VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Rainelės atpažinimas

Iris Recognition

Magistro baigiamasis darbas

Atliko: Jurgita Misulaitytė

Darbo vadovas: Doc. dr. Algirdas Bastys

Recenzentas: Lekt. Irmantas Naujikas

Vilnius – 2009

Padėkos organizacijoms

Dėkoju UAB „Neurotechnologijai“ už duotas ir magistro baigiamajame darbe leistas naudoti rainelių paveiksliukų duomenų bazes, informacijos apie paveiksliukus rinkmenas ir ROC kreivių peržiūros programinį įrankį.

Santrauka

Šiame magistro baigiamajame darbe pateiktas darbo atlikimo planas, literatūros šaltinių apžvalga, aprašyti mano sukurti du rainelių segmentacijos metodai, teoriniai principai, kuriais jie remiasi ir programinis įrankis, kuris juos įgyvendina, su šiuo programiniu įrankiu atlikti tyrimai ir jų rezultatai.

Vienas iš mano sukurtų rainelių segmentacijos metodų remiasi žymėmis kontroliuojama vandens takoskyros transformacija ir momentiniais invariantais. Derinant vandens takoskyros transformaciją su momentiniais invariantais rainelių segmentacijoje išvengiama slenksčių ar kitokių iš anksto nustatytų apribojimų paveiksliukams. Atlikti tyrimai parodė, kad algoritmą įgyvendinantis programinis įrankis vyzdį suranda 73% paveiksliukų, o rainelę 67% paveiksliukų.

Kitas mano sukurtas metodas – aktyvaus kontūro paieška. Jis pagal duoto pradinio rainelės apskritimo matmenis suranda tikslų arba bent tikslesnį nei apskritimas rainelės išorinį kraštą. Šis metodas gerai (neprarandant daug rainelės duomenų) segmentuoja 93% rainelės paveiksliukų ir 1,6 karto sumažina rainelių atpažinimo klaidų įverčius.

Raktiniai žodžiai: žymėmis kontroliuojama vandens takoskyros transformacija, momentai, momentiniai invariantai, vyzdžio ir rainelės segmentacija.

Summary

In this report of my postgraduate work you can find an execution plan, a literary source review, a description of two iris segmentation methods created by me, theoretical principles my methods relies on, a description of application which realizes the methods and results of a research performed with the application.

One of the iris segmentation methods created by me relies on the marker-controlled watershed transformation and moment invariants. Combining the marker-controlled watershed transformation with moment invariants in iris segmentation avoids thresholds and other pre-determined limitations for pictures. Performed research revealed that the algorithm realizing application finds pupil in 73% of all examined pictures and iris in 67% of the pictures.

The second method created by me – active contour detection. By given initial iris circle dimensions it detects precise or at least more accurate than the circle external iris contour. The method gives good segmentation results (not loosing much iris data) for 93% iris images and 1,6 times decreases iris recognition error rates.

Keywords: marker-controlled watershed transformation, moments, moment invariants, pupil and iris segmentation.

Turinys

1. Įvadas

Rainelė yra turtingos tekstūros žiedo formos matoma akies dalis tarp vyzdžio ir stiklakūnio. Ji turi daug unikalių požymių, kurie sėkmingai taikomi identifikuoti žmogų. Rainelė formuojasi per pirmus 18 mėnesių nuo žmogaus gimimo ir beveik nesikeičia per visą gyvenimą (pasikeitimai dažniausiai yra susiję su ligomis). Rainelę padirbti ar užmaskuoti yra daug sunkiau negu pirštų atspaudus arba veidą. Šie ypatumai daro atpažinimą pagal rainelę vienu iš patikimiausių žmogaus identifikavimo būdų [1 lentelė].

1936 m. oftalmologas (akių ligų specialistas) Frank Burch suformulavo mintį apie akių rainelių panaudojimą asmens identifikavimui. Pereito amžiaus 8-ajame dešimtmetyje akies rainelės panaudojimo identifikacijai idėja pasirodė James Bond seriale, tačiau praktiškai ji dar nebuvo realizuota. 1987 m. du oftalmologai Aran Safir ir Leonard Flom užpatentavo šią idėją ir 1989 m. jie paprašė John Daugman (Harvardo universiteto lektoriaus) sukurti tinkantį kompiuteriui akių rainelių atpažinimo algoritmą. Daugmano pradinis algoritmas aprašytas [Dau93] straipsnyje. Šiuolaikinės komercinės rainelių atpažinimo sistemos naudoja įvairias originalaus Daugman algoritmo modifikacijas, tačiau principinė atpažinimo schema nesikeičia. Teises į patentus išpirko Iridian Technologies [ITe] kompanija ir sėkmingai vysto asmenų identifikavimo pagal akies rainelę biznį. [Bas]

1. 1. Rainelės palyginimas su kitomis biometrijomis

Žmogaus biometrinė charakteristika - tai išmatuojama jo fiziologinė charakteristika (pvz. veidas, akies rainelė, piršto atspaudas) arba elgesio (pvz., balso, parašo) ypatybės.

Ideali charakteristika turi pasižymėti tokiomis savybėmis:

- *Universalumas* – savybė, kuri leidžia apibūdinti žmogų viena biometrine charakteristika.
- *Unikalumas* – savybė, kuri nusako, ar skirtingi individai gali turėti vienodą arba labai panašią biometrinę charakteristiką.
- *Pastovumas* – savybė, kuri reikalauja, kad charakteristika laikui bėgant visiškai nekistų arba kistų tik labai nežymiai.
- *Požymių surenkamumas* – savybė, įgalinanti greitai bei nesunkiai išskirti, gauti ir išmatuoti tam tikrą požymį ar požymių rinkinį, charakterizuojantį žmogų.
- Nesunku surinkti charakteristikos požymius.

Lentelėje [1 lentelė] pateiktas dažniausiai naudojamų žmogaus biometrinių charakteristikų palyginimas.

1 lentelė. Biometrinių charakteristikų palyginimas. [Wikb]

Charakteristika	**Universalumas**	**Unikalumas**	**Pastovumas**	**Požymių surenkamumas**
Veido atvaizdas	+++	+	++	+++
Veido termograma	+++	+++	+	+++
Piršto atspaudas	++	+++	+++	++
Rankos geometrija	++	++	++	+++
Akies rainelė	+++	+++	+++	++
Akies tinklainė	+++	+++	++	+
Parašas	+	+	+	+++
Balsas	++	+	+	++
Lūpų atspaudas	+++	+++	++	+
Ausies ypatumai	++	++	++	++
Rašto dinamika	+++	+++	+	+++
Eisena	+++	++	+	+

1. 2. Praktiniai taikymai

Biometrinės atpažinimo technologijos gali automatiškai identifikuoti žmogų, naudodamos jo fiziologines arba elgesio charakteristikas. Kadangi šias charakteristikas, priešingai nei daugelį kitų informacijos saugos priemonių, pamesti arba suklastoti yra gana sunku, biometriniai atpažinimo metodai darosi vis populiaresni ir labiau paklausūs komerciniame bei valstybiniame sektoriuose. Vis didesnį populiarumą įgyja rainelės atpažinimas, nes rainelė yra vienas iš geriausių žmogaus identifikatorių, lyginant su kitomis biometrinėmis charakteristikomis (tai galite matyti iš ankstesnio skyrelio lentelės [1 lentelė]).

Pagrindinės rainelės atpažinimo technologijų taikymo sritys: automatizuota skrydžių saugumo kontrolė, imigracijos kontrolė, duomenų bazių ir kompiuterinių sistemų naudotojų identifikavimas, prisijungimo teisių kontrolė, medicina. Planuojama sukurti biometrines asmenų identifikavimo korteles [QAI]. [Dau93]

Tarptautinė civilių skrydžių organizacija [ICAO] akies raineles, kartu su pirštų atspaudais ir veidais, pripažino ir standartizavo ateities pasų panaudojime. Nuo 2001 m. asmenų identifikavimas pagal raineles taikomas Olandijos Schiphol oro uosto imigracijos tarnyboje [Sch Iris], [Sch]. Tais pačiais metais rainelės atpažinimo procedūra pradėta naudoti Jungtinių Arabų Emyratų visuose 17 oro, sausumos ir jūros pasienio kontrolės punktuose [Dau04]. Anglijoje nuo 2006 m. funkcionuoja pasienio kontrolės programa IRIS – „Iris Recognition Immigration System“ [Bas], [BTB], [UK Iris]. Atėnų, Frankfurto ir keletas Kanados oro uostų taipogi naudoja rainelės atpažinimą, kontroliuojant imigraciją ir tikrinant keleivius.

Rainelės atpažinimo panaudojimas numatomas ir kitose srityse, kuriose dabar asmuo identifikuojamas, pateikus asmens dokumentą, kortelę, PIN kodą, slaptažodį ar bet kokią kitą

identifikavimo priemonę. Tokios sritys yra elektroninė komercija, teismo medicinos ir policijos sritys, įėjimas į pastatą, automobilio užvedimas, informacijos saugumo užtikrinimas, teisių nustatymas, prisijungimas prie tinklo bei kompiuterinių programų ir kitos sritys, kuriose reikalingas asmens identifikavimas. [Dau93]

Dar viena rainelės atpažinimo panaudojimo sritis – medicina. Dauguma rainelės pokyčių yra susiję su žmogaus sveikata. Dėl ligų gali pasikeisti rainelės spalva, atsirasti pigmentinių dėmių, deformuotis vyzdys, pasikeisti rainelės kraštas ir t. t.. Šie rainelės pokyčiai naudojami ligoms diagnozuoti ar nustatyti rizikos faktorių susirgti tam tikra liga. [Wikb] Taip pat rainelės atpažinimas taikomas identifikuoti kūdikius gimdymo namuose.

1. 3. Darbo tikslas ir planas

Darbo tikslas – sukurti rainelės segmentacijos algoritmą ir jį įgyvendinantį programinį įrankį.

Šio tikslo siekiau pagal tokį darbo planą:

1. išanalizuoti literatūros šaltinius apie egzistuojančius rainelės atpažinimo metodus, didžiausią dėmesį skiriant rainelės segmentacijai;
2. sukurti rainelės segmentacijos metodą ir algoritmą;
3. sukurti programinį įrankį, įgyvendinantį mano sukurtą algoritmą;
4. atlikti tyrimus su akių paveikslėliais algoritmo efektyvumui įvertinti.

1. 4. Rainelės segmentacija

Yra paskelbta nemažai straipsnių, kuriuose aprašytas rainelių atpažinimo algoritmas. Jį sudaro tokie pagrindiniai žingsniai:

1. akies rainelės duomenų skenavimas,
2. rainelės skaitmeninio vaizdo pirminis apdorojimas,
3. rainelės lokalizavimas (segmentacija),
4. vyzdžio ir rainelės kraštų nustatymas,
5. rainelės normalizavimas,
6. binarinių požymių išskyrimas, šablonų sukūrimas,
7. šablonų palyginimas, sprendimo priėmimas.

Plačiausiai šiame darbe mano nagrinėtas rainelių atpažinimo etapas – rainelės segmentacija. Šiame etape susiduriama su problemomis, kurios apsunkina rainelės srities paiešką akies paveikslėlyje:

- akis paveikslėlyje gali būti truputį pasisukusi į kurią nors pusę;
- blakstienos dažnai patenka į kadrą ir uždengia rainelės dalį;

- vokai gali dengti rainelės dalį;
- gali būti lęšių ir atspindžių akies vyzdyje;
- akys gali būti žvairos.

Šiame darbe, antrajame skyriuje, aprašiau kiekvieno iš rainelės atpažinimo žingsnių atlikimo metodus, išskyrus skenavimą ir vaizdo pirminį apdorojimą, kuris mano magistriniame darbe nebus nagrinėjamas, remdamasi literatūros sąraše išvardintais straipsniais. Trečiajame šio magistro baigiamojo darbo skyriuje aprašyti mano atlikti rainelės segmentacijos praktiniai darbai remiasi kai kuriais literatūros apžvalgos skyriuje [2.] aprašytais metodais ([2. 1. 11.] ir [2. 1. 12.]).

Mano sukurtas rainelių segmentacijos algoritmas remiasi žymėmis kontroliuojama vandens takoskyros transformacija (naudojama segmentacijai) ir momentiniais invariantais (naudojami išskirtoms sritims, konkrečiai – vyzdžiui, atpažinti). Ši metodika yra unikali, kadangi literatūroje vandens takoskyros transformacijos panaudojimas rainelėms segmentuoti nėra aprašytas.

Žymės – tai taškai akies paveikslėlyje, nurodantys, kur yra vyzdys, rainelė ir fonas, kuriais remiasi transformacija. Programoje, kuri realizuoja mano algoritmą, šiuos taškus gali sužymėti programos naudotojas paveikslėlyje programos lange arba jie parenkami automatiškai. Programa akies paveikslėlį paverčia nespalvotu (jei paduodamas spalvotas paveikslėlis), iš jo paruošia gradientinį paveikslėlį ir suranda bei skirtingomis spalvomis nuspalvina tris sritis: vyzdį, rainelę ir foną.

Aktyvaus kontūro paieška pagal duotą pradinį rainelės apskritimą suranda tikslesnį rainelės kontūrą. Toks kontūro patikslinimas sumažina rainelių atpažinimo klaidų tikimybę. Šis metodas taip pat yra unikalus ir neaprašytas literatūroje.

2. Literatūros apžvalga

2. 1. Rainelės radimas paveikslėlyje (lokalizavimas)

2. 1. 1. „Nuo apytikslio iki tikslaus“ strategija

Rainelės lokalizavimą sudaro du žingsniai: rainelės vidinio krašto (vyzdžio krašto) lokalizavimas ir išorinio rainelės krašto suradimas. Straipsnio „Statistical Pattern Recognition of the Iris“ [GKN03] autoriai siūlo prieš tai dar akies paveikslėliui atlikti pirminį apdorojimą, normalizuojant iki standartinio ryškumo ir pritaikyti slenksčius bei Sobel gradientą, tačiau plačiau apie tokį normalizavimą nerašo.

John Daugman savo straipsniuose [Dau93] ir [Daua] aprašo metodą, kuris tiksliai lokalizuoja vidines ir išorines rainelės ribas. Šis metodas remiasi „nuo apytikslio iki tikslaus“ strategija, kuri naudoja vieno vaizdo elemento (angl. pixel) tikslumu įvertintas centro

koordinates ir spindulį tiek rainelei, tiek vyzdžiui lokalizuoti. Vyzdžio centras dažnai nesutampa su rainelės centru, o vyzdžio spindulys gali būti nuo 0,1 iki 0,8 rainelės spindulio. Taigi visi trys parametrai, nusakantys vyzdžio apskritimą, turi būti įvertinti atskirai nuo rainelės parametrų.

$$\max_{(r,x_0,y_0)} \left| G_\sigma(r) * \frac{\partial}{\partial r} \int_{r,x_0,y_0} \frac{I(x,y)}{2\pi r} ds \right| \quad (1)$$

I(x,y) – paveiksliukas su akimi.

(x,y) – paveiksliuko sritis, kurioje ieškosime.

r – spindulys.

(x_0, y_0) - centro koordinatės.

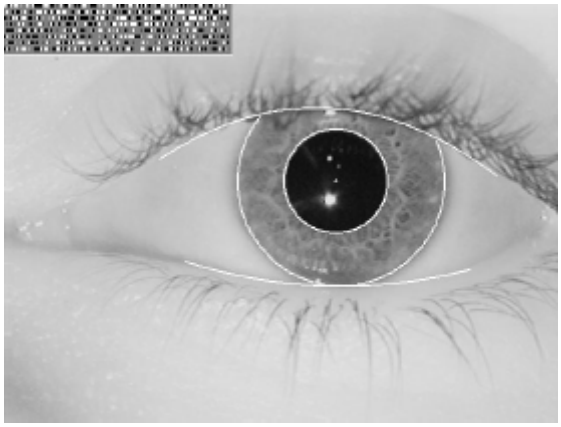
$G_\sigma(r)$ - glodinimo (triukšmo sušvelninimo) funkcija, tokia kaip Gauso, dydžio σ. Gauso funkcija nusakoma tokia formule: $G_\sigma(r) = \sqrt{2\pi}^{-1} e^{\frac{-r^2}{2}}$ [CIS].

Operatorius (1) ieško paveiksliuko srityje (x,y) maksimumo tiriamame dalinės išvestinės darinyje, atsižvelgdamas į didėjantį spindulį r, normalizuoto kontūro integralo paveiksliuke I(x,y) išilgai apskrito lanko ds su spinduliu r ir centro koordinatėmis (x_0, y_0). Simbolis * išreiškia operatoriaus ėjimą vingiu, naudojant glodinimą. Kitaip sakant, šis operatorius elgiasi kaip apskritimo krašto ieškiklis su nuokrypiu σ, kuris iteratyviškai ieško maksimalios kontūro integralo dalinės išvestinės, augant spinduliui, trimis parametrais (centro koordinatėmis ir spinduliu (x_0, y_0, r)) nusakytame plote. Tie trys parametrai ir nusako kontūro integravimo kelią. (Straipsnyje [GKN03] pateikiamas toks pats operatorius tik be glodinimo funkcijos, taigi John Daugman aprašytas operatorius yra geresnis.)

Operatorius (1) leidžia surasti tiek vyzdžio ribas, tiek tolimesnes rainelės ribas. Reikia paminėti, kad pirminė rainelės krašto paieška taip pat apima ir vyzdžio požymių išryškinimą, kadangi kraštas dažnai būna labai neryškaus kontrasto, kai naudojamos didelio ilgio infraraudonosios bangos rainelei skenuoti.

Kai „nuo apytikslio iki tikslaus“ strategijos iteratyviškos paieškos abejiems rainelės kraštams pasiekia vieno vaizdo elemento tikslumą, tuomet naudojamas panašus krašto kreivės radimo metodas lokalizuoti aukštesnįjį ir žemesnįjį akies voko kraštus. Kontūro integravimo kelias anksčiau aprašytame operatoriuje yra keičiamas iš apskritiminio į lankišką, o parametrai keičiami į tokius, kurie standartiniuose statistiniuose įvertinimo metoduose taikomi optimaliai apibūdinti kiekvieno akies voko krašto požymius. Kokie tai parametrai John Daugman savo straipsnyje nepasako.

Visų šiame skyrelyje aprašytų lokalizavimo operacijų rezultatas – iš kitų paveiksliuko sričių izoliuota rainelė. Paveikslėlyje [1 pav.] ji pavaizduota baltomis linijomis.



1 pav. Rainelės pavyzdys, monochromatiškai nuskenuotas 35 cm atstumu. Baltos linijos rodo rainelės ir vyzdžio lokalizavimo ir akies voko nustatymo žingsnių rezultatus. Bitų srautas paveiksliuko viršuje kairėje yra demoduliacijos su kompleksinėmis 2D Gabor bangelėmis, koduojančiomis fazinę rainelės seką, rezultatas. [Daua]

2. 1. 2. Apskritiminės Hough transformacijos ir koncentrinių apskritimų metodas

Straipsnio „Iris Recognition using Corner Detection“ autoriai siūlo kiek kitokį (daug paprastesnį) lokalizavimo metodą [GMR+]. Vidinis vyzdžio kraštas (angl. boudary) lokalizuojamas, naudojant Apskritiminę Hough transformaciją (angl. Circular Hough Transformation), kadangi ši technologija neblogai veikia ne tik tokiuose paveikslėliuose, kur vyzdys yra nesunkiai randamas, bet ir tokiu atveju, kai paveikslėliuose yra šešėlių, triukšmo ar kitų neigiamų faktorių, kurie apsunkina vyzdžio lokalizavimą. Išorinis rainelės kraštas nustatomas žiediniu intensyvumų sumavimu pagal vyzdžio centrą ir spindulį.

2. 1. 2. 1. Vyzdžio suradimas

Paveikslėlis konvertuojamas į juodai baltą, kad panaikintume apšvietimo efektą. Kadangi vyzdys yra didžiausia juoda sritis intensyvumų paveikslėlyje, ji gali būti nesunkiai nustatyta iš binarizuoto paveikslėlio, naudojant tinkamą slenkstį. Tačiau problema iškyla, jei žmogus turi tamsią rainelę. Tokiu atveju, vyzdžio lokalizacija nepavyksta. Norint išspręsti šias problemas, vyzdžio aptikimui gali būti naudojama Apskritiminė Hough transformacija. Pagrindinė šios technologijos idėja yra surasti kreives, kurios gali būti parametrizuotos kaip tiesios linijos, polinomai, apskritimai ir t.t. (vyzdžio paieškos atveju mums bus reikalingi apskritimai) tam tikroje tinkamoje parametrų erdvėje. Ši technologija gerai veikia netgi esant įvairiems neigiamiems dalykams, apsunkinantiems lokalizavimą (šešėliai, triukšmas ir pan.).

Bendrą Apskritiminės Hough transformacijos idėją galima rasti straipsnyje [CIS], kuriame pateikta tokia formulė:

$$(x-c)^2 + (y-d)^2 = r^2, \qquad (2)$$

kur (x, y) yra vaizdo elemento paveikslėlyje koordinatės, c ir d nusako apskritimo centrą, o r nusako jo spindulį. Pagal formulę (2) randami visi apskritimai erdvėje (x, y) ir parenkama ta erdvės sritis (erdvės sritį nusako parametrų c, d, ir r intervalas), kur yra didžiausia sankaupa tų surastų apskritimų.

Straipsnio [GMR+] autorių aprašyta vyzdžio suradimo procedūra pirmiausiai randa paveikslėlio intensyvumų gradientą visose duoto paveikslėlio vietose, naudodama sobel filtrus. Gradientiniai paveikslėliai ($G_{vertical}$ ir $G_{Horizontal}$) x ir y kryptimis, gaunami pagal branduolius, kurie nustato horizontalius ir vertikalius pokyčius paveikslėlyje. Sobel filtro branduoliai yra tokie:

$C_{vertical}$={-1 -2 -1; 0 0 0;1 2 1}, $C_{horizontal}$={-1 0 1;-2 0 2;-1 0 1}. (3)

Absoliuti reikšmė gradientinių paveikslėlių vertikalia ir horizontalia kryptimis yra gaunama suformuoti absoliutinį gradientinį paveikslėlį, naudojant tokią lygtį:

$$G_{abs} = G_{Vertical} + G_{Horizontal}\,. \qquad (4)$$

Čia $G_{vertical}$ yra paveikslėlis nufiltruotas su $C_{vertical}$, o $G_{horizontal}$ yra paveikslėlis nufiltruotas su $C_{horizontal}$.

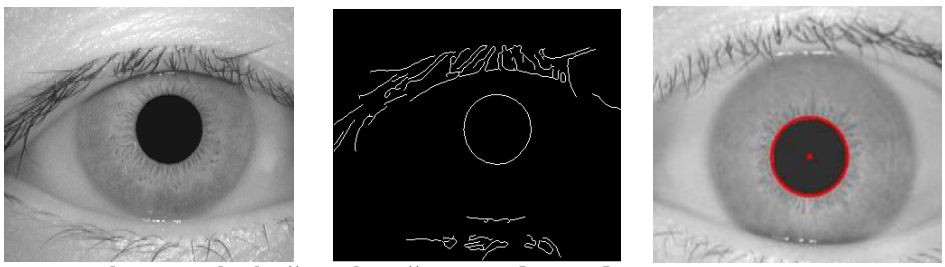
Absoliutinis gradientinis paveikslėlis yra naudojamas rasti kampus, naudojant Canny pasiūlytą metodą. Apie Canny metodą straipsnyje [GMR+] nieko nerašoma. Norint apie jį sužinoti, reikalinga kitų straipsnių paieška ir analizė.

Canny metodu gautas kampų paveikslėlis yra nuskenuojamas, ieškant vaizdo elementų (angl. pixel) (P), turinčių „true" reikšmę ir centras nustatomas pagal šias lygtis:

$$\begin{aligned} xc &= x - r * \cos(\theta), \\ yc &= y - r * \sin(\theta). \end{aligned} \qquad (5)$$

Čia x ir y yra vaizdo elemento P koordinatės, r yra galimas diapazonas spindulio reikšmių, θ kinta intervale [0:π].

Konkrečiai r reikšmei, skaičiuojamos bei išsaugomos xc ir yc reikšmės. Kiekvieną kartą, kai xc ir yc reikšmės tenkina paveikslėlio dydžio kriterijų, tam tikro skaitliuko vertė padidinama vienetu. Maksimali skaitliuko vertė duos vyzdžio centrą ir spindulį, kaip pavaizduota paveikslėlyje [2 pav.].



2 pav. Vidinio rainelės krašto radimo žingsniai. [GMR+]

2. 1. 2. 2. Išorinio rainelės krašto lokalizavimas

Pašalinti triukšmui, išsaugant rainelės kraštus, naudojamas medianinis filtras. Po filtravimo paveikslėlio kontrastas padidinamas, paryškinant paveikslėlio kraštus, naudojant histograminį išlyginimą, kaip parodyta paveikslėlyje [3 pav. (a)]. Šis padidinto kontrasto paveikslėlis naudojamas surasti išorinį rainelės kraštą, piešiant koncentrinius apskritimus, kaip pavaizduota paveikslėlyje [3 pav. (b)]. Jie turi skirtingo dydžio spindulius, einančius iš vyzdžio centro. Intensyvumai, išsidėstę per apskritimo perimetrą yra sudedami. Iš visų rainelės apskritimo kandidatų išrenkamas apskritimas, turintis didžiausią intensyvumo pokytį, atsižvelgiant į anksčiau nupieštą apskritimą, yra išorinis rainelės kraštas. Paveikslėlis [3 pav. (c)] vaizduoja rainelės lokalizavimo pavyzdį.

3 pav. (a) Padidinto kontrasto paveikslėlis
(b) Koncentriniai skirtingo spindulio apskritimai
(c) Lokalizuotos rainelės paveikslėlis [GMR+]

2. 1. 3. Rainelės segmentacija ir Fuzzy K-means klasterizacija

H. Proenca ir L. A. Alexandre savo straipsnyje [PA06] siūlo tokią rainelės segmentacijos metodologiją: pirma išrenkami originalaus akies paveikslėlio vaizdo elementų požymiai, tuomet atliekama klasterizacija, po kurios gaunamas normalizuotas akies paveikslėlis, randami kampai Canny metodu (šis metodas taip pat buvo naudotas skyrelyje [2. 1. 2. 1.]), pritaikoma apskritiminė Hough transformacija (taip pat naudota [2. 1. 2. 1.]), po kurios gaunama segmentuota rainelė. Straipsnio [PA06] autoriai teigia, jog klasterizacija pagerina tolesnį kampų išskyrimą, o tai savo ruožtu duoda didesnį apskritiminės Hough transformacijos tikslumą. Ištyrę įvairius vaizdo elementų požymius, jie priėjo išvadą, kad geriausiai kiekvieną vaizdo paveikslėlį charakterizuoja trys diskretūs komponentai {x, y, z}, kur (x, y) yra vaizdo elemento pozicijos paveikslėlyje koordinatės, o z yra to vaizdo elemento intensyvumas. Taigi šiuos požymius straipsnio autoriai rekomenduoja naudoti rainelės segmentacijai. Jie taipogi išnagrinėjo klasterizaciją (klasifikaciją), kuomet į vienai klasei turi būti priskirti visi vaizdo elementai, priklausantys rainelei, o kitai klasei – visi likę vaizdo elementai. Tyrimui pasirinko keturis algoritmus: Kohen žemėlapius (angl. Kohen‘s self-organising maps), K-means, Fuzzy K-means

ir tikimybių maksimizavimo (angl. expectation maximization). Geriausi klasterizacijos rezultatai buvo pasiekti su Fuzzy K-means algoritmu.

Fuzzy K-means algoritmo esmė yra tokia: kiekvieno duomenų taško priklausomybė kuriam nors iš k klasterių yra neapibrėžta. Duomenų taškai priklauso klasteriams su tikimybėmis $\hat{P}(w_i \mid x_j, \hat{\theta})$, kur $\hat{\theta}$ yra parametrų vektorius priklausomybių funkcijoms. Šis klasterizacijos algoritmas siekia minimizuoti heuristinę globalinę kainų funkciją J_{fuz}:

$$J_{fuz} = \sum_{i=1}^{c}\left(\sum_{j=1}^{n}\left(\hat{P}(w_i \mid x_j, \hat{\theta})^b \left\|x_j - \mu_i\right\|^2\right)\right), \quad (6)$$

kur b yra parametras, reguliuojantis skirtingų klasterių persidengimą. Trumpai sakant, kiekvienam pateiktam duomenų rinkiniui visi klasteriai turi sureguliuoti savo svorius, atsižvelgdami į atstumą tarp pateiktų duomenų ir klasterių svorių, kurie yra tikimybės, kad pateikti duomenys priklauso kiekvienam klasteriui.

2. 1. 4. Rainelės ir vyzdžio, kaip koncentrinių apskritimų, segmentacija

2. 1. 4. 1. Greitas segmentacijos algoritmas

Straipsnio [CIS] autoriai aprašo vyzdžio ir rainelės segmentacijos algoritmą, kuris remiasi prielaida, jog vyzdžio ir rainelės kraštai yra koncentriniai apskritimai, nors dažniausiai iš tiesų taip nebūna. Ši prielaida leidžia supaprastinti ir pagreitinti rainelės krašto paiešką. Straipsnio autoriai pateikia algoritmą su tokiais žingsniais:

1. Vyzdžio centro (X_p, Y_p) paieška pagal formules:

   $$X_p = \arg\min_x\left(\sum_y I(x,y)\right), \quad (7)$$

   $$Y_p = \arg\min_y\left(\sum_x I(x,y)\right), \quad (8)$$

   kur I(x,y) yra akies paveikslėlis. Minimumai imami dėl to, kad vyzdys paprastai būna žemo intensyvumo. Prieš vyzdžio centro paiešką paveikslėliui dar galima pritaikyti medianinį filtrą, kuris sumažintų triukšmo įtaką ir sustiprintų aukšto ir žemo intensyvumo sritis.
2. Sudaromas binarinis 120x120 dydžio paveikslėlis, kurio centras yra taške (X_p, Y_p), naudojant tos srities histogramą ir pritaikant tam tikrą slenkstį. Slenkstis užtikrina, kad vyzdžio intensyvumas skirtųsi nuo rainelės intensyvumo. Straipsnio autoriai naudojo slenkstį, kuris sudarė 26% maksimalaus galimo intensyvumo (didinant kontrast1,

slenkstis taip pat išauga). Suformuotoje srityje kartojamas ankstesnis žingsnis ir iš naujo įvertinamas vyzdžio centras, kuris pakeičia anksčiau gautąjį.

3. Pritaikant Canny operatorių, skaičiuojami tikslūs apskritimų parametrai bei gaunamos ribos ir, naudojant Hough transformaciją, surandami apskritimai. Canny operatoriaus ir Hough transformacijos straipsnio autoriai neaprašo.

Algoritmo privalumas yra tas, kad pirmuosiuose dviejuose žingsniuose sumažinama sritis, kurioje ieškosime vyzdžio ir rainelės.

2. 1. 4. 2. Vyzdžio bei rainelės centrai ir spinduliai

Jei vyzdį ir rainelę laikysime koncentriniais apskritimais, tai juos galima nesunkiai surasti, apskaičiavus vyzdžio centrą ir vyzdžio bei rainelės spindulius. Straipsnyje [CIS] aprašoma, kaip juos galima apskaičiuoti. Norint surasti vyzdžio spindulį, pirmiausia reikia rasti apytikslę jo centro poziciją (X_p, Y_p), kurią galima apskaičiuoti pagal formules, pateiktas 2. 1. 4. 1. skyriuje. Tuomet reikia eiti vaizdo elementais nuo to apytikslio centro, kol prieisime intensyvumų pokyčio erdvę, kur juoda spalva I(x,y)=0 keičiasi į baltą I(x,y)=1. x ašies kairiojoje ir dešiniojoje kryptyse apskaičiuojama x_l ir x_r atitinkamai, o y ašies viršutinėje ir apatinėje kryptyse apskaičiuojama y_u ir y_d atitinkamai. Tuomet galime rasti vyzdžio spindulį pagal šias formules:

$$R_p^x = \frac{\left|X_p - x_l\right| + \left|X_p - x_r\right|}{2}, \quad (9)$$

$$R_p^y = \frac{\left|Y_p - y_u\right| + \left|Y_p - y_d\right|}{2}, \quad (10)$$

$$R_p = R_p^x \approx R_p^y, \quad (11)$$

kur R_p, R_p^x ir R_p^y yra apytikslės vyzdžio spindulio reikšmės abiejose ašyse. R_p prisiima R_p^x reikšmę, kadangi ją mažiau paveikia akies vokai ir blakstienos.

Norint surasti tikrąjį vyzdžio centrą, pirmiausia reikia apskaičiuoti klaidą $\varepsilon = | R_p^x - R_p^y |$ ir patikrinti, ar ji nėra didesnė už maksimalią leidžiamą klaidą. Jei didesnė, tai paveikslėliui reikia pritaikyti Gauso triukšmą. Tuomet paveikslėliui vėl tikriname klaidos dydį. Kai klaida neviršija maksimalios leidžiamos reikšmės, galime apskaičiuoti vyzdžio centro koordinates:

$$(X_p, Y_p) = \left(\frac{x_r + x_l}{2}, \frac{y_u + y_d}{2} \right). \quad (12)$$

Norėdami surasti rainelės spindulį, paimsime tik tą paveikslėlio dalį, kurioje yra mažiausiai trikdžių, tokių kaip akių vokai ir blakstienos. Tokios yra sritys, esančios arčiausiai vyzdžio centro. Sudarysime langą su centru (X_p, Y_p), kurio ilgis ir plotis:

$$[M \times N] = \lfloor 2(R_p + eRi_{\max}) \times height \rfloor, \quad (13)$$

kur $eRi_{\max}$ yra maksimalus rainelės spindulio nukrypimas, atsižvelgiant į centrą. Jis priklauso nuo daugelio faktorių, susijusių su tuo kaip paveikslėlis buvo gautas, tokių kaip apšviestumas, atstumas nuo kameros ir panašių. Parametrui height straipsnio autoriai rekomenduoja 6 taškų dydį. Langui dar galima pritaikyti medianinį filtrą, sušvelninantį rainelės kraštus. Tuomet gautame lange žemo intensyvumo taškuose, horizontalioje ašyje esančiuose iš kairės pusės ir iš dešinės pusės, yra apytikslės rainelės ribos. Formulė rainelės spinduliui surasti:

$$R_i = \frac{|X_p - x_l| + |X_p - x_r|}{2}. \quad (14)$$

Surasti rainelės centrą (X_i, Y_i) padėtų ši formulė:

$$X_i = |x_r + x_l| / 2. \quad (15)$$

2. 1. 5. Aktyvių kontūrų metodai

Aktyvus kontūras (angl. active contour) [Wika], dar vadinamas **gyvate**, yra konstrukcija, įgalinanti apibrėžti objekto kontūrą iš galimai triukšmingo 2D paveiksliuko. Ši konstrukcija stengiasi minimizuoti energiją (darbo kiekį), susijusią su kontūru, kaip vidinės ir išorinės energijos sumą:

- *Išorinė energija* laikoma minimalia, kai gyvatė yra objekto krašto pozicijoje. Pats paprasčiausias požiūris pasireiškia duodant mažas reikšmes, kai reguliarizuotas gradientas aplink kontūro poziciją pasiekia savo kraštutinę vertę.
- *Vidinė energija* laikoma minimalia, kai gyvatė turi formą, kuri laikoma tinkama ieškomo objekto forma. Paprasčiausias požiūris leidžia didelę energiją prailgintiems kontūrams ir didelio kreivumo kontūrams, laikant, kad forma turi būti reguliari ir glodi kiek tik įmanoma.

2. 1. 5. 1. John Daugman siūlomas metodas

Paprasčiausia būtų laikyti, kad rainelė yra žiedinio pavidalo. Rainelės vidinis ir išorinis kraštas dažniausiai nėra koncentriniai. Tuomet paprasčiausias sprendimas būtų sukurti nekoncentrinę pseudo-polinę koordinačių sistemą rainelei atvaizduoti, atmetant prielaidą, kad rainelė ir vyzdys turi tą patį centrą ir reikalaujant tiktai to, kad vyzdys pilnai būtų rainelės viduje. Tačiau reikia pastebėti, kad dažnai vyzdžio ir rainelės kraštai nėra apskritimo pavidalo. Norint padidinti rainelės atpažinimo kokybę, segmentacijos metodas turėtų atsižvelgti ir į šį dalyką bei tiksliai nustatyti ir sumodeliuoti abu rainelės kraštus, kad ir kokios formos jie būtų. Būtent tokį metodą siūlo John Daugman savo straipsnyje „New Methods in Iris Recognition“ [Dau07].

Kadangi išorinis rainelės kraštas dažnai yra iš dalies dengiamas akies vokų, o vidinis kraštas gali būti iš dalies dengiamas apšvietimo atspindžių, o taip pat abu kraštai dar gali būti dalinai dengiami akinių stiklų atspindžių, šis segmentacijos metodas siūlo naudoti lanksčius kontūrus, kurie leistų pertraukimus ir tęstų savo trajektoriją per juos, remdamiesi principiniu pagrindu ir naudodami kitur esančius duomenis. Metodui taikomas toks apribojimas, kad abu vidinio ir išorinio krašto modeliai turi būti uždaros kreivės. Taip pat dar vienas tikslas yra įvesti glodinimo apribojimą, kuris remiasi patikimumu požymių nesuglodintoje kreivėje.

Tinkamas būdas pasiekti aprašytus segmentacijos tikslus yra aprašyti rainelės vidinį ir išorinį kraštus „aktyvių kontūrų“ terminais, kurie remiasi kontūro duomenų skaidymu diskrečiomis Fourier eilėmis. Naudojant Fourier komponentus, kurių dažnumas yra $1/(2\pi)$ kartotinis, užtikrinamas uždarumas, ortogonalumas ir užbaigtumas. Dažnumo komponentų skaičiaus pasirinkimas leidžia kontroliuoti glodinimo laipsnį ir aproksimavimo tikslumą. Iš esmės, sutrumpinimas diskrečių Fourier eilių po tam tikro skaičiaus nustatytų terminų prilygsta žemo lygio krašto kreivės duomenų filtravimui aktyvių kontūrų modelyje.

Šis metodas pavaizduotas paveikslėlyje [4 pav.] ir [5 pav.]. Kiekvieno paveikslėlio kairėje apatiniame kampe pavaizduota dvi „gyvatės“, kurių kiekvieną sudaro neapibrėžtas kaspino pavidalo duomenų pasiskirstymas ir taškinė kreivė kiekvienai „gyvatei“ (atitinkamai pavaizduota ir rainelės paveikslėlyje), kuri yra diskrečiųjų Fourier eilių aproksimacija duomenims, apimanti pratęsimą pertraukimų tarpuose. Žemiau esanti „gyvatė“ kiekvieno paveikslėlio kairiniame apatiniame kampe yra kreivinis vyzdžio krašto žemėlapis, o aukščiau esanti „gyvatė“ yra kreivinis išorinio rainelės krašto žemėlapis, kurio pradžios ir pabaigos taškai jungiasi šeštos valandos pozicijoje. Išorinio rainelės krašto duomenų „gyvatės“ pertraukimai atitinka akies voko uždengimus arba veidrodinius atspindžius. Duomenys pavaizduoti pilkai kiekvienai „gyvatei“ yra paveikslėlio gradientas spindulio kryptimi. Todėl kiekvienos „gyvatės“ storis apytikriai vaizduoja atitinkamo spindulinio krašto ryškumą. Jei rainelės kraštas būtų aprašytas kaip apskritimas, tai jį atitinkanti „gyvatė“ būtų lygi ir tiesi, tačiau dažniausiai taip nebūna.

Taškine kreive žymimų rainelės kontūrų nustatymo procedūra reikalauja apskaičiuoti Fourier skaidymą į N reguliariai išdėstytų tarpais kampinių modelių, sudarytų iš krašto duomenų, einant spindulio kryptimi. Jie žymimi $\{r_\theta\}$, kur θ kinta nuo 0 iki N-1. M diskrečiųjų Fourier koeficientų C_k, kai k kinta nuo 0 iki M-1, yra suskaičiuojama iš duomenų sekos $\{r_\theta\}$ taip:

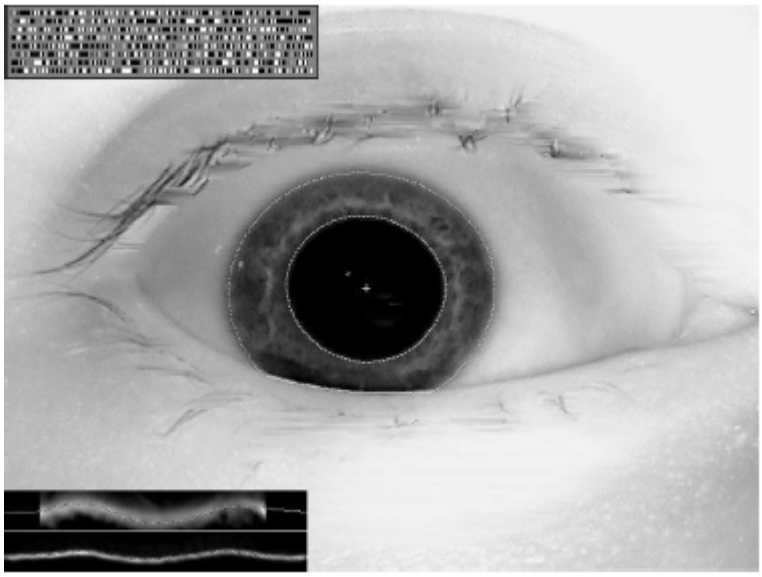
$$C_k = \sum_{k=0}^{N-1} r_\theta e^{-2\pi i k\theta / N}. \quad (16)$$

Reikia pastebėti, kad nulinės eilės koeficientas C_0 išreiškia informaciją apie vidutinę vyzdžio ar išorinio rainelės krašto kreivę, arba kitaip sakant, apie spindulį, kai vyzdys ar rainelė aproksimuojama kaip apskritimas.

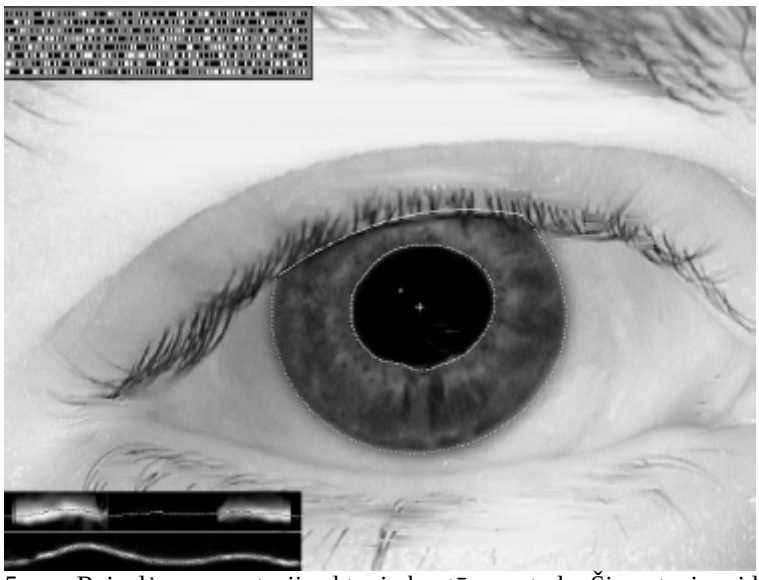
Iš šių M diskrečių Fourier koeficientų aproksimacija atitinkamam rainelės kraštui (dabar be pertraukimų ir rezoliucijos, priklausančios nuo M) gaunama kaip nauja seka $\{R_\theta\}$, kur θ kinta nuo 0 iki N-1 ir išreiškiama taip:

$$R_\theta = \frac{1}{N}\sum_{k=0}^{M-1} C_k e^{2\pi i k\theta / N} . \qquad (17)$$

Aktyvių kontūrų metoduose siekiama kompromiso tarp to, kaip tiksliai norime modelį pritaikyti visiems duomenims (tai pasiekiama, didinant M) ir to, kaip norime išlaikyti modelį paprastą ir žemos dimensijos kreivės (pasiekiama, mažinant M, pvz., M=1 duoda apskritimo modelį). Taigi Fourier koeficientų skaičius M nusako laisvės laipsnius kontūrų modelyje. Nustatyta, jog tinkamas M pasirinkimas tiksliai nustatyti vyzdžio kraštą yra M=17, o tinkamas pasirinkimas nustatyti išorinį rainelės kraštą, kur duomenys dažniausiai yra daug silpnesni, yra M=5. Taipogi verta įvesti monotoniškai mažėjančius svorius Fourier koeficientams $\{C_k\}$, kurie reguliuotų aproksimavimą $\{R_\theta\} \approx \{r_\theta\}$, ir prilygtų žemo lygio Fourier išraiškos kreivinio žemėlapio filtravimui. Šios manipuliacijos įgyvendina idėją, kad stiprūs duomenys (vyzdžio kraštas) gali būti sumodeliuoti su silpnais apribojimais, o silpni duomenys (išorinis rainelės kraštas) – su stipriais apribojimais.



4 pav. Rainelės segmentacija aktyvių kontūrų metodu. Apačioje kairėje vaizduojami kreiviniai vidinio bei išorinio rainelės krašto žemėlapiai, kurie būtų lygūs ir tiesūs, jei rainelės kraštai būtų apskritimai. Šiuo atveju, išorinis rainelės kraštas (ir jį atitinkanti viršutinė kreivė) nėra apskritimo pavidalo. Taškinės kreivės apatiniame kampe ir ant rainelės yra Fourier eilių aproksimacijos. [Dau07]



5 pav. Rainelės segmentacija aktyvių kontūrų metodu. Šiuo atveju, vidinis rainelės kraštas (ir jį atitinkanti apatinė kreivė) nėra apskritimo pavidalo. [Dau07]

2. 1. 5. 2. Vidinių ir išorinių veiksnių metodas vyzdžiui rasti

Straipsnio „Iris Segmentation: Detecting Pupil, Limbus and Eyelids“ [AT06] autoriai aprašo kiek kitokį aktyvių kontūrų metodą nei John Daugman.

Vyzdžio kontūro centrinis taškas apibrėžiamas kaip viršūnių vidurkis ir skaičiuojamas pagal tokią formulę:

$$C = (x_C, y_C) = \frac{1}{N}\sum_{i=1}^{N} V_i \,, \qquad (18)$$

kur N yra viršūnių skaičius, o V_i - i-toji viršūnė.

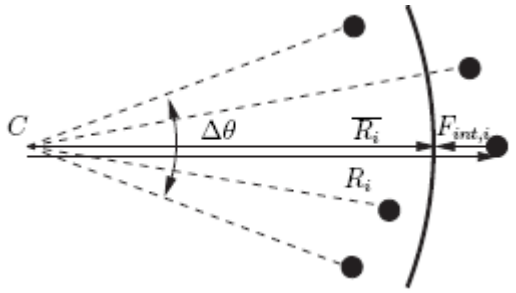
Straipsnio autoriai laiko, kad kontūras yra nenutrūkstantis, o jo kampinę rezoliuciją pasirinko pagal CASIA duomenų bazės akių paveikslėlių vidutinį vyzdžių spindulį. Vidutinis spindulys buvo nustatytas 45 vaizdo elementai, apskritimo perimetras – apytiksliai 285 vaizdo elementai, todėl kampinė rezoliucija pasirinkta 400 kampų. Segmentacijos proceso metu viršūnių skaičius yra konstanta, o kiekviena viršūnė atvaizduoja tam tikrą kampą. Tai vadinama kampiniais veiksniais, kurie iteratyviškai nustato kampinėse pozicijose esančias viršūnes, atsižvelgdami į atnaujintą centro tašką.

Kiekviena kontūro viršūnė vaizduojama kaip vektorius, turintis tam tikrą spindulį ir kryptį centrinio taško atžvilgiu. Norint gauti glodžią kreivę artimos apskritimui formos, taikomi vadinamieji vidiniai veiksniai spindulio kryptimi padaryti, kad viršūnės tam tikrame kampiniame intervale $\Delta\theta$ turėtų vienodą spindulį – viršūnių vidutinį spindulį (pastumti viršūnes). Vidinio veiksnio dydis vienai viršūnei apibrėžiamas taip:

$$\left|F_{\text{int},i}\right| = \frac{1}{N_{\Delta\theta}} \times \left(\sum_{\theta_k=\theta_i-\frac{\Delta\theta}{2}}^{\theta_k=\theta_i+\frac{\Delta\theta}{2}} R_{\theta_k} \right) - R_i \,, \qquad (19)$$

kur $\Delta\theta$ - kampinis intervalas, $N_{\Delta\theta}$ - viršūnių skaičius kampiniame intervale, o R_i - i-tosios viršūnės spindulys.

Paveikslėlyje [6 pav.] pavaizduoti apibrėžti vidiniai veiksniai.



6 pav. Vidiniai veiksniai viršūnei. [AT06]

Išoriniai veiksniai apibrėžiami patraukti kreivę link maksimalaus apskritiminio gradiento ir taikomi spindulio kryptimi. Apskritiminį gradientą galima būtų gauti skirtingiems kampams intervale $\Delta\theta$, naudojant Daugman'o integro-diferencialinį operatorių [2. 1. 1.] (1). Tačiau šis operatorius integruoja intervale [0, 2π], o tai reiškia apskritimą. Norint gauti artimą apskritimui, bet na visai apskritimo formą, reikia imti mažesnį kampinį intervalą. Straipsnio autoriai siūlo imti $\Delta\theta = \frac{\pi}{10}$ tiek vidiniams, tiek išoriniams veiksniams skaičiuoti. Išoriniai veiksniai apibrėžiami pagal atstumą nuo centrinio taško iš maksimalaus apskritiminio gradiento, gauto kiekviename kampe, atėmus spindulinę viršūnės vietą toje pačioje kampinėje pozicijoje. Šie veiksniai išreiškiami formule:

$$\left|F_{ext,i}\right| = \arg\max_{(R_C)} \left| \frac{\partial}{\partial r} \int_{\theta_i - \frac{\Delta\theta}{2}}^{\theta_i + \frac{\Delta\theta}{2}} \frac{I(x,y)}{2\pi r} ds \right| - R_i , \qquad (20)$$

kur R_C yra kontūro centro atstumas nuo maksimalaus apskritiminio gradiento kampu θ_i.

Toliau skaičiuojama vidinių ir išorinių veiksnių kompozicija kontūro viršūnėms ir kiekvienos viršūnės pakeitimas spindulio kryptimi gaunamas iteratyviškai pagal formulę:

$$R_{V_i}(n+1) = R_{V_i}(n\} + F_{\text{int},i} + \alpha \times F_{ext,i} , \qquad (21)$$

kur R_{V_i} yra i-tosios viršūnės spindulys, o α - svoris, kuris kontroliuoja kontūro judėjimo žingsnį ir palaiko vidinių bei išorinių veiksnių pusiausvyrą.

2. 1. 6. Iteratyvus algoritmas akies vokams ir išoriniam rainelės kraštui rasti

Straipsnio „Iris Segmentation: Detecting Pupil, Limbus and Eyelids" [AT06] autoriai vyzdžio kraštui rasti naudoja Aktyvių kontūrų metodą [2. 1. 5. 2..], o išoriniam rainelės kraštui rasti – iteratyvų algoritmą, randantį ir akies vokus. Akies voko radimo procedūra remiasi elipsiniais kontūrais, modeliuojamais pagal akies obuolio sferinę formą ir ieškomą akies voko

kreivę, esant skirtingiems akies atvėrimo laipsniams. Akies atvėrimas apibrėžiamas kaip kampinė akies voko pozicija, atsižvelgiant į sferos centrą. Kadangi laikoma, kad akies obuolys yra sferinės formos, tai jis nusakomas tokia formule:

$$x^2 + y^2 + z^2 = R_{eyeball}^2 . \qquad (22)$$

Akies voko kreivę galima gauti, kertant sferą plokštuma, einančia per x ašį kampu ϕ z ašies atžvilgiu. Tą plokštumą galima apibrėžti taip:

$$\tan(\phi) = \frac{y}{z} . \qquad (23)$$

Akies voko kontūras vaizduojamas paveikslėlyje [7 pav.]. Kirtimo kreivė yra elipsinio pavidalo:

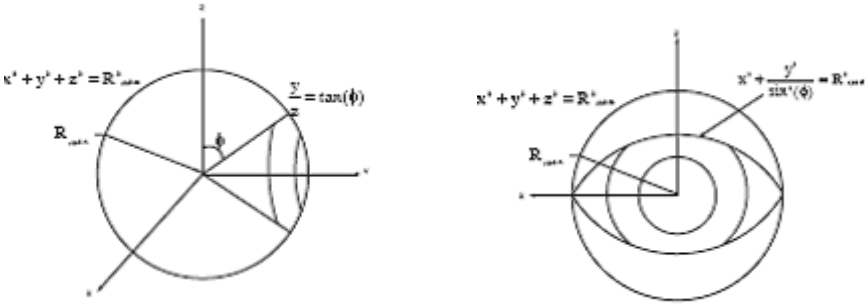
$$x^2 + \frac{y^2}{\sin(\phi)^2} = R_{eyeball}^2 . \qquad (24)$$

Ši kreivė transformuojama į polines koordinates ir išreiškiama per kampinę informaciją:

$$x = r\sin(\theta), \quad y = \cos(\theta), \qquad (25)$$

$$r = \frac{R_{eyeball}}{\sqrt{\sin(\theta)^2 + \frac{\cos(\theta)^2}{\sin(\phi)^2}}} . \qquad (26)$$

Algoritmas rainelės ir akies vokų kraštų ieško iteratyviškai ir pašalina surastas akies vokų sritis kitai iteracijai. Atsižvelgiant į vyzdžio centrą, šalinamos vaizdo elementų reikšmės, kur rainelės spindulys yra didesnis negu viršutinio arba apatinio akies voko spindulys. Taip šis procesas užmaskuoja sritis, kur rainelė yra dengiama akies vokų ir yra kartojamas tol, kol pasiekimas fiksuotas centras ir rainelės spindulys.



7 pav. Šoninis ir priekinis akies voko kreivės modelio, besiremiančio akies atvėrimo laipsniu, vaizdas. [AT06]

2. 1. 7. Rainelės ir akies vokų lokalizacija, besiremianti tekstūros segmentacija

Straipsnio „A Fast and Robust Iris Localization Method Based on Texture Segmentation“ [CWT+] autoriai aprašo rainelės lokalizavimo algoritmą, vykdomą „nuo lengvo prie sunkaus“ principu, t. y., pirmiausia atliekama vyzdžio segmentacija, paskui išorinio rainelės krašto lokalizacija, o galiausiai akies vokų aptikimas.

2. 1. 7. 1. Vyzdžio ir rainelės išorinio krašto lokalizacija

Vyzdžiui lokalizuoti straipsnio [CWT+] autoriai naudoja Haar bangeles ir perėjimo nuo abstraktaus iki tikslaus lokalizavimo strategiją. Apie Haar bangeles ir minėtą strategiją savo straipsnyje plačiau nerašo, tik pamini, kad jas naudojo. Išorinį rainelės kraštą lokalizavo naudodami integralinį diferencialinį operatorių. Diferencialinis operatorius apibrėžiamas taip:

$$f'(i) = f(i+1) + (i+2) - f(i-1) - f(i-2). \qquad (27)$$

Šis operatorius padidina rainelės išorinio krašto kontrastą.

2. 1. 7. 2. Viršutinio akies voko lokalizacija

Siūlomas metodas naudoja dažnumo savybę segmentuoti blakstienoms ir pagal jas aptinkamas akies vokas. Algoritmą sudaro tokie žingsniai:

1. Segmentuojama blakstienų sritis paveikslėlyje, norint surasti akies voko vietą. Norint suskirstyti segmentuotais blakstienas, reikia apskaičiuoti aukšto spektro energiją kiekviename regione. Jei dažnis yra pakankamai aukštas, laikome, kad tai blakstienų sritis.
2. Viršutinei blakstienų sričiai segmentuoti naudojama vyzdžio pozicijos informacija.
3. Blakstienoms pritaikomas parabolinis lankas (viršutinio voko kraštas) $y = ax^2 + bx + c$. Jei blakstienų srities taškus gausime tokiu pavidalu: $(x_1; y_1), (x_2; y_2) \ldots (x_N; y_N)$, tai lanko parametrai bus

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = (A^T A)^{-1} A^T Y, \text{ kur } A = \begin{bmatrix} x_1^2 & x_1 & 1 \\ \cdots & \cdots & \cdots \\ x_N^2 & x_N & 1 \end{bmatrix}, \text{ o } Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}.$$

4. Ieškoma kaimyninėje paraboliniam lankui, išreikštam per parametrus (a, b, c), srityje tam, kad gauti galutinį rezultatą. Pažymėkime $curve(c) = ax^2 + bx + c - y = 0$ parabolinio lanko klasterį, su kintamu parametru c ir $c_0 - \arg\max_{curve(c)} \left| \frac{\partial}{\partial c} \int_{curve(c)} I(x,y)ds \right|$, tuomet c_0 yra tikroji viršutinio akies voko pozicija.
5. Tikrieji parabolinio lanko parametrai yra (a, b, c_0).

2. 1. 7. 3. Apatinio akies voko lokalizacija

Apatinis vokas segmentuojamas, naudojant originalaus paveikslėlio histogramą. Slenkstis nustatomas, apskaičiavus rainelės vaizdo elementų reikšmių vidurkį ir nuokrypį. Pasirenkame viršutinius apatinio voko taškus ir ieškome krašto pagal tokius žingsnius:

1. Apatinio voko sritis suskirstoma segmentais.
2. Apskaičiuojami apatinio voko krašto taškai.
3. Apatiniam vokui pritaikomi taškai, gauti 2. žingsnyje.
4. Galutinis rezultatas ieškomas kaimyninėje srityje.

3. ir 4. žingsniai panašūs į viršutinio voko lokalizacijos algoritmo 4. ir 5. žingsnius atitinkamai.

2. 1. 8. Blakstienų pašalinimas

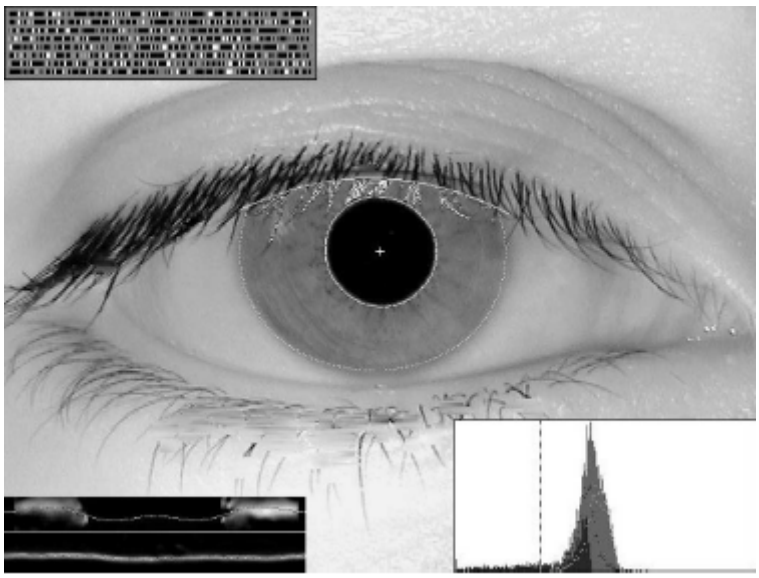
2. 1. 8. 1. Blakstienų pašalinimas statistiniu būdu

John Daugman straipsnyje [Dau07] aprašytas metodas, pašalinti blakstienoms, dengiančioms rainelės dalį paveikslėlyje, kad jos neigiamai neįtakotų rainelės požymių surinkimo. Statistinis blakstienų pašalinimo metodas priklauso nuo to, ar rainelės vaizdo elementų pasiskirstymas yra daugialypis. Jei žemesnioji dalis railelės vaizdo elementų histogramos palaiko daugialypiškumo hipotezę, gali būti suskaičiuojamas slenkstis ir jo išorėje atsidūrę vaizdo elementai išmetami, kad neįtakotų rainelės kodo.

Šį metodą iliustruoja paveikslėlis [8 pav.]. Apatiniame dešininiame kampe nubrėžtos keturios histogramos, suskaičiuotos tik iš segmentuotos rainelės (tarp nustatytų akies vokų) vaizdo elementų. Pilka spalva pavaizduotas išsidėstymas yra histograma iš visų rainelės vaizdo elementų (intervale nuo 0 iki 255). Dvi taškinės histogramos padalija ją į dvi dalis, iš kurių viena tik apatinei rainelės daliai (balta taškinė kreivė), o kita tik viršutinei (juoda taškinė kreivė). Juodos spalvos histograma yra skirtumas tarp minėtų dviejų taškinių histogramų. Svarbu, ar skirtumo diagramos didėjimas iš kairės gali būti statistiškai atskiriamas nuo motininio (pilko) pasiskirstymo. Jei taip būtų, remiantis nukrypimais tarp jų atitinkamų reikšmių ketvirtuoju laipsniu, galėtume patvirtinti hipotezę, kad ant rainelės yra užkritusios blakstienos.

Vertikali brūkšninė linija histogramos panelėje nurodo apskaičiuotą slenkstį, kur šiam rainelės paveikslėliui yra palaikoma minėta hipotezė. Jei ta hipotezė dar išlaiko tolimesnį nukrypimo tarp slenksčio ketvirtu laipsniu ir motininio pasiskirstymo vidurkio testą, kuris patvirtina, kad ne tik yra dvi populiacijos, bet jos yra tarpusavyje pakankamai skirtingos, tuomet slenkstis patvirtinamas ir vaizdo elementai žemiau jo yra laikomi atsiradusiais dėl užkritusių blakstienų.

Pačiame rainelės paveikslėlyje [8 pav.], nustatytos blakstienos, užeinančios ant rainelės yra pažymėtos baltais vaizdo elementais. Jų pozicijos įrašytos maskavimo masyve, kad jie neįtakotų duomenų, koduojančių rainelės tekstūrą.



8 pav. Statistinis blakstienų aptikimas iš rainelės vaizdo elementų histogramos ir slenksčio nustatymas joms pašalinti (pažymėtos baltai), kad neįtakotų rainelės kodo. [Dau07]

2. 1. 8. 2. Atskirų ir daugialypių blakstienų pašalinimo metodas

W. K. Kong ir D. Zhang savo straipsniuose [KZa], [KZb] išskiria du blakstienų tipus: atskiras blakstienas, kurios išsiskiria iš kitų blakstienų (kaimyniniai vaizdo elementai aplink atskirą blakstieną turi nepriklausyti kitoms blakstienoms) ir daugialypes blakstienas, kurios persidengia tarpusavyje ir jų neįmanoma atskirti. Jie aprašo metodus abiejų tipų blakstienoms surasti akies paveikslėlyje, kad jos neįtakotų rainelės kodo.

Atskiras blakstienas galima aptikti, naudojant Gabor filtro realiąją dalį. Pvz. 1-D Gabor filtro forma:

$$G(x,u,\sigma) = \exp\{x^2/2\sigma^2\}\cos(2\pi ux), \qquad (28)$$

kur u yra sinusoidinės bangos dažnis, o σ - standartinė Gauso išvestinė.

Jei gautas atskiros blakstienos funkcijos sandaugos su $G(x,u,\sigma)$ rezultatas yra mažesnis už tam tikrą nustatytą slenkstį, laikoma, kad taškas priklauso blakstienai. Matematiškai tai išreiškiama taip:

$$f(x) * G(x,u,\sigma) < K_1, \qquad (29)$$

kur K_1 yra iš anksto nustatytas slenkstis. Straipsnio [KZb] eksperimentuose šis slenkstis buvo parinktas -45.

Daugybines blakstienas galima aptikti pagal mažą intensyvumo pokytį mažame lange. Jei tas pokytis mažesnis už slenkstinę reikšmę, tai lango centras laikomas tašku blakstienoje. Matematiškai tai išreiškiama taip:

$$\frac{\sum_{i=-N}^{N}\sum_{j=-N}^{N}(f(x+i,y+j)-M)^2}{(2N+1)^2-1} < K_2, \qquad (30)$$

kur M yra intensyvumų vidurkis mažame lange, (2N+1) yra lango dydis, o K_2 - slenkstis. Straipsnių [KZa], [KZb] autoriai savo eksperimentuose naudojo lango dydį 5 ir 5. Straipsnio [KZb] eksperimentuose šis slenkstis buvo parinktas 6.

Norint padidinti blakstienų aptikimo metodo tikslumą, įvedami susijungimo kriterijai, kuris reiškia, jog kiekvienas blakstienos taškas turi jungtis su kitu blakstienos tašku arba su akies voku. Jei kuris nors taškas patenkina, kurį nors iš minėtų dviejų kriterijų, tikrinami jo kaimyniniai vaizdo elementai, ar jie priklauso blakstienai arba akies vokui. Jei nei vienas kaimyninis vaizdo elementas nepriklauso nei blakstienai, nei vokui, tai tas vaizdo elementas, kurio kaimynai tikrinami, laikomas nepriklausančiu blakstienoms.

2. 1. 9. Atspindžių aptikimas

W. K. Kong ir D. Zhang savo straipsniuose [KZa], [KZb] aprašo atspindžių akyje aptikimo metodą. Jie išskiria du atspindžių tipus: stiprus atspindys ir silpnas atspindys. Vaizdo elementas priskiriamas stipriam atspindžiui, jei jo intensyvumas yra didesnis už tam tikrą slenkstį. Silpnas atspindys yra perėjimas iš stipraus atspindžio į rainelę.

Stiprus atspindys matematiškai nusakomas tokia nelygybe:

$$f(x,y) < K_3, \qquad (31)$$

kur $f(x,y)$ - paveikslėlio intensyvumas taške (x,y), K_3 - slenkstis, W. K. Kong ir D. Zhang eksperimentuose turintis reikšmę 180.

Silpną atspindį nusako tokia statistinė nelygybė:

$$\mu + \alpha\sigma < f(x,y), \qquad (32)$$

kur μ ir σ yra rainelės intensyvumų pasiskirstymo vidurkis ir standartinis nuokrypis, o α - parametras kontroliuoti klaidingo priėmimo ir klaidingo atmetimo klaidoms. Jei kuris nors taškas aplink stiprų atspindį tenkina silpno atspindžio tikrinimo nelygybę, jis laikomas silpno atspindžio tašku. μ ir σ yra apytiksliai lygūs $\overline{X}$ ir S, kurie skaičiuojami pagal tokias formules:

$$\overline{X} = \frac{\sum_{(x,y)\in P} f(x,y)}{N_P}, \qquad (33)$$

$$S = \sqrt{\frac{\sum_{(x,y)\in P} f^2(x,y) - N_P \overline{X}^2}{N_P - 1}}, \qquad (34)$$

kur P – rainelės vaizdo elementų, neįtakojamų jokio triukšmo (nepriklausančių blakstienoms, akies vokui ar atspindžiams) aibė, o N_P - vaizdo elementų skaičius aibėje P.

Silpno atspindžio taškų aptikimo iteratyvus algoritmas yra toks:

1. Nustatyti $P = P_j$ ir j=0. P_j - vaizdo elementų, nepriklausančių blakstienoms, akies vokui ar atspindžiams, aibė. Pagal aukščiau pateiktas $\overline{X}$ ir S skaičiavimo formules apskaičiuoti $\overline{X}_j$ ir S_j. Taip pat reikia suskaičiuoti vaizdo elementų skaičių N_j aibėje P_j. Q_j pažymėkime vaizdo elementus, priklausančius stipriam atspindžiui.
2. Pagal silpno atspindžio tikrinimo statistinę nelygybę patikrinti visus vaizdo elementus iš aibės P_j, kurie jungiasi su kuriuo nors vaizdo elementu iš Q_j. Jei vaizdo elementas x tenkina nelygybę, jis išimamas iš aibės P_j ir priskiriamas Q_j. Dabar reikia atnaujinti $\overline{X}_j$, S_j ir N_j pagal šias lygtis:

$$N_{jnew} = N_j - 1, \qquad (35)$$

$$\overline{X}_{jnew} = \frac{\overline{X}_j N_j - x}{N_{jnew}}, \quad (36)$$

$$S_{jnew} = \sqrt{\frac{N_{jnew}(S_j^2 - \overline{X}_{jnew}^{\,2}) + N_j \overline{X}_j^2 - x^2}{N_{jnew} - 1}}. \qquad (37)$$

3. Jei nei vienas vaizdo elementas antrajame žingsnyje nepašalinamas iš aibės P_j, nustatyti $P = P_j$ ir išeiti iš ciklo. Priešingu atveju, kartoti antrąjį algoritmo žingsnį.

2. 1. 10. Fourier trigonometrija ir nukrypęs žvilgsnis

Nukrypęs žvilgsnis yra viena iš rainelės segmentacijos problemų. Dažnai būna sunku nuskenuoti nejudančio tiesaus žvilgsnio rainelę, kuomet žmogus sulygina savo optinę ašį su kameros optine ašimi. Sutampančių ašių reikalavimo galima atsisakyti, pakoreguojant projekcinę rainelės deformaciją, kai ji nuskenuota nesutampant ašims, nustačius faktinius žvilgsnio parametrus. John Daugman aprašė metodą, kuris šiuos parametrus įvertina, remiantis Fourier trigonometrija [Dau07].

Ieškomi žvilgsnio parametrai apima du sferinius akies pozos kampus, tačiau projekcinė geometrija taipogi priklauso nuo atstumo tarp akies ir kameros, kuris gali būti nežinomas, ir rainelės išlinkimo į išorę, kuris paprastai nėra nulinis. Jei šiems faktoriams pritaikomos supaprastinančios prielaidos ir aproksimacijos, tuomet gali pakakti paprasčiausios afininės projekcinės transformacijos padaryti taip, kad rainelė būtų atpažįstama, lyginant su ja pačia nuskenuota skirtingose pozose. Šiame žvilgsnio įvertinimo ir paveikslėlio pataisymo metode nesilaikoma prielaidos, kad vyzdys yra apskritimo formos.

Šis metodas kilo iš pastebėjimo, kad skaidymas Fourier eilėmis X ir Y vyzdžio krašto koordinačių turi kontūrų deformacijos informaciją, susijusią su nukrypusiu žvilgsniu,

santykiuose tarp kiekvieno eilių skaidymo mažiausio pasikartojimo termino realių ir menamų koeficientų.

Pirmiausia išnagrinėkime paprasčiausią atvejį, kai rainelė yra apskritimo pavidalo. Vyzdžio krašto koordinačių vektorius pažymėkime X(t) ir Y(t). t kinta intervale nuo 0 iki 2π viename cikle, einant aplink uždarą kreivę. Vyzdžio spindulį pažymėkime A ir supaprastintai laikykime, kad vyzdys yra rainelės centre. Tuomet reikiamos funkcijos bus $X(t) = A\cos(t)$ ir $Y(t) = A\sin(t)$. Tuo atveju, kai žvilgsnis nukrypęs pagrindinės ašies kryptimi ir jei laikysime, kad kameros atstumas yra didelis lyginant su rainelės skersmeniu ir dėl to paveikslėlis sumažintas pagrindinės ašies kryptimi, funkcijos taps tokiomis: $X(t) = A\cos(t)$ ir $Y(t) = B\sin(t)$, kur $A \neq B$. Galiausiai, jei žvilgsnio nukrypimas yra ne pagal pagrindinę ašį, bet labiau kryptimi θ, tai funkcijos aprašomos taip:

$$X(t) = [A\cos^2\theta + B\sin^2\theta]\cos(t) + [(B-A)\cos\theta\sin\theta]\sin(t), \qquad (38)$$

$$Y(t) = [(B-A)\cos\theta\sin\theta]\cos(t) + [B\cos^2\theta + A\sin^2\theta]\sin(t). \qquad (39)$$

Mes ieškome žvilgsnio nukrypimo krypties ir dydžio, bet turime šiuos parametrus Fourier koeficientų forma harmoninėse funkcijose cos(t) ir sin(t), vaizduojamose kaip kontūro duomenų tiesinė kombinacija X(t) ir Y(t). Tiksliau sakant, mažiausias kompleksinis koeficientas funkcijos X(t) Fourier eilių skaidyme yra sudarytas iš realios ir menamos dalių, koeficientų a ir b atitinkamai, kurie nusako (38) išraišką tokiu būdu:

$$a = A\cos^2\theta + B\sin^2\theta, \qquad (40)$$

$$b = (B-A)\cos\theta\sin\theta. \qquad (41)$$

Panašiai, mažiausias kompleksinis koeficientas funkcijos Y(t) Fourier eilių skaidyme yra sudarytas iš realios ir menamos dalių, koeficientų c ir d atitinkamai, kurie nusako (39) išraišką tokiu būdu:

$$c = (B-A)\cos\theta\sin\theta, \qquad (42)$$

$$d = B\cos^2\theta + A\sin^2\theta. \qquad (43)$$

Taigi galime išvesti žvilgsnio nukrypimo parametrus, kurių ieškome, suskaičiuodami kontūro funkcijų X(t) ir Y(t) Fourier koeficientus (a, b, c ir d). Šis skaičiavimo procesas nepriklauso nuo aukštesnio laipsnio Fourier koeficientų, kurie egzistuotų, jei vyzdys turėtų sudėtingesnę formą nei apskritimas. Metodas nėra apribotas prielaida apie apskritimo formą.

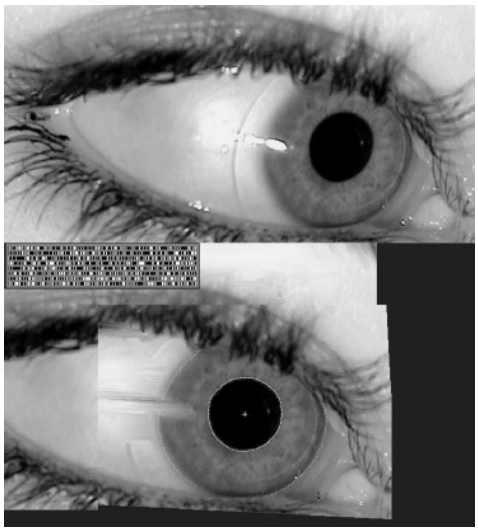
Reikia pastebėti, kad nors b ir c koeficientų skaičiavimo formulės ((41) ir (42)) iš pažiūros atrodo vienodos, bet jos skiriasi tuo, kad jos remiasi skirtingais empiriniais duomenimis. a ir b yra gaunami iš X(t), o c ir d - iš Y(t). Apskaičiuota žvilgsnio nuokrypio (modulio π) kryptis turi formą Fourier fazės informacijos, kuri nusakoma pavyzdžiui taip:

$$\theta = 0.5 \arctan\left(\frac{-b-c}{a-d}\right). \qquad (44)$$

Žvilgsnio nukrypimo dydis kryptimi θ yra išreiškiamas ne kaip kampas, o kaip projekcinis santykis $\nu = B/A$, kuris duoda kitą afininės transformacijos parametrą tokiu pavidalu:

$$\nu = \frac{(a+d)\cos(2\theta)+a-d}{(a+d)\cos(2\theta)-a+d}. \qquad (45)$$

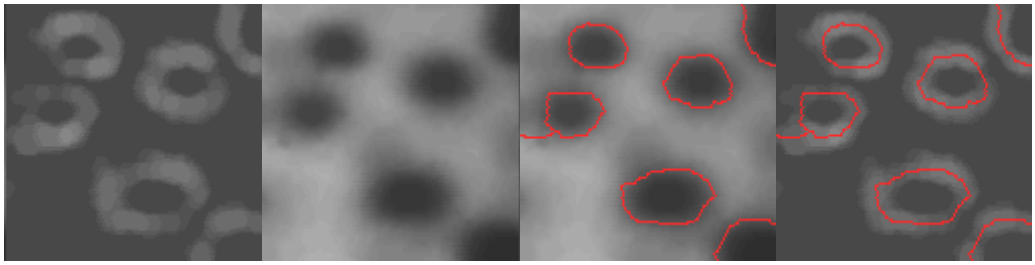
Šių parametrų įvertinimas „Fourier trigonometrijos“ metodui leidžia projekcinę geometrinę deformaciją, įvykusią dėl žvilgsnio nukrypimo, panaikinti afininės rainelės paveikslėlio transformacijos būdu. Tai pavaizduota paveikslėlyje [9 pav.], kurio viršuje parodyta akis su nukrypusiu žvilgsniu, o apačioje tas pats akies paveikslėlis po žvilgsnio nukrypimo „pataisymo“ afininės transformacijos būdu, remiantis parametrais (θ, ν). Šis metodas leidžia transformuoti paveikslėlius, kad būtų galima atpažinti tos pačios akies paveikslėlius. Metodo apribojimas yra toks, kad afininė transformacija laiko, kad rainelė yra plokščia, nors iš tiesų ji yra truputį išlinkusi.



9 pav. Fourier trigonometrija įgalina žvilgsnio įvertinimą ir akies paveikslėlio su nukrypusiu žvilgsniu transformavimą į aiškiai žiūrintį tiesiai į kamerą. Be šios transformacijos, tokių paveikslėlių neišeitų palyginti tarpusavyje. [Dau07]

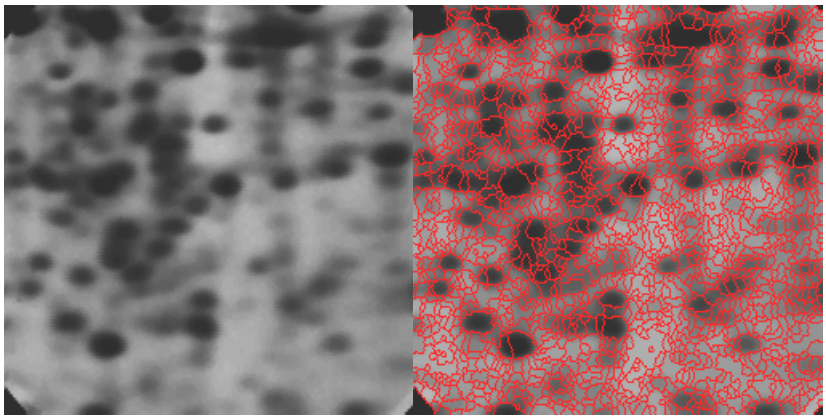
2. 1. 11. Vandens takoskyros transformacija paveikslėlių segmentacijai

Vandens takoskyros (angl. watershed) transformacija [CMM] taikoma paveikslėlių segmentacijai. Bet kuris nespalvotas (juodai baltas) paveikslėlis gali būti laikomas topografiniu paviršiumi. Jei užpildysime šį paviršių, pradėdami nuo mažiausio pilkumo lygio reikšmės, ir jei uždrausime skirtingų sričių susiliejimą, tai padalinsime paveikslėlį į sritis ir vandens takoskyros linijas. Jei pritaikytume šią transformaciją paveikslėlio gradientui, tai gautos sritys turėtų teoriškai atitikti homogenines paveikslėlio pilkumo lygių sritis [10 pav.].



10 pav. Iš kairės į dešinę: originalus paveikslėlis, gradientinis paveikslėlis, vandens takoskyra gradientiniam paveikslėliui ir galutiniai kontūrai. [CMM]

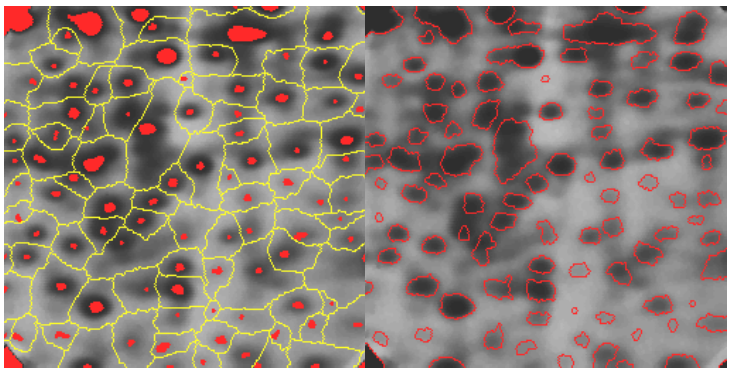
Problema yra ta, kad dažnai praktikoje ši transformacija dėl triukšmo ir netaisyklingumų gradientiniame paveikslėlyje padaro daug daugiau sričių negu reikia [11 pav.].



11 pav. Gradientinis paveikslėlis ir šio paveikslėlio vandens takoskyros transformacija. [CMM]

Šios problemos galima išvengti, naudojant patobulintą transformaciją – žymekliu kontroliuojamą vandens takoskyros (angl. marker-controlled watershed) transformaciją [12 pav.]. Pagrindinis patobulinimas yra tas, kad topografiniame paviršiuje sritys užpildomos pradedant nuo iš anksto pažymėtų žymių. Tokiu būdu išvengime per smulkios segmentacijos. Vandens takoskyros transformacija besiremianti segmentacija yra dviejų žingsnių procesas:

- Žymių ir segmentavimo kriterijaus suradimas (kriterijus ar funkcija, kuri bus naudojama atskirti sritims dažniausiai yra kontrastas arba gradientas, bet nebūtinai). Žymės gali būti parenkamos ir sužymimos paprastesniu atveju rankiniu būdu, o sudėtingesniu atveju - automatiškai.
- Žymekliu kontroliuojamos vandens takoskyros transformacijos atlikimas, remiantis ankstesniame punkte išvardintais dviem elementais.



12 pav. Dėmių bei fono žymės ir žymekliu kontroliuojama vandens takoskyros transformacija. [CMM]

2. 1. 12. Momentiniai invariantai paveikslėlių analizėje

Paveikslėlių momentai ir momentiniai invariantai yra naudojami objektų atpažinimo ar klasifikavimo metoduose objektų formai aprašyti. Paveikslėlių momentai yra tam tikri konkretūs pasverti vidurkiai (matematiniai momentai), skaičiuojami paveikslėlio elementų intensyvumų reikšmėms, arba tų momentų funkcijos, parinktos tokios, kad turėtų tam tikras reikiamas savybes ar interpretaciją [Wikd]. Jie yra naudojami aprašyti objektams po segmentacijos. Tarp paveikslėlio savybių, kurios randamos, naudojant paveikslėlių momentus, yra geometrinis srities plotas (ar visas intensyvumas) ir centrinis vaizdo elementas. Šiame skyrelyje toliau pateiktos momentų, šių paveikslėlio savybių ir momentinių invariantų apskaičiavimo formulės.

Norint sukonstruoti momentinius invariantus, pirmiausia reikia apibrėžti paveikslėlio funkcijos, geometrinio momento ir centrinio momento sąvokas [Flu06].

Paveikslėlio funkcija (arba *paveikslėliu*) laikome bet kurią realią 2-D funkciją f(x,y), kuri turi ribas ir baigtinį nenulinį integralą.

Geometrinis paveikslėlio f(x,y) *momentas* m_{pq}, kur p ir q yra neneigiami sveikieji skaičiai ir suma (p + q) vadinama momento *eile*, apibrėžiamas tokia formule:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy, \quad (46)$$

kur p, q = 0, 1, 2, …

Diskrečiu atveju, kai momentai skaičiuojami skaliariniam (pilkumo lygių) paveikslėliui, integralas momento apibrėžime turi būti pakeistas suma. Tuomet formulė (46) pavirsta į tokią:

$$m_{pq} = \sum_{x=0}^{N}\sum_{y=0}^{N} x^p y^q f_{ij}, \quad (47)$$

kur N yra paveikslėlio vaizdo elementų skaičius, o f_{ij} - atskirų vaizdo elementų pilkumo lygių reikšmės.

Tarp paveikslėlio savybių, gaunamų iš momentų, yra šios [Wikd]:

- *Plotas* (binariniams paveikslėliams) arba *pilkumo lygių reikšmių suma* (pilkumo lygių paveikslėliams): m_{00};
- *Centrinis vaizdo elementas*: $\{\bar{x}, \bar{y}\} = \{m_{10} / m_{00}, m_{01} / m_{00}\}$. (48)

Centrinis momentas μ_{pq} apibrėžiamas tokia formule [Flu00], [Flu06], [Wikd]:

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\bar{x})^p (y-\bar{y})^q f(x,y)dxdy, \quad (49)$$

kur koordinatės $\{\bar{x}, \bar{y}\}$ nurodo f(x,y) centrą (centrinį paveikslėlio vaizdo elementą, apskaičiuojamą pagal formulę (48)).

Toks pats integralo pakeitimas suma kaip paprastajam momentui (formulėje (47)) skaitmeninio paveikslėlio atveju pritaikomas ir centriniam momentui. Tuomet formulė (49) pavirsta į tokią:

$$\mu_{pq} = \sum_{x=0}^{N}\sum_{y=0}^{N}(x-\bar{x})^p (y-\bar{y})^q f_{ij}. \quad (50)$$

Pirmos ir antros eilės centriniai momentai yra tokie [Wikd]:

$$\mu_{00} = m_{00},$$
$$\mu_{01} = 0,$$
$$\mu_{10} = 0, \quad (51)$$
$$\mu_{11} = m_{11} - \bar{x}m_{01} = m_{11} - \bar{y}m_{10},$$
$$\mu_{20} = m_{20} - \bar{x}m_{10},$$
$$\mu_{02} = m_{02} - \bar{y}m_{01}.$$

Šias formules galima patikrinti išreiškiant (50)- tą formulę per (47)- tą ir imant p, q = 0, 1, 2.

Poslinkio invariantų rolę gali atlikti centriniai momentai. 1962 m. M. K. Hu paskelbė septynis poslinkio, mastelio keitimo ir *posūkio invariantus* (objektui paveikslėlyje skaičiuojami momentai, kurių reikšmės nekinta perkeliant, didinant ar mažinant bei pasukant tą objektą), sudarytus iš antros ir trečios eilės momentų. Pirmi du iš jų yra tokie [Flu00], [Flu06], [Wikd]:

$$\phi_1 = \mu_{20} + \mu_{02}, \quad (52)$$

$$\phi_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2. \quad (53)$$

2. 2. Rainelės normalizavimas

Lokalizuota rainelė transformuojama iš Cartesian į poliarinę koordinačių sistemą. Tokiu būdu sutvarkomi skirtingi rainelių dydžiai, vyzdžio plėtimasis ir apšviestumo pakitimai**.** „Iris Recognition using Corner Detection“ autoriai [GMR+] aprašo tą patį rainelės normalizavimo

metodą, kaip ir John Daugman [Daua], [Dau06], [Dau07], [Dau93], tik jie išreiškia formules per sinusus ir kosinusus.

Rainelės lokalizacija pažymi žiedo pavidalo rainelę paveikslėlyje. Atsižvelgiant į vyzdžio plėtimosi galimybę, kuomet skirtinguose paveikslėliuose jis yra skirtingo dydžio, reikia pakeisti koordinačių sistemą, išskiriant rainelę ir visus jos taškus pakeičiant į jų poliarinius atitikmenis kaip pavaizduota paveikslėlyje [13 pav.]. Šis paveikslėlis turi 80 x 360 vaizdo elementų. Tai reiškia, kad kiekvienu kampu žingsnio dydis yra tas pats. Todėl, jei vyzdžiui plečiantis parenkami tie patys taškai ir tiems taškams priskiriami jų poliariniai atitikmenys, tai atitikmenų parinkimo procesas yra nekintantis plėtimosi atžvilgiu. Žiedinė rainelės sritis transformuojama į poliarinį jos atitikmenį, naudojant tokias lygtis:

$$I(x(\rho,\theta), y(\rho,\theta)) \rightarrow I(\rho,\theta), \qquad (54)$$

$$x_p(\rho,\theta) = x_{p0}(\theta) + r_p * \cos(\theta),$$

$$y_p(\rho,\theta) = y_{p0}(\theta) + r_p * \sin(\theta),$$

$$x_i(\rho,\theta) = x_{i0}(\theta) + r_i * \cos(\theta),$$

$$y_i(\rho,\theta) = x_{i0}(\theta) + r_i * \sin(\theta).$$

Čia r_p – vyzdžio spindulys, r_i - rainelės spindulys, $(x_p(\theta), y_p(\theta))$ ir $(x_i(\theta), y_i(\theta))$ – vyzdžio krašto ir išorinio rainelės krašto koordinatės kryptimi θ, $\theta \in [0;2\pi]$, $\rho \in [0;1]$.

Straipsnio [CIS] autoriai siūlo kitokias formules rainelės paveikslėliui normalizuoti (rainelės apskritimą išskleisti į stačiakampį paveikslėlį):

$$I_n(X,Y) = I_0(x,y), \qquad (55)$$

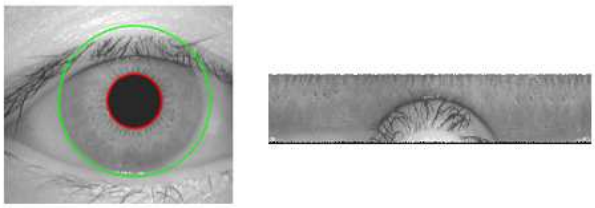
$$x = x_p(\theta) + (x_i(\theta) - x_p(\theta))\frac{Y}{M}, \qquad (56)$$

$$y = y_p(\theta) + (y_i(\theta) - y_p(\theta))\frac{Y}{M}, \qquad (57)$$

$$\theta = 2\pi X / N, \qquad (58)$$

kur I_n yra naujasis MxN dydžio paveikslėlis. Straipsnyje [CIS] buvo parinkta M=64, N=512. Taip pat siūloma normalizuotam paveikslėliui padidinti kontrastą, kad išryškėtų tos sritys, kuriose yra rainelių palyginimui reikalingi požymiai.

Taigi, normalizuojant rainelę, ji atvaizduojama į bedimensę, normalizuotą koordinačių sistemą, kuri nekinta priklausomai nuo dydžio, kintančio skenavimo atstumo, akies pozicijos paveikslėlyje, plečiantis vyzdžiui arba optinio padidinimo atveju. Ši koordinačių sistema nėra griežtai polinė, nes nereikalauja būtinai daryti prielaidą, kad vyzdžio ir rainelės apskritimai yra koncentriški (kadangi vyzdžio tikrasis centras dažnai nesutampa su rainelės centru) ir kad jų kraštai yra apskriti (pvz., dalį rainelės gali dengti akies vokas).

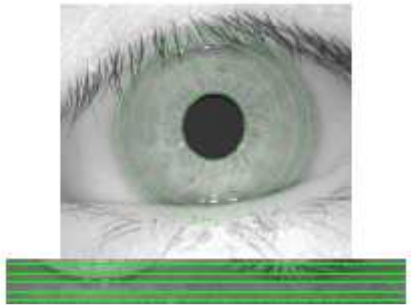


13 pav. Rainelės normalizavimas [GMR+]

2. 3. Požymių išrinkimas

2. 3. 1. Koncentrinių apskritimų ir Greitosios Fourier transformacijos metodas

Straipsnyje [GKN03] aprašytas paprastas metodas, kai rainelės požymiai surenkami tam tikrais intervalais, einant apskritimais ir jiems pritaikoma Greitoji Fourier transformacija (angl. Fast Fourier Transform – FFT). Metodą iliustruoja paveikslėlis [14 pav.]. Šiuo atveju požymiai buvo parenkami, imant ryškumo lygius kas 50 vaizdo elementų intervalų išilgai apskritimų. Žinoma, būtų geriausia akių vokus ir blakstienas pašalinti dar prieš požymių surinkimą, tačiau straipsnyje aprašytame metode tai nebuvo atlikta.



14 pav. Koncentrinių apskritimų požymių parinkimo metodas. [GKN03]

2. 3. 2. Rainelės požymių kodavimas 2D bangelių demoduliacijos būdu

Vienas iš metodų išrinkti rainelės požymius yra John Daugman pasiūlyta 2D bangelių demoduliacija [Daua], [Dau06], [DD01]. Izoliuotos rainelės fazinė informacija išgaunama demoduliacijos būdu, naudojant kvadratūrines 2D Gabor bangeles. Šis kodavimo procesas yra pavaizduotas paveikslėlyje [15 pav.]. Tai prilygsta rainelės fazės skaidymui dalimis, nustatant kuriame kompleksinės plokštumos kvadrante kiekviena fazės dalis (vaizduojama kaip sinusinė banga) turi būti, kai duota rainelės sritis yra suprojektuota ant kompleksinių 2D Gabor bangelių:

$$h_{\{\mathrm{Re,Im}\}} = \mathrm{sgn}_{\{\mathrm{Re,Im}\}} \int_{\rho}\int_{\phi} I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} \cdot e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho d\rho d\phi . \quad (59)$$

$h_{\{\mathrm{Re,Im}\}}$ - kompleksinis bitas, kurio realioji ir menamoji dalys yra arba 1, arba 0 (sgn), priklausomai nuo 2D integralo ženklo.

$I(\rho,\phi)$ - neapdorotas rainelės paveiksliukas bedimensėje polinėje koordinačių sistemoje, kuri yra pastovaus dydžio, nekinta priklausomai nuo akies padėties paveikslėlyje, t.y. nepriklauso nuo poslinkio, ir kurią galima pritaikyti netgi tuo atveju, kai vyzdys išsiplečia, kadangi spindulinė koordinatė ρ atvaizduoja rainelę nuo jos vidinio krašto iki jos išorinio krašto į vienetinį intervalą *[0;1]* ir tokiu būdu panaikina požymių deformacijas, kurias sukelia vyzdžio plėtimasis.

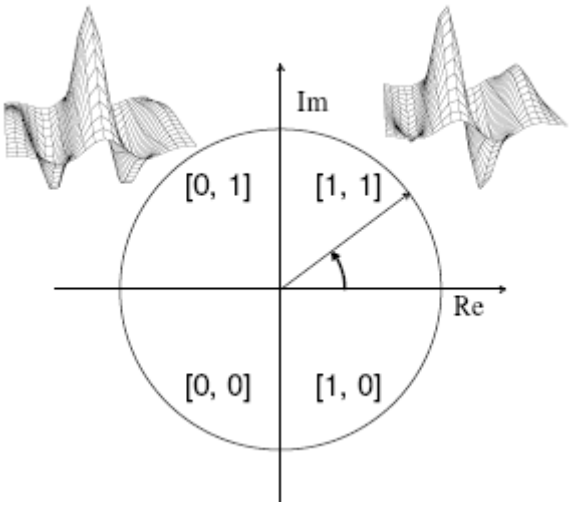
α ir β - 2D bangelės dydžio parametrai (bangelės gali būti daugelio dydžių), apimantys aštuonias bangelės sritis nuo 0.15 mm iki 1.2 mm.

ω - bangelės dažnis, apimantis 3 oktavas, atvirkščiai proporcingas β.

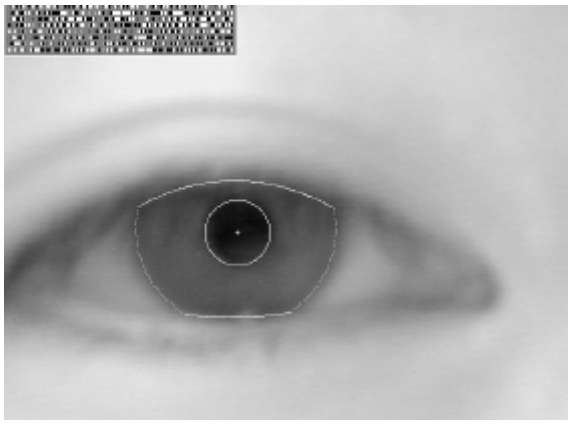
(r_0,θ_0) - polinės koordinatės kiekvieno rainelės regiono, kuriam skaičiuojamos sinusinės bangos koordinatės $h_{\{\mathrm{Re,Im}\}}$.

Tokia fazių kvadrantų kodavimo seka yra pailiustruota vienai rainelei bitų srautu ir grafiškai parodyta paveikslėlyje [1 pav.]. Paveikslėlyje [15 pav.] pavaizduotas ciklinis fazės kodas: pereinant tarp dviejų bet kurių gretimų fazės kvadrantų, keičiasi tik vienas bitas, ne taip kaip dvejetainiame kode, kur gali pasikeisti du bitai ir kai kurios klaidos tampa svarbesnėmis už kitas. Iš viso 2,048 faziniai bitai (256 baitai) suskaičiuojami kiekvienai rainelei, tačiau lyginant su ankstesniais algoritmais, buvo įvestas pagerinimas, kad dabar dar toks pats skaičius maskuojančių bitų apskaičiuojamas pažymėti, ar kuri nors rainelės sritis yra dengiama akies voko, ar užkrenta blakstiena, ar yra atspindys, ar yra riba kontaktinių lęšių ir šie dalykai turi būti ignoruojami demoduliacijos kode.

Tiktai fazės informacija naudojama rainelėms atpažinti, nes amplitudės informacija nelabai tinka rainelėms atskirti ir priklauso nuo šalutinių faktorių, tokių kaip paveikslėlio kontrastas, apšvietimas ir kameros savybės. Fazinių bitų, kurie koduoja projekcijos kvadrantų seką, kaip parodyta paveikslėlyje [15 pav.], nustatymai kaupia informaciją, ar bangelės nesusikerta. Tai matyti iš ženklo operatoriaus sgn formulėje. Dar vienas fazės informacijos išgavimo privalumas yra tas, kad fazės kampai (angl. angles) yra priskiriami nekreipiant dėmesio į tai, koks žemas gali būti paveikslėlio kontrastas, kaip matome iš paveikslėlio [16 pav.], kuris yra labai prastai sufokusuotas. ([16 pav.] taip pat iliustruoja ir veikimą rainelės, vyzdžio bei akies voko radimo operatorių, nepaisant prasto sufokusavimo.) Nauda, jog fazės bitai yra nustatomi net ir prastai sufokusuotiems paveiksliukams, net jei tai pasiekiama remiantis atsitiktiniu triukšmu, yra tokia, kad skirtingos prastai sufokusuotos rainelės niekada nebus supainiotos viena su kita, kai bus lyginami jų fazių kodai. Priešingai, skirtingų veidų paveiksliukai atrodo vis labiau panašūs, kuo prasčiau sufokusuoti jų paveiksliukai ir gali būti supainioti tarpusavyje veidų atpažinimo algoritmų.



15 pav. Fazių-kvadrantų demoduliacijos kodas. Fazių demoduliacijos procesas naudojamas užkoduoti rainelėms. Rainelės sritys yra suprojektuotos ant kvadratūrinių 2D Gabor bangelių, generuojančių kompleksinius koeficientus, kurių realiosios ir menamosios dalys apibrėžia sinusinės bangos vaizdo (angl. phasor) koordinates kompleksinėje plokštumoje (rodyklė). Kiekvienos sinusinės bangos kampas yra priskiriamas vienam iš keturių kvadrantų, nustatant du bitus fazės informacijos. Šis procesas yra kartojamas per visą rainelę su daug bangelių dydžių, dažnių ir orientacijų, išgauti 2,048 bitus. [Daua]



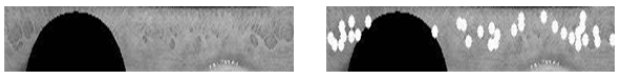
16 pav. Iliustracija, kad netgi prastai sufokusuotiems akies paveiksliukams, demoduliacijos fazinės sekos bitai yra vis vien nustatomi, iš pradžių atsitiktiniu triukšmu. Tai padeda išvengti to, kad prastai sufokusuoti akių paveikslėliai būtų panašūs vienas į kitą jų palyginimo stadijoje. Dėl tokių pat priežasčių prastos kokybės veido paveiksliukai taip pat atrodo panašūs, todėl gali būti supainioti vienas su kitu. [Daua]

2. 3. 3. Rainelės požymių išrinkimas kampų nustatymo būdu

Normalizuoto rainelės paveikslėlio kampai gali būti naudojami išrinkti požymius, pagal kuriuos bus galima atskirti du rainelės paveikslėlius [GMR+]. Kampų taškai gali būti nustatomi normalizuotame rainelės paveikslėlyje, naudojant kovariacijų matricą su intensyvumo pokyčiais kiekviename taške. 3x3 dydžio langas su centru taške p, naudojamas surasti kovariacijų matricą M_{cv}:

$$M_{cv} = \begin{bmatrix} \sum D_x^2 & \sum D_x D_y \\ \sum D_x D_y & \sum D_y^2 \end{bmatrix}. \qquad (60)$$

Čia D_x yra intensyvumų pokytis, einant per stulpelius, o D_y yra intensyvumų pokytis, einant per eilutes. Sumavimas atliekamas visam 3x3 langui. Jei M_{cv} tikrinė vertė yra didesnė nei slenkstis, ji laikoma kampu. Akies vokai prideda triukšmo normalizuotam rėžiui ir yra nustatomi naudojant mažiausių kvadratų aproksimavimo metodą. Šie kampai nustatomi po to, kai akies vokai jau pašalinti iš normalizuoto rainelės paveikslėlio. Kampų nustatymo rezultatas pavaizduotas paveikslėlyje [17 pav.]. Visi nustatyti kampai laikomi rainelės požymiais.



17 pav. Kampų nustatymas [GMR+]

2. 4. Rainelių palyginimas

2. 4. 1. Palyginimas pagal Hammingo atstumą

John Daugman savo straipsniuose aprašo rainelių palyginimo pagal Hammingo atstumą metodą [Dau06], [Daua], [Dau07]. Lyginant bet kuriuos du rainelių kodus, faziniams duomenų bitams atliekama operacija exclusive-OR ($\otimes$) nustatyti skirtingumą ir tokiu būdu įvertinti panašumą tarp tų dviejų rainelių. Taip pat atliekama AND ($\cap$) operacija maskavimo bitams su šia XOR kombinacija, siekiant apriboti palyginimą tiems bitams, kurie abejoms rainelėms yra neuždengti blakstienų, vokų ar atspindžių. Tuomet skaičiuojamos norminės ($\| \ \|$) reikšmės duomenų vektoriams, gautiems po operacijos XOR, ir maskavimo vektoriams, gautiems po operacijos AND, norint gauti neapdorotą (ang. raw) Hammingo atstumą HD_{raw} kaip dalį bitų (kurie laikomi reikšmingais), kurie nesutampa dviems lyginamoms rainelėms. Bet kurioms dviems skirtingoms rainelėms statistiškai tikėtina reikšmė yra $HD_{raw} = 0.5$. Jei pažymėtume dviejų rainelių fazių duomenų vektorius kaip {codeA, codeB} ir su jais susijusius maskavimo vektorius kaip {maskA, maskB}, tuomet jų neapdorotasis nepanašumo matavimas būtų toks:

$$HD_{raw} = \frac{\left\| (codeA \otimes codeB) \cap maskA \cap maskB \right\|}{\left\| maskA \cap maskB \right\|}. \qquad (61)$$

Tačiau skirtingi žmonės atidengia skirtingą plotą rainelės tarp akies vokų, taip pat matomas rainelės plotas priklauso nuo blakstienų uždengimo, atspindžių ir kitų aplinkybių. Dėl šios priežasties, bitų skaičius, kurį galima gauti palyginti dviejų rainelių kodams, yra kintantis. Artimas atitikimas (tarkim, Hammingo atstumas $HD_{raw} = 0.10$), kuris buvo apskaičiuotas remiantis tik keliais palygintais bitais, mažiau gali identifikuoti rainelę negu akivaizdžiai prastesnis atitikimas (tarkim, $HD_{raw} = 0.20$), kuris remiasi dideliu skaičiumi palygintų bitų. Todėl reikia perskaičiuoti (normalizuoti) bet kurią gautą neapdorotą Hammingo atstumo reikšmę

HD_{raw} į HD_{norm}, kurios nuokrypis nuo statistiškai tikėtinos reikšmės skirtingoms rainelėms $HD_{raw} = 0.5$ yra pakeičiamas, kad būtų atsižvelgiama į statistinį reikšmingumą, kuris remiasi bitų skaičiumi n, kurie iš tiesų buvo palyginti dviems rainelių kodams. Normalizuotoji reikšmė apskaičiuojama taip:

$$HD_{norm} = 0.5 - (0.5 - HD_{raw})\sqrt{\frac{n}{911}}. \qquad (62)$$

Lygties parametrai įtakoja standartinį nuokrypį normalizuoto Hammingo atstumo verčių pasiskirstymo ir suteikia šiam pasiskirstymui stabilią formą, kuri leidžia pradėti veikti stabilaus sprendimo taisyklei.

2. 4. 2. Palyginimo pagal kampus metodas

Rainelių atpažinimui, nustatomi visi kampai abiejų rainelių ir gaunamas sutampančių kampų skaičius. Du rainelės paveikslėliai vaizduoja to paties žmogaus rainelę, jeigu sutampančių kampų skaičius yra didesnis už tam tikrą nustatytą slenkstinę reikšmę. Palyginimo pagal kampus metodą [GMR+] sudaro du žingsniai:

1. Nustatyti kampai tarp duomenų bazės ir užklausos paveikslėlių yra naudojami surasti kryžminės koreliacijos koeficientą
2. Jei koreliacijos koeficientų skaičius tarp nustatytų minėtų dviejų paveikslėlių kampų yra didesnis už slenkstinę reikšmę, tuomet sistema priima kandidatą

2. 4. 2. 1. Kryžminė koreliacija tarp kampų

Tegu C_{ij} būna kryžminės koreliacijos koeficientas tarp dviejų kampų I_i ir J_i. Tuomet jei požymių sritys I_i ir J_i yra atvaizduojamos kaip du wxw masyvai A ir B atitinkamai, tai C_{ij} galima apskaičiuoti taip:

$$C_{ij} = \frac{\sum_{u=1}^{W}\sum_{v=1}^{W}(A_{uv} - A_{avg}).(B_{uv} - B_{avg})}{W^2.\sigma(A).\sigma(B)}. \qquad (63)$$

Čia A_{avg} yra srities aplink tašką I_i vidurkis, o $\sigma(A)$ yra jos standartinis nuokrypis. Du kampai laikomi koreliuotais, jei kryžminės koreliacijos koeficientas yra didesnis nei duotoji slenkstinė vertė.

2. 4. 2. 2. Rainelių verifikavimas, naudojant kampus

Tegu A ir B būna du rainelės paveikslėliai, kuriuos norime verifikuoti. Tegu P ir Q būna rinkinys kampų taškų, nustatytų paveikslėliuose A ir B. Kiekvienam taškui p iš P, paimkime m taškų (q_1……q_m) iš Q, kurie yra arčiau nei d Euklido atstumu nuo p. Tegu C_1…….C_m būna

kryžminės koreliacijos koeficientas tarp p ir (q_1……q_m). Jei maksimali reikšmė iš C_1…….C_m yra didesnė už slenkstį, tai p yra laikomas sutampančiu tarp A ir B.

Tegu I_1 ir I_2 būna to paties žmogaus rainelės paveikslėliai, o I_3 kitos rainelės paveikslėlis. M_1 pažymėkime skaičių taškų, kurie sutampa paveikslėliams I_1 ir I_2, o M_2 tegu būna skaičius krašto taškų, kurie sutampa paveikslėliams I_1 ir I_3.

Paveikslėliai I_1 ir I_3 laikomi priklausančiais tam pačiam asmeniui, jeigu Hamming‘o atstumas (angl. Hamming distance) tarp M_1 ir M_2 yra mažesnis už slenkstinę reikšmę:

$$\frac{M_1 - M_2}{M_2} < \psi . \qquad (64)$$

Čia Ψ yra slenkstis, nustatytas eksperimentiniu būdu.

Šį metodą įgyvendinanti sistema buvo ištestuota su dvejomis duomenų bazėmis, iš kurių vienoje buvo 900 paveikslėlių. Buvo nustatytas 95,4 % algoritmo tikslumas.

2. 5. Rainelių palyginimo tikslumas. Statistinė sprendimų teorija ir Neyman-Pearson kreivė

John Daugman statistinę sprendimų teoriją aprašo abstrakčiai [Dau00], [Dau93], nes ją galima pritaikyti bet kokiems taip/ne sprendimams įvertinti, todėl ji tinka ir rainelių palyginimo sprendimams.

Taip/ne atpažinimo sprendimai gali duoti keturių tipų rezultatus: ar modeliai sutampa, ar ne ir kiekvienu iš šių atvejų atpažinimo algoritmo padarytas sprendimas gali būti teisingas arba neteisingas. Biometrinių sprendimų kontekste šie keturių tipų rezultatai vadinami neteisingu priėmimu (angl. False Accept – FA), teisingu priėmimu (angl. Correct Accept – CA), neteisingu atmetimu (angl. False Reject – FR) ir teisingu atmetimu (angl. Correct Reject – CR). Pirmojo ir trečiojo tipo sprendimai yra klaidingi (atitinkamai vadinami pirmojo ir antrojo tipo klaidomis), o antrojo ir ketvirtojo tipo sprendimai yra laukiamieji. Svarbu nustatyti laipsnį, kuriuo bet koks sumažinimas vieno klaidos rodiklio yra apmokamas kitos klaidos rodiklio pabloginimu. Ši koncepcija yra naudinga vertinant atpažinimo sprendimų priėmimo patikimumą ir lyginant skirtingus biometrinius požiūrius bei jų pajėgumus tarpusavyje.

Paveikslėlis [18 pav.] vaizduoja sprendimų formalizmą. Matome, jog yra du pasiskirstymai, nepilnai atsiskyrę vienas nuo kito. Abscisė yra bet koks panašumo ar nepanašumo matas. Šiuo atveju tai yra Hammingo atstumas (bitų dalis, kuri skiriasi dviems binarinėms eilutėms). Panašumo kriterijus, kuris apsprendžia, ar tai to paties objekto (rainelės) modeliai, ar skirtingų objektų, pažymėtas taškine linija. Panašumas iki tam tikro Hammingo atstumo (šiuo atveju 0.4) laikomas pakankamu modelius laikyti to paties objekto, o toliau už to taško modeliai laikomi skirtingais. Kitame John Daugman straipsnyje, kuris remiasi realiais

statistiniais duomenimis [Daub], visuose tyrimuose naudojamas Hammingo atstumas 0.32, kuris reiškia, kad rainelių kodai laikomi sutampančiais, jei ne daugiau kaip 32% jų bitų nesutampa.

Tikėtinumus, ar tai yra teisingi sprendimai, ar ne atitinka keturi užbrūkšniuoti plotai. Suprantama, kad sprendimo kriterijaus pastūmimas į dešinę arba į kairę (žiūrint liberaliau ar konservatyviau) keičia keturių sprendimų tipų tikėtinumą. Problema yra labiau išsprendžiama ir sprendimai patikimesni, jei dvi sritys abejose kriterijaus pusėse mažiau persidengia. Vienas, bet ne vienintelis, išsprendžiamumo matas yra d‘ (d – pirminis):

$$d' = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{1}{2}(\sigma_1^2 + \sigma_2^2)}}. \qquad (65)$$

Čia μ_1 ir μ_2 yra dviejų pasiskirstymų vidurkiai, o σ_1 ir σ_2 - jų standartiniai nuokrypiai. d‘ viena skaitine reikšme nusako suderinamumą dviejų klaidų tipų rodiklių. Jis matuoja atsiskyrimą dviejų pasiskirstymų [18 pav.], todėl kuo jis didesnis, tuo geriau. Paveikslėlyje [18 pav.] d‘=2.

Pažymėkime du pasiskirstymus $P_{\mathrm{Im}}(x)$ ir $P_{Au}(x)$, kurie atitinkamai reiškia tikimybę išmatuoto nepanašumo x (tokio kaip Hammingo atstumas) dviems skirtingiems biometriniams šaltiniams ([18 pav.] „Impositors“) arba vienam ir tam pačiam šaltiniui ([18 pav.] „Authentics“). Tuomet kiekvieno sprendimo tipo FA, CR, CA ir FR tikimybės yra lygios sritims tuose dviejuose tikimybių pasiskirstymuose abejose kriterijaus C pusėse:

$$P(FA) = \int_0^C P_{\mathrm{Im}}(x)dx, \qquad (66)$$

$$P(CR) = \int_C^1 P_{\mathrm{Im}}(x)dx, \qquad (67)$$

$$P(CA) = \int_0^C P_{Au}(x)dx, \qquad (68)$$

$$P(FR) = \int_C^1 P_{Au}(x)dx. \qquad (69)$$

Šioms tikimybėms galioja nelygybės:

$$P(CA) + P(FR) = 1, \qquad (70)$$

$$P(FA) + P(CR) = 1, \qquad (71)$$

$$P(CA) > P(FA), \qquad (72)$$

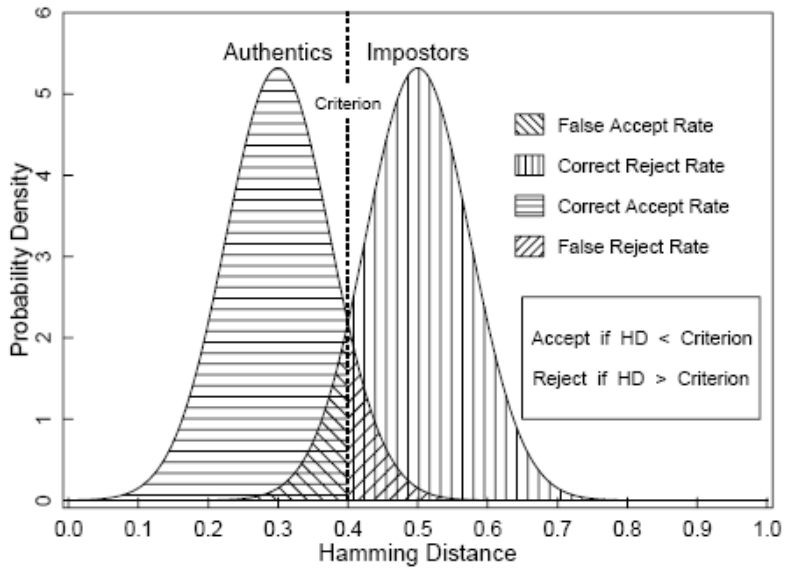
$$P(CR) > P(FR). \qquad (73)$$

Manipuliavimas kriterijumi C integraluose (66) – (69), siekiant įgyvendinti skirtingas sprendimų strategijas su atitinkamomis abiejų tipų klaidų kainomis, schematiškai pavaizduotas paveikslėlyje [19 pav.]. Tokia sprendimų strategijos diagrama, kartais vadinama Gavėjo

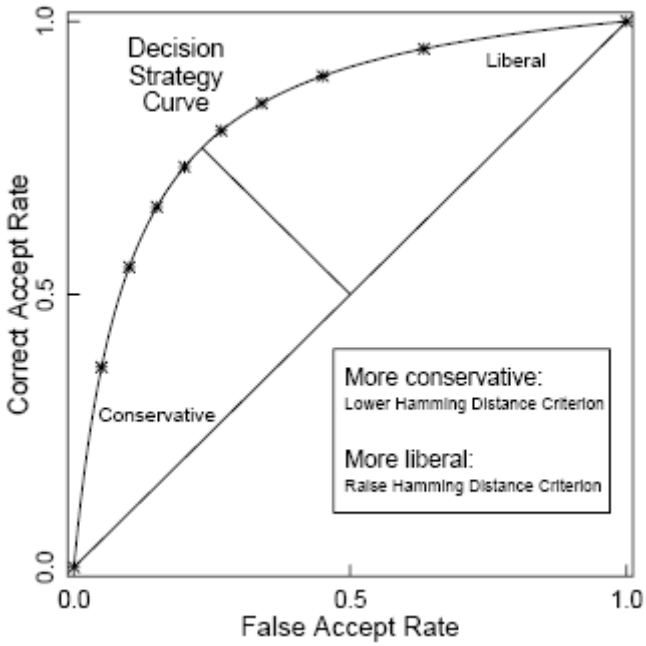
operavimo charakteristika (angl. Receiver Operating Characteristic) arba Neyman-Pearson kreive, žymi P(CA) (68) ir P(FA) (66) kaip taškų rinkinį. Kiekvienas taškas tokioje kreivėje vaizduoja tam tikrą sprendimų strategiją su skirtingu kriterijaus C pasirinkimu.

Nelygybė (72) reiškia, kad Neyman-Pearson strategijos kreivė [19 pav.] visada bus įstrižos linijos viršuje. Strategija gali būti pasirenkama liberalesnė ar konservatyvesnė slankiojant per kreivę. Atpažinimo metodo patikimumas gali būti nustatomas pagal tai, kiek išlinkusi yra ROC kreivė. Trumposios linijos paveikslėlyje [19 pav.] ilgis yra monotoniškai susietas su dydžiu d‘ (65). Idealiu atveju biometrija generuotų tokius sprendimus, kad ROC kreivė būtų ekstremaliai išlinkusi ir siektų taip toli, kiek tik įmanoma viršutiniame kairiajame kampe paveikslėlyje [19 pav.], kadangi tos ribos pasiekimas atitinka tai, jog teisingo priėmimo (CA) matavimas būtų 100%, o neteisingo priėmimo matavimas (FA) būtų 0%.

Statistinės sprendimų teorijos praktiniai taikymai rainelių atpažinimo algoritmams įvertinti aprašyti John Daugman straipsnyje [Daub]. Ten pateiktos realių statistinių duomenų kreivės (taip pat ir ROC).



18 pav. Statistinė sprendimų teorija. Bendras formalizmas biometriniams sprendimams daryti. [Dau00]



19 pav. Neyman-Pearson (ROC) sprendimų strategijos kreivė. [Dau00]

3. Praktinė dalis. Metodai ir tyrimai.

3. 1. Rainelės segmentacija, remiantis vandens takoskyros transformacija ir momentiniais invariantais

3. 1. 1. Algoritmas

Mano sukurtas akių rainelių segmentavimo algoritmas remiasi vandens takoskyros transformacija ir momentinių invariantų skaičiavimu. Jis susideda iš keleto žingsnių:

- Jei paveikslėlis, kurį norime segmentuoti, yra spalvotas, tai jis paverčiamas nespalvotu, kadangi algoritmas tinka tik nespalvotiems paveikslėliams: vandens takoskyros transformacijos metu sritys plečiasi pagal pilkumo lygių reikšmes.
- Sukuriamas gradientinis paveikslėlis tam, kad išryškėtų objektų kraštai paveikslėlyje. Kuo ryškesni vyzdžio ir rainelės kraštai, tuo geriau pavyksta segmentacija.
- Neautomatinis arba automatinis segmentacijos žymių (taškų) parinkimas ir sužymėjimas.
- Atliekama vyzdžio ir rainelės segmentacija. Vandens takoskyros transformacijos metu surandamos ir skirtingomis spalvomis paveikslėlyje nuspalvojamos trys sritys: vyzdys, rainelė ir fonas. Segmentacija remiasi automatiniu arba rankiniu būdu sužymėtais taškais. Segmentacijos metu gautoms sritims apskaičiuojami momentai. Iš jų gaunami centriniai momentai, o iš centrinių momentų –

momentiniai invariantai. Invariantais remiasi automatinis taškų parinkimas. Pagal invariantus nustatoma, ar jau yra gautos reikiamos sritys, ar reikia parinkti kitus taškus ir segmentuoti iš naujo.

- Pagal momentus piešiami vyzdžio ir rainelės sričių apskritimai.
- Rezultatai išsaugomi rinkmenose.

Automatinio taškų parinkimo atveju trečias ir ketvirtas algoritmo žingsniai kartojami iteratyviškai, kol surandamos reikiamos sritys paveikslėlyje (vyzdžio, rainelės ir fono sritys) arba kol nepasibaigia nustatytas žingsnių kartojimo limitas.

3. 1. 1. 1. Spalvoto paveikslėlio pavertimas į nespalvotą (pilkumo lygių) paveikslėlį

Nespalvoto paveikslėlio vaizdo elementų pilkumo lygių reikšmės apskaičiuojamos pagal formulę [Wikc]

$$grayLevel = element.R * 0.3 + element.G * 0.59 + elementB * 0.11, \quad (74)$$

kur element.R, element.G ir element.B yra spalvoto paveikslėlio i-tojo (i = 1, 2, 3,..., N, N – paveikslėlio vaizdo elementų skaičius) vaizdo elemento raudonoji, žalioji ir mėlynoji spalvos komponentės. Iš gautų kiekvienam vaizdo elementui pilkumo lygių reikšmių intervale [0; 255] sudaromas naujas, jau nespalvotas, paveikslėlis.

3. 1. 1. 2. Gradientinio paveikslėlio sukūrimas

Kiekvienam vaizdo elementui, patikrinant jo ir aštuonių jo kaimynų pilkumo lygių reikšmes nespalvotame paveikslėlyje (toliau jį vadinsiu pradiniu paveikslėliu), išrenkama mažiausia ir didžiausia pilkumo lygių reikšmės. Tuomet to vaizdo elemento pilkumo reikšmė naujame, gradientiniame, paveikslėlyje bus apskaičiuota pagal formulę

$$grayLevel = \max - \min . \quad (75)$$

Gradientinis paveikslėlis toliau naudojamas segmentacijoje.

3. 1. 1. 3. Rankinis taškų parinkimas

Rankinis taškų parinkimas – tai toks taškų parinkimas, kai juos parenka ir sužymi ne pati programa, o programos naudotojas, pvz., sužymi taškus pele (kairiuoju klavišu) ant paveikslėlio programos lange. Mano sukurtame algoritme reikia parinkti ir sužymėti taškus trims sritims: vyzdžiui, rainelei ir fonui. Taškų skaičius kiekvienai sričiai rankinio parinkimo atveju yra neribotas. Baigus žymėti kurios nors srities taškus, reikia pranešti programai, kad pereinama prie kitos srities taškų žymėjimo, pvz., paspausti dešiniuoju pelės klavišu.

Segmentacijoje naudojami ne patys parinktieji taškai, bet jų lokalieji minimumai. Lokalieji minimumai skaičiuojami taip [Algoritmas 1]: tarp parinkto taško vaizdo elemento ir jo kaimynų (keturių arba aštuonių, mano programoje – keturi, *Algoritme 1* jų skaičius pažymėtas *NO*) ieškoma vaizdo elemento su mažiausia pilkumo lygio reikšme. Išrinktam kaimynui su mažiausia pilkumo lygio reikšme tikrinami jo kaimynai, ieškant vaizdo elementų su dar mažesne pilkumo lygio reikšme. Taip ciklas kartojamas, kol kurioje nors iteracijoje tarp kaimynų nerandama nei vieno vaizdo elemento su mažesne pilkumo lygio reikšme negu centrinio vaizdo elemento reikšmė. Taigi surastas parinkto taško lokalusis minimumas ir naudojamas toliau segmentacijoje kaip srities žymė.

Algoritmas 1: Lokaliojo minimumo paieška

Įvestis: vaizdo elemento, kuriam ieškoma lokalaus minimumo, indeksas $xy = x + W * y$, kur x ir y – vaizdo elemento koordinatės
Išvestis: vaizdo elemento lokaliojo minimumo indeksas

```
1:  while (true)
2:     ij ← xy
3:     for n = 1 to NO do
4:        if not coloredPixel[ index_n ] and grayLevel[ index_n ] < grayLevel[ij] then
5:           ij ← index_n
6:        end if
7:     end for
8:     if (ij = xy) then
9:        break;
10:    end if
11:    xy ← ij
12: end while
13: return xy
```

Čia *coloredPixel* – loginių reikšmių masyvas, nusakantis, kurie vaizdo elementai jau yra priskirti kuriai nors sričiai (nuspalvoti). $index_n$ - kaimyninio vaizdo elemento indeksas. *grayLevel* – vaizdo elementų pilkumo lygių reikšmių masyvas.

3. 1. 1. 4. Automatinis taškų parinkimas

Vyzdžiui parenkamas vienas taškas, nes jo pakanka, norint surasti vyzdžio sritį (tai nustačiau eksperimentiniu būdu). Ieškant vyzdžio, reikia atsižvelgti į paveikslio centrą, kadangi didelė tikimybė, kad vyzdys bus paveikslio centre arba netoli jo, o ne paveikslio krašte (tokiu atveju būtų nekokybiška akies nuotrauka).

Paprasčiausias būdas ieškant vyzdžio taško atsižvelgti į paveikslio centrą yra parinkti vyzdžio tašką paveikslio centre. Jei po segmentacijos pasirodo, kad vyzdžio sritis nerasta, tai parinkti kitą tašką tam tikru atstumu nuo paveikslio centro. Ir taip eiti per paveikslį tolyn

nuo paveiksliuko centro tam tikra nustatyta tvarka (pvz., vis didesnio virtualaus jau prieš tai parinktus taškus gaubiančio kvadrato kraštinių taškais). Tokiu būdu pirmiausia savo programoje ieškojau vyzdžio taškų. Šis metodas gerai veikė tais atvejais, kai vyzdys iš tiesų yra paveiksliuko centre arba gana arti jo, bet tikrinant algoritmą, buvo nemažai paveiksliukų, kuriuose vyzdžio ieškojo labai ilgai, nes vyzdys buvo nemažu atstumu nuo paveiksliuko centro. Šį metodą naudojau ankstesnėse savo programos versijose, o paskui jį pakeičiau kitu (žemiau aprašytu) metodu.

Kitas metodas ieškoti vyzdžio taško remiasi pilkumo reikšmių minimumais ir paveiksliuke nustatomu virtualiu langu. Pagal mano sukurtą algoritmą, pradiniame (ne gradientiniame) paveiksliuke reikia ieškoti minimalios pilkumo lygio reikšmės. Jos ieškant, atsižvelgiama į tai, kaip toli vaizdo elementas yra nuo paveiksliuko centro. Paveiksliuke nustatomas langas, kurio kairiojo viršutinio kampo centrinio taško koordinatės yra {W / 4, H / 4}, o dešiniojo apatinio kampo centrinio taško koordinatės yra {W * 3 / 4, H * 3 / 4}, kur W – paveiksliuko plotis, o H – paveiksliuko aukštis. Į šį langą patenkantiems vaizdo elementams, ieškant minimalios pilkumo lygio reikšmės paveikslėlyje, imamos jų realios pilkumų lygių reikšmės, o tiems vaizdo elementams, kurie nepatenka į langą, prie jų pilkumo lygių reikšmių pridedamos „baudos“, kurių dydis priklauso nuo to, kaip toli vaizdo elementas yra nuo nustatyto lango kraštų (o taip pat ir nuo paveiksliuko centro). „Baudų“ pridėjimo algoritmo pseudo kodas pateiktas žemiau [Algoritmas 2]. Čia lango kairiojo viršutinio kampo centrinio taško koordinačių komponentės yra pažymėtos *windowMinW* ir *windowMinH*, o dešiniojo apatinio kampo – *windowMaxW* ir *windowMaxH*. *gValue* pažymėta tikrinamo vaizdo elemento realioji (be „baudų“) pilkumo lygio reikšmė, o *constValue* – iš anksto nustatyta konstanta, svoris, įtakojantis baudos dydį (pvz., 5). Pridėjus „baudas“, programa įgyvendinanti automatinį taškų parinkimą ir segmentaciją, pirmiausiai ieško vyzdžio taškų arčiau paveiksliuko centro ir mažiau atsižvelgia į mažą pilkumo reikšmę turinčius vaizdo elementus paveiksliuko kraštuose, kur labai maža tikimybė, kad gali būti vyzdys. Programa „įsimena“ vaizdo elementus, kuriuos laikė vyzdžio srities žymėmis ir su kuriais jau ieškojo vyzdžio bei tuos vaizdo elementus, kurie pateko į išsiplėtusią sritį, ir, kol vyzdys nerastas, parinkdama naujas žymes šių taškų pakartotinai nebetikrina.

Algoritmas 2: „Baudų“ pridėjimas už nustatyto lango ribų esantiems paveikslėlio vaizdo elementams

Įvestis: tikrinamo vaizdo elemento koordinatės *{x, y}*
Išvestis: vaizdo elemento pilkumo reikšmė *grayValue* su „bauda“ arba be jos

```
1:  if (y < windowMinH) then
2:      grayValue ← gValue + (windowMinH – y) * constValue
3:  else if (y > windowMaxH) then
4:          grayValue ← gValue + (y - windowMaxH) * constValue
5:      else if (x < windowMinW) then
```

```
6:                grayValue ← gValue + (windowMinW – x) * constValue
7:             else if (x > windowMaxW) then
8:                grayValue ← gValue + (x - windowMaxW) * constValue
9:             else
10:               grayValue ← gValue
11:            end if
12:         end if
13:      end if
14:   end if
15:   return grayValue
```

Vaizdo elemento su minimalia pilkumo lygio reikšme lokalusis minimumas (lokaliojo minimumo nustatymas yra toks pat kaip rankinio taškų parinkimo atveju [3. 1. 1. 3.]) laikomas vyzdžio tašku (žyme) ir atliekama vandens takoskyros transformacija, kurios metu paveiksliukas segmentuojamas į dvi sritis: vyzdį ir foną. Transformacija pradedama su minimaliu slenksčiu, kuris gaunamas prie surasto vaizdo elemento su minimalia pilkumo lygio reikšme pilkumo reikšmės pridėjus nedidelę konstantą (pvz., 16) ir tęsiama vis didinant slenkstį iki reikšmės 255. Kai sritys plečiasi, reikia tikrinti, ar vyzdžio sritis neišsiplėtė per visą paveiksliuką. Tai galima patikrinti, pažiūrėjus ar vyzdžio srities spalva nenuspalvintas koks nors konkretus taškas arti paveikslėlio krašto ir/arba pagal vyzdžio srities dydį.

Po transformacijos apskaičiuojami vyzdžio momentiniai invariantai ir pagal juos patikrinama, ar surasta sritis yra apskritimas (arba artimas apskritimui objektas). Skaičiuojami du pirmieji momentiniai invariantai [2. 1. 12.] (52), (53). Apskritimo pirmasis geometrinis invariantas turi būti lygus $1/(2\pi)$, o antrasis turi būti lygus nuliui (keturių skaitmenų po kablelio tikslumu). Kadangi vyzdžio sritis dažnai yra elipsė artima apskritimui (jos vertikalusis ir horizontalusis skersmenys nedaug skiriasi), o ne geometrinis apskritimas ir kadangi išskirtos (nuspalvotos) vyzdžio srities kraštuose gali būti nelygumų, tai prie geometrinių invariantų reikšmių reikia pridėti tam tikras eksperimentiniu būdu nustatyto dydžio konstantas. Dėl to pirmasis vyzdžio srities geometrinis invariantas ϕ_1 turi tenkinti sąlygą $\phi_1 \in [0{,}159\,;0{,}162)$, o antrasis geometrinis invariantas ϕ_2 turi tenkinti sąlygą $\phi_2 \in [0\,;0{,}001)$. Jei geometriniai invariantai tenkina šias sąlygas (pvz., g1= 0,1594, g2= 0,0000) ir jei vyzdžio srities plotas m_{00} ne didesnis ir ne mažesnis nei tam tikros nustatytos reikšmės, tai laikome, kad parinktas taškas yra ant vyzdžio ir kad suradome vyzdžio sritį. Mano programoje maksimalus vyzdžio srities plotas nustatytas W * H / 2, kur W ir H yra akies paveiksliuko plotis ir aukštis atitinkamai. Minimalus vyzdžio srities plotas nustatytas 40, kai paveikslėlis programoje yra du kartus sumažintas ir 650, kai paveikslėlis nesumažintas. Minimalų vyzdžio plotą reikia tikrinti dėl to, kad kartais blakstienose būna apskritimo formos juodų dėmių, kurias programa galėtų palaikyti

vyzdžiu. Suradus vyzdžio žymę ir sritį, apskaičiuojamas vyzdžio centras (pagal formulę (48)) ir jo lokalusis minimumas naudojamas kaip pradinis vyzdžio srities taškas, ieškant rainelės srities. Jei nuspalvinus vyzdžio sritį, vaizdo elementai, abiejose centro pusėse nutolę pusės spindulio atstumu nuo centro, liko nespalvoti, tai jų lokalūs minimumai parenkami kaip dar dvi vyzdžio žymės tam, kad vyzdžio sritis būtų pilnai nuspalvota dar prieš ieškant rainelės srities. Tiesa, toks atvejis pasitaiko labai retai.

Jei geometriniai invariantai netenkina minėtų sąlygų, tai reikia ieškoti kito taško, kuris galbūt jau bus vyzdžio taškas. Programoje galima nustatyti vyzdžio taško ieškojimo iteracijų skaičių. Jei, tarkim, po 20 iteracijų programa neranda vyzdžio taško, tai nebeieško, o išrenka geriausią iki tol rastą vyzdžio srities variantą ir pradeda ieškoti rainelės srities. Tokią ar kitokią pabaigos sąlygą reikia nustatyti tam, kad nereikėtų ilgai laukti rezultatų, jei programa ilgai neranda vyzdžio (pvz., kai akies paveikslėlyje yra ryškūs antakiai ar blakstienos, kurie duoda daug mažų pilkumo lygių reikšmių). Geriausiu vyzdžio srities variantu laikoma ta vyzdžio sritis, kuri yra artimiausia apskritimui, tai yra, kurios pirmieji du momentiniai invariantai tenkina sąlygas $\min(|\phi_{1\,real} - \phi_{1\,ideal}|_i)$ ir $\min(|\phi_{2\,real} - \phi_{2\,ideal}|_i)$ (čia $\phi_{1\,real}$ ir $\phi_{2\,real}$ yra apskaičiuoti vyzdžio srities realūs momentiniai invariantai, o $\phi_{1\,ideal}$ ir $\phi_{2\,ideal}$ yra atitinkami apskritimo momentiniai invariantai, i= 1, 2, ...).

Rainelei parenkami du taškai pagal vyzdžio centrą ir spindulį. Vyzdžio spindulys apskaičiuojamas pagal formulę

$$pupilR = \sqrt{m_{00} / \pi}\,. \quad (76)$$

Vienas taškas parenkamas vyzdžio kairėje, o kitas dešinėje pusėje (jei jis dar nebuvo priskirtas rainelės sričiai, vykstant vandens takoskyros transformacijai). Jie yra tam tikru nustatytu atstumu nutolę nuo vyzdžio. Rainelės taškų ieškoma iteratyviškai, t. y., jei su nustatytu atstumu nutolusiais nuo vyzdžio taškais rainelės sritis nerasta, tai parenkamas kitas (didesnis už prieš tai buvusį) atstumas nuo vyzdžio ir ieškoma naujų pradinių taškų. Tam, kad rainelės taškų neieškotų per toli paveikslėlyje, reikia nustatyti tinkamą atstumą nuo vyzdžio kiekvienoje iteracijoje ir iteracijų skaičių. Mano programoje yra nustatytos 5 iteracijos, pirmojoje iteracijoje atstumas nuo vyzdžio krašto yra 15, o kiekvienoje kitoje iteracijoje prie atstumo pridedama 3. Plečiantis sritims reikia tikrinti, ar rainelės sritis neišsiplėtė per visą paveikslėlį. Mano programoje rainelei nustatytas toks pat maksimalus plotas kaip ir vyzdžiui. Segmentacijoje naudojami ne patys parinkti taškai, o jų lokalūs minimumai (kaip ir vyzdžio atveju).

Fonui parenkami keturi taškai: po tašką kiekviename paveikslėlio krašte. Taškai parenkami su tam tikromis iš anksto nustatytomis koordinatėmis. Pavyzdžiui, mano programoje parenkami taškai {30, H – 30}, {W – 30, H – 30}, {30, 30}, {W – 30, 30} (W ir H – paveikslėlio plotis ir

aukštis atitinkamai). Taškas parenkamas, jei jis dar nebuvo priskirtas fono sričiai vandens takoskyros transformacijos metu. Segmentacijoje kaip srities žymė naudojamas ne pats šis taškas, o jo lokalus minimumas (kaip ir vyzdžio bei rainelės atvejais). Tie patys taškai fonui parenkami tiek ieškant vyzdžio, tiek rainelės. Suradus rainelę, fonas taip pat jau būna surastas ir segmentacija laikoma baigta.

3. 1. 1. 5. Vyzdžio ir rainelės segmentacija. Sričių plėtimasis

Segmentacija į dvi sritis: vyzdį ir foną, vyksta jau tada, kai automatiškai ieškoma vyzdžio taškų [3. 1. 1. 4.]. Kai automatiškai ieškoma rainelės taškų [3. 1. 1. 4.], segmentacija vykdoma iš naujo, atsižvelgiant į vyzdžio paieškos rezultatus, ir jai pasibaigus, gaunamas rezultatas – vyzdžio, rainelės ir fono sritys. Rankinio taškų žymėjimo atveju [3. 1. 1. 3.], segmentuojama iš karto į tris minėtas sritis jau taškų žymėjimo metu. Sužymėjus taškus, segmentacija vyksta toliau, kol pilkumo lygio slenkstis pakyla iki 255. Sritys pradeda plėstis nuo sužymėtų taškų tokiu principu: sričiai priskiriamas vaizdo elementas, jeigu jo pilkumo lygmens reikšmė neviršija reikšmės, kurią turi plėtimąsi inicijavęs vaizdo elementas ir jeigu jis yra kaimyninis vaizdo elementas tos srities, į kurią pretenduoja patekti, vaizdo elementams.

Algoritmas3, *Algoritmas4* ir *Algoritmas5* (pateikti žemiau) – tai pseudo-kodas sričių plėtimusi aprašyti. Šiuos algoritmus naudojau trijų didžiųjų sričių (vyzdžio, rainelės ir fono) plėtimusi savo programoje. Į šiuos algoritmus neįtraukiau mažųjų sričių (pvz., sritys aplink kiekvieną pažymėtą tašką rainelėje) plėtimosi. Šiuose algoritmuose *front, last* masyvai bei *frontFirst* ir *frontLast* kintamieji atlieka steko funkciją. Į šiuos masyvus vaizdo elementai dedami LIFO (angl. „last in first out“) principu: dedami į steko galą, o imami iš pradžios. Masyve *gr* saugomos vaizdo elementų pilkumo lygių reikšmės. *W* – akies paveiksliuko plotis. *NO* – kaimyninių vaizdo elementų skaičius (savo programoje naudojau 4). $index_n$ - n– tojo kaimyninio vaizdo elemento indeksas. *coloredPixel* – loginių reikšmių masyvas, nusakantis, kurie vaizdo elementai jau yra priskirti kuriai nors sričiai (nuspalvoti). Vaizdo elementai, kurie šiame masyve turi teigiamą loginę reikšmę, vėliau plečiant sritis jau nebetikrinami. *threshold* – einamasis pilkumo lygio slenkstis, iki kurio plečiasi sritys. *regionNr* masyve saugomi numeriai, kuriai sričiai priklauso kiekvienas paveiksliuko vaizdo elementas. *xyInd* masyve saugoma, kuris vaizdo elementas (vaizdo elemento indeksas) kokią pilkumo lygio reikšmę turi. *neighbourRegion* funkcija patikrina, ar visi kaimyniniai vaizdo elementai priklauso tai pačiai sričiai. Jei taip, tai grąžina tos srities numerį (0 – vyzdžio sritis, 1 – rainelės sritis, 2 – fono sritis). Jei visi kaimyniniai vaizdo elementai nepriklauso tai pačiai sričiai, tai ši funkcija grąžina neigiamą skaičių -2.

Algoritmas 3: Sričių plėtimosi vandens takoskyros principu pagrindinis algoritmas. Funkcija, iškviečiama vaizdo elementui, nuo kurio pradeda plėstis sritis.

Įvestis: vaizdo elemento, nuo kurio pradeda plėstis sritis, indeksas $xy = x + W * y$, kur x ir y – vaizdo elemento koordinatės

```
1:    front[frontLast++] ← xy
2:    Algoritmas4()
3:    if (gr[xy] > threshold) then
4:       Algoritmas5(gr[xy])
```

Algoritmas 4: Sričių plėtimasis, spalvojant sritims priskirtus taškus.

```
1:  while (frontFirst <> frontLast)
2:     xy ← front[frontFirst++]
3:     y ← xy / W
4:     x ← xy – W * y
5 :    Spalvina vaizdo elementą su koordinatėmis {x, y} spalva srities, kuriai jis priklauso.
6:     for n = 1 to NO do
7:        if not coloredPixel[ index_n ] then
8:           ij ← index_n
9:           g ← gr [ij]
10:          coloredPixel[ij] ← true
11:          if (g <= threshold) then
12:             if (neighbourRegion(ij) = regionNr[xy]) then
13:                front[frontLast++] ← ij
14:                regionNr[ij] ← regionNr[xy]
15:             else
16:                nuspalvinti vaizdo elementą {ij % W, ij / W} srities krašto spalva
17:             end if
18:          else
19:             xyInd[g][last[g]++] ← ij
20:          end if
21:       end if
22:    end for
```

Algoritmas 5: Pilkumo lygio slenksčio didinimas, plečiantis sritims.

Įvestis: *highThreshold* - slenkstis, iki kurio reikia padidinti einamuoju momentu esamą slenksčio reikšmę *threshold*

```
1:  t ← threshold
2:  while (t < highThreshold)
3:     threshold ← t + 1
4:     while (first[t] <> last[t])
5:        xy ← xyInd[t][first[t]++]
6:        coloredPixel[xy] ← true
7:        nr ← neighbourRegion(xy)
8:        if (nr >= 0) then
9:           regionNr[xy] ← nr
10:          Algoritmas3(xy)
11:       else
12:          nuspalvoti vaizdo elementą {xy % W, xy / W} srities krašto spalva
13:       end if
14:    end while
15:    t ← threshold
```

16: **end while**

3. 1. 1. 6. Momentų, centrinių momentų ir momentinių invariantų skaičiavimas

Mano sukurtoje programoje momentai skaičiuojami pagal skyrelyje [2. 1. 12.] pateiktą formulę (47), segmentuotų sričių centrai – pagal formulę (48), centriniai momentai – pagal formules (51), o momentiniai invariantai – pagal formules (52) ir (53). Formulėse f_{ij} įgyja reikšmę 1, jei vaizdo elementas su koordinatėmis {x, y} patenka į segmentuotą sritį, priešingu atveju f_{ij} įgyja reikšmę 0.

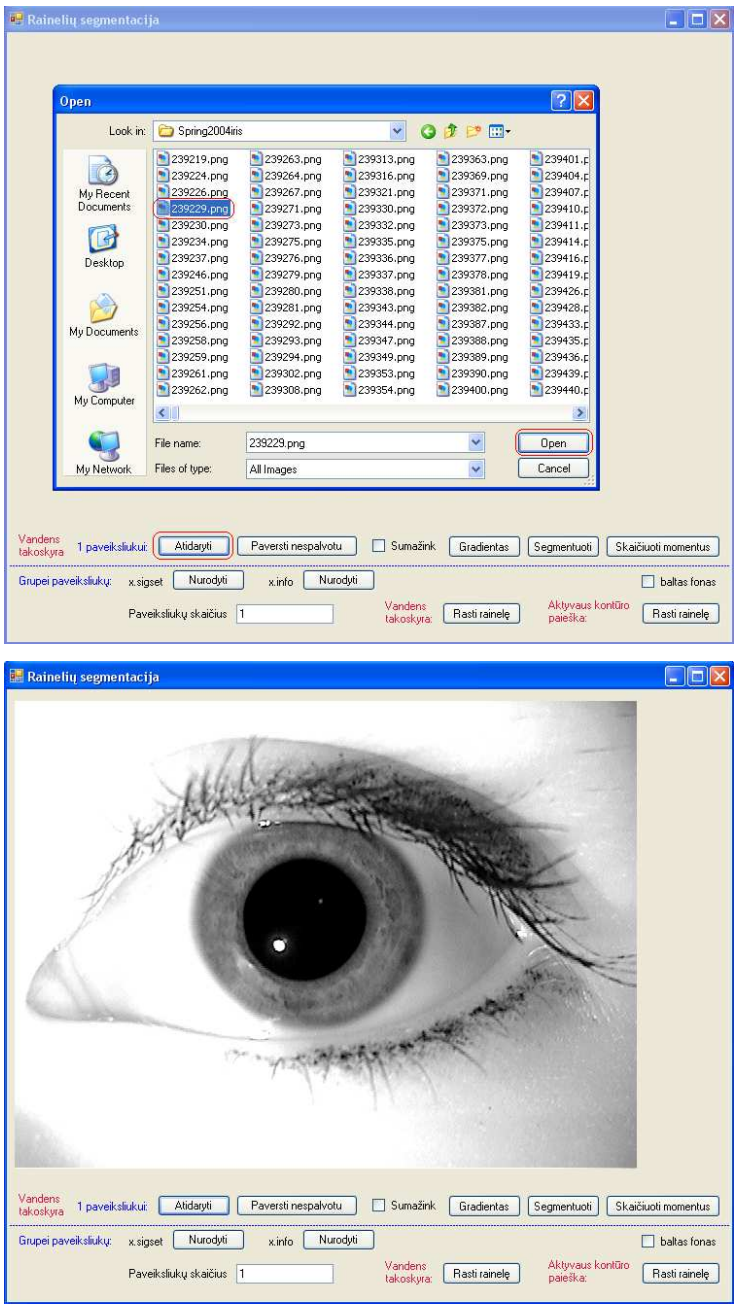
3. 1. 1. 7. Vyzdžio ir rainelės apskritimų piešimas

Vyzdžio ir rainelės apskritimams nupiešti reikia apskaičiuoti jų centrus ir spindulius. Vyzdžio ir rainelės centrai apskaičiuojamai pagal formulę (48), o spinduliai – pagal formulę (76). Vyzdžio ir rainelės apskritimus reikia nupiešti tam, kad galėtume turėti tikslesnes sritis. Pvz., kartais rainelės sritis apima ir stiklakūnį, o nupiešus rainelės apskritimą, patikslinama jos vieta paveikslėlyje. Žinoma, rainelės apskritimas - tai dar nėra rainelės kontūras, nes jis dažnai apima ir dalį akies voko bei blakstienų. Tačiau suradę šį apskritimą (be jokių apribojimų paveikslėliams), jau turime rainelės sritį (kad ir kiek per didelę), kurioje taikydami kitus metodus, galime tikslinti rainelės poziciją.

3. 1. 2. Programinis įrankis

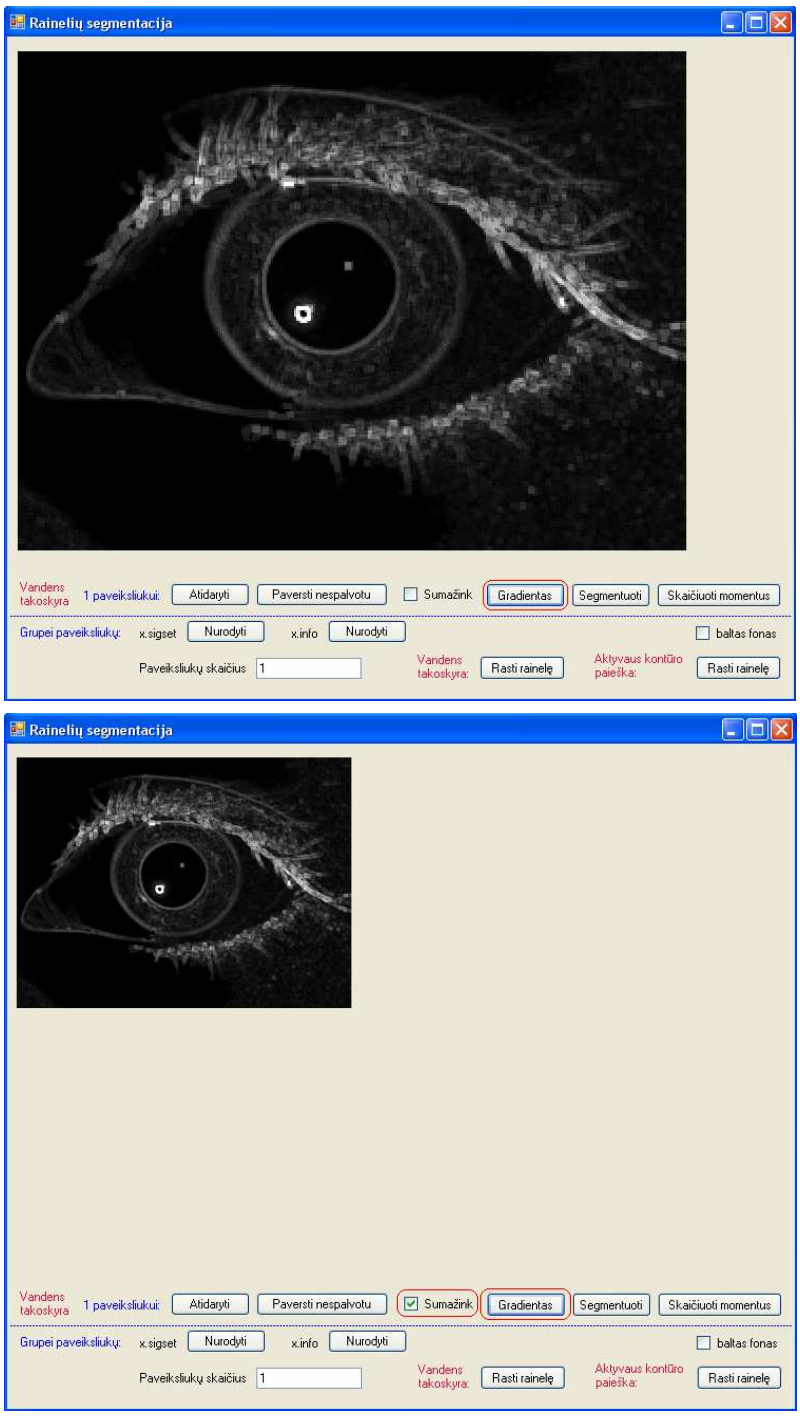
Rainelių segmentacijos programą sukūriau su Microsoft Visual Studio. Visual Studio pasirinkau todėl, kad ji įgalina patogų programinio įrankio naudotojo sąsajos kūrimą. Programavau C# kalba.

Atidarius mano sukurtą rainelių segmentacijos programą, pirmiausia reikia pateikti programai akies paveikslėlį, kurį norėsime segmentuoti. Reikia spausti mygtuką „Atidaryti“, atsidariusiame paieškos lange išsirinkti paveikslėlį savo kompiuteryje ir spausti mygtuką „Open“ [20 pav.]. Tuomet pasirinktas paveikslėlis bus parodytas programos lange ir programa pasiruoš segmentacijai. Mano sukurtas programinis įrankis dirba su jpg, gif, bmp, png ir tiff formato paveikslėliais. Segmentacijos algoritmas yra pritaikytas tik nespalvotiems paveikslėliams (vandens takoskyros transformacijos metu sritys plečiasi pagal pilkumo lygių reikšmes), todėl jeigu parinktas paveikslėlis yra spalvotas, reikia spausti mygtuką „Paversti nespalvotu“ ir iš spalvoto paveikslėlio bus padarytas nespalvotas (pilkumo lygmenų) paveikslėlis.



20 pav. Paveikslėlio parinkimas programoje. Viršuje pavaizduoti programos pagrindinis ir paveikslėlio parinkimo dialogo langai, o apačioje – segmentacijai parinktas paveikslėlis. Ką reikia spausti, norint atidaryti paveikslėlį, šioje ir tolesnėse iliustracijose apvesta raudonai.

Prieš atliekant segmentaciją, iš pradinio programai pateikto paveikslėlio reikia padaryti gradientinį. Tai programa atlieka, paspaudus mygtuką „Gradientas“. Jeigu paveikslėlis yra didelis, prieš tai dar reikia pažymėti „Sumažinti“ ir jis bus du kartus sumažintas [21 pav.]. Gradientą skaičiuoti reikės ir tolesnius veiksmus programa atliks jau sumažintam paveikslėliui.



21 pav. Gradientinio paveikslėlio sukūrimas iš pradinio paveikslėlio, kai paveikslėlio dydis nepakeistas (viršuje) ir kai paveikslėlis dukart sumažintas (apačioje).

Paveikslėlio segmentacija gali būti atliekama dviem režimais:

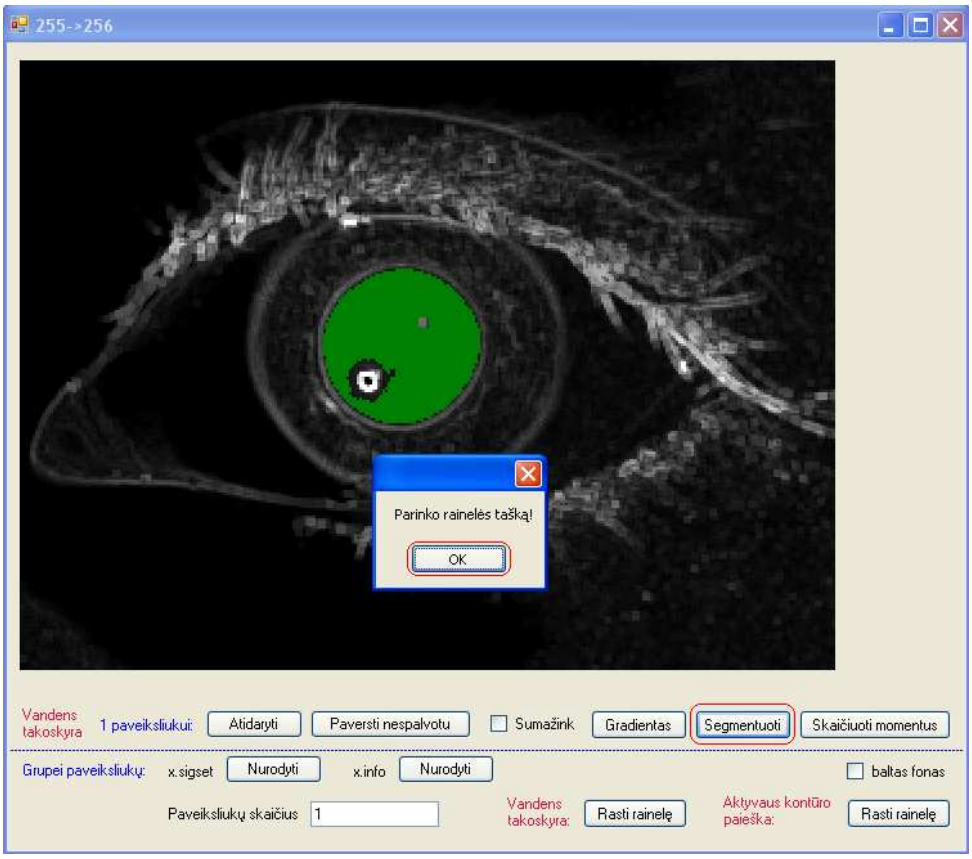
- kai pradiniai taškai sritims parenkami rankiniu būdu (pažymint pele programos lange paveikslėlyje);
- kai pradinius taškus parenka ir sužymi programa automatiškai.

Segmentacija remiasi vandens takoskyros transformacija. Jos metu paveikslėlyje išskiriamos trys sritys: vyzdžio sritis, rainelės sritis ir fono sritis. Jei segmentacija vykdoma pirmuoju režimu (taškai žymimi rankiniu būdu), tai galima parinkti kiek norint taškų kiekvienai

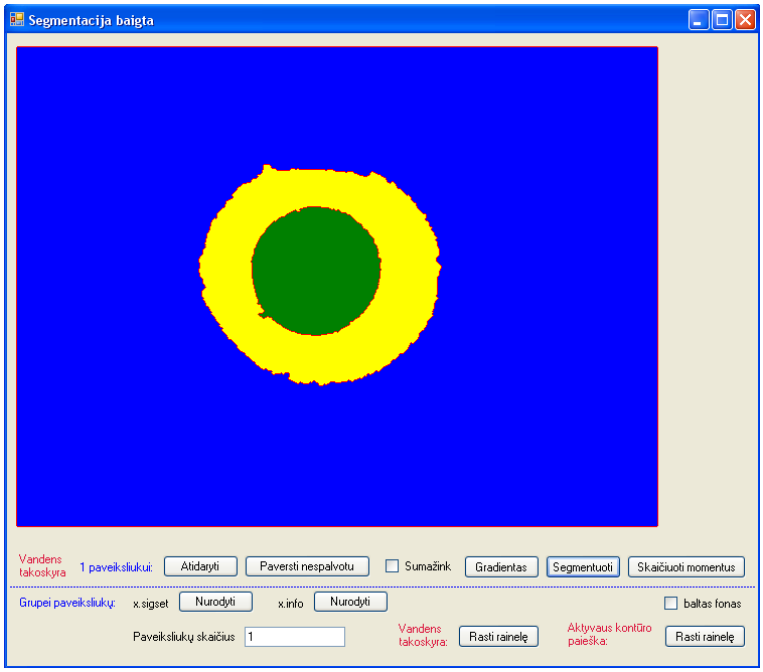
sričiai. Taškai parenkami žymint kairiuoju pelės klavišu programos lange paveikslėlyje. Pirmiausia žymimi vyzdžio taškai, paskui - rainelės, o galiausiai - fono taškai. Baigus žymėti kiekvienos srities taškus, reikia spausti dešinįjį pelės klavišą bet kurioje paveikslėlio vietoje – tai bus ženklas programai, kad vienos srities taškai jau sužymėti ir pereinama prie kitos srities taškų žymėjimo. Žymint taškus, iš karto vaizduojama, kaip plečiasi sritys. Vyzdžio srities elementai (taškai paveikslėlyje) spalvinami žalia spalva, rainelės – geltona, o fono – mėlyna spalva. Pažymėti taškai ir surastų sričių kraštai spalvinami raudonai. Baigus žymėti visų trijų sričių taškus, reikia spausti mygtuką „Segmentuoti“ ir programa pabaigs segmentaciją bei atvaizduos segmentacijos rezultatą naudotojo sąsajos lange [23 pav.].

Segmentacija vykdoma automatinio taškų žymėjimo režimu tuomet, kai nepažymėjus nei vieno taško pele, spaudžiamas mygtukas „Segmentuoti“. Tuomet programa pati parenka ir pažymi sričių pradinius taškus ir atlieka segmentaciją. Vyzdžiui parenkamas vienas taškas, rainelei – du, o fonui – keturi taškai. Taškų parinkimo eigoje naudotojui išmetami langai su trumpais pranešimais ir kaskart perpiešiamas paveikslėlis, kad būtų galima stebėti taškų parinkimo ir segmentacijos eigą [22 pav.]. Pranešimai išmetami tokiais atvejais: kai pažymimas paveikslėlio centras, kai parenkamas kurios nors srities taškas, kai nuspalvinama kuri nors sritis, kai baigiamas i-tasis bandymas (i = 1, 2, ..., 20 - vyzdžiui, i = 1, 2, ..., 5 - rainelei) išskirti vyzdį ar rainelę, kai suranda vyzdį ar rainelę (rašo iš kelinto bandymo surado), jei neranda vyzdžio ar rainelės po nustatyto skaičiaus (20 - vyzdžiui, 5 - rainelei) bandymų. Kai segmentacija pasibaigia, viršutinėje lango juostoje apie tai išvedamas pranešimas [4 pav.]. Po segmentacijos (tiek automatinio, tiek rankinio taškų parinkimo atveju) gautas sričių paveikslėlis išsaugomas atskiroje rinkmenoje gif formatu, kurios pavadinimas sudaromas iš pradinio paveikslėlio rinkmenos pavadinimo ir gale prirašomo „-rez“. Ši rinkmena įrašoma į katalogą „regions“ esantį C:\rezWaterShed\regions. Jei katalogų „rezWaterShed“ ir „regions“ C diske nėra, jie sukuriami automatiškai.

Norint apskaičiuoti sričių paveikslėlio momentinius invariantus, reikia spausti mygtuką „Skaičiuoti momentus“. Tuomet momentiniai invariantai, dar vadinami geometriniais invariantais, (g1 ir g2) bus apskaičiuoti vyzdžio, rainelės ir fono sritims bei kiekvienos iš tų sričių mažesnėms sritims, susidariusioms aplink pažymėtus taškus. Momentinių invariantų skaičiavimo rezultatai išsaugomi tekstinėje rinkmenoje, kurios pavadinimas sudaromas iš pradinio paveikslėlio rinkmenos pavadinimo ir gale prirašomo „-invariants“. Ši rinkmena įrašoma į katalogą „invariants“ esantį C:\rezWaterShed\invariants. Jei katalogo „invariants“ C diske nėra, jis sukuriamas automatiškai.



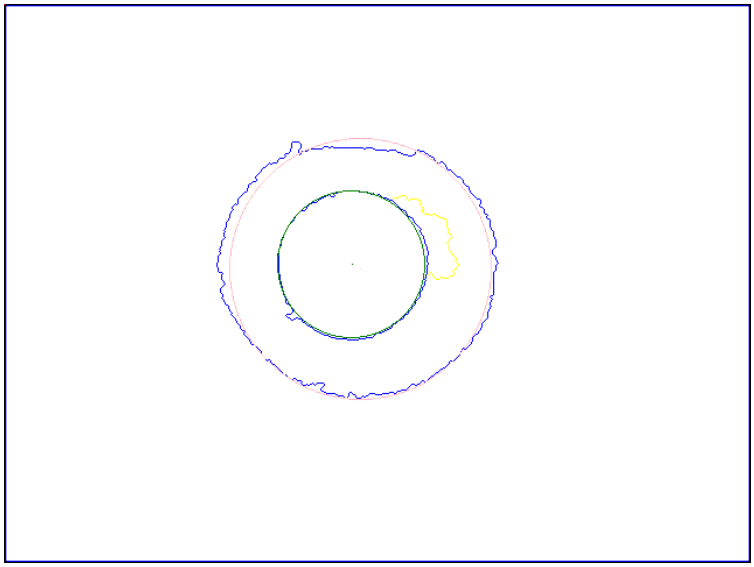
22 pav. Automatinis sričių taškų parinkimas ir žymėjimas bei programos parnešimai, skirti taškų parinkimo ir segmentacijos eigai stebėti. Šiame paveikslėlyje vyzdys jau surastas ir programa parinko pirmąjį rainelės srities tašką.



23 pav. Segmentacijos rezultatas programos naudotojo sąsajos lange.

Pagal momentus apskaičiuojami vyzdžio ir rainelės centrai bei spinduliai ir pagal juos piešiami apskritimai. Vaizdas, kuriame parodyti po segmentacijos gautų sričių kraštai ir pagal

momentus gauti apskritimai (vyzdžio – žalias, o rainelės – rausvas) išsaugomi gif formato rinkmenoje, kurios pavadinimas sudaromas iš pradinio paveikslėlio rinkmenos pavadinimo ir gale prirašomo „-rez2“ [24 pav.]. Ši rinkmena įrašoma į katalogą „regionContours“ esantį C:\rezWaterShed\regionContours. Jei katalogo „regionContours“ C diske nėra, jis sukuriamas automatiškai. Vyzdžio, rainelės ir fono sričių kraštai paveikslėlyje [24 pav.] yra mėlyni, o mažesnės sritys, susidariusios pagal pažymėtus taškus, tarpusavyje atskirtos geltonai (geltoni yra tie kraštai, kurie nesutampa su pagrindinių trijų sričių kraštais).



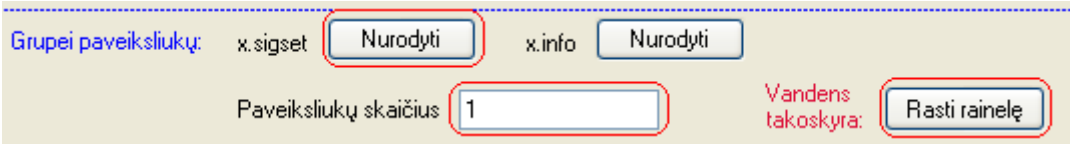
24 pav. Po segmentacijos gautų sričių kraštai ir pagal momentus nupiešti vyzdžio ir rainelės apskritimai (antrasis programos rezultatų paveikslėlis „-rez2“).

Programinis įrankis suteikia galimybę segmentuoti iš karto daug rainelių paveiksliukų ir sukuria ataskaitą apie segmentacijos rezultatus: kuriuose paveiksliukuose ir po kelių iteracijų rado vyzdį ir rainelę, kuriuose tik vyzdį ir kuriems segmentacija nepavyko. Segmentacijos rezultatų paveiksliukai su nuspalvintomis sritimis išsaugomi kataloge C:\rezWaterShed\regions, o ataskaitos rinkmena results.txt – kataloge C:\rezWaterShed. Sričių kraštų paveiksliukai (su pavadinimo gale prirašomu „-rez2“) nekuriami taupant laiką.

Norint segmentuoti grupę paveiksliukų, reikia turėti tiems paveiksliukams paruoštą rinkmeną x.sigset (x – bet koks rinkmenos vardas). x.sigset – tai xml kalba aprašyta informacija apie duomenų bazę, paveiksliukų vietą kompiuteryje, jų formatą ir pavadinimus. Pvz., dviem paveiksliukams iš Spring2004iris paveiksliukų bazės ši informacija atrodo taip, kaip parodyta pavyzdyje [1 pavyzdys]. Segmentacijos procesas prasideda, nurodžius x.sigset rinkmenos vietą kompiuteryje, skaičių, kiek paveiksliukų norime segmentuoti ir paspaudus mygtuką „Rasti rainelę“ šalia užrašo „Vandens takoskyra“ [25 pav.].

```
<?xml version="1.0"?>
<biometric-signature-set xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="ICE2005 Right
Iris" xmlns="http://www.bee-biometrics.org/schemas/sigset/0.1">
  <biometric-signature name="iris/ICE/nd1/Spring2004iris/242912" true-
identity="iris/ICE/nd1/Spring2004iris/291138R">
    <presentation name="iris/ICE/nd1/Spring2004iris/242912" modality="iris"
file-name="iris/ICE/nd1/Spring2004iris/242912.png" file-format="png"/>
  </biometric-signature>
  <biometric-signature name="iris/ICE/nd1/Spring2004iris/244081" true-
identity="iris/ICE/nd1/Spring2004iris/291168R">
    <presentation name="iris/ICE/nd1/Spring2004iris/244081" modality="iris"
file-name="iris/ICE/nd1/Spring2004iris/244081.png" file-format="png"/>
  </biometric-signature>
</biometric-signature-set>
```

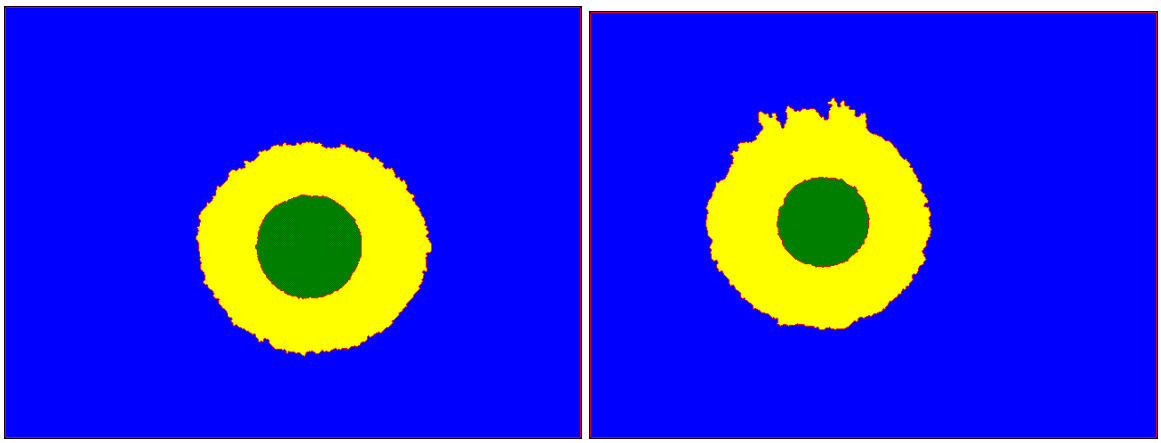
1 pavyzdys. x.sigset rinkmena dviem Spring2004iris paveiksliukų bazės paveiksliukams.



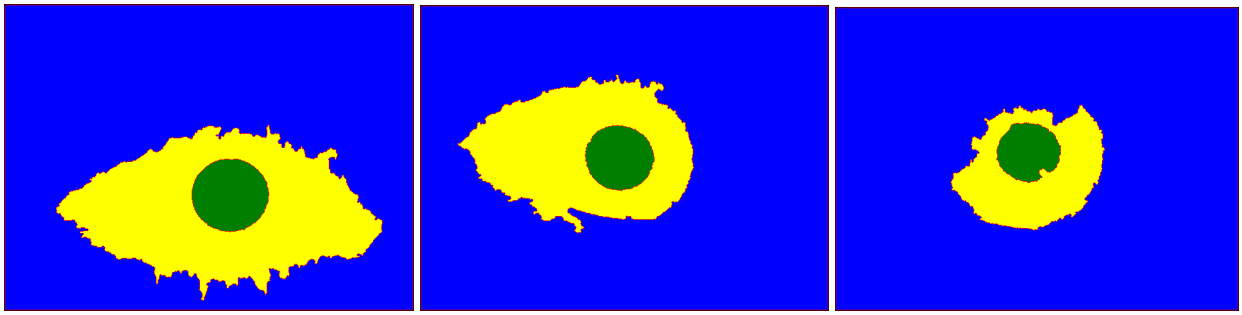
25 pav. Vandens takoskyros segmentacija grupei paveiksliukų iškart.

3. 1. 3. Tyrimų rezultatai

Naudodama savo magistro darbo programą atlikau tyrimus su 132 rainelių paveiksliukais, kuriuos paėmiau iš http://uosis.mif.vu.lt/atpazinimas/iris/ice2005/Spring2004iris/. Gautus tyrimų rezultatus pateikiau lentelėje šios ataskaitos prieduose [1 priedas]. Pirmame stulpelyje yra paveiksliuko rinkmenos pavadinimas, antrame ir trečiame stulpeliuose parašiau po kelių iteracijų programa rado vyzdį ir rainelę atitinkamai. Trečiajame stulpelyje pliuso ženklas reiškia, kad rainelei rasta pakankamai gera sritis, t. y., ne per daug didelė [26 pav.], ženklas + - reiškia, kad sritis rasta kiek per didelė ar per maža (dažniau per didelė) [27 pav.], o minuso ženklas reiškia, kad rainelė nerasta. Ketvirtajame stulpelyje žymėjau pliusą, jei segmentacija nepavyko, t. y., jei nerastas nei vyzdys, nei rainelė. + - žymėjau, jei vyzdys rastas, o rainelė nerasta. Penktajame stulpelyje žymėjau pliusą, jei vyzdžiui surasti neužteko 20 iteracijų (tokios eilutės lentelėje nuspalvintos pilka spalva).



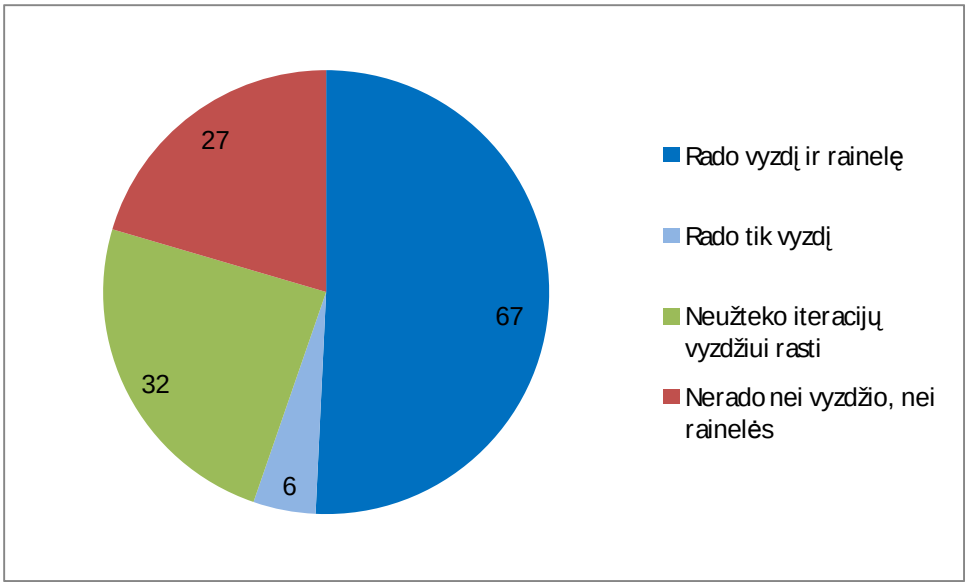
26 pav. Sėkmingai išskirta rainelė.



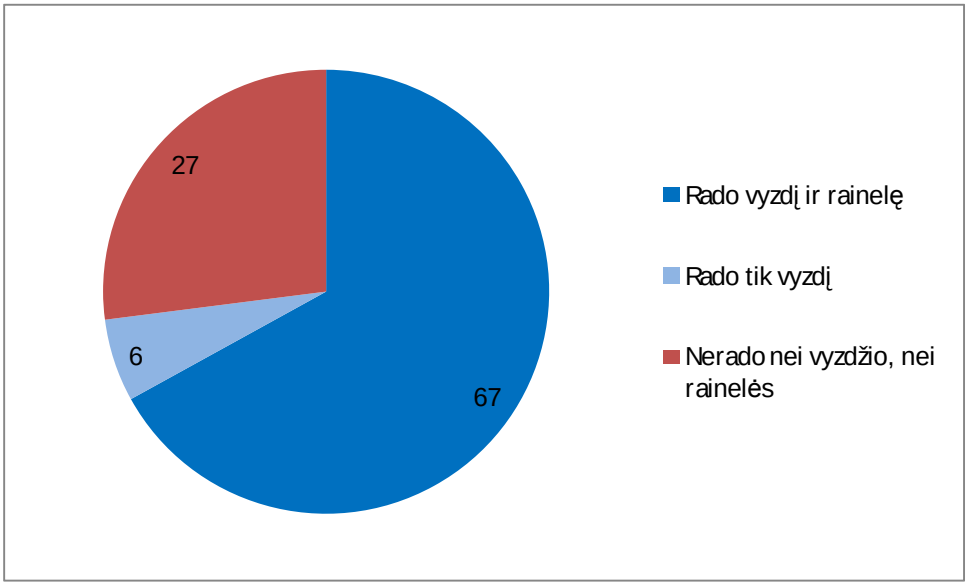
27 pav. Nesėkmingai išskirta rainelė: rainelės sritis išskirta per didelė (pirmi du paveikslėliai iš kairės) arba per maža (trečiasis paveikslėlis iš kairės).

Iš 132 rainelių paveiksliukų 32 neužteko dvidešimties iteracijų vyzdžiui surasti, vyzdys buvo rastas 73 paveiksliukuose, o rainelė 67 paveiksliukuose [1 diagrama]. Į tuos paveiksliukus, kuriems neužteko nustatyto skaičiaus iteracijų vyzdžiui rasti, segmentacijos rezultatuose galime nekreipti dėmesio [2 diagrama], kadangi nežinome, ar segmentacija būtų sėkminga, ar ne, jei būtų nustatytas kitoks iteracijų skaičius. Iš 67 paveikslėlių, kuriuose rainelės segmentacija pavyko, pakankamai gera (ne per daug didelė) rainelės sritis buvo rasta 52 paveikslėliuose [3 diagrama]. Iš tyrimų rezultatų [1 priedas] galima pastebėti, kad rainelei surasti dažniausiai užtenka tik vienos iteracijos, kai vyzdys jau rastas.

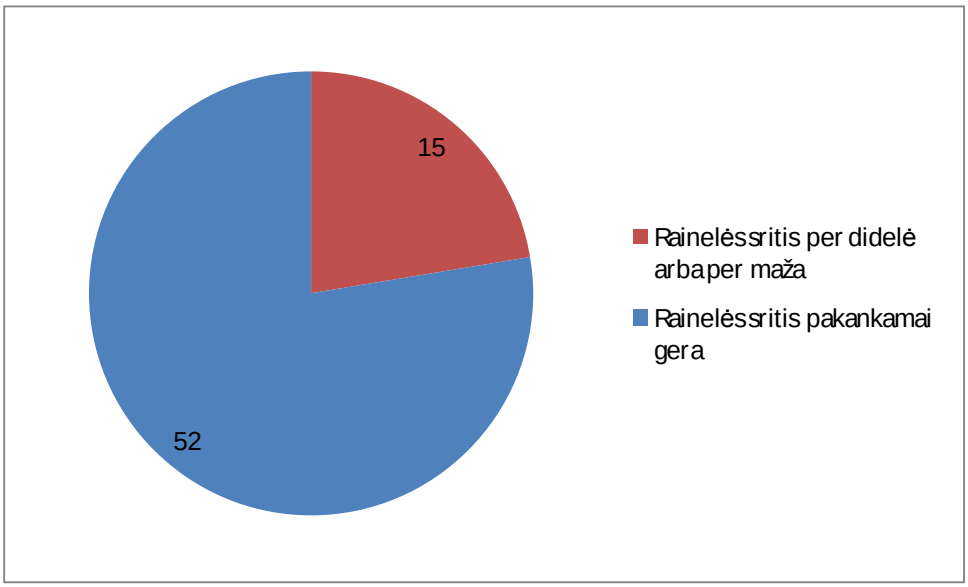
1 diagrama. Segmentacijos tyrimų rezultatai, įskaitant ir tuos paveiksliukus, kuriems neužteko nustatyto skaičiaus iteracijų vyzdžiui rasti.



2 diagrama. Segmentacijos tyrimų rezultatai, neįskaitant tų paveiksliukų, kuriems neužteko nustatyto skaičiaus iteracijų vyzdžiui rasti.



3 diagrama. Rastos rainelės srities įvertinimas.



Dažniausia priežastis, dėl ko segmentacija nepavykdavo buvo ta, kad, ieškant pradinio taško vyzdžio sričiai ir parenkant jį blakstienose, sričiai plečiantis buvo užliejamas ir vyzdys. Jau patikrintose srityse, kurios, parinkus pradinį tašką, užlietos pasirodė esančios ne apskritimo pavidalo (ne vyzdys), mano algoritmas daugiau nebeparenka taškų vyzdžiui ir bando jų ieškoti kitur paveikslėlyje. Taigi jei visas vyzdys patekdavo į sritis su pradiniu tašku blakstienose, tai paskui vyzdyje niekada taško nebeparinkdavo, todėl vyzdžio nesurasdavo. Kita priežastis, kodėl nesurasdavo vyzdžio – tai neryškus vyzdžio kraštas gradientiniame paveikslėlyje. Tokiu atveju užliejant vyzdžio sritį, užliejama ir rainelė.

3. 2. Aktyvaus kontūro paieška

3. 2. 1. Metodas ir algoritmas

Mano sukurtas ir realizuotas aktyvaus kontūro paieškos metodas ieško tikslesnio nei apskritimas rainelės išorinio krašto. Metodas tikslaus rainelės išorinio krašto ieško pagal jam duotus optimalaus rainelę gaubiančio apskritimo matmenis: centro koordinates ir apskritimo spindulį. Taigi, jei turime apytikslį rainelės kraštą – rainelės vietą paveikslėlyje nurodantį apskritimą, naudodami šį aktyvaus kontūro paieškos metodą, galime surasti tikslų arba bent jau tikslesnį negu apskritimas rainelės kraštą.

Rainelei nustatomi minimalus ir maksimalus apskritimai, turintys atitinkamai mažesnį ir didesnį spindulį už optimalų gaubiantįjį apskritimą. Mano programoje minimalaus apskritimo spindulys buvo parinktas $rMin_{raineles} = (R_{raineles} + 3 * R_{vyzdzio}) / 4$ (čia $R_{raineles}$ - optimalaus rainelės apskritimo spindulys, o $R_{vyzdzio}$ - optimalaus vyzdžio apskritimo spindulys) ir maksimalaus apskritimo spindulys - $rMax_{raineles} = (R_{raineles} + 2) * 1{,}2$. Kitaip sakant, rainelės išorinio krašto

paieškai parenkamas žiedas rainelės srities dalyje, kuri yra arti išorinio rainelės krašto, ir akies obuolio srityje. Paveikslėlio vaizdo elementai tarp minimalaus ir maksimalaus apskritimų kraštų (esantys žiede) gali priklausyti ieškomam rainelės išoriniam kraštui. Kitaip sakant, tas kraštas negali išeiti už minimalaus ir maksimalaus apskritimų sankirtos sudaryto žiedo ribų. Ieškomas kontūras turi būti uždaras, iškilas ir jungus.

Kiekvienam iš vaizdo elementų, esančių žiede, pasirenkama vaizdo elemento aplinka – aplink jį tam tikromis nustatytomis kryptimis esantys vaizdo elementai. Krypčių nustatymo algoritmas pateiktas žemiau [Algoritmas 6]. Mano programoje K = 4. Tai reiškia, kad parenkama po 10 vaizdo elementų (k4 = 10), esančių skirtingomis kryptimis nuo centrinio vaizdo elemento, kiekviename ketvirtyje – taigi iš viso parenkama 40 vaizdo elementų (K4 = 40), simetriškai pasikartojančiomis kryptimis. Atlikę algoritmo žingsnius, masyvuose xo ir yo gautume surašytas atstumų nuo centrinio vaizdo elemento iki jo aplinkos vaizdo elementų koordinačių sistemos x ir y ašių kryptimis reikšmes.

Algoritmas 6: Vaizdo elemento aplinkos krypčių nustatymas

Įvestis: *K* – simetriškai pasikartojančių tame pačiame ketvirtyje krypčių skaičius (pvz., 4)

```
1:  kk ← 0
2:  xo[kk] ← 1
3:  yo[kk++] ← 0
4:  xo[kk] ← K + 1
5:  yo[kk++] ← 1
6:  for k = kk to K-1 do
7:      xo[k] ← K + 1 – k
8:      yo[k] ← 1
9:  end for
10: kk ← K
11: xo[kk] ← 3
12: yo[kk++] ← 2
13: xo[kk] ← 1
14: yo[kk++] ← 1
15: k_ ← kk – 2
16: k4 ← 2 * K + 2
17: for k = kk to k4-1 do      - simetriškas K krypčių pasikartojimas 1-ajame ketvirtyje
18:     xo[k] ← yo[k_]
19:     yo[k] ← xo[k_]
20:     k_←k_ - 1
21: end for
22: K4 ← 4 * k4
23: for k = 0 to K4-1 do       - simetriškas k4 krypčių pasikartojimas 2, 3, ir 4-ame ketvirčiuose
24:     if (k >= k4) then
25:         xo[k] ← -yo[k - k4]
26:         yo[k] ← xo[k - k4]
27:     end if
28: end for
```

Gradientiniame paveikslėlyje einant per galimus krašto vaizdo elementus nustatytame žiede ir jų aplinkos vaizdo elementus pagal laikrodžio rodyklę, ieškoma tiksliausio kontūro. Kontūro tikslumas nustatomas pagal tam tikrą skaitinę vertę, kurią sudaro krypties vektorių ilgių ir kontūro vaizdo elementų gradientinių pilkumo lygių reikšmių sandaugų suma, dar įtakojama kontūrui priskiriamo vaizdo elemento artumo pradiniam apskritimui – vidutinis kontūro gradientas. Kuo tikslesnis kontūras, tuo jo vidutinis gradientas didesnis.

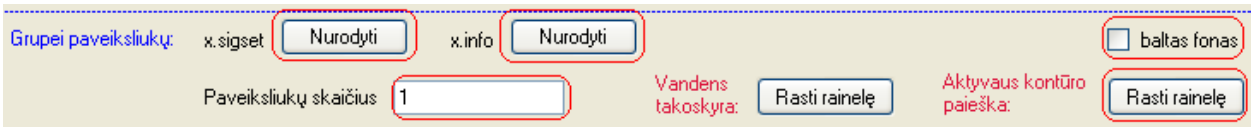
Einant per vaizdo elementus ir ieškant tiksliausio kontūro, tikrinama ar neišeiname už žiedo ribų, kad gautas kontūras būtų iškilas. Taip pat, ieškant rainelės kontūro, tikrinama, ar gausime elipsę. Elipsiškumo sąlyga reikalauja, kad kiekviename iš keturių ketvirčių, einant cikliniu kontūru, atstumas nuo žiedo centro didėtų arba mažėtų. Konkretūs ketvirčiai ir didėjimas arba mažėjimas parenkamas pagal rainelės kontūro susispaudimo pobūdį. Vizuali analizė rodo, jog nemažoje dalyje rainelės paveikslėlių rainelės dalį dengia viršutinis bei apatinis vokai ir pradinis apskritimas susispaudžia iš viršaus ir iš apačios.

3. 2. 2. Programinis įrankis

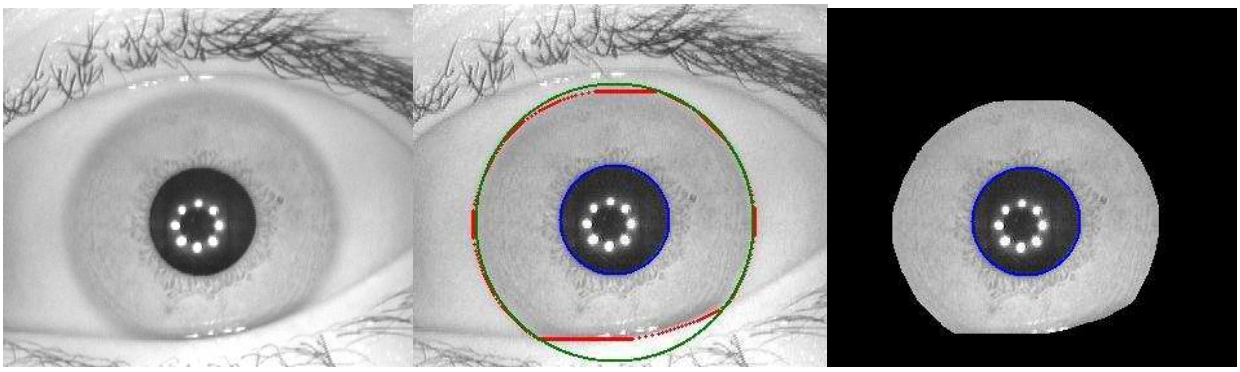
Mano magistro darbo tyrimams skirtas programinis įrankis realizuoja ne tik vandens takoskyros, bet ir aktyvaus kontūro paieškos algoritmą. Aktyvaus kontūro paieškos metodo atveju programa išorinio rainelės kontūro ieško pagal šį metodą, o vyzdžio kontūru laiko optimalų vyzdį gaubiantį apskritimą, kurio matmenys yra iš anksto žinomi ir pateikiami programai. Tyrimams naudojau CASIA3 ir ICE Spring2004iris duomenų bazių paveiksliukus, tačiau gali būti naudojami ir bet kurios kitos duomenų bazės paveiksliukai, kuriems yra paruoštos rinkmenos su reikiama informacija apie paveiksliukus: x.sigset ir x.info. Čia x – bet koks rinkmenos vardas. CASIA3 duomenų bazei tai buvo rinkmenos Casia3-Int.sigset ir Casia3-Int.info. x.info rinkmenoje turi būti surašyti vyzdžio ir rainelės apskritimų centrų koordinatės ir spinduliai (kiekvienam paveiksliukui atskiroje eilutėje iš eilės, atskiriant tarpais: vyzdžio apskritimo centro x koordinatė, vyzdžio apskritimo centro y koordinatė, vyzdžio apskritimo spindulys, rainelės išorinio apskritimo centro x koordinatė, y koordinatė ir spindulys). x.sigset – xml kalba aprašyta informacija apie duomeų bazę, paveiksliukų vietą kompiuteryje ir jų formatus bei pavadinimus (tokia pati rinkmena kaip ir vandens takoskyros transformacijos grupei paveiksliukų atveju [1 pavyzdys]). Šių rinkmenų vietą savo kompiuteryje reikia nurodyti programinio įrankio naudotojo sąsajoje, spaudžiant mygtuką „Nurodyti“ šalia užrašo „x.sigset“ ir „x.info“ atitinkamai rinkmenai. Pradiniai rainelių paveiksliukai, paduodami programai apdoroti, turi būti C diske kataloge, kuris nurodytas x.sigset rinkmenoje. Pvz., CASIA3 duomenų bazei tai buvo katalogas C:\iris\CASIA3\CASIA-IrisV3-Interval.

Programos rezultatai – paveikslėliai su nuspalvintais rainelės kontūrais ir paveikslėliai, kur palikta tik surasta rainelės sritis ir vyzdys, o visas fonas nuspalvintas juodai arba baltai, išsaugomi atskiruose kataloguose C diske: rezCyclic\rezContour ir rezCyclic\rezIris atitinkamai kontūrų ir rainelės srities paveikslėliams. Jei katalogų „rezCyclic", „resContour" ir „rezIris" C diske nėra, jie sukuriami automatiškai.

Paleidus programą (rinkmena IrisRecognition.exe), pirmiausia reikia nurodyti x.sigset ir x.info rinkmenų vietas, paskui įvesti skaičių laukelyje „Paveiksliukų skaičius" - kiek paveiksliukų iš CASIA3 duomenų bazės programai reikės apdoroti [28 pav.]. Pagal nutylėjimą ten būna įrašyta 1. Norint, kad rainelės fonas būtų spalvinamas balta spalva (pagal nutylėjimą jis yra spalvinamas juodai), reikia pažymėti langelį šalia užrašo „baltas fonas". Rainelės foną spalvinti balta spalva reikia tuomet, kai paveiksliukai yra gana tamsūs ir rainelė juose yra tamsi, kad juos būtų patogiau panaudoti rainelių palyginimo programose. Galiausiai reikia spausti mygtuką „Rasti rainelę" šalia užrašo „Aktyvaus kontūro paieška". Programa lange nerodo jokių rezultatų, o rezultatų paveiksliukus įrašo iš karto į reikiamus katalogus. Iliustracijoje [29 pav.] galite pamatyti, kaip atrodo programos rezultatų paveiksliukai su pažymėtu rainelės vidiniu bei išoriniu kontūru ir su išskirta rainelės sritimi. Kai programa baigia apdoroti paveiksliukus, išveda savo lange viršutinėje juostoje žodį „Pabaiga".



28 pav.: Aktyvaus kontūro paieškos metodo paveiksliukų skaičiaus įvedimo laukelis ir mygtukas programos naudotojo sąsajoje.



29 pav.: Kairėje pavaizduotas originalus akies paveikslėlis, viduryje – akies paveikslėlis su mėlyna spalva pažymėtu minimaliu vyzdį gaubiančiu apskritimu, raudonai pažymėtu aktyvaus kontūro paieškos metodu surastu išoriniu rainelės kontūru ir žalia spalva pažymėtu minimaliu rainelę gaubiančiu apskritimu, dešinėje – paveikslėlis, kuriame minimalus vyzdį gaubiantis apskritimas pažymėtas mėlyna spalva, o rainelės išorės fonas nuspalvintas juodai.

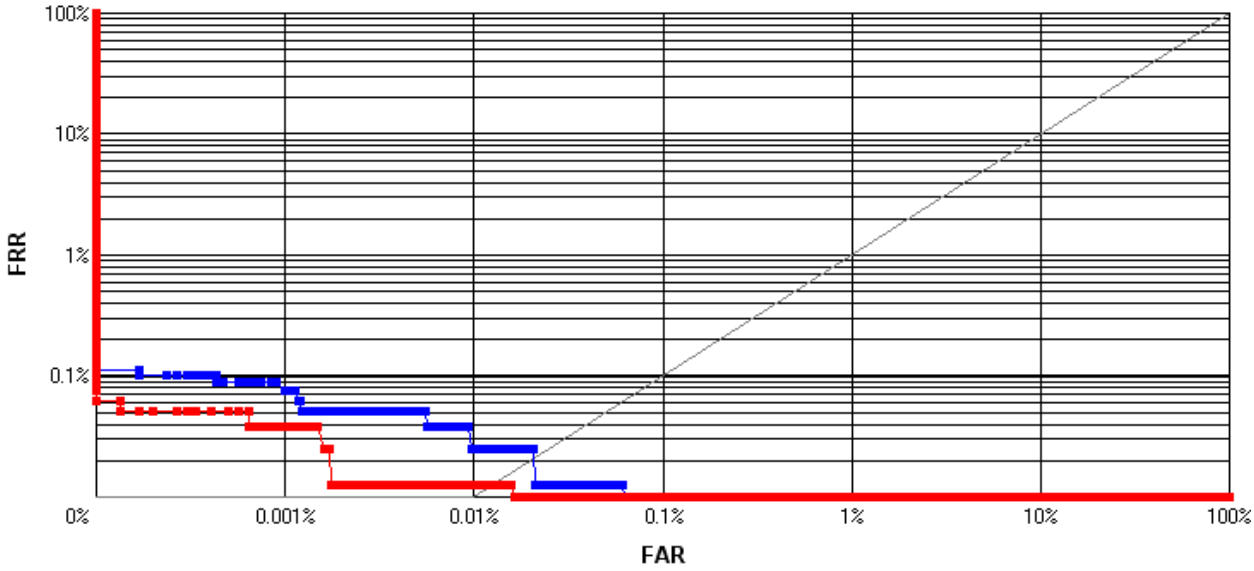
3. 2. 3. Tyrimų rezultatai

3. 2. 3. 1. Segmentacijos ir rainelių atpažinimo rezultatai

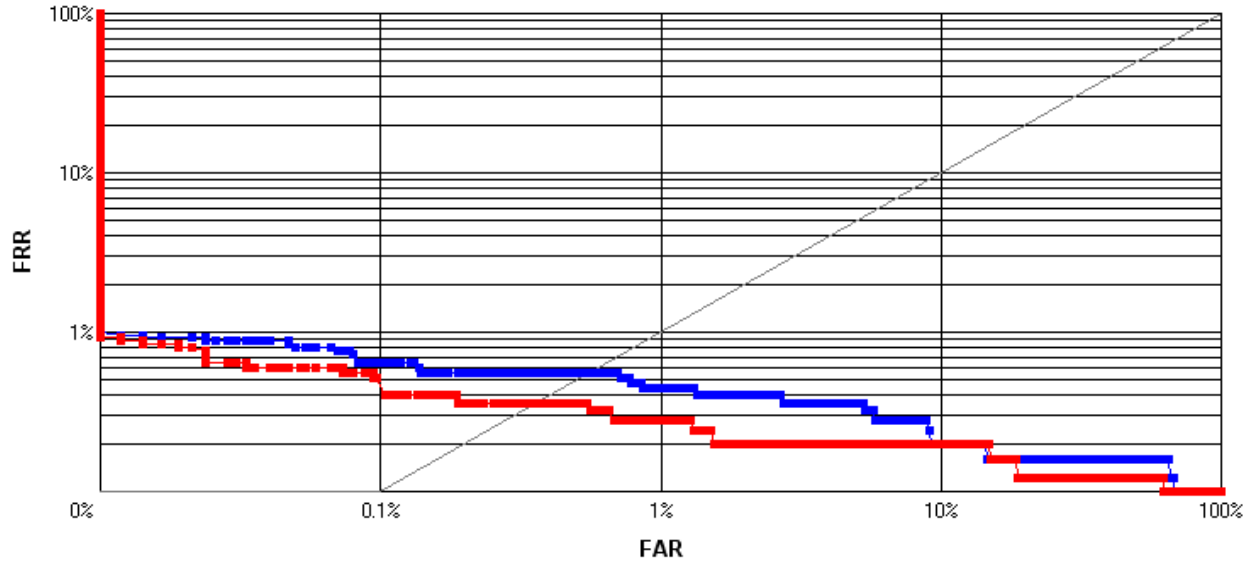
Naudodama savo sukurtą aktyvaus kontūro paieškos metodą įgyvendinantį programinį įrankį, atlikau segmentaciją visiems (2655) CASIA3 duomenų bazės paveiksliukams. Daugumoje paveiksliukų programa surado tikslų arba apytikslį rainelės kontūrą, neprarandant daug rainelės duomenų ir tik 7% paveiksliukų nukirto didesnę ar mažesnę rainelės dalį. Kai kuriuose paveiksliukuose surasta rainelės sritis buvo kiek per didelė, tačiau lyginant raineles tai žymiai mažiau sąlygoja klaidų atsiradimą nei surasta per maža rainelės sritis, todėl tokius paveiksliukus priskyriau prie gerų segmentacijos rezultatų. Taip pat atlikau segmentaciją 300 ICE Spring2004iris duomenų bazės paveiksliukų. Segmentacijos rezultatų paveiksliukuose foną šiuo atveju spalvinau ne juodai, o baltai, nes originalūs rainelių paveiksliukai yra gana tamsūs. Programa visus 300 paveiksliukų susegmentavo gerai, nenukirsdama dalies rainelės srities, tik kai kuriuose paveiksliukuose rainelės sritis rasta kiek per didelė. Lyginant su vandens takoskyros segmentacijos metodo tyrimų rezultatais, galima pastebėti, jog aktyvaus kontūro paieškos metodu surastos kiek per didelės rainelės sritys yra daug mažesnės nei vandens takoskyros metodu surastos per didelės sritys.

Tinkamai segmentuoti CASIA3 duomenų bazės paveiksliukai su tikslesniu nei apskritimai kontūru, kur nėra akivaizdžiai nukirsta rainelės dalis, ir visi segmentuoti Spring2004iris duomenų bazės paveiksliukai buvo naudojami rainelių atpažinimo tyrimams, kuriuos atliko Kompiuterijos katedros studentas Valdemaras Pašvenskas [Paš]. Tyrimams taip pat buvo naudojami tų pačių rainelių paveiksliukai su pažymėtais rainelės ir vyzdžio apskritimais. Visiems šiems paveiksliukams buvo sukurti požymių šablonai. Paveiksliukuose, kurie buvo segmentuoti aktyvaus kontūro paieškos metodu, tuose taškuose, kurie užeina už surasto rainelės kontūro ribų (CASIA3 paveiksliukuose nuspalvinti juodai, o Spring2004iris paveiksliukuose - baltai), atitinkami šablono požymiai buvo pakeisti atsitiktinai parinktais baitais, kad šablonuose jie nesutaptų. Paveiksliukams su rainelės apskritimais ir su aktyvaus kontūro paieškos metodu rastais rainelės kontūrais buvo atlikti šablonų palyginimai ir sudarytos ROC kreivės (apie ROC kreives galima paskaityti literatūros apžvalgos skyrelyje [2. 5.]).

Gautas ROC kreivės galite matyti paveikslėliuose [30 pav.] – CASIA3 paveiksliukams ir [31 pav.] – Spring2004iris paveiksliukams. Akivaizdu, kad aktyvaus kontūro metodo segmentacijos atveju gautos kreivės yra žemiau už originaliems paveiksliukams su rainelės ir vyzdžio apskritimais gautas kreives – tai reiškia, jog atpažinimo klaidų tikimybė yra mažesnė.



30 pav.: Mėlyna kreivė - tai ROC kreivė 2469 CASIA3 duomenų bazės paveiksliukams su pažymėtais rainelės ir vyzdžio apskritimais, o raudona – ROC kreivė tiems patiems paveiksliukams su aktyvaus kontūro paieškos metodu rastu rainelės kontūru.



31 pav.: Mėlyna kreivė - tai ROC kreivė 300 Spring2004iris duomenų bazės paveiksliukų su pažymėtais rainelės ir vyzdžio apskritimais, o raudona – ROC kreivė tiems patiems paveiksliukams su aktyvaus kontūro paieškos metodu rastu rainelės kontūru.

CASIA3 paveiksliukams su pažymėtais vyzdžio ir rainelės apskritimais buvo gauti tokie rainelės atpažinimo klaidų įverčiai: Zero FAR = 0,127% (Zero FAR nurodo mažiausią FRR reikšmę, kuriai esant sistema niekada klaidingai nepalaiko dviejų skirtingų individų biometrinių duomenų vieno individo biometriniais egzemplioriais, t. y., FAR = 0. FRR – klaidingo atmetimo rodiklis, FAR – klaidingo atpažinimo rodiklis. [Paš]), o EER = 0,0208% (EER, angl. Equal Error Rate, nurodo panašumo įverčių slenkstį t, kuriam esant FAR(t) = FRR(t). Čia FAR(t) ir FRR(t) –

funkcijos, priklausančios nuo slenksčio t. [Paš]). Tos pačios duomenų bazės paveiksliukams, segmentuotiems aktyvaus kontūro paieškos metodu, gauta Zero FAR = 0,076%, o EER = 0,0127%. Taigi aktyvaus kontūro metodo atveju Zero FAR įvertis mažesnis 1,67 karto, o EER – 1,64 karto.

Spring2004iris paveiksliukams su pažymėtais vyzdžio ir rainelės apskritimais gauti tokie klaidų įverčiai: Zero FAR = 1,61%, o EER = 0,562%. Aktyvaus kontūro paieškos metodo atveju – Zero FAR = 1,08%, o EER = 0,361%. Vadinasi, paveiksliukams su patikslintu rainelės kontūru Zero FAR įvertis mažesnis 1,49 karto, EER – 1,56 karto.

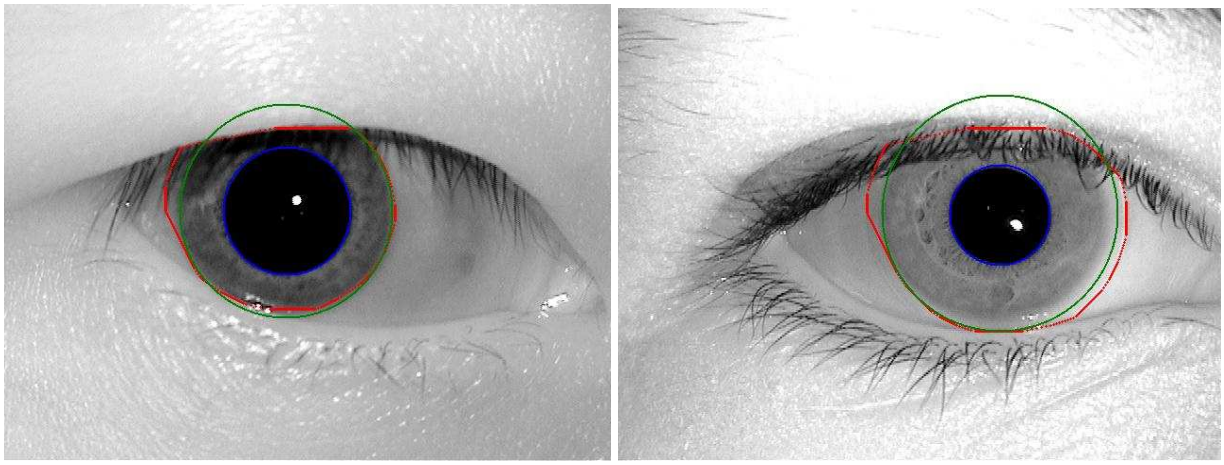
Iš tyrimų rezultatų galima daryti išvadą, kad susegmentavus aktyvaus kontūro paieškos metodu, rainelių atpažinimas bus tikslesnis negu rainelės kontūrą laikant apskritimu. Maždaug 1,6 karto bus mažesnė tikimybė į teisingai atpažintų rainelių rinkinį patekti apsimetėlių (apsimetėlis – asmuo, kuris nėra tas, kuo jis teigia esąs) rainelėms. Vadinasi, aktyvaus kontūro paieškos metodas iš tiesų patikslina rainelės kraštą, kuris iš pradžių laikomas apskritimu, ir yra naudingas rainelių atpažinimui.

3. 2. 3. 2. Rasto rainelės kontūro kokybės analizė

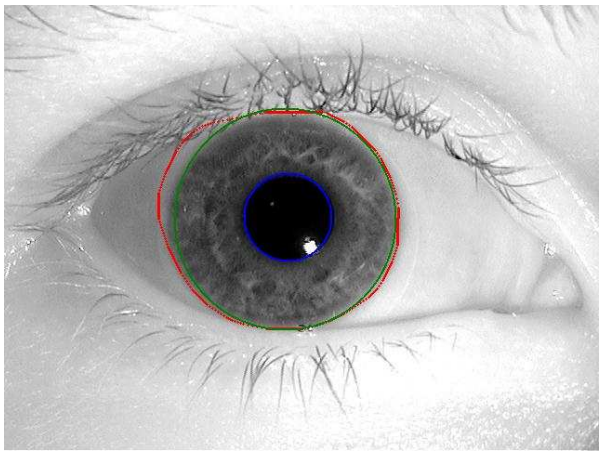
Naudodama aktyvaus kontūro paieškos metodą, atlikau segmentaciją tiems patiems ICE Spring2004iris duomenų bazės paveiksliukams, kurie buvo naudoti vandens takoskyros segmentacijos metodo tyrimams, aprašytiems [3. 1. 3.] skyrelyje. Metodas visus paveiksliukus susegmentavo nenukirsdamas dalies rainelės srities, tik kai kuriuose paveiksliukuose rainelės sritis rasta truputį per didelė.

Kiekvienam paveiksliukui gauto rainelės kontūro kokybės įvertinimą pažymėjau tyrimų rezultatų lentelės [1 priedas] paskutiniajame stulpelyje. Ženklas „++“ reiškia, kad rastas rainelės kontūras, lyginant su minimaliuoju gaubiančiuoju apskritimu, yra nedaug tikslesnis arba rainelei priskirta kiek per didelė sritis, apimanti ir akies stiklakūnio arba voko dalį, yra vizualiai tokio pat dydžio kaip minimalus gaubiantysis apskritimas. Tokių paveiksliukų pavyzdžius galite pamatyti paveikslėliuose [32 pav.]. Ženklas „-+“ šviesiai pilka spalva nuspalvintoje paskutiniojo lentelės stulpelio celėje reiškia, kad rainelei priskirta kiek per didelė sritis vizualiai atrodo truputį didesnė nei minimalus gaubiantysis apskritimas. Šiuo ženklu lentelėje pažymėto paveiksliuko pavyzdys [33 pav.] pateiktas žemiau. Ženklas „+-“ tamsiai pilka spalva nuspalvintoje celėje reiškia, jog rainelei priskirta truputį per didelė sritis, vizualiai atrodanti didesnė nei minimalus gaubiantysis apskritimas ir didesnė negu ženklo „-+” atveju. Tokių paveiksliukų pavyzdžiai yra paveikslėliai [34 pav.]. Ženklai „-+” ir „+-“ žymi atskirus atvejus, norint tiksliau sugrupuoti gautus rezultatus pagal rainelės kontūro kokybę. Na, o ženklas „+” reiškia geriausios kokybės kontūrą, kuris vizualiai sutampa su rainelės kraštu ar yra labai artimas jam arba užeina už rainelės krašto ribų,

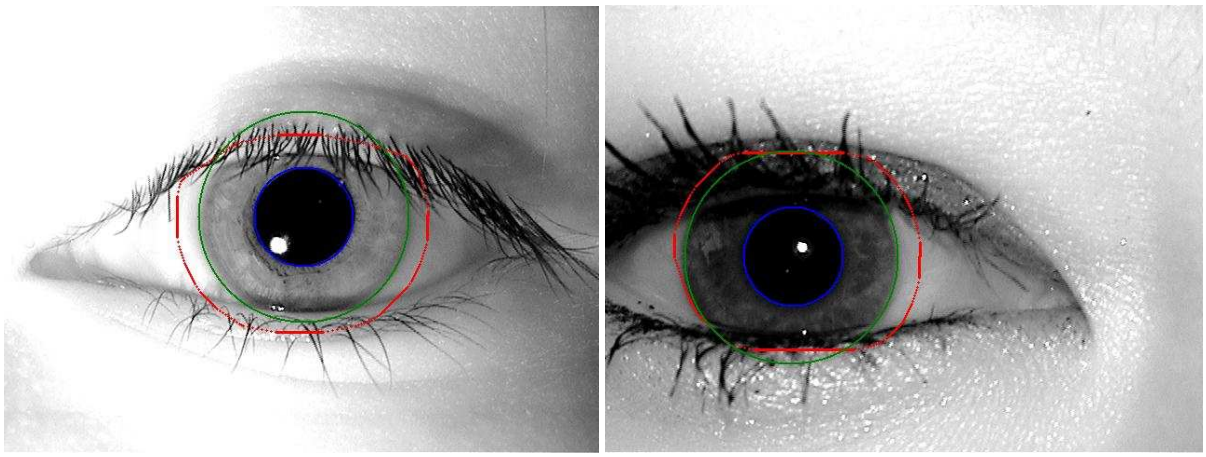
bet daug patikslina minimalų gaubiantįjį apskritimą. Šiuo ženklu pažymėtą kontūro kokybės įvertinimą turinčių paveiksliukų pavyzdžiai [35 pav.] pateikti žemiau.



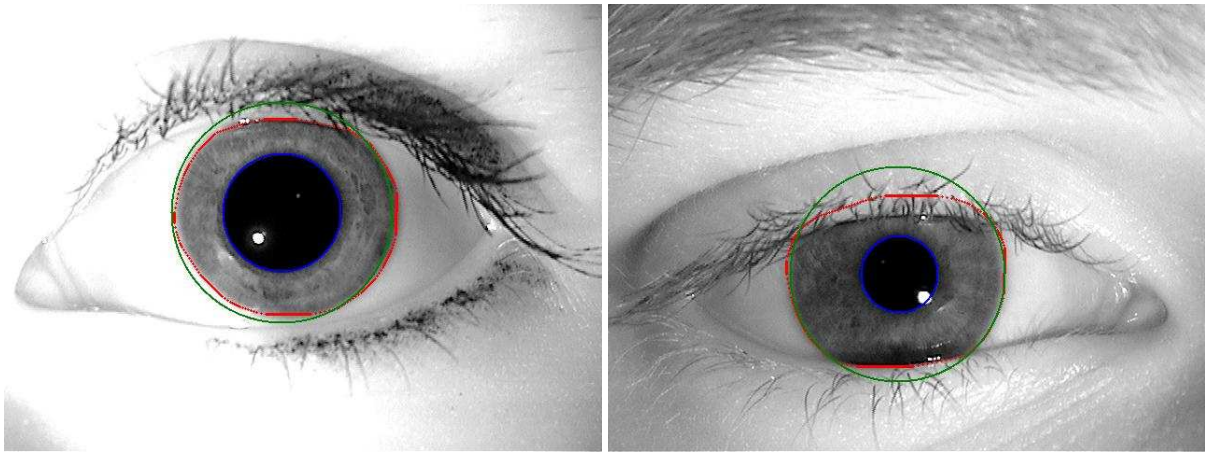
32 pav.: „++“ ženklu lentelėje [1 priedas] žymimas aktyvaus kontūro paieškos metodu surasto rainelės kontūro kokybės įvertinimas. Kairėje esančiame paveikslėlyje rainelės sritis su rastu kontūru yra nedaug mažesnė ir tikslesnė už minimaliuoju gaubiančiuoju apskritimu apribotą sritį. Dešinėje esančiame akies paveikslėlyje surasta rainelės sritis yra vizualiai maždaug lygi apskritimu apribotai sričiai.



33 pav.: „-+“ ženklu lentelėje [1 priedas] žymimas aktyvaus kontūro paieškos metodu surasto rainelės kontūro kokybės įvertinimas. Paveikslėlyje rainelės sritis su rastu kontūru yra truputį didesnė negu minimaliuoju gaubiančiuoju apskritimu apribota sritis.



34 pav.: „+-“ ženklu lentelėje [1 priedas] žymimas aktyvaus kontūro paieškos metodu surasto rainelės kontūro kokybės įvertinimas. Paveikslėlyje rainelės sritis su rastu kontūru yra didesnė negu minimaliuoju gaubiančiuoju apskritimu apribota sritis ir didesnė nei ženklo „-+” atveju.

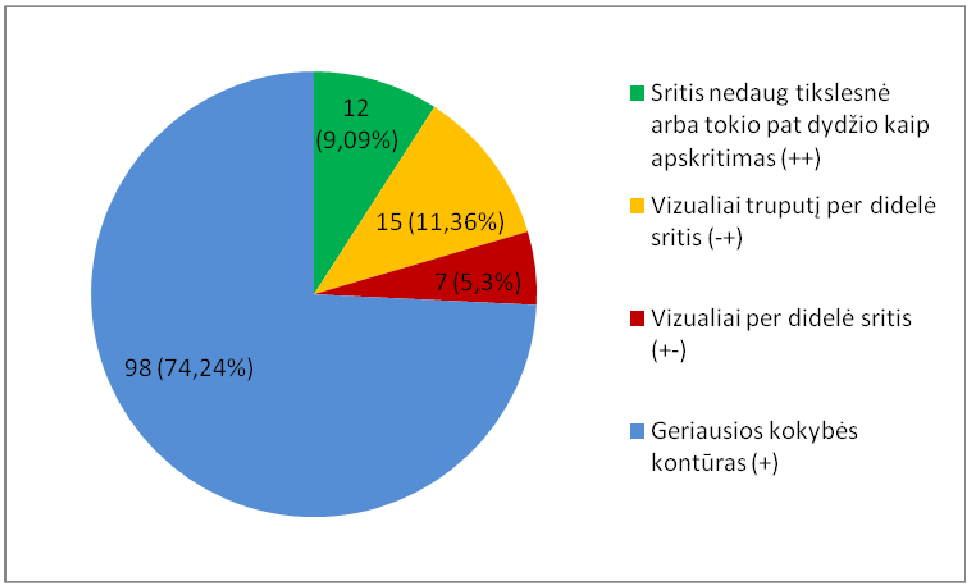


35 pav.: „+“ ženklu lentelėje [1 priedas] žymimas aktyvaus kontūro paieškos metodu surasto rainelės kontūro kokybės įvertinimas. Kairėje esančiame paveikslėlyje surastas rainelės kontūras sutampa su rainelės kraštu, o dešinėje esančiame paveikslėlyje surastas kontūras ne visas sutampa su rainelės kraštu, bet daug patikslina minimalų rainelę gaubiantį apskritimą.

Iš viso šiame metodų rezultatų palyginimo tyrime analizuojami 132 paveiksliukai. 12 (9,09%) iš jų paskutiniame lentelės [1 priedas] stulpelyje gavo ženklą „++”, 15 (11,36%) – ženklą „-+”, 7 (5,3%) – ženklą „+-“ ir 98 (74,24%) – ženklą „+”. Diagrama [4 diagrama] grafiškai vaizduoja šiuos tyrimo rezultatus.

Peržvelgus tyrimo rezultatus, galima teigti, kad daugeliui paveiksliukų aktyvaus kontūro paieškos metodas suranda tikslų arba bent tikslesnį nei apskritimas kontūrą. Tais atvejais, kai (16,66% paveiksliukų) rasta rainelės sritis yra didesnė nei apskritimu apribota sritis, didesnėje dalyje tokių paveiksliukų kontūro netikslumas nėra didelis.

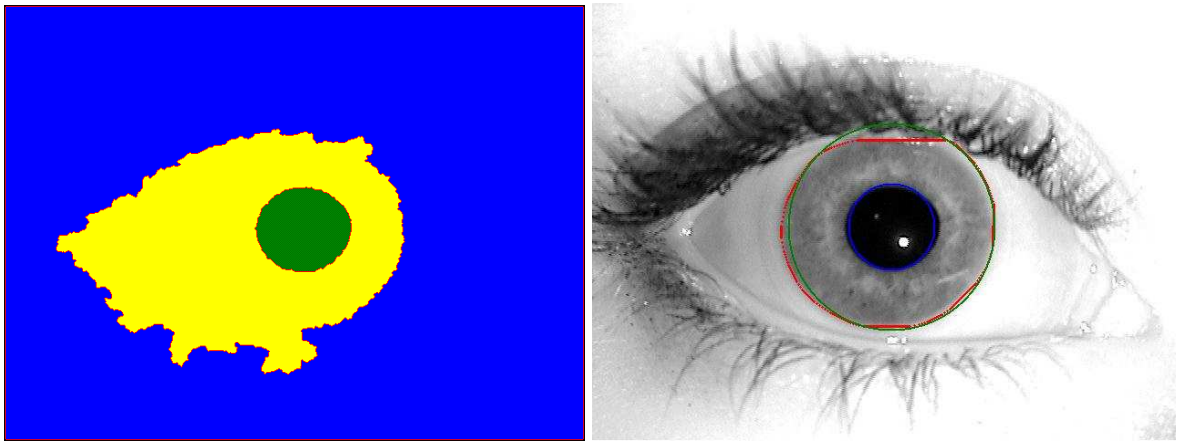
4 diagrama. Rastos rainelės srities įvertinimas. Ženklai „++”, „-+“, „+-“ ir „+“ atitinka lentelės [1 priedas] paskutiniajame stulpelyje naudojamus rasto rainelės kontūro kokybės įvertinimo žymėjimus.



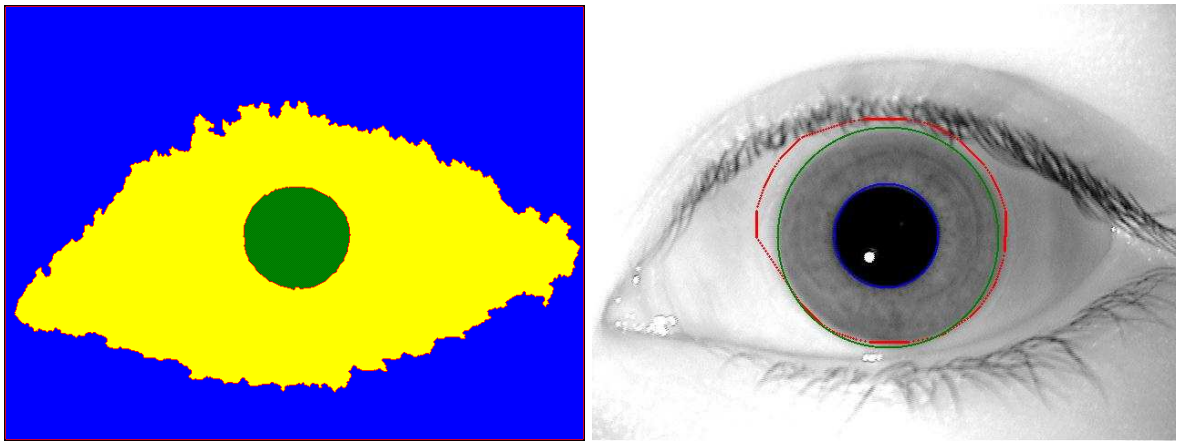
3. 3. Vandens takoskyros segmentacijos ir aktyvaus kontūro paieškos metodų rezultatų palyginimas

Aktyvaus kontūro paieškos metodas duoda rezultatus visiems paveiksliukams. Tai savaime suprantama, kadangi apytiksliai rainelės kraštus nurodančių minimalių gaubiančiųjų apskritimų matamenys yra iš anksto žinomi. Tuo tarpu vandens takoskyros metodas (automatinio žymių aprinkimo atveju) ne visuose paveiksliukuose suranda rainelę ir vyzdį, nes šiuo atveju jokia apytikslė rainelės pozicija paveikslėlyje ar kokia kita paiešką palengvinanti informacija nėra iš anksto žinoma. Dėl tos pačios priežasties aktyvaus kontūro paieškos metodas rezultatus pateikia kur kas greičiau nei vandens takoskyros transformacija ir jam nereikia riboti algoritmo iteracijų skaičiaus.

Kaip jau minėjau skyrelyje [3. 2. 3.], aktyvaus kontūro paieškos metodu surastos kiek per didelės rainelės sritys yra daug mažesnės nei vandens takoskyros metodu surastos per didelės sritys. Pavyzdžiui, palyginkime pirmuosius du paveiksliukus iš [27 pav.] su paveiksliukais [33 pav.] ir [34 pav.]. Tai nereiškia, kad jei rainelei rasta per didelė sritis naudojant vandens takoskyros algoritmą, tai ji bus per didelė ir naudojant aktyvaus kontūro paieškos metodą. Iš lentelėje [1 priedas] pateiktų tyrimų rezultatų (lentelėje pateikti rezultatai ir žymenys aprašyti skyreliuose [3. 1. 3.] ir [3. 2. 3. 2.]) galime nesunkiai pastebėti, kad dažniausiai tam paveiksliukui, kuriam vandens takoskyros metodu rasta per didelė rainelės sritis, aktyvaus kontūro paieška surado tikslų ar gerokai tikslesnį nei apskritimas rainelės kontūrą [36 pav.]. Tik retais atvejais abu metodai tam pačiam paveiksliukui duoda per didelę rainelės sritį [37 pav.]. Tokiu atveju, kaip jau minėjau, aktyvaus kontūro paieška visvien duoda žymiai mažesnę sritį.

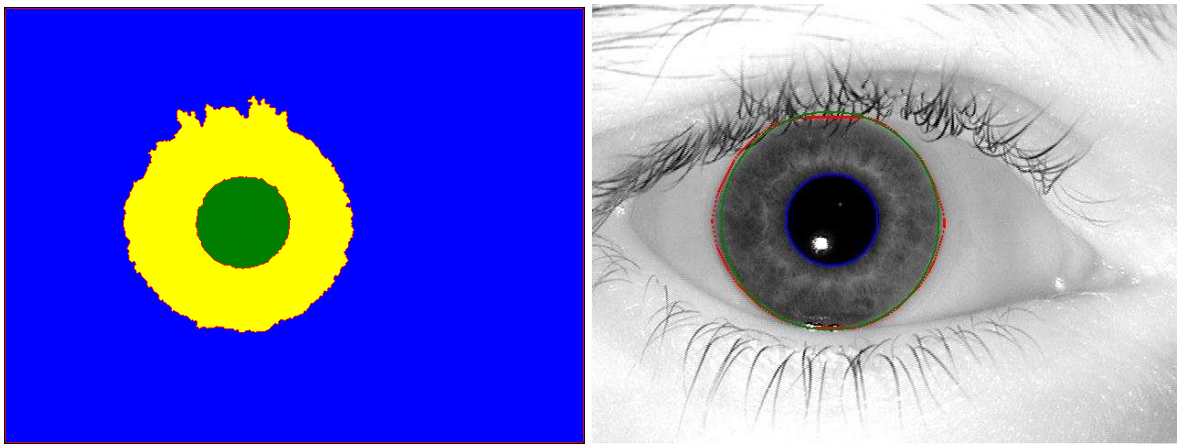


36 pav. Kairėje – vandens takoskyros metodu segmentuotas paveiksliukas. Rasta per didelė rainelės sritis. Dešinėje – aktyvaus kontūro paieškos metodu rastas vizualiai tikslus rainelės išorinis kontūras.



37 pav. Kairėje – vandens takoskyros metodu segmentuotas paveiksliukas. Rasta per didelė rainelės sritis. Dešinėje – aktyvaus kontūro paieškos metodu rastas rainelės išorinis kontūras, ribojantis truputį per didelę rainelės sritį.

Netgi tokiems paveiksliukams, kuriems vandens takoskyros segmentacija yra sėkminga ir rasta rainelės sritis laikoma pakankamai gera, aktyvaus kontūro paieškos metodas suranda tikslesnį rainelės kontūrą [38 pav.].



38 pav. Kairėje – vandens takoskyros metodu segmentuotas paveiksliukas. Rasta rainelės sritis laikoma pakankamai gera. Dešinėje – aktyvaus kontūro paieškos metodu rastas vizualiai tikslus rainelės išorinis kontūras.

Iš šiame skyrelyje aprašyto metodų palyginimo galima padaryti išvadą, jog aktyvaus kontūro metodas duoda daug tikslesnius rezultatus, tačiau vandens takoskyros segmentacijos rezultatus taip pat reikėtų laikyti pakankamai gerais, atsižvelgiant į tai, kad šis metodas nereikalauja jokios pradinės informacijos apie rainelės poziciją paveikslėlyje.

4. Išvados

Pagrindinis tikslas buvo pasiektas – sukurti du metodai: rainelių segmentacijos algoritmas, kuris remiasi vandens takoskyros transformacija ir momentiniais invariantais, bei aktyvaus rainelės kontūro paieška ir juos įgyvendinantis programinis įrankis. Atlikti tyrimai parodė, kad vandens takoskyros rainelių segmentacijos programa vyzdį suranda 73% paveiksliukų, o rainelę 67% paveiksliukų.

Derinant vandens takoskyros transformaciją su momentiniais invariantais rainelių segmentacijoje išvengiama slenksčių ar kitokių iš anksto nustatytų apribojimų paveiksliukams. Be to, segmentacija nepriklauso nuo akies pozicijos ir posūkio kampo paveikslėlyje.

Mano sukurta vandens takoskyros rainelių segmentacijos programa automatinio taškų (segmentacijos žymių) parinkimo atveju demonstruoja visą taškų parinkimo procesą, o ne tik gautą rezultatą, todėl galima stebėti visą žymių parinkimo ir segmentacijos eigą. Taip pat programa suteikia galimybę naudotojui pačiam pažymėti sričių taškus paveikslėlyje, todėl su ja galima atlikti vandens takoskyros transformaciją su žymėmis bet kuriose pageidaujamose paveikslėlio vietose.

Aktyvaus kontūro paieškos metodas patikslina rainelės kontūrą, iš pradžių laikomą apskritimu. Jis, nenukirsdamas rainelės dalies, segmentuoja 93% rainelės paveiksliukų. Rastas rainelės kontūras sutampa su rainelės kraštu arba yra artimas jam ir daug patikslina minimalų rainelę gaubiantį apskritimą 74,24% paveiksliukų. Rainelių atpažinimas, naudojant šį metodą, tampa tikslesnis nei apskritimo kontūro atveju: Zero FAR įvertis mažesnis 1,6 karto.

Aktyvaus kontūro paieškos metodu rasta rainelės sritis visais atvejais yra tikslesnė nei vandens takoskyros metodu rasta sritis. Nepaisant to, sėkmingus vandens takoskyros segmentacijos rezultatus taip pat reikėtų laikyti pakankamai gerais, atsižvelgiant į tai, kad šis metodas, priešingai nei aktyvaus kontūro paieškos metodas, nereikalauja jokios pradinės informacijos apie rainelės poziciją paveikslėlyje.

Toliau mano vandens takoskyros segmentacijos algoritmą būtų galima tobulinti po segmentacijos gautoje rainelės srityje tikslinant rainelės poziciją ir išorinį rainelės kraštą. Rainelės srityje galima bandyti surasti akies voką(-us) bei blakstienas, jei jos užeina ant rainelės. Būtų galima atlikti daugiau tyrimų su automatiniu ir neautomatiniu taškų parinkimu bei

žymėjimu, remiantis tyrimų rezultatais ieškoti būdų patobulinti automatinį taškų parinkimą. Iš momentinių invariantų galbūt būtų galima išvesti formulę rainelei atpažinti.

Vandens transformacijos segmentacijos algoritmą įgyvendinantį programinį įrankį galima patobulinti įgyvendinant galimybę naudotojui nutraukti segmentacijos procesą anksčiau nei segmentacija ar nustatytas žingsnių skaičius baigiasi.

Literatūros sąrašas

[AT06] E. M. Arvacheh, H. R. Tizhoosh, „Iris Segmentation: Detecting Pupil, Limbus and Eyelids“, 2006

[Bas] A.Bastys - Rainelės atpažinimas http://www.mif.vu.lt/~bastys/academic/ATE/iris/rainele.htm

[BTB] British Test Biometrcs Systems at U.K. Borders http://www.cio.com/article/109002/British_Test_Biometrics_Systems_at_U.K._Borders

[CIS] Roger F. Larico Chavez, Yuzo Iano, Vincente B. Sablon, „Process of Recognition of Human Iris: Fast Segmentation of the Iris“

[CMM] Centre de Morphologie Methematique „Image segmentation and mathematical morphology“ http://cmm.ensmp.fr/~beucher/wtshed.html

[CWT+] Jiali Cui, Yunhong Wang, Tieniu Tan, Li Ma, Zhenan Sun, „A Fast and Robust Iris Localization Method Base don Texture Segmentation“

[Daua] John Daugman, „How Iris Recognition Works”

[Daub] John Daugman, „Flat ROC Curves, Steep Predictive Quality Metrics: Response to NISTIR-7440 and FRVT/ICE2006 Reports“

[Dau00] John Daugman, „Biometric decision landscapes“, 2000

[Dau04] John Daugman, „Iris recognition border-crossing system in the UAE”, 2004 http://www.cl.cam.ac.uk/~jgd1000/UAEdeployment.pdf

[Dau06] John Daugman, „Probing the Uniqueness and Randomness of IrisCodes: Results From 200 Billion Iris Pair Comparisons”, 2006

[Dau07] John Daugman, „New Methods in Iris Recognition”, 2007

[Dau93] John Daugman, „High Confidence Visual Recognition of Persons by a Test of Statistical Independence“, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15(11), 1993 http://www.cl.cam.ac.uk/~jgd1000/PAMI93.pdf

[DD01] John Daugman and Cathryn Downing, „Epigenetic randomness, complexity and singularity of human iris patterns“, 2001

[Flu00] Jan Flusser, „On the independence of rotation moment invariants“, 2000 http://wotan.liu.edu/docis/lib/tian/rclis/dbl/patrec/(2000)33%253A9%253C1405%253AOTIORM%253E/www.utia.cas.cz%252Flibrary%252Fprace%252F20000033.pdf

[Flu06] Jan Flusser, „Moment Invariants in Image Analysis“, 2006 http://www.waset.org/pwaset/v11/v11-35.pdf

[GKN03] James Greco, David Kallenborn, Dr. Michael C. Nechyba, „Statistical Pattern Recognition of the Iris“, 2003

[GMR+] P. Gupta, H. Mehrotra, A. Rattani, A. Chatterjee, A. K. Kaushik, „Iris Recognition using Corner Detection“

[ICAO] International Civil Aviation Organization, http://www.icao.int/

[ITe] Iridian Technologies kompanijos tinklalapis http://www.iridiantech.com/

[KZa] W. K. Kong, D. Zhang, „Accurate Iris Segmentation Based on Novel Reflection and Eyelash Detection Model“

[KZb] Wai-Kin Kong, David Zhang, „Detecting Eyelash and Reflection for Accurate Iris Segmentation“

[PA06] H. Proenca, L. A. Alexandre, „Iris segmentation metodology for non-cooperative recognition“, 2006

[Paš] Magistrinį darbą tuo pačiu pavadinimu rašo 2 magistrantūros kurso Kompiuterijos katedros studentas Valdemaras Pašvenskas

[QAI] Q&A: Identity card plans http://news.bbc.co.uk/1/hi/uk_politics/3127696.stm

[Sch] Iris scans take off at Schiphol http://www.expatica.com/actual/article.asp?subchannel_id=63&story_id=59

[Sch Iris] Iris Scan http://www.schiphol.nl/privium/portlet/Irisscan.jsp?PORTLET%3C%3Ecnt_id=10134198673765529&FOLDER%3C%3Efolder_id=2534374302572265&ASSORTMENT%3C%3East_id=1408474395729234&VIRTUAL_TEMPLATE%3C%3Evt_id=10134198673766991&bmUID=1134246540515

[UK Iris] IRIS - Iris recognition immigration system http://www.ind.homeoffice.gov.uk/applying/iris/

[Wika] Wikipedia, „Active contour“ http://en.wikipedia.org/wiki/Active_contour

[Wikb] Wikipedia.Биометрия http://wiki.oszone.net/index.php/Биометрия._Радужная_оболочка_глаза http://wiki.oszone.net/index.php/Биометрия._Введение

[Wikc] Wikipedia, „Grayscale“ http://en.wikipedia.org/wiki/Grayscale

[Wikd] Wikipedia, „Image moments“ http://en.wikipedia.org/wiki/Image_moments

Priedai

1 priedas. Segmentacijos tyrimų rezultatų lentelė.

Paveiksliukas	Iš kelinto karto rado vyzdį	Iš kelinto karto rado rainelę	Segmentacija nepavyko	Neužteko iteracijų vyzdžiui rasti	Aktyvaus kontūro paieška. Kontūro kokybė
239219	15	1 + -			++
239224				+	+-
239226	1	1 +			+
239229	14	1 +			+
239230	4	1 + -			+
239234	3	5 +			+
239237	7	1 +			+
239246	4	2 +			++
239251				+	+
239254				+	+
239256			+		+
239258	2	1 + -			+
239259	4	-	+ -		+
239261	1	4 +			+
239262			+		+
239263			+		+
239264	1	1 +			-+
239267			+		+
239271				+	+
239273	16	1 + -			+
239275			+		-+
239276	3	1 +			+
239279	1	1 +			+
239280	1	-	+ -		+
239281	12	1 + -			++
239292	4	3 +			+
239293	1	1 +			-+
239294	14	1 +			+
239302			+		++
239308	18	2 +			+
239313			+		+
239316				+	+-
239321				+	-+
239330	6	3 +			+
239332	1	2 +			+

239335				+	-+
239336			+		+
239337				+	+
239338	2	1 +			+
239343				+	+
239344				+	-+
239347	1	1 +			+
239349				+	+
239353	7	2 +			+
239354	1	-	+ -		+
239363	1	2 +			+
239369	5	1 +			+
239371			+		+
239372				+	+
239373			+		++
239375				+	+
239377	1	1 +			+
239378	4	1 +			+
239381	1	1 +			+
239382			+		+
239387				+	+
239388	1	2 +			+
239389	1	1 +			+
239390	1	1 +			+
239400	1	1 +			+
239401	1	1 +			+
239404	3	1 +			+
239407	3	1 +			+
239410	1	1 +			+
239411			+		+-
239414				+	-+
239416	14	-	+ -		+
239419			+		++
239426	10	1 +			+
239428	5	1 +			+
239433	13	2 +			++
239435	2	1 +			+
239436				+	+
239439			+		+
239440				+	+
239443	5	1 +			+

239444	1	1 +			+
239445			+		+
239447	10	1 +			+
239448			+		+
239449	1	1 +			+
239454	3	1 +			+
239455				+	++
239457	1	-	+ -		+
239458	16	1 + -			++
239461			+		+
239463	4	1 + -			+
239465				+	+
239469			+		+
239470			+		+
239471	6	1 + -			+
239473	1	-	+ -		+
239477			+		+
239478			+		+
239479				+	+-
239483				+	+-
239485	5	1 + -			+
239489	1	1 + -			-+
239495	2	1 +			+
239499	7	1 +			+
239501	1	1 +			+
239505	3	1 + -			+
239509				+	+
239511				+	-+
239512	2	1 + -			+
239513	1	1 + -			++
239515				+	-+
239518			+		+
239519				+	-+
239520				+	++
239522				+	+
239527				+	-+
239538			+		+
239540				+	-+
239542	3	1 +			+
239543			+		+
239544				+	+-

239545	1	1 +			+
239546				+	+
239547			+		+
239554	10	1 + -			+
239555	4	1 +			+
239556	2	1 +			-+
239557			+		+
239559			+		+
239561	2	1 +			+
239563	5	1 +			++
239565	2	1 +			+
239566	1	1 + -			-+
239568				+	+-
239571	3	1 +			+
239572	2	1 +			+