VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Eva Sinkevič
Informatikos studijų programa

Geometriniai ir funkcionaline analize paremti vaizdo segmentavimo metodai (Geometric and functional-based image segmentation)

Magistro baigiamasis darbas

Vadovas lektorius Irus Grinis

Recenzentas docentas Antanas Lenkevičius

Vilnius 2012

Turinys

Įvadas

Vaizdų segmentavimas yra naudojamas daugelyje mokslo sričių, tokių kaip geografija, geologija ar biologija, tam, kad galėtume atskirti objektus apdoruojamuose paveikslėliuose. Segmentavimas – tai tam tikrų duomenų suskirstymas į atskiras identifikuojamas dalis. Šiuo atveju tai vaizdų suskirstymas į smulkesnes dalis (segmentus). Segmentavimo tikslas yra paveiksliuko supaprastinamas arba toks struktūros pakeitimas, kad tam tikros vaizdo dalys būtų ryškesnės ir jas būtų galima lengviau išanalizuoti. Segmentavimo rezultatu yra aibė vaizdo segmentų arba kontūrų. Rinkdamiesi vaizdų segmetavimo algoritmą dažniausiai atkreipiame dėmesį į jo efektyvumą, sudėtingumą, vykdymo laiką bei pritaikymą skirtingiems vaizdų tipams. Kadangi dažniausiai segmentuojame sudėtingos struktūros bei blogesnės kokybės vaizdus, labai svarbu, kad segmentavimo algoritmas tiktų visų klasių vaizdams. Todėl pagrindinis šio darbo tikslas būtų išsamus algoritmo išnagrinėjimas bei nustatymas su kokiomis parametrų konfigūracijomis gauname geriausius rezultatus atskiroms vaizdų klasėms. Kas sudarytų pagrindą efektyviai vaizdų segmentavimo programos realizacijai naudojant Mumford Shah metodą.

Vienu populiariausių ir efektyviausių vaizdų segmentavimo įrankiu laikomas Mumford Shah funkcionalas. Algoritmas pasižymi lengva realizacija programoje bei greitu įvykdymu. Ieškomo objekto kraštus išgauname minimizuodami duotąjį funkcionalą. Didžiausiu šio algoritmo privalumu yra galimybė pritaikyti jį skirtingo tipo vaizdams. Keisdami į Mumford Shah funkcionalo formulę įeinančių parametrų reikšmes galime įtakoti segmentavimo rezultatus. Tokiu atveju atsiranda galimybė gauti geriausius segmentavimo rezultatus tam tikroms vaizdų klasėms. Kadangi mums yra svarbu, kad algoritmas tiktų daugeliui vaizdų klasių, šio darbo tikslas būtų nustatyti tinkamiausias parametrų reikšmes kiekvienai vaizdų klasei, taip optimizuojant skirtingų vaizdų tipų segmentavimą. Tokiu atveju, svarbu yra tiksliai žinoti kokios parametrų reikšmės geriausiai tinka kokiai vaizdų klasei. Suskirstę vaizdus į atitinkamas klases ir kiekvieną vaizdą segmentuodami su skirtinga funkcionalo formule, galime gauti optimaliausius segmentavimo rezultatus. Darbo tikslas būtų išnagrinėti kokiu atveju algoritmas veikia efektyviausiai. Darbe yra išsamiai išnagrinėjami visi įmanomi funkcionalo formulės atvejai su skirtingomis parametrų reikšmėmis ir bandoma išanalizuoti, kada kokioms vaizdų klasėms gauname geriausius rezultatus. Visi gauti rezultatai ir tezės yra pagrindžiami statistiniu tyrimu.

Algoritmo efektyvumą nemažai įtakoja minimizuojamos kreivės forma. Kai kuriems nesudėtingos struktūros paveiksliukams tinka bet kokia pradinės kreivės forma. Geriausiai objektai yra aptinkami, kai jie yra išsidėstę paveiksliuko centre ir turime vienspalvį foną. Tačiau kaip žinome, dažniausiai segmentuojame sudėtingus vaizdus, kur turime ne vieną objektą ir dauguma jų

yra sunkiai aptinkami. Taip pat kai kurie objektai gali būti išsidėstę palei vaizdo kraštus, kas apsunkina jų aptikimą. Todėl turime išsamiai išnagrinėti minimizuojamos kreivės įtaką vaizdų aptikimui bei parinkti labiausiai tinkančią minimizuojamos kreivės formą, taip optimizuodami algoritmo veikimą.

Taip pat labai svarbu yra algoritmą išnagrinėti statistiškai. Tokiu būdu galime pagrįsti tam tikrų parametrų reikšmių tinkamumą atskiriems paveiksliukų tipams. Svarbu yra išanalizuoti algoritmo efektyvumą palyginant jį su kitais segmentavimo algoritmais. Kas yra išsamiai pateikta darbe.

1. Mumford Shah metodas

Vienas iš vaizdų segmentavimo būdų yra tam tikro funkcionalo minimizavimas. Vaizdų segmentavimui galime parinkti Mumford Shah funkcionalą [5], [6], [7], [8], kuris yra laikomas vienu efektyviausių vaizdų segmentavimo įrankių. Mumford Shah funkcionalo minimizavimas leidžia apsikeitimą duomenimis tarp dvimatės paveiksliuko struktūros ir vienmatės parametrizuotos kreivės/kontūrų. Spręsdami segmentavimo uždavinį, paveiksliuką apžvelgiame kaip funkciją g(x, y) – šviesos intensyvumą taške (x, y). Paveiksliukas g(x, y) gali būti trūkus palei objekto kraštus. Bendra Mumford Shah funkcionalo formulė atrodo atitinkamai:

$$F_{MS}(u,K) = \int_{B\setminus K} |\nabla u|^2 + |K| + \alpha \int_B |u-g|^2 \qquad (1)$$

kur $K \subset B$ vienmatė aibė vaizduojanti kraštus, o |K| - tos aibės dydis. F_{MS} minimizatorius {u, K} sukūria reikiamą rezultatų paveiksliuką u, aptikdamas objektų kraštus.

Galime apžvelgti alternatyvų funkcionalą, nepriklausomą nuo parametrų – pavyzdžiui, geometrinį funkcionalą, vaizduojanti paveiksliuką bei paveiksliuko duomenis paviršiaus erdvėje. Paveiksliuką u laikydami dvimačiu paviršiumi, privalome pirmus du formulės elementus pakeisti to paviršiaus sritimi. Trečią elementą pakeičiame Dirac matu, vaizduojančiu atstumą tarp dviejų paviršių. Pastebėkime, jog paskutinis formulės elementas lokaliai sujungia paveiksliuką u(b) kiekvienam $b \in B$ su duomenimis g tame pačiame taške b. Taip gauname vienas prie vieno atitikimą tarp duomenų ir paveiksliuko taškų.

Skaičiuojant funkcionalą be parametrų paveiksliukas yra vaizduojamas kaip grafas: $U^Y := f : B \to Y$. Tuomet funkcionalas atrodo atitinkamai:

$$A(U) = \int_S |\Gamma(U)(s)|\, ds = \int_B \sqrt{\gamma^2 + \gamma\,|\nabla f\,|^2}\, db \qquad (2)$$

Matavimams skalę γ padalijame i γ_1 ir γ_2. Sritis normalizuojame su $\frac{1+\sqrt{\gamma_1}}{\sqrt{\gamma_1}}$. Tuomet gauname funkcionalo formulę:

$$F^{\beta}_{\gamma_1,\gamma_2}(f) = (1+\sqrt{\gamma_1})\int_B \sqrt{\gamma_1 + |\nabla f\,|^2}\, db + \alpha\, D^2_{\beta,\gamma_2}(\mu, f) \qquad (3)$$

Galutinė funkcionalo formulė atrodo atitinkamai:

$$F^{\beta}_{\gamma_a,\gamma_{\dot{e}}}(f) = (1+\sqrt{\gamma_1})\int_B \sqrt{\gamma_1 + |\nabla f\,|^2}\, db + \alpha\beta \ln\left[\int_B \Psi^f_{\beta,\gamma_2}(b)db\right] \qquad (4)$$

kur

$$\psi^f_{\beta,\gamma_2}(b) = \frac{1}{\int_B e^{-[|f(b)-g(b)|^2+\gamma_2|b-b'|^2]/\beta} db'} \qquad (5)$$

Funkcionalo minimumu yra funkcija f, kuri sprendžia kraštų reikšmės problemą:

$$-div\left(\frac{(1+\sqrt{\gamma_1})\nabla f}{\sqrt{\gamma_1 + |\nabla f\,|^2}}\right) + \frac{2\alpha\psi^f_{\beta,\gamma_2}(b)}{\int_B \psi^f_{\beta,\gamma_2}(b)db}\, f - \frac{2\alpha(\psi^f_{\beta,\gamma_2}(b))^2}{\int_B \psi^f_{\beta,\gamma_2}(b)db}\int_B g(b')e^{-[|f(b)-g(b')|^2+\gamma_2|b-b'|^2]/\beta} db' = 0 \qquad (6)$$

su Neuman kraštų sąlyga:

$$\frac{\partial f}{\partial n} = 0 \text{ per } \partial \mathrm{B}$$

kur g(b) – paveiksliuko grafas.

Taip pat turime parinkti atitinkamą segmentavimo techniką Mumford Shah funkcionalui. Yra trys variacinės segmentavimo technikos: difuzinė, regionų ir plėtojamos kreivės. Difuzinėje technikoje pikselius suliejame su kaimyniniais pikseliais. Regionų technikoje pirmiausia paveiksliukas yra padalijamas į atskirus regionus, vėliau regionai yra suliejami su kitais tinkančiais regionais. Pastebėkime, kad vienas pikselis gali sudaryti vieną regioną (tokiu atveju turėsime m x n regionų). Tuomet iš eilės skaičiuojame ar sujungus du kaimyninius regionus sumažėja funkcionalo reikšmė. Jeigu taip, juos suliejame. Plėtojamos kreivės technikoje paveiksliukas yra susegmentuojamas kreivei dinamiškai judant paveikslėlyje. Reikėtų pastebėti, jog šis vaizdų segmentavimo karkasas yra ypač efektyvus. Kadangi tokiu atveju gauname greičiausią segmentavimą. Ši technika dar yra vadinama aktyviais kontūrais. Aktyvūs kontūrai [1], [2], [3], [4] gali būti naudojami segmentuojant paveiksliuką automatiškai. Tačiau šioje technikoje yra vienas nemažas trūkumas – turi būti iš anksto žinoma segmentuojamo regiono topologija. Viena geriausių išeičių šiai problemai yra aktyvių kontūrų be viršūnių technika. Pastebėkime, jog šis modelis gali aptikti objekto kraštus naudojant ir nenaudojant gradientą. Nemažu trūkumu taip pat yra šios technikos (kartu su Mumford Shah funkcionalu) nesugebėjimas segmentuoti triukšmingus paveiksliukus.

2. Aktyvių kontūrų modelis

Aktyvių kontūrų modelis [1], [2], [3] , [9], [10], [11], [12]., taip pat kartais vadinamas „gyvačiuke“, tai yra karkasas objekto kraštų aptikimui galimai triukšmingame 2D paveikslėlyje. Karkasas bando minimizuoti duotojo kontūro energiją, kuri yra suma vidinės ir išorinės energijos. „Gyvačiuke” galėtume pavadinti tam tikrą kreivę, dinamiškai judančią per paveiksliuką ir bandančią sumažinti jo energiją.

Pagrindinė metodo idėja yra kreivės plėtojimas duotame paveikslėlyje u_0 tam, kad aptiktume objektus. Tarkime, pradinė kreivė yra aplinkui ieškomą objektą. Tuomet, kreivė judės į vidų ir turės sustoti objekto kraštuose. Pagrindinį funkcionalą apibrėžiame kaip:

$$F(c_1,c_2,C)=\mu\int_{\Omega}\delta(\phi(x,y))\,|\nabla\phi(x,y)|\,dxdy+\gamma\int_{\Omega}H(\phi(x,y))dxdy+\lambda_1\int_{\Omega}|u_0(x,y)- \tag{8}$$

$$-c_1|^2\,H(\phi(x,y))dxdy+\lambda_2\int_{\Omega}|u_0(x,y)-c_2|^2\,(1-H(\phi(x,y)))dxdy$$

kur $\delta(x)$ - Dirac matas, skaičiuojamas panašiai kaip ir Heaviside funkcija;

$\nabla\phi(x,y)$ - paveiksliuko gradientas;

H(x) – Heaviside funkcija.

c_1, c_2 – atitinkamai vidurkiai kreivės viduje (c_1) ir kreivės išorėje (c_2).

u(x,y) – paveiksliuko pikseliai

Vidurkiai c_1, c_2 yra skaičiuojami pagal duotas formules:

$$c_1(\phi) = \frac{\int_\Omega u_0(x,y)H(\phi(x,y))dxdy}{\int_\Omega H(\phi(x,y))dxdy} \tag{9}$$

$$c_2(\phi) = \frac{\int_\Omega u_0(x,y)(1-H(\phi(x,y)))dxdy}{\int_\Omega (1-H(\phi(x,y)))dxdy} \tag{10}$$

Pirmasis formulės elementas kontroliuoja kreivės reguliarumą. Antrasis elementas kontroliuoja srities kreivės viduje dydį. Trečiasis ir ketvirtasis elementai kontroliuoja skirtumus tarp įvesties paveiksliuko bei ieškomo modelio.

Pastebėkime, jog skaičiuodami funkcionalo formulę paveiksliukui, atitinkamus formulės integralus traktuojame kaip pikselių sumas.

Eulerio lygtis šiuo atveju atrodys atitinkamai:

$$\delta_0(\phi)[\mu div\left(\frac{\nabla\phi(x,y)}{|\nabla\phi(x,y)|}\right) - \gamma - \lambda_1(u_0(x,y)-c_1)^2 + \lambda_2(u_0(x,y)-c_2)^2] = 0 \tag{11}$$

Parametrų μ, γ, λ_1, λ_2 reikšmes galime atitinkamai keisti pagal paveiksliukų tipologiją. Parametrų reikšmes keičiame norėdami funkcionalą pritaikyti įvairaus tipo paveiksliukų segmentavimui. Bendrai numatytos parametrų reikšmės yra tokios:

$$\mu \geq 0,\ \gamma \geq 0,\ \lambda_1, \lambda_2 \geq 0$$

Reikėtų pastebėti, jog funkcionalas yra bendresnė Mumford Shah funkcionalo versija. Dėl ko lengvai metodą galime pakeisti į Mumford Shah segmentavimo techniką.

Taigi minimizuodami minėtojo funkcionalo formulę, turime spręsti Eulerio lygtį (11). Minimizuodami funkcionalą, kontūro pikselius perkeliame ten, kur skaičiuojama reikšmė būtų mažesnė.

Mūsų aktyvių kontūrų modelis su $\lambda_1 = \lambda_2 = 1$ ir $\gamma = 0$ yra atskiru minimalaus padalijimo problemos atveju, kuriame mes ieškome geriausios aproksimacijos u į u_0, funkcija gali turėti tik dvi reikšmes:

$$u = \begin{cases} average(u_0) inside C \\ average(u_0) outside C \end{cases}$$

Duotasis minimalaus padalijimo problemos atskiras atvejis gali būti suformuluotas ir išspręstas naudojant geodezinį aktyvių kontūrų metodą.

3. Vaizdų segmentavimo uždavinys naudojant aktyvių kontūrų modelį ir Mumford Shah funkcionalą

Segmentuodami vaizdą pirmiausia turime spręsti sudėtingą matematinį uždavinį. Norėdami išgauti objektų kraštus, turime priversti kreivę, segmentuojančią objektus, judėti paveikslėlyje objektų link ir sustoti palei aptiktus objektų kraštus. Tam tikslui minimizuosime kreivės energiją naudojant Mumford Shah funkcionalą (8). Minimizavimo problemą galėtume apibrėžti atitinkamai: $\inf_{c_1, c_2, C} F(c_1, c_2, C)$. Minimizuodami funkcionalą ieškome, kur turėtume perkelti kreivės taškus, kad gautume mažesnę funkcionalo reikšmę. Taigi šiuo atveju turime optimizavimo uždavinį – esamas reikšmes norime pakeisti mažesnėmis. Prisiminkime, jog optimizavimas – tai tikslo funkcijos f(x) minimumo ar maksimumo radimo procesas esant tam tikriems apribojimams. Net jei optimizacijos uždavinys turi nelygybių apribojimus, kaip $g_j(x) \leq 0$, jį galima pakeisti lygybių apribojimais.Tokiu būdu gauname kreivės „judėjimą“ paveikslėlyje – kreivės taškai yra perkeliami kur funkcionalo reikšmė būtų mažesnė. Reikėtų pastebėti, jog kreivės viduje funkcija turi neigiamas, o kreivės išorėje teigiamas reikšmes. Taigi minimizavimo uždavinį galėtume apibrėžti atitinkamai:

$$\mathrm{F}(c_1, c_2, C) \to \min, \text{ kai} \quad \begin{array}{l} C = \{(x, y) \in \Omega : \phi(x, y) = 0\} \\ in(C) = \{(x, y) \in \Omega : \phi(x, y) > 0\} \\ out(C) = \{(x, y) \in \Omega : \phi(x, y) < 0\} \end{array} \tag{3.1}$$

Šiuo atveju turime optimizavimą su ribojimais. Mūsų uždavinys būtų išspręsti optimizavimo uždavinį su nelyginiais apribojimais: $\phi(x, y) > 0$ ir $\Omega : \phi(x, y) < 0$. Pagrindinė uždavinio problema yra funkcionalo minimizavimas esant aukščiau išvardintiems apribojimams.

Kreivę minimizuojame pagal aukščiau išvardintą funkcionalą (8). Kaip išsiaiškinome ankstesniuose skyriuose, funkcionalo minimizavimui skaičiuojame Lagranžo lygtį (11). Pirmiausia pastebėkime, jog skaičiuojant funkcionalą paveiksliukas yra padalijamas į dvi sritis (jeigu turime vieną uždarą kreivę). Tuomet galime apskaičiuoti vidurkius c_1 ir c_2 - atitinkamai ϕ_0 viduje ir ϕ_0 išorėje. Kadangi į vidurkių formules įeina ir Heaviside funkcija, pirmiausia apskaičiuojame ją. Prisiminkime, jog Heaviside funkciją skaičiuojame pagal formulę:

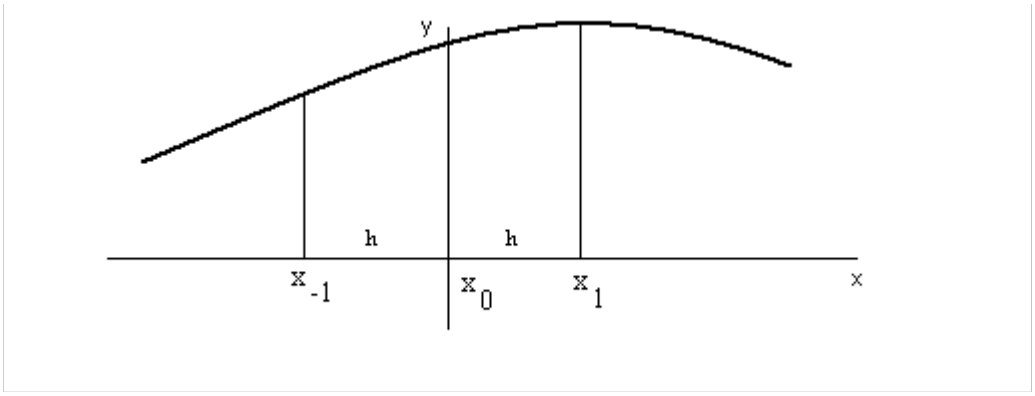
$$H(x) \approx \frac{1}{2} + \frac{1}{2}\tanh(x) = \frac{1}{1 + e^{-2x}} \tag{3.2}$$

kur tanh – hiperbolinė funkcija. Prisiminkime, kad

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3.3}$$

Nustatome reikšmę, su kuria lyginsime likusius pikselius. Tarkime, tai bus ε=0.01. Pirmiausia surandame reikšmes didesnes bei mažesnes nuo nei minėtoji reikšmė. Pikselių reikšmes didesnes už ε pakeičiame 1-tu, o mažesnės reikšmės pikseliams skaičiuojame Heaviside funkcijos reikšmes pagal aukščiau pateiktą formulę.

Apskaičiavus Heaviside funkciją galime skaičiuoti funkcionalo formulę. Kaip matome, į funkcionalo formulę taip pat įeina gradiento funkcija. Todėl patogiausia būtų pirmiausia apskaičiuoti gradientą. Tačiau apdoruojant paveiksliuko pikselius, šią funkciją apskaičiuoti nėra lengva. Tokiu atveju gradiento funkciją siūloma skaičiuoti pagal centrinę išvestinę. Prisiminkime, kas tai yra centrinė išvestinė. Tarkime, turime grafiką:



3.1 pav. Pradinis grafikas

Tuomet pirmos eilės išvestinė bus skaičiuojama:

$$f'(x_0) = \frac{f_1 - f_{-1}}{2h} + O(h^2) \tag{3.4}$$

kur $f_1 \equiv f(x_0 + h)$ (3.5)

Mūsų atveju turėsime skaičiuoti antros eilės išvestinę pagal formulę:

$$f''(x_0) \cong \frac{f'(x_0 + h/2) - f'(x_0 - h/2)}{h} = \frac{f_1 - 2f_0 + f_{-1}}{h^2} + O(h^2) \tag{3.6}$$

Reikėtų paminėti, jog labai svarbu yra centrinę išvestinę apskaičiuoti visiems pikseliams. Todėl tam, kad galėtume apskaičiuoti centrinę išvestinę visiems pikseliams (net ir kraštiniams), laikinai skaičiavimams yra siūloma prie paveiksliuko matricos prirašyti po du stulpelius ir dvi eilutes vienetų. Taip gauname detalesnius rezultatus.

Kadangi mūsų atveju turime ne vieną kintamąjį, centrinė išvestinė bus skaičiuojama pagal x‘us ir y‘us. Tam bus naudojamos formulės:

$$f_x(x, y) = \frac{f(x + h, y) - f(x - h, y)}{2h} \tag{3.7}$$

$$f_y = \frac{f(x, y+k) - f(x, y-k)}{2k} \tag{3.8}$$

$$f_{xx}(x, y) = \frac{f(x+h, y) - 2f(x, y) + f(x-h, y)}{h^2} \tag{3.9}$$

$$f_{yy} = \frac{f(x, y+k) - 2f(x, y) + f(x, y-k)}{k^2} \tag{3.10}$$

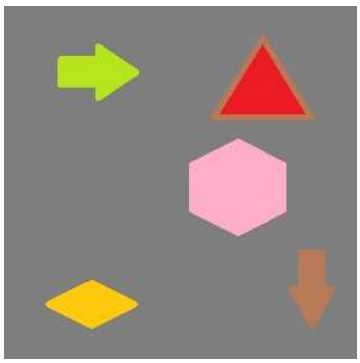
$$f_{xy}(x, y) = \frac{f(x+h, y+k) - f(x+h, y-k) - f(x-h, y+k) + f(x-h, y-k)}{4hk} \tag{3.11}$$

Turėdami Heaviside funkcijos, vidurkių cvienas ir cdu bei iš anksto nustatytų parametrų reikšmes, galime apskaičiuoti visą funkcionalo formulę. Tuomet tikriname, ar kaimyninių pikselių vietoje turėsime mažesnę funkcionalo reikšmę. Jeigu taip, tuomet perkeliame minimizuojamos kreivės taškus.

4. Algoritmo veikimo efektyvumas

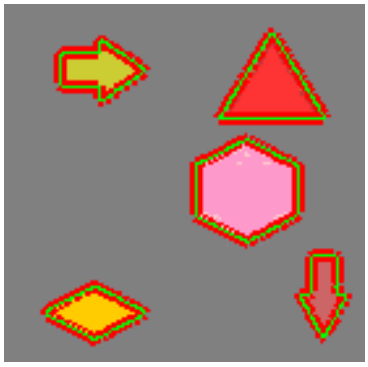
Galime išnagrinėti algoritmo veikimo efektyvumą su įvairaus tipo paveiksliukais, kad tiksliai žinotume, kokiai vaizdų klasei turime rasti tinkamesnes parametrų reikšmes. Geriausiai susegmentuojami nesudėtingi paveiksliukai kurti Paint programos pagalba bei pilko skalės paveiksliukai.

Tarkime, turime pradinį paveiksliuką:



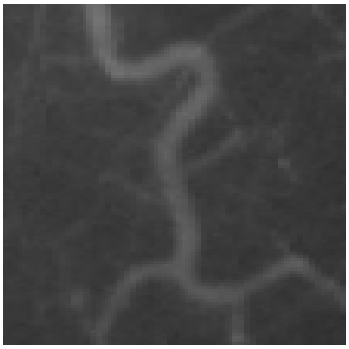
4.1 pav. Pradinis paveiksliukas

Rezultatų paveiksliukas atrodys atitinkamai:



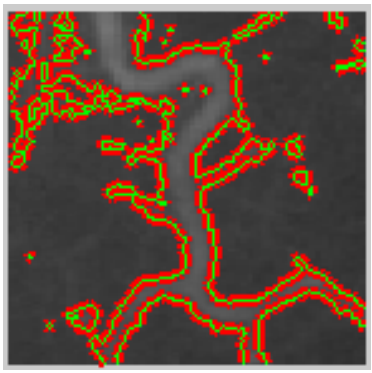
4.2 pav. Rezultatų paveiksliukas

Taip pat algoritmas neblogai įveikia kiek sudėtingesnius pilkus paveiksliukus.



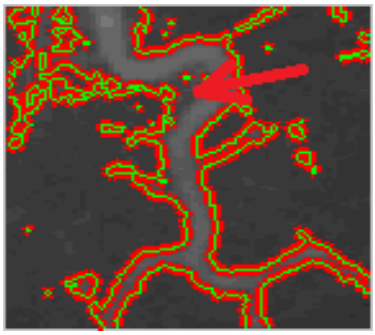
4.3 pav. Pradinis paveiksliukas

Paveiksliukas po segmentavimo:



4.4 pav. Rezultatų paveiksliukas

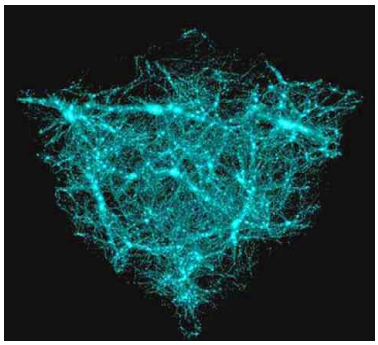
Tačiau šiuo atveju galime pastebėti nedidelį trūkumą, kad sudėtingesnio objekto kraštai nėra labai detaliai aptikti:



4.5 pav. Rezultatų paveiksliukas

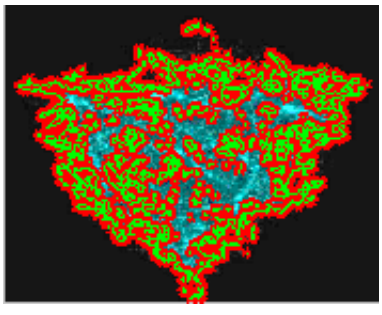
Apytikslis kai kurių objektų aptikimas gali būti nepakankamas, kadangi dažniausiai norime gauti detaliai apibrėžtus kraštus. Todėl šiuo atveju turėtume paieškoti optimalesnio funkcionalo formulės atvejo, kad gautume detalesnius segmentavimo rezultatus duotajai vaizdų klasei.

Algoritmas visai neblogai susegmentuoja paveiksliukus su triukšmu. Pavyzdžiui:



4.6 pav. Pradinis paveiksliukas

Rezultatų paveiksliukas po segmentavimo:



4.7 pav. Rezultatų paveiksliukas

Algoritmas tai pat gana gerai apdoruoja kiek sudėtingesnius spalvotus jpg paveiksliukus:

4.8 pav. Pradinis paveiksliukas

Po segmentavimo paveiksliukas atrodo atitinkamai:

4.9 pav. Rezultatų paveiksliukas

Tačiau šiuo atveju taip pat galime pastebėti vieną trūkumą. Segmentuojant nuotraukas algoritmas blogiau aptinka vidinius kraštus (kai turime objektą kitame objekte). Tą galime pastebėti susegmentuotame rezultatų paveikslėlyje:

4.10 pav. Rezultatų paveiksliukas

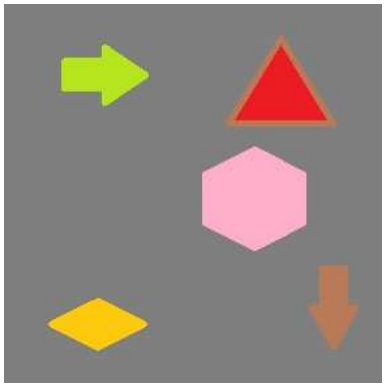
Algoritmas aptikęs tam tikrus kraštus sustojo ir nebandė ieškoti jokių vidinių kraštų/linijų objekto viduje. Šiuo atveju gavome tik objektus atskirtus nuo fono. Vidiniai objekto kraštai neliko aptikti. Išnagrinėję algoritmo veikimą matome, jog daugiausia problemų iškyla su sudėtingesnės struktūros bei blogesnės kokybės paveiksliukais, kur yra mažesni spalvų kontrastai ir kai kurie objektų kraštai yra blogiau pastebimi. Vienas didžiausių algoritmo trūkumų yra jautrumas vaizdo bei objektų kraštų tipui. Sunkiau yra segmentuojami JPG formato, spalvoti ir sudėtingesni paveiksliukai. Mūsų tikslas būtų priversti algoritmą aptikti net mažiausiai pastebimus bei sudėtingus vidinius objektų kraštus, kas duotų detalų ir tikslingą segmentavimą. Taip pat pagrindiniu tikslu būtų algoritmą pritaikyti skirtingo tipo vaizdams. Svarbu, kad algoritmas veiktų greitai bei sugebėtų detaliai apdoruoti skirtingo tipo paveiksliukus. Galime pabandyti paieškoti tinkamiausios pradinės kreivės formos bei optimaliausių parametrų reikšmių, kas pagerintų segmentavimo rezultatus.

5. Pradinės kreivės formos įtaka algoritmo efektyvumui

Kreivę, arba kitaip dar aktyvius kontūrus, naudojame objektų kraštų aptikimui. Minimizavus kreivės energiją, kreivė sustoja palei objekto kraštus taip išryškindama ieškomus objektus. Nemažai algoritmo efektyvumą įtakoja pradinės kreivės forma. Dažniausiai kreivės forma yra pasirenkama atitinkamai pagal paveiksliuko struktūrą arba objektų padėtį paveikslėlyje. Kadangi tokius dalykus iš anksto žinoti yra sunku, geru sprendimu yra surasti tokią kreivės formą, kuri tiktų daugeliui paveiksliukų.

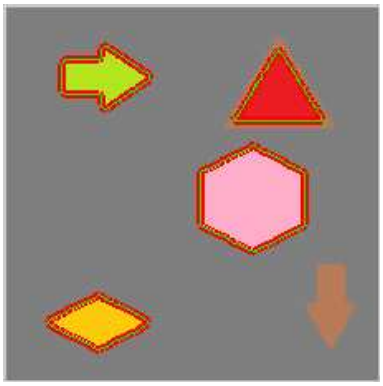
Pirmiausia turime pastebėti, jog minimizuojama kreivė būtinai turėtų būti uždara. Kadangi vykdydami algoritmą paveiksliuką padaliname į dvi sritis: kreivės vidus, kreivės išorė. Vėliau skaičiavimuose apskaičiuojame vidurkius kreivės išorėje ir kreivės viduje – atitinkamai c_1 ir c_2. Todėl minimizuojamą kreivę būtinai turime sudaryti kaip vieną uždarą kreivę arba aibę uždarų kreivių.

Išanalizuokime kreivės formos įtaką vaizdų segmentavimui. Dažniausiai naudojama kreivės forma yra vienas apskritimas išsidėtęs beveik per visą paveiksliuką. Tarkime, turime pradinį paveiksliuką:



5.1 pav. Pradinis paveiksliukas

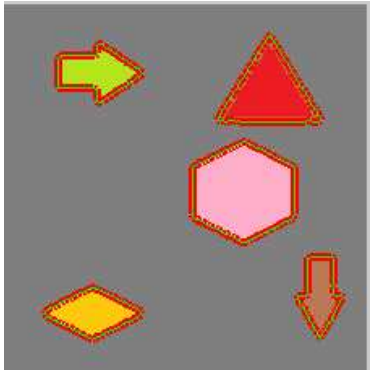
Šiuo atveju segmentavimo rezultatas būtų:



5.2 pav. Rezultatų paveiksliukas

Kaip matome, vienas objektas liko neaptiktas, todėl tokia kreivės forma nėra efektyvi visiems paveiksliukams. Tokį patį rezultatą gauname, kai pradinę kreivę sudaro du ar daugiau apskritimų.

Gera idėja yra pradinę kreivę sudaryti kaip stačiakampį, juosiantį paveiksliuko kraštus. Tuomet bus didesnė tikimybė, jog minimizuojama kreivė neaplenks jokio objekto. Taip gauname rezultatų paveiksliuką:



5.3 pav. Rezultatų paveiksliukas

Šiuo atveju gavome labai gerus segmentavimo rezultatus, todėl galime teigti, jog tokios formos pradinė kreivė puikiai tinka vaizdų segmentavimui. Tačiau netgi pakeistos parametrų reikšmės bei tinkamai parinkta pradinės kreivės formą neužtikrina detalaus objektų aptikimo sudėtingesniems jpg formato paveiksliukams. Ką galime pamatyti sekančiame pavyzdyje. Tarkime, turime pradinį paveiksliuką:

5.4 pav. Pradinis paveiksliukas

Rezultatų paveiksliukas atrodytų atitinkamai:

5.5 pav. Rezultatų paveiksliukas

Kaip matome, vėl gavome nesusegmetuotus vidinius kontūrus. Todėl šiuo atveju galime padaryti išvadas, jog sudėtingiems spalvotiems paveiksliukams arba nuotraukoms geriausiai tinka kiek „stipresnis“ segmentavimo algoritmas. Geriausiai algoritmą realizuotume pirmiausia programoje nustatydami paveiksliuko tipą bei topologiją, o vėliau atitinkamai pritaikydami kiek skirtingus segmentavimo algoritmus.

6. Algoritmo pritaikymas skirtingo tipo vaizdams

Prisiminkime, jog minimizuojamą funkcionalą galime atitinkamai pritaikyti skirtingo tipo paveiksliukams keisdami formulės parametrų μ, γ, λ_1, λ_2 reikšmes. Pasirinkę atitinkamas minėtų parametrų reikšmes galime optimizuoti funkcionalo efektyvumą atskiriems paveiksliukų tipams. Kaip jau minėjome, dažniausiai skaičiavimuose turime apibrėžtas tokias parametrų reikšmes: μ=0.2, $\lambda_1 = \lambda_2 = 1$ ir $\gamma = 0$. Su duotomis parametrų reikšmėmis gauname bendrą funkcionalo formulę, kuri apytiksliai gerai susegmentuoja visų tipų paveiksliukus. Tačiau šiuo atveju negauname detalių segmentavimo rezultatų – tam tikri paveiksliukai lieka susegmentuoti su nemaža paklaida. Toks segmentavimas yra tinkamas, kai norime maždaug nustatyti objektų kraštus arba tik atskirti objektus nuo fono. Didelė tikimybė, jog smulkesni vidiniai objektų kraštai liks neaptikti. Norėdami gauti detalius segmentavimo rezultatus turime minimizuojamą funkcionalą ir segmentavimo techniką pritaikyti prie skirtingo tipo vaizdų. Todėl geriausiu sprendimu būtų paieškoti optimaliausių parametrų reikšmių atskiriems paveiksliukų tipams. Tokiu atveju, pilkus, spalvotus ir kitokio tipo paveiksliukus segmentuotume naudodami skirtingas funkcionalo formules - tai yra formules su skirtingomis parametrų reikšmėmis. Bendros visiems paveiksliukams formulės naudojimas būtų pranašesnis tuo, jog šiuo atveju būtų žymiai lengvesnė algoritmo realizacija programoje.

Taip atsiranda dvi idėjos algoritmo realizacijai. Galime sukurti vaizdų segmentavimo programą, kurį būtų bendra visų tipų paveiksliukams – visi paveiksliukai būtų apytiksliai gerai susegmentuojami, tačiau sudėtingesnės topologijos paveiksliukai būtų segmentuojami su nedidele paklaida. Taip pat galime sukurti programą, kuri skirtingiems paveiksliuko tipams naudotų skirtingas funkcionalo formules – tai yra formules, su skirtingomis parametrų reikšmėmis. Tokiu atveju gautume žymiai detalesnį vaizdų segmentavimą.

Kaip jau minėjome, funkcionalo formulės parametrų reikšmes dažniausiai nustatome pagal vaizdo tipą. Keisdami parametrų reikšmes galime įtakoti segmentavimo rezultatus. Todėl mūsų tikslas būtų tokiu būdu optimizuoti segmentavimo rezultatus skirtingiems vaizdų tipams. Jeigu tikimės regiono su glodžiais kraštais, tuomet turime parametrą μ padaryti „sunkesni“ – tai yra nustatyti didesnę jo reikšmę. Jeigu turime binarinį paveiksliuką, kur pikseliai gali turėti tik dvi reikšmes, dažniausiai nustatome $\lambda_1 = \lambda_2 = 1$. Tačiau vieną iš elementų galime padaryti „sunkesnį“. Jeigu nustatysime parametrus $\lambda_1 = 2$ ir $\lambda_2 = 1$, turėsime labiau vientisą vidinį regioną. Reikėtų pastebėti, jog jeigu tikimės paveiksliukų su juodu fonu ir pilkais objektais, parametrus nustatome $\lambda_1 < \lambda_2$. Kaip matome, parametrų reikšmių keitimas galėtų kažkiek pagelbėti segmentuojant

įvairaus tipo paveiksliukus, tačiau neišspręstų problemos, jeigu turėtume mišraus tipo paveiksliukus bei labai sudėtingos topologijos objektus. Todėl parametrų reikšmių keitimą geriausia naudoti kaip pagalbinę priemonę prie sudėtingesnės segmentavimo technikos.

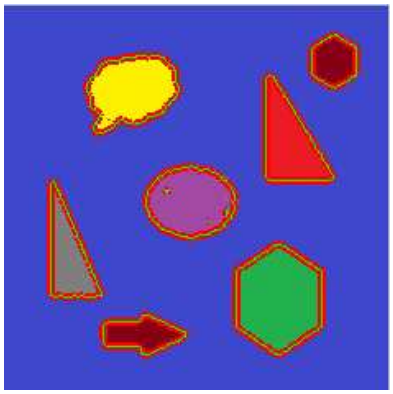
Pirmiausia galime išbandyti bendrą funkcionalo formulę, kurią galėtume pavadinti tinkančia daugumai paveiksliukų tipų ir atliekančią daugmaž gerą nors ir ne itin detalų segmentavimą. Parametrų reikšmės yra atitinkamos: $\lambda_1 = 1$, $\lambda_2 = 1$, μ=0.2, γ=0. Kaip žinome, tokios parametrų reikšmės geriausiai tinka binariniams paveiksliukams. Tačiau praktikoje tokios funkcionalo parametrų reikšmės pasirodė veiksmingos daugumai paveiksliukų tipų. Šiuo atveju segmentavimas vyksta gana greitai ir objektų kraštai yra aptinkami. Taigi šiuo atveju funkcionalo formulės realizacija programoje atrodys atitinkamai:

```
% Formulės δ0(φ)[μdiv(∇φ(x,y)/|∇φ(x,y)|)−γ−λ1(u0(x,y)−c1)^2+λ2(u0(x,y)−c2)^2], skirtos
funkcionalo skaičiavimui realizacija programoje. Pagrindiniame cikle
pirmiausia paskaičiuojame vidurkius c1 ir c2 – atitinkamai kreivės
viduje ir kreivės %išorėje. Turėdami vidurkius skaičiuojame
funkcionalą pagal aukščiau išvardytą formulę.


for n = 1:iteraciju_skaicius
    vid_indeksas = find(phi0>=0);
    isor_indeksas = find(phi0<0);
    paveiksliuko_energija = 0;
    for i=1:sluoksnis
        L = im2double(P(:,:,i));         % apdoruojamas paveiksliukas

        c1 = sum(sum(L.*Heaviside(phi0)))/  % skaičiuojame vidurkį
             /(length(vid_indeksas)+eps);   % kreivės viduje

        c2 = sum(sum(L.*(1- Heaviside(phi0))))/  %skaičiuojame vidurkį
             /(length(isor_indeksas)+eps);       % kreivės išorėje

        paveiksliuko_energija = -(L-c1).^2+(L- c2).^2+
             +paveiksliuko_energija         % skaičiuojame funkcionalą

    end

    energija = Dirac(phi0)*mu*
         *gradientas(phi0)./max(max(abs(gradientas(phi0))))-
         -1+paveiksliuko_energija;        % skaičiuojame funkcionalą

                                .......

end
```

Kaip jau anksčiau minėjome, duotoji funkcionalo formulė yra bendra visiems paveiksliukų tipams. Taigi rezultate gauname objektus atskirtus nuo fono, tačiau trūksta detalesnio objektų kraštų aptikimo – mažiau ryškios vidinės linijos dažniausiai lieka neaptiktos. Kas ypač ryškiai pasireiškia segmentuojant paveiksliukus su triukšmu arba nuotraukas. Galime teigti, jog šiuo atveju gauname visai neblogus rezultatus, tačiau duotoji funkcionalo formulė nėra optimali jokiai vaizdų klasei. Galime apžvelgti vaizdų segmentavimo rezultatus panauduojant šią funkcionalo formulę. Tarkime, turime pradinį paveiksliuką:

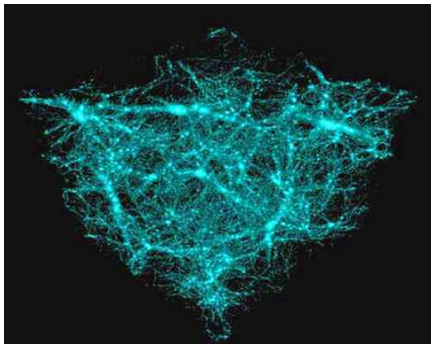
6.1 pav. Pradinis paveiksliukas

Po segmentavimo paveiksliukas atrodo atitinkamai:



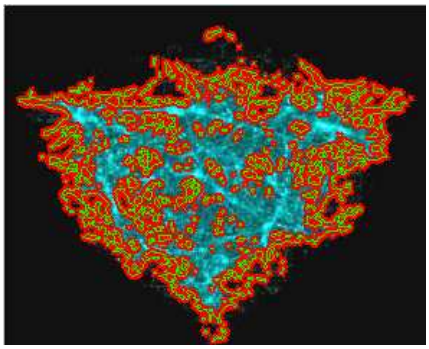
6.2 pav. Rezultatų paveiksliukas

Vizualiai matome, jog vaizdas liko pakankamai sėkmingai susegmentuotas. Tačiau turime pastebėti, jog paveiksliukas yra gana paprastas ir su aiškiai matomais objektais. Galime pabandyti susegmentuoti kiek sudėtingesnį paveiksliuką su triukšmu. Tarkime, turime paveiksliuką:



6.3 pav. Pradinis paveiksliukas

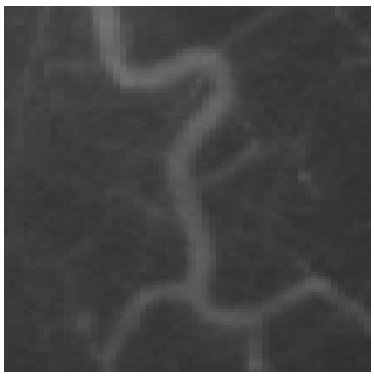
Po segmentavimo gauname rezultatą:



6.4 pav. Rezultatų paveiksliukas

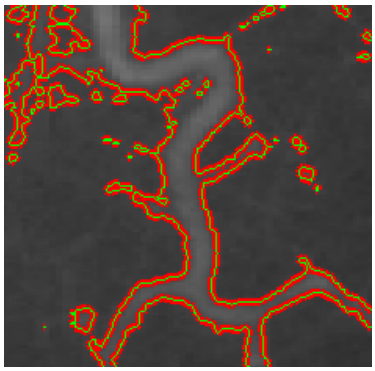
Šį kartą galime pastebėti, jog kraštų linija yra trūki bei blankesnės vidinės linijos nėra labai detaliai aptiktos. Taip pat segmentavimas vyko pakankamai lėtai.

Taip pat galime pabandyti susegmentuoti blogesnės kokybės pilkos skalės paveiksliuką. Tarkime, turime pradinį paveiksliuką:



6.5 pav. Pradinis paveiksliukas

Po segmentavimo gauname rezultatą:



6.6 pav. Rezultatų paveiksliukas

Gautame rezultatų paveikslėlyje kraštų linija kažkiek yra trūki ir ne visi objektų kraštai liko detaliai aptikti. Todėl šiuo atveju galėtume siekti geresnių segmentavimo rezultatų.

Iš gautų segmentavimo rezultatų matome, jog labiausiai reikėtų patobulinti pilkų bei „triukšmingų“ paveiksliukų segmentavimą. Tam tikslui galime išbandyti įvairias keičiamų parametrų reikšmių kombinacijas ir išrinkti tinkamiausias.

Eksperimentiniu būdu geriausią pilkų ir triukšmingų paveiksliukų segmentavimą gavome, kai parametrų reikšmės buvo atitinkamos: $\lambda_1 = 1$, $\lambda_2 = 2$, $\gamma = 1$. Tai atitinka aukščiau pateiktą apibrėžimą, jog pilkiems paveiksliukams parametrų reikšmės turėtų būti: $\lambda_1 < \lambda_2$. Taip pat pastebėkime, jog šiuo atveju turime nenulinį γ parametrą, nors daugeliu atvejų šio parametro reikšmė būna lygi 0. Nenulinis γ parametras ypač pagelbsti, kai turime pilkos skalės paveiksliukus su triukšmu bei nelabai išraiškingais kraštais. Panaudojus šį parametrą tokių paveiksliukų segmentavimas vyksta greičiau bei detaliau. Taip pat tokios parametrų reikšmės puikiai tinka, kai turime blogesnės kokybės vaizdus su nelabai aiškiai apibrėžtais kraštais. Šios funkcionalo versijos realizacija programoje atrodo atitinkamai:

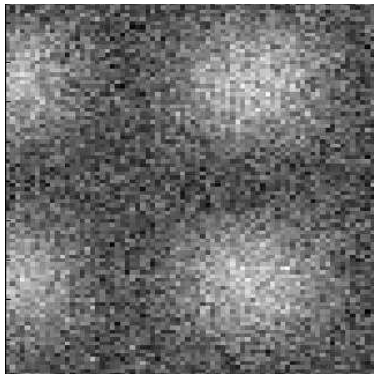
```
1.   %  Formulės δ0(φ)[μdiv(∇φ(x,y)/|∇φ(x,y)|) − γ − λ1(u0(x,y)−c1)^2 + λ2(u0(x,y)−c2)^2], skirtos
2.   %  funkcionalo skaičiavimui realizacija programoje. Pagrindiniame cikle
3.   %  pirmiausia paskaičiuojame vidurkius c1 ir c2 – atitinkamai kreivės
4.   %  viduje ir kreivės išorėje. Turėdami vidurkius skaičiuojame
5.   %  funkcionalą pagal aukščiau išvardytą formulę.
6.
7.
8.
9.    for n = 1:iteraciju_skaicius
10.       vid_indeksas = find(phi0>=0);
11.       isor_indeksas = find(phi0<0);
12.       paveiksliuko_energija = 0;
13.       for i = 1:sluoksnis
14.           L = im2double(P(:,:,i));              % įvesties paveiksliukas
15.
16.           c1 = sum(sum(L.*Heaviside(phi0)))/    % vidurkis kreivės viduje
17.                /(length(vid_indeksas)+eps);
18.
19.           c2 = sum(sum(L.*(1- Heaviside(phi0))))/   % vidurkis kreivės
20.               /(length(isor_indeksas)+eps);         % išorėje
21.
22.           paveiksliuko_energija = -(L-c1).^2+2*(L- c2).^2+
23.               +paveiksliuko_energija;          %skaičiuojame funkcionalą
24.
25.       end
26.
27.       energija = Dirac(phi0)mu*
28.              *gradientas(phi0)./max(max(abs(gradientas(phi0))))-
29.                -1+paveiksliuko_energija;      % skaičiuojame  funkcionalą
30.
31.
32.                                  .......
33.
34.    end
```

Tačiau minėtos parametrų reikšmės nelabai tinka spalvotiems paveiksliukams. Todėl spalvotiems paveiksliukams privalome paieškoti kitų parametrų reikšmių. Spalvoti paveiksliukai geriausiai susegmentuojami turint bendrąją funkcionalo formulę. Tokiu atveju parametrų reikšmės yra atitinkamos: $\lambda_1 = 1$, $\lambda_2 = 1$, $\gamma = 0$. Nors kaip pamename iš apibrėžimo, parametrų reikšmės $\lambda_1 = \lambda_2 = 1$ geriausiai tinka binariniams paveiksliukams, duotos parametrų reikšmės puikiai tiko spalvotiems paveiksliukams. Taip pat pastebėkime, jog šiuo atveju nenaudojame parametro γ, kadangi jo nulinė reikšmė pagerino segmentavimo rezultatus. Šiuo atveju algoritmo realizacija programoje atrodys atitinkamai:

```
1.   % Formulės δ0(φ)[μdiv(∇φ(x,y)/|∇φ(x,y)|) − γ − λ1(u0(x,y) − c1)^2 + λ2(u0(x,y) − c2)^2], skirtos
2.   % funkcionalo skaičiavimui realizacija programoje. Pagrindiniame cikle
3.   % pirmiausia paskaičiuojame vidurkius c1 ir c2 – atitinkamai kreivės
4.   % viduje ir kreivės %išorėje. Turėdami vidurkius skaičiuojame
5.   % funkcionalą pagal aukščiau %išvardytą formulę.
6.
7.
8.   for n=1:iteraciju_skaicius
10.      vid_indeksas = find(phi0>=0);
11.      isor_indeksas = find(phi0<0);
12.      paveiksliuko_energija = 0;
13.      for i=1:sluoksnis
14.         L = im2double(P(:,:,i));              % įvesties paveiksliukas
15.
16.         c1 = sum(sum(L.*Heaviside(phi0)))/    % vidurkis kreivės viduje
17.              /(length(vid_indeksas)+eps);
18.
19.         c2 = sum(sum(L.*(1- Heaviside(phi0))))/  % vidurkis kreivės
20.              /(length(isor_indeksas)+eps);       % isorėje
21.
22.
23.        paveiksliuko_energija=-(L-c1).^2+(L- c2).^2+
24.             +paveiksliuko_energija;          % skaičiuojame funkcionalą
25.
26.      end
27.
28.      energija = Dirac(phi0)*mu*
29.              *gradientas(phi0)./max(max(abs(gradientas(phi0))))+
30.              +paveiksliuko_energija;       % skaičiuojame funkcionalą
31.
32.
33.                                  .......
34.
35.   end
```

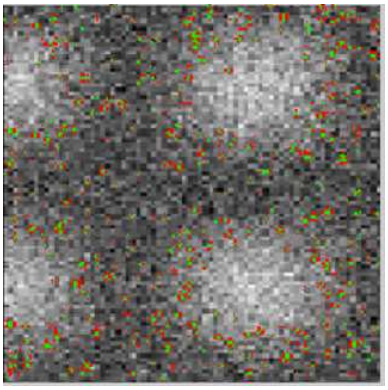
Taip pat atskirai turėtume išnagrinėti „triukšmingų“ paveiksliukų segmentavimą, kadangi tokio tipo paveiksliukai dažnai pasitaiko ir puikiai atspindi blogesnės kokybės vaizdus. Segmentavimo programa turėtų charakterizuotis sugebėjimu atskirti klasterius triukšmingame paveikslėlyje. Tai yra, programa turėtų sugebėti atskirti net objektus neturinčius aiškių kraštų. Galime įsivaizduoti paveiksliuką sudarytą iš aibės taškų, kur tam tikrose vietose taškai yra išdėstyti tankiau – sudaro klasterius. Tokiu atveju programa turėtų atskirti tas vietas vaizde, kur būtų tankesnė taškų koncentracija – klasteris. Tuomet galime tikėtis, kad programa sugebės atskirti net mažiau ryškius objektus ar objektus blogesnės kokybės vaizduose. Segmentavimas triukšminguose paveikslėliuose yra kiek sudėtingesnis, todėl programa gali lėčiau veikti, priklausomai nuo vaizdo sudėtingumo ir triukšmo lygio. Šiuo atveju rezultatas priklauso nuo visų parametrų λ_1, λ_2 μ ir γ .

Kuo parametro μ reikšmė yra didesnė, tuo detaliau yra segmentuojami minėtojo tipo paveiksliukai. Atvirkščiai yra su parametru γ - kuo mažesnė šio parametro reikšmė, tuo geresnius rezultatus gauname duotiems vaizdų tipams. Geriausiai tinkanti parametro γ reikšmė lygi 0. Taip pat segmentuodami paveiksliukus su triukšmu turime nustatyti kitokias λ_1 ir λ_2 parametrų reikšmes. Šiuo atveju turėtų būti $\lambda_1 > \lambda_2$. Geriausius segmentavimo rezultatus gavome panaudoję tokias parametrų reikšmes: λ_1=3 ir λ_2=1. Tarkime, turime triukšmingą pilkos skalės paveiksliuką, kur balti taškai sudaro klasterius:

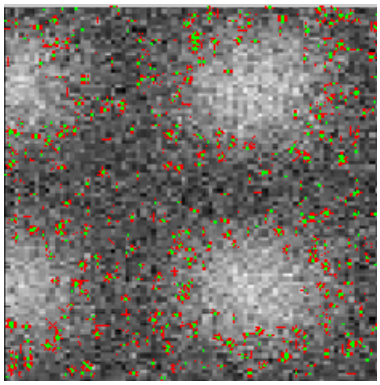


6.7 pav. Pradinis paveiksliukas

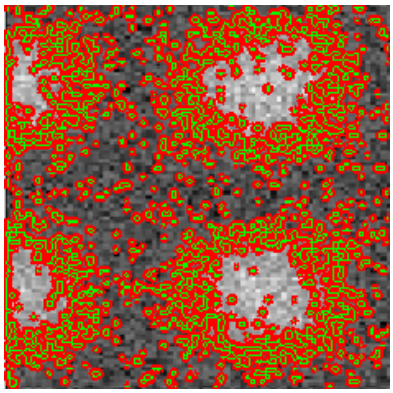
Tokiu atveju minimizuojama kreivė turėtų apibrėžti linijas aplink baltus taškus. Išnagrinėkