ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Gediminas Kavaliauskas

Informatikos magistro specialybės II kurso dieninio skyriaus studentas

Dirbtinio intelekto atpažinimo metodų analizė ir taikymai ranka rašyto teksto atpažinimui

The Analysis of Recognition Methods Based on Artificial Intelligence and their Application in Handwritten Text Recognition

Magistro darbas

Darbo vadovas:

Lekt.G. Felinskas

Recenzentas:

Doc. S. Turskienė

Šiauliai, 2012

„Tvirtinu, jog darbe pateikta medžiaga nėra plagijuota ir paruošta naudojant literatūros sąraše pateiktus informacinius šaltinius bei savo tyrimų duomenis“

Darbo autorius _____

(vardas, pavardė, parašas)

Pagrindinis darbo tikslas yra sukurti ranka rašyto teksto atpažinimo įrankį, pritaikant dirbtinio intelekto algoritmus.

Siekiant šio tikslo iškelti uždaviniai:

1. Apžvelgti dirbtinio intelekto atpažinimo metodus;
2. Išanalizuoti ir palyginti ranka rašyto teksto atpažinimo algoritmų efektyvumą;
3. Sukurti programą, skirtą ranka rašyto teksto atpažinimui;
4. Pasitelkiant sukurto įrankio testų rezultatus, atlikti darbo srities probleminių sričių analizę.

Darbo vadovas _____

(vardas, pavardė, parašas)

TURINYS

ĮVADAS

Dirbtinis intelektas – mokslo šaka, kurios pagrindinis tikslas suprojektuoti ir sukurti tokius protingus mechanizmus ir programas, kurių veikimas imituotų žmonių mastymą. Viena iš dažniausiai pasitaikančių šios mokslo šakos probleminių sričių yra atpažinimo metodų (angl. *pattern recognition*) uždavinių sprendimas. Ši probleminė sritis dar skyla į tokias atskiras dalis: spausdinto teksto atpažinimas (angl. *optical character recognition*), rašytinio teksto atpažinimas (angl. *handwriting recognition*), kalbos atpažinimas (angl. *speech recognition*) ir veidų atpažinimas (angl. *face recognition*). Programos, sukurtos remiantis atpažinimo metodu pagrindu, bendrauja su aplinka ir gautą informaciją iš jos lygina su duomenų bazėje esančiais šablonais. Po palyginimo, priklausomai nuo programos paskirties, ji imituoja tam tikrus veiksmus, kuriuos gali atlikti į tą pačią aplinką žiūrintis žmogus. Tokių programų poreikis atsiranda kai susiduriame su dideliais informacijos kiekiais, kuriuos norint apdoroti gali prireikti daug laiko ar ne vieno žmogaus pastangų. Pavyzdžiui, filmuotoje medžiagoje surasti tam tikrą asmenį, surūšiuoti didžiules nuotraukų kolekcijas pagal tam tikrus bruožus, įvairius rankraščius paversti kompiuteriniu tekstu ir panašiai.

Taip pat tokių programų poreikį didina, sparčiai populiarėjantys ir tobulėjantys, įvairūs mobilieji įrenginiai. Jie mums suteikia įvairių galimybių, apie kurias anksčiau niekas net negalėjo pagalvoti. Dar visai neseniai tokius darbus kaip žmonių, objektų, garsų ar teksto atpažinimas, galėjo atlikti tik galingi kompiuteriai su specialia ir brangiai kainuojančia programine įranga, o dabar tai galime atlikti tiesiog mūsų mobiliųjų telefonų pagalba.

Šiame darbe bus susitelkta į vieną iš minėtų sričių, kuri turi dvi panašias ir tuo pačiu skirtingas pagal savo taikymo sritį šakas, tai ranka rašyto teksto atpažinimas. Viena iš šakų yra realiu metu rašomo teksto atpažinimas (angl. *online recognition*). Šios šakos įrankiais naudojamasi gan plačiai įvairiuose mobiliuosiuose įrenginiuose. Šie įrankiai yra labai ištobulinti ir pasiekia aukštą patikimumo procentą. Tuo tarpu antroji šaka minima gan retai, tai ranka rašyto teksto atpažinimas iš statinio vaizdo (angl. *offline recognition*). Ši sritis turi nemažai problemų[14], pagrindinės jų yra: parašyto teksto išskaidymas atskirais simboliais, kompiuterinės sistemos apmokymas pagal tam tikrą raštą ir galiausiai paties teksto atpažinimas. Šioms problemoms spręsti galima taikyti įvairius metodus. Dėl tos priežasties, viena iš šio darbo dalių yra veikiančio įrankio sukūrimas, kurio pagalba bus galima atlikti teksto atpažinimą. Sukurto įrankio pagalba bus atliekami tyrimai, kurių dėka išskirsime pagrindines problemines sritis, dėl ko tokie įrankiai naudojami gana retai ir pateiksime apie tai išvadas bei galimus problemų sprendimo variantus.

Pagrindinis darbo tikslas yra sukurti ranka rašyto teksto atpažinimo įrankį, pritaikant dirbtinio intelekto algoritmus.

Siekiant šio tikslo iškelti uždaviniai:

1. Apžvelgti dirbtinio intelekto atpažinimo metodus;
2. Išanalizuoti ir palyginti ranka rašyto teksto atpažinimo algoritmų efektyvumą;
3. Sukurti programą, skirtą ranka rašyto teksto atpažinimui;
4. Pasitelkiant sukurto įrankio testų rezultatus, atlikti darbo srities probleminių sričių analizę.

1. ANALITINĖ DALIS

1.1. RANKA RAŠYTO TEKSTO ATPAŽINIMO POREIKIO ATSIRADIMAS

Kiekvieno žmogaus raštas yra jo asmenybės dalis, kurią tyrinėja atskira mokslo šaka „grafologija“. Žmogaus raštas nusako žmogaus charakterį, jo nuotaiką ir dar daugelį kitų jo savybių. Tačiau atėjus laikui kai kiekviename mūsų žingsnyje mus lydi įvairios technologijos, neretas žmogus vis rečiau naudojasi šia savo asmenybės dalimi, nes jos paprasčiausiai nebereikia. Tačiau ne visi nori su tuo sutikti ir net technologijų pilname pasaulyje nori informacija dalintis ranka parašyto teksto pavidalu. Dėl šios priežasties informacija dalinantis tokiu būdu atsiranda poreikis tą informaciją paversti kompiuteriui suprantamu simboliu rinkiniu.

Žmonių noras panaudoti ranka rašytą tekstą įvairių įrenginių pagalba siekia net 1888 metus, būtent tais metais išduotas patentas įrenginiui kuris pavadintas teleautografu (angl. *Teleautograph*)[2]. Šio įrenginio tikslas ant specialios planšetės parašytą tekstą perduoti į kitą tokį pat įrenginį, kuris sugebėtų atkurti tą patį vaizdą kuris buvo suformuotas anksčiau. Kitaip tariant, teleautografas tai tos pačios telegramos, tik vietoje spausdintų simbolių yra perduodamas ranka rašytas tekstas. 1915 metais buvo užpatentuotas pirmasis įrenginys, kuris jau turėjo galimybę atpažinti rašomus simbolius analizuojant atliekamus veiksmus[3].

Tai buvo tik pradžia įrenginių raidos, kurie leido mums informaciją pateikti ar perduoti rašant ranka. Visi įrenginiai kuriais galime naudotis dabar perėjo ilgą kelią ir ne kartą buvo tobulinami (žr. 2 priedas). Visą tai parodo koks žmogui svarbus šis informacijos pateikimo būdas.

1.2. RAŠYTINIO TEKSTO ATPAŽINIMO GALIMYBĖS ŠIUOLAIKINIUOSE ĮRENGINIUOSE

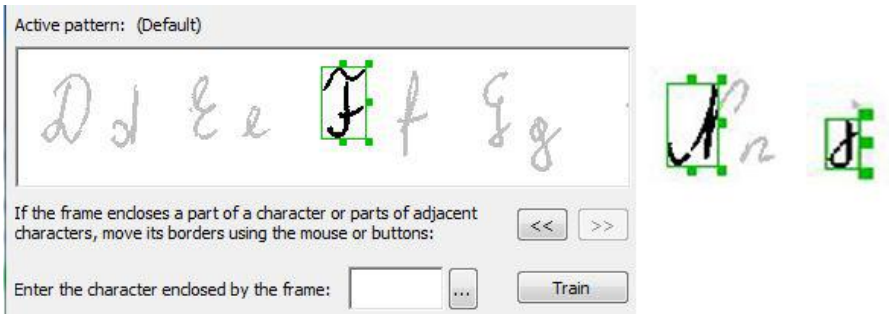
Technologijoms sparčiai žengiant pirmyn, visi įrenginiai tampa vis mažesni, o jų techninės specifikacijos vis geresnės. Praėjus beveik 20 metų nuo pirmųjų planšetinių kompiuterių, kišeninių kompiuterių ir išmaniųjų telefonų pasirodymo, dabar jų faktiškai nebegalima atpažinti. Visų jų dydžiai ir svoriai sumažėjo dvigubai ar net trigubai, o techninės charakteristikos pirmykščius įrenginius lenkia jau tūkstančiais kartų. Tačiau ranka rašyto teksto atpažinimas išlieka aktualia problema ir dabartiniuose įrenginiuose. Šią funkcionalumo

trūkumo problemą sprendžia operacinių sistemų kūrėjai, kuriems į pagalbą ateina ir įvairūs programinės įrangos gamintojai. Tačiau tiek vieni, tiek kiti susiduria su rašytinio teksto atpažinimo sistemos apmokymo poreikiu. Tai yra, pradžioje sistema turi susieti kiekvieną jūsų rašomą simbolį su kompiuteriniu atitikmeniu. Neretai nebereikia net apmokyti teksto atpažinimo sistemos, nes įdiegiant tokią sistemą į savo įrenginį, būna įdiegiamos ir duomenų bazės su jose esančiais rašytiniais simbolių šablonais.

Šiuo metu labiausiai išsiskirianti, teksto atpažinimo galimybę teikianti programinė įranga yra „MyScript Stylus“, sukurta kompanijos „Vision Objects“. Ši programinė įranga veikia visose šiuo metu populiariausiose operacinėse sistemose (Linux, Mac OS, Windows, Android, Symbian ir kt.) ir ranka rašytą tekstą sugeba atpažinti 87 kalbomis. Verta paminėti, kad ši programa naudojasi teksto atpažinimo metodu, kai yra sekamas kiekvienas simbolio rašymo metu atliekamas judesys. Dar viena „MyScript Stylus" savybė yra jos nuolatinis mokymasis. Kaskart naudojantis šia programa ji renka informaciją apie jūsų rašomus simbolius ir jų rašybos stilių, o jei atpažintas simbolis buvo klaidingas, galima programai tai nurodyti, po ko ši iš savo duomenų bazės pašalina sąsają su pritaikytu šablonu. Tokiu būdu ilgiau pasinaudojus šia sistema galima pasiekti beveik 100% teksto atpažinimą[9]. Šiuo metu panašaus veikimo principo programų galima rasti ir daugiau. Net naujausioje Microsoft kompanijos operacinėje sistemoje „Windows 7“ skirtoje planšetiniams kompiuteriams, naudojama sistema kuri pagrįsta apmokymu, iš ko galime teigti jog tokių technologijų naudojimas yra efektyvus[10]. Tačiau tai taip pat yra sistema, kuri atpažinimą vykdo tuo metu kai tekstas yra rašomas.

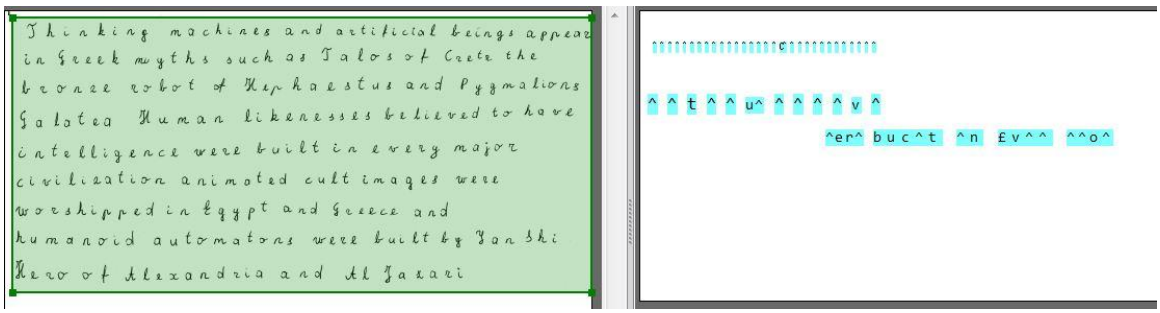
Kaip matome realiojo laiko rašytinio teksto atpažinimo metodas yra naudojamas plačiau. Tuo tarpu rasti efektyvų teksto atpažinimo įrankį, kuris atpažinimą darytų iš nuotraukos o ne realiu laiku, sudėtinga. To priežastis – daug laiko reikalaujantis tokių sistemų realizavimas, kurio metu reikia priimti nemažai sudėtingų sprendimų[14]. Pavyko surasti tik vieną įrankį kuris siūlytų tokio metodo atpažinimą. Tai „ABBYY Finereader 11“, kuris iki šiol buvo naudojamas tik spausdinto teksto atpažinimui. Tačiau paskutinėse šios programinės įrangos versijose gamintojai pridėjo papildomą funkcionalumą, leidžiantį programą apmokyti pagal raštą ir taip atlikinėti ranka rašyto teksto atpažinimo užduotis.

Programos apmokymui pateikiama angliška abėcėlė su didžiosiomis ir mažosiomis raidėmis. Apmokymo procesas atliekamas principu, kad kiekvienam surastam simboliui turime įvesti jo atitikmenį (1 pav.).



1 pav. Finereader programos apmokymas (kairėje), klaidos atpažįstant simbolių ribas (dešinėje)

Tai nesudėtingas procesas, tačiau atliekant jį iškarto pastebėta, kad programa kartais netiksliai nustato simbolio ribas, kurios vaizduojamas žaliu keturkampiu. Tai pataisoma, nes sistema leidžia keisti tas ribas rankiškai. Tačiau buvo atveju kai net rankiškai neįmanoma apibrėžti visos raidės (žr. 3 priedas). To pasėkoje nebuvo įmanoma sistemą apmokyti visomis abėcėlės raidėmis. Apmokius sistemą, pradžioje buvo bandoma atpažintį tą pačią abėcėlę iš kurios buvo atliekamas apmokymas. Tačiau to nepavyko padaryti. Programa vis gražindavo tuščią lapą lyg nerastu nei vieno simbolio. Nuspręsta sistemą apmokyti dar keliais duomenų rinkiniais ir tada pakartoti testus. Gauti rezultatai dviprasmiški. Vienu atveju kai buvo bandoma atpažinti abėcėles, atpažįstama buvo daugiau nei pusė raidžių, tačiau kitu atveju (2 pav.), programai pavyko atpažinti tik kelias raides.



2 pav. Atpažinimo testui naudojamas paveikslėlis ir rezultatas

Panašūs rezultatai gauti ir su kitais testavimui paruoštais šablonais. Daugiau rezultatų kurie gauti su programa „Finereader“ pateikiami prieduose (žr. 3 priedas).

1.3. TEKSTO ATPAŽINIMO METODAI

Spausdinto teksto atpažinimas ir rašytinio teksto atpažinimas yra vienos dirbtinio intelekto atpažinimo metodų šakos probleminės sritys. Jos yra nemažai kuom panašios, o tuo pačiu ir labai skirtingos. Pavyzdžiui spausdinto teksto atpažinimo technologijos jau taikomos daug seniau nei rašytinio teksto atpažinimo, todėl nemažai teksto atpažinimo metodų ir technologijų perimta būtent iš spausdinto teksto atpažinimo šakos. Nepaisant to pagrindinis

skirtumas tarp šių technologijų yra tas, kad kitaip nei spausdintame tekste, rašytiniame tekste kiekvienas simbolis gali turėti begalo daug variacijų[11]. Tai yra, kad vienas ir tas pats simbolis, net ir parašytas to paties asmens, vieną kartą gali atrodyti vienaip, o kitą kartą kitaip. Atsižvelgiant į tai, kad kiekvienas žmogus dar turi savitą rašymo stilių, kuris įneša dar didesnę simbolių atvaizdavimo įvairovę, mes susiduriame su problema, kad programos turinčios apdoroti šitokį tekstą, turi gebėti apdoroti ir atpažinti šimtus ar net tūkstančius vienos raidės vaizdavimo būdų.

Dar vienas spausdinto ir rašytinio teksto atpažinimo skirtumų yra tai, jog spausdintas tekstas apdorojamas iš statinio vaizdo. Tai yra, teksto atpažinimo programa gauna visą teksto vaizdą, dažniausiai paveikslėlio pavidalu, ir tik tada pasitelkiant įvairias technologijas ir algoritmus yra vykdomas spausdintų simbolių vertimas kompiuteriniu formatu. Tuo tarpu, rašytinio teksto atveju susiduriama su keliais variantais. Teksto atpažinimu iš paveikslėlio, bei teksto atpažinimu pagal tai kaip jis buvo rašomas realiu metu, išsaugant kiekvieno judesio koordinates[12].

1.3.1. REALIU LAIKU RAŠOMO TEKSTO ATPAŽINIMO METODAS

Kaip jau buvo minėta anksčiau, šis rašytinio teksto atpažinimas pasižymi tuo, kad kai simbolis yra rašomas, sistema renka papildomus duomenis apie visą rašomo simbolio procesą ir tai padeda lengviau atpažinti ir išskirti kas buvo parašyta. Yra išskiriamos penkios pagrindinės tokios sistemos savybės[13]:

- Viskas vyksta realiu laiku ir įvesti simboliai yra iškart atpažįstami;
- Galima teksto atpažinimo adaptacija realiu laiku, tai yra rašantis asmuo iškart mato kokį simbolį atpažino sistema ir neteisingo atpažinimo atveju gali jį pakeisti;
- Teksto įvedimo metu yra renkama papildoma informacija apie rašiklio koordinates, kiekvieno judesio eiliškumą, kryptį ir greitį;
- Nereikia atlikti daug papildomų veiksmų prieš vykdant simbolio atpažinimo procesą;
- Lengva atskirti simbolius vieną nuo kito remiantis informaciją apie rašiklio atitraukimą nuo rašymo lentos.

Remiantis šiomis savybėmis galime išskirti vieną pagrindinį šio metodo privalumą. Kadangi nereikalingas papildomas įvesto simbolio apdorojimas, sutaupoma daug laiko ir atminties teksto atpažinimo procesui atlikti savo darbą. Tačiau čia pat atsiskleidžia ir

pagrindinis šio metodo trūkumas. Tekstas yra rašomas po vieną simbolį, o tai priklausomai nuo asmens rašymo stiliaus, gali sukelti diskomfortą tais atvejais kai žmogus yra įpratęs žodžius rašyti suliejant vieną su kitu. Taip pat atsisakius papildomų teksto atpažinimo operacijų, smarkiai apribojamos efektyvaus teksto atpažinimo galimybės. Tai yra norint kad sėkmingai būtų atpažintas tekstas, vartotojas privalo kiekvieną simbolį rašyti pagal tam tikras taisykles, kurios nustatomos kuriant teksto atpažinimo algoritmą arba prieš tai turi apmokyti teksto atpažinimą atliekančią sistemą savo raštu.

1.3.2. RAŠYTINIO TEKSTO ATPAŽINIMO METODAS IŠ NUOTRAUKOS

Priešingai nei realiu metu rašyto teksto atpažinimo atveju, sistemos dirbančios su statiniu vaizdu, prieš pradėdamos simbolio atpažinimo procedūrą neturi jokios papildomos informacijos apie parašytus simbolius. Tai reiškia, kad norint pradėti teksto atpažinimą turi būti atlikta tam tikrų veiksmų seka. Ši seka pradedama išankstinio apdorojimo operacija (angl. *preprocessing*), kurios metu dažniausiai atliekami tokie veiksmai:

- Teksto atvaizdas išvalomas nuo triukšmo, tai yra pašalinami atsitiktiniai pikseliai kurie galėjo atsirasti perkeliant atvaizdą iš lapo popieriaus į kompiuterį, jį skenuojant;
- Atliekama vaizdo aštrinimo (angl. *sharpening*) procedūra, kurios pagalba išryškinami simbolių kontūrai;
- Atliekamas teksto ploninimas, kurio metu parašytame tekste siekiama palikti tik pagrindinius pikselius kurie reikalingi teksto atpažinimui.

Šių atliekamų veiksmų paskirtis, priklausomai nuo vėliau planuojamo taikyti teksto atpažinimo metodo, yra įvairi. Vienu atveju tiesiog siekiama palengvinti ir pagerinti simbolių atpažinimo tikimybę, kitu atveju siekiama sumažinti duomenų kiekį, kurį vėliau turės apdoroti teksto atpažinimo algoritmas. Taip yra sutaupomas ir reikiamos atminties kiekis, ir laikas reikalingas atlikti visą procesą.

Kita operacija reikalinga teksto atpažinimui yra segmentacija. Jos metu paprasčiausiai siekiama iš vientiso teksto išgauti, atskiras jo eilutes, iš pastarųjų atskirus žodžius ir galiausiai atskirus simbolius, kuriuos jau po to galėtų apdoroti teksto atpažinimo algoritmas.

Pati pagrindinė operacija teksto atpažinimo procese yra bruožų išskyrimo. Būtent jos metu priklauso kokie algoritmai bus taikomi siekiant atpažinti parašytus simbolius ir kaip efektyviai tai bus atliekama.

Galiausiai lieka klasifikavimo operacija, tai yra pats simbolių palyginimas su duomenų bazėje turimais šablonais ir jų priskyrimas labiausiai tinkamam variantui.

Visos šios operacijos būtų bevertės jei nebūtų naudojami sprendimo priėmimo modeliai. Populiariausi ir dažniausiai naudojami yra genetiniai, paslėpti Markovo (angl. *hidden Markov*), Bajeso tinklų, dirbtinių neuroninių tinklų modeliai. Būtent pasirinktas modelis sudaro didžiausią įtaką tam kaip yra bandoma atpažinti ranka rašytą tekstą. Dabar vis dažniau tie modeliai dar yra apjungiami tarpusavyje, siekiant optimalesnių teksto atpažinimo sistemos rezultatų.

Nepriklausomai nuo to kokį teksto atpažinimo modelį naudosime, visos teksto atpažinimo sistemos turi būti „apmokytos“. Tai vienas iš reikšmingiausių dirbtinio intelekto sistemų bruožas, be kurio nebūtų įmanoma gauti jokių rezultatų. Apmokymas, tai pradinių šablonų ar taisyklių rinkiniai, kurie padeda sistemai reaguoti į tam tikrus veiksnius. Teksto atpažinimo atveju apmokymo duomenys yra simbolių šablonai, pagal kuriuos bandoma nustatyti kokia yra analizuojama raidė. Didžiausia problema su kuria susiduria apmokytos sistemos yra tas, jog dažniausiai šabloniniai duomenys yra tik tam tikro asmens, o tai reiškia kad atpažįstant rašytinį tekstą kurį parašė kitas žmogus bus sunku gauti teisingus rezultatus.

1.4. OCR, ICR, IWR

Priklausomai nuo to kokį tekstą norime atpažinti ar nuo to kaip mes norime vykdyti teksto atpažinimą iš paveikslėlio, šis dar skaidomas į 3 kategorijas:

- OCR (Optical character recognition) – spausdintų simbolių atpažinimas;
- ICR (Intelligent character recognition) – ranka rašytų simbolių atpažinimas;
- IWR (Intelligent word recognition) – ranka rašytų žodžių atpažinimas.

Teksto atpažinimo ir vertimo į redaguojamus kompiuterinius dokumentus uždaviniai sprendžiami jau virš 30 metų, tačiau ir net dabar dar nėra sukurtos tokios programos ar algoritmo, kurie teksto atpažinimą atliktų 100% tikslumu. Pirmoji taikymo sritis šių uždavinių sprendimui yra spausdintų simbolių atpažinimas (OCR). Ji yra labiausiai ištobulinta ir atpažįstant geros kokybės nuskenuotą dokumentą yra pasiekiamas 99,8% tikslumas[16].

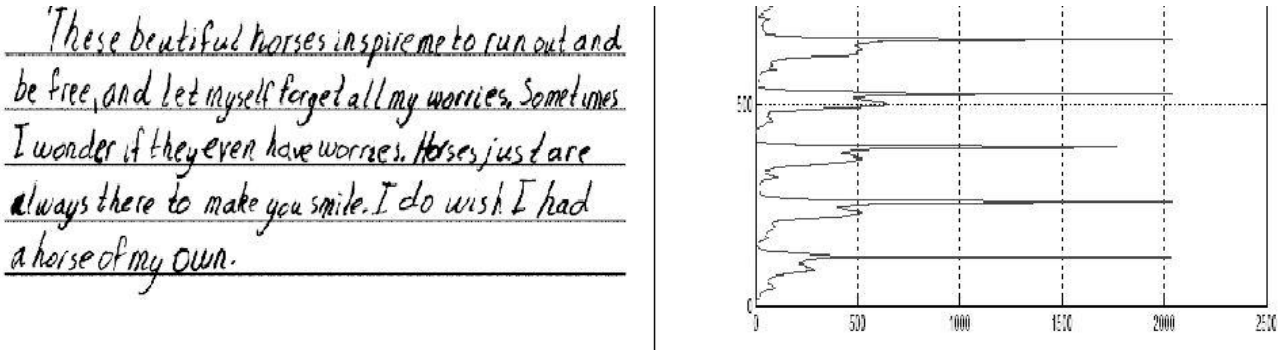
1.5. SEGMENTACIJA

Bene didžiausia probleminė sritis rašytinio teksto atpažinime yra segmentacija. Nuo jos priklauso ar teisingai buvo atskirti simboliai kurie perduoti tolimesniam apdorojimui. Priešingu atveju net ir optimaliausias teksto atpažinimo algoritmas būtų nieko vertas. Dėl šios priežasties būtent segmentacija kelia didelį susidomėjimą mokslininkams.

Atliekant teksto segmentaciją iškyla nemažai problemų priklausančių nuo žmogaus rašymo stiliaus. Jas galima padalinti į tokias dalis[14]:

- Teksto eilutės išskyrimas iš teksto – ši dalis sukelia sunkumų tuomet kai teksto eilutės yra viena kitos labai arti ir vienų raidžių dalys papuola į kitas eilutes;
- Žodžio atskyrimas iš eilutės – problemų išskirti žodžius atsiranda tada kai to paties žodžio raidės ne visada yra sujungiamos;
- Didžiųjų ir mažųjų raidžių skirtumo atpažinimas – problema galinti iškilti rašant tokias raides kaip c,o,s,u.
- Sakinio pradžios ir pabaigos aptikimas.

Taigi pirmas žingsnis atliekant teksto segmentaciją yra eilutės išskyrimas. Šiam veiksmui populiaru naudoti pikselių skaičiaus nustatymą eilutėse. Pasinaudojus šiuo veiksmu aiškiai matosi kur yra tarpai tarp eilučių (3 pav.). Tačiau net pasinaudojus šiuo algoritmu mes neišvengiamai susidursime su situacija kai eilutėse atsiranda simbolių iš kitų eilučių dalys. Atsikratyti šios problemos sprendimui pritaikomi specialūs filtrai, kurie aptinka tokias raidžių liekanas ir jas pašalina.



3 pav. Pikselių skaičiaus eilutėje nustatymas

Atliekant žodžių atskirimą eilutėje svarbu nustatyti kokį atstumą tarp simbolių mes laikysime vieno žodžio pabaiga o kito pradžia. Ši problema ypač akivaizdi pateiktame paveikslėlyje (3 pav.), kuriame matome jog kai kurie tarpai tarp raidžių yra net didesni nei

tarp skirtingų žodžių. Tokiu atveju paprasto sprendimo neužtenka ir segmentavimo operacijos metu reikia panaudoti papildomus mechanizmus[15].

Neretai žodžių padalinimui į simbolius naudojamas tas pats principas kaip teksto dalinimas į eilutes, tik šiuo atveju pikseliai yra skaičiuojami stulpeliais ir tose vietose kur pikselių skaičius yra mažas laikoma jog baigiasi vienas ir prasideda kitas simbolis.

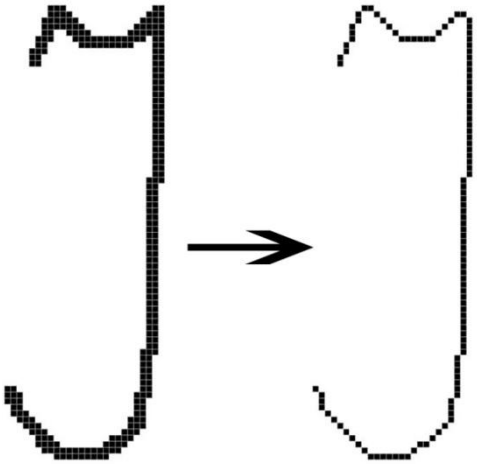
1.6. DIRBTINIO INTELEKTO METODAI

1.6.1. AŠTUONETAINIO GRAFO MOTODAS

Šio metodo esmė yra apdorojamo simbolio vertimas aštuonetainiu grafu, kur kiekvienas simbolio pikselis atitinka grafo viršūnę. Toks sudarytas grafas leidžia pasiekti jog parašytas simbolis įgauna tam tikrą formą, kuri nepriklauso nuo rašysenos stiliaus. Siekiant atlikti atpažinimo procedūrą, grafo viršūnėms ir viršūnes jungiančioms briaunoms, suteikiami svoriai. Taip pat būtina susižymėti visus grafo bruožus, tokius kaip kilpas, vertikalias linijas, horizontalias linijas, vingiuotas linijas ir taip pat jų svorius. Tai yra svarbu, nes kitaip būtų neįmanoma palyginti gautą grafą su duomenų bazėje esančiais raidžių šablonais. Atliekant atpažinimo procedūrą yra lyginami visi grafo bruožai su duomenų bazėje esančiais bruožais, o lyginimo rezultatas išreiškiamas panašumo koeficientu. Po to turimas grafas yra šiek tiek modifikuojamas pakeičiant jo bruožų svorius ir vėl ieškomas panašumo koeficientas. Galiausiai gavus norimą panašumo koeficientą, parašyto simbolio grafui yra priskiriama raidė. Remiantis šiuo metodu atlikto eksperimento rezultatais [11], gauta jog 82% atvejų bandant atpažinti parašytus simbolius, jie buvo nustatyti teisingai.

1.6.2. AŠTUONETAINIO GRAFO ALGORITMAS

Tarkime mes parašome raidę „I“, tada pritaikius aštuonetainio grafo algoritmą parašyto simbolio atpažinimas prasideda normalizacijos operacija. Šios operacijos metu įvestą simbolį bandoma „suploninti“ iki tiek kad visas jo kontūras būtų 1 pikselio storio. Koks gaunamas parašytos raidės vaizdas prieš ir po normalizacijos operacijos pateikiamas 4 paveikslėlyje.



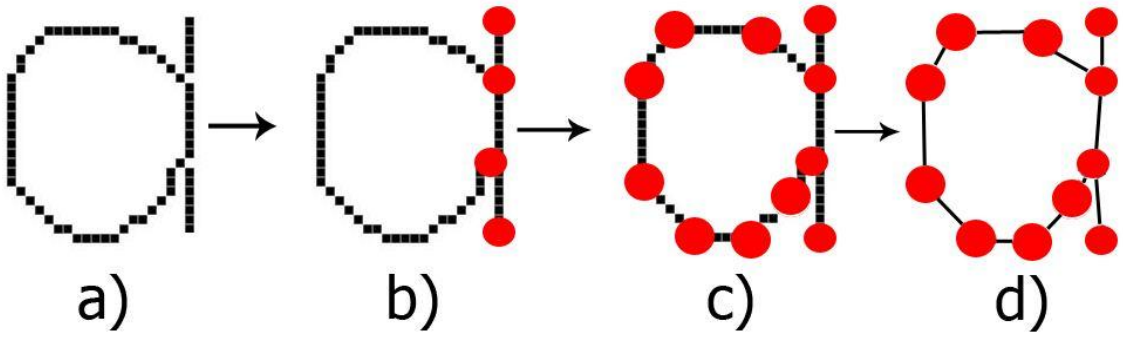
4 pav. I raidė prieš ir po normalizacijos

Turint normalizuotą raidės atvaizdą pereinama prie aštuonetainio grafo sudarymo. Šiame žingsnyje pabrėžiamos 2 pagrindinės savybės. Pirma, tai kad atstumas tarp grafo viršūnių turi būti pakankamai didelis ir taip galėtų atvaizduoti tam tikro simbolio savybes. Tuo pačiu atstumas tarp viršūnių neturi būti ir per daug didelis, kad užimtų kuo mažiau atminties. Antra savybė tai kryptys kuriomis gali būti jungiamos grafo viršūnės. Aštuonetainis grafas čia tinka dėl to jog 8 galimos kryptys pakankamai gerai atvaizduoja simbolio savybes ir tuo pačiu sumažina skirtingas to pačio simbolio interpretacijas.

Paties grafo sudarymo procesas susideda iš tokių dalių:

- Peržiūrimi visi normalizuoto parašytos raidės atvaizdo pikseliai ir jei aptinkamas toks kuris turi 1 arba nemažiau 3 kaimynų, pažymimas kaip viršūnė;
- Toliau peržiūrimi visi atvaizdo pikseliai pradedant nuo viršutinio kairio kampo. Jei surandamas pikselis kuriame keičiasi simbolio kontūro kryptis, jis pažymimas kaip viršūnė;
- Visos pažymėtos viršūnės sujungiamos briaunomis.

Vizualiai šis procesas parodytas paveikslėlyje 5, kurio b dalyje matome pažymėtas tik 4 viršūnes kurios turi 1 arba 3 kaimynus. C dalyje jau atsiranda viršūnės kurių pagalba būtų galima lengviau atvaizduoti raidės a savybes. Galiausiai sužymėjus visas viršūnes, jas sujungiame briaunomis ir gauname grafą pavaizduotą paveikslėlio d dalyje.



5 pav. a raidės vertimas į aštuonetainį grafą

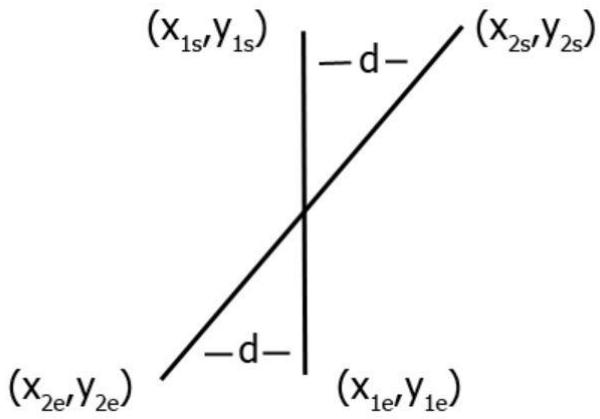
Tolimesnė operacija aštuonetainio grafo modelyje yra viršūnių ir briaunų svorių uždėjimas. Priskiriant viršūnių svorius svarbu atkreipti dėmesį kokį vaidmenį tam tikra viršūnė atlieka, tai yra ar pasirinkta viršūnė yra simbolio kilpoje, ar vingyje. Tuo tarpu briaunų svoriai priklauso nuo atstumo tarp viršūnių. Galiausiai yra suskaičiuojami visi briaunų ir viršūnių svoriai, suskaičiuojamos horizontalios ir vertikalios linijos ir ši informacija yra lyginama su šablonais esančiais teksto atpažinimo sistemoje. Kadangi kiekvieno žmogaus rašysena skiriasi ir jei atlikus surinktos informacijos palyginimą su sistemoje esančiais šablonais nerandamas reikiamas atitikmuo, atliekama vadinama savybių sutapatinimo operacija (angl. *feature matching*). Jos metu yra keičiami įvairūs parašyto simbolio svoriai, po ko vėl lyginamas simbolio panašumas su turimais šablonais. Visa tai yra atliekama tol kol nepasiekiamas norimas panašumo lygis tarp tokių lyginamų simbolių savybių kaip kilpų, vingių, horizontalių ir vertikalių linijų skaičius ir jų svoriai.

1.6.3. GENETINIS METODAS

Genetinio algoritmo panaudojimas leidžia išspręsti vieną iš didžiausių rašytinio teksto atpažinimo problemų. Šiuo metodu paremtose sistemose, ne taip kaip įprasta kituose metoduose, nebūtina didžiulė šabloninių duomenų bazė. Tai yra todėl, kad genetinio algoritmo principas suteikia galimybę iš kelių turimų duomenų variacijų gauti kitą duomenų rinkinį. Šis sugeneruotas šablonas papildo turimų problemos sprendimų aibę, tuo pačiu atsiranda galimybė pasinaudojus juo kurti dar kitas sprendimo variacijas, kol galiausiai gaunamas problemos sprendimas, kuris jau tenkina mūsų poreikius. Imant konkrečiai rašytinio teksto atpažinimo atvejį, genetinio algoritmo sugeneruoti nauji sprendimai tarsi simuliuoja kitų žmonių rašyseną, taip smarkiai padidinant tikimybę taisyklingai nustatyti parašytą raidę. Tai patvirtina [12] atliktas tyrimas, kurio rezultatais remiantis, panaudotas genetinio algoritmo metodas leido pasiekti 98,44% teisingai nustatytų atvejų.

1.6.4. GENETINIO METODO ALGORITMAS

Genetinio kaip ir aštuonetainio grafo metodai turi kelias bendras savybes. Pirmiausia norint tiksliai atlikti simbolio atpažinimo operaciją, reikia atlikti įvairias parašyto simbolio atvaizdo modifikacijas. Dažniausiai tai normalizacija, kurios metu simbolio briaunos padaromos vieno pikselio storumo. Kitas šių metodų panašumas yra tas, kad abiem atvejais yra sudaromi simbolių grafai. Tačiau lyginant su aštuonetainio grafo modeliu, čia nėra lyginamos grafų savybės pagal jų svorius. Genetinio metodo atveju sprendinio tinkamumas yra tikrinamas pasinaudojant tinkamumo funkcija (angl. *fitness function*). Šios funkcijos pagalba yra ieškomi nukrypimai tarp grafo viršūnių $d(e_1,e_2)$, kuriuos suskaičiavus galima gauti viso grafo nukrypimą nuo tam tikro šablono grafo.



6 pav. Nukrypimas tarp dviejų linijų

Tarkime mes turime dvi linijas ir norime paskaičiuoti koks yra nuokrypis tarp šių linijų d. Tai galime padaryti paskaičiavus nuokrypį tarp šių linijų galų pasinaudojus tokia funkcija:

$$d_1(e_1,e_2) = (x_{1s} - x_{2s})^2 + (y_{1s} - y_{2s})^2 + (x_{1e} - x_{2e})^2 + (y_{1e} - y_{2e})^2 \quad (1)$$

Šiuo atveju paskaičiuotas nuokrypis kai tariama, jog abiejų linijų kraštai sutampa. Tačiau galimas atvejis kai vienos linijos pradžia gali būti kitos linijos pabaiga ir atvirkščiai. Tuo atveju nuokrypis gaunamas:

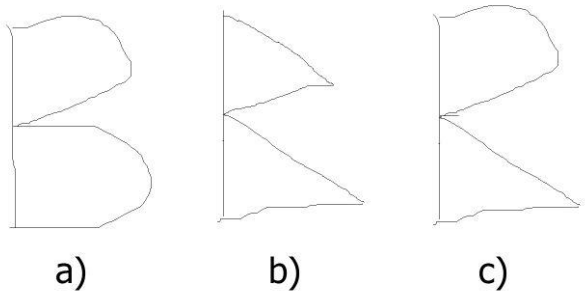
$$d_2(e_1,e_2) = (x_{1s} - x_{2e})^2 + (y_{1s} - x_{2e})^2 + (x_{1e} - x_{2s})^2 + (y_{1e} - x_{2s})^2 \quad (2)$$

Suskaičiavus d_1 ir d_2 laikoma, kad nuokrypis tarp dviejų linijų yra lygus:

$$d = \min(d_1, d_2) \quad (3)$$

Tokiu pačiu principu remiamasi kai skaičiuojami nuokrypiai tarp grafų, tik šiuo atveju skaičiuojami nuokrypiai ne tarp jų kraštinių taškų, bet tarp atitinkamų viršūnių koordinačių. Tada norint gauti nuokrypį tarp dviejų grafų reikia sudėti visus nuokrypius tarp atitinkamų viršūnių. Galiausiai lyginame savo parašytą simbolį su visais sistemoje esančiais šablonais, stengdamasi surasti mažiausią nuokrypį, kuris mums ir nurodys kokią raidę mes parašėme.

Tačiau tinkamumo funkcija visiškai nėra susijusi su genetiniu algoritmu ir atpažinimo procese gali būti naudojami įvairūs metodai. Kaip jau buvo minėta anksčiau, genetinio algoritmo išskirtinis bruožas yra galimybė iš kelių simbolių šablonų gauti kitą. Tai dažniausiai atliekama kryžminimo operacijos pagalba. Šios operacijos metu tam tikros dalys ar bruožai yra paimami iš vieno raidės šablono, o kita dalis iš kito. Taip paėmus atskiras dalis ir jas sujungus į vieną, gauname naują mūsų raidės šabloną, kuris gali būti panašesnis į mūsų parašytą simbolį ir labiau tenkinti mūsų tinkamumo funkciją. Pavyzdys kaip gali atrodyti sugeneruoti nauji parašytos B raidės atvaizdai pateikiami 7 paveikslėlyje. Šiame paveikslėlyje a ir b atvejais parodyta kokie šablonai buvo pateikti sistemai apmokymo metu. Tarkime mūsų genetinis algoritmas realizuotas taip jog turimus šablonus tiesiog dalina horizontaliai pusiau ir taip iš vieno atvaizdo paima viršutinę dalį o iš kito apatinę, šiuo atveju gautume atvaizdą kuris atrodytų taip kaip variante c.



7 pav. B raidės šablonai (a,b apmokymo duomenys; c sugeneruotas vaizdas)

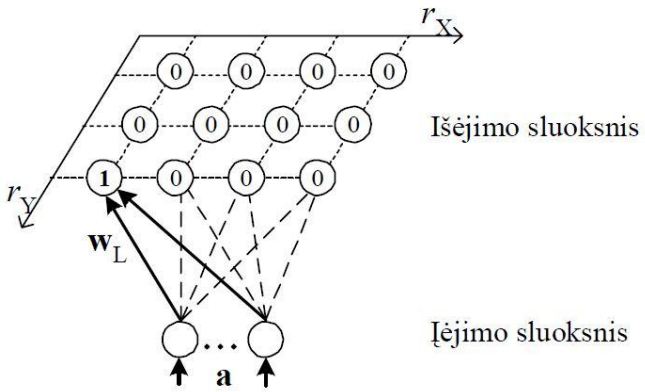
Tačiau tai tik labai paprastas genetinio algoritmo pritaikymo pavyzdys ir toks dalinimas pusiau dažniausiai jokios naudos neduos, arba gali net pakenkti. Dėl šios priežasties yra kuriami ir galvojami sudėtingesni algoritmai. Pavyzdžiui pradžioje sudaromi abiejų šablonų grafai, tarp jų ieškoma panašumų, galiausiai radus viršūnę ir grafo dalį kuri yra panaši, jie yra skeliami į dvi dalis. Iš kiekvieno grafo paimamos priešingos dalys iš kurių gaunamas naujas grafas, kuris atitinka tam tikros raidės šabloną[12].

1.6.5. LAŠELIO APTIKIMO METODAS

Lašelio aptikimo (angl. *Blob detection*) metodas žinomas ir minimas literatūroje jau 1981 metais[1]. Tačiau net ir šiais laikais įvairūs šio metodo algoritmai naudojami plačiai. Viena iš priežasčių kodėl taip yra, tai jog realizuoti lašelio aptikimo algoritmai dažniausiai veikia sparčiai ir reikalauja nedidelių kompiuterinės technikos resursų. Tai ypač aktualu sparčiai tobulėjant įvairiems mobiliesiems įrenginiams, kuriuose panaudojus tokius algoritmus įmanomi objektų aptikimo uždavinių sprendimai realiuoju laiku. Pavyzdžiui, automobilių numerių aptikimas, teksto vertimas[17], judesio sekimo sistemos, panašių objektų paieška ir kiti panašios paskirties uždaviniai. Šio metodo veikimas pasižymi tuo, kad įvairių taškų ar jų sankaupų (objektų) aptikimas atliekamas pagal jų savybių skirtumą, lyginant su tą objektą supančia aplinka. Tos savybės gali būti spalva, ryškumas, šviesumas ar kontrastas. Matematikos požiūriu šis objektas ar jo savybės atvaizde, būtų aptinkamas kaip lokalus minimumas ar maksimumas. Dėl šio metodo plataus naudojimo, nuspręsta jį pasirinkti ir ranka rašyto teksto atpažinimo įrankio kūrimui, panaudojant jį teksto segmentacijos užduotims atlikti.

1.6.6. DIRBTINIŲ NEURONŲ TINKLŲ METODAS

Neuronų tinklų metodas, tai mėginimas imituoti žmogaus smegenų darbą. Žinoma, kad žmogaus smegenis sudaro begalės tarpusavyje sujungtų neuronų, kurie sudaro savitą tinklą, o jame yra apdorojama ir tuo pačiu perduodama įvairi informacija[18]. Panašiu principu veikia ir dirbtinių neuronų tinklų algoritmai (DNT). Tokie tinklai yra sudaromi iš dviejų ar daugiau sluoksnių, kurių vienas visad yra įėjimo, o kitas išėjimo (8 pav.).



8 pav. Neuronų tinklų vaizdas

Iš čia matome, kad neuronų tinklui galime paduoti tam tikrą informaciją, kuri perėjusi per tinklą gražina tam tikrą rezultatą. Šitoks veikimas patogus atliekant teksto atpažinimą, nes

sistemai kuri naudoja dirbtinius neuronų tinklus padavus kokį nors simbolio atvaizdą, ji tą atvaizdą apdoroja ir gauna rezultatą. Savo ruožtu gautas rezultatas yra išsaugomas, dar kitaip tai yra vadinama neuronų tinklų apmokymu. Tokiai apmokytai sistemai vėliau davus apdoroti kitą simbolį ir įvedus tam tikrą paklaidą, galima nustatyti koks simbolis buvo jai paduotas[19]. Dėl šio metodo universalumo ir gebėjimo įvesti įvairias paklaidas, būtent šis modelis pasirinktas kuriamo įrankio teksto atpažinimo procedūrai atlikti.

2. PROJEKTINĖ DALIS

2.1. KOMPIUTERINIO VAIZDO APDOROJIMO BIBLIOTEKŲ PASIRINKIMAS

Atlikus rašytinio teksto atpažinimo metodų bei modelių analizę, buvo nuspręsta tolimesnius darbus atlikti taikant „offline“ rašytinio teksto atpažinimo metodus. Siekiant sukurti įrankį kuriame būtų pritaikyti minėti metodai, atlikta įvairių kompiuterinio vaizdo apdorojimo (angl. *computer vision*) programavimo bibliotekų apžvalga. Tačiau iš visų bibliotekų labiausiai išsiskyrė dvi, kurių teikiamos funkcijų pritaikymo galimybės buvo labai plačios ir pateikiamos su detalia dokumentacija, tai „OpenCV“ ir „Aforge.NET“. Toliau bus bandoma detaliau paanalizuoti kiekvieną iš minėtų bibliotekų.

2.1.1. OPENCV BIBLIOTEKA

Pirmoji biblioteka yra platinama BSD licenzijos pagrindu. Šią biblioteką nemokamai galima naudoti tiek mokslo, tiek ir komercijos tikslais. Jos branduolys parašytas naudojant C programavimo kalbą, tačiau tuo pačiu joje yra realizuoti pilni C++ ir Python programavimo kalbų sąsajos. Šiuo metu biblioteką sudaro daugiau nei 2000 įvairių algoritmų, kurie yra suskirstyti į 9 grupes:

- core – į šia grupę įeina visi duomenų tipai ir struktūros, naudojamos visose kitose šios bibliotekos funkcijose. Taip pat šiai grupei priklauso įvairių operacijų ir duomenų masyvų funkcijos.
- imgproc – šios grupės pagrindas – įvairios atvaizdų analizės ir transformacijos funkcijos. Pavyzdžiui histogramų paskaičiavimas, filtrų uždėjimas ant atvaizdo ar bruožų paieška atvaizde.
- features2d – tai dvimačių objektų analizės ir duomenų apie objektų bruožus išgavimo funkcijos.
- flann – duomenų klasterizavimo ir paieškos funkcijos daugiamačiose dimensijose.
- objdetect – įvairių bruožų ir savybių nustatymo funkcijos ekrane arba nurodytame atvaizde.
- video – panašių funkcijų grupė kaip ir imgproc ar features2d grupėse, tačiau pritaikytos darbui su video rinkmenomis.

- highgui – funkcijos skirtos paprastos vartotojo sąsajos kūrimui, dažniausiai naudojama atliekant įvairius bandymus, bet ne išbaigtose programose.
- calib3d – metodų rinkinys skirtas darbui su vaizdo kamera, kameros ir garso kalibravimu.
- ml – kompiuterinių sistemų apmokymo (angl. *machine learning*) klasių ir funkcijų grupė, kurią sudaro statistinio klasifikavimo, duomenų regresijos ir klasterizavimo metodai.

2.1.2. AFORGE.NET BIBLIOTEKA

Antroji biblioteka yra platinama LGPL v3 licenzijos pagrindu. Tai taip pat atvirojo kodo biblioteka, kurią gali naudoti įvairių programų kūrėjai bei mokslininkai, kurie dirba „Computer vision“ ir dirbtinio intelekto srityse. Priešingai nei anksčiau minėta OpenCV biblioteka, Aforge.NET kaip jau galima suprasti vien iš jos pavadinimo, yra orientuota į .NET karkasą (angl. *framework*) ir yra parašyta C# programavimo kalbos pagrindu. Išskiriamos 8 pagrindinės šios bibliotekos taikymo sritys (biblioteką sudaro bibliotekų rinkinys):

- Imaging – šią biblioteką sudaro įvairios klasės ir funkcijos skirtos darbui su paveikslėliais.
- Vision – biblioteka skirta judesio suradimo ir jo analizei video rinkmenose.
- Video – klasių ir funkcijų rinkinys, skirtas gauti prieigą prie įvairaus tipo vaizdo šaltinių.
- Neuro – šioje grupėje apjungtos klasės ir metodai skirti neuroninių tinklų sudarymui ir skaičiavimų atlikimui juose.
- Genetic – šios bibliotekos pagrindas – klasės leidžiančios spręsti įvairius optimizavimo ir prognozavimo uždavinius, pritaikant genetinio algoritmo metodus.
- Fuzzy – įvairių metodų rinkinys, kurių pagalba leidžiama dirbti su putliais (angl. *fuzzy*) duomenų rinkiniais.
- Robotics – bibliotekos pagrindas sudarytas iš klasių skirtų darbui su robotais, turinčiais tam tikras valdymo plokštes.
- MachineLearning – šioje bibliotekoje apjungtos klasės ir metodai kurių pagalba realizuojami įvairūs kompiuterinių sistemų apmokymo algoritmai.

Apžvelgtos dvi bibliotekos savo realizuotais metodais labai panašios viena į kitą. Pagrindinis jų skirtumas tai naudojama programavimo kalba ir pritaikymo lankstumas. „OpenCV“ realizuota C++ kalbos pagrindu todėl ją kiek paprasčiau naudoti įvairiose operacinių sistemų platformose. Taip pat ši biblioteka turi realizuotą sąsają su QT. Tuo tarpu Aforge.NET paremta .NET karkaso pagrindu, todėl nepaisant to kad sparčiai augantis ir tobulėjantis Mono-Project projektas suteikia galimybę sukurtas programas paruošti naudoti Linux platformoje, ši biblioteka visgi yra ne tokia lanksti. Atsižvelgiant į platesnes OpenCV pritaikymo galimybes, rašytinio teksto atpažinimo programai realizuoti pasirinkta būtent ši biblioteka.

2.2. KITŲ ĮRANKIŲ PASIRINKIMO ANALIZĖ

Kadangi atlikus darbinės srities analizę nustatyta jog realizuojant rašto atpažinimo uždavinio sprendimo įrankį susiduriama su keliomis didelėmis probleminėmis sritimis, nutarta rinktis tokius įrankius su kuriais turima daugiausiai patirties. Todėl realizuojant magistrinio darbo projektinę dalį, kaip programavimo aplinką nuspręsta naudoti „Microsoft Visual Studio 2010“. Dar vienas šios aplinkos privalumas tas, kad joje galima atlikti ir pagrindines projektavimo užduotis.

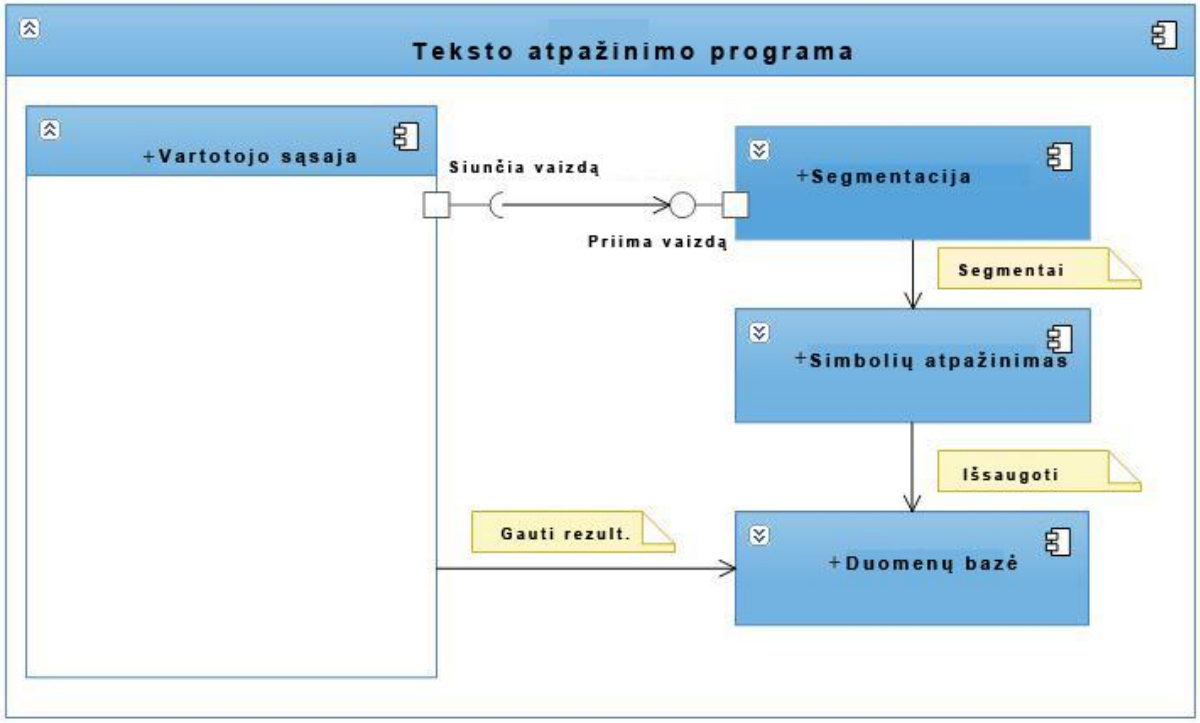
Ranka rašyto teksto atpažinimo funkcijom realizuoti pasirinkta „OpenCV“ biblioteka. Pagrindinis šios bibliotekos pasirinkimo kriterijus yra detalus, išsamus ir tvarkingas dokumentacijos pateikimas. Taip pat joje realizuota pilna integracija su Visual Studio programavimo aplinka, kas turėtų suteikti kuriamai sistemai stabilumo ir patikimumo. Verta paminėti, kad ši biblioteka, remiantis internete aptiktais šaltiniais, bene plačiausiai naudojama ir dėl tos pačios priežasties didžiausią bendruomenę turinti biblioteka, kuri naudojama tokio pobūdžio darbams atlikti.

Kadangi kuriamas įrankis turės apdoroti nemažai duomenų, siekiant tų duomenų apdorojimo ir atrinkimo paprastumo, neišvengiamai reikalinga ir duomenų bazė. Duomenų bazę šiame projekte nuspręsta pasirinkti tokią, kad ją būtų galima lengvai integruoti į visą projektą ir kad jos veikimui nereikėtų papildomų programinių priemonių. Tokius reikalavimus geriausiai tenkina „SQLITE3“ duomenų bazė. Nors teksto atpažinimo įrankyje dėl didelio skaičiavimų kiekio, svarbiausias vaidmuo tenka procesoriui, tačiau renkantis duomenų bazių valdymo sistemą buvo atsižvelgta ir į spartą. Ši duomenų bazių valdymo sistema, lyginant su savo pagrindinėmis alternatyvomis „MySQL“ ir „PostgreSQL“, juos lenkia ir šioje srityje.

2.3. RAŠYTINIO TEKSTO ATPAŽINIMO PROGRAMOS PROJEKTAS

Siekiant supaprastinti programos realizavimą, nuspręstą ją padalinti į 3 pagrindinius komponentus (9 pav.):

- GUI – tai vartotojo sąsaja, per kurią bus paduodami norimi atpažinti paveikslėliai ir atvaizduojamas gautas rezultatas.
- FeatureExtraction – modulis atsakingas už teksto simbolių išskirimo iš pateikto teksto atvaizdo.
- CharacterRecognition – dar vienas modulis, kuris atsakingas už gautų simbolių atpažinimą ir pavertimą į kompiuteriui suprantamą teksto formatą.



9 pav. Programos komponentų modelis

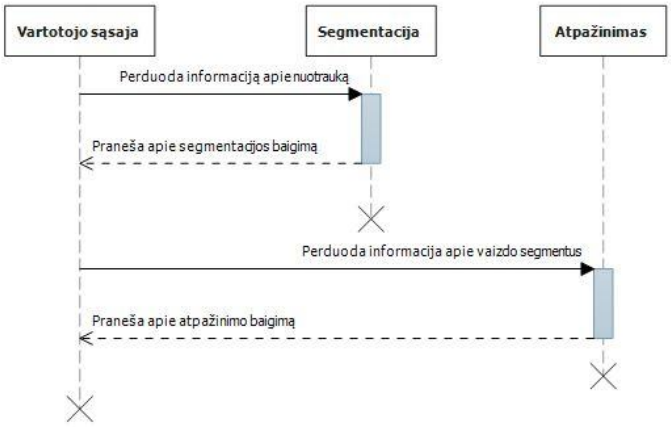
3. REALIZACINĖ DALIS

3.1. GALUTINIS PROJEKTO APRAŠYMAS

Atliekant darbą susidurta su problemomis dėl kurių galutinis realizuojamos sistemos veikimas skiriasi nuo to kas buvo planuota. Tačiau nepaisant to pagrindinės sistemos dalys buvo išlaikytos tokios pačios, tik jų veikimas šiek tiek pakitęs. Realizuotą ranka rašyto teksto atpažinimo sistemą ir toliau sudaro trys pagrindinės dalys:

- Teksto segmentaciją atliekanti paprogramė;
- Teksto atpažinimą atliekanti paprogramė;
- Už vartotojo sąsają ir visų sistemos dalių veikimą atsakinga pagrindinė programa.

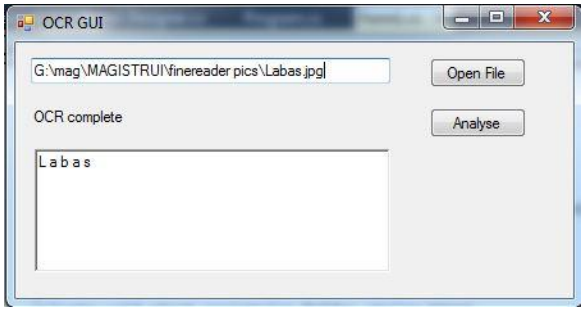
Galutinę sistemą sudaro kelios paprogramės su jas valdančia pagrindine programa, o ne viena programa iš kelių modulių kaip buvo planuota. Šitoks pakeitimų įtraukimas į sistemą atliktas nenorint apriboti jos pritaikymo galimybių ar tolimesnio tobulinimo.



10 pav. Sistemos komponentų sąveika

Sistemos veikimas kaip matome pagal pateiktą vaizdą (10 pav.), prasideda programos kuri atsakinga už vartotojo sąsają paleidimu. Šioje programoje įvestas valdymo mechanizmas, kuris kitas paprogrames paleidžia tik tam tikromis sąlygomis. Pavyzdžiui segmentavimo paprogramė paleidžiama tik tada kai iš pagrindinės programos nurodome „jpg“ tipo paveikslėlį. Jei sistemai paduotas tinkamas paveikslėlis, ši perduoda apie tai informaciją paprogramei kuri atliks segmentavimą ir toliau nieko neatlieka kol negauna pranešimo, kad segmentavimo paprogramė baigė darbą. Lygiai taip pat kontroliuojamas ir teksto atpažinimą

vykdančios paprogramės darbas, kuriai atlikus visas užduotis, pagrindinė programa tik tada pasiima gautus rezultatus ir juos pateikia ekrane (11 pav.). Taigi visos sistemos darbas pagrįstas sinchronizuotų pranešimų apsikeitimu.

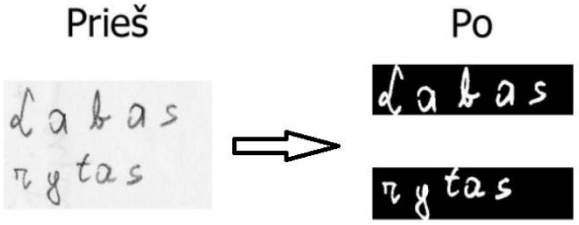


11 pav. Programos vaizdas

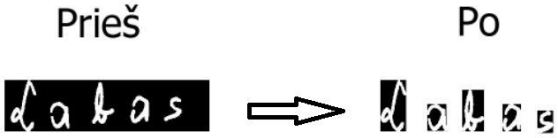
Segmentacijos paprogramės veikimą galime išskaidyti į tokius pagrindinius žingsnius:

- Priimti informaciją apie paveikslėlį;
- Paveikslėlyje atlikti visų objektų aptikimą ir informacijos apie rastus objektus išsaugojimą, pasinaudojant OpenCV bibliotekos „Lašelio aptikimo“ funkcija;
- Paskaičiuoti vidutinį vieno objekto aukštį;
- Remiantis vidutinio objekto aukščio parametru, išskaidyti pateiktą paveikslėlį į eilutes;
- Analizuoti kiekvieno objekto eilutėje koordinates x ir y ašyje ir taip sudaryti sąrašą kokia eilės tvarka turėtų eiti objektai vienas po kito;
- Nustatyta eilės tvarka, apie kiekvieną objektą sukurti virtualų keturkampį ir tame keturkampyje atsidūrusį vaizdą išsaugoti naujo paveikslėlio pavidalu;
- Pranešti pagrindinei programai kad segmentavimas baigtas.

Žemiau pateikiamas realus programos veiksmų rezultatas (12, 13 pav.). Daugiau segmentavimo rezultatų pateikiama priede nr. 6.



12 pav. Teksto segmentacija į eilutes



13 pav. Teksto segmentacija į simbolius

Teksto atpažinimą atliekančios paprogramės veikimas:

- Priimti informaciją apie tai kur sudėti duomenys gauti po segmentavimo;

- Pasinaudojant „Encog“ bibliotekos funkcija, normalizuoti kiekvieną iš turimų atvaizdų į tam tikrą vieno dydžio šabloną;
- Gautus šablonus versti į bitų masyvus;
- Kiekvieno objekto bitų masyvą paduoti į „Encog“ bibliotekos siūlomą neuronų tinklą;
- Gautą rezultatą iš neuronų tinklo lyginti su iš anksto sistemos apmokytais duomenimis ir taip išrinkti labiausiai tenkinantį variantą;
- Visus rezultatus išsaugoti į CSV ir txt formato rinkmenas, tolimesniems sistemos tyrimams atlikti;
- Pranešti apie darbo pabaigą.

3.2. PROBLEMOS IR JŲ SPRENDIMO BŪDAI

Atliekant darbą susidurta su trimis pagrindinėmis problemomis:

1. Pirmoji problema su kuria susidurta atliekant darbą tai taisyklingas aptiktų simbolių eiliškumo išlaikymas po segmentacijos. Problema pasireiškė tuo, kad toje pačioje eilutėje esantys simboliai susikeisdavo vietomis, nes naudojama biblioteka objektų aptikimą pirmiausiai atlikdavo nuo aukščiausiai nuotraukoje esančių objektų. Dėl šios priežasties toks simbolių segmentavimas buvo visiškai netinkamas tekstui ir šiai problemai spręsti buvo ieškoma įvairių sprendimų, tačiau galiausiai nuspręsta realizuoti papildomą funkciją. Ši funkcija, pagal aptiktų objektų koordinates, pirmiausiai nuotrauką skaido eilutėmis iš viršaus į apačią, o tik po to eilutes skaido į atskirus simbolius iš kairės į dešinę.

2. Su antrąja problema susidurta perėjus prie teksto atpažinimo modulio realizacijos. Bandant realizuoti teksto atpažinimo modulį paaiškėjo, kad naudojama OpenCV biblioteka gali atpažinti simbolius juos pateikus tik specifiniu tai bibliotekai duomenų masyvu. Siekiant išvengti tokių apribojimų, buvo ieškoma kitų bibliotekų, kurios teikiamas funkcionalumas neturėtų tokių apribojimų ir būtų galima naudoti jau realizuotą teksto segmentavimo modulį. Atlikus papildomą įrankių paiešką, rasta biblioteka „Encog“, kuriai norimus atpažinti simbolius galima pateikti tiesiog „jpeg“ formato paveikslėliais, dėl ko nereikia atlikti jokių papildomų veiksmų po segmentacijos.

3. Trečioji problema kilo iš antrosios. Iki šios vietos visą teksto atpažinimo įrankį buvo bandoma realizuoti pasitelkiant vien C++ programavimo kalbą, kuria ir parašyta OpenCV biblioteka. Tačiau atsiradus papildomai bibliotekai „Encog“, kuri realizuota C# programavimo kalba, teko ieškoti būdų kaip visą sistemą sujungti į vieną įrankį. Projektuojant įrankį buvo planuojama visą programą padalinti į tris nepriklausomus modulius, tačiau

siekiant išspręsti šią problemą to neužteko. Visas programos dalis teko dalinti ne tik į modulius, bet į visiškai nepriklausomas paprogrames, kurios atliko skirtingas funkcijas. Pirmoji atliko atvaizdo segmentavimą į simbolius, antroji gautų simbolių atpažinimą ir trečia sąsaja tarp kitų dviejų ir kontroliavo jų darbą.

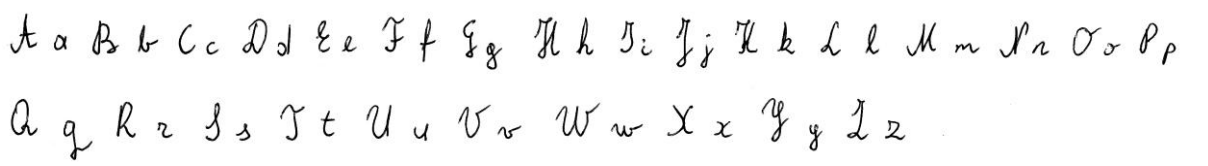
3.3. DARBO REZULTATŲ ANALIZĖ

Sukurtas teksto atpažinimo įrankis, atpažinimo procesą atliekantis iš nuotraukos. Įrankyje panaudoti du dirbtinio intelekto metodai: „lašelio aptikimo“ ir „dirbtinių neuronų tinklų“. Kiekvienas iš metodų buvo orientuotas į skirtingos probleminės srities uždavinių sprendimą. „Lašelio aptikimo“ metodą naudojame, norėdami atvaizde rasti simbolius (jų koordinates), kurių pagalba toliau atliekamas vaizdo skaidymas. Tuo tarpu „dirbtinių neuronų tinklų“ metodas, naudotas simbolio atpažinimo procedūrai atlikti.

Realizuojant teksto atpažinimo sistemą pastebėta, kad naudojant pasirinktą „OpenCV“ kompiuterinio vaizdo apdorojimo biblioteką, visas sistemos veikimas yra pririšamas prie tik tos bibliotekos naudojamų duomenų tipų. Dėl šios priežasties iš bibliotekos, sistemoje panaudota tik „lašelio aptikimo“ funkcija. Toks pasirinkimas leidžia programoje ir toliau naudoti visus standartinius duomenų tipus, bei taip dirbtinai nesukurti apribojimų ir išlaikyti jos universalumą. Šis sprendimas leido naudoti kitas bibliotekas ir tuo pačiu lengvai prijungti savo realizuotus metodus. Tai leido sukurti rašytinio teksto atpažinimo įrankį, kuris atpažinimo procesą atlieka iš nuotraukos. Šioje sistemoje panaudoti du dirbtinio intelekto metodai: „lašelio aptikimo“ ir „dirbtinių neuronų tinklų“. Kiekvienas iš metodų buvo orientuotas į skirtingos probleminės srities uždavinių sprendimą. „Lašelio aptikimo“ metodą naudojame, norėdami atvaizde rasti simbolius (jų koordinates), kurių pagalba toliau atliekamas vaizdo skaidymas. Tuo tarpu „dirbtinių neuronų tinklų“ metodas, naudotas simbolio atpažinimo procedūrai atlikti.

Atliekant pirminius sistemos rašto atpažinimo efektyvumo testus pastebėta, kad gaunami rezultatai labai žemi. Siekiant išvengti problemų atliekant sistemos tyrimą, lygiagrečiai „dirbtinių neuronų tinklų“ algoritmui buvo sukurtas papildomas metodas. Šio metodo veikimo principas pagrįstas gauto objekto, paversto bitų masyvu, analize. Šį metodą realizuoti buvo labai patogu, dėl to jog dirbtinis neuronų tinklas, kurio pagalba vyksta teksto atpažinimas, taip pat naudoja panašią duomenų struktūrą.

Visa atpažinimo sistema testuota angliška abėcėle su didžiosiomis ir mažosiomis raidėmis. Siekiant paprastumo sistema apmokyta tik vienu duomenų rinkiniu (14 pav.) iš kurio kiekviena gauta raidė normalizuojama į 40 pikselių plotį ir 60 pikselių aukštį.



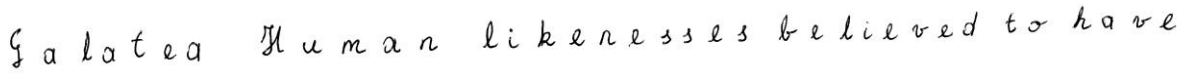
14. pav. Sistemos apmokymui naudotas šablonas

Tai leido išlaikyti pakankamai detalų vaizdą bitų masyvų analizės metodui, aukščiausią patikimumo procentą neuronų tinklų metodui ir tuo pačiu greitą sistemos veikimą. Kadangi sistemos sparta nėra pagrindinis tyrimo kriterijus, apie tai duomenys nebuvo renkami. Reikia paminėti, kad atliekant pirminius patikrinimus, buvo nustatyta jog dirbtinių neuronų tinklu metodas veikia ne taip efektyviai kaip iš jo tikimasi. Bandant atpažinti tuos pačius simbolius su kuriais prieš tai sistema buvo apmokyta, tikslumo procentas svyravo nuo 5% iki 35% ir tik nustačius raidės dydį 40x60, rezultatai tapo stabilesni – apie 30%. Tapo akivaizdu, kad gauti rezultatai netenkinta ir su jais nebus įmanoma atlikti tolimesnius tyrimus. Būtent šioje vietoje nuspręsta realizuoti antrą atpažinimo algoritmą.

Tolimesni duomenys apie programos veikimą buvo renkami trimis etapais. Pirmuoju etapu buvo bandoma atpažinti paveikslėlį kuriame surašytos abėcėlės, antruoju – tvarkingai parašytas tekstas, trečiuoju – tekstas kuriame atsitiktinai rašomos didžiosios ir mažosios raidės.

Kaip ir buvo galima tikėtis iš pirminių testų rezultatų, dirbtinių neuronų tinklų algoritmas nesusitvarko su savo užduotimi. Iš 228 pateiktų simbolių teisingai atpažinti buvo tik 45, tai sudaro beveik 20%. Tuo tarpu atvaizdų bitų masyvų palyginimo atveju, teisingai atpažintų simbolių skaičius – 89%.

Antrasis testinių duomenų rinkinys pats aktualiausias. Jame pateikiamas logiškas, rišlus tekstas. Dirbtinis neuronų tinklų algoritmas pasiekė vos 6% tikslumą, o atvaizdų palyginimo algoritmas pasiekė 68,5%. Turint tokio procento atpažinimo lygį jau įmanoma įskaityti tam tikrus teksto blokus. Pavyzdžiui iš paveikslėlio 15, gautas tekstas yra: „alatra HQmOn eiVenessls believeM to kuvl“. Tekstas beveik nesuprantamas, tačiau jo pagalba galima įžvelgti dėsningumus, kuriais atvejais sistemai sunkiausiai sekasi atpažinti raides.



15. pav. Teksto eilutė iš testams naudotų nuotraukų

Paskutiniu atveju gaunami panašūs rezultatai kaip ir prieš tai buvusiuose testuose, atitinkamai 8% ir 65%. Apibendrinti rezultatai pateikti lentelėje:

Algoritmas / Duomenų rinkinys	Dirbtinių neuronų tinklas		Bitų masyvų analizė		Viso simbolių rinkinyje
	Teisingų vnt.	Teisingų %	Teisingų vnt.	Teisingų %	
Abėcėlė	45	15,96	203	71,99	282
Tvarkingas tekstas	20	5,93	231	68,55	337
Atsitiktinis tekstas	31	8,2	247	65,34	378
Viso:	96	9,63	681	68,30	997

1 lentelė. Teksto atpažinimo sistemos efektyvumo duomenys

Apibendrinus atliktų testų rezultatus galime teigti, kad bitų masyvų analizės algoritmas teisingai atpažįsta kiek daugiau nei du iš trijų simbolių. Toks rezultatas atsižvelgiant į tai, kad sistema atpažinimus vykdė pagal vieną jai pateiktą šabloną yra gana aukštas ir vertas atlikti tolimesnius patobulinimus siekiant dar didesnio patikimumo. Tuo tarpu dirbtinio neuronų tinklo algoritmas buvo visiškai netinkamas.

Paanalizavus gautus rezultatus detaliau (žr. 4 priedas) galime pastebėti tam tikrus sistemos klaidų dėsningumus. Pavyzdžiui dažniausiai suklystama atpažįstant didžiąsias ir mažąsias raides. Ypač dažnai tai nutinka su raidėmis C ir c ar O ir o. Taip pat labai dažnai suklysta ir raidžių e-l bei h-k atveju (2 lent.).

Neteisingas derinys	a-u	A-N	c-C	d-M	e-l	h-k	I-S	Y-T	m-x	n-Q	o-O	p-P	s-l	u-d
Kiekis	7	6	16	6	33	21	7	5	5	8	6	7	7	5

2 lentelė. Dažniausiai pasitaikiusių neteisingų derinių lentelė

Matome, kad kelios klaidos pasikartoja ypač dažnai ir to priežastis paprasta. Šie simboliai rašomi labai panašiai. Vienas iš galimų šios klaidos pašalinimo variantų galėtų būti analizuojamos raidės padidinimas. Tai leistu detaliau perteikti atvaizduojamos raidės savybes ir taip galimai sumažintų klaidų tikimybę, tačiau tai labai priklauso nuo kiekvieno žmogaus rašymo stiliaus, todėl šis variantas nėra universalus.

Atsižvelgiant į gautus rezultatus, šiuo metu negalime sakyti, kad dabartinė sistema yra pritaikoma efektyviam naudojimui. To priežastys yra kelios. Pirmoji priežastis kyla iš to, kad realizuojamai sistemai keliami uždaviniai turi nemažai probleminių sričių. Kelėtai jų, segmentavimo ir teksto atpažinimo, išspręsti reikalingas atskiras tyrimas sukoncentruotas į tas sritis. Kadangi nepavyko rasti efektyviai veikiančios tokio pobūdžio teksto atpažinimo sistemos, atliekant darbą teko spręsti visas problemas su kuriomis susiduriama atliekant teksto atpažinimo procedūrą.

Antroji priežastis gaunama kaip pirmosios rezultatas. Iškeltas darbo tikslas sukurti ir ištirti ranka rašyto teksto atpažinimo sistemų problemines sritis, o ne tik tam tikrą jos dalį. Tačiau neturint efektyviai veikiančio komponento kuris taptu sistemos pagrindu, nėra atskaitos taško kurio rezultatais remiantis būtų galima atlikti tolimesnį sistemos realizavimą ir testavimą. Šiuo atveju tai labai svarbu, nes nuo vieno sistemos komponento priklauso kito komponento rezultatai, vienam iš jų suveikus netiksliai, yra įtakojamas visas sistemos gautas rezultatas. Pavyzdžiui siekiant, kad atpažintas tekstas būtų perskaitomas, svarbu ne tik ar raidės buvo atpažintos teisingai, bet ir tai ar atliekant segmentaciją raidės buvo išskaidytos be klaidų ir ar išdėliotos reikiama tvarka (žr. 5 priedas).

Sistemoje naudotas dirbtinių neuronų tinklų algoritmas, pasirodė neefektyvus. Gautas 9,63% tikslumas žymiai atsilieka nuo paprasto atvaizdų analizės algoritmo. Tačiau verta paminėti, kad testuojant egzistuojančio komercinio įrankio veikimą su ranka rašytu tekstu, gautas panašus rezultatas. Siekiant pagerinti šio algoritmo rezultatus bandyta, keisti kiekvieno simbolio dydį, kuris paduodamas sistemai atpažinti, tačiau tai neturėjo didesnės įtakos. Analizuojant gautus testavimo rezultatus, nebuvo įmanoma įžvelgti jokių dėsningumų kur sistema klysta (žr. 4 priedas). Algoritmu bandant atpažinti tą patį tekstą kelis kartus, visais atvejais buvo gaunamas vis kitoks rezultatas. Tai galėjo nutikti dėl to, kad kiekvienas algoritmas veikia skirtingai ir jam paduodamas atpažinti vaizdas turi būti paruoštas pagal tam tikras procedūras[11,12,14].

Apibendrinant gautus sistemos testavimo rezultatus, galime teigti, kad kuriant betkokį teksto atpažinimo įrankį, kuris atpažinimo procesą turės atlikti iš nuotraukos, neišvengiamai turėsime išspręsti tokias problemas:

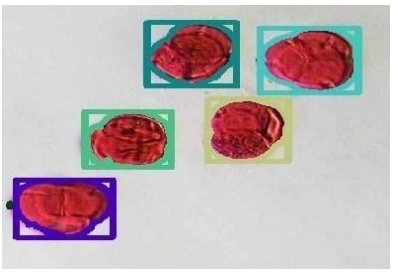
- Kaip teisingai išskaidyti tekstą į eilutes, o šias į simbolius;
- Kaip paruošti ir kokiu formatu gautus simbolius pateikti teksto atpažinimo funkciją atliekančiam moduliui;
- Pasirinkti ar sugalvoti tinkamą teksto atpažinimo algoritmą.

3.4. DARBE NAUDOTŲ PRIEMONIŲ PRITAIKYMO GALIMYBĖS KITUOSE PROJEKTUOSE

Atliekant teksto atpažinimo programos realizavimą ir testavimą, pastebėta, kad pasirinktos priemonės, atlikus tam tikras korekcijas gali būti naudojamos ir kitų uždavinių sprendimui. Viena pagrindinių priežasčių kuri leidžia pritaikyti sukurtą sistemą kitoms sritims, tai „lašelio aptikimo“ algoritmo taikymas. Šis algoritmas leidžia aptikti ne tik teksto simbolius, bet ir kitus objektus kurie kokiu nors požymiu išsiskiria nuo juos supančios aplinkos.

Kaip pavyzdį galime paimti, mikroskopu padarytas įvairių žiedadulkių nuotraukas. Tokiai nuotraukai pritaikius tam tikrus filtrus, galime atlikti žiedadulkių segmentaciją. Priešingai nei teksto atpažinimo atveju kai pateiktam vaizdui svarbiausi iš taikomų filtrų yra slenkstinis (angl. *threshold*) ir aštrinimo (angl. *sharpening*), žiedadulkių atveju gali tekti pritaikyti įvairių spalvų, kontrasto ar ryškumo filtrus. Tačiau pati priežastis kodėl atvaizdui yra taikomi filtrai išlieka ta pati – objekto ribų išskyrimas iš fono, be ko nebūtų galima atlikti tolimesnių žingsnių.

Net neatlikus jokių korekcijų egzistuojančioje programoje, vietoj teksto padavus žiedadulkių atvaizdą gaunamas įdomus vaizdas (16 pav.)



16 pav. Rododendro žiedadulkių vaizdas po lašelio aptikimo metodo pritaikymo

Iš šio vaizdo matyti, kad sistema reaguoja į žiedadulkių vaizdus nuotraukoje ir jas tiksliai išskiria. Tačiau ne visais atvejais gaunamas rezultatas yra toks tikslus (žr. 6 priedas). Taip yra todėl, kad realizuota programa yra labiausiai pritaikyta atpažinti objektus, kurių spalva labai skiriasi nuo fono. Verta paminėti kad naudojamas įrankis „lašelio aptikimo“ metodui atlikti, turi galimybes pritaikyti įvairius filtrus pateiktam atvaizdui. Neabejotina, kad patestavus sistemą su tokio tipo atvaizdais, taip pat kaip buvo daroma ir teksto atveju, galima pasiekti daug geresnių rezultatų.

Deja realizuotos sistemos atpažinimo modulis nėra toks lankstus kaip segmentacijos. Atpažinimo modulio veikimas pagrįstas veikimo principu kai dirbtinio neuronų tinklo neuronas įgyją reikšmę 1 arba 0 priklausomai nuo to ar atvaizdo pikselis yra juodas ar baltas. Kadangi žiedadulkių atpažinimui yra svarbi ir spalva, tai neutronų tinklų ir bitų masyvų analizės metodai nėra tiesiogiai pritaikomi šio uždavinio sprendimui. Vienas iš galimų šios kliūties sprendimų variantų gali būti atvaizdo dalinimas į atskirus sluoksnius. Šiuo atveju kiekvienas sluoksnis turėtų tik tam tikrą spalvų grupę, t. y. pikselis būtų arba baltas ir įgautų reikšmę 0 arba tam tikros spalvos ir įgautų reikšmę 1. Atlikus tokį veiksmą su visais sluoksniais, būtų galima juos apjungti į vieną objektą, žiedadulkės šabloną, pagal kurį sistema galėtų atlikti jos atpažinimą.

Taigi nors dabar realizuotas įrankis nėra optimalus žiedadulkių aptikimui atlikti, tačiau padarius tam tikrus pakeitimus, galėtų tapti vienu iš galimu šio uždavinio sprendimo variantu.

3.1. REKOMENDACIJOS

Norint pasiekti geresnius rezultatus, pirmiausiai reikėtų realizuoti papildomą patikrinimą su kiekvienos raidės koordinatėmis. Tai padėtų atsikratyti tų atvejų, kai raidės kurių aukštis labai skiriasi nuo eilutėje esančių raidžių aukščio vidurkio, yra nukeliamos į kitą eilutę. Tolimesnis segmentacijos modulio patobulinimas turėtų būti pavienių objektų prijungimas. T.y. atliekant segmentavimą, reikėtų gautus mažus objektus pridėti prie šalia esančių raidžių. Pavyzdžiui prie į raidės pridėti jos taškiuką, o nelaikyti jo kitu objektu. Šitoks pakeitimas jau leistų efektyviai naudoti lietuvišką abėcėlę.

Siekiant geresnio atpažinimo modulio veikimo, dirbtinių neuronų tinklų atveju, pastebėta, kad šis veikia tiksliau su mažais objektais. Todėl darbe analizuotas normalizacijos procesas, kurio metu iš parašytos raidės padaromas 1 pikselio storio objektas, galimai padidintų šio metodo efektyvumą.

Bitų analizės metodas, remiantis darbo rezultatų analize, būtų kur kas efektyvesnis, jei į jo veikimą būtų galima įtraukti parametrą, kurio pagalba galima nustatyti ar rašoma raidė buvo didžioji ar mažoji. Tai padėtų išvengti didžiosios dalies klaidų su tokiomis raidėmis.

IŠVADOS

1. Teorinė analizė nulėmė, kad realizuojant ranka rašyto teksto atpažinimo programą buvo naudojami šie dirbtinio intelekto metodai: „lašelio aptikimo“ ir „dirbtinių neuronų tinklų“.
2. Pagrindinės teksto atpažinimo sistemų probleminės sritys yra: teksto skaidymas į simbolius ir taisyklingas panašiai rašomų simbolių atpažinimas.
3. Sukurtos programos teisingai atpažintų simbolių dalis naudojant bitų masyvų analizės algoritmą yra 68%, dirbtinių neuronų tinklų algoritmą – 9%.
4. Kiekvienam teksto atpažinimo algoritmui reikalingos skirtingos paruošiamosios procedūros, nuo kurių priklauso algoritmo veikimo efektyvumas.
5. Kuriamoje programoje nenumatyta funkcija, kuri simbolius verstų į vieno pikselio storio objektus. Tik vėliau nustatyta, kad šios funkcijos trūksta efektyviam „dirbtinių neuronų tinklų“ algoritmo panaudojimui.

NAUDOTA LITERATŪRA:

1. Danker, Alan J. *Blob Detection by Relaxation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1981, Vol.* 3 Issue 1, p79 – 92.
2. E. Gray. *Teleautograph* [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://www.freepatentsonline.com/386815.pdf>.
3. H.E. Goldberg. *Controller* [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://www.freepatentsonline.com/1117184.pdf>.
4. Pencept. *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, Inc. 2004. 5 [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://www.enotes.com/topic/Pencept >.
5. Ashish A. Tiwari. *Touchscreens that changed the world* [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://ashishtiwari.posterous.com/guifx-touchscreens-that-changed-the-world>.
6. *PenPoint OS* [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://en.wikipedia.org/wiki/PenPoint_OS>.
7. *IBM Simon* [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://www.retrocom.com/bellsouth_ibm_simon.htm>.
8. *Newton MP100* [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://www.nzeldes.com/HOC/Newton.htm>.
9. *MyScript Stylus* [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://www.visionobjects.com/en/myscript/about-myscript/>.
10. Hui Ma. *Handwriting Recognition's Odyssey to Windows 7* [interaktyvus] [žiūrėta 2010m. gruodžio 15d.]. Prieiga per internetą: <http://research.microsoft.com/en-us/news/asia/features/ciw_handwritingrec.aspx>.
11. Kannan, R. Jagadeesh; Prabhakar, R.. *An Improved Handwritten Tamil Character Recognition System using Octal Graph*. Journal of Computer Science, 2008, Vol. 4 Issue 7, p509-516.
12. Kala, Rahul; Vazirani, Harsh; Shukla, Anupam; Tiwari, Ritu. *Offline Handwriting Recognition using Genetic Algorithm*. International Journal of Computer Science Issues (IJCSI), Jul2010, Vol. 7 Issue 4, p16-25.

13. *Moment in Online Handwritten Character Recognition* [interaktyvus] [žiūrėta 2010m. gruodžio 05d.]. Prieiga per internetą: < www.rimtengg.com/coit2007/proceedings/pdfs/46.pdf >.
14. Shridhar, M.; Houle, G. F.; Kimura, F. *Recognition strategies for general handwritten text documents*. Integrated Computer-Aided Engineering, 2009, Vol. 16 Issue 4, p299-314.
15. Bhattacharyya, Kaustubh; Sarma, Kandarpa Kumar. *ANN-based Innovative Segmentation Method for Handwritten text in Assamese*. International Journal of Computer Science Issues (IJCSI), Jul2010, Vol. 7 Issue 4, p9-16.
16. Rose Holley. *Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs.* D-lib Magazine 2009, Vol. 15 Number 3/4.
17. Marc Petter, Victor Fragoso, Matthew Turk, Charles Baur. *Automatic text detection for mobile augmented reality translation.* Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, 2011, p48-55.
18. R. Laptik, D. Navakauskas. *Dirbtinių neuronų tinklų taikymas automobilių registracijos numeriams atpažinti.* ELEKTRONIKA IR ELEKTROTECHNIKA. 2005. Nr. 8(64). ISSN 1392 – 1215.
19. Saleh Ali K. Al-Omari, Putra Sumari, Sadik A. Al-Taweel, Anas J.A. Husain. *Digital Recognition using Neural*. Network Journal of Computer Science 5 (6), 2009, p427-434.
20. John McCarthy. *What Is Artificial Intelligence?.* [interaktyvus] [žiūrėta 2012m. gegužės 19d.]. Prieiga per internetą: <http://www-formal.stanford.edu/jmc/whatisai/whatisai.html>

ANOTACIJA

Lietuviškai

Autorius: Gediminas Kavaliauskas

Tema: Dirbtinio intelekto atpažinimo metodų analizė ir taikymai ranka rašyto teksto atpažinimui

Šiaulių universitetas 2012

Pagrindinis darbo tikslas yra pritaikant dirbtinio intelekto algoritmus sukurti ranka rašyto teksto atpažinimo įrankį. Siekiant šio tikslo buvo apžvelgti dirbtinio intelekto atpažinimo metodai, atlikta teksto atpažinimo algoritmų analizė. Remiantis analizės rezultatais, sukurta ranka rašyto teksto atpažinimo programa, kurioje teksto segmentavimo operacija atliekama „lašelio aptikimo“ algoritmu. Teksto atpažinimo operacijai atlikti naudojamas bitų masyvų analizės algoritmas. Realizuojant programą bei atliekant jos testavimą nustatytos pagrindinės šio tipo programų probleminės sritys: taisyklingas teksto skaidymas į simbolius ir taisyklingas panašiai rašomų simbolių atpažinimas. Remiantis gautais programos testų rezultatais pasiektas 68% simbolių atpažinimo tikslumas.

English

Author: Gediminas Kavaliauskas

Subject: The Analysis of Recognition Methods Based on Artificial Intelligence and their Application in Handwritten Text Recognition

Šiaulių university 2012

The aim of this work is to create an application for handwritten text recognition using artificial intelligence algorithms. For this purpose a number of recognition methods based on artificial intelligence were reviewed. Based on the review information an application was created for the purpose of recognizing handwritten text. The text segmentation was implemented using a blob detection algorithm. Text recognition was performed using bit array analysis algorithm. During the implementation and testing stage the main problem areas of such application were identified: text segmentation into correct symbols and recognition of symbols that are written in a similar way. Based on the testing results an accuracy of 68% was achieved.

PRIEDAI

1 priedas

„CD“ Turinys

Kompaktinės plokštelės šakniniame kataloge, pateikiama teksto atpažinimo programa, kurią diegti reikia paleidus setup.exe. Programą reikia diegti į katalogą kuriame nebūtų tarpų, pvz.: c:\programa.

Taip pat šakniniame kataloge pateikiamas aplankas testai. Jo viduje klaidu_analize.xls rinkmena, kurioje visi nesusisteminti programos testų rezultatai.

Kataloge „ziedadulkes“ pateikiamos programos segmentacijos modulio rezultatai su žiedadulkėmis.

Kataloguose 1-11 pateikiami programos darbiniai duomenys atliekant įvairius testavimus.

Ranka rašyto teksto įrenginių raida

Pirmieji komerciniai produktai skirti kompiuterį valdyti ranka atsirado 1982 metais. Tai kompanijos Pencept Penpad 200 įrenginys. Jame visiškai atsisakyta tradicinio informacijos įvedimo būdo klaviatūra ir visa sistema valdoma elektroninio rašiklio pagalba[4].

17 pav. Pencept Penpad 200

Tai pat verta paminėti, kad tai buvo pirmasis įrenginys kurio pagalba rašomas tekstas buvo atpažįstamas „online“ režimu, t.y. neturėjo jokios sistemos kurią reikėjo apmokyti ir nebuvo priklausoma nuo tam tikro asmens rašymo stiliaus.

Nešiojamų įrenginių su ranka rašyto teksto atpažinimo galimybe pradininkas yra GRiD Systems kompanijos 1989 metais sukurtas GRiDPad, kuris buvo komplektuojamas su MS-DOS operacine sistema[5]. Tiesa įvesto teksto atpažinimo procesas buvo labai lėtas ir vienam simboliui atvaizduoti kompiuteryje po jo įvedimo tekdavo laukti apie 2 sekundes. Nepaisant to būtent šis įrenginys paklojo pamatus būsimiems planšetiniams, nešiojamiems ir kišeniniams kompiuteriams.

18 pav. GRiDPad nešiojamasis kompiuteris

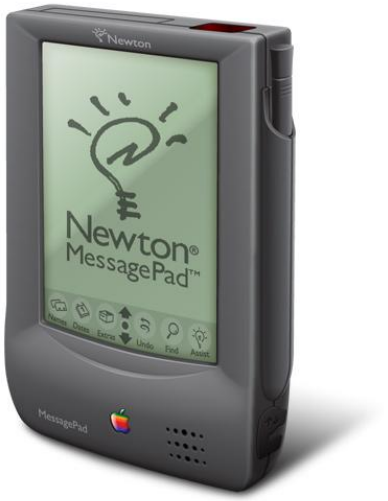
Dar spartesnį kompiuterių kūrimą su galimybe juos valdyti specialiu pieštuku, pasirenkant norimus veiksmus, lėmė pirmosios dedikuotos operacinės sistemos, tokiems įrenginiams, išleidimas. Šios srities pradininkė kompanija GO Corp sukūrusi operacinę sistemą PenPoint OS. Ši operacinė sistema nuo įprastų tuo metu buvusių operacinių sistemų išsiskyrė dar ir tuo, kad suprato įvairius gestus atliekamus pieštuku, kas leido dar greičiau ir patogiau atlikti norimus veiksmus[6].

Tuo tarpu išmaniųjų telefonų pradininke laikoma kompanija IBM 1992 metais sukūrusi ir 1993 metais pardavinėti pradėjusi savo įrenginį vadinama „Simon“. „Simon“ pasižymėjo tokiomis funkcijomis: kalendorius, adresų knygelė, pasaulio šalių laikrodis, skaičiuoklė, užrašų knygele, elektroninis paštas, žaidimai. Tačiau nepaisant to, kad šis įrenginys jau buvo su lietimui jautriu ekranu ir buvo valdomas specialiu pieštuku, jame buvo atsisakyta teksto įvedimo rašant simbolius ekrane. Vietoj to, jo ekrane norint ką nors rašyti buvo galima iškviesti pilną QWERTY klaviatūrą[7].

19 pav. IBM Simon išmanusis telefonas

Vienas pirmųjų plačiai visuomenei prieinamų PDA kompiuterių, tapo įrenginys pavadintas „Newton MessagePad 100“. Šį įrenginį 1993 metais bendradarbiaudamos sukūrė Apple, Sharp kompanijos ir nors jo dydis labiau primena planšetinį kompiuterį, jis buvo priskirtas kišeninių kompiuterių kategorijai. Jame jau buvo realizuotas pilnas ranka rašyto teksto palaikymas, o taip pat ir išorinės duomenų laikmenos ir infraraudonųjų spindulių ryšys. Šio įrenginio techninės charakteristikos buvo 20MHz ARM 610 procesorius su 4MB ROM ir 640KB RAM atmintimis, ekranas pasižymėjo 320x248 pikselių ekrano skiriamąja geba. Tuo metu nešiojamajam įrenginiui tokios charakteristikos buvo ganėtinai didelės, tačiau čia ir baigėsi visi „Newton MessagePad 100“ privalumai. Plačiai išgirta rašto atpažinimo technologija veikė labai prastai, labai retai kada pavykdavo parašyti bent vieną žodį be klaidų.

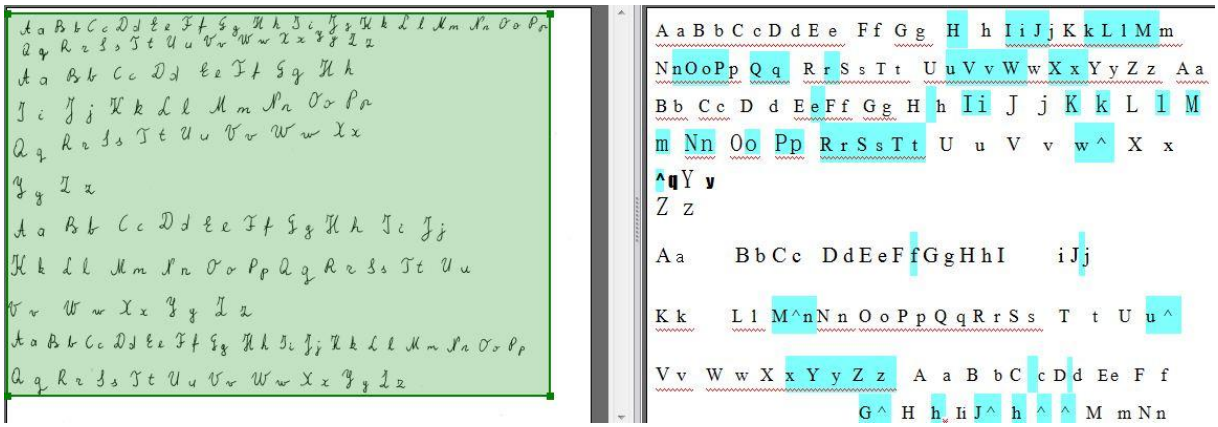
Tai tapo pagrindine priežastimi kodėl net ir naujesni „Newton“ modeliai, kurie jau turėjo ištaisytas teksto atpažinimo klaidas, nesulaukė didesnio populiarumo[8].



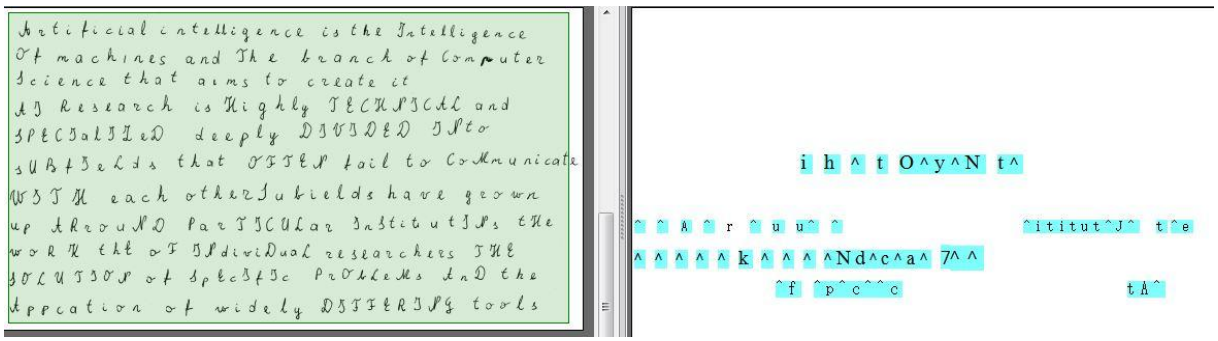
20 pav. Apple Newton kišeninis kompiuteris

Finereader programos testų rezultatai

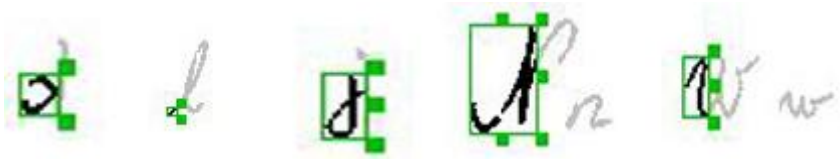
Žemiau esančiame paveikslėlyje pateikiamas programos atpažinimo rezultatas, naudojant testo rinkinį su abėcėlėmis. Šio testo metu programa atpažino didžiąją dalį simbolių.



Testo duomenys su atsitiktinio dydžio raidėmis parašytu tekstu daug prastesni.



Sistemos apmokyme klaidingai atpažintos simbolių ribos:



4 priedas

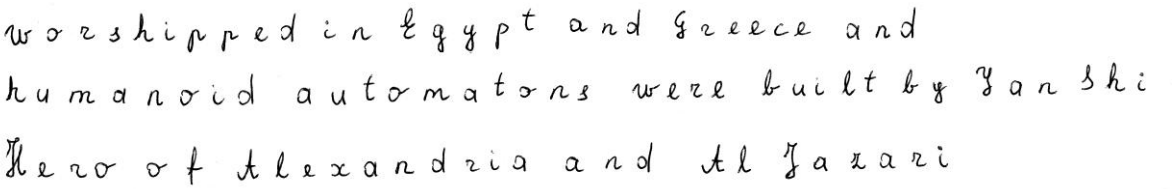
Testų rezultatai

Teisingas variantas	Bitų palyginimo rezultatas	Neuronų tinklų rezultatas
A	A	Z
A	n	d
A	N	n
c	C	d
C	C	C
e	L	n
e	e	D

Daugiau duomenų pateikta CD laikmenos aplanke testai, rinkmenoje klaidu analize.xls

Segmentavimo rezultatai

Simboliais išdalintas žodis iš teksto



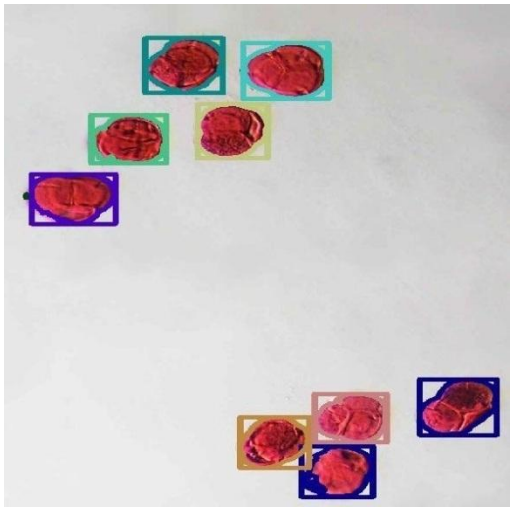
Daugiau duomenų pateikta CD laikmenos aplanke testai esančiuose aplankuose 1-11.

Programos rezultatai su žiedadulkių atvaizdais

Programai testuoti buvo naudojami žiedadulkių atvaizdai rasti internete.

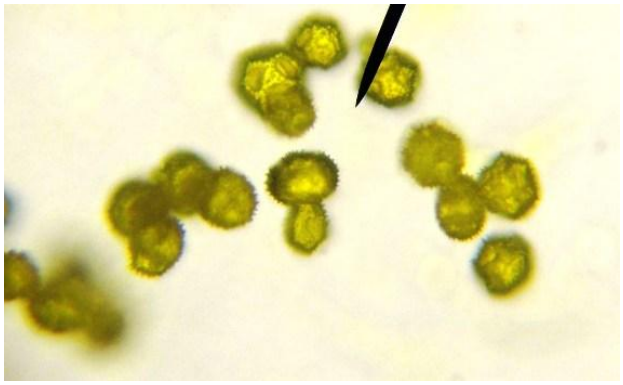
Rododendro žiedadulkės:

Rezultatas:

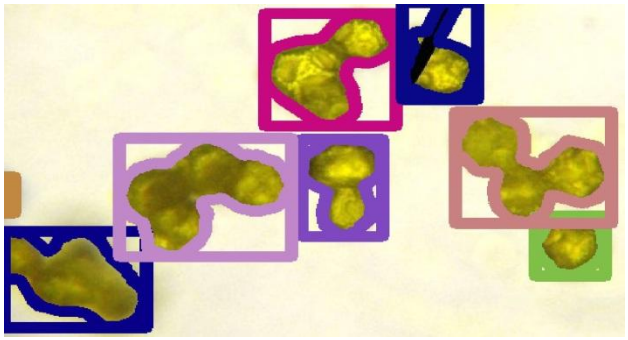


Žiedadulkės tiksliai atskiriamos dėl jos kontrastingumo ir ryškių kontūrų.

Pienių žiedadulkės:

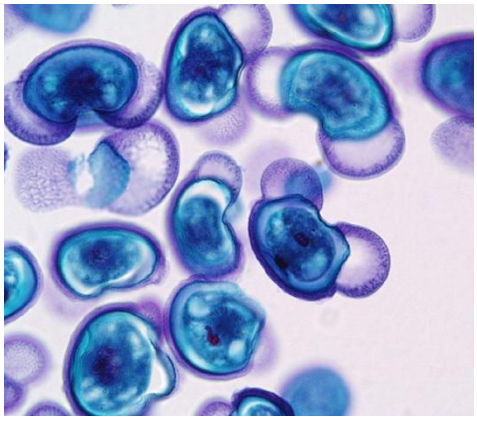


Rezultatas:

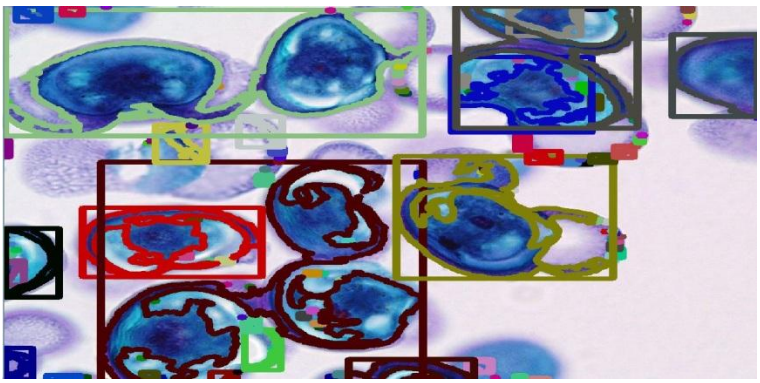


Kaip ir rododendro atveju žiedadulkės turi ryškias ribas su jose esančiu fonu dėl to aptikimas vyksta tiksliai.

Pušų žiedadulkės:



Rezultatas:



Pušų žiedadulkių struktūra ir atspalviai labai skiriasi. Tai suklaidina programą ir žiedadulkių ribas išskiria netiksliai. To galima išvengti pritaikius kontrasto filtrus apdorojamai nuotraukai. Kol atliekama lašelio aptikimo procedūra, galima naudoti vieną atvaizdą kuriame išskirtos ribos, o kai reikia atskirti atskiras žiedadulkes į pavienius objektus galime naudoti pirminį vaizdą. Taip būtų padidinamas programos tikslumas ir tuo pačiu išsaugojami žiedadulkių atvaizdai.

Šie ir dar kiti pavyzdžiai, aukštesnės kokybės paveikslėliuose pateikiami CD laikmenoje, aplanke testai/ziedadulkes.