resource/en/user_manual/dm00105879.pdf.Dostopano:16.9.2018.`

[16] ST. STM32CubeMX. Dostopno na: `https://www.st.com/en/development-tools/stm32cubemx.html\#sw-tools-scroll.Dostopano:16.9.2018.`

[17] Tutorialspoint. Fuzzy Logic Applications. Dostopno na: `https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_applications.htm.Dostopano:7.8.2018.`

# Razpoznavanje literature v dokumentih in določanje njihovih sklicev v besedilu

**Matej MORAVEC**

*Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46,
2000 Maribor, Slovenija
matejmoravec19@gmail.com*

**David LETONJA**

*Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46,
2000 Maribor, Slovenija
david.letonja@gmail.com*

**Jan TOVORNIK**

*Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46,
2000 Maribor, Slovenija
jan.tovornik@gmail.com*

**Borko BOŠKOVIČ**

*Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46,
2000 Maribor, Slovenija
borko.boskovic@um.si*

## POVZETEK

Pojav plagiatorstva je vedno bolj pogost, saj v današnjem času obstaja zelo veliko elektronskih virov, preko katerih lahko dostopamo do želene vsebine. V ta namen je bilo razvitih veliko različnih pristopov za detekcijo plagiatov. V članku predstavimo pristop, kjer v dokumentih razpoznamo navedeno literaturo in znotraj dokumenta poiščemo, kje se nahajajo sklici v besedilu. Uspešnost našega pristopa smo preizkusili s pomočjo različnih metrik. Dosegli smo 67 % natančnost in 80 % mero F1 za razpoznavo virov.

## Kjučne besede

razpoznavanje literature, analiza citatov, plagiatorstvo, podobnost datotek

## 1. UVOD

Plagiat je delo, ki je prepisano, povzeto od drugod in objavljeno, prikazano kot lastno [3]. Večina strokovnih dokumentov vsebuje navedeno literaturo. Če se literatura nahaja v dokumentu ali ne, je seveda odvisno od vrste dokumenta.

Razpoznavanja literature in kasneje iskanja njihovih sklicev znotraj besedila smo se lotili z idejo, da bi na tak način pripomogli k večji natančnosti programov, ki preverjajo, ali je določeni dokument plagiat ali ne. Definicija plagiata v SSKJ pravi, da je plagiat tisto, "kar je prepisano, prevzeto od drugod in objavljeno, prikazano kot lastno, navadno v književnosti" [3].

Pristop razpoznavanja literature in iskanja njihovih sklicev v dokumentih, predstavljen v tem članku, temelji na uporabi regularnih izrazov in posameznih korakih pregledovanja dokumentov, predstavljenih v naslednjih poglavjih. Glavna prednost našega pristopa je, da je neodvisen od jezika, v katerem je dokument napisan.

Članek je strukturiran tako, da v naslednjih poglavjih predstavimo sorodno delo, nato opišemo delovanje našega pristopa, opišemo rezultate eksperimenta in na koncu podamo kratek zaključek, kjer povzamemo naše ugotovitve skozi celotno delo.

## 2. SORODNA DELA

V članku [7] sta avtorja predstavila nov pristop k odkrivanju plagiatorstva, imenovan Analiza vrstnega reda citatov ali COA (Citation Order Analysis). Ta deluje na podlagi analize citatov in sklicevanja na vire oziroma literaturo ter vsebuje naslednje korake:

1. preoblikovanje dokumenta za procesiranje citatov in njihovih pojavitev v dokumentu,
2. ujemanje citatov z njihovimi navedbami v literaturi,
3. med dokumenti se preveri podobnost citatov. V osnovni različici sistema se upošteva samo vrstni red citatov, v naprednejši različici sistema pa se ocenjuje tudi razdalja med dvema citatoma. Če se dokument prevede v drugi jezik, se lahko vrstni red citatov znotraj stavkov ali odstavkov spremeni zaradi različnih struktur stavkov ali drugačnega načina pisanja.

Za ocenitev pristopa so opravili preizkuse na 800.000 prosto dostopnih znanstvenih publikacij, med katere so tudi skrili 20 posebej zasnovanih plagiariziranih dokumentov. Da je scenarij izgledal bolj realističen, so nekaj citatov izbrisali, dodali par novih, nekaterim spremenili stil, nekatere pa med seboj zamenjali. Pristop je od 20 testnih dokumentov uspešno odkril 19 in na stotine drugih dokumentov, ki so vsebo-

vali vsaj nekaj plagiariziranih odsekov. Več kot ima dokument navedenih citatov, težje je odkriti plagiarizem.
Preoblikovanje dokumentov iz formata pdf v xml je trajalo 2 sekundi za vsak dokument. V 96 % vseh zgoraj omenjenih dokumentov/publikacij je bilo ujemanje citatov z njihovimi navedbami v bibliografiji uspešno.
Največja prednost opisanega pristopa je neodvisnost od jezika, v katerem je dokument napisan, in neodvisnost do parafraziranja. Časovna zahtevnost je v primerjavi s pristopi, ki temeljijo na podobnosti besedil, zanemarljiva. Največja slabost pristopa pa je odvisnost od pravilnega citiranja.
Slabost tega pristopa je ob enem prednost pristopov, ki temeljijo na podobnosti besedil, zato se za doseganje najboljših rezultatov priporoča kombinirana uporaba obeh pristopov.

V članku [8] so še opisani rezultati, ki so jih avtorji pridobili ob avtomatski obdelavi ogromne količine znanstvenih člankov v skladu z natančno določenimi kriteriji ocenjevanja.
Populacijo znanstvenih člankov, ki so jo uporabljali, so uredili v skupine tako, da so zadostili kriterijem medsebojne povezanosti, tj. povezanost v tematiki. Dokumenti iste tematike kažejo na zelo visoko stopnjo korelacije med seboj. Avtorji dokažejo, da lahko z njihovim pristopom zelo dobro določimo tematiko dokumenta le s pomočjo razpoznane literature v dokumentih.

Delo Irene Marshakove [9] opisuje novo formalno metodo za klasifikacijo dokumentov. Ta metoda temelji na analizi referenc oziroma sklicev s pomočjo indeksiranja literature znanstvenih člankov. Vsakemu sklicu se dodeli indeks od začetka do konca dokumenta. Po dodelitvi indeksov dokumentom se le-ti indeksi in številka reference oziroma sklica medsebojno primerjajo. Primer, da imata dve ali več del veliko sklicev z enakimi indeksi, nakazuje, da je morebiti eno ali več del izmed teh plagiat.

Članek [4] opisuje, kako je potekalo prvo mednarodno tekmnovanje v odkrivanju plagiatorstva. Na tem tekmovanju se je izvajala delavnica PAN - Delavnica o odkrivanju plagiatorstva, avtorstva in zlorabe socialne programske opreme (zloraba osebnih podatkov preko družabnih omrežij).
To tekmovanje oziroma natečaj je bil razdeljen na več področij (13 različnih: različne težavnosti, več vrst nalog itd.) in to z enim razlogom, da bi preprečili kršitve, kot so plagiatorstvo in kraja osebnih podatkov. "Stranski efekt" tega tekmovanja je bil postaviti oceno, kako dobro detektiramo plagiatorstvo in varnost osebnih podatkov bodisi iz avtorskih del ali družabnih omrežij. Hkrati pa izbrati množico ljudi, ki bi razvijala različne algoritme za bolj varni jutri.

Članek [10] opisuje novo obliko združevanja dokumentov imenovano: so-citiranje. Je pristop, kjer ob navedbi vira navedemo skupaj več virov (dokumente, knjige, internetne vire, na katere se nato sklicujemo). Avtorji predpostavljajo uporabo so-citiranja zato, da zmanjšamo količino literature. Tako združujemo vire, ki temeljijo na isti tematiki, so iz iste založbe, imajo iste avtorje, ali pa preprosto so-citiramo zato, ker najdemo eno stvar iz dveh različnih virov.

## 3. PREDLAGANA METODA

Predlagana metoda vsebuje tri faze. V prvi fazi detektira poglavje, ki vsebuje navedeno literaturo. V drugi fazi znotraj poglavja razpozna posamezno literaturo in te oznake literature doda v seznam. V tretji fazi pregleda celoten dokument in poišče, na kateri strani in v katerem odstavku znotraj strani se nahajajo sklici na literaturo.

Preden smo pričeli z razpoznavanjem in gradnjo seznama literature, smo zaradi različno strukturiranih dokumentov naleteli na prvi izziv. Naša prva naloga je bila, da smo v dokumentu najprej našli, kje se nahaja poglavje z navedeno literaturo. To smo rešili tako, da smo zapisali tak regularni izraz, ki je razpoznal čim več različnih načinov zapisa poglavja, ki vsebuje literaturo.

Vsebino poglavja z navedeno literaturo smo ravno tako preiskali s pomočjo regularnih izrazov. Z namenom razpoznavanja čim večjega števila virov znotraj poglavja smo zapisali večje število regularnih izrazov. Rešitve smo se lotili tako, da smo razpoznavali vrstico po vrstico znotraj poglavja z navedeno literaturo. Ob posameznem najdenem viru smo zapisali njegovo oznako (številko), v za to namenjen seznam. Ta seznam smo zgradili z namenom za kasnejše iskanje sklicev na posamezen vir v besedilu.

Množica dokumentov, ki smo jo uporabili kot učno množico za naš pristop je vsebovala strokovna oziroma znanstvena dela z različnih fakultet. Vsi dokumenti so bili zapisani v slovenskem jeziku in so po večini bili drugače strukturirani. Zaradi različno strukturiranih dokumentov je bilo razpoznavanje sklicev toliko večji izziv.

Za prepoznavanje poglavja, ki je vsebovalo navedeno literaturo, smo uporabili regularni izraz, specifičen za element pagraf (<p>). Ta regularni izraz lahko razbijemo na več delov:

- V prvem delu, poiščemo vse odstavke <p>, ki imajo vsebino atributa *xml:id*, ob pb (page break) in p (odstavku) celo število.
- V drugem delu se prvemu regularnemu pravilu, pridružuje iskanje odstavkov <p> v poljubnem jeziku. V primeru, da bi želeli poiskati le določen jezik, bi morali vsebino atributa *xml:lang* specifično navesti.
- V zadnjem delu pa k prej navedenima praviloma dodamo pravilo za iskanje odstavkov <p> po specifičnih naslovih, pod katerimi se predstavlja literatura.

Regularni izraz poišče celoten odstavek, v katerem je podan naslov poglavja literature. Ker naslov poglavja najverjetneje vsebuje oštevilčenje, mora regularni izraz to tudi razpoznati. Upoštevati moramo tudi dejstvo, da obstaja več vrst oštevilčevanja, kar nam dodatno oteži razpoznavanje. Po zaporedni številki poglavja sledi naslov poglavja, pri katerem preverjamo naslednje besedne zveze (upoštevajo se velike in male črke):

- seznam virov,
- seznam literature,
- viri,

- literatura,
- seznam uporabljenih virov in
- reference.

Če vzorec ustreza določenemu izrazu, se najdeno poglavje tretira kot poglavje literature, ki vsebuje seznam literature oziroma virov.

Literatura je lahko podana na več načinov:

- oštevilčena z navadnimi oklepaji,
- oštevilčena z oglatimi oklepaji ali
- neoštevilčena.

Da smo razpoznali te tri načine, smo uporabili 2 različna regularna izraza, enega za oštevilčen način in enega za neoštevilčen način podajanja literature. Oba regularna izraza, za vsako podano literaturo, podata v seznam naslednje podatke:

- številka strani, na kateri je literatura podana,
- zaporedna številka v seznamu (če le-ta obstaja),
- prva beseda (kot priimek avtorja).

Ta seznam se v nadaljevanju uporabi za lažjo in bolj organizirano iskanje referenc.

S ciljem poiskati čim več sklicev v besedilu, smo zapisali različne regularne izraze. V našem pristopu smo zagotovili odkrivanje sklicev tam, kjer avtor dokumenta navaja sklice na način, da v oglate ali navadne oklepaje zapiše posamezno oznako vira ali pa navede več virov znotraj oklepajev. Naš pristop najde tudi sklice, kjer avtor v oklepajih navede prvega avtorja.

Za pravilno prepoznavanje poglavij, literature/virov in sklicev smo uporabili regularne izraze. Za vsak korak v opisanem postopku se je uporabil svoj regularni izraz.

Za prepoznavanje referenc oziroma sklicev na literaturo smo uporabili kar 4 različne regularne izraze, ki razpoznavajo naslednje načine sklicevanja:

- enoštevilčno v navadnih in oglatih oklepajih ((0), [0]),
- večštevilčno v navadnih in oglatih oklepajih ((1, 2, 3), [1, 2, 3]),
- avtor brez oklepajev (Vir: avtor ...),
- avtor z navadnimi in oglatimi oklepaji ((avtor ...), [avtor ...]).

Iskanje in prepoznavanje referenc se opravlja nad celotnim dokumentom in z največ regularnimi izrazi, zato je ta korak v večini primerov časovno najzahtevnejši.

## 4. REZULTATI

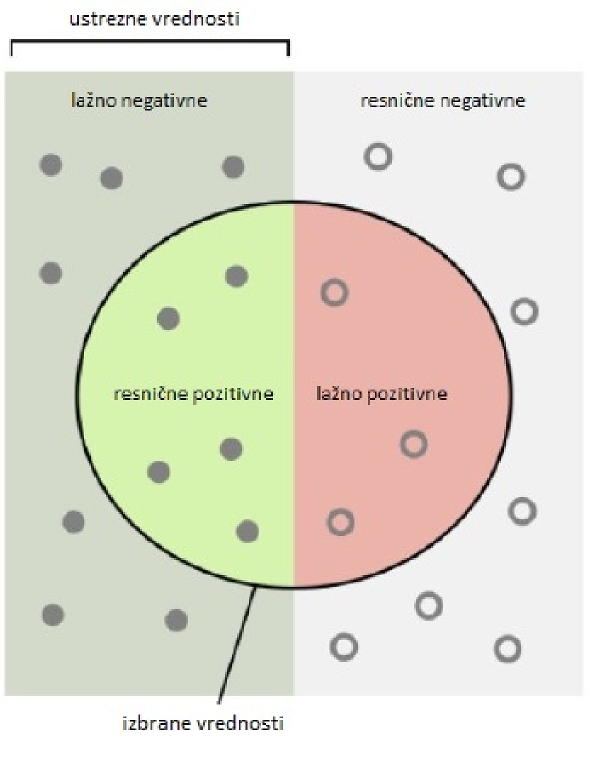
V tem poglavju bomo primerjali dobljene rezultate z ročno dobljenimi rezultati in prikazali uspešnost našega pristopa.

Kontingenčna tabela je tip tabele, zapisan v matrični obliki, ki prikazuje frekvenčno porazdelitev več spremenljivk. Veliko se uporablja pri inženirskih, znanstvenih raziskavah ipd. Tabela prikaže osnovno idejo medsebojne povezave med dvema spremenljivkama in lahko pomaga pri iskanju medsebojne interakcije [1].

**Tabela 1: Zapis kontingenčne tabele [5].**

| **podatki / klasifikacija** | **pravilen sklic** | **napačen sklic** |
|---|---|---|
| **pravilen sklic** | resnični pozitivni (tp) | lažno pozitivni (fp) |
| **napačen sklic** | lažno negativni (fn) | resnični negativni (tn) |

Na Sliki 1 so prikazane vrednosti in njihove oznake, ki jih lahko klasificiramo iz podatkov. V našem primeru *lažno negativne* vrednosti predstavljajo sklice, ki jih je program našel, a v dokumentu ne obstajajo. *Resnično pozitivne* vrednosti predstavljajo sklice, ki v dokumentu dejansko obstajajo in jih je program našel, ter *lažno pozitivne* vrednosti so tiste, ki v dokumentu obstajajo, a jih program ni našel.



**Slika 1: Prikaz vrednosti, ki jih lahko vrne klasifikator [2].**

.

Kontingenčna matrika je za ocenjevanje uspešnosti našega pristopa uporabna, saj lahko s pomočjo le-te izračunamo natančnost, preciznost, priklic in mero F1, kot prikazujejo naslednje enačbe:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad (1)$$

$$precision = \frac{tp}{tp + fp} \quad (2)$$

$$recall = \frac{tp}{tp + fn} \quad (3)$$

$$F1 = \frac{2PR}{P + R} \quad (4)$$

Za izračun uspešnosti smo uporabili 100 dokumentov. Od tega smo jih 31 izločili, saj so bili sklici navedeni tako, da niti človek ne bi znal razpoznati, na kateri vir se oseba v dokumentu sklicuje. Največji problem v dokumentih je torej predstavljalo naštevanje virov tako, da sta bila pod isto številko našteta dva ali več virov. Zato smo za izračun uporabili 69 dokumentov in na koncu izračunali točnost, natančnost, priklic in metriko F1. Za izračun uspešnosti smo uporabili mikro povprečenje, kjer smo združili dobljene rezultate klasifikatorja.
V Tabeli 2 so prikazani rezultati posameznih izračunov.

**Tabela 2: Metrike uspešnosti razpoznavanja literature.**

| **Metrika** | **Rezultat** |
|---|---|
| **Točnost (angl. accuracy)** | 0,67 |
| **Natančnost (angl. precision)** | 0,88 |
| **Priklic (angl. recall)** | 0,74 |
| **Metrika F1 (angl. F1 score)** | 0,80 |

Uspešnost našega programa, ki smo jo izračunali na podlagi mikro povprečenja, je 67 %.

### 4.1 Eksperiment za ugotavljanje plagiatorstva

Z namenom prikaza uporabnosti našega pristopa v smislu ugotavljanja plagiatorstva med dokumenti smo izvedli en manjši eksperiment. Pri tem smo uporabili dva dokumenta. En dokument je predstavljal že objavljeno delo, drug dokument pa smo s tem dokumentom v smislu navajanja literature in navajanja sklicev primerjali. Navedeni potek primerjave smo izvedli trikrat (z različnimi dokumenti).

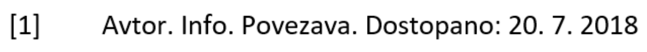
Pri prvi primerjavi dveh dokumentov, smo v objavljenem dokumentu z našo metodo našli 34 pravilnih sklicev in nič napačnih. V dokumentu za primerjavo z objavljenim pa smo našli 52 pravilnih sklicev in nič napačnih. Ugotovili smo, da je od teh 52 sklicev 34 takih, ki se pojavijo na isti strani, v istem odstavku kot v objavljenem dokumentu. Na podlagi teh rezultatov lahko z veliko verjetnostjo trdimo, da gre za plagiat, saj je zelo majhna verjetnost, da bi dva dokumenta imela na istih straneh, v istih odstavkih tako veliko sklicev na literaturo. V primerjavi drugih dveh dokumentov smo v objavljenem dokumentu našli 37 pravilnih sklicev, 2 sklica pa nismo našli. V dokumentu za primerjavo pa smo našli le 20 pravilnih sklicev, a 5 sklicev nismo našli. Med vsemi sklici ni bilo nobenega, ki bi se nahajal na isti strani, v istem odstavku. Na podlagi takih rezultatov z našo metodo nismo ugotovili oziroma ne moremo ugotoviti, če gre za plagiat ali ne. Na podlagi rezultatov tretje izvedbe primerjave dveh dokumentov z našo metodo prav tako ne moremo ugotoviti, če gre za plagiat ali ne. Primerjave sploh ne moremo narediti, saj naša metoda od 6 sklicev v objavljenem dokumentu ni našla nobenega. V dokumentu za primerjavo je sicer našla 20 sklicev, vendar jih v tem primeru ne moremo uporabiti za primerjavo.

Na podlagi izvedenega eksperimenta lahko vidimo, v katerih primerih je predlagana metoda uporabna. Če so avtorji dokumentov dosledni in "pravilno" navajajo sklice in literaturo, potem lahko z uporabo predlagane metode dokaj učinkovito ugotovimo, ali je dokument plagiat ali ne, oziroma lahko v primeru uporabe ostalih metod za preverjanje plagiatorstva s to metodo povečamo oziroma zmanjšamo verjetnost, da je delo plagiat.

## 5. UGOTOVITVE IN PROBLEMI

Skozi razvoj našega pristopa k odkrivanju plagiatorstva smo naleteli na kar nekaj težav. Glavna težava v našem delu je bil način navajanja in sklicevanja na vire. Kot je zabeleženo v navodilih za pisanje zaključnih del [6], vidimo, da je navedenih kar nekaj napotkov, kako pravilno navajati in se nato sklicevati na vire.

Navedbo virov in literature je potrebno zapisati kot seznam na koncu zaključnega dela. V ta seznam je potrebno zabeležiti vse vire, na katere smo se sklicevali v zaključnem delu. Vsakega izmed virov je potrebno oštevilčiti z arabskimi številkami znotraj oglatih oklepajih (Slika 2).


[1] Avtor. Info. Povezava. Dostopano: 20. 7. 2018

**Slika 2: Pravilna navedba vira.**

Seznam literature in virov je potrebno urediti po abecednem vrstnem redu avtorjev. V primeru, da ni navedenih avtorjev, uredimo po naslovu vira. Pomanjkljivost in napačno sklicevanje virov zmanjšuje vrednost zaključnega dela [6].

Težave nam je povzročalo tudi iskanje navedenih sklicev v seznamu literature in virov. Ta seznam so nekateri študentje navajali napačno in na več različnih načinov:

1. naštevanje z vezajem (-),
2. naštevanje s piko (•),
3. seznam literature in virov NI bil naveden za vsemi poglavji na svoji strani, ampak je bil mnogokrat pred mnogimi poglavji, včasih kar na enaki strani kot neko drugo poglavje.

Zaradi vseh teh napak naša programska rešitev ni mogla uspešno napolniti svojega seznama z viri, na podlagi katerih je nato iskala sklice po dokumentu. Ta problem je eden izmed glavnih razlogov, zakaj smo morali izmed 100 dokumentov zavreči 31 dokumentov.

Razlog, kateremu se prav tako nismo morali izogniti, je bilo napačno sklicevanje virov. Avtorji so se na sam vir sklicevali napačno na več načinov:

1. sklicevanje brez oklepajev () ali brez zavitih oklepajev [], v katerih bi naj bila navedena zaporedna številka vira oziroma avtor,
2. sklicevanje v obliki 1. ali 1,
3. sklicevanje v obliki (vir1 vir2 vir3 itd.), kjer sklici med seboj niso bili pravilno ločeni. Za ločevanje sklicev so študentje uporabljali znake, kot je na primer podpičje (;),
4. sklicevanje na vir, ki na seznamu virov ne obstaja,
5. sklicevanje po začetnicah imena oziroma priimka avtorja,
6. nekateri študentje so se sklicevali na prilogo na enak način kot na vir, zato je naša programska rešitev napačno razpoznala sklic.

Probleme nam je povzročala tudi implementacija regularnih izrazov. Problem je bil ta, da so avtorji včasih navajali in se sklicevali na vire na dva ali več različnih načinov. Potrebno je bilo zapisati vgnezdene regularne izraze (glej poglavje 3), ki so v nekaterih dokumentih delovali, v nekaterih pa ne. Nekatere vgnezdene regularne izraze je bilo potrebno ločiti na več regularnih izrazov. Tako smo najprej preverili najpogosteje uporabljene načine navedbe in sklicevanja virov, nato pa tiste manj pogoste, da smo tako pokrili čim več načinov navedb.

## 6. ZAKLJUČEK

Plagiatorstvo je zaradi čedalje lažjega dostopa do elektronskih virov vse bolj pogosto. Skozi raziskovalno nalogo smo prikazali, kako bi znotraj dokumenta razpoznali literaturo in poiskali njihove sklice v besedilu. Vključitev takega načina preverjanja plagiarizma bi po našem mnenju veliko pripomogel k bolj natančni detekciji plagiata, saj veliko tistih, ki poizkusijo oddati plagiat, spremenijo vsebino, literatura pa ostane nespremenjena. Največja prednost našega pristopa je torej ta, da je pristop neodvisen od jezika, v katerem je dokument napisan.

Z uspešnostjo našega pristopa smo zadovoljni, vendar je uspešnost razpoznave literature in kasneje določitve njihovih sklicev v besedilu v veliki meri odvisna od "pravilnosti" sestave dokumenta. Uspešnost bi lahko izboljšali tako, da bi zapisali še več bolj specifičnih regularnih izrazov in tako učinkoviteje razpoznali literaturo v različnih dokumentih.

## 7. LITERATURA

[1] Kontingenčna tabela. https://en.wikipedia.org/wiki/Contingency_table. Dostopano 10. 5. 2018.

[2] Metrika f1. https://en.wikipedia.org/wiki/F1_score. Dostopano 10. 5. 2018.

[3] Plagiat. http://bos.zrc-sazu.si/cgi/a03.exe?name=sskj_testa&expression=ge%3Dplagiat*&hs=1. Dostopano 29. 4. 2018.

[4] E. S. M. K. E. A. Benno Stein, Paolo Rosso. Pan workshop. uncovering plagiarism, authorship and social software misuse. https://pdfs.semanticscholar.org/160b/400d726eb042d0867d537c447e858716e7b7.pdf. Dostopano 8. 6. 2018.

[5] J. D. Borko Boškovič and J. Brest. *Študijska literatura pri predmetu Jezikovne tehnologije*. 2018.

[6] FERI. Navodila za pisanje zaključnih del na študijskih programih prve in druge stopnje um feri. 2018.

