VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

PIRŠTO ANTSPAUDO MODELIS MINIMALIAIS ŽIEDAIS

Atliko: 6 kurso kompiuterinio modeliavimo
grupės studentas
Eduard Liudkevič

Darbo vadovas:
Dr. Algirdas Bastys

Vilnius
2006

Rodyklė

Anotacija

Šiame darbe yra nagrinėjamas piršto antspaudo atpažinimo uždavinio vieną iš sudedamųjų dalių: skaitmeninės informacijos apie piršto antspaudą gavimas. Aprašomas metodas, paremtas kreivių kreivumų įvertinimu bei minimalaus žiedo sąvoka. Taip pat, aprašytas naujas minimalaus žiedo skaičiavimo bei kreivių kreivumų įvertinimo algoritmai. Darbo tikslas - pagerinti piršto antspaudo atpažinimo algoritmo kokybę, bei greitį.

Rezultatų analizė ir palyginimas vykdomas pasitelkus UAB Neurotechnologija programine įranga (SDK). Rezultatams gauti, buvo sukurta atskira programa (JAVA programavimo kalba). Prieduose yra pateikiami grafikai vaizduojantys pasiektus rezultatus.

Rezultate, UAB Neurotechnologijos gaunami duomenys kartu su duomenimis, gautais pagal naują metodą, davė geresnį atpažinimo prasme rezultatą.

Summary

The recognition of fingerprint is discussed in this article. The goal of this work is to increase the quality of fingerprint recognition method, and to improve algorithm speed.

The new method of fingerprint data for fingerprint matching is analyzed. It concentrates on calculating values of curve curvatures, and minimum-width annuli. Some new methods of evaluation of this properties are described step by step. UAB Neurotechnologija software is used to analyze new algorithm efficiency. A new Java application with SDK was developed to implement the new algorithm.

The results of this work are analyzed at the final part of this paper and the illustration by the ROC curves are given in the appendix.

Įvadas

Piršto antspaudo atpažinimo uždavinys yra aktualus tokiose srityse kaip saugumas, kriminalistika, bei pan. srityse. Ieškoma pakankamai greitų bei tikslių, atpažinimo prasme, algoritmų.

Piršto antspaudo identifikavimo procesas susideda iš kelių dalių. Pradedant nuo skenavimo, skaitmeninės informacijos gavimo iki sprendimo priėmimo dėl tikrumo, priėmimo.

Šiame darbe nagrinėjama viena iš šio proceso dalių: skaitmeninės informacijos apie piršto antspaudo gavimą lyginimo operacijai naudoti. Tikslas pagerinti atpažinimo kokybę bei greitį.

Pirmoje dalyje apžvelgiami algoritmai, kurie bus naudojami piršto antspaudo skaitmeninio modelio sudarymui. Taip pat aprašomi naujai sukurti algoritmai.

Antroje dalyje nagrinėjamas piršto antspaudo modelis, skaitmeninės informacijos gavimo principai, jų praktinio naudojimo mechanizmas.

Trečiojoje dalyje aprašoma, kokiom priemonėm yra vykdoma analizė. Bei pati analizė.

Praktiniam darbui atlikti buvo parašyta programa JAVA programavimo kalba. Taip pat sukurtas specializuotas SDK programos veikimui, palengvinantis uždavinio sprendimo programavimą.

1. Algoritmų apžvalga

Šiame skyriuje apžvelgsime keletą algoritmų, kurie bus naudojami modeliuojant piršto antspaudą. Apie juos plačiau galima pasiskaityti šaltiniuose **[Скв02], [Sib73], [Ско98], [Иль85]**.

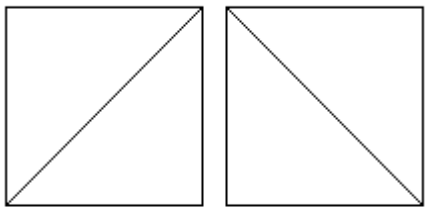
1.1. Delaunė trianguliacija

Delaunė trianguliacijos apibrėžimas:

Delaunė trianguliacija vadinama toks plokštumos taškų M_i trejetų sujungimas trikampiais, kad nubrėžtame per bet kurio trikampio viršūnes apskritimo viduje nėra nei vieno M_i taško.

Jei atsiranda N>3 taškų, per kuriuos galima nubrėžti apskritimą, kuriame nėra daugiau taškų, tada tokių taškų aibei Delaunė trianguliaciją galima atlikti nevienareikšmiškai, o tokie duomenys vadinami išsigimusiais.

Pvz.: tarkime turime 4 taškus, kurie sudaro kvadratą:



Pav. 1. Nevienareikšmės trianguliacijos pavyzdys

Akivaizdu, kad Delaunė trianguliacija bus nevienareikšmė.

Praktiškai tokia situacija pasitaiko retai. Tokių duomenų atveju galima nežymiai pakeisti trijų koordinačių padėtis taip, kad gauti vienareikšmę Delaunė trianguliaciją.

Delaunė trianguliacija skaido plokštumą į tokius trikampius, kurie siekia būti kuo labiau lygiakraščiais.

Kadangi trikampis yra paprasčiausias poligonas ir jo viršūnės vienareikšmiškai nurodo briaunas, tai trikampiai plačiai naudojami 3D grafikoje, paviršių modeliavime ir t.t. Be to, kiekvieną daugiakampį garantuotai galima suskaidyti į trikampius.

Delaunė trianguliaciją N taškams greičiausiai galima surasti per O(N log N) laiką.

Duotų taškų Delaunė trianguliacija apibrėžia iškiląjį daugiakampį.

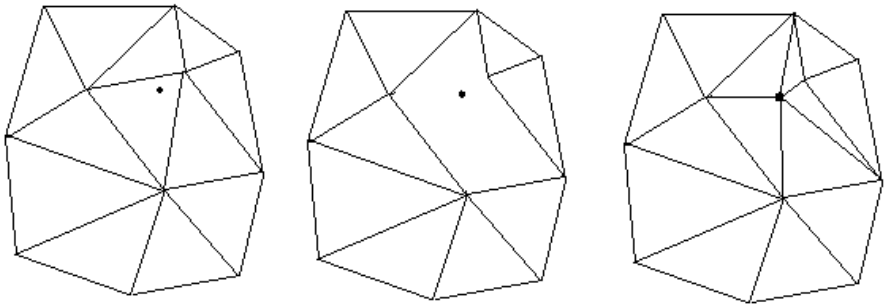
Delaunė trianguliacija neblogai tinka mastelio keitimui. Pvz. vienmačiu atveju taškus galima surūšiuoti reikšmių didėjimo arba mažėjimo tvarka ir išmesti kas antrą. Kadangi Delaunė

trianguliacija apibrėžia vieną plokštumos taškų rūšiavimo būdų, galime elgtis analogiškai. Iš duotųjų plokštumos taškų pasirenkame tokius, kad atstumas tarp jų būtų dvi Delaunė trianguliacijos kraštinės. Likusiems taškams galima vėl atlikti Delaunė trianguliaciją ir kartoti procedūrą rekurentiškai. Gausime hierarchinę pradinių taškų išretinimo struktūrą.

Delaunė trianguliacijai duotiems taškams rasti yra sugalvota daug algoritmų. Kelis iš jų paminėsiu.

1.1.1. Iteracinis algoritmas "Naikink ir statyk"

Tarkime turime kažkokią pradinę Delaunė trianguliaciją. Pridėjus naują tašką, iškart naikinami tie trikampiai, į kurių apibrėžtus apskritimus pakliūna naujas taškas. Panaikinti trikampiai netiesiogiai apibrėžia daugiakampį. Taigi, panaikintų trikampių vietoje sudaroma daugiakampį užpildanti trianguliacija. Tai padaroma sujungiant naują tašką su daugiakampiu.



Pav. 2. Algoritmo žingsniai

Šiame algoritme į pirmą planą išeina daugiakampio kontūro radimo procedūra, nuo kurios efektyvumo priklauso algoritmo greitis. Plačiau yra aprašyta straipsnyje **[Скв02]**.

1.1.2. Sujungimo algoritmas "Dalink ir valdyk"

Šiame algoritme taškų aibė dalinama į kaip galima lygesnes dvi dalis, be to, skirtinguose rekursijos lygiuose dalijimas vyksta paeiliui horizontaliom arba vertikaliom linijom. Algoritmas rekursiškai naudojamas kiekvienai daliai, o tada vykdomas trianguliacijų suliejimas. Rekursija sustabdoma, kai taškų visuma suskirstoma į pakankamai mažas dalis, kurias galima lengvai sutrianguliuoti kitu lengvu būdu. Praktikoje naudojamas dalijimas į 3-4 taškus. Jei taškų skaičius $N > 5$, tai taškų aibę visada galima suskaidyti į dalis po 3-4 taškus.

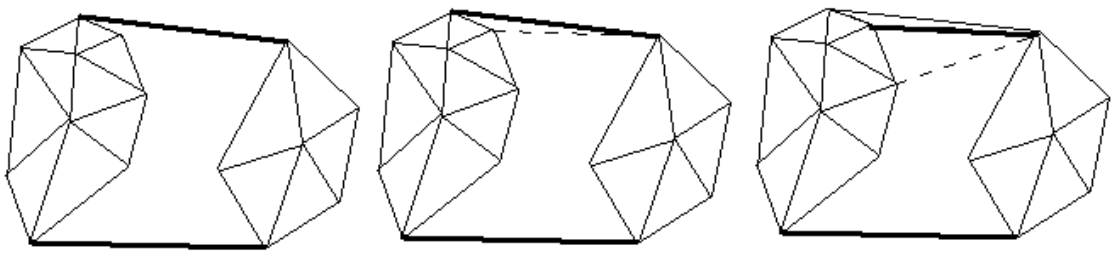
Algoritmo schema:

1. Jei taškų skaičius N = 3, tai pagaminame trianguliaciją iš vieno trikampio;
2. Jei taškų skaičius N = 4, tai pagaminame trianguliaciją iš 2 ar 3 trikampių;
3. Jei taškų skaičius N = 8, tai taškus reikia suskaidyti į dvi grupes po 4 taškus, rekursiškai panaudoti algoritmą ir suklijuoti gautas trianguliacijas;
4. Jei taškų skaičius N < 12, tai taškus reikia padalinti į dvi dalis po 3 ir N – 3 taškus, rekursiškai panaudoti algoritmą ir suklijuoti gautas trianguliacijas;
5. Jei taškų skaičius N > 11, tai taškų aibę dalinti į dvi dalis po lygiai, ar beveik lygiai taškų ir rekursiškai panaudoti algoritmą.

Pagrindinis algoritmo aspektas yra suklijuoti dalines trianguliacijas. Tam gali būt naudojamas suliejimo algoritmas “Naikink ir statyk”. Plačiau yra aprašyta straipsnyje **[Скв02]**.

1.1.3. Suliejimo algoritmas “Naikink ir statyk”

Pradžioje dviems trianguliacijoms randamos bendros liestinės, iš kurių viena tampa aktyviąja bazine linija. Tuomet nuo šios linijos pradedamas tarpo tarp trianguliacijų užpildymas. Tam reikia rasti artimiausią tašką (iš bet kurios turimos trianguliacijos) bazinei linijai. Iš bazinės linijos ir rasto taško gaminamas naujas Delaunė trikampis. Gali pasitaikyti tokių atvejų, kai reikia sunaikinti jau esamą trianguliacijos trikampį, kurį perdengia naujasis trikampis. Po šių veiksmų, nauja bazine linija tampa atvira trikampio kraštinė. Skaičiavimai vykdomi iki kol bus pasiekta antroji liestinė.



Pav. 3. Algoritmo žingsniai

Šiam algoritmui būdingos pakankamai didelės laiko sąnaudos naujo taško radimui.

1.1.4. Rekurentinis minimalių atkarpų algoritmas Delaunė trianguliacijai rasti

Šis algoritmas yra naujas autoriaus sugalvotas algoritmas. Veikimo principas paremtas atkarpų kryptingumo įvertinimų.

Pirmas žingsnis:

perrenkant visus taškus, randame taškų porą: $M_a = (x_a, y_a)$ ir $M_b = (x_b, y_b)$, tarp kurių atstumas yra mažiausias:

$M_a = (x_a, y_a), M_b = (x_b, y_b) : | M_a – M_b| = \min(|M_i – M_j|), i,j \in (0, N-1), i \neq j.$

Surastieji taškai M_a ir M_b sudaro pradinę briauną, sudarytą iš dviejų briaunos vektorių M_aM_b ir M_bM_a. Šiuos vektorius įtraukiame į miegančių briaunų vektorių sąrašą. Delaunė trianguliacijų paieškos algoritmą pradedame nuo bet kurio iš jų.

Vykdant perrinkimo operaciją minimaliam atstumui rasti, kartu galime rasti ir didžiausią atstumą tarp dviejų taškų (šį dydį panaudosime minimalaus žiedo radimo algoritme).

Antras žingsnis:

1. pasirenkam bet kurį miegančios briaunos vektorių;
2. perrenkant taškus M_j, j =0..N-1, ieškome tokio taško, kad tenkintų dešininę sąlygą pasirinkto vektoriaus atžvilgiu;
3. jei taškas yra kairėje arba ant tiesės, tęsiam perrinkimą (reikia pastebėti, kad pirmame žingsnyje joks taškas nepriklauso pradinei briaunai, kadangi ji trumpiausia iš visų galimų);
4. jei taškas dešinėje ir neturime trikampio kandidato, tuomet tariame, kad turime Delaunė trikampį – kandidatą (jį sudarys naujas taškas ir briaunos vektorius), apskaičiuojame apskritimo, apibrėžto per turimus 3 taškus, centrą ir spindulį, tęsiame taškų perrinkimą;
5. jei taškas dešinėje ir turime trikampį kandidatą, tuomet patikrinam, ar šis taškas pakliūna į kandidato apibrėžiamą apskritimą:
    - jei pakliūna, tuomet kandidatą formuojame iš naujo, panaudojant briaunos vektorių bei naują tašką;
    - jei nepakliūna, tuomet tęsiame taškų perrinkimą.

Perrinkimas baigiamas, kai yra peržiūrėti visi taškai.

Jei perrinkus visus taškus neatsirado kandidato, vadinasi briaunos vektorius priklauso iškilajam apvalkalui. Pridedam briauną prie iškilojo apvalkalo briaunų sąrašo, o briaunos vektorių išmetam iš miegančių briaunų vektorių sąrašo.

Suradę naują trikampį, kiekvieną trikampio vektorių patikrinam, ar jis priklauso kuriai nors miegančiai briaunai:

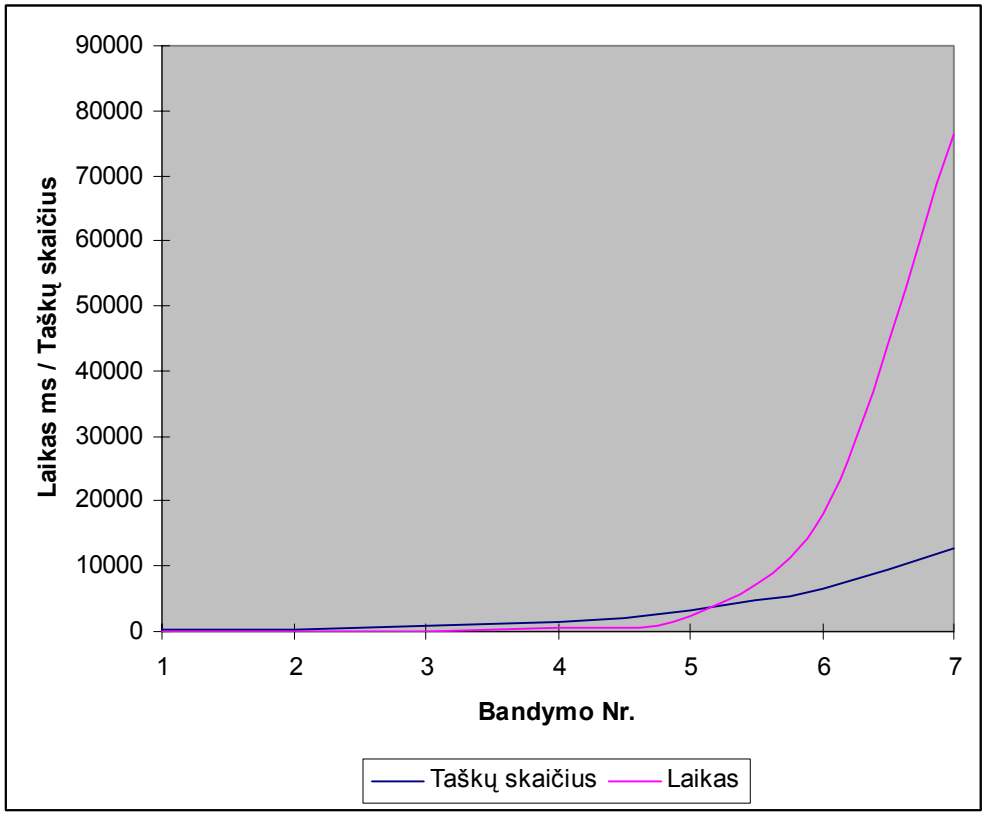
- jei priklauso, tai šią briauną padarome mirusią ir jungtinį briaunos vektorių išmetam iš miegančių briaunų vektorių sąrašo;
- jei nepriklauso, tai šį briaunos vektorių pridedam prie miegančių briaunų vektorių sąrašo.

Trikampį kandidatą įtraukiam į trikampių sąrašą.

Algoritmo greitumui patikrinti buvo atlikti praktiniai skaičiavimai. Rezultatai pateikti lentelėje bei grafike.

Bandymo Nr.	**Taškų skaičius**	**Laikas (10^{-3} sek)**
1	200	31
2	400	31
3	800	141
4	1600	562
5	3200	2297
6	6400	18141
7	12800	76328

Lentelė 1. Skaičiavimo laikas skirtingam taškų skaičiui.



Pav. 4. Skaičiavimo laiko augimo bei taškų didėjimo grafikas

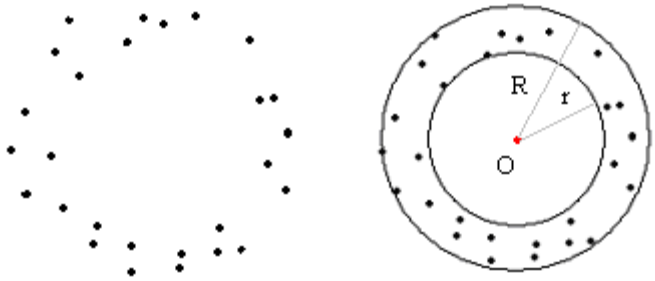
Kaip matyti iš testinių skaičiavimų, taškų skaičiui padidėjus dvigubai, skaičiavimo laikas vidutiniškai padidėja keturgubai. Laiko prasme algoritmas vidutiniu atveju yra $O(N^2)$ sudėtingumo. Palyginimui iteracinis algoritmas „naikink ir statyk“ yra $O(N^{3/2})$ sudėtingumo, tačiau jo realizacija yra gana sudėtinga. Be to, naujasis algoritmas turi tam tikrų pliusų, kurie bus reikalingi minimalaus žiedo skaičiavime modeliuojant piršto antspaudą.

Naujo algoritmo pliusai yra tokie:

- skaičiavimo laikas nedideliam taškų skaičiui yra labai mažas;
- įvykdžius ši algoritmą kartu su Delaunė trianguliacija be papildomų skaičiavimų gaunami keletas turimos taškų aibės parametrų: didžiausias atstumas tarp dvejų taškų, bei iškiląjį apvalkalą. Šie parametrai labai pravers minimalaus žiedo skaičiavime.
- paprasta realizacija.

1.2. Minimalus žiedas

Plokštumos taškų minimaliu žiedu vadiname minimalaus storio žiedą, į kurį įeina visi duotieji taškai.



Pav. 5. Minimalaus žiedo pavyzdys

Minimaliam žiedui gauti reikia rasti:

1. centrą O;
2. apskritimo, kurio viduje būtų visi duotieji taškai, spindulį R;
3. apskritimo, kurio viduje nebūtų nei vieno taško, spindulį r;

kad žiedo storis S = R-r būtų minimalus.

Fiksavus centrą O, galima rasti artimiausią tašką, iki kurio atstumas nuo O sutaps su r, ir tolimiausią tašką, iki kurio atstumas nuo O sutaps su R. Taigi r visuomet mažesnis nei R.

Minimalus žiedas yra taikomas:

- kompiuterinėje grafikoje;

- aproksimuojant duomenis (kompiuterinė geometrija, optimizavimas);
- atpažinime.

Keletas teoremų, kuriomis naudosiuos minimalaus žiedo suradimui.

Teorema 1

Tarkime yra duota N>3 plokštumos taškų M_0, M_1, ..., M_{N-1}, kuriems reikia rasti minimalų žiedą. Tuomet tarp šių taškų atsiras keturi taškai A=M_{i1}, B=M_{i2}, C=M_{i3}, D=M_{i4}, iš kurių pirmieji du (A ir B) priklausys vidiniam apskritimo minimaliam žiedui, o likę du (C ir D) priklausys išoriniam minimaliam žiedui.

Minimalaus žiedo centras bus statmenų, išvestų per atkarpų [A,B] ir [C,D] vidurio taškus, susikirtimo taškas.

Teorema 2

Tarkime yra duota N>3 plokštumos taškų M_0, M_1, ..., M_{N-1}, kuriems reikia rasti minimalų žiedą. Tuomet tarp šių taškų atsiras keturi taškai A=M_{i1}, B=M_{i2}, C=M_{i3}, D=M_{i4}, iš kurių pirmieji du (A ir B) priklausys vidiniam apskritimo minimaliam žiedui, o likę du (C ir D) priklausys išoriniam minimaliam žiedui.

Vienas iš algoritmų minimaliam žiedui rasti yra aprašytas **[PAA99+],** tačiau bus pasiūlytas naujas algoritmas, naudojantis Delaunė trianguliaciją.

Taškai A ir B yra kurio nors Delaunė trianguliacijos trikampio viršūnės, o C ir D yra taškų M_n tiesinio apvalkalo krašte. Minimalaus žiedo centras bus statmenų, išvestų per atkarpų [A,B] ir [C,D] vidurio taškus, susikirtimo taškas.

1.2.1. Minimalaus žiedo skaičiavimo algoritmas

Suradę turimos taškų aibės $M_i \in R^2$ kur $i \in (0, N-1)$ Delaunė trianguliaciją, turime šiuos duomenis:

- Briaunų, priklausančių Delaunė trianguliacijai, sąrašą;
- Iškiliojo apvalkalo briaunų sąrašą;
- Didžiausią atstumą tarp dviejų taškų.

Skaičiuojant minimalų žiedą dirbsime tik su briaunomis (nekreipiant dėmesio į kryptį).

Žingsniai:

1. fiksuojam bet kurią iškilojo apvalkalo briauną;
2. fiksuojam Delaunė trianguliacijos briauną;
3. turimoms dviems briaunoms atitinkamai surandam statmenas tieses, kurios eina per briaunų vidurio taškus;
4. randam tašką O, kuriame kertasi šios tiesės (šis taškas yra žiedo-kandidato centras, vidinio apskritimo spindulys yra atstumas nuo O iki bet kurio pasirinktos Dėlaunė trianguliacijos briaunos kraštinio taško, o išorinio apskritimo spindulys yra atstumas nuo O iki pasirinktos iškilojo apvalkalo briaunos kraštinio taško);
5. patikrinam, ar vidinio apskritimo viduje bei išorinio apskritimo išorėje nėra taškų:
    - jei yra, tuomet rastas žiedas nėra kandidatas būti minimaliu žiedu;
    - jei nėra, tuomet rastas žiedas yra kandidatas būti minimaliu žiedu. Tikrinam, ar jo storis mažesnis už prieš tai buvusį. Jei mažesnis, fiksuojam jo parametrus, jei ne, kandidatas yra netinkamas ir pereinam į 6-ą žingsnį;
6. jei fiksuotai iškilojo apvalkalo briaunai dar nepatikrinom visų Delaunė trianguliacijos briaunų, tuomet vykdom 2-ą žingsnį, kitu atveju vykdom 1-ą žingsnį.
7. algoritmas baigiamas, kai perrenkamos visos iškiliojo apvalkalo briaunos.

Didžiausią dalį skaičiavimų užima patikrinimas, ar visi taškai priklauso surastam žiedui – kandidatui. Skaičiuojant taškų aibės Delaunė trianguliaciją, kartu randame vieną šios taškų aibės savybę: didžiausią atstumą tarp dviejų taškų. Tai yra, pats mažiausias minimaliausio išorinio apskritimo spindulys R turi tenkinti šią sąlygą:

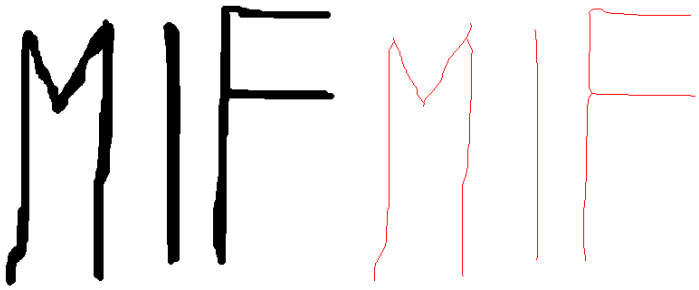
$$2*R \geq |M_a - M_b| = \max(|M_i - M_j|),\ i,j \in (0, N-1),\ i \neq j.$$

Tačiau į šį apskritimą nebūtinai paklius visi aibės taškai. Bet turint šį dydį, galima atsisakyti dalies skaičiavimų, iš anksto žinant, jog jie bus beverčiai.

Taigi, šią sąlyga galime panaudoti algoritme. Ją vykdome prieš 5-ą žingsnį. Tai yra, suradę žiedą-kandidatą, patikrinam sąlygą. Jei ji patenkinama, tuomet vykdome žingsnį 5, jei ne - vykdome žingsnį 6. Tokiu būdu sumažiname taškų priklausomybės žiedui tikrinimo kiekį ~50% (Toks rezultatas gautas skaičiuojant praktiškai).

1.3. Skeletizavimas

Skeletizavimo algoritmas naudojamas vaizdo skeletui išgauti. Skeletas - tokia taškų struktūra, kuri turimą vaizdą atvaizduoja minimalaus storio kreivėmis:



a) b)

Pav. 6. Skeleto pavyzdys. a) originalus vaizdas, b) vaizdo skeletas.

Vaizdo skeletui gauti pasirinkau Hilditch 3x3 skeletizavimo algoritmą.

1.3.1 Hildicht 3x3 algoritmas:

Dirbame su 8-iais taško kaimynais:

P9	P2	P3
P8	P1	P4
P7	P6	P5

Pav. 7. Taško kaimynai

Mums yra svarbu patikrinti, ar taškas yra skeleto dalis ir j netrinti, ar j galima išmesti. Aprašome dvi funkcijas:

- A(p1) = 0,1 šablon skai ius sekoje p2, p3, p4, p5, p6, p7, p8, p9, p2;
- B(p1) = netuš i kaimyn skai ius.

Taškas yra išmetamas iš vaizdo, jei yra tenkinamos šios 4 s lygos:

- 2 <= B(p1) <= 6;
- A(p1) = 1;
- p2.p4.p8 = 0 arba A(p2) != 1;
- p2.p4.p6 = 0 arba A(p4) != 1.

Algoritmas sustabdomas, kai nei vienas taškas n ra pakei iamas.

1.4. Kreivės didžiausio kreivumo įvertinimas

Turint kreivę, galima įvertinti, kurioje vietoje ji įgyja didžiausią kreivumą. T.y. pasirinkus kreivės atkarpą, ją galima aproksimuoti apskritimu ir taip gauti apskritimo centro koordinates bei spindulį. Spindulys ir apibrėžia kreivumą. Kadangi mums nėra svarbu turėti tikslų geometrinį kreivumą, mes galime šiek tiek kitaip įvertinti šį dydį.

Vienas iš algoritmų būtų toks:

- kreivės atkarpos ilgį pasirenkam kaip parametrą N;
- kreivės atkarpos pirmą ir paskutinį tašką jungiam tiese AB;
- randam atkarpos AB vidurio tašką C;
- randam mažiausią atstumą nuo taško C iki kreivės.