[7] B. Gipp and J. Beel. Citation based plagiarism detection - a new approach to identify plagiarized work language independently. *HT '10 Proceedings of the 21st ACM conference on Hypertext and hypermedia*, 2010.

[8] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 1963.

[9] I. V. Marshakova. System of document connections based on references. http://garfield.library.upenn.edu/marshakova/marshakovanauchtechn1973.pdf. Dostopano 8. 6. 2018.

[10] H. Small. *Co-citation in the scientific literature: a new measure of the relationship between two documents.* 1973.

# Klasifikacija samomorilskih pisem

Dejan Rupnik
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
Maribor, Slovenija
dejan.rupnik@student.um.si

Denis Ekart
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
Maribor, Slovenija
denis.ekart@student.um.si

Gregor Kovačević
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
Maribor, Slovenija
gregor.kovacevic@student.um.si

Dejan Orter
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
Maribor, Slovenija
dejan.orter@student.um.si

Alen Verk
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
Maribor, Slovenija
alen.verk@student.um.si

Borko Bošković
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
Maribor, Slovenija
borko.boskovic@um.si

## POVZETEK

V našem delu smo se osredotočili na klasifikacijo poslovilnih pisem samomorilcev. Pozornost smo posvetili na razpoznavo pristnih pisem te narave, od pisem, ki to niso, oz. so le-ta lažna. S pomočjo procesiranja naravnega jezika in algoritmov strojnega učenja želimo doseči, da se pristna pisma v večini ločijo od lažnih. Implementirali smo program v programskem jeziku Python, pripravili korpus in izvedli nadzorovano strojno učenje. Pisma smo klasificirali z metodami: DecisionTreeClassifier, SVC, GaussianProcessClassifier, AdaBoostClassifier, KNeighborsClassifier, RandomForestClassifier, MLPClassifier, GaussianNB in QuadraticDiscriminantAnalysis. Najboljše rezultate smo dosegli z odločitvenim drevesom, kjer smo dosegli 68% natančnost.

## KLJUČNE BESEDE

klasifikacija samomorilskih pisem, procesiranje naravnega jezika, odločitvena drevesa

## 1. UVOD

Zaradi samomora vsako leto umre več kot 800.000 oseb (v Sloveniji več kot 300), približno 25-krat toliko pa jih samomor poskuša narediti [14][11]. Velikokrat, ko vidimo samomorilen zapis, smo v dilemi ali ta oseba misli resno ali pa je to poizkus iskanja pozornosti [7]. Nekatera pisma so tudi lažna - na primer pri umorih, kjer bi želel storilec prikazati, da je žrtev storila samomor. Posledično je njihova klasifikacija pomembna za preventivo ter za razreševanje morebitnih nejasnosti.

Za samo izvedbo klasifikacije smo najprej potrebovali pristna pisma samomorilcev ter lažna pisma. Korpus pristnih pisem smo pridobili s spletnih strani [12, 3, 6, 4], lažna pa smo sestavili sami. Vsa pisma so v angleškem jeziku. Uporabili smo 63 pristnih in 19 lažnih pisem. Postopek analize pisem je potekal v več korakih. Prvi izmed njih je bilo predprocesiranje besedila, kjer smo kot rezultat dobili prečiščeno besedilo (tj. samo standardni znaki, brez simbolov in ločil). Nato smo izvedli oblikoslovno označevanje besed in povezavo ključnih besed s posameznimi koncepti iz tega področja. Naredili smo statistiko posameznih pisem (povprečno število besed ipd.). Izvedli smo tudi teste berljivosti, kateri so izračunali dve različni metriki. Tudi te smo uporabili pri strojnem učenju. Dodatno smo opravili tudi analizo čustev, nato pa vse rezultate združili v datoteke CSV. Te so bile osnova za gradnjo odločitvenih dreves pri strojnem učenju. Uporabljeno je bilo nadzorovano strojno učenje. Rezultati eksperimenta so pokazali, da je uporabljen pristop uspešno ločeval med pristnimi in lažnimi pismi.

## 2. SORODNA DELA

V članku [10] so avtorji uporabili 66 poslovilnih pisem, od katerih je bilo 33 pristnih in 33 lažnih. V raziskavi je prisostvovalo 11 strokovnjakov s področja mentalnega zdravja in 31 psihiatrov pripravnikov. Zbrana mnenja so primerjali z 9 algoritmi strojnega učenja:

- LMT,
- LinSMO,
- Descision,
- JRip,
- NB,
- PART,
- J48,
- Logistic,
- IB3 in
- OneR.

Kombinacija določenih algoritmov je v povprečju dala 78% natančnost. Specialisti izbranega področja so dosegli 63% natančnost, pripravniki pa zgolj 49% natančnost pri ločevanju pisem.

V članku [2] so naredili študijo vsebine samomorilskih pisem ljudi, ki so samomor skušali storiti ter druge skupine ljudi, ki je samomor dejansko storila. Študijo so izvedli s programom za tekstovno analizo, kateri uporablja jezikovno povpraševanje in štetje besed (angl. Linguistic Inquiry and Word Count).

Članek [1] opisuje študijo samomorilskih pisem iz Mehike. Preučevali so vzorec 212 samomorov, kjer je 106 oseb napisalo poslovilno pismo, ostalih 106 pa tega ni storilo. Ugotovili so, da se osebnostne lastnosti tistih, ki poslovilna pisma napišejo in tistih oseb, ki jih ne, pretirano ne razlikujejo.

## 3. PRIPRAVA BESEDILA

### 3.1 Predprocesiranje

Ko smo pripravili korpus s poslovilnimi pismi samomorilcev, je bilo le-te potrebno pripraviti za nadaljnjo uporabo. Iz pisem smo odstranili ločila, številke ter vse ostale nestandardne znake. Pri predprocesiranju besedila smo prav tako pretvorili velike črke v male.

**Algoritem 1:** Priprava besedila.

```
Input   : neobdelano besedilo ($besedilo$)
Output: obdelano besedilo ($izhod$)
for $znak$ $in$ $besedilo$ do
    if $znak$=$JE\_CRKA()$ then
        if $znak$= $JE\_VELIKA()$ then
            DODAJ_V_IZHOD(SPREMENI_V_MALO($znak$))
        else
            DODAJ_V_IZHOD($znak$)
        end
    else if $znak$=$JE\_PRESLEDEK()$ then
        DODAJ_V_IZHOD($znak$)
    else if $znak$=$JE\_OPUŠČAJ()$ then
        DODAJ_V_IZHOD($znak$)
    end
end
```

### 3.2 Oblikoslovno označevanje

Za meritve povprečij in ostalih statističnih podatkov smo naredili oblikoslovno označevanje (angl. parts of speech tagging), kjer smo implementirali tokenizacijo in označili besede glede na pomen v povedi. Vsaki besedi se je po zagonu algoritma dodala oznaka, katera nam je povedala za kakšno vrsto besede gre. Uporabili smo knjižnico NLTK [8]. Bila je dodana podpora za večnitno delovanje, ker je lahko proces na velikih korpusih časovno zelo potraten. Metoda z večnitnim delovanjem rezultatov ne shranjuje v spremenljivke v pomnilniku, temveč jih v datoteke na trdem disku.

Po uspešni implementaciji algoritma smo sešteli vse pojavitve oznak ter naredili slovar povprečnih pojavitev določenih označb. Torej, koliko glagolov, ločil, samostalnikov itd., vsebuje povprečno pismo. Nato smo seznam označb normalizirali na omejen nabor značk. Pridobili smo tudi statistiko za vsako posamezno pismo, kjer ohranimo podatek, kateri slovar pripada kateremu pismu.

### 3.3 Test berljivosti

Nad korpusom smo izvedli testa berljivosti po Flesch in Flesch-Kicaidovi metodi. Ocena berljivosti besedila (angl. Flesh score) smo izračunali po enačbi (1), kjer RE predstavlja enostavnost branja (angl. readability ease).

$$RE = 206,83 - (1,015 * ASL) - (84,6 * ASW) \tag{1}$$

V enačbi (2) ASL predstavlja povprečno dolžino stavka (angl. average sentence length).

$$ASL = \frac{stBesed}{stStavkov} \tag{2}$$

V enačbi (3) ASW predstavlja povprečno število zlogov na besedo (angl. average number of syllables per word).

$$ASW = \frac{stZlogov}{stBesed} \tag{3}$$

Višji je RE (ocena berljivosti besedila) manjša je kompleksnost besedila. Metoda vrne oceno berljivosti besedila (angl. readability score), katera je realno število med 0 in 100, vendar pa zaradi napak v formatiranju lahko pride do odstopanja. Ta odstopanja smo normalizirali na pričakovane vrednosti. Flesch-Kinicaid metoda nam pove nivo berljivosti besedila in se izračuna po enačbi (4), kjer TW predstavlja število vseh besed (angl. total words), TS je število vseh stavkov (angl. total sentences) in TSYL število vseh zlogov (angl. total syllables). Pričakovan rezultat je decimalno število med 0 in 13, katero sovpada s povprečnim nivojem berljivosti v šolskih razredih v Združenih Državah Amerike. Zaradi zahteve po številu besed v stavkih, smo teste bralnosti izvajali na osnovnem - neprocesiranem besedilu [13].

$$RL = 0,39 * (\frac{TW}{TS}) + 11,8 * (\frac{TSYL}{TW}) - 15,59 \tag{4}$$

Vrednosti RE in RL smo vključili v datoteko CSV.

### 3.4 Analiza čustev

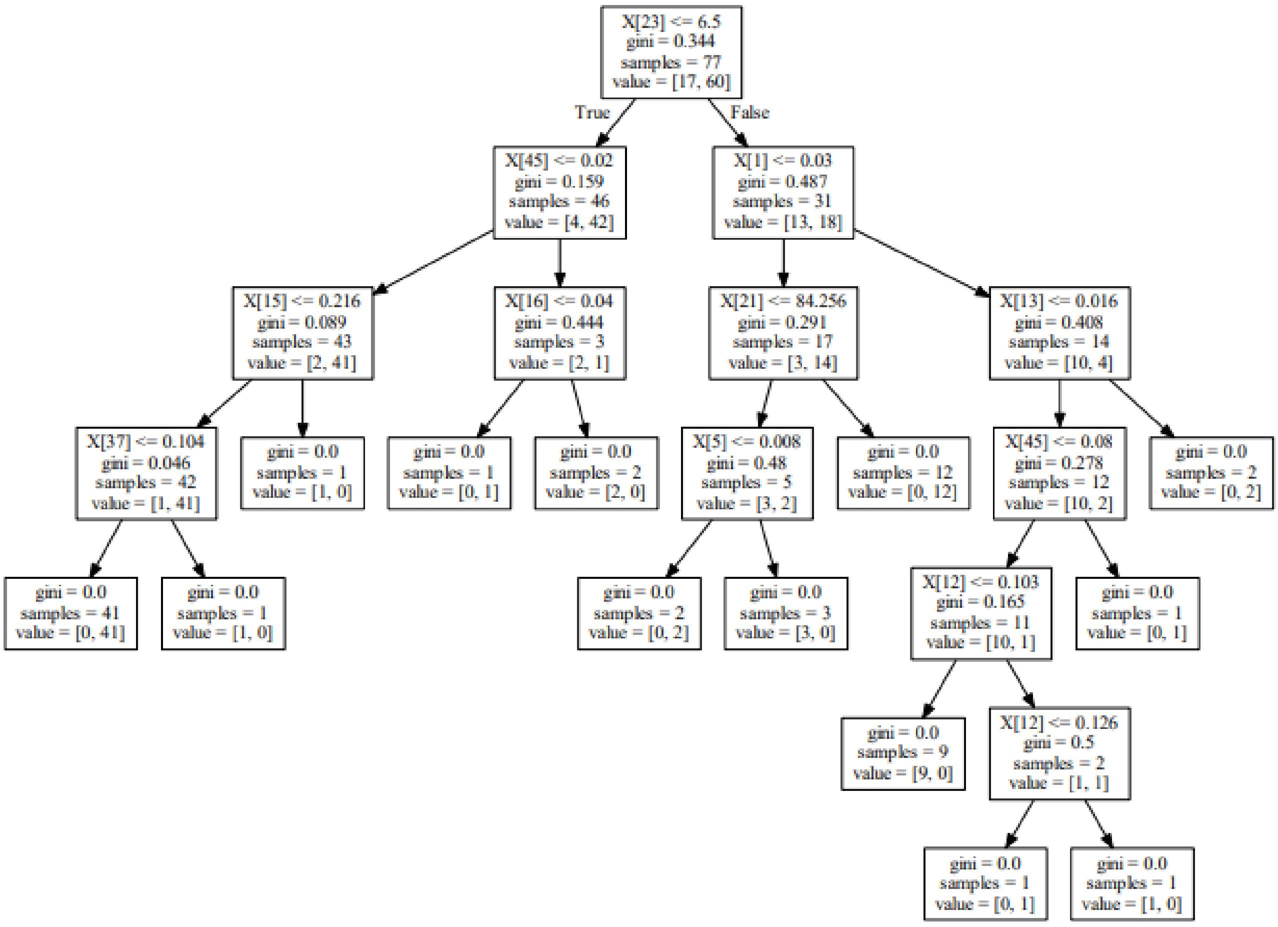
Za uspešno delovanje algoritma smo potrebovali tudi analizo čustev. Ta nam iz vsakega pisma izlušči emocije, ki se jih da razpoznati na podlagi zapisanega besedila. Pri analizi smo si pomagali z modulom WNAffect [5]. Procesirano besedilo smo vstavili v funkcijo in kot rezultat dobili slovar čustev, ki se pojavijo v besedilu ter vsoto njihovih magnitud. Na koncu smo iz vseh obdelanih pisem in pridobljenih unikatnih čustev ročno generirali statični slovar vseh emocij.

**Algoritem 2:** Pridobivanje čustev.

```
function GetEmotion ($besedilo$);
Input   : označeno besedilo ($besedilo$)
wna=WNAffect()
for $znacka$ $in$ $besedilo$ do
    if $znacka$ $!=$ $LOČILO$ then
        $custva$=PRIDOBI_ČUSTVA()
        if $custvo$ $!=$ $NONE$ then
            return ($znacka$, $custvo$, $stopnja$)
        end
    end
end
```

## 4. ZDRUŽITEV REZULTATOV FUNKCIJ

Ko smo implementirali vse omenjene funkcije, smo morali rezultate združiti ter jih pripraviti za strojno učenje. Rezultate smo zato združili v datoteke CSV. V prvo izmed datotek



**Slika 1: Generirano odločitveno drevo.**

smo po vrsticah shranili imena pisem, oznake, ocene ter nivo bralnosti in statistiko (št. besed, št. stavkov, itd.) za vsako posamezno pismo.

V drugo datoteko smo shranili skupna povprečja vseh pristnih in nepristnih pisem za vsakega izmed klasifikacijskih atributov posebej. Omeniti je potrebno, da so posamezne vrednosti, kot na primer: značke oblikoslovnega označevanja in magnitude posameznih čustev, normalizirane s številom besed v besedilu. Tabela 1 prikazuje del datoteke povprecja.csv, ki vsebuje povprečja atributov.

## 5. NADZOROVANO STROJNO UČENJE

Ko smo pripravili datoteki CSV smo se morali odločiti katero vrsto strojnega učenja bomo uporabili. Uporabili smo nadzorovano strojno učenje in zgradili odločitveno drevo, prikazano na sliki 1.

Za generiranje odločitvenega drevesa smo uporabili knjižnico sklearn [9], katera loči bazo klasifikacijskih podatkov na učno in testno množico ter na dva vektorja klasifikatorjev. S pomočjo teh spremenljivk smo nato generirali napovedni model. Iz napovednega modela pa izračunali preciznost, priklic in mero F1.

Skupno smo uporabili deset različnih klasifikatorjev, podrobnejši rezultati štirih najuspešnejših pa so predstavljeni v poglavju 5.1:

- DecisionTreeClassifier(max_depth=5),
- SVC(kernel=”linear”, C=0,025),
- GaussianProcessClassifier(1,0 * RBF(1,0)),
- AdaBoostClassifier(),
- KNeighborsClassifier(3),
- SVC(gamma=2, C=1),
- RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
- MLPClassifier(alpha=1),
- GaussianNB() in
- QuadraticDiscriminantAnalysis().

### 5.1 Rezultati

V sledečih tabelah so prikazani rezultati klasifikacije z uporabo različnih klasifikatorjev. Rezultati so bili izračunani iz povprečja 10.000 instanc klasifikacije, kjer se je vsakič naključno izbralo 95% pisem za učno in 5% pisem za testno

**Tabela 1: Povprečne vrednosti atributov, ki jih uporabimo pri strojnem učenju.**

| | Pristna pisma | Nepristna pisma |
|---|---|---|
| Ocena berljivosti | 85,14 | 84,35 |
| Nivo berljivosti | 4,40 | 5,38 |
| MD | 0,02 | 0,02 |
| PRP | 0,04 | 0,05 |
| RB | 0,06 | 0,07 |
| NN | 0,18 | 0,19 |
| VBP | 0,06 | 0,07 |
| JJ | 0,09 | 0,09 |
| IN | 0,09 | 0,10 |
| DT | 0,08 | 0,6 |
| Število besed | 79,59 | 173,42 |
| Število zlogov | 94,24 | 209,79 |
| Število znakov | 304,98 | 691,32 |
| Število povedi | 7,37 | 14,05 |
| Upanje | 0,02 | 0,00 |
| Ljubezen | 0,05 | 0,02 |
| Obup | 0,01 | 0,00 |
| Sreča | 0,01 | 0,00 |
| Nejasnost | 0,01 | 0,00 |
| Spokojnost | 0,02 | 0,00 |
| Začudenje | 0,02 | 0,01 |
| Sproščenost | 0,01 | 0,00 |
| Zgroženost | 0,01 | 0,00 |
| Žalost | 0,01 | 0,01 |
| Odpuščanje | 0,01 | 0,00 |
| Dobrohotnost | 0,01 | 0,00 |
| Stiska | 0,01 | 0,00 |
| Gotovost | 0,01 | 0,00 |
| Strah | 0,01 | 0,00 |
| Naklonjenost | 0,01 | 0,00 |

množico. Instanca v našem primeru predstavlja naključno razdelitev pisem na učno in testno množico (iz nabora vseh pisem - pristnih in lažnih). Prikazani so rezultati preciznosti in priklica za posamezen razred, kot tudi povprečne vrednosti. Za mero F1 so izračunane povprečne vrednosti. Za tovrsten pristop testiranja smo se odločili zaradi majhne učne množice.

Za najboljši klasifikator se je izkazala metoda odločitvenih dreves (angl. decision tree classifier), katera je dosegla 68% natančnost, ostale metrike pa so prikazane v tabeli 2. Za izračun povprečja je bilo v vseh tabelah uporabljeno makro povprečje. Tudi metoda AdaBoost se je izkazala kot zelo natančno, vendar je imela malenkost nižjo vrednost priklica. Pri našem testiranju se je tudi izkazala kot bistveno počasnejša, kot metoda odločitvenih dreves.

Rezultati klasifikatorja SVC z linearnim jedrom so zapisani v tabeli 3, klasifikatorja GaussianProcess v tabeli 4 in rezultati klasifikatorja AdaBoost v tabeli 5.

Po analizi klasifikatorjev smo odločitveno drevo še grafično predstavili v formatu PDF. Po pregledu korenov smo ugotovili, da izločanje določenih atributov vrne boljše rezultate. Najboljše rezultate smo dosegli pri izločitvi atributov za število znakov, besed in črk. Preciznost je bila izboljšana za 4%, priklic za 6% in mera F1 za 5%.

**Tabela 2: Rezultati klasifikatorja DecisionTree.**

| | preciznost | priklic | mera F1 |
|---|---|---|---|
| nepravilna | 0,52 | 0,50 | 0,49 |
| pravilna | 0,84 | 0,89 | 0,85 |
| skupno povprečje | 0,68 | 0,70 | 0,67 |

**Tabela 3: Rezultati klasifikatorja SVC.**

| | preciznost | priklic | mera F1 |
|---|---|---|---|
| nepravilna | 0,27 | 0,27 | 0,27 |
| pravilna | 0,77 | 1,00 | 0,86 |
| skupno povprečje | 0,52 | 0,63 | 0,56 |

**Tabela 4: Rezultati klasifikatorja GaussianProcess.**

| | preciznost | priklic | mera F1 |
|---|---|---|---|
| nepravilna | 0,38 | 0,34 | 0,35 |
| pravilna | 0,80 | 0,94 | 0,85 |
| skupno povprečje | 0,59 | 0,64 | 0,60 |

**Tabela 5: Rezultati klasifikatorja AdaBoost.**

| | preciznost | priklic | mera F1 |
|---|---|---|---|
| nepravilna | 0,51 | 0,50 | 0,49 |
| pravilna | 0,84 | 0,89 | 0,84 |
| skupno povprečje | 0,68 | 0,69 | 0,67 |

## 6. ZAKLJUČEK

V tem delu smo se osredotočili na klasifikacijo samomorilnih pisem. Program pisma analizira postopoma. Najprej nad korpusom izvedemo predprocesiranje besedila, da dobimo čisto besedilo (same male črke, brez ostalih znakov). Nato se izvede oblikoslovno označevanje besed in statistika posameznih pisem. Opravili smo tudi test berljivosti besedila, ki ga uporabimo pri strojnem učenju. Analiziramo tudi čustva, nato pa vse skupaj združimo v datoteko CSV in izvedemo nadzorovano strojno učenje.

Kot je razvidno iz rezultatov smo zadani projekt razpoznave pristnih poslovilnih pisem uspešno izvedli. Po podatkih avtorjev članka [10] so strokovnjaki s področja psihologije dosegli 63% natančnost, medtem ko naša metoda vrača rezultate z 68% natančnostjo.

Tukaj je pomembno opozoriti, da v našem članku nismo uporabili istega nabora pisem, kot so ga avtorji prej omenjenega članka. Posledično rezultatov naših eksperimentov ni mogoče neposredno primerjati z rezultati v sorodnih člankih. Pridobitev pristnih in lažnih pisem, ki ustrezajo pogojem, je izredno zapleteno opravilo, pri katerem bi za večjo zanesljivost pri zagotavljanju norm potrebovali ustrezno usposobljene osebe, ki bi obenem imele dostop do tovrstnih arhivov.

V prihodnosti bi lahko metodo razširili tako, da ne bi bila omejena samo na angleški jezik. Izboljšave bi tudi prinesla večja množica pisem in dodatne metode za ocenjevanje strukture besedila.

## 7. VIRI

[1] L. A. L. L. Chávez-Hernández AM1, Páramo D. Suicide notes in mexico: what do they tell us? *Suicide Life Threat Behav*, 36:709–15, December 2006.

[2] L. D. Handelman LD. The content of suicide notes from attempters and completers. *Oxford University Press*, 28:102–104, 2007.

[3] J. Harper. A collection of real suicide notes | historic mysteries, 2015.

[4] A. A. Leenaars. Suicide notes in the courtroom. 1999.

[5] C. Michard. clemtoy/wnaffect a python module to get the emotion of a word., 2017.

[6] O. P. M. Mukta Rani, Shalini Girdhar. Suicide note: The last words. 2015.

[7] NIJZ. Svetovni dan preprečevanja samomora: ”vzemi si trenutek, reši življenje”, 2017.

[8] NLTK. Natural language toolkit, 2017.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] J. Pestian, H. Nasrallah, P. Matykiewicz, A. Bennett, and A. Leenaars. Suicide note classification using natural language processing: A content analysis. *Biomedical Informatics Insights*, 2010:19–28, August 2010.

[11] Politikis. Zaradi samomora vsako leto v sloveniji umre več kot 300 ljudi, a zaznati je upad tega žalostnega dejanja, 2017.

[12] J. Russel. A collection of suicide notes & letters - russel’s cyber journal, 2008.

[13] stephenhky. Flesch-kincaid readability measure - everything about data analytics big data, data analytics, 2016.

[14] Wikipedia. Suicide, 2018.

# 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 prevzemajo 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.

## Kjučne besede

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

[*]Dopisni avtor

[†]Pomoč pri upravljanjem 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 uporabiškega 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. Določitve 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.

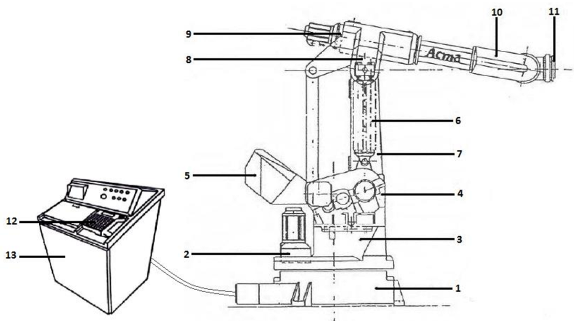
Struktura članka je v nadaljevanju naslednja. V poglavju 2 predstavimo robota ACMA XR701. Opis predlagane rešitve je vsebina poglavja 3. Rezultati so predstavljeni v poglavju 4. Članek zaključimo s poglavjem 5, kjer povzamemo opravljeno delo in napovemo možnosti za nadaljnje delo.

## 2. ROBOT ACMA XR701

Robot ACMA XR701 je bil zasnovan leta 1994 v podjetju Renault Automation. To je antropomorfni 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
2. Motor z gonilom (1. os)
3. Vrtljivo podnožje
4. Motor z gonilom (2. os)
5. Protiutež
6. Ravnotežna vzmet
7. Drugi člen robota
8. Tretji člen robota (3. os)
9. Motor z gonilom (4. os)
10. Sklep drugega in tretjega člena (5. os)
11. Pritrdilna prirobnica (6. os)
12. Prenosna konzola
13. Krmilna omara

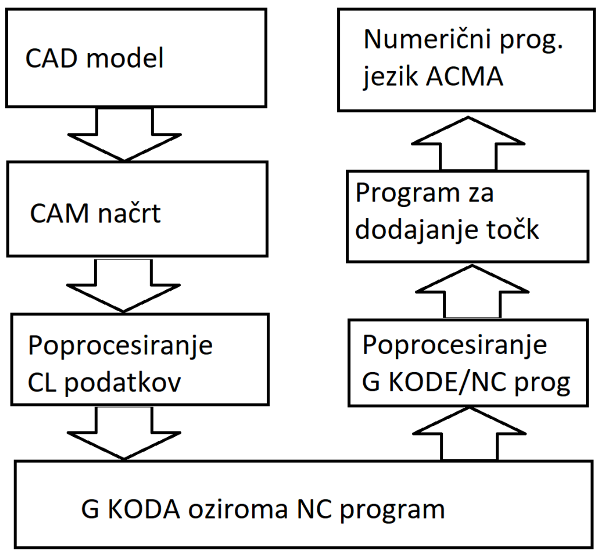
### 2.1 Primerjava CNC stroja z robotom

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.



CAD model
CAM načrt
Poprocesiranje CL podatkov
G KODA oziroma NC program
Poprocesiranje G KODE/NC prog
Program za dodajanje točk
Numerični prog. jezik ACMA

**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 kompleksnost naprav in programov, saj jih praviloma uporabljajo manj izobraženi kadri. Zato se za pohitritev in zmanjševanje napak pri krmiljenju 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 nastavitev 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 programa 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 opremiljen 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 Psevdokodom 1. Kjer spremenljivke z velikimi tiskanimi črkami predstavljajo sistemski nizi pridobljen iz vmesnika GUI. Psevdokod 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: OUTAACMA->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_G kode* 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 podpoglavje 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" +@68
```

**Slika 3: Sekvenca za nastavitev ničelne točke**

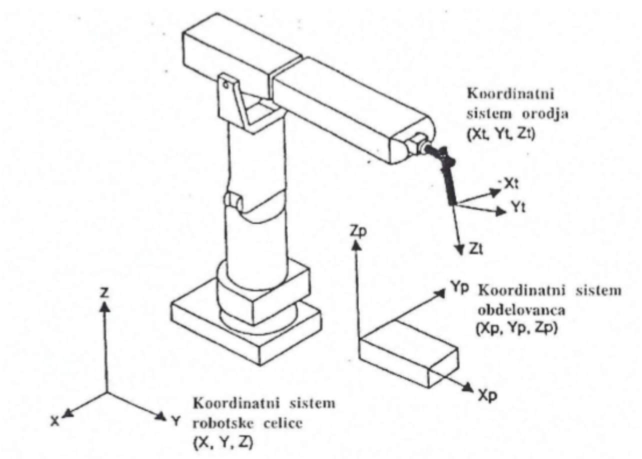
```
'8' @77 13 +0 13 +0 @77 @77 @77 13 +0
```

**Slika 4: Sekvenca za nastavitev načina dodajanje koordinat**

**Algoritem 2** Psevdokoda za določanje sekvence za nastavitev konfiguracije robota

```
if $Rama$ is $Levo$ then
    AcmaIzhod.append("13 +0 13 +0\n");
else if $Rama$ is $Spredaj$ then
    AcmaIzhod.append("13 +0\n");
end if
if $Komolec$ is $Zgoraj$ then
    AcmaIzhod.append("@77 13 +0\n");
end if
if $Zapestje$ is $Nizko$ then
    AcmaIzhod.append("@77 @77 13 +0\n");
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.


Koordinatni sistem orodja (Xt, Yt, Zt)
Xt
Yt
Zt
Zp
Yp
Koordinatni sistem obdelovanca (Xp, Yp, Zp)
Xp
Z
X
Y
Koordinatni sistem robotske celice (X, Y, Z)

**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 zapolnjenost 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 $\backslash n$ do
 2:     if $TrenutniZnak$ is $G$
    and ($NaslednjiZnak$ is $Stevilo\,0-3$) then
 3:         Nastavi na naslednji znak;
 4:         Vrstica[0] = Trenutna vrednost;
 5:         UkazInicializiran = true;
 6:         UkazVpisan[0] = true;
 7:         Zmanjšaj StPraznihPolj;
 8:     end if
 9:     if $TrenutniZnak$ is $X$
    and $Naslednjiznak$ is $Stevilo$
    and $UkazVpisan$[0] is $true$
    and $UkazInicializiran$ is $true$ then
10:         Nastavi na naslednji znak;
11:         Vrstica[1] = Pridobi_Vrednost();
12:         UkazVpisan[1] = true;
13:         Zmanjšaj StPraznihPolj;
14:     end if
15:     //Ponovitev za parametre Y, Z, A, B, C, I, J, K
16:     Nastavi na naslednji znak;
17: 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 dvojiško vrednost.

**Algoritem 5** Preverjanje zapolnjenost vrstice

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

### 3.2.3 *Pretvorba krožnih izsekov G2 in G3*

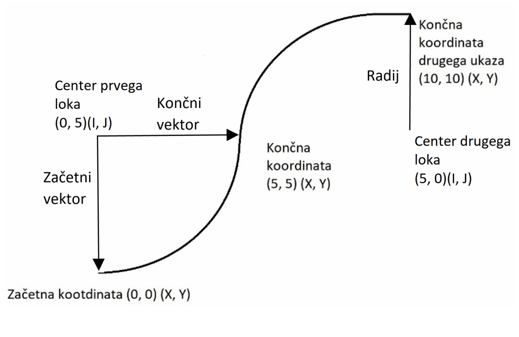
Kot je bilo že omenjeno, v GUI dodajamo program NC za 3/5 osnega robota. Program za 5 osnega robota preprosto vrednosti polja *Vrstica* prišteje nastavitev robota in strukturira izhodni program ACMA za dodajanje linearnih točk z določenimi rotacijami. Določanje koordinat gibov v programu NC za 3 osnega 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 predstavo smo pretvorbo *G2* in *G3* krožnega giba predstavili z realnim primerom, prikazanim na Sliki 8 in Sliko 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 literature [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.

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

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

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

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

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

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

$$\theta = arctan\ \frac{y}{x} \tag{4}$$

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

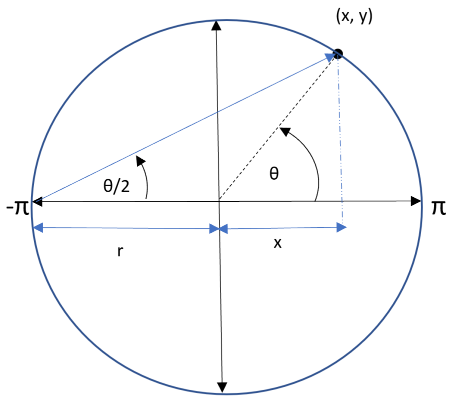
Za rešitev težave s Slike 10 predpostavimo, da velja $y = sin\ \theta$ in $x = r + cos\ \theta$ in določimo novo Enačbo (5) ter jo preuredimo v Enačbo (6). S pomočjo skalarnega produkta in Pitagorovega izreka izračunamo polmer krožnega izseka in s tem dobimo končno Enačbo (7).

$$\frac{\theta}{2} = arctan\ \frac{y}{r + x} \tag{5}$$

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

$$\theta = 2\ arctan\ \frac{y}{\sqrt{x^2 + y^2} + x} \tag{7}$$

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



**Slika 10: Prikaz izpeljave inverznega tangensa 2**

$$l = 2 \cdot \pi \cdot r \cdot \frac{\theta}{360} \tag{8}$$

S tem izračunamo število interpolacij z dolžino loka v odvisnosti od natančnosti robota in določimo kot interpolacije s kotom med vektorjema v odvisnosti od število 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 katerga sledi Enačba (9).

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

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

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

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 $R_\psi$), če imamo gib v ravnini $YZ$, prilagajamo rotacijo okoli osi $y$ (rotacija $R_\theta$). 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 generiranem 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 preveri 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 poprocesor 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.

# Using constrained exhaustive search vs. greedy heuristic search for classification rule learning

Jamolbek Mattiev
PhD student, teaching assistant
University of Primorska
Slovenia
(+386)-70366331
jamolbek.mattiev@famnit.upr.si

Branko Kavšek
Assistant Professor
University of Primorska, Jožef Stefan Institute
Slovenia
(+386)-56117654
branko.kavsek@upr.si

## ABSTRACT

Existing classification rule learning algorithms use mainly greedy heuristic search to find regularities (e.g., a decision tree or a set of rules) in data for classification. In recent years, extensive research was done in the machine learning community on learning rules using exhaustive search – association rule mining. The objective there is to find all rules in data that satisfy the user-specified minimum support and minimum confidence constraints. Although the whole set of rules may not be used directly for accurate classification, effective and efficient classifiers have been built using these, so called, classification association rules.

In this paper we compare a "classical" classification rule learning algorithm that uses greedy heuristic search to produce the final classifier (a set of decision rules) with a class association rule learner that uses constrained exhaustive search to find classification rules on a "real life" dataset.

The results show that the "classical" classification rule learning algorithm generates a more compact classifier whose individual rules are somehow "less accurate". On the other hand, the class association rule learner produces individual classification rules that are "highly accurate" but the overall classification accuracy of the classifier remains yet to be checked.

## CCS Concepts

**• Computing methodologies → Machine learning → Machine learning approaches → Rule learning**
**• Computing methodologies → Machine learning → Cross-validation**
**• Computing methodologies → Machine learning → Learning paradigms → Supervised learning → Supervised learning by classification**

## Keywords

Data mining, machine learning, classification rules, association rules, search, heuristics.

## 1. INTRODUCTION

Classification rule mining and association rule mining are two important data mining techniques. Classification rule mining aims at discovering a small set of rules in the database to form an accurate classifier. Association rule mining finds all rules in the database that satisfy some minimum support and minimum confidence constraints [2]. For association rule mining, the target of mining is not predetermined, while for classification rule mining there is one and only one pre-determined target, i.e., the class or dependent attribute.

Association rule mining is the discovery of what are commonly called association rules. Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in [1]. It studies the frequency of items occurring together in transactional databases, and based on a threshold called support, identifies the frequent itemsets. Another threshold, confidence, which is the conditional probability that an item appears in a transaction when another item appears, is used to pinpoint association rules.

In classification [4], by the help of the analysis of training data we develop a model which is then used to predict the class of objects whose class label is not known. The model is trained so that it can distinguish different classes in the data. The training data is having data objects whose class label is known in advance. Classification analysis is the organization of data in given classes. Also known as supervised classification, the classification uses given class labels to order the objects in the data collection. Classification approaches normally use a training set where all objects are already associated with known class labels.

There are many classification approaches such as statistical [8], divide-and-conquer [6] and covering [3] approaches. Based on these approaches numerous algorithms have been derived such as PART [5], Naive Bayes [8], See5 [12], C4.5 [11], Prism [3] and RIPPER [6]. However, traditional classification techniques often produce a small subset of rules, and therefore usually tend to miss detailed rules that might play an important role.

The aim of our research presented in this paper is to compare a "classical" state-of-the-art classification rule learning algorithm that uses greedy heuristic search (PART [5]) with a class association rule learner that uses constrained exhaustive search (a modified version of the APRIORI algorithm [2]).

## 2. PRELIMINARY CONCEPTS

Let $D$ be a dataset with $n$ attributes $\{A_1, A_2, \dots, A_n\}$ and $|D|$ records (objects) where each record has an object identifier (OID). Let $C = \{c_1, c_2, \dots, c_k\}$ be a list of class labels. A specific value of an attribute $A_i$ and class $C$ is denoted by lower-case letters $a_{im}$ and $c_j$ respectively.

**Definition 1**. An item is described as an attribute and a specific value for that attribute, denoted by $\langle(A_i, a_{im})\rangle$, e.g. $\langle(A_1, a_{11})\rangle$, $\langle(A_1, a_{12})\rangle$, $\langle(A_2, a_{21})\rangle$, etc.

**Definition 2.** An itemset is a set of items, e.g., $\langle(A_1, a_{11}),(A_2, a_{21})\rangle$, $\langle(A_1, a_{11}),(A_3, a_{31})\rangle$, etc.

**Definition 3.** A *Constraint_Itemsets* is a user-defined set of itemsets in which each itemset is said to be a constrained itemset, e.g., $Constraint\_Itemsets = \{\langle(A_1, a_{11}),(A_2, a_{21})\rangle, \langle(A_3, a_{31})\rangle\}$.

**Definition 4.** A class association rule $R$ has the form $itemset \rightarrow c_j$ where $c_j \in C$ is a class label.

**Definition 5.** A rule *R* satisfies *Constraint_Itemsets* if its antecedent (itemset) contains at least one itemset in *Constraint_Itemsets*.

**Definition 6**. The actual occurrence *ActOcc(R)* of a rule *R* in *D* is the number of records of *D* that match *R*'s antecedent.

**Definition 7.** The support of rule *R*, denoted by *Supp(R)*, is the number of records of *D* that match *R*'s antecedent and are labeled with *R*'s class.

**Definition 8**. The confidence of rule *R*, denoted by *Conf(R)*, is defined by equation (1) as follows:

$$Conf(R) = \frac{Supp(R)}{ActOcc(R)} \tag{1}$$

## 3. PROBLEM STATEMENT

Consider a relational data set *D* with *n* attributes. A record of *D* is a set of attribute-value pairs, denoted by *T*. A pattern is a subset of a record. We say, a pattern is a *k*-pattern if it contains *k* attribute-value pairs. All the records in *D* are categorized by a set of classes *C*.

It is required to generate the complete set of Class Association Rules by Apriori and PART algorithms. Another focus will be on comparing the advantages and disadvantages of using these two algorithms.

### 3.1 Apriori Algorithm

Association rule generation [14] is usually split up into two separate steps:

1. First, minimum support is applied to find all frequent itemsets in a database.
2. Second, these frequent itemsets and the minimum confidence constraint are used to form rules.

While the second step is straight forward, the first step needs more attention. Finding all frequent itemsets in a database is time consuming since it involves searching all possible itemsets (item combinations). The set of possible itemsets is the power set over *I* (the set of all items) and has size $2^n - 1$ (excluding the empty set which is not a valid itemset). Although the size of the powerset grows exponentially in the number of items *n* in *I*, efficient search is possible using the downward-closure property of support (also called anti-monotonicity) which guarantees that for a frequent itemset, all its subsets are also frequent and thus for an infrequent itemset, all its supersets must also be infrequent. Exploiting this property, efficient algorithms (e.g., Apriori [2]) can find all frequent itemsets.

Figure 1 gives the sketch of the Apriori algorithm. Apriori uses a "bottom up" approach, breadth-first search and a tree structure to count candidate item sets efficiently. The first pass of the algorithm simply counts item occurrences to determine the large 1-itemsets. A subsequent pass, say pass *k*, consists of two phases. First, the large itemsets $L_{k-1}$ found in the (*k-1*)-th pass are used to generate the candidate itemsets $C_k$. Next the database is scanned and the support of candidates in $C_k$ is counted. For fast counting, we need to efficiently determine the candidates in $C_k$ that are contained in a given transaction *t*. All the missing details of the Apriori algorithm can be found in [1] and [2].

```
1)  $L_1$ = {Large 1-itemsets};
2)  for ( $k$ =2; $L_{k-1} \neq \varnothing$ ; $k$ ++ ) do begin
3)        $C_k$ =apriori-gen( $L_{k-1}$ ); // New candidates
4)        Forall transaction $t \in D$ do begin
5)            $C_t$ =subset( $C_k$ , $t$ ); // Candidates contained in $t$
6)            Forall candidates $c \in C_t$ do
7)                $c$.count++;
8)        end
9)        $L_k$ ={ $c \in C_k$ | $c$.count $\geq$ minsup}
10) end
11) Answer = $\bigcup_k L_k$ ;
```

**Figure 1: The Apriori algorithm**

### 3.2 PART Algorithm

The PART rule learning algorithm [5] combines the two approaches C4.5 [11] and RIPPER [6] in an attempt to avoid their respective problems. Unlike both C4.5 and RIPPER it does not need to perform global optimization to produce accurate rule sets, and this added simplicity is its main advantage. It adopts the separate and conquer strategy in that it builds a rule, removes the instances it covers, and continues creating rules recursively for the remaining instances until none are left. It differs from the standard approach in the way that each rule is created. In essence, to make a single rule a pruned decision tree is built for the current set of instances, the leaf with the largest coverage is made into a rule, and the tree is discarded.

The prospect of repeatedly building decision trees only to discard most of them is not as bizarre as it first seems. Using a pruned tree to obtain a rule instead of building it incrementally by adding conjunctions one at a time avoids the over-pruning problem of the basic separate and conquer rule learner. Using the separate and conquer methodology in conjunction with decision trees adds flexibility and speed. It is indeed wasteful to build a full decision tree just to obtain a single rule, but the process can be accelerated significantly without sacrificing the above advantages.

The key idea is to build a "partial" decision tree instead of a fully explored one. A partial decision tree is an ordinary decision tree that contains branches to undefined subtrees. To generate such a tree, we integrate the construction and pruning operations in order to find a "stable" subtree that can be simplified no further. Once this subtree has been found tree-building ceases and a single rule is read off.

The tree-building algorithm is summarized as follows: it splits a set of examples recursively into a partial tree. The first step chooses a test and divides the examples into subsets accordingly. The PART implementation makes this choice in exactly the same way as C4.5. Then the subsets are expanded in order of their average entropy, starting with the smallest (the reason for this is that subsequent subsets will most likely not end up being expanded, and the subset with low average entropy is more likely to result in a small subtree and therefore produce a more general rule). This continues recursively until a subset is expanded into a leaf, and then continues further by backtracking.

But as soon as an internal node appears which has all its children expanded into leaves, pruning begins: the algorithm checks whether that node is better replaced by a single leaf. This is just the standard "subtree replacement" operation of decision-tree pruning, and the PART implementation makes the decision in exactly the same way as C4.5. If replacement is performed the algorithm backtracks in the standard way, exploring siblings of the newly-replaced node. However, if during backtracking a node is encountered whose every child is not a leaf (and this will happen as soon as a potential subtree replacement is not performed) then the remaining subsets are left unexplored and the corresponding subtrees are left undefined. Due to the recursive structure of the algorithm this event automatically terminates tree generation. Additional details about the PART algorithm can be found in [5].

## 4. EXPERIMENTAL EVALUATION AND RESULTS

In this paper, we used a real-life dataset [13] called "Laptop prices" to illustrate the class association and classification rules learning process. This dataset consists of 1.304 examples which have 11 "independent" attributes each and one class attribute. Three numeric attributes (Inches, Weight, and Price_euros) are discretized into nominal attributes (as illustrated in Table 1).

**Table 1. Discretization of numeric attributes**

| Inches | | Weight | | Price_euros | |
|---|---|---|---|---|---|
| ***Bins*** | ***Name*** | ***Bins*** | ***Name*** | ***Bins*** | ***Name*** |
| [10.1 – 14] | Small | [0.69 – 1.75] | Light | [174 – 800] | Low |
| (14 – 16] | Standard | (1.75 – 2.21] | Medium | (800 – 1300] | Mid-range |
| (16 – 18.4] | Gaming | (2.21 – 4.7] | Heavy | (1300 – 6099] | High |

In Table 2, the best class association rules which have at least 90% confidence are shown according to the Apriori algorithm. Table 3 shows the classification rules which are found by the PART algorithm (Conf. and Cov. In these tables stand for confidence or accuracy and coverage, respectively).

Results show individual class association rules are more accurate (confident) and have a higher coverage compared to individual classification rules. On the other hand, classification rules cover the search space "completely" and do not overlap while class association rules may (heavily) overlap and are not guaranteed to cover all the examples.

**Table 2. Class Association Rules found by the Apriori algorithm**

| Class association rules | Conf. | Cov. |
|---|---|---|
| { Inches=Standard, Cpu=Intel Core i3} ==> { Price_euros=Low } | **100%** | **114** |
| { ProductTypeName=Notebook, Inches=Standard, Cpu=Intel Core i3} ==> { Price_euros=Low } | **100%** | **110** |
| { Inches=Standard, Cpu=Intel Core i3, Ram=4GB } ==> { Price_euros=Low } | **100%** | **89** |
| { ProductTypeName=Notebook, Inches=Standard, Cpu=Intel Core i3, Ram=4GB } ==> { Price_euros=Low } | **100%** | **85** |
| { Inches=Standard, Cpu=Intel Core i3, OpSys=Windows 10} ==> {Price_euros=Low} | **100%** | **84** |
| { Inches=Standard, Cpu=Intel Core i3, Gpu=Intel } ==> {Price_euros=Low} | **100%** | **82** |
| { Inches=Standard, Cpu=Intel Core i3, Memory=HDD } ==> { Price_euros=Low } | **100%** | **76** |
| { ScreenResolution=1366x768, Cpu=Intel Celeron, Gpu=Intel } ==> { Price_euros=Low } | **100%** | **67** |
| { Inches=Standard, Cpu=Intel Core i3, Ram=4GB, Weight=Medium } ==> { Price_euros=Low } | **100%** | **67** |
| { ProductTypeName=Gaming, ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia } ==> { Price_euros=High } | **94%** | **72** |
| { ProductTypeName=Gaming, ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia, OpSys=Windows 10} ==> { Price_euros=High } | **94%** | **72** |
| { Inches=Standard, Ram=4GB, Memory=HDD, Weight=Medium } ==> { Price_euros=Low } | **94%** | **143** |
| { ProductTypeName=Notebook, Ram=4GB, Memory=HDD, Weight=Medium }==> { Price_euros=Low } | **94%** | **142** |
| { ProductTypeName=Gaming, ScreenResolution=Full HD, Cpu=Intel Core i7, Ram=16GB, Memory=SSD + HDD} ==> { Price_euros=High } | **94%** | **71** |
| { ProductTypeName=Gaming, ScreenResolution=Full HD, Cpu=Intel Core i7, Ram=16GB, Memory=SSD + HDD, OpSys=Windows 10} ==> { Price_euros=High } | **94%** | **71** |
| { ScreenResolution=1366x768, Ram=4GB, Weight=Medium } ==> { Price_euros=Low } | **93%** | **120** |
| { ProductTypeName=Notebook, ScreenResolution=1366x768, Ram=4GB, Weight=Medium } ==> { Price_euros=Low } | **93%** | **112** |
| { Inches=Standard, ScreenResolution=1366x768, Ram=4GB, Weight=Medium } ==> { Price_euros=Low } | **93%** | **120** |
| { ProductTypeName=Notebook, Inches=Standard, ScreenResolution=1366x768, Ram=4GB, Weight=Medium} ==> { Price_euros=Low } | **93%** | **120** |
| { ProductTypeName=Gaming, ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD } ==> { Price_euros=High } | **93%** | **75** |
| { ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia }==> {Price_euros=High } | **93%** | **75** |
| { ProductTypeName=Gaming, ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD, OpSys=Windows 10} ==> {Price_euros=High} | **93%** | **75** |
| { ScreenResolution=Full HD, Cpu=Intel Core i7, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia } ==> {Price_euros=High} | **93%** | **74** |
| { ProductTypeName=Gaming, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia, OpSys=Windows 10, Weight=Heavy } ==> {Price_euros=High} | **93%** | **72** |

**Table 3. Classification Rules found by the PART algorithm**

| Classification rules | Conf. | Cov. |
|---|---|---|
| { Ram = 16GB, Memory = SSD, ScreenResolution = Full HD } ==> { Price_euros= High } | 80% | 36 |
| { Ram = 16GB, Memory = SSD } ==> { Price_euros= High } | 94% | 49 |
| { Ram = 16GB, Inches = Gaming } ==> { Price_euros= High } | 94% | 49 |
| { Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Core i3, ScreenResolution = 1366x768 } ==> { Price_euros=Low } | 98% | 52 |
| { Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Core i5, Memory = HDD } ==> {Price_euros=Low} | 74% | 48 |
| { Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Celeron } ==> {Price_euros=Low} | 100% | 48 |
| { Cpu = Intel Core i3, Memory = HDD } ==> { Price_euros=Low } | 98% | 47 |
| { Ram = 16GB } ==> { Price_euros= High } | 70% | 35 |
| { Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Core i5 } ==> { Price_euros= Low } | 49% | 17 |
| { Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Other } ==> { Price_euros= Low } | 100% | 30 |
| { Ram = 4GB, ProductTypeName = Notebook } ==> { Price_euros= Low } | 97% | 58 |
| { Cpu = Intel Core i7, ProductTypeName = Ultrabook, Inches = Standard } ==> { Price_euros= High } | 70% | 31 |
| { Cpu = Intel Core i7, ProductTypeName = Gaming, Inches = Standard } ==> { Price_euros= Mid-Range } | 61% | 33 |
| { Cpu = Intel Core i7, Inches = Standard, Memory = SSD } ==> { Price_euros= Mid-Range } | 51% | 33 |
| { Cpu = Intel Core i7, Inches = Small } ==> { Price_euros=High } | 71% | 41 |
| { Cpu = Intel Celeron } ==> { Price_euros= Low } | 100% | 40 |
| { ProductTypeName = Gaming, Cpu = Intel Core i7} ==> { Price_euros= High } | 76% | 29 |
| { Memory = SSD + HDD } ==> { Price_euros= Mid-Range } | 58% | 42 |
| { Cpu = Intel Core i7} ==> { Price_euros= Mid-Range } | 49% | 23 |
| { Cpu = Intel Core i5, Weight = Light, Company = Dell } ==> { Price_euros= Mid-Range } | 49% | 18 |
| { Cpu = Intel Core i5, Weight = Light, ProductTypeName = Ultrabook } ==> {Price_euros= Mid-Range } | 50% | 21 |
| { Cpu = Intel Core i5, Memory = SSD, Weight = Light } ==> {Price_euros= Mid-Range } | 49% | 30 |
| { Cpu = Intel Core i5, Gpu = Intel } ==> {Price_euros= Mid-Range } | 48% | 38 |

## 5. CONCLUSIONS AND FUTURE WORK

Overall, it can be observed from the experimental results (Apriori algorithm) that the features of the cheaper laptops (rules in Table 2 with the right-hand side *Price_euros=Low*) were inches: Standard or Small, CPU: Intel Core i3 or Celeron, RAM: 4GB, Weight: Light or Medium, Memory: HDD, GPU: Intel, Operative System: Windows 10 and Screen Resolution: 1366x768.