Taip pereinam per visą kreivę. Didžiausias iš turimų mažiausių atstumų apibrėš didžiausią kreivės kreivumą.

Kitas algoritmas:

- kreivės atkarpos ilgį pasirenkam kaip parametrą N;
- randam kreivės atkarpos vidurio tašką D;
- kreivės atkarpos pirmą ir paskutinį tašką jungiam tiese AB;
- randam atstumą nuo taško D iki atkarpos AB.

Kaip ir pirmame algoritme, pereinam per visą kreivę. Didžiausias toks atstumas apibrėš didžiausią kreivumą.

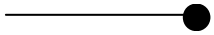
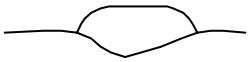
Taigi, rezultate turėsim tokius duomenis:

- taško koordinates (pirmo algoritmo atveju – taško C koordinates, antru atveju – D koordinates);
- skaitinę kreivumo išraišką, t.y. atstumą.

2. Piršto antspaudas

2.1. Piršto antspaudo modelis

Vienas iš asmens identifikavimo būdų yra piršto antspaudas. Piršto antspaudas yra unikalus kiekvienam asmeniui ir netgi kiekvienam pirštui. Jis susideda iš linijų, jų nutrūkimų, išsišakojimų, persikirtimų. Taškai, kuriuose įvyksta linijų nutrūkimas arba išsišakojimas, naudojami skaitmeniniam piršto antspaudo modeliui sudaryti. Tokie taškai vadinami ypatingais (angl. Minutiae). Kartais yra analizuojamos sudėtingesnės linijų kombinacijų struktūros. Pagrindiniai tipai parodyti lentelėje:

	Nutrūkimas (Termination)
	Bifurkacija (Bifurcation)
	Ežeras (Lake)
	Nepriklausoma atkarpa (Independent ridge)
	Taškas (Point)
	Pentinas (Spur)
	Perėja/Susikirtimas/Tiltas (Crossover/Bridge)

Lentelė 2. Ypatingų taškų tipai

Šiame darbe nagrinėjami dviejų rūšių taškai: nutrūkimas ir bifurkacija. Iš esmės nutrūkimas ir bifurkacija yra viena kitai dualios. Padarius vaizdo spalvų apvertimą (invert) , akivaizdu, kad taškas, buvęs pradiniame vaizde nutrūkimų, apverstame vaizde bus bifurkacija. Ir atvirkščiai. Antspaudų lyginimo algoritmas veikia šių taškų pagrindu. Efektyviam atpažinimui vien tik taškų koordinačių neužtenka, reikia papildomos informacijos, kuri kuo tiksliau apibrėžtų kiekvieną tašką, įvertinus deformacijas (poslinkio, posūkio) skenavimo metu.

Pirštams skenuoti yra įvairių įrenginių, kurių skenavimo principai yra skirtingi. Taip pat išgaunama įvairi vaizdo rezoliucija. Kuo didesnė rezoliucija, tuo kokybiškiau galima vykdyti atpažinimą.

Pradinis skenuotas antspaudas yra binarizuojamas. Binarizacija vyksta pagal tam tikrus sudėtingus algoritmus. Apie keletą iš jų galima pasiskaityti knygoje **[MMJ03+]**. Savo darbe pasinaudoju UAB Neurotechnologijos VeriFinger programa, kuri binarizuoja pirmin pirštо antspaudo vaizd (žr. paveiksliuk).

Pav. 8. a) skenuotas pirštо antspaudas, b) VeriFinger programos apdorotas vaizdas

Po šio žingsnio, pirštо antspaudo binarinis vaizdas yra skeletizuojamas. Skeletizavimą galima atlikti pagal įvairius algoritmus. Šiame darbe yra pasinaudota Hidlich 3x3 skeletizavimo algoritmu, kuris yra aprašytas 1.3 paragrafe.

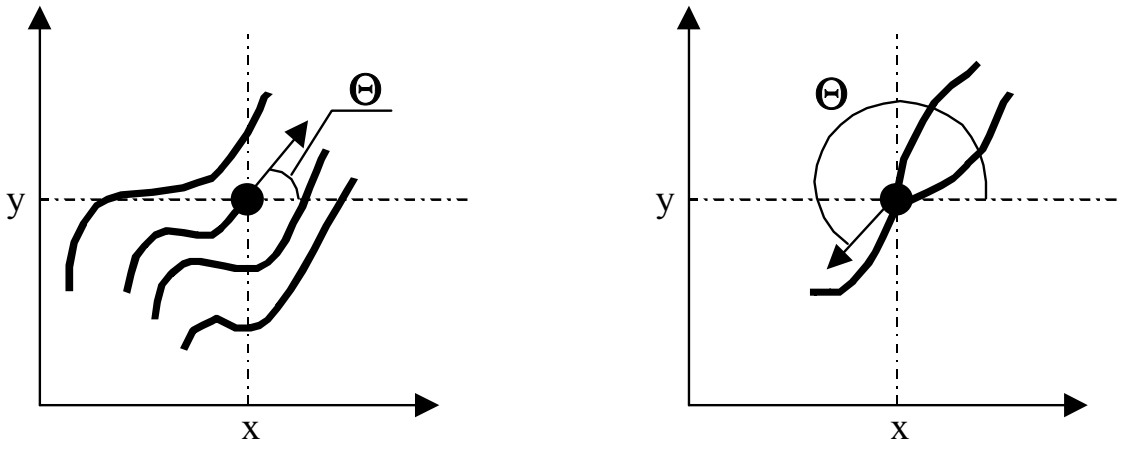
Pav. 9. Skeletas

Hidlich 3x3 skeletizavimo algoritmas gerai tinka skeletizuoti piršto antspaudo vaizdą, kadangi šis algoritmas garantuoja, jog kiekvienas taškas turės nuo 0 iki 3 kaimynų. Taigi, labai paprasta išrinkti nutrūkimo bei bifurkacijos taškus. Kiekvienas taškas skeletizuotame vaizde yra tam tikro tipo, pagal šį, taškų išskirstymo pagal kaimynų skaičių, sąrašą:

- 0: nepriklausomas taškas;
- 1: taškas yra kreivės galas arba pradžia, t.y. nutrūkimo taškas;
- 2: taškas yra eilinis kreivės taškas;
- 3: bifurkacija.

Taigi, mus domina taškai, kurie turi 1 arba 3 kaimynus, bei kreives, priklausančias šiems taškams. Surinkti kreivę iš skeletizuoto vaizdo irgi yra praprasta, tereikia pradėti nuo ypatingo taško ir eiti 2-o tipo taškais iki sekančio ypatingojo taško.

Paprasčiausias duomenų rinkinys, kuris leis atlikti atpažinimą, yra ypatingų taškų koordinatės bei šių taškų kampų su horizontu tangentai.



Pav. 10. Ypatingų taškų kampai su horizontu.

Toks būdas yra paprastas ir dažnai aprašomas literatūtoje. Kampo skaičiavimo metodika detaliai nebus aprašyta, kadangi praktiniams skaičiavimams bus naudojamas UAB Neurotechnologija skaičiavimo algoritmas.

Šio darbo tikslas - rasti papildomos informacijos, kuri pagerintų atpažinimo kokybę bei paspartintų piršto antspaudų identifikavimo algoritmą.

Idėja būtų tokia, kad įvertinti, ypatingiems taškams priklausančių kreivių, kreivumus. Kiekvienas nutrūkimo taškas turi vieną, jam priklausančią, kreivę, o bifurkacijos taškas – trys. Šios kreivės turi savo kreivumo maksimumus, kurie nepriklauso nuo piršto antspaudo posūkio.

Taškus, kuriuose pasiekiami kreivių maksimalūs kreivumai, bus bandoma surasti bei apibrėžti jų teikiamą informaciją.

Taigi, principinė algoritmo schema būtų tokia:

- rasti, kurioje kreivės vietoje pasiekiamas didžiausias kreivumas;
- ji įvertinti skaitiškai;
- įvertinti kreivumo kryptį;
- rasti minimalų žiedą atkarpos, kurioje pasiekiamas didžiausias kreivumas.

Šių duomenų įtaka atpažinimui bus įvertinama juos pridedant prie UAB Neurotechnologijos gautų duomenų ir vykdant jų atpažinimo algoritmą. Atpažinimo algoritmas nėra aprašomas, kadangi tai yra konfidenciali įmonės informacija.

Apie vieną iš piršto antspaudų lyginimo algoritmų galima paskaityti moksliniame straipsnyje **[KKK06].**

2.2. Papildomos informacijos gavimo algoritmas

Tikslaus kreivės kreivumo mes negalime nustatyti. Kreivės iškrypimai visoje kreivėje gali būti skirtingų krypčių. Taigi, tam, kad sąlyginai stabiliai ir tiksliai rastume kreivumo tašką, jo reikės ieškoti tam tikroje aplinkoje. Arba kitais žodžiais tariant, tam tikroje atkarpoje. Atkarpos ilgis negali būti vienareikšmiškas bendru atveju, kadangi keičiantis skenuoto piršto antspaudo rezoliucijai, visi dydžiai (atstumai tarp ypatingų taškų, kreivių kreivumai ir pan.) keičiasi. Taigi, kreivės ilgis bus imamas kaip parametras. Jo optimalią reikšmę galima nustatyti tik bandymų metu tiriant atpažinimo kokybę, prie skirtingų parametrų.

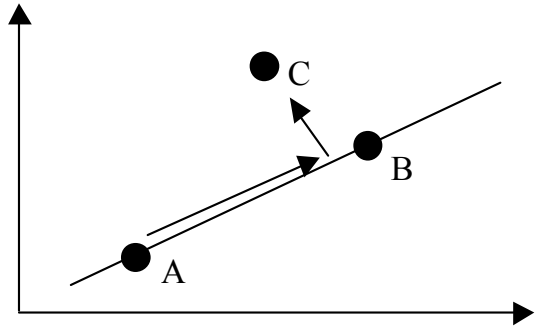
Taigi, nagrinėjam papildomos informacijos gavimo algoritmą. Kaip parametrą pasirenkam atkarpos ilgį, pagal kurią įvertinam kreivumą. Pasinaudojam kreivės didžiausio kreivumo radimo algoritmu, kuris aprašytas šio darbo 1.4 paragrafe.

Radę atkarpą kurioje kreivė įgyja didžiausią kreivumą, ją aproksimuojam minimaliu žiedu pagal minimalaus žiedo algoritmą aprašytą 1.2.1 paragrafe. Taip pat, įvertinam kreivės kreivumo kryptį. Tai galime padaryti, kadangi žinome kreivės pradžią, ir judėjimo ja kryptį. Krypties įvertinimas vykdomas pagal taško poziciją, kryptingos tiesės atžvilgiu. Pasinaudojam formule:

$$Sgn((B.x - A.x) * (C.y - A.y) - (C.x - A.x) * (B.y - A.y)),$$ kur

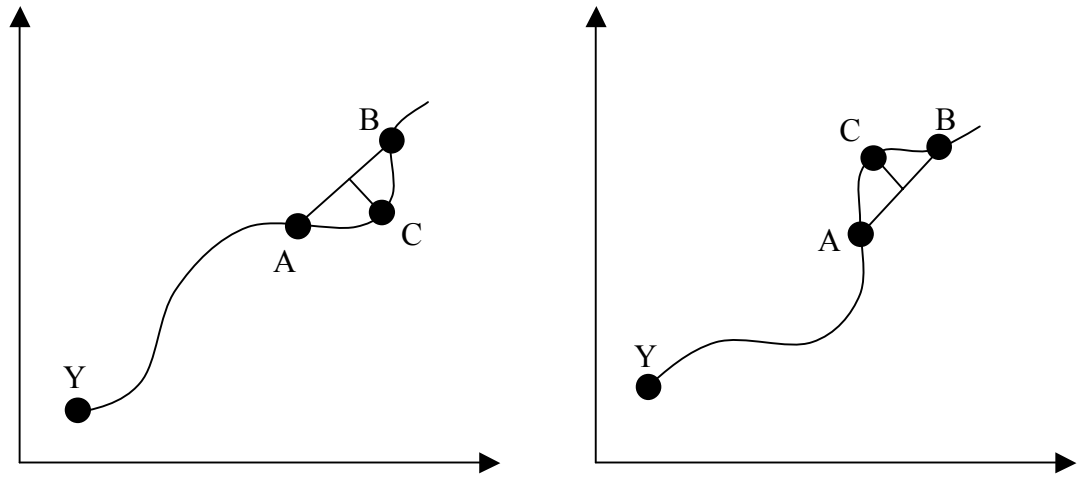
A, B – taškai, priklausantys tiesei, C – taškas, kurio pozicijos ieškome.

Iš esmės, krypties ženklas nustatomas pagal dešinės rankos taisyklę:



Pav. 11. Dešinės rankos taisyklė

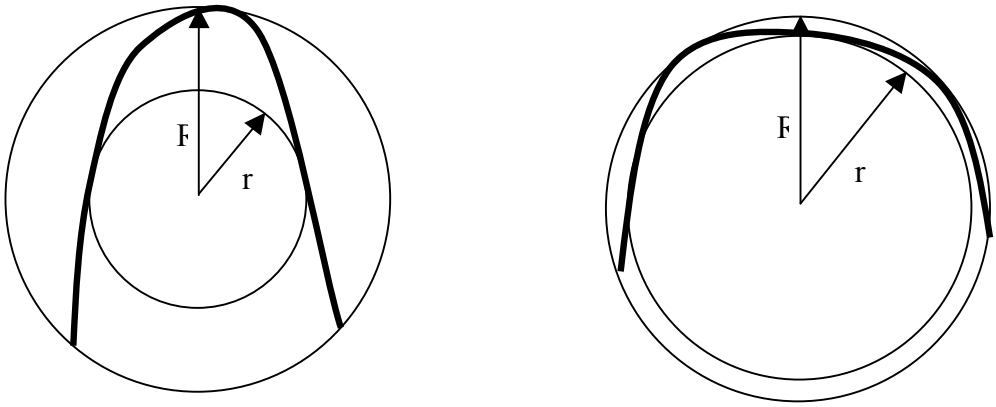
Kreivės kreivumo krypčių pavyzdžiai parodyti paveiksliuke:



Pav. 12. a) kreivumas neigiamas, b) kreivumas teigiamas

Paaiškinimas: Y – ypatingas taškas, A - kreivės atkarpos, kurioje pasiekiamas didžiausias kreivumas, pradžia, B – galas, C – kreivės atkarpos vidurio taškas.

Sekantis žingsnis - kreivės atkarpą AB aproksimuoti minimaliu žiedu. Tam pasinaudojama algoritmu, aprašytu 1.2.1 paragrafe. Minimalus žiedas mums suteikia tam tikros informacijos apie kreivės atkarpą. Kadangi didžiausias kreivumas ieškomas atkarpoje, o atkarpa nėra apskritimo formos, tai minimalaus žiedo storis apibrėžia kreivės atkarpos kreivumų svyravimą, kitaip sakant tam tikrą kreivės charakteristiką. Tarkim, turim dvi krevės atkarpas ir jas aproksimuojame. Kaip parodyta paveikslėlyje, minimalių žiedų centrai gali sutapti, tačiau minimalaus žiedo storis bus skirtingas.

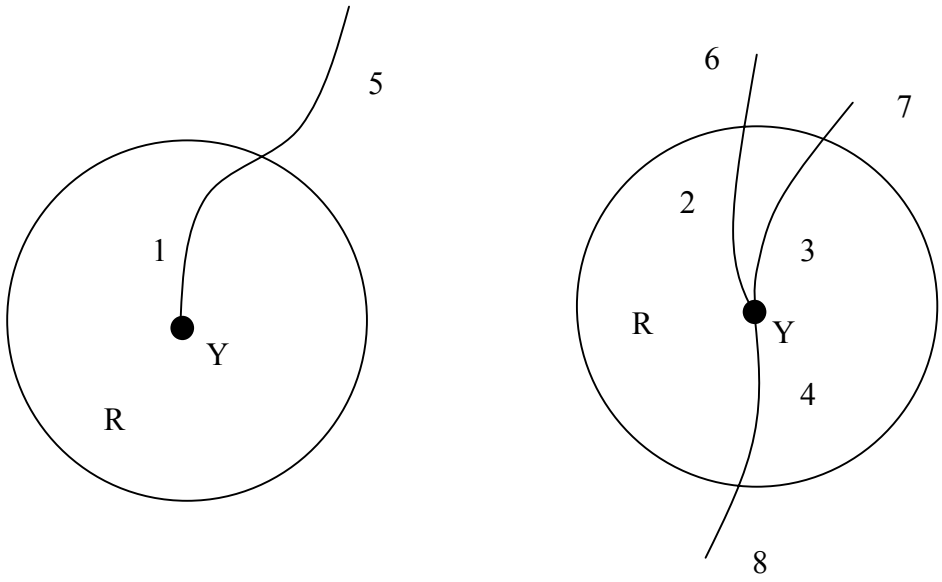


Pav. 13. Kreivės aproksimavimas minimaliu žiedu

Atlikę visus algoritmo veiksmus, turėsime tokius duomenis apie kreivės kreivumą:

- kreivės taško, kuriame įgyjamas didžiausias kreivumas, koordinates;
- kreivumo parametrą;
- minimalaus žiedo centro koordinates;
- minimalaus žiedo storį.

Kaip jau buvo minėta, nutrūkimo taškas turi jam priklausančią vieną kreivę, o bifurkacija - trys. Tikslumui padidinti, apsibrėžiam aplinka R. T.y. kreivės atkarpos ilgis, pradedant nuo ypatingojo taško. Algoritmą taikom kiekvienai kreivei: aplinkoje R ir per visą kreivę. Rezultate gausim aštuonis duomenų rinkinius.



Pav. 14. Duomenų rinkinių tipai

Duomenų rinkinių tipai: priklausomai nuo kreivės, ir kurioje jie randami (1-4 aplinkoje R, 5-8 per visą kreivę).

Sekančiame paragrafe bus nagrinėjamas šių duomenų poveikis atpažinimo algoritmui.

3. Rezultatų analizė

3.1. ROC (Receiver Operating Characteristic curve) kreivė

Atpažinimo algoritmo kokybės vertinimą priimta vykdyti sudarant ROC (Receiver Operating Characteristic curve) kreives.

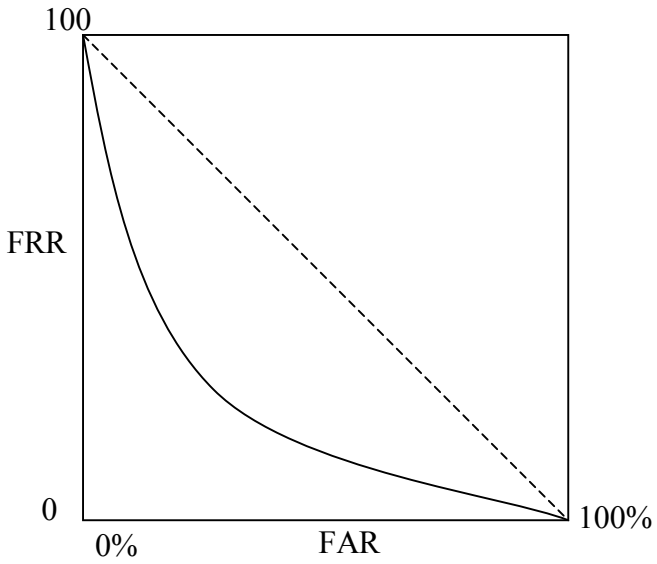
Vykdant atpažinimą, galimi 4 atpažinimo tipai:

1. teisingas atpažinimas;
2. teisingas neatpažinimas;
3. neteisingas atpažinimas;
4. neteisingas neatpažinimas.

Mus labiausiai domina 3 ir 4 tipai. Neteisingas atpažinimas – kai du skirtingų asmenų piršto antspaudai atpažįstami kaip vieno asmens. Neteisingas neatpažinimas – kai du vieno asmens piršto antspaudai atpažįstami kaip skirtingų asmenų. Šie klaidų tipai statistikoje įvardijami kaip I ir II tipo klaidos. Mūsų atveju jie vadinami FAR ir FRR, neteisingas atpažinimas ir neteisingas neatpažinimas atitinkamai. ROC kreivė apibrėžia priklausomybę tarp šių dydžių.

Atpažinimo algoritmui įvertinti reikia turėti testinių duomenų rinkinį. Atpažinimo algoritmas apskaičiuoja tam tikrą lyginimo skaitinę reikšmę. Tai pritaikom visiems testiniams duomenims. Turint reikšmių seką, reikia priimti sprendimą, kurios lyginimo operacijos davė teigiamą, o kurios neigiamą rezultatą. Sprendimui priimti įvedam slenkstį - tam tikrą skaitinę reikšmę, pagal kurią priimamas sprendimas.

Sudarant ROC kreivę, yra pereinama per visus galimus slenksčius ir tikrinant, koks rezultatų procentas pakliuvo į FAR arba FRR klaidas.



Pav. 15. ROC kreivės pavyzdys

Tokiu būdu sudarant ROC kreives, galima įvertinti atpažinimo algoritmo kokybę. Iš esmės, kuo mažesnis plotas lieka tarp kreivės ir procentų ašių, tuo algoritmas geresnis. Bet tai yra sąlyginai. Kadangi yra daug niuansų, kokiam tikslui algoritmas naudojamas. Pavyzdžiui, kriminalistikos srityje, siekiama, kad FRR klaida būtų kuo mažesnio procento. Nes yra svarbu, kad vykdant paiešką pagal duomenų bazę, nebūtų praleistas nei vienas potencialus nusikaltėlis. Tuo tarpu, jei įvyks blogas atpažinimas (FAR), kriminalistinio tyrimo eigai didesnės įtakos neturės, kadangi galutinį sprendimą priiminės žmogus.

Kitas pavyzdys: durų spyna, pagrįsta piršto antspaudo atpažinimu. Šiuo atveju siekiama, kad neįvyktų blogo atpažinimo klaidų (FAR), kadangi toks variantas reikš, jog pro duris galės praeiti nepageidaujamas asmuo. Tuo tarpu, jei įvyks blogas neatpažinimas (FRR), nieko blogo neįvyks. Tiesiog reikės pakartotinai skenuoti piršto antspaudą.

3.2. Gautų rezultatų analizė

Taigi, jau aprašyta, kaip išgauti papildomos informacijos apie piršto antspaudo ypatingus taškus. Reikia patikrinti, kiek šie duomenys turi įtakos atpažinimo algoritmui. Šio darbo duomenų patikrinimas paremtas pridedant šią informaciją prie UAB Neurotechnologijos gaunamos informacijos bei vykdant atpažinimo algoritmą. Tada, pagal gautus rezultatus konstruojamos ROC kreivės ir lyginamos su vien tik UAB Neurotechnologijos gautais rezultatais.

Lentelėje pateikti duomenys, formuojant ROC kreives su kiekvienu duomenų rinkiniu atskirai.

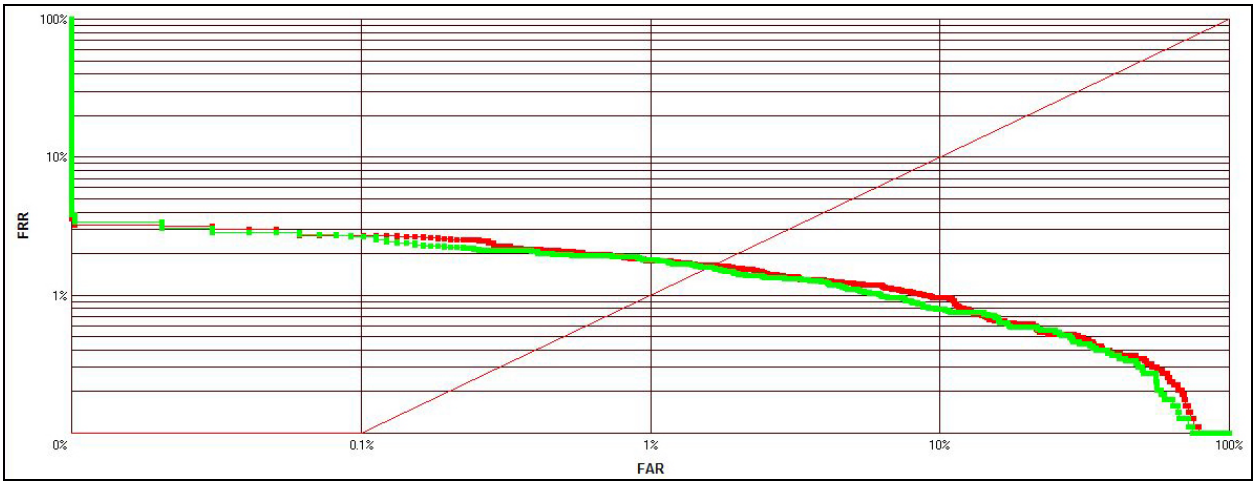
Duomenys	EER	Zero FAR	FAR 0.1%	FAR 1%	FAR 10%
UAB Neurotechnologija	1.65	3.59	2.70	1.78	0.952
+1 (50)	1.51	3.32	2.68	1.78	0.794
+2 (50)	1.60	3.81	2.75	1.81	0.952
+3 (50)	1.63	3.67	2.78	1.86	0.937
+4 (50)	1.68	3.65	2.78	1.84	0.952
+5 (50)	1.65	3.44	2.92	1.86	0.873
+6 (50)	1.65	3.84	2.86	1.87	0.937
+7 (50)	1.67	3.71	2.78	1.89	1.020
+8 (50)	1.65	3.71	2.81	1.84	0.968
+1 +4 (50)	1.59	3.86	2.65	1.81	0.794
+1 +4 (30)	1.67	4.90	2.78	1.87	0.952

Lentelė. 3. Gautų ROC kreivių duomenų lentelė

Paaiškinimas: eilutė UAB Neurotechnologija parodo reikšmes, kurios gautos vien tik UAB Neurotechnologija algoritmu. Reikšmės +1, +2 ir t.t. nurodo, su kokio tipo papildomais duomenų rinkiniais, pridėtais prie UAB Neurotechnologija duomenų, gauti rezultatai. Skliausteliuose nurodytas kreivės ilgio parametras taškais.

ROC kreivės pridedamos prieduose (žr. Priedą 1, 2 ir 3).

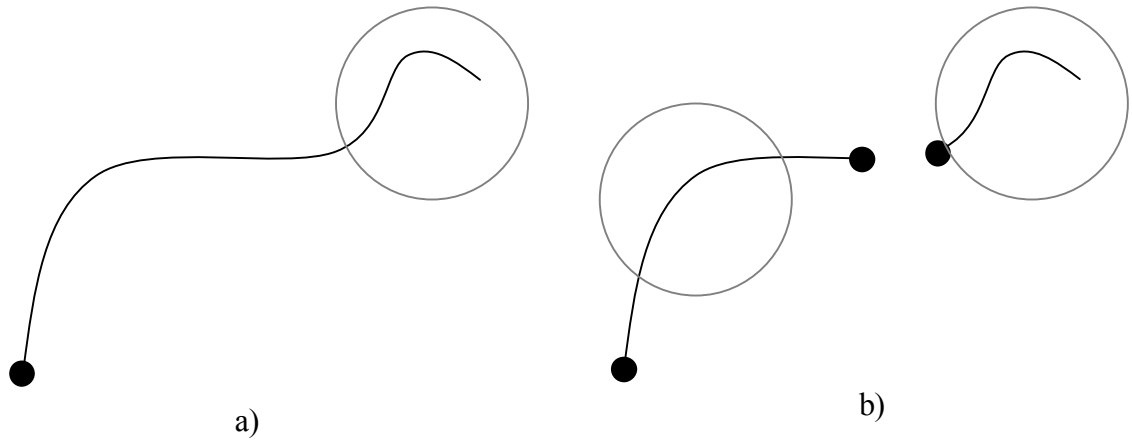
Išnagrinėjus gautus rezultatus matosi, kad esminio atpažinimo kokybės pagerėjimo nėra. Jis yra tik tam tikrose ROC kreivės vietose. Taip pat galima padaryti išvadą, kad didžiausias kokybės pagerėjimas pasiekiamas, kai naudojami 1-o arba 1-o ir 4-o tipo duomenų rinkiniai. Tai įvyksta todėl, kad šie rinkiniai geriausiai susisieja su ypatingų taškų tipais: nutrūkimu ir bifurkacija.



Pav. 16. Geriausias pagerėjimas pasiekiamas pridedant 1 ir 4 tipo duomenų rinkinius.
Raudona linija ROC kreivė atitinkanti UAB Neurotechnologija rezultatus,
žalia – UAB Neurotechnologijos kartu su 1 ir 4 tipo rinkiniais rezultatus.

Vienas svarbiausių rodiklių yra Zero FAR. Jis nurodo FRR (blogo neatpažinimo) tikimybę, kai FAR (blogo atpažinimo) tikimybė prilyginta nuliui. Su naujais duomenimis pagerėjimas yra 0.27%. T.y. 10000 lyginimo operacijoms 27-iais atvejais daugiau asmenį atpažinsime teisingai.

Kaip matome, didelio atpažinimo kokybės pagerėjimo nepasiekiame. Tai įvyksta todėl, kad, iš esmės, kreivių kreivumai, esantys arti ypatingų taškų, stipriai nesiskiria. O kai ieškoma per visą kreivę, nepavyksta pakankamai stabiliai nustatyti didžiausio kreivumo koordinačių. Dėl skenavimo subtilybių bei binarizavimo algoritmo galimas vientisos kreivės atpažinimas kaip dvejų atskirų kreivių.



Pav. 17. a) kreivė atpažinta kaip vientisa; b) kaip dvi atskiros kreivės. Pilki rutuliai nurodo vietas, kuriuose bus pasiekiamas didžiausias kreivumas. Atkreipiu dėmesį, kad kreivei suskilus i kelias kreives, atsiranda papildomi netikrieji ypatingi taškai.

Akivaizdu, kad įvykus tokiai situacijai, nauji duomenys jokios naudos neduoda, arba pablogina atpažinimą.

Taip pat rezultatai priklauso nuo kreivės atkarpos ilgio pasirinkimo. Kaip matyti trečiam priede, pasirinkus mažesnį atkarpos ilgį, rezultatai yra blogesni. Šis parametras priklausomai nuo vaizdo rezoliucijos bus kintamas.

Panagrinėkime ką mums duoda kreivės kreivumo krypties nustatymas. Geriausią atpažinimo kokybės pagerėjimą duoda nustatytas kreivės kreivumas tam tikroje ypatingo taško aplinkoje R ir tik nutrūkimo taškams. Bifurkacijos atveju, kadangi jai priklauso trys kreivės, yra didesnė tikimybė, kad skenavimo arba vaizdo binarizavimo metu, kreivės bus labiau iškraipytos arba kai kurios kreivės bus atpažintos kaip trūkusios. Lygiai tas pats gali atsitikti ir nutrūkimo taškams, tačiau jie apibrėžti tik viena kreive. Taigi tikimybė, kad bifurkacijos taškas kartu su jam priklausančiomis kreivėmis kiekvienu atveju duos skirtingus parametrus, yra 3 kartus didesnė nei nutrūkimo taškui. Kiekvieną ypatingą tašką, apibrėžus per jam priklausančios kreivės kreivumo kryptį, galima skirstyti dar į kelis tipus. Nutrūkimo taškas bus dvejų rūšių: su teigiamu kreivumu, bei neigiamu. Bifurkacijos atveju, tai yra sudėtingiau. Reikia apsibrėžti, kurios kreivės kreivumą naudoti, arba visų trijų kreivių kreivumų derinį.

Ypatingų taškų išskirstymas pagal naujus tipus leis greičiau atlikti atpažinimo algoritmą. Taškus laikyti skirtingais vien tik pagal kreivumo ženklą negalima. Reikia įsivesti tam tikrą kreivumo slenkstį, kurį peržengus būtų galima laikyti du taškus visiškai skirtingais. Tai reikia dėl to, kad kai kreivės yra artimos tiesėms, dėl skenavimo niuansų, ta pati kreivė vienu atveju gali duoti vienokį kreivumą, kitu atveju kitokį. Jei kreivumo įvertinimas yra pakankamai ryškus, galima teigti, kad kreivumas apibrėžtas vienareikšmiškai.

Lyginimo algoritme turint du taškus su griežtai įvertintais kreivumais, galima iš karto spręsti, ar tai potencialiai geri taškai, ar ne. Jei kreivumai skiriasi ženklu, automatiškai nėra prasmės toliau vykdyti lyginimo operacijas šiems taškams. Jei kreivumai vieno ženklo, pereinama prie gilesnės taškų analizės.

Kad matyti naujo požymio pasiskirstymą, praktiniu būdu paskaičiuota kiek kreivių turi teigiamą arba neigiamą kreivumą. Iš viso išnagrinėta 1400 piršto antspaudų. Kreivumo skaičiavimo metu išnagrinėtos 65851 kreivės. Neigiamo kreivumo kreivių 36188, teigiamo

29663, t.y. ~55% ir ~45% atitinkamai. Taigi, pasinaudojus kreivių kreivumų teikiama informacija, galima sutaupyti arti ~45% gilesnės ypatingųjų taškų analizės laiko.

Plačiau apie piršto antspaudų lyginimo algoritmą galima paskaityti moksliniame straipsnyje **[KKK06]**.

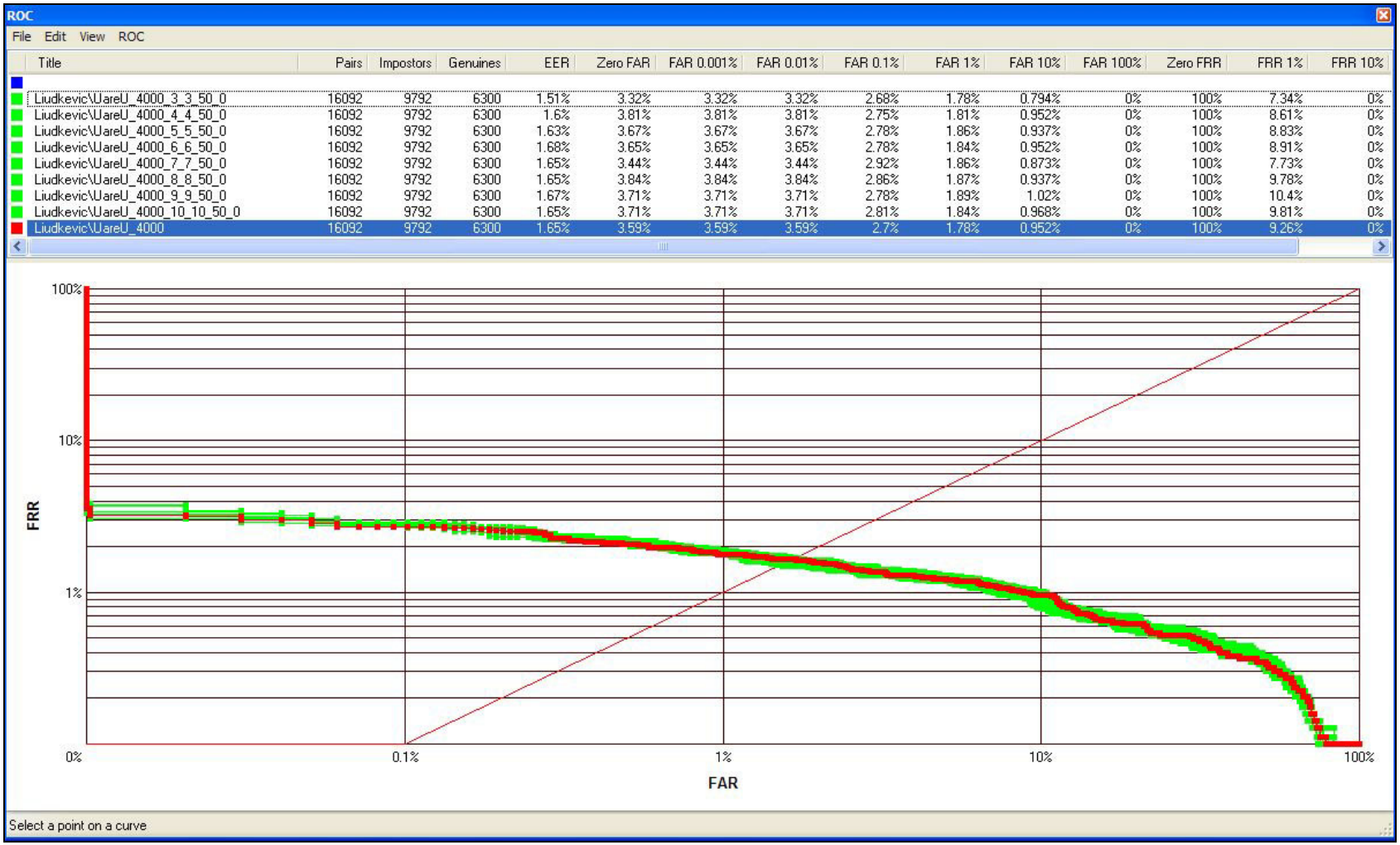
Išvados ir rekomendacijos

Atlikus praktinius skaičiavimus ir rezultatų analizę padarytos išvados :

- piršto antspaudo atpažinimo algoritmo kokybės pagerėjimas – pasiektas, tačiau jis nėra didelis;
- kreivės išlinkimo krypties arti ypatingojo taško įvertinimas įgalina pagreitinti atpažinimo algoritmą.

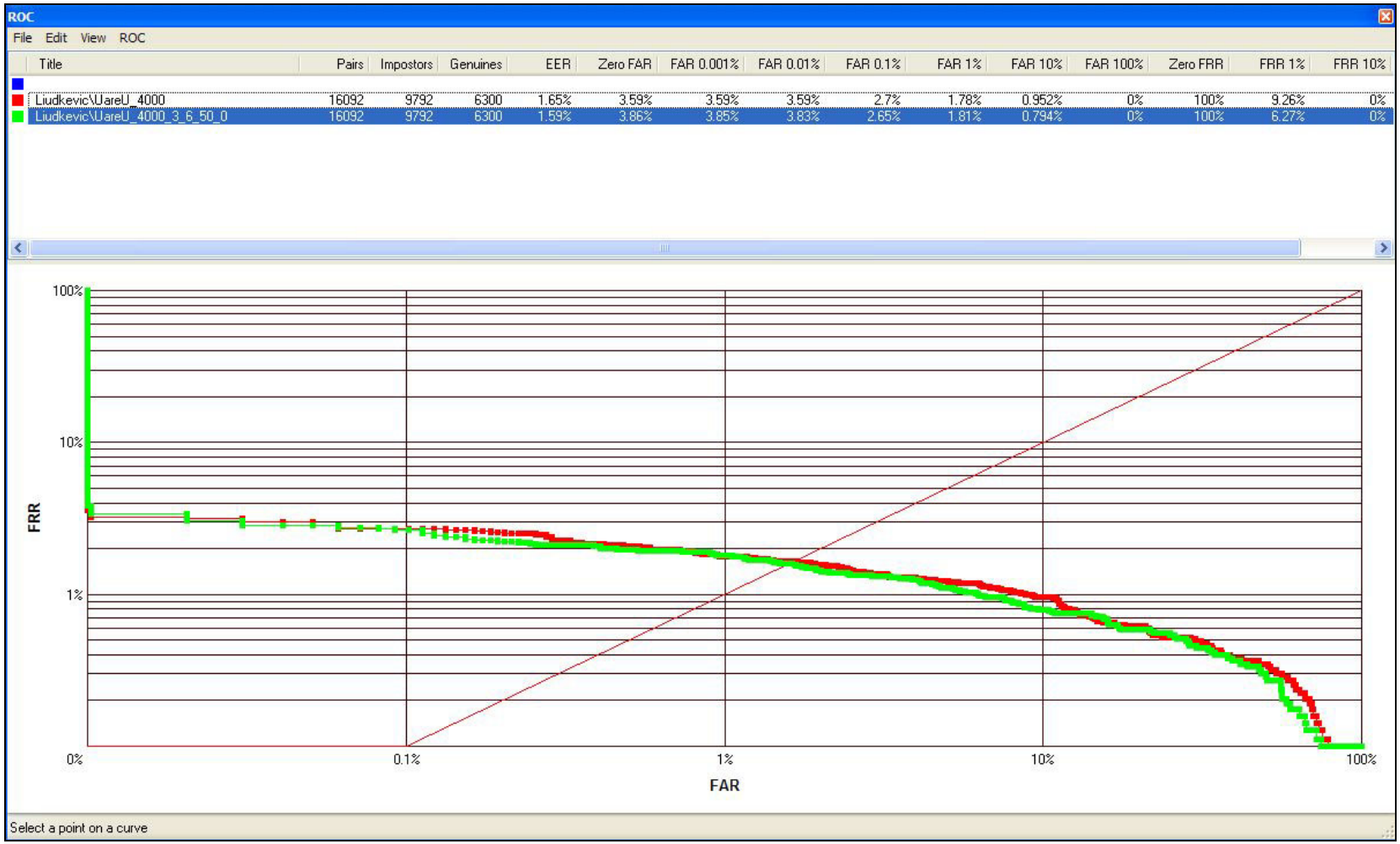
Atpažinimo kokybės pagerėjimas nėra labai ryškus, dėl to, šia linkme ieškoti sprendimų nėra labai tikslinga. Didesnis efektyvumas bus pasiektas, nagrinėjant atpažinimo algoritmo paspartinimą antros išvados pagalba.

Priedas 1



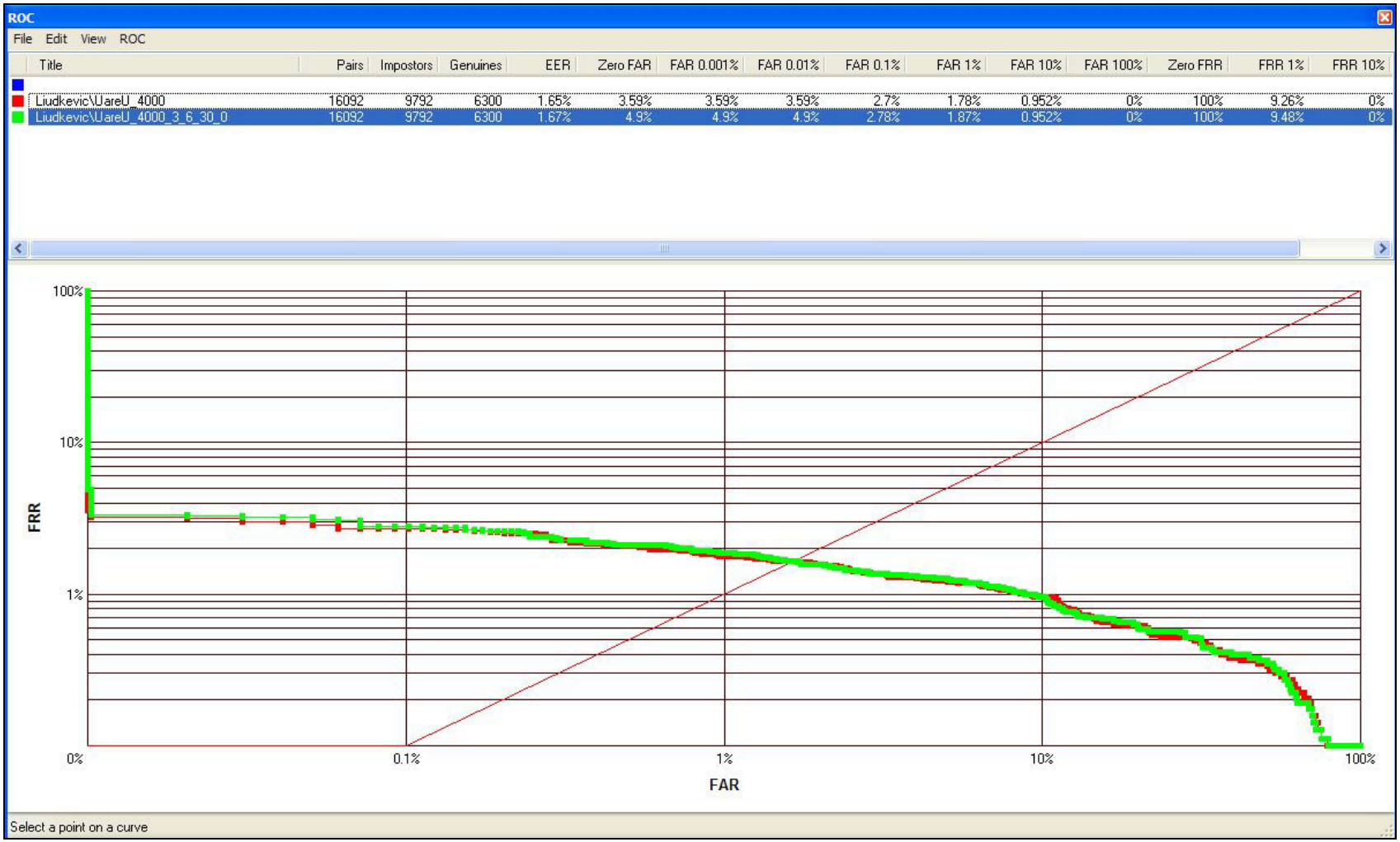
ROC kreivės su kiekvienu duomenų rinkiniu

Priedas 2



ROC kreivės su 1-o ir 4-o tipo duomenų rinkiniais

Priedas 3



ROC kreivės, kai paimtas mažesnis kreivės atkarpos ilgis

Literatūros sąrašas

[Скв02] „Обзор алгоритмов построения триангуляции Делоне“. Скворцов А. В. 2002 г.

[Sib73] R.Sibson. „Locally equiangular triangulations“. The Computer Journal Vol.2 (3): 243-245, 1973

[Rom] Cuk Roman. „Construction of Voronoi diagrams using Fortune's method : A look on an Implementation“, http://www.mif.vu.lt/~bastys/academic/ATE/delaunay/Fortune/FortuneDemo.htm

[PAA99+] Pankaj K. Agarwal, Boris Aronov, Sariel Har-Peled, Micha Sharir. „Approximation and Exact Algorithms for Minimum-Width Annuli and Shells“ (1999), http://citeseer.ist.psu.edu/agarwal99approximation.html

[Ско98] Скворцов А.В., Костюк Ю.Л. „Эффективные алгоритмы построения триангуляции Делоне“. Вып. 1. Томск: Изд-во Томского ун-та, 1998.

[Иль85] Ильман В. М. „Экстремальные свойства триангуляции Делоне“. Вып. 10 (88). М., 1985.

[Lee78] Lee D. „Proximity and reachability in the plane“. Techn. Report R-831. Coordinated Sci. Lab., Univ. of Illinois at Urbana. Urbana, 1978.

[MMJ03+] Davide Maltoni, Dario Maio, Anil K. Jain, Salil Prabhakar, „Handbook of Fingerprint Recognition”, 2003

[KKK06] Moklsinis straipsnis: Andrej Kisel, Alexej Kochetkov, and Justas Kranauskas, “Fingerprint Minutiae Matching Without Global Alignment Using Local Structures”, 2005

Internetinis žurnalas: Новые вычислительные технологии. http://num-meth.srcc.msu.su