The expensive laptops (rules in Table 2 with the right-hand side *Price_euros=High*) were mostly Gaming laptops with SSD or SSD+HDD memory, 16GB RAM, Intel Core i7 CPU, Nvidia GPU, Heavy weight and Full HD Screen Resolution.

The results on the chosen dataset show that the Apriori algorithm generates more accurate individual classification rules with higher coverage/accuracy than the PART algorithm. However, class association rules may be highly overlapping which can affect the overall accuracy of the classifier.

In future work we shall check the overall accuracy and coverage of "complete" classifiers generated by class association rules algorithm, broaden our comparison to more (diverse) datasets and compare also the learning times of both algorithms.

## 6. REFERENCES

[1] Agrawal, R., Amielinski, T., and Swami, A. (1993). "Mining association rule between sets of items in large databases". In Proceeding of the ACM SIGMOD International Conference on Management of Data, pp. 207-216.

[2] Agrawal, R., and Srikant, R. (1994). "Fast algorithms for mining association rules". Research Report CA 95120, IBM Almaden Research Center, San Jose, California.

[3] Cendrowska, J. (1987). "PRISM: An algorithm for inducing modular rules". International Journal of Man Machine Studies. volume 27, no. 4, pp. 349-370.

[4] Duda, R., and Hart, P. (1973). "Pattern Classification and Scene Analysis". John Wiley & Sons.

[5] Frank, E., and Witten, I. (1998). "Generating accurate rule sets without global optimization". Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann, San Francisco, pp. 144-151.

[6] Fürnkranz, J. (1996). "Separate-and-conquer rule learning". Technical Report TR-96-25, Austrian Research Institute for Artificial Intelligence, Vienna.

[7] Fürnkranz, J., and Widmer, G. (1994). "Incremental reduced error pruning". Proceedings of the 11th Annual Conference on Machine Learning, Morgan Kaufmann, pp. 70-77.

[8] John, G.H., and Langley, P. (1995). "Estimating Continuous Distributions in Bayesian Classifiers". Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Mateo. pp. 338-345.

[9] Liu, B., Hsu, W., and Ma, Y. (1998). "Integrating Classification and Association Rule Mining". Singapore.

[10] Patel, R., Vala, J., and Patel, K. (2014). "Comparative Study on Associative Classification techniques".

[11] Quinlan, J. R. (1993). "C4.5: Programs for Machine Learning". San Mateo, Morgan Kaufmann.

[12] Quinlan, J. R. (2008). "Data mining tools see5 and c5".

[13] https://www.kaggle.com/ionaskel/laptop-prices/home (accessed on 17.8.2018).

[14] http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab 8-Apriori (accessed on 17.8.2018).

# Real-time visualization of 3D digital Earth

Aljaž Jeromel
Faculty of Electrical Engineering and Computer Science, University of Maribor
Koroška cesta 46, 2000 Maribor, Slovenia
aljaz.jeromel@student.um.si

## ABSTRACT

A new method for real-time visualization of Earth in 3D is presented. The method dynamically calculates visualization data and works in several steps. Firstly, the vertex coordinates in 3D are calculated. Then, the UV coordinates for texturing are calculated from vertex' position. The vertices are connected into triangles and sent to the shader pipeline, where they are processed by vertex shader, geometry shader and fragment shader. The vertex shader transforms the vertices from the world space to screen space. Geometry shader corrects possible interpolation errors by inserting extra vertices where needed, while the fragment shader assigns the colour to resulting fragments. The main advantage of the proposed method over the state of the art methods is its simplicity. Experimental results have shown that the proposed method produces a high FPS (frames per second), which makes it suitable for real-time visualization.

## Keywords

Computer Graphics, 3D Rendering, Object Geometry

## 1. INTRODUCTION

With the use of 3-dimensional (3D) graphics, a lot of things can be visualized, from simple cubes to complex models of real-life objects. The need for an application that visualizes geographical and other data about the Earth in 3D has been expressed as early as 1998 by then vice president of the US Al Gore [1]. He proposed many practical uses for such an application, named "Digital Earth". The proposed uses include education, national border negotiations, redeployment of police force to the most problematic areas, biodiversity preservation, climate change prediction and improvement of agricultural productivity. Because most world maps use the Mercator projection, which deforms the view of the world, the visualization of Earth in 3D can teach the children a lot more about our planet than the conventional 2D maps can. The digital model of Earth, combined with many kinds of data (e.g. historical, geodetic, etc.) could also lead to better understanding of the planet [2].

Quite a few methods for visualization of Earth and planets have been developed and optimized since the historical Gore speech. During the years 2003 to 2006, a group of Italian computer scientists presented a method for high performance terrain visualization, named BDAM, and two of its improvements, PBDAM and CBDAM [3, 4, 5]. Schneider and Westermann have also proposed a method using nested meshes for high quality terrain rendering with minimal GPU overhead in 2006 [6]. In 2007, Schafhitzel et al. proposed a method for rendering of the planets, including a simulation of the atmosphere, in real-time [7]. A short book on planet visualization was written in 2008 by Östergaard, which discusses the implementation challenges of a digital globe and presents the implementation solution with some optimizations [8]. Cozzi and Ring took that achievement one step further in 2011, by writing an exhaustive book on virtual globe design, along with multiple propositions on how to deal with implementation issues that can occur, and the full C# implementation [9].

This paper is focused on real-time rendering of 3D Earth map. A method for a dynamic visualization is presented and explained, and the efficiency of the method is discussed. The proposed method firstly calculates visualization vertices on the screen, then calculates their UV coordinates. The results of these steps fill the Vertex Buffer Object, which is sent to the graphics card. There, the calculated vertices are processed by the vertex shader, geometry shader, and the fragment shader programs, before the map is drawn on the screen. According to experimental results, the proposed method produces high enough FPS (frames per second) to be used in high performance applications. The proposed method is much simpler and light-weight than the state of the art visualization methods. The floating point precision errors are handled by means of normalizing the coordinates to the visible part of the Earth, while the interpolation errors are processed by the geometry shader on the GPU. This paper also does not concern itself with acquisition of texture data; the implementation behind it is out of scope of this paper.

This paper is organized as follows. The method for visualization is explained in Section 2. The results, produced by the method, namely frames per second (FPS) in relation to the number of triangles rendered, are presented in Section 3. The conclusion is given in Section 4.

## 2. METHODOLOGY

The rendering procedure, presented in this paper, consists of following steps. Firstly, the visible tiles are calculated. Then, the vertices and their UV coordinates are calculated. The vertex data is sent to the graphics card for drawing. Finally, the vertex, geometry, and fragment shader finalize the calculations, and the result is drawn to the screen. These steps are presented in more detail in the following subsections.

### 2.1 Vertex generation

The first step in 3D graphics pipeline is the generation of vertices. In many applications that use 3D graphics, some modelling software is used to generate the model, already complete with vertices, UV coordinates, and normals of the object. However, such models are not suitable for high-resolution realistic display applications because of three reasons:

- A model file with sufficient number of vertices for realistic display of the highest quality would be extremely large,
- Rendering of too many vertices results in a very low FPS (frames per second) environment when using mediocre graphics cards, due to insufficient available graphics memory or shader processing cores,
- Older versions of shader programs don't support the use of 64-bit precision, which is required for rendering of the most detailed views of the Earth.

Because our goal was compatibility with some of the older versions of shading language, 64-bit precision could not be used in shader programs. Therefore, in presented application, dynamic vertex generation was used, which also meant that the precision-critical data could not be processed by shader programs on the GPU. Because of that, the FPS (frames per second) in our application is a bit lower than it could have been in an ideal setting.
For the vertex generation, a ray casting algorithm is used. The screen is divided in up to 121 equally spaced points, and from each of those points, a ray is sent out. This produces a grid on the screen with up to 10 rows and columns. The rays' intersections with sphere representing the Earth are calculated. If we have a ray $\vec{r} = (rx, ry, rz)$, sphere with radius $R$, the camera positioned at $\vec{c} = (cx, cy, cz)$, and the dot product between ray direction and camera position vectors $dt$, the closest intersection, $\vec{p}$, between ray and the sphere can be calculated by Equation **??**.

$$\vec{p} = \vec{c} - (dt + \sqrt{dt^2 + R^2 - |\vec{c}|^2})\ \vec{r}. \tag{1}$$

Note that the part under the square root is negative when the ray and the sphere do not intersect. In that case, the ray's direction is adjusted a bit, so that it touches the sphere. The tangent point, $\vec{t}$, is calculated using Equation **??**, with intermediate vectors $\vec{a}$ (Equation **??**) and $\vec{b}$ (Equation **??**)

$$\vec{a} = \vec{c} + \frac{|\vec{c}|^2}{-dt}\ \vec{r} \tag{2}$$

$$\vec{b} = \frac{\vec{a}}{|\vec{a}|}|\vec{c}|\frac{R}{\sqrt{|\vec{c}|^2 - R^2}} \tag{3}$$

$$\vec{t} = \vec{c} + \sqrt{|\vec{c}|^2 - R^2}\,\frac{\vec{b} - \vec{c}}{|\vec{b} - \vec{c}|} \tag{4}$$

Finally, the coordinates of the points are normalized.

### 2.2 UV coordinate calculation

The UV coordinates for texturing are calculated from the vertex' coordinates. If the map tiles use Mercator projection, which distorts the image in the direction of vertical axis (V coordinate), that has to be reversed first [10]. The U coordinate is calculated simply from the normalized vertex position, $p = (x, y, z)$, as in Equation **??**.

$$U = \frac{\text{arc tan}(\frac{x}{z})}{2\pi} + 0.5, \tag{5}$$

With the help of constant M (Equation 7), the V coordinate is then calculated using Equation **??**.

$$V = \frac{\text{arc sinh}[\tan(2\ M\ \text{arc sin}[y])]}{2\pi} + 0.5, \tag{6}$$

$$M = \text{arc tan}[\sinh(\pi)]. \tag{7}$$

Because shader programs use 32-bit precision, an additional step is required when calculating the UV coordinates. When rendering a close-up view, only a really small part of the Earth is visible, which means the difference between minimum and maximum UV coordinates is really small too. Because of that, the UV coordinates can be normalized to the visible part of the Earth. The minimum and maximum visible row and column of tiles were already calculated before vertex generation step. Instead of having the V value 0 at the south pole and 1 at the north pole, we can adjust it to be 0 at the bottom of the bottommost visible row and 1 to be at the top of the topmost visible row. The U coordinate is also mapped from 0 at the left of leftmost visible column and 1 at the right of rightmost visible column. That way, the precision errors in shader programs are kept in check when rendering the high resolution close-up views of the Earth.

### 2.3 Vertex shader

The vertex shader program's only task is to transform the vertex position from the world space into screen space by multiplying with the standard model-view-projection matrix. In case the model vertex buffer is used instead of the dynamically calculated vertex buffer, the vertex shader also remaps the UV coordinates to adapt them to the Mercator projection, using Equation **??** and the constant M from Equation 7.

$$V' = 0.5 - \frac{\text{arc sinh}[\tan(2M[V - 0.5])]}{2\pi}. \qquad (8)$$

## 2.4 Geometry shader

Geometry shader program has one of the most important tasks in this whole process. It handles the interpolation errors that happen at the edges of the map. Because OpenGL fragment interpolation is always linear, errors occur when one of the points in the triangle has the U coordinate of 0.1 and another point in the same triangle has the U coordinate of 0.9. In that case, the shader should interpolate from 0.1 down to 0, then jump to 1 and from there continue interpolating down to 0.9. The models circumvent this issue by duplicating vertices at the UV coordinate wrap-arounds, so if we're using a model, the geometry shader can be skipped. The geometry shader program takes in the set of vertices representing a primitive, in our case a triangle. It checks the U coordinate of all 3 vertices, and if the difference between any 2 of them is higher than 0.7, it inserts another vertex between them, to split the triangle into 2 or more triangles, setting the inserted point's U coordinate to 0 for one triangle and 1 for the other. It then outputs all of the resulting triangles to the following stages of the rendering pipeline.

## 2.5 Fragment shader

The fragment shader program's task is to determine the colour of each fragment. OpenGL allows the shader program to access up to 32 different textures during each draw call. But because a lot of textures converge at the north and south pole, multiple draw calls are needed during each frame. Each of those calls is passed one of the tile textures. The fragment shader program is responsible for mapping of UV coordinates to the texture, and discarding all of the fragments that do not sample from the texture, passed in the current call. If the fragment is not discarded in this draw call, the fragment shader program samples the passed texture to obtain the fragment's colour, which is finally output to the depth testing.

## 3. RESULTS

In this section, the results of rendering the Earth with the proposed method are presented. The average time needed to process and render a frame was measured, and the average FPS (Frames per second) was calculated. The measurements were done on a personal computer with the following configuration: Intel Core i5-3570K CPU 3.40 GHz, 32 GB of RAM, GeForce GTX 1060 Windforce OC 6GB, running 64 bit operating system Windows 10 Education. The width and height of all the texture units used were 256 pixels. Their total size on disk was 9560 terabytes. The result of rendering three different levels of detail (LOD) of the same area is shown in Figure 1.

During standard rendering, the screen was divided into a grid of $10 \times 10$ rectangles, each of which was split into two triangles. FPS calculation was done by lowering the grid's dimensions all the way down to 1 rectangle, and measuring the average time required to render a frame. Using that

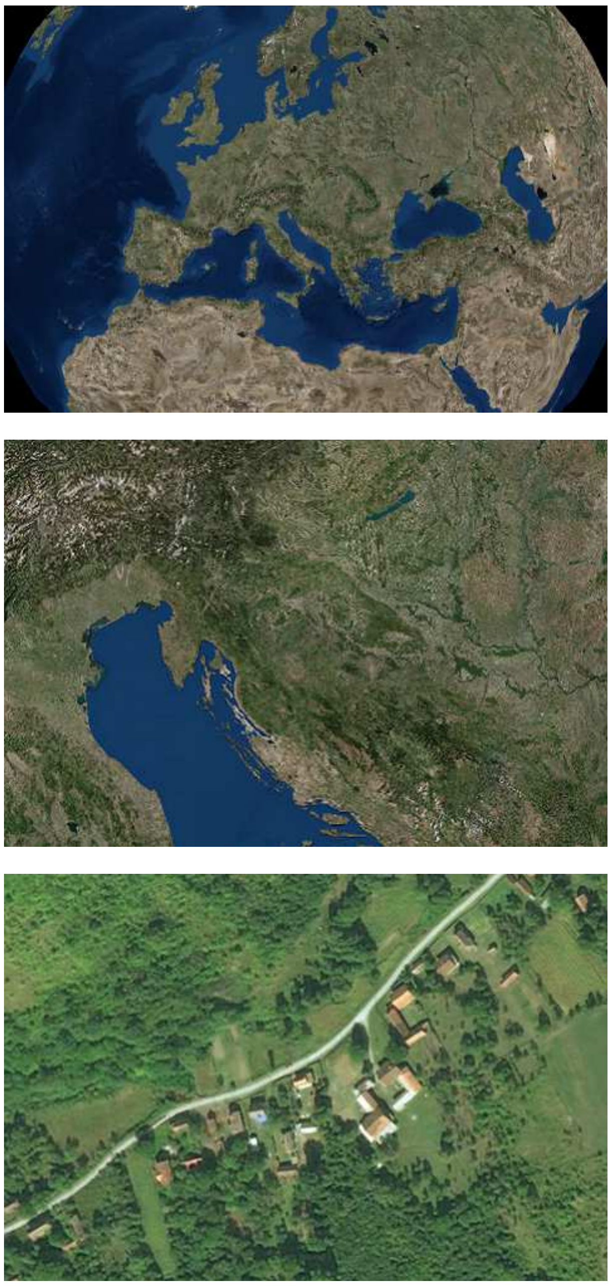


**Figure 1: The result of rendering the same area in three different LOD-s**

information, the average FPS was calculated. The results are shown in Table 1.

**Table 1: Average FPS when rendering a pre-set number of triangles**

| Number of triangles | Average FPS |
|---|---|
| 2 | 452.90 |
| 8 | 350.62 |
| 18 | 397.89 |
| 32 | 407.09 |
| 50 | 405.80 |
| 72 | 364.40 |
| 98 | 356.11 |
| 128 | 370.66 |
| 162 | 382.86 |
| 200 | 359.95 |

As seen from the table, though the results vary, the average FPS generally tends to decrease with a higher number of triangles. This can be seen in Figure 2, where the dotted line represents the 6th order of the polynomial trending line.

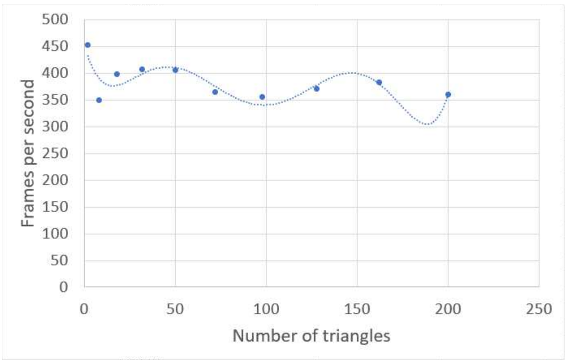


**Figure 2: Graph representing the average FPS in relation to the number of rendered triangles, with the 6th order polynomial trending line (dotted)**

## 4. CONCLUSION

In this paper, a method for real-time visualization of Earth in 3D is presented. It works by dynamically calculating the visualization data and uploading it to the graphics card, where it is processed by the vertex shader, geometry shader, and fragment shader. The experimental results have shown that the method produces a very high FPS (frames per second), which allows for the usage in applications, where high performance is important.

## 5. REFERENCES

[1] A. Gore. The digital earth: Understanding our planet in the 21st century. *Australian Surveyor*, 43(2):89–91, June 1998.

[2] M. F. Goodchild. The use cases of digital earth. *International Journal of Digital Earth*, 1(1):31–42, March 2008.

[3] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, R. Scopigno. Bdam - batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(2):505–514, November 2003.

[4] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, R. Scopigno. Planet-sized batched dynamic adaptive meshes (p-bdam). In *Proceedings of IEEE Visualization*, pages 149–155. IEEE, October 2003.

[5] E. Gobbetti, F. Marton, P. Cignoni, M. Di Benedetto, F. Ganovelli. C-bdam - compressed batched dynamic adaptive meshes for terrain rendering. *Computer Graphics Forum*, 25(3):333–342, December 2006.

[6] J. Schneider and R. Westermann. Gpu-friendly high-quality terrain rendering. *Journal of WSCG*, 14(1-3):49–56, February 2006.

[7] T. Schafhitzel, M. Falk, T. Ertl. Real-time rendering of planets with atmospheres. *Journal of WSCG*, 15(1-3):91–98, January 2007.

[8] J. Östergaard. *Planet Rendering Using Online High-Resolution Datasets*. Linköping University, Norrköping, Sweden, 2008.

[9] P. Cozzi and K. Ring. *3D Engine Design for Virtual Globes*. A K Peters, Ltd., Natticks, Massachusetts, USA, 2011.

[10] D. H. Maling. *Coordinate Systems and Map Projections, Second Edition*. Pergamon Press, Oxford, England, 1992.

# Towards an enhanced inertial sensor-based step length estimation model

Melanija Vezočnik
University of Ljubljana, Faculty of Computer and Information Science
Ljubljana, Slovenia
mv6082@student.uni-lj.si

Matjaž Branko Jurič
University of Ljubljana, Faculty of Computer and Information Science
Ljubljana, Slovenia
matjaz.juric@fri.uni-lj.si

## ABSTRACT

Inertial sensors found in Internet-of-Things devices such as smartphones and smartwatches can be used to track user's activity as well as step length. In particular, rapidly growing interest in pedestrian dead reckoning – indoor positioning approach that calculates current position as the sum of previous position and vector with step length and heading information – prompts to develop refined step length estimation models to overcome inaccuracy in determining step length. In this work we present our research towards an enhanced inertial sensor-based step length estimation model. For this purpose, we extend an existing step-frequency-based model with acceleration and make it less user-specific. We evaluated the performance of the proposed model for one sensor position and different walking speeds and obtained very promising results comparable to related models. We will further improve the accuracy of the proposed model and modify it, so it will require less time for tuning, and thoroughly address the pre-processing of the sensor data. We will also test the proposed model for different sensor positions and pedestrian-dead-reckoning-based indoor positioning.

## Keywords

inertial sensors, pedestrian dead reckoning, step length estimation

## 1. INTRODUCTION

Over the past few years, user engagement enabling Internet-of-Things (IoT) technologies are emerging. In particular, inertial sensors can be utilized to track user's activity as well as step length since they can be found in smartphones, smartwatches and other IoT devices. They also exhibit advantages such as easy accessibility, inexpensiveness and small-size. Moreover, estimating step length is also applicable in diverse areas, especially in indoor positioning approach called pedestrian dead reckoning [1, 2, 3, 4] that determines the current position by adding vector with step length and heading information to the previous position.

Throughout this work we address inertial sensor-based step length estimation models. According to [5], they can be classified into four categories based on the parameters they require as an input. These parameters are obtained from inertial sensor measurements. The categories are: step-frequency-based [1, 2, 6], acceleration-based [3, 7], angle-based [8] and multiparameter [4, 9]. Step-frequency-based models require step frequency as an input, acceleration-based models require acceleration, angle-based models require the opening angle of the leg and the input of multiparameter models combines two or all three of the previously listed parameters.

In this work we present our research towards an enhanced inertial sensor-based step length estimation model that increases the accuracy of estimated step length. Section 2 overviews step length estimation models. Section 3 describes the derivation and Section 4 the evaluation of the proposed model. Section 5 presents results, whereas Section 6 discusses them. Finally, Section 7 concludes this work presenting future research directions.

## 2. OVERVIEW OF STEP LENGTH ESTIMATION MODELS

Step length is determined in several phases. In order to acquire inertial sensor measurements, a sensor has to be set-up on user's body, most commonly in hand, pocket or centre of body mass. Acquired data – usually accelerometer measurements – have to be pre-processed to eliminate errors. Steps are detected before step length is determined using a step length estimation model that often has to be tuned. Next, we briefly present the categories of the models (step-frequency-based, acceleration-based, angle-based and multiparameter).

### 2.1 Step-frequency-based models

Step-frequency-based models require step frequency as an input. They commonly exploit linear relationship between step length and step frequency and have to be tuned prior to utilization. The following two models include user's height. Renaudin et al. [6] proposed a model based on linear relationship between step length and step frequency, whereas Tian et al. [2] exploited the link between square root of step frequency and step length. On the contrary, Zhang et al. [1] included user's leg length in their model, but based it on the linear relationship between step length and step frequency, similarly as the model proposed by Renaudin et al. [6].

### 2.2 Acceleration-based models

Acceleration-based models require acceleration as an input. They often exploit the link between step length and acceleration properties such as minimum and maximum acceleration values within the step. Weinberg [7] proposed such model. It calculated step length as the product of a tuneable constant and fourth principal root of the difference between maximum and minimum vertical acceleration values within a step. Similarly, Do et al. [3] based their model on vertical displacement of the centre of body mass. This model includes user's leg length, but it does not include any tuneable constants.

### 2.3 Angle-based models

Angle-based models require the opening angle of the leg as an input. They often exploit linear relationship between step length and the opening angle of the leg. Diaz and Gonzales [8] proposed such model. It includes two tuneable constants.

### 2.4 Multiparameter models

The input of multiparameter models combines two or all three of the previously listed parameters: step frequency, acceleration or the opening angle of the leg. These models usually extend existing models with additional parameters. For example, Mikov et al. [4] included the inverse of step frequency in the Weinberg model [7]. Similarly, Bylemans et al. [9] also based their model on the difference between maximum and minimum acceleration values within a step, but they included step frequency and absolute average acceleration value in walking direction in the model.

## 3. THE DERIVATION OF THE PROPOSED MODEL

Before we started deriving a new model, we carried out an in-depth analysis and comparison of 13 relevant representative inertial sensor-based step length estimation models [5], where we studied the influence of four typical sensor positions and different walking speeds on their performance. We tuned the models with personalized and universal sets of constants. We also established a benchmark repository for the performance evaluation of inertial sensor-based step length estimation models that includes more than 22 km of gait measurements obtained from 15 adults using off-the-shelf smartphone. This repository is openly available at: `https://github.com/repositoryadmin/SLERepository`.

In [5], we obtained the best overall evaluation results for universal sets of constants for the model proposed by Tian et al. [2] and therefore started deriving our enhanced model from this model. The model proposed by Tian et al. [2] calculates step length as:

$$SL = K \cdot \sqrt{F} \cdot h, \qquad (1)$$

where $SL$ represents estimated step length, $K$ tuneable constant, $h$ user's height and $F$ step frequency.

We observed that average walking speed during tests affected the values of tuneable constant in the model as shown in Figure 1. We can see that on average the values of the constant increase with the increased walking speed. We therefore include mean absolute acceleration in walking direction in our model. To make the proposed model less user-specific, we exclude user's height and calculate step length as:

$$SL = K_1 \cdot \sqrt{F} \cdot a_{mean}^{K_2}, \qquad (2)$$

where $SL$ represents estimated step length, $K_1$ and $K_2$ are tuneable constants, $a_{mean}$ mean absolute acceleration value in walking direction and $F$ step frequency.

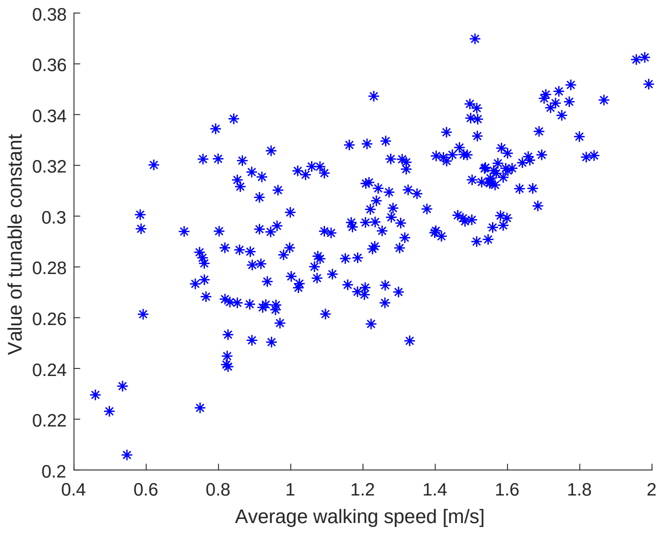


**Figure 1: Values of tuneable constants in the model proposed by Tian et al. [2] for different walking speeds.**

## 4. EVALUATION

For the evaluation of the proposed model, we used data from the repository described in the previous section that includes measurements acquired in two independent sets of field tests on two straight test paths: 15- and 108-m-long. We used the data collected on the shorter path to tune the models and the data collected on the longer path to evaluate the performance of the models.

To eliminate the impact of smartphone orientation, we only considered hand-reading position as shown in Figure 2, where the user is carrying the smartphone in hand. Smartphone's local coordinate system is considered as follows: $y$-axis is pointing in the walking direction and $z$-axis is pointing in the opposite direction of the floor. Smartphone's position is fixed. Three steady walking speeds were tested: slow, normal and fast. Participants self-selected walking speeds to their preference, but they were asked to maintain the walking speed as similar as possible for the time of the experiment. One person always monitored the execution of the experiments and counted the number of participants' steps. Despite the fact that persons self-selected slow, normal and fast walking speeds to their preference, we observed that average walking speeds ranged from 0.45 m/s to 1.25 m/s for slow walking speed, 0.90 m/s to 1.65 m/s for normal walking speed and 1.40 m/s to 2.00 m/s for fast walking speed [5].

Measurements were acquired using off-the-shelf Samsung Galaxy S7 edge smartphone. Sampling frequency was 100 Hz with standard deviation of 8 Hz. Therefore, acquired measurements were resampled to 100 Hz by employing linear interpolation. We used peak detection algorithm for step detection and determined personalized constants of the proposed model and the models selected for the comparison using optimization analysis on the data collected on the shorter path. We employed personalized sets of constants to evaluate the performance of the models on the data collected on the longer path. For every test, we have calculated the

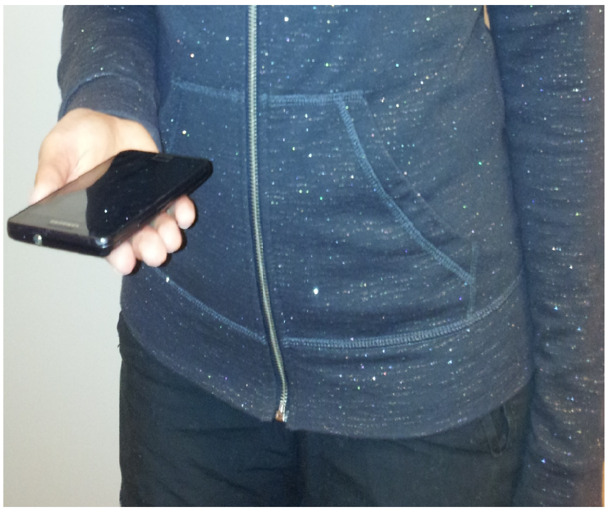


**Figure 2: Hand-reading position.**

performance as:

$$e = \frac{|d - d^*|}{d^*} \cdot 100\% \quad (3)$$

where $e$ is error, $d$ is the estimated distance of walked path and $d^*$ the real distance of walked path.

## 5. RESULTS

Table 1 shows the performance of the models. It includes mean errors and standard deviations of the models with respect to total walked distance. Mean errors range from 3.76 % to 16.84 % and standard deviations from 2.41 % to 7.56 %.

**Table 1: The performance of the models.**

| **Models** | **Mean errors [%]** | **Standard deviations [%]** |
|---|---|---|
| Tian et al. [2] | 4.35 | 2.78 |
| Zhang et al. [1] | 8.91 | 6.86 |
| Renaudin et al. [6] | 8.91 | 6.86 |
| Do et al. [3] | 16.84 | 7.56 |
| Weinberg [7] | 4.36 | 3.80 |
| Diaz and Gonzales [8] | 5.25 | 4.54 |
| Mikov et al. [4] | 7.44 | 5.93 |
| Bylemans et al. [9] | 5.68 | 4.61 |
| Proposed model | 3.76 | 2.41 |

## 6. DISCUSSION

Overall, the proposed model slightly outperformed all the other models. Quite similar performance achieved the models proposed by the Tian et al. [2], Weinberg [7], Diaz and Gonzales [8], Bylemans et al. [9] and Mikov et al. [4]. Even though the model proposed by Mikov et al. [4] extends the Weinberg model [7] it did not perform better on average. The models proposed by Zhang et al. [1] and Renaudin et al. [6] performed similarly since they can be rewritten to the same model by employing substitution [5]. The model proposed by Do et al. [3] expectedly performed the worst since it does not contain any tuneable constants.

## 7. CONCLUSION

In this work we presented our research towards an enhanced inertial sensor-based step length estimation model. We based our model on the model proposed by Tian et al. [2] since we obtained the best evaluation results for it for universal sets of constants [5].

We eliminated user's height from the model and included mean absolute acceleration in walking direction instead. We tested the proposed model for one sensor position in hand and obtained promising evaluation results comparable to related models.

We will further improve the accuracy of the proposed model and test it for more sensor positions. We also plan to modify the proposed model, so it would require less time for tuning and thoroughly address the pre-processing of the sensor data. Moreover, we will conduct additional experiments using different IoT devices and test the proposed model for pedestrian-dead-reckoning-based indoor positioning.

## Acknowledgments

M. V. received funding for doctoral studies from the University of Ljubljana – 2016 generation. Authors would like to thank all the volunteers who participated in the experiment.

## 8. REFERENCES

[1] P. Zhang, X. Chen, X. Ma, Y. Wu, H. Jiang, D. Fang, Z. Tang, and Y. Ma, "SmartMTra: Robust Indoor Trajectory Tracing Using Smartphones," *IEEE Sensors Journal*, vol. 17, no. 12, pp. 3613–3624, 2017.

[2] Q. Tian, Z. Salcic, K. I. K. Wang, and Y. Pan, "A Multi-Mode Dead Reckoning System for Pedestrian Tracking Using Smartphones," *IEEE Sensors Journal*, vol. 16, no. 7, pp. 2079–2093, 2016.

[3] T. N. Do, R. Liu, C. Yuen, M. Zhang, and U. X. Tan, "Personal Dead Reckoning Using IMU Mounted on Upper Torso and Inverted Pendulum Model," *IEEE Sensors Journal*, vol. 16, no. 21, pp. 7600–7608, 2016.

[4] A. Mikov, A. Moschevikin, A. Fedorov, and A. Sikora, "A localization system using inertial measurement units from wireless commercial hand-held devices," in *2013 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2013, pp. 1–7.

[5] M. Vezočnik and M. B. Jurič, "Inertial Sensor-Based Step Length Estimation Models: A Review," *In Review Process*, 2018.

[6] V. Renaudin, M. Susi, and G. Lachapelle, "Step Length Estimation Using Handheld Inertial Sensors," *Sensors*, vol. 12, no. 7, pp. 8507–8525, 2012.

[7] H. Weinberg, "Using the ADXL202 in pedometer and personal navigation applications," *Analog Devices AN-602 application note*, vol. 2, no. 2, pp. 1–6, 2002.

[8] E. M. Diaz and A. L. M. Gonzalez, "Step detector and step length estimator for an inertial pocket navigation system," in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2014, pp. 105–110.

[9] I. Bylemans, M. Weyn, and M. Klepal, “Mobile phone-based displacement estimation for opportunistic localisation systems,” in *Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 2009. UBICOMM’09.*, 2009, pp. 113–118.

# Breast Histopathology Image Clustering using Cuckoo Search Algorithm

Krishna Gopal Dhal[1], Iztok Fister Jr.[2], Arunita Das[3], Swarnajit Ray[4], Sanjoy Das[5]

[1] Dept. of Computer Science and Application, Midnapore College (Autonomous), Paschim Medinipur, West Bengal, India. *Email: krishnagopal.dhal@midnaporecollege.ac.in*
[2]Faculty of Electrical Engg. and Computer Sc., University of Maribor, Slovenia, *Email: iztok.fister1@um.si*.
[3]Dept. of Information Technology, Kalyani Govt. Engineering College, Kalyani, Nadia, India.
*Email: arunita17@gmail.com*.
[4]Skybound Digital LLC, Kolkata, West Bengal, India. *Email: swarnajit32@gmail.com*.
[5]Dept. of Engg. and Technological Studies,University of Kalyani, Kalyani, India, *Email: dassanjoy0810@hotmail.com*

## ABSTRACT

Breast histopathological image segmentation is exigent due to the existence of imperceptibly correlated and indistinct multiple regions of concern. Clustering based segmentation is one of the most significant approaches to perform proper segmentation of such complex images. K-means is the well-known clustering techniques but very sensitive to initial cluster centers and easy convergences to local optima. Therefore, researchers are employing Nature-Inspired Optimization Algorithms (NIOA) in this domain. This study develops Cuckoo Search (CS) algorithm based image clustering model for the proper segmentation of breast histopathology images. Experimental results show that CS provides better-quality segmented images compare to classical K-means algorithm by considering the computational time, fitness values and the values of quality parameters.

## KEYWORDS

Clustering, K-means, Image Segmentation, Optimization, Swarm intelligence, Histopathology image.

## 1 Introduction

Breast Cancer is the mainly widespread kind of cancer in women worldwide [1]. Present breast cancer clinical practice and treatment mostly depend on the evaluation of the disease's diagnosis. Bloom-Richardson grading system [2] describes the scoring of three morphological features of the dubious tissue, which are fraction of tubule construction, amount of nuclear pleo-morphism, and mitotic cell count. However, the scoring had been performed by pathologists based on the visual assessment of the tissue's biopsy sample under the microscope [3]. Therefore, researchers concentrate and suggest the use of image analysis methods to mitigate the said issue [4]. Digital histopathology and microscopy images carry out an important role in decision making in disease diagnosis, as they could provide wide information for computer-aided diagnosis (CAD), which facilitates quantitative analysis of digital images with a high throughput processing rate [5, 6]. At present, analysis of digital histopathology through image processing significantly assists pathologists and has attracted many attentions in both research and clinical practice.

A critical requirement in computer-aided diagnosis is segmentation, which is typically measured as the basis of automated image analysis. It provides assistances for various quantitative analyses such as shape, size, texture, and other imagenomics [5, 6]. However, it is difficult to achieve stout and perfect pathological image segmentation as these images frequently reveal background clutter with many noises, artifacts such as blurred regions introduced during image acquisition, and poor contrast between the foreground and the background. Second, there exist significant variations on cell size, shape, and intracellular intensity heterogeneity [5, 6].

In this study, proper segmentation of breast histopathology images is the main aim. Many efforts have been performed to attain automated segmentation of breast histopathology images which includes thresholding [7, 8], watershed method [9, 10], Active Contour model [8, 11], edge based approach [14], neural network [15, 16] etc. A Particle Swarm Optimization (PSO) with Otsu criterion based multi-level thresholding technique was proposed by Jothi and Rajam [7] to automatically segment the nuclei from hematoxylin and eosin (H&E) – stained breast histopathology images. To remove noise, the input image filtered by 3x3 gaussian filter. Experimental result proved that this method automatically segmented the nuclei effectively. A Localized Active Contour Model (LACM) in conjunction with an automatic technique for optimal curve placement with Krill Herd Algorithm (KHA) was developed by Beevi et. al. [8] for the segmentation of nuclei from breast histopathology images. Based on Hausdorff (HD) and Maximum Address Distance (MAD) measures segmentation performance was investigated. The proposed segmentation approach provided superior results compared to GA, Bacterial Foraging Algorithm (BFA), Harmony Search (HS) algorithm and FCM clustering method by considering the convergence rate, objective function values, computational time, sensitivity, precision and F-score. Shen et. al. [9] developed one improved watershed algorithm by incorporating opening-closing reconstruction and the distance transform with chamfer algorithm after color deconvolution, and H-minima. It was claimed that the proposed segmentation model accurately detect the nuclei and overcome the limitation of classical watershed algorithm like over-segmentation due to the sensitivity to noise.

Another novel approach is to segment the nuclei regions and then resolve the overlapping or clump nuclei separation through heuristic approaches like the Concave Point Detection [11]. Wienert et. al. [11] presented novel contour-based "minimum-model" cell detection and segmentation approach that used minimal a priori information and detects contours independent of their shape. Experimental results proved that the proposed segmentation model capable to avoid the segmentation bias with respect to shape features and allows for an accurate segmentation with high precision (0.908) and recall (0.859).

A few researchers also presented different voting algorithms, which cast votes along gradient directions amplifying votes inside the centre of nuclei thereby locating the seed points as ones having maximum votes [12, 13, 14]. Consequently, the detected nuclei seed points had been either utilized to initialize active contours [12, 13] or an edge grouping algorithm [14]. Recently, deep neural network proved its effective performance in breast histopathology image segmentation field [15, 16]. Su et. al. [15] employed one fast Deep convolutional neural network (CNN) for pixel-wise region segmentation of breast histopathology images. Experimental results proved that the proposed segmentation model gave superior performance over both the LBP feature-based and texton-based pixel-wise methods within less computational time. Naylor et. al. [16] developed one hybrid nuclei segmentation model by combining deep learning and mathematical morphology. Test results showed the promising performance of the proposed model.

Although, clustering based segmentation which shows its effective performance in hematopathology [17] or other histopathology [18] image segmentation domain, but have not been used in breast histopathology image field according to best of the knowledge. Therefore, this study concentrates to apply the clustering based segmentation to segmentize the different regions of the breast histopathology images. K-means is the well-known clustering techniques but sensitive to initial cluster centres and easy convergences to local optimization. Therefore, Nature-Inspired Optimization Algorithms (NIOA) are successfully employed to overcome the problems of K-means in image clustering domain [19-23]. For example, Orman et. al. [20] developed Particle Swarm Optimization (PSO) based satellite and MRI image clustering model and claimed that it outperformed some state-of-the-art methods for image classifier such as K-means, Fuzzy C-means, K-Harmonic means and Genetic Algorithm based model. In this study, Cuckoo Search (CS) algorithm has been employed and compared with classical K-means algorithms. Experimental results show that CS provides better-quality segmented images compare to classical K-means by considering the values of objective (fitness) function, computational time and quality parameters.

The paper is organized as follows: section 2 demonstrates the problem formulation and brief implementation of CS algorithm. Section 3 describes the experimental results and the paper is concluded in section 4.

## 2. Image Clustering using Cuckoo Search (CS) Algorithm

Clustering is a process of organizing data into clusters that have high intra-cluster and low inter-cluster similarity. It is clear that intra-cluster similarity should be maximized and inter-cluster similarity should be minimized. Based on this idea, objective functions are defined [24]. The best partitioning of a given data set can be attained by minimizing/maximizing one or more objective functions. The objective functions can be formed by capturing a certain statistical–mathematical relationship among the individual data items and the candidate set of representatives of each cluster (also known as cluster centroids) [25].

### 2.1 Problem Formulation

Suppose one specific dataset consists of $C$ classes (i.e. $C_1$, $C_2, \dots, C_c$) and $N$ features. Therefore, the clustering problem is the finding of the optimal position of $C$ centroids in an $N$-dimensional space i.e. each centroid is an $N$-dimensional vector. With this premises, the $i^{th}$ individual or solution of the applied optimization algorithm is a vector with $N.C$ components which can be denoted as follows [19, 26]:

$$X_i = \left(X_i^1, X_i^2, \dots \dots \dots . X_i^C\right) \quad (1)$$

Where, $X_i^j = \left(x_{1,i}^j, x_{2,i}^j, \dots \dots \dots . x_{N,i}^j\right)$

So any solution in the population of the employed optimization algorithm consists of $N.C$ components, each of which can take any real value.

The fitness function $Fit$ has been calculated by summing out the Euclidean distance between the data vector instance $b_k$ and the centroid of class $CL$ it belongs to according to minimum distance criterion to the corresponding centroid i.e. $\left(X_i^{CL_{min}(b_k)}\right)$ as in K-means.

$$Fit\,(X_i) = \sum_{k=1}^{D_v} d\left(b_k, X_i^{CL_{min}(b_k)}\right) \quad (2)$$

$D_v$ is the number of data vectors to be clustered. $d(.,.)$ is the Euclidean distance, $X_i$ is the ith solution of the population. So by choosing the discussed fitness function, the problem can be considered as a minimization problem which is defined as follows:

$$X_o = \mathrm{Arg}\left[\mathrm{Min}_{\forall i}\left(Fit(X_i)\right)\right] \quad (3)$$

$X_o$ is the optimal set of centroids. In the case of image clustering, $C$ depends on user, $N = 3$ for RGB colour image, $D_v$ is equal to image size. The partitions into $C$ classes should maintain the following properties:

1. Each cluster should have at least one data vector assigned i.e.,
   $C_i \neq \emptyset \; \forall i \in \{1,2,\ldots\ldots,c\}$
2. Two different clusters should have no data vector in common i.e.,
   $C_i \cap C_j = \emptyset, \forall i \neq j \; and \; i,j \in \{1,2,\ldots,c\}$.
3. Each data vector should definitely be attached to a cluster i.e.
   $\cup_{i-1}^{c} C_i = D_v$

## 2.2 Cuckoo Search (CS) Algorithm

Cuckoo search (CS) algorithm is a powerful optimization algorithm proposed by Xin-she Yang and Suash Deb in 2009 under the inspiration of the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds [27, 28].
A solution in the original cuckoo search algorithm corresponding to cuckoo nests represents the position of the cuckoo egg within the search space. Mathematically, this position is defined as:

$$x_i^t = \{x_{i,j}^t\}, \quad \text{for } i = 1,\ldots.,n \text{ and for } j = 1,\ldots\ldots,d \qquad (4)$$

Where, n denotes the number of cuckoo nests in the population, d is the dimension of the problem to be solved, and t the generation number. Generation of new solution signifies the exploitation of the current solutions is carried out by using the Lévy flight distribution expressed as:

$$x_i^{t+1} = x_i^t + \alpha. L\acute{e}vy() \qquad (5)$$

$\alpha > 0$ represents a scaling factor of the step size drawn from Lévy distribution i.e. $L\acute{e}vy()$. Lévy distribution has the ability of exploring a large amount of search space. In this study, Mantegna's algorithm [28] has been used to generate Lévy distribution. It produces random numbers according to a symmetric Lévy stable distribution as described below—

$$\sigma = [\Gamma(1+\beta)\, sin(\pi\beta/2)/\Gamma((1+\beta)/2)\beta 2^{(\beta-1)/2})]^{1/\beta} \qquad (6)$$

Where, Γ is the gamma function, 0< β ≤ 2. σ is the standard deviation. As per Mantegna's algorithm, the step length v can be calculated as,

$$v = {}^{x}\!\Big/\!{|\,y\,|}^{1/\alpha} \qquad (7)$$

Here, x and y are taken from normal distribution and $\sigma_x = \sigma, \sigma_y = 1$. Where $\sigma$ is the standard deviation. The resulting distribution has the same behavior of Lévy distribution for large values of the random variables. Mantegna's algorithm associates with faster computational speed in the range 0.75 ≤ α ≤ 1.95 [28].

In CS, exploration of the search space has been done by the following expression:

$$x_{i,j}^t = (Ub_j - Lb_j) * U_j(0,1) + Lb_j \qquad (8)$$

Where, $Ub_j$ and $Lb_j$ are the upper and lower bound of the specific variable. $U_j(0,1)$ is the random variable drawn from the uniform distribution. But, this exploration ability of the CS algorithm crucially depends on the probability $p_a \in [0,1]$ as this fraction of nests is abandoned.

Pseudo Code of the traditional CS is given as Algorithm 1.

**Algorithm 1: Cuckoo Search Algorithm**

```
Step-1:  Take objective function and generate initial
         population of $n$ solution randomly using Eq.8.
Step-2:  **While** *termination condition* does not meet **Do**
Step-3:     **for** $i$=1 to $n$ **Do**
Step-4:           Generate new solutions around $\boldsymbol{x}_i$ with Lévy
                  Flight as per Eq.5
Step-5:           $f_t$ = Suppose the new solution is $\boldsymbol{u_i}$ and find the fitness
                  values of $\boldsymbol{u_i}$
Step-6:           $j = [rand\,(0,1) * n + 1]$
Step-7:           **if** $f_t$<$f_i$ then
Step-8:              $\boldsymbol{x}_j = \boldsymbol{u}_i$; $f_j$= $f_t$
Step-9:           **end if**
Step-10:          **if** $rand\,(0,1) < P_a$ then
Step-11:             **Do** the initialization of worst nest according to Eq.8
                  and $P_a$
Step-12:          **end if**
Step-13:          **if** $f_t$<$f_{min}$
Step-14:             $\boldsymbol{x}_{best}$ =$u_i$ ; $f_{min}$=$f_t$; //replace the global best solution
Step-15:          **end if**
Step-16:    **end for**
Step-17: **end While**
```

## 3. Experimental Results

The experiment has been performed over 40 colour breast histopathology images with MatlabR2012b and Windows-7 OS, x64-based PC, Intel(R) Pentium (R)-CPU, 2.20 GHz with 4 GB RAM. The proposed methods are tested on images taken from UCSB Bio-Segmentation Benchmark dataset [29, 30]. Fig.1 represents the original images of different breast histopathology images.

## 3.1 Parameter Setting

Parameter setting is very crucial for any Nature-Inspired Optimization Algorithm (NIOA) and most of the cases it is performed from experience. The parameter setting of CS algorithm is as follows: $p_a = 0.2, \beta = 1.5, \alpha = 0.01$, population size ($n$) = 50. The Stopping Criterion for both CS and K-means [22] is the number of Fitness Evaluations (FEs), and the maximum number of FEs (i.e. MAX_FE) has been taken as $1000 \times D$. Where, $D$ is the number of clusters. In K-means, new centroid calculation has been considered as Fitness Evaluations (FEs).

## 3.2 Segmentation Quality Parameters

Segmentation efficiency of the employed methods are judged with the help of three well-known image quality assessment parameters namely Peak Signal to Noise Ratio (PSNR), Quality Index based on Local Variance (QILV) and Feature Similarity Index (FSIM).

QILV is a performance measurement technique proposed by Santiago Aja-Fernandez [31, 32]. QILV is used to measure the structural information of the image. A great amount of the structural information of an image is coded in its local variance distribution. Local variances features of an image can assists to compare two images properly. Greater QILV values indicate better segmentation quality.

PSNR [33, 34] is a pixel difference measurement technique. This is computed by averaging the squared intensity of original image and output image. Large value of PSNR demonstrates better-segmented image.

Zhang et al. proposed a new human vision measurement technique in 2011 called Feature Similarity Index (FSIM) [35]. FSIM is mainly designed for gray scale images or luminance component of colour images. This method is a combination of two low level features namely Phase Congruency (PC) and Gradient Magnitude (GM). Greater value of FSIM represents better-segmented image.

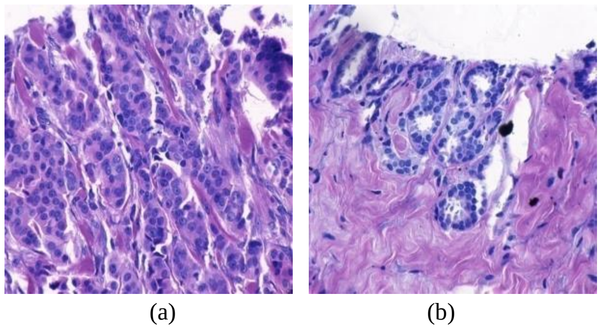
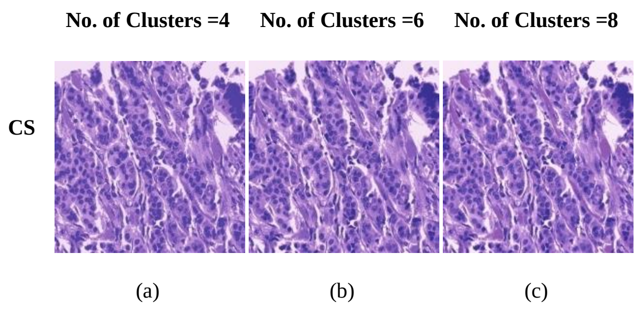


Fig.1 Original breast cancerous histopathology images



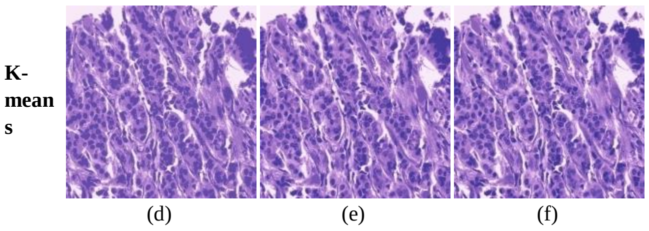


**Fig.2. Results of clustering for Fig.1(a): (a)-(c) represent results of CS using cluster number 4, 6 and 8 respectively; (d)-(f) represent results of K-means using cluster number 4, 6 and 8 respectively.**

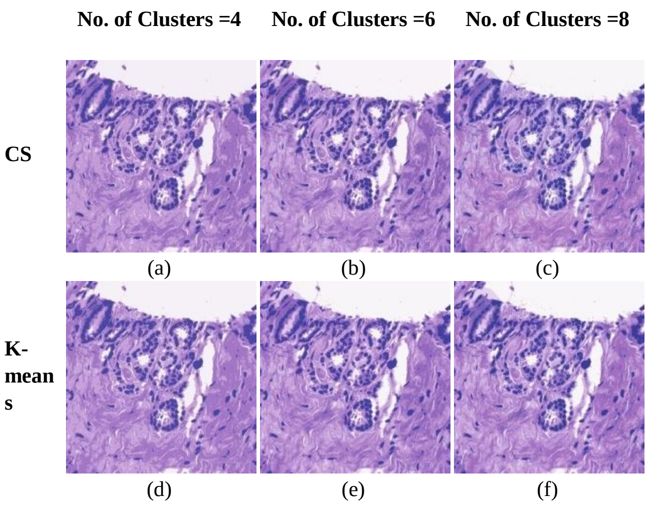


**Fig.3. Results of clustering for Fig.1(d): (a)-(c) represent results of CS using cluster number 4, 6 and 8 respectively; (d)-(f) represent results of K-means using cluster number 4, 6 and 8 respectively.**

Table 1. Average Quality parameters and other numerical results over 40 images

| Alg. | Clus. | PSNR | QILV | FSIM | Fitness | Time | STD. |
|---|---|---|---|---|---|---|---|
| CS | 4 | **25.69** | **0.98** | **0.95** | **0.097** | **2.003** | 3.31 E-17 |
| | 6 | **28.82** | **0.99** | **0.96** | **0.072** | **2.665** | 8.37 E-14 |
| | 8 | **30.72** | **0.99** | **0.97** | **0.068** | **3.108** | 1.23 E-14 |
| K-means | 4 | 24.81 | 0.97 | 0.90 | 0.189 | 45.14 | ............ |
| | 6 | 26.11 | 0.97 | 0.91 | 0.127 | 50.99 | ............ |
| | 8 | 27.80 | 0.98 | 0.93 | 0.102 | 56.84 | ............ |

**/* Best results obtained are given in bold*/**

## 3.3. Analysis of the experimental results

Performance analysis of the CS and classical K-means algorithms has been performed in the breast histopathology image clustering domain by computing their objective (fitness) function minimization ability, robustness in term of standard deviation (STD.) and computational time. Segmentation quality parameters like QILV, PSNR and FSIM are also calculated to judge the segmentation accuracy of the clustering models over cluster number 4, 6 and 8. CS algorithm based clustering model has been run 40 times for each image. The segmented images corresponding to Figs.1(a)-(b) are given as Figs.2-3 respectively. On the other hand, Table 1 represents the average values of fitness, standard deviation (STD.), computational time and quality parameters over 40 images. Table 1 shows that CS based clustering model provide minimized fitness value within less computational time. The standard deviation also proves the significant robustness of CS algorithm. The resultant segmented images of CS based model associated with greater QILV, FSIM and PSNR over cluster number 4, 6 and 8. Therefore, it can be said that CS is the better than traditional K-means algorithm with the MAX_FE based termination condition.

## Conclusion

A Cuckoo Search (CS) algorithm based clustering model has been proposed for the proper segmentation of breast histopathology images. The performance of the CS algorithm based clustering model has been compared to K-means algorithms in terms of fitness, computational time and quality parameters. Values of quality parameters indicate that CS based clustering model provide segmented images with higher QILV, FSIM and PSNR compares to K-means based clustering model. Analysis of the results also shows that K-means also associates with huge computational time when number of clusters increases and this time consume problem has been successfully surmounted by using CS algorithm.

Several future directions exist of this study such as use of fuzzy logic based clustering, rough set based clustering, formulation of the multi-objective clustering models, and use these clustering models for different kinds of images.

## REFERENCES

[1] Ferlay, J., Soerjomataram, I., Ervik, M., Dikshit, R., Eser, S., Mathers, C., ... & Bray, F. (2015). GLOBOCAN 2012 v1. 0, Cancer Incidence and Mortality Worldwide: IARC CancerBase No. 11. Lyon, France: International Agency for Research on Cancer; 2013.

[2] Elston, C. W., & Ellis, I. O. (1991). Pathological prognostic factors in breast cancer. I. The value of histological grade in breast cancer: experience from a large study with long-term follow-up. Histopathology, 19(5), 403-410.

[3] Robbins, P., Pinder, S., De Klerk, N., Dawkins, H., Harvey, J., Sterrett, G., ... & Elston, C. (1995). Histological grading of breast carcinomas: a study of interobserver agreement. Human pathology, 26(8), 873-879.

[4] Meijer, G. A., Beliën, J. A., Van Diest, P. J., & Baak, J. P. (1997). Origins of... image analysis in clinical pathology. Journal of clinical pathology, 50(5), 365.

[5] Gurcan, M. N., Boucheron, L., Can, A., Madabhushi, A., Rajpoot, N., & Yener, B. (2009). Histopathological image analysis: A review. IEEE reviews in biomedical engineering, 2, 147.

[6] Irshad, H., Veillard, A., Roux, L., & Racoceanu, D. (2014). Methods for nuclei detection, segmentation, and classification in digital histopathology: a review—current status and future potential. IEEE reviews in biomedical engineering, 7, 97-114.

[7] Jothi, J. A. A., & Rajam, V. M. A. (2015). Segmentation of nuclei from breast histopathology images using PSO-based Otsu’s multilevel thresholding. In Artificial Intelligence and Evolutionary Algorithms in Engineering Systems (pp. 835-843). Springer, New Delhi.

[8] Beevi, S., Nair, M. S., & Bindu, G. R. (2016). Automatic segmentation of cell nuclei using Krill Herd optimization based multi-thresholding and localized active contour model. Biocybernetics and Biomedical Engineering, 36(4), 584-596.

[9] Shen, P., Qin, W., Yang, J., Hu, W., Chen, S., Li, L., ... & Gu, J. (2015). Segmenting multiple overlapping Nuclei in H&E Stained Breast Cancer Histopathology Images based on an improved watershed.

[10] Veta, M., Van Diest, P. J., Kornegoor, R., Huisman, A., Viergever, M. A., & Pluim, J. P. (2013). Automatic nuclei segmentation in H&E stained breast cancer histopathology images. PloS one, 8(7), e70221.

[11] Wienert, S., Heim, D., Saeger, K., Stenzinger, A., Beil, M., Hufnagl, P., ... & Klauschen, F. (2012). Detection and segmentation of cell nuclei in virtual microscopy images: a minimum-model approach. Scientific reports, 2, 503.

[12] Qi, X., Xing, F., Foran, D. J., & Yang, L. (2012). Robust segmentation of overlapping cells in histopathology specimens using parallel seed detection and repulsive level set. IEEE Transactions on Biomedical Engineering, 59(3), 754-765.

[13] Xu, J., Janowczyk, A., Chandran, S., & Madabhushi, A. (2011). A high-throughput active contour scheme for segmentation of histopathological imagery. Medical image analysis, 15(6), 851-862.

[14] Paramanandam, M., Thamburaj, R., Manipadam, M. T., & Nagar, A. K. (2014, May). Boundary extraction for imperfectly segmented nuclei in breast histopathology images–a convex edge grouping approach. In International Workshop on Combinatorial Image Analysis (pp. 250-261). Springer, Cham.

[15] Su, H., Liu, F., Xie, Y., Xing, F., Meyyappan, S., & Yang, L. (2015, April). Region segmentation in histopathological breast cancer images using deep convolutional neural network. In Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on (pp. 55-58). IEEE.

[16] Naylor, P., Laé, M., Reyal, F., & Walter, T. (2017, April). Nuclei segmentation in histopathology images using deep neural networks. In Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on (pp. 933-936). IEEE.

[17] Shafique, S., & Tehsin, S. (2018). Computer-Aided Diagnosis of Acute Lymphoblastic Leukaemia. Computational and mathematical methods in medicine, 2018.

[18] Tosta, T. A. A., Neves, L. A., & do Nascimento, M. Z. (2017). Segmentation methods of H&E-stained histological images of lymphoma: a review. Informatics in Medicine Unlocked, 9, 35-43.

[19] Das, S., & Konar, A. (2009). Automatic image pixel clustering with an improved differential evolution. Applied Soft Computing, 9(1), 226-236.

[20] Omran, M., Engelbrecht, A. P., & Salman, A. (2005). Particle swarm optimization method for image clustering. International Journal of Pattern Recognition and Artificial Intelligence, 19(03), 297-321.

[21] Xin Ye & Yong-Xian Jin (2016) A Fuzzy C-Means Algorithm Based on Improved Quantum Genetic Algorithm," International Journal of Database Theory and Application, vol. 9-1, pages 227-236.

[22] Kapoor, S., Zeya, I., Singhal, C., & Nanda, S. J. (2017). A Grey Wolf Optimizer Based Automatic Clustering Algorithm for Satellite Image Segmentation. Procedia Computer Science, 115, 415-422.

[23] Li, H., Zhang, S., Zhang, C., Li, P., & Cropp, R. (2017). A novel unsupervised Levy flight particle swarm optimization (ULPSO) method for multispectral remote-sensing image classification. International Journal of Remote Sensing, 38(23), 6970-6992.

[24] Liang, Y., Zhang, M., & Browne, W. N. (2014, December). Image segmentation: a survey of methods based on evolutionary computation. In Asia-Pacific Conference on Simulated Evolution and Learning (pp. 847-859). Springer, Cham.

[25] Bong, C. W., & Rajeswari, M. (2012). Multiobjective clustering with metaheuristic: current trends and methods in image segmentation. IET image processing, 6(1), 1-10.

[26] De Falco, I., Della Cioppa, A., & Tarantino, E. (2007). Facing classification problems with particle swarm optimization. Applied Soft Computing, 7(3), 652-658.

[27] Yang, X. S., & Deb, S. (2009, December). Cuckoo search via Lévy flights. In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on (pp. 210-214). IEEE.

[28] Yang, X. S. (2010). Nature-inspired metaheuristic algorithms, Luniver press.

[29] Gelasca, E. D., Byun, J., Obara, B., & Manjunath, B. S. (2008, October). Evaluation and benchmark for biological image segmentation. In Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on (pp. 1816-1819). IEEE.

[30] Gelasca, E. D., Obara, B., Fedorov, D., Kvilekval, K., & Manjunath, B. S. (2009). A biosegmentation benchmark for evaluation of bioimage analysis methods. BMC bioinformatics, 10(1), 368

[31] Aja-Fernandez, S., Estepar, R. S. J., Alberola-Lopez, C., & Westin, C. F. (2006, August). Image quality assessment based on local variance. In Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE (pp. 4815-4818). IEEE.

[32] Dhal, K. G., Sen, M., & Das, S. (2018). Multi-Thresholding of Histopathological Images Using Fuzzy Entropy and Parameterless Cuckoo Search. In Critical Developments and Applications of Swarm Intelligence (pp. 339-356). IGI Global.

[33] Suresh, S., & Lal, S. (2017). Multilevel thresholding based on Chaotic Darwinian Particle Swarm Optimization for segmentation of satellite images. Applied Soft Computing, 55, 503-522.

[34] Dhal, K. G., Fister Jr., I. & S. Das (2017). Parameterless Harmony Search for image Multi-thresholding. 4th Student Computer Science Research Conference, University of Maribor, Slovenia, pp.5-12, 2017.

[35] Zhang, L., Zhang, L., Mou, X., & Zhang, D. (2011). FSIM: a feature similarity index for image quality assessment. IEEE transactions on Image Processing, 20(8), 2378-2386.

# Time series classification with Bag-Of-Words approach

Domen Kavran
Faculty of Electrical Engineering and Computer Science
Koroska cesta 46
Maribor, Slovenia
domen.kavran@um.si

## ABSTRACT

The amount of data generated every day increases each year and the pace is accelerating with development of Internet of Things (IoT). Gathered data solely doesn't contain much information, but with machine learning additional informations and hidden patterns can be obtained to contribute to time series analysis.

General purpose and problem specific time series feature extraction methods have been developed over the years. New feature extraction approach, derived from Bag-Of-Words, is presented in this paper. Main part of the approach is obtaining a dictionary of time series segments – the so-called words. K-Means clustering is used to form a dictionary containing K words, which is then used to define a feature vector of an individual time series as a histogram of word occurances inside it. Described approach can be used for feature extraction of time series without prior knowledge of data's nature. Moreover, the approach is robust and produces good classification results. Highest accuracy of 99.96% was achieved using datasets, presented in Results.

## Keywords

time series, classification, machine learning, bag-of-words

## 1. INTRODUCTION

Technological progress made in industries, like automotive, healthcare and electronics, over the past years resulted in increased amount of produced data. Large complex datasets, often referred to as Big Data, must be analysed quickly and efficiently to reveal additional informations and hidden patterns. Data analysis aids companies with their product development, research and customer services [5]. Different technologies and programming libraries have been developed to help engineers in creating pipelines for manipulating data and extracting features. Unsupervised machine learning algorithms are often used for analysing data and finding correlation between variables. Various advanced supervised techniques have been developed to solve regression and classification tasks. Time series data have an important role in today's largest industries and this paper presents a general purpose time series feature extraction approach.

Broader interest in time series classification began at the start of the century [9], which led to development of many different feature extraction methods [16]. For machine learning algorithms to be successful at classifying time series, appropriate features must be selected. Features can be obtained through statistical analysis or by calculating features specifically selected for a given dataset, though expert knowledge is needed. Presented alternative approach, derived from Bag-Of-Words [17], needs no prior knowledge about provided data. Main part is the definition of dictionary words – segments, which are extracted from training time series data and later clustered. Segments of individual time series are then compared with dictionary words and histogram of word occurances is formed. Histogram is a feature vector representing individual time series.

Bag-Of-Words approach was originally intended for document and image classification. Same concepts can be applied to time series with great results. Presented approach uses elements of well-known image patch extraction algorithm to obtain overlapping windows or segments, containing parts of time series data. To speed up the training phase and classification process, discrete wavelet transform was used. The transform is known for its role in data compression and image processing [3]. That provided a suitable way of reducing each segment's feature vector dimensionality. Mini Batch K-means clustering was used to speed up the dictionary creation process.

In second section of the paper, procedure of feature extraction is described. Classification was done by 1-nearest neighbor algorithm, using Chi-squared distance measure and then compared with the results of support vector machines and random decision forests in Results.

## 2. TIME SERIES CLASSIFICATION

Classification pipeline is shown on Figure 1. Time series must be described with features, which appropriately present occured events in time series and are independent of its length. Beforehand, input time series dataset has to be split on training and test datasets. Words inside time series are segments feature vectors, with features being approximation coefficients of discrete wavelet transform. Dictionary words are the result of clustering segments feature vectors, extracted from training time series set. Feature vector of each time series is the created histogram of dictionary words occurances.

Original approach has a name Bag-Of-Words for classifying documents and Bag-Of-Features for image classification [6]. Advantage of the approach is its robustness, simplicity and good results on real-world data.

DOI: https://doi.org/10.26493/978-961-7055-26-9.55-59

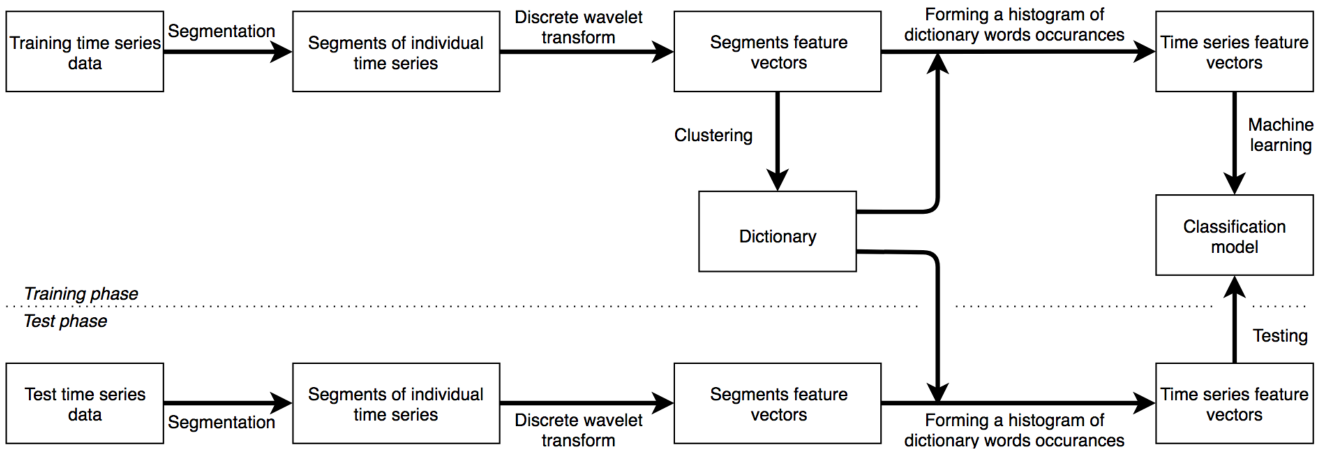

Training time series data
Segmentation
Segments of individual time series
Discrete wavelet transform
Segments feature vectors
Forming a histogram of dictionary words occurances
Time series feature vectors
Clustering
Machine learning
Dictionary
Classification model
Training phase
Test phase
Testing
Test time series data
Segmentation
Segments of individual time series
Discrete wavelet transform
Segments feature vectors
Forming a histogram of dictionary words occurances
Time series feature vectors

Figure 1: Classification pipeline

Histograms are often used in natural language processing and computer vision problems, because local informations of observed object are presented. Those characteristics justify use of histograms as time series feature vectors.

## 2.1 Extraction of segments

### 2.1.1 *Segmentation with sliding window algorithm*

Both training and test time series are split into overlapping segments with sliding window of length $W_c$ and step length $t_c < W_c$.

Example of segments extraction is shown on Figure 2. Time series of length 3072 values is split onto segments of length $W_c = 256$ with step $t_c = 192$. Segments are shown in orange color with grey overlapping parts. In most cases, individual segment's length should not exceed 10% of average time series length, although this percantage can differ between datasets. Selected window length has a big impact on classification accuracy and should be carefully selected.

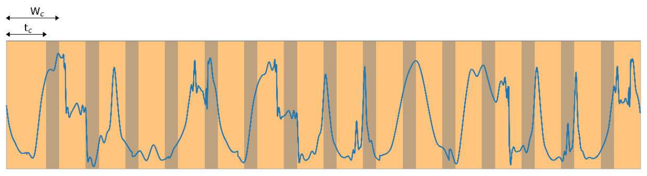


Figure 2: Segmentation of a time series

### 2.1.2 *Feature extraction*

Every segment is described with features being approximation coefficients of discrete wavelet transform (DWT) function for a selected decomposition level. With every increase of decomposition level, number of approximation coefficients is reduced by nearly a factor of two, but enough information is preserved to reconstruct original data. With selected transform, frequency informations at lower frequencies are analyzed and time informations in higher frequencies are gathered [17]. Additional benefit of the transform is data noise reduction. Db3 discrete wavelet transform function at first decomposition level was used to define segment's feature vector. That results in almost half of segment's length number of approximation coefficients [17]. Figure 3 shows application of discrete wavelet transform on a time series.

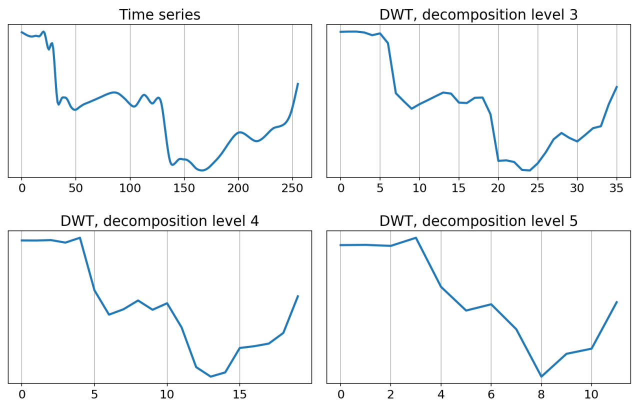


Figure 3: Approximation coefficients of applied db3 discrete wavelet transform with different decomposition levels

## 2.2 Dictionary words

K-means clustering has been applied on a set of segments feature vectors, extracted from training time series in previous step. Result of clustering is K number of clusters with centroids being dictionary words. In terms of speed, K-means algorithm is not suitable for large datasets. Because of the latter fact, alternative algorithm Mini Batch K-means, was used. Algorithm uses subset of the dataset per iteration and, consequently, less computations are needed.

For visualization purposes, conversion of segments multidimensional feature vectors into three dimensional was done, using principal component analysis (PCA). Mathematical transformation converts large number of potentially dependant variables to fewer independant variables – principal components, which hold the most information. Applying principal component analysis is one of the first steps in analysis of large, multivariate datasets [13]. Result of clustering PCA output is visible on Figure 4 with individual cluster being presented in unique color and have a centroid marked with a black dot.

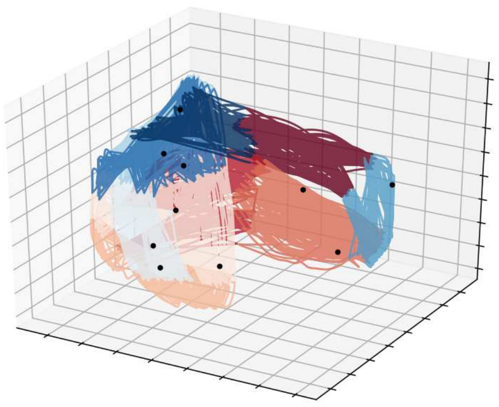


Figure 4: Vizualization of ten clusters as a result of applying clustering on first three principal components of training segments feature vectors

Recommended size of dictionary is ranging from 1000 to 3500 words. The reason is that classification done with fewer number of words doesn't reach the highest accuracy possible, but it starts to stabilize with a dictionary size above 1000 words [17].

## 2.3 Time series features

Defining time series features starts with segmentation and feature extraction, described in Chapter 2.1. Segments feature vectors are compared with dictionary words using 1-nearest neighbors algorithm using Euclidean distance as distance measure. Histogram of dictionary word occurances is created and normalized with $L^2$-norm. Example of time series histogram is shown in Figure 5.

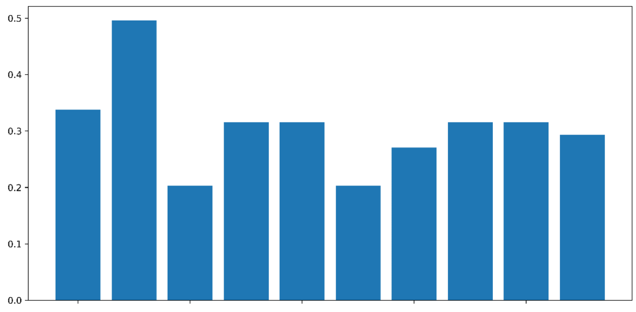


Figure 5: Normalized histogram based on a dictionary of ten words

Training time series, described with feature vectors, are used to train a classification model. For classification model testing, time series from test dataset are used.

## 2.4 Classification algorithms

Training classification models was done using the following described classification models.

### *2.4.1 K-Nearest neighbors*

K-Nearest neighbors (KNN) algorithm classifies a sample based on distance between training samples and unclassified sample. Euclidean distance is usually used as a distance measure and label is determined based on K nearest training samples. Odd number should be selected for parameter K value, because classification label decision-making is done by majority vote [11]. Selection of parameter K value has a big effect on final classification accuracy. The best approach to select appropriate parameter K is by trial and error. Time complexity of the algorithm is large, because most computational operations are being done during classification and not in training phase [10].

Chi-Square distance measure [18] was used, because it can measure differences between histograms. It is frequently used in classification of textures, objects and shapes. Chi-square distance measure takes distribution of values and their frequencies into account and also has an import characteristic of weighing rarely occured values in histogram [1]. Equation of Chi-square distance measure $\chi$ is

$$\chi^2(x_i, x_j) = \frac{1}{2} \sum_{k=1}^{d} \frac{(x_{ik} - x_{jk})^2}{x_{ik} + x_{jk}} \tag{1}$$

where:

| | |
|---|---|
| $x =$ | feature vector |
| $i, j =$ | feature vectors indices |
| $d =$ | length of feature vectors |
| $k =$ | feature index |

### *2.4.2 Support vector machine*

Algorithm is a part of linear classifiers group. It forms an optimal hyperplane to maximize distance between parallel planes, placed on outer boundaries of training samples feature vectors, based on their labels [12, 15]. Support vector machine (SVM) is a binary classifier, but can also be used for multilabel classification. Multiple binary classifiers are formed, with $i$-th label being treated as positive and the rest as negative. Described approach is called One-Vs-All [14]. Alternative approach has a name One-Vs-One, where a binary classifier is formed for every pair of labels. One-Vs-One approach is useful at dealing with unbalanced datasets, but that comes at a cost of higher time complexity during training and classification process [8].

### *2.4.3 Random forest*

Training starts by forming a set of decision trees with induced randomness. A sample is classified based on majority vote of decision trees [7]. Number of trees, their depth, minimal number of training samples to split a node and minimal number of samples required to be at leaf node must be adjusted to optimize classification accuracy. Random forest is a fast learning and general-purpose classification algorithm, which is resistant to overfitting at large number of features [4]. To achieve high accuracy, large set of decision trees has to be created, but that also means slower classification process.

## 3. RESULTS

Presented feature extraction approach was implemented in Python using scientific computing library NumPy. Machine learning library scikit-learn was used for clustering and classification. For discrete wavelet transform, open-source software PyWavelets was used. Training and testing was done on the following computer hardware:

- Intel Core i7-6700K processor
- 16GB of DDR4 3200MHz memory

Various open-source time series datasets [2] were used for testing and are described in Table 1. Each dataset contains time series of equal length and split in two classes or more. Testing was done using 5-fold cross-validation with F1 score as classification accuracy measurement for all classification algorithms, presented in Chapter 2.4. Dictionaries containing 2000 words were used. Parameters $W_c$ and moving step $t_c$ were choosen individually for every time series. The reason being is that their lengths differ and experimenting with different parameter values was needed to reach best possible classification score. Classification results are presented in Table 2.

Table 1: Time series descriptions

| Time series | Description |
|---|---|
| CBF | Simulated time series |
| ECG5000 | ECG signal of a patient having a heart failure |
| FordA | Engine sounds |
| Mallat | Simulated time series |
| ShapesAll | Binary images contours distances from center |
| UWaveGestureLibraryAll | Accelerometer data generated during hand movement |
| StarlighCurves | Brightness of astronomical objects through time |
| Wafer | Measurements recorded from sensors during the processing of silicon wafers |

Highest average classification F1 score 91.82% was achieved with SVM classification model, followed by random forest and 1-nearest neighbor using Chi-square distance measure. Biggest difference in classification results can be seen at UWaveGestureLibraryAll, where 1-NN classifier achieved F1 score of 25.53%, opposed to SVM's score of 80.64%. F1 scores of all classification models are closest at classifying time series CBF, with a difference of only 0.75% between them.

## 4. CONCLUSION

Highest achieved classification accuracy on test datasets was 99.96%, with average across all three classification models being 85.75%. Feature extraction approach is appropriate for use in various industry related problems and at analysis of stock market, but medical use is questionable, because achieved accuracy must be at highest level possible to ensure medical safety. Best such case are classification scores for time series ECG5000, where highest achieved F1 score is 86.89%, much lower than satisfying for critical medical data.

Parallel computing can be used for extraction of segments and at defining time series feature vectors. Presented feature extraction approach is appropriate for integration in real-time systems and IoT devices, however training phase should be done separately from working environment. Higher classification accuracy could be achieved with use of multivariate time series.

## 5. REFERENCES

[1] V. Asha, N. U. Bhajantri, and P. Nagabhushan. GLCM based Chi-square Histogram Distance for Automatic Detection of Defects on Patterned Textures. *International Journal of Computational Vision and Robotics*, 2(4):302–313, 2011.

[2] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. The UEA and UCR Time Series Classification Repository. Available at `www.timeseriesclassification.com`.

[3] P.M. Bentley and J.T.E. McDonnell. Wavelet transforms: an introduction. *Electronics and Communication Engineering Journal*, 6(4):175–186, 1994.

[4] G. Biau. Analysis of a Random Forests Model. *Journal of Machine Learning Research*, 13:1063–1095, 2012.

[5] Thomas H. Davenport. *Big Data at Work*. Harvard Business Review Press, 2014.

[6] F. Fang, H. Lu, and Y. Chen. Bag of Features Tracking. *International Conference on Pattern Recognition*, 2010(46):153–156, 2010.

[7] K. Fawagreh, M. Gaber, and E. Elyan. Random forests: from early developments to recent advancements. *Systems Science and Control Engineering*, 2(1):602–609, 2014.

[8] A. Gidudu, G. Hulley, and T. Marwala. Image classification using SVMs: One-Against-One vs One-Against-All. *Clinical Orthopaedics and Related*

Table 2: Classification F1(%) results

| Time series | No. of classes | Parameters | 1-Nearest neighbor, Chi-square distance | SVM, polynomial kernel, degree = 2 | Random forest |
|---|---|---|---|---|---|
| CBF | 3 | $W_c$=50, $t_c$=5 | 99.14 | 99.89 | 99.35 |
| ECG5000 | 5 | $W_c$=10, $t_c$=5 | 77.91 | 86.89 | 84.18 |
| FordA | 2 | $W_c$=90, $t_c$=5 | 70.41 | 89.96 | 81.92 |
| Mallat | 8 | $W_c$=10, $t_c$=5 | 98.78 | 99.96 | 97.08 |
| ShapesAll | 60 | $W_c$=100, $t_c$=5 | 83.31 | 80.14 | 62.19 |
| UWaveGestureLibraryAll | 8 | $W_c$=50, $t_c$=5 | 25.53 | 80.64 | 57.86 |
| StarlightCurves | 3 | $W_c$=220, $t_c$=5 | 91.47 | 97.56 | 96.89 |
| Wafer | 2 | $W_c$=10, $t_c$=5 | 98.52 | 99.54 | 99.04 |
| F1 average | / | / | 80.63 | 91.82 | 84.81 |

*Research*, 0711.2914, 2007.

[9] Juan J. Rodr Guez and Carlos J. Alonso. Boosting Interval-Based Literals: Variable Length and Early Classification. *Series in Machine Perception and Artificial Intelligence*, 57:149–171, 2002.

[10] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer. KNN Model-Based Approach in Classification. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 986–996, 2003.

[11] E. M. Kamel, M. Wafy, H. M. Ibrahim, and I. A. Badr. Comparison of different classification algorithms for certain weed seeds' species and wheat grains identification based on morphological parameters. *IJCSI International Journal of Computer Science Issues*, 12(5):110–116, 2015.

[12] G. Oreški and S. Oreški. An experimental comparison of classification algorithm performances for highly imbalanced datasets. *Central European Conference on Information and Intelligent*, pages 4–11, 2014.

[13] M. Richardson. Principal Component Analysis. 2009.

[14] R. Rifkin and A. Klautau. In Defense of One-Vs-All Classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[15] D. Srivastava and L. Bhambhu. Data Classification using support vector machine. *Journal of Theoretical and Applied Information Technology*, 12(1), 2005.

[16] Michele A. Trovero and Michael J. Leonard. Time Series Feature Extraction. 2018.

[17] J. Wang, P. Liu, M. F. H. She, S. Nahavandi, and A. Kouzani. Bag-of-words representation for biomedical time series classification. *Biomedical Signal Processing and Control*, 8(6):634–644, 2013.

[18] W. Yang, L. Xu, X. Chen, F. Zheng, and Y. Liu. Chi-Squared Distance Metric Learning for Histogram Data. *Mathematical Problems in Engineering*, 2015:1–12, 2015.

# The problem of quantum computers in cryptography and post-quantum cryptography

Dino Vlahek
Faculty of Electrical Engineering and Computer Science, Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia
dino.vlahek@gmail.com

## ABSTRACT
This paper presents the problem that quantum computing brings to modern cryptography. The basics of post-quantum cryptography and concepts of modern cryptography with an emphasis on the most useful asymmetric encryption algorithms are shown. The paradigms of post-quantum cryptographic algorithms, which were implemented in the Open Quantum Safe project and whose effectiveness can be compared with the most popular modern ones, were explained and analysed further in this paper. To compare performances, a test of the time and communication capabilities of selected algorithms was made, the results of which were graphically and descriptively presented. The results of the experiment showed that there is an effective quantum-resistant alternative to existing asymmetric encryption algorithms.

## Categories and Subject Descriptors
E.3 [**DATA ENCRYPTION**]: Miscellaneous; E.4 [**CODING AND INFORMATION THEORY**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

## General Terms
Theory, Experiment

## Keywords
cryptography, post-quantum cryptography, quantum computer, asymmetric encryption algorithms, keys

## 1. INTRODUCTION
Quantum computer is a computer model that works differently from the classical one. It is based on quantum mechanical phenomena, which makes it easier to solve certain problems that classical computers can not solve or need for it infinite resources [8]. For example, the problem of discrete logarithm (Diffie-Hellman Key Exchange (DH), Elliptic-Curve Cryptography (ECC)) or problem of factoring positive integers is an example of (Rivest-Shamir-Adleman (RSA)) the problems that quantum computers can solve faster and more reliably. Despite the fact that there is a few more years (or decades) before the emergence of effective physical quantum computers [6], today experts in the field of cryptography and security are already preventively solving the problems brought by quantum computing. It is necessary to know and understand which solutions, resistant to quantum computing, already exist and on which mathematical models they were based. Existing solutions of asymmetric cryptographic algorithms that are resistant to quantum attacks will be compared to existing most popular asymmetric cryptographic algorithms (RSA, DH, ECC). Asymmetric cryptographic algorithms, implemented within the Open Quantum Safe project (Open Quantum Safe (OQS)) will be used, the efficiency of which will be compared with RSA, DH and ECC asymmetric encryption algorithms. Efficiency measurements will be made, which will yield the results of comparing the efficiency of quantum-resistant asymmetric algorithms (i.e., post-quantum cryptography) and existing asymmetric encryption algorithms (RSA, DH, ECC).

## 2. ASYMMETRIC CRYPTOGRAPHY
The concept of asymmetric cryptography occurred due to an attempt to attack two biggest problems of symmetric cryptography: the problem of key distribution and digital signing. The key distribution problem in symmetric cryptography is defined by the way two participants share the same key over an unsecured communication channel. A digital signature gives the recipient a reason to believe that a message has been created by a known sender who can not decline the sent message and that the message has not been changed in the transition. Asymmetric cryptography is based on two different keys: the encryption key and the decryption key. The basic steps of asymmetric cryptography are as follows [11]:

- Each user makes a pair of keys that will be used to encrypt and decrypt the message.
- Each user gives one key in a public register or in a publicly accessible file. This key is a public key, and the other is a private key.
- Each user makes his own collection of public keys of other users with whom he communicates. If the sender wishes to send a confidential message to the recipient,

**Table 1: Use of most popular asymmetric algorithms**

| Algorithm | E/D | Digital signature | Key exchange |
|---|---|---|---|
| RSA | Yes | Yes | Yes |
| DH | No | No | Yes |
| ECC | Yes | Yes | Yes |
| DSS | No | Yes | No |

**Table 2: The influence of quantum computers on standard cryptographic algorithms [2]**

| Algorithm | Type | Influence |
|---|---|---|
| AES | symmetric | required larger key size |
| SHA-3 | hash function | requiring larger output |
| RSA | asymmetric | it is no longer safe |
| ECC | asymmetric | it is no longer safe |
| DSA | asymmetric | it is no longer safe |

the sender encrypts the message using the public key of the recipient.

## 3. POST-QUANTUM CRYPTOGRAPHY

The most critical parts of the communication protocols were based mainly on three key cryptographic operations: public key encryption, digital signing and key exchange. These operations were performed using DH method for key exchange, RSA cryptosystem and cryptosystem of elliptic curves. The security of these algorithms depends on the complexity of mathematical problems, such as the search for a common factor or discrete logarithm problem. The Shor's algorithm efficiently solves these mathematical problems, making all the public key cryptosystems weak. When we arrive in the quantum computer period, the most useful encryption methods will become obsolete [2]. It is assumed that the probability that the RSA-2048 will get broken i.e., that a sufficiently large quantum computer will be build by 2026, is 1/7, and 1/2 by 2031 [7]. Due to the fact that in symmetric encryption, with the enhancement of the key size, reliable protection against quantum computers is also achieved, in the research of the protection of asymmetric encryption the emphasis is on post-quantum cryptography or quantum-resistant cryyptography(QRC). There are several sets of methods for post-quantum cryptography: Code-based, SHA-based, multivariate polynomials, the super singular isogenic Diffie-Hellman, etc. The algorithms are still in the research phase and are not guaranteed to be safe [12] [2].

### 3.1 Open Quantum Safe

The goal of the OQS project is to develop quantum-resistant cryptography and integration of existing post-quantum cryptographic asymmetric algorithms into one Libqos library [47]. Libqos is an open-source library of quantum-resistant cryptographic algorithms written in C language. The importance of the library lies in quantum-resistant cryptographic asymmetric algorithms that focuses on key exchange [44]. In addition, OQS allows integration of Liboqs library into OpenSSL [10]. The algorithms that the library includes are:

- Supersingular isogeny Diffie–Hellman based: MSR SIDH algoritem, MSR SIKE algoritem
- Learning with errors based: FRODO
- Ring learning with errors based: BCNS15, NewHope, MSR LN16
- Module learning with errors based: Crystals-Kyber
- Lattice-based based: NTRU
- Error-correcting code based: McBits

## 4. ANALYSIS OF THE EFFICIENCY OF POST-QUANTUM CRYPTOGRAPHIC ALGORITHMS

The experiment examines the efficiency of asymmetric cryptographic encryption algorithms compared to existing asymmetric standards (DH, RSA, ECC). Cryptographic algorithms implemented within the OQS project are used, the efficiency of which are compared with RSA, DH and ECC asymmetric encryption algorithms. There is no need to analyze symmetric encryption algorithms because the length of the keys in symmetric encryption algorithms provides a higher level of security than in asymmetric ones. Efficiency measurements are made to verify the following hypothesis: there are quantum-resistant encryption asymmetric algorithms that could replace existing classical encryption asymmetric algorithms (RSA, DH, ECC).

The experiment is chosen because of its suitability, which allows us to measure the time of execution of algorithms in a controlled environment, thus assessing their effectiveness and performance. It is necessary to make a test case i.e., a test software that will measure the number of iterations of selected asymmetric encryption algorithms at a given time. Post-quantum asymmetric encryption algorithms are selected from the OQS project,while modern ones are from the Crypto++ library. Selected asymmetric encryption algorithms are [5]:

- OQS project: LWE Frodo recommended, RLWE BCNS15, RLWE MSR LN16, RLWE NewHope, SIDH MSR p503 and SIDH MSR p751, SIKE MSR p503 and SIKE MSR p751.
- Crypto++: RSA (1024, 2048, 3072), DH (512, 1024), ECC256

All unnecessary background processes of the computer was turned off to minimize the impact of its background activity. Executional file (.exe) of tests was prepared and launched. The time in which iterations of algorithms are performed is limited to 30 seconds. Each test will be triggered 40 times + 10% of all tests (40) = 44. A total of 330 minutes or approximately 5 hours and 30 minutes is foreseen for performing all tests. To reduce the deviation of the number of iterations of individual algorithms that happens due to unexpected deviations in the work of the computer, it is necessary to repeat the test numerous times.

### 4.1 Working environments, frameworks and tools

The main tool for working on the experiment is the Intel® Core ™ i7-5500U processor, 8GB RAM and the Windows 10 Pro version 1709. Post-quantum asymmetric encryption algorithms were implemented in the Libqos library, written in C, as part of the OQS project. Modern asymmetric encryption algorithms are implemented within the Crypto++ open source library, written in C++ language. We installed these libraries for the Visual Studio 2017 development environment, on Windows SDK version 10.0.17134.0, on a 64bit platform. Once all tests where performed, the data was aggregated and analyzed in the Microsoft Excel tool. GraphPad Prism tool was used to draw graphs that will show the results of the experiment.

## 4.2 Implementation of the experiment

To perform an experiment, i.e., measurements of the efficiency of individual asymmetric modern and post-quantum cryptographic algorithms, two test cases had to be performed: One test case for measuring modern asymmetric encryption algorithms and the other for measuring post-quantum asymmetric encryption algorithms. Modern asymmetric encryption algorithms were used from the Crypto++ library version 7.0.0. Test case, the console application, was built in Release Mode on a 64bit platform. The algorithms selected for measurement were: RSA (1024, 2048, 3072), DH (512, 1024), DH (1024) with predefined parameters, ECC256. Each algorithm was running over a time period of 30 seconds, where we measured the number of iterations. The number of runs for each algorithm was saved in the .csv file, together with the name and spent time. The test case was limited to 11 repetitions of measurement of each algorithm, each of the 7 algorithms was run 11 times in 30 seconds. The test case was performed four times. Each lasted about 40 minutes, totalling about 2 hours and a half. For RSA, the number of key generation and validation were measured, while for DH and EEC the number of generations and key aggrements. Post-quantum asymmetric encryption algorithms were used from the libqos library, which is part of the OQS prototype project. Test case, the console application was generated in Release Mode on a 64bit platform. The test case is similar to the test case for modern cryptographic encryption algorithms. The algorithms selected for measurement were: LWE Frodo recommended, RLWE BCNS15, RLWE MSR LN16, RLWE NewHope, SIDH MSR p503 and SIDH MSR p751, SIKE MSR p503 and SIKE MSR p751. For each algorithm, the number of implementations over a time period of 30 seconds was measured. The number of runs for each algorithm was saved in the .csv file, together with the name and spent time. The test case was limited to 11 repetitions of measurement of each algorithm, each of the algorithms was run 11 times in 30 seconds. The test case was performed four times. Each lasted about 4 minutes, totalling about 3 hours. For all algorithms the number of generation and key agreements were measured.

## 4.3 Results and data analysis

Apart from reliable cryptographic properties, the ideal algorithm for asymmetric encryption should have as little communication requirements as possible, low memory consumption and the least possible execution time. This paper takes into account the speed of execution and communication requirements as the main metrics for comparison, while the memory consumption is ignored.

**Table 3: Display of communication between parties and security bits**

| Name | A → B (bytes) | B → A (bytes) |
|---|---|---|
| Frodo | 11377 | 11296 |
| BCNS15 | 4096 | 4096 |
| MSR LN16 | 1824 | 2048 |
| NewHope | 1824 | 2048 |
| SIDH p503 | 378 | 378 |
| SIDH p751 | 564 | 564 |
| SIKE p503 | 378 | 402 |
| SIKE p751 | 564 | 596 |
| DH-1024 | 128 | 128 |
| DH-512 | 64 | 64 |
| ECDH256 | 32 | 32 |
| RSA-1024 | 128 | 128 |
| RSA-2048 | 256 | 256 |
| RSA-3072 | 384 | 384 |

**Table 4: Display of classical and quantum security bits**

| Name | Classical bits | Quantum bits |
|---|---|---|
| Frodo | 144 | 130 |
| BCNS15 | 86 | 87 |
| MSR LN16 | 128 | 112 |
| NewHope | 229 | 206 |
| SIDH p503 | 192 | 128 |
| SIDH p751 | 256 | 192 |
| SIKE p503 | 192 | 128 |
| SIKE p751 | 256 | 192 |
| DH-1024 | 73 | - |
| DH-512 | 50 | - |
| ECDH256 | 136 | - |
| RSA-1024 | 73 | - |
| RSA-2048 | 103 | - |
| RSA-3072 | 128 | - |

### *4.3.1 Communication requirements*

The communication between the parties, A → B and B → A, was obtained from the program code and the implementation of algorithms which was verified with data in sources [10] [12] [3] [4] [1] [5] [6] [9]. The table 3 shows the communication requirements between parties, the amount of data exchanged by the parties when the keys were generated and the table 4 shows the classical and quantum security bits of the algorithm.

### *4.3.2 Time complexity*

Figure 1 shows the time required for the tested algorithm is executed once in milliseconds. From the graph, it can be determined which algorithms are more successful i.e., faster.

### *4.3.3 Interpretation of results*

The most effective post-quantum asymmetric encryption algorithms are BCNS15, LN16 and NewHope. LN16 and NewHope were faster than all post-quantum and also of all classic asymmetric encryption algorithms. The disadvantage of the fastest post-quantum algorithms is relatively high communication complexity. In generating and distributing the keys, the LN16 and NewHope algorithms make 3,872

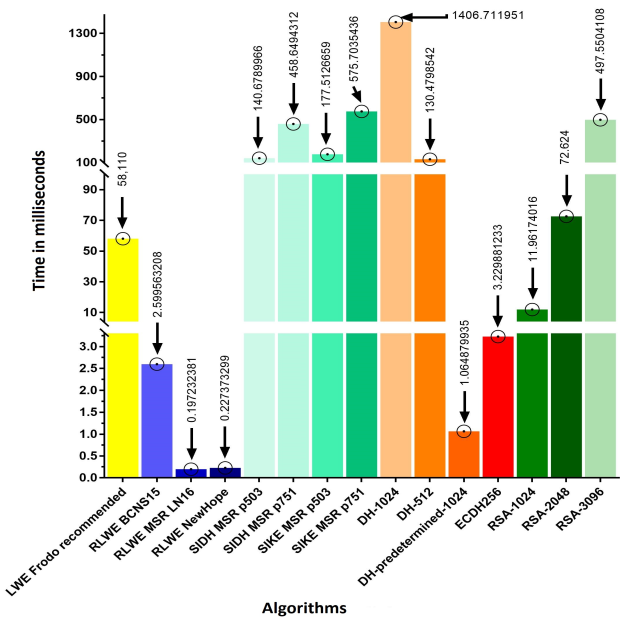


**Figure 1: Display of time complexity**

bytes of communication, BCNS15 8,320, and classical algorithms ∼ 10 times less (DH-1024, RSA-1024 256 bytes, RSA-2048 512 bytes, RSA-3072 768 bytes, ECCDH 64 bytes ). Frodo algorithm has average time complexity, but the greatest communication. The algorithms of the super-singular Diffie-Hellman group (SIDH and SIKE) have the smallest communication complexity, but are the slowest of all measured post-quantum encryption algorithms. The SIDH and SIKE keys are comparable to the RSA equivalent. In practice, asymmetric encryption algorithms have predefined parameters, which results in faster performance. Figure 1 shows that BCNS15, LN16 and NewHope are also faster than DH-1024, which have predefined parameters. BCNS15, LN16 and NewHope do not have predefined parameters in our case. RSA, DH and ECC have a smaller, better, communication complexity than most OQS algorithms, except the RSA-3072, which has an imperceptibly worse communication complexity than the SIDH MSR p501 and SIKE MSR p501. When talking about communication complexity, RSA, DH and ECC are noticeably more efficient, but the time complexity is on the BCNS15, LN16 and NewHope side. Of the most effective, the BCNS15 has the smallest keys, the size of 86 bits or 78 quantum bits, the LN16 has a size of 128 bits, 112 quantum bits and NewHope size of 229 bits, respectively 206 quantum bits.

## 5. CONCLUSIONS

The purpose of this paper is the prevention, i.e., to get the results of the effectiveness of existing quantum-resistant algorithms that can replace classical, non-resistant. The emphasis was on an experiment that gave the results of comparing the efficiency of post-quantum asymmetric cryptographic algorithms implemented within the OQS project and the existing asymmetric encryption algorithms (RSA, DH, ECC). For the purposes of the experiment, it was necessary to make a test case that measures certain parameters: time, communication between parties and the size of the output from the algorithm - key. The experiment has been successfully performed, with many repetitions for better confidence into the results, which are then analyzed and graphically presented. Some potential candidate algorithms proved to be very promising. The interpretation of the results presents the most successful algorithms in three different domains: time complexity, communication complexity and key size. Today, when online communication is very fast, the difference in communication complexity, given the speed of online communication, is negligible, and therefore it is safe to conclude that BCNS15, LN16 and NewHope algorithms, which have proven to be the fastest but have relatively large communication requirements, can become an alternative to RSA, ECC and DH.

## 6. REFERENCES

[1] Alkim, E., Ducas, L., Pöppelmann, T., and Schwabe, P. Post-quantum key exchange - a new hope. *IACR Cryptology ePrint Archive 2015* (2015), 1092.

[2] Bernstein, D. J., Buchmann, J., and Dahmen, E. *Post-quantum cryptography.* Springer-Verlag Berlin Heidelberg, Tiergartenstraße 17, 69121 Heidelberg, Germany, 2009.

[3] Bos, J. W., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., and Stebila, D. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. *IACR Cryptology ePrint Archive 2016* (2016), 659.

[4] Costello, C., Longa, P., and Naehrig, M. Efficient algorithms for supersingular isogeny diffie-hellman. *IACR Cryptology ePrint Archive 2016* (2016), 413.

[5] Crypto++. Crypto++ library 7.0 | free c++ class library of cryptographic schemes.

[6] Mermin, N. D. *Quantum Computer Science: An Introduction.* Cambridge University Press, Cambridge University Press University Printing House Shaftesbury Road, Cambridge CB2 8BS United Kingdom, 2007.

[7] Mosca, M. Cybersecurity in an era with quantum computers: will we be ready? *IACR Cryptology ePrint Archive 2015* (2015), 1075.

[8] Nielsen, M. A., and Chuang, I. L. *Quantum Computation and Quantum Information.* Cambridge University Press, Cambridge University Press University Printing House Shaftesbury Road, Cambridge CB2 8BS United Kingdom, 2010. pg. 555.

[9] OQS. Github - open-quantum-safe/liboqs: C library for quantum-resistant cryptographic algorithms.

[10] OQS. Github - open-quantum-safe/openssl: Fork of openssl that includes quantum-resistant algorithms and ciphersuites based on liboqs.

[11] Stallings, W. *Cryptography and Network Security: Principles and Practice*, 5th ed. Prentice Hall Press, Upper Saddle River, NJ, USA, 2010.

[12] Stebila, D., and Mosca, M. Post-quantum key exchange for the internet and the open quantum safe project. In *IACR Cryptology ePrint Archive* (2016).

# Named Entity Recognition and Classification using Artificial Neural Network

Luka Bašek
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenija
luka.basek@student.um.si

Borko Bošković
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenija
borko.boskovic@um.si

## ABSTRACT

In this paper, we will analyze variants of Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) based models for sequence tagging, special for Name Entity Recognition (NER). The models which will be analyzed include LSTM network, bidirectional LSTM network (BI-LSTM), GRU network and bidirectional GRU network (BI-GRU) and using pre-trained GloVe vectors [1], Part of speech and characters embedding for features. We evaluated our models with data set for sequence labeling task CoNLL 2003 corpus. We obtain F1 score on this dataset - 86.04% [2].

## Keywords

Natural Language Processing, Named Entity Recognition, Neural Network, Recurrent Neural Network, LSTM, GRU, Keras, GloVe vectors

## 1. INTRODUCTION

By developing information technologies, people today have a quick and wide access to a large amount of data. Everyday we meet a lot of sources of knowledge such as social networks, news, reviews, blogs, etc. There is a lot of data but the data is simple and raw isolated facts which have some meaning [15]. In the world is more and more data every day and data have to be analyzed to become a useful information. The computer can quickly process data and store the information. However, the computers without human presence can't analyze text and provide information about text. For this purpose, computer science has developed a lot of methods and techniques for analyzing and discovering knowledge. The field of Natural Language Processing (NLP) is an interdisciplinary area at the crossroads of computer science, artificial intelligence and linguistics. Our goal is Information Retrieval, more specifically the task of Named entity recognition.

To solve the problem of Named entity recognition, a variety of machine learning algorithms were used. The set of algorithms was composed of several statistical models such as the Hidden Markov Models (HMM) Florian et al. (2003) [4], Maximal Entropy Models (MaxEnt) Chieu et al. (2003) [1], Conditional Random Fields (CRF) McCallum et al. (2003) [12]. There are some machine learning methods but in this paper, we decide to use deep learning methods. On CoNLL 2003 one of the solutions contain Recurrent Neural Networks (RNN) with LSTM cells Hammerton (2013) [7] and this is the start point of our work.

The task is to explore deep learning methods in NLP and more specific for Named Entity Recognition (NER). Back in the few years, the neural networks have proved to be effective in the NLP area such as Text Classification Zhang et al. (2015) [21], Sentiment Analysis Severyn et al. (2015) [17], Part of speech tagging Wang et al. (2015) [19].

In the case of solving the problem of Name Entity Recognition, Hammerton (2003) [7] using RNN with LSTM cells presented by Hochreiter et al. (1997) [8]. Huang et al. (2015) [9] presents more complex models for NER based on the LSTM network, Bi-directional LSTM network, and various combinations with the CRF layer.

In this paper, we propose a variety of recurrent neural networks based models which include LSTM networks, bidirectional LSTM networks (BI-LSTM), GRU networks and bidirectional (BI-GRU) networks. Beside network architectures, we include additional features to help neural networks learn about the context of words. We include pre-trained GloVe for word representation, Part of Speech tags and Characters Embedding. We evaluate our models on English data from CoNLL 2003 shared task Sang et al. (2003) [16]. Our best F1 score is 86.05%. Slightly worse than current attractive solutions which have F1 score 91.21%, Xuezhe et al. (2016) [11] but this is a good starting point for improvements.

## 2. MODELS

In this section, we describe the models used in this paper: LSTM, BI-LSTM, GRU, BI-GRU.

### 2.1 LSTM Network

For operating sequential data the RNN is the most used approach. RNN takes as input a sequence of vectors $\vec{x} = (x_1, ..., x_n)$ and return another sequence $\vec{h} = (h_1, ..., h_n)$ that represents some information about the sequence by one-time unit. Reading articles about RNN, in theory, it works

[1]https://nlp.stanford.edu/projects/glove/
[2]The code of the this project is available at: https://github.com/lbasek/named-entity-recognition

but not in practice. The LSTM have been developed to avoid classic RNN problem about memory-cell and can handle the long-range dependencies.
LSTM was introduced in 1997 by Sepp Hochrieiter and Jürgen Schmidhuber [8]. LSTM cell has three gates: the forget gate, input gate, and output gate. Figure 1 shows the structure of the LSTM cell. All gates contain sigmoid function where the cell decide how much information will be blocked or passed from the cell. It's known the sigmoid function takes input and presents the output values in the range [0,1]. The value 0 means let nothing through, and the value of 1 means let everything through. These gates have own weights that are adjusted via gradient descent method.

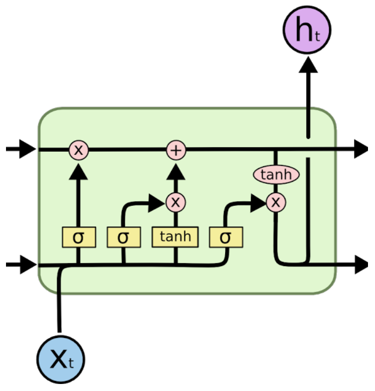


**Figure 1: The structure of the LSTM cell [13].**

The LSTM cell is performing by the following formulations [11]:

$$\begin{aligned}
f_t &= \sigma(W_f * h_{t-1} + U_f * x_t + b_f)\\
i_t &= \sigma(W_i * h_{t-1} + U_i * x_t + b_i)\\
\widetilde{C}_t &= \tanh\left(W_c * h_{t-1} + U_c * x_t + b_C\right)\\
C_t &= f_t * C_{t-1} + i_t * \widetilde{C}_t\\
o_t &= \sigma(W_o * h_{t-1} + U_o * x_t + b_o)\\
h_t &= o_t * \tanh\left(C_t\right)
\end{aligned}$$

where $\sigma$ is the element-wise sigmoid function. $x_t$ is the input vector at time $t$. $h_t$ is the hidden state vector which storing useful information at time $t$. Weight matrices are denoted by $U_i$, $U_f$, $U_c$, $U_o$ for input $x_t$ and $W_i$, $W_f$, $W_c$, $W_o$ for hidden state $h_t$. Bias vector is denoted by $b_i$, $b_f$, $b_c$, and $b_o$.

## 2.2 GRU Network

Gated Recurrent Unit or GRU is a variant of LSTM memory cell introduced in 2014 by Chung et al.[2]. GRU is a little bit simpler than LSTM, GRU has two gates: update gate $z$ and reset gate $r$. Figure 2 shows the structure of the GRU cell. The update gate helps the model to determine how much of the past information needs to be passed along to the future while with the reset gate helps to determine how much information has to be forgotten.The GRU cell is performing by the following formulations [3]:

$$\begin{aligned}
z_t &= \sigma(W_z * h_{t-1} + U_f * x_t)\\
r_t &= \sigma(W_r * h_{t-1} + U_i * x_t)\\
\widetilde{h}_t &= \tanh\left(W * [h_{t-1} * r_t] + W * x_t\right)\\
h_t &= (1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t
\end{aligned}$$

where the output from update gate with $z_t$ and the output from the reset gate is calculated with $r_t$ at the time $t$. $h_t$ is the vector which contains information for the current cell at the time $t$ and forwards this information into the network.

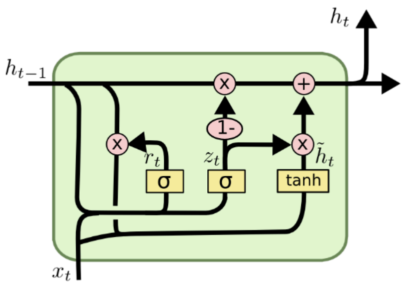


**Figure 2: The structure of the GRU cell [13].**

## 2.3 BI-LSTM / BI-GRU Network

In the task of NER, it would be nice to have access to both sides of input features, in our case sentences. Therefore, we using bidirectional LSTM/GRU network as proposed in Graves et al. (2013) [6]. The shortcoming of the layer with only LSTM cells is that it is only able to make use of the previous context of words and knowing nothing about the future. The idea of the bidirectional layer is using two separated hidden layers which are capture past and future information. In the end, the value of output is equal of concatenated these two hidden states. The graphic representation of bidirectional RNN is presented in Figure 3. Using bidirectional layer we have access to the long-range context in both input directions.

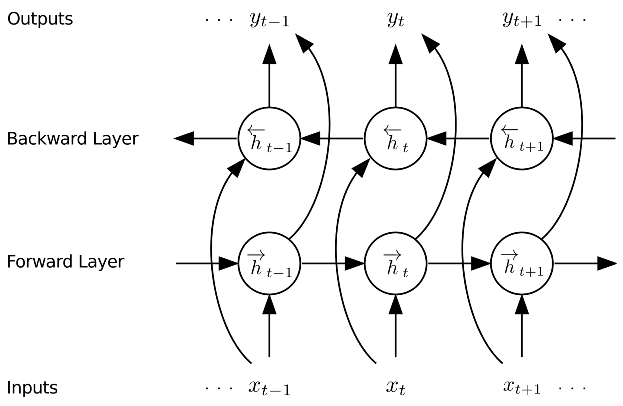


**Figure 3: Bidirectional layer of Recurrent Neural Network [5].**

## 3. TRAINING

In this section, we will present details about the training procedure. Our project is created using the Python[3] programming language and popular library Keras[4] for implementing neural networks. For all models, we train our networks using the back-propagation algorithm with categorical cross-entropy loss function. For parameter optimization we using RMSprop optimizer with learning rate $\eta = 0.001$ as recommended by Keras library documentation. At the output layer of neural networks we using the Softmax activation function. The Softmax activation function calculates the probability distribution over all classes/entities and chooses the class with highest probability for each word. For a vector $z$ of dimensionality $D$, the Softmax function is defined as [10]:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \qquad \text{for j = 1,...,D}$$

To regularize our model we using dropout technique for reducing overfitting in neural networks [18]. We apply dropout on each input layer with 0.1 and spatial dropout after concatenating all inputs with 0.3. Hidden layers contain RNN cells which have own dropout rate and a number of LSTM or GRU units. Hidden layers are configurable, details are described below.
Model training and model structure are completely configurable with *json* file. The example of *json* file is displayed in Listing 1.

```
{
    "max_epochs": 10,
    "batch_size": 32,
    "save_path": "models/gru-words-pos-chars/",
    "inputs": "words-pos-chars",
    "embeddings_trainable": false ,
    "embeddings_type": "glove",
    "rnn_type": "GRU",
    "rnn_num_layers": 3,
    "rnn_bidirectional": true ,
    "rnn_hidden_size": 100,
    "rnn_dropout": 0.2,
    "model_name":"GRU-example"
  }
```

**Listing 1: Example of model configuration over json file.**

With *json* file we made the list of different models and run our experiments. Model training is configurable with a number of epochs and batch size for epoch training. The model structure is configurable with input features such as word embedding, part of speech and character embedding. Hidden layers are configurable with RNN cell type, LSTM or GRU, number of cells in the layer, number of hidden layers, use of bidirectional layers and the dropout rate for each RNN layer.

[3]https://www.python.org/
[4]https://keras.io/

## 4. EXPERIMENTS

In this section will describe the dataset for evaluation and our experiments which include graphs and results.

### 4.1 Data Set

We test our model on dataset intended for named entity recognition. The dataset was introduced in 2003. On CoNLL conference where was the shared task the language-independent named entity recognition. The dataset was concentrate on four type of named entities: persons (PER), locations (LOC), organizations (ORG), names of miscellaneous entities (MISC) that don't belong to previous three groups and no entity token (O). The line of the dataset contains four fields: the word, part of speech tag, chunk tag, and name entity tag. Our focus was on name entity tag and the format of named entity tag is IOB (Inside, Outside, Beginning) where tokens are labeled as B-*label* if the token is the beginning of a named entity, I-*label* if it is inside a named entity but the first token within the named entity, or O otherwise. The tagging scheme is the IOB scheme originally presented by Ramshaw et al. (1995) [14]. The dataset was contain training, testing and validation data for two languages, English and German [16]. In this paper, we pick the English language.
The statistics of the dataset are shown in Table 1. We use the true data, without any pre-processing.

**Table 1: Dataset statistic, where columns Token and Sentence refers to the number of tokens and sentences in the CoNLL 2003 dataset.**

| Dataset | Token | Sentence |
|---|---|---|
| Training | 204567 | 14987 |
| Validation | 51578 | 3466 |
| Testing | 46666 | 3684 |

### 4.2 Features

In this paper, we use three types of additional features to help our models to improve. To most important features is Word Embedding. The main problem of using methods of deep learning in processing natural language is how to convert word into numbers. Word embedding is the technique of creating a language model which mapping words from the vocabulary into a corresponding vector of real numbers. We tried randomly initialized word vectors and pre-trained GloVe vectors with 100 dimensions. The results of using GloVe vectors are significantly better than in the case of the random initialization.
Word embedding is able to capture syntactic and semantic information but an example of task like NER is very useful to have morphological and shape information about the word. A lot of NLP system with additional character-level features are reported better results [20]. Based on the good experience from other articles, we apply character-level input to one layer with LSTM cells and we notice improvements in results.

### 4.3 Results

We were running our two experiments with four different models (LSTM, GRU, BI-LSTM, BI-GRU). The first exper-

iment was launched without any additional features and the second experiment was launched with all additional features: pre-trained word embedding, part of speech information and character embedding. All neural network structure contains 3 hidden layers with 100 LSTM/GRU cells and dropout rate 0.2. Neural networks were trained on 10 epoch with batch size 32. Results showed in Tables 2 and 3 are obtained by using macro-averaging. A macro-average computing the metric independently for each class and then take the average, whereas a micro-average aggregate all classes and then compute the average. The main reason for using macro-average is a large number of tag $O$, with micro-average, we would have unfair results.
The results shown in Table 2 are the results of models without any additional features. The best result of the first experiment has a BI-GRU where $F_1$ score result is 78.61%. Beside pre-trained word embedding and character embedding, we use the part of speech (POS) information for additional features.
On the second experiment where we've included all additional features like pre-trained word embedding, part of speech information and character embedding we've got the results shown in Table 3. The best result of the second experiment has BI-GRU where $F_1$ score result is 86.04%, however, the model BI-LSTM is very close with $F_1$ score result is 84.93%.

**Table 2: Performance of our first experiment without any additional features evaluated on the test set presented by evaluation metrics: Precision, Recall, and $F_1$ score.**

| Model | Precision | Recall | F1 |
|---|---|---|---|
| LSTM | 0.6487 | 0.7527 | 0.6821 |
| GRU | 0.6985 | 0.7589 | 0.7161 |
| BI-LSTM | 0.7995 | 0.7098 | 0.7414 |
| BI-GRU | 0.8529 | 0.7372 | 0.7861 |

**Table 3: Performance of our second experiment with additional features like word-embedding, POS information, and character-embedding, evaluated on the test set presented by evaluation metrics: Precision, Recall, and $F_1$ score.**

| Model | Precision | Recall | F1 |
|---|---|---|---|
| LSTM | 0.8123 | 0.8162 | 0.8138 |
| GRU | 0.8085 | 0.8071 | 0.8056 |
| BI-LSTM | 0.8408 | 0.8616 | 0.8493 |
| BI-GRU | 0.8610 | 0.8604 | 0.8604 |

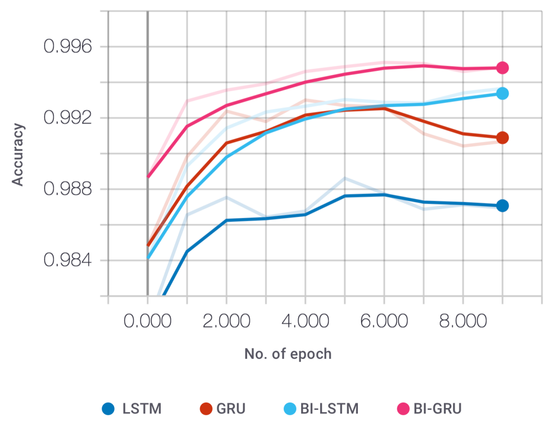


**Figure 4: Epoch-by-epoch model evaluation on the validation set with accuracy metric where the model without any additional features is used. LSTM is shown with the dark blue, GRU with the red, BI-LSTM with the light blue, and BI-GRU with the pink line.**

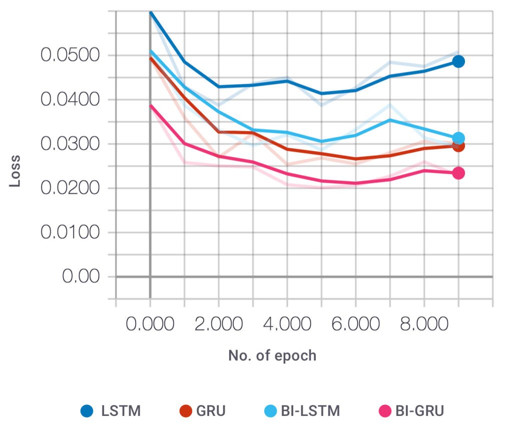


**Figure 5: Epoch-by-epoch result of loss function calculated on the model without any additional features. LSTM is shown with the dark blue, GRU with the red, BI-LSTM with the light blue, and BI-GRU with the pink line.**

## 5. CONCLUSIONS

In this paper, our focus was on classification and named entity recognition using neural network architectures. Our best results with model BI-GRU with additional features achieved satisfactory results compared with previously developed systems. We have achieved F1 score result 86.04%. There is a lot of space for improvement in the future. For example, it's necessary to explore the Conditional Random Fields method in cooperation with neural networks output layer which is used in some known NER systems. This would be the good start point for future work.

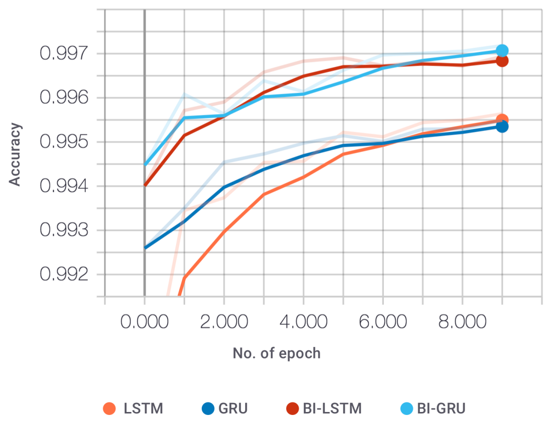


**Figure 6: Epoch-by-epoch model evaluation on the validation set with accuracy metric where the model with all additional features is used. LSTM is shown with the orange, GRU with the dark blue, BI-LSTM with the red, and BI-GRU with the light blue line.**

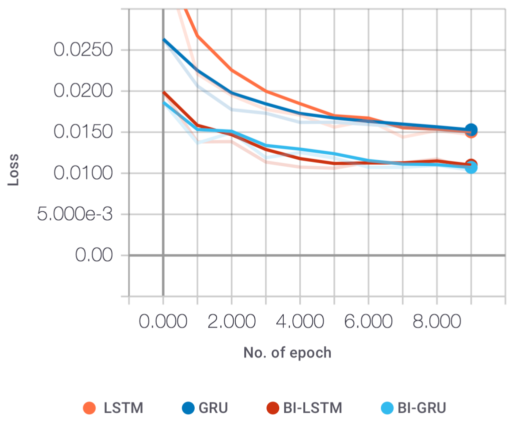


**Figure 7: Epoch-by-epoch result of loss function calculated on the model with all additional features. LSTM is shown with the orange, GRU with the dark blue, BI-LSTM with the red, and BI-GRU with the light blue line.**

## 6. REFERENCES

[1] H. L. Chieu and H. T. Ng. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 160–163, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[2] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[3] R. Dey and F. M. Salem. Gate-variants of gated recurrent unit (GRU) neural networks. *CoRR*, abs/1701.05923, 2017.

[4] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 168–171, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[5] A. Graves, N. Jaitly, and A. rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *In IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU*, 2013.

[6] A. Graves, A. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

[7] J. Hammerton. Named entity recognition with long short-term memory. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 172–175, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. 9:1735–80, 12 1997.

[9] Z. Huang, W. Xu, and K. Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

[10] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[11] X. Ma and E. H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354, 2016.

[12] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[13] C. Olah. Understanding lstm networks. 2015.

[14] L. A. Ramshaw and M. P. Marcus. *Text Chunking Using Transformation-Based Learning*, pages 157–176. Springer Netherlands, Dordrecht, 1999.

[15] J. Rowley. The wisdom hierarchy: representations of the dikw hierarchy. *Journal of Information Science*, 33(2):163–180, 2007.

[16] E. F. T. K. Sang and F. D. Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0306050, 2003.

[17] A. Severyn and A. Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 959–962, New York, NY, USA, 2015. ACM.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014.

[19] P. Wang, Y. Qian, F. Soong, L. He, and H. Zhao. Part-of-speech tagging with bidirectional long

short-term memory recurrent neural network. 10 2015.

[20] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *CoRR*, abs/1708.02709, 2017.

[21] X. Zhang, J. J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626, 2015.

# Context-dependent chaining of heterogeneous data

[Extended Abstract]

Rok Gomiščček
University of Ljubljana,
Faculty of Computer and Information Science
Večna pot 113,
1000 Ljubljana, Slovenia
rok.gomiscek@fri.uni-lj.si

Tomaž Curk
University of Ljubljana,
Faculty of Computer and Information Science
Večna pot 113,
1000 Ljubljana, Slovenia
tomaz.curk@fri.uni-lj.si

## ABSTRACT

A typical heterogeneous data set may contain different types of objects, where a small subset of direct relations among object types is supported by data. In our research we develop methods to predict the relation between indirectly related object types by chaining connections on intermediate object types, and methods to discover contexts of indirect relations.

A chain $\mathcal{C}_{XZ}$ is a sequence of relations that connect object types $X$ and $Z$, while $\mathcal{C}_{XZ|T}$ specifies that $X$ and $Z$ are connected through objects of type $T$. Two object types can have multiple parallel chains connecting them, each chain providing a different prediction. Different chains might result in different predictions, each correctly describing the relation, but in a different context.

We wish to ensure that different chains between two indirectly connected object types result in similar predictions, but only when sharing the same context. When predicting a new connection between object types using different paths between them, we might obtain different predictions. Our proposal on how to deal with this is to group predictions of different chains together based on their similarity.

First we will propose a method to improve the results of chaining matrices on different paths between two indirectly connected object types. To predict the relation between indirectly connected object types matrices on a given chain are multiplied and then all predictions can be combined together. Transitivity of relations tells us that two objects that belong to two indirectly connected object types are connected if they connect to intermediate objects on some path. The number of intermediate objects between two objects can vary greatly, but that number is not taken into account. We will propose an approach to normalize the predictions by accounting for the number of different paths connecting objects from the two object types.

Since parallel chains between two indirectly connected object types can return different relations, we can see each chain representing a different context which presents a different kind of relation. For example, the structure of the protein-protein interaction networks, which are used to represent functional relations among gene products, may vary greatly across different tissues. We propose to group parallel chains according to their contexts and change the optimization criteria so that similar chains will be clustered together. We will evaluate various clustering methods, such as k-means, k-NN and hierarchical clustering. We will also explore various possible data representations for clustering, such as original data, approximations and product of factor matrices on a path.

We will validate our model on synthetic and real data. We will create synthetic data where we will guarantee that the target relation between indirectly connected object types can be predicted by a chain of connecting relations. We will generate the data by creating relations between object types where transitivity can be applied, that is, if $x_k \mapsto v_l$ and $v_l \mapsto z_n$ are true then $x_k \mapsto z_n$ must also be true, where $\mapsto$ denotes a function that maps objects to objects. This value will be used to evaluate the predictions of our method by computing the squared Euclidean distance between the actual data and the prediction. Some values will be removed from the data to simulate missing values and used for evaluation. We will also validate our model on real data where there are multiple parallel paths between two indirectly connected object types. We can find these kind of data in biological data sets with relations connecting genes, proteins, diseases, gene function annotation (gene ontology terms), pathways, and other biological entities.

## Keywords

machine learning, matrix factorization, data fusion, chaining, transitivity

StuCoSReC

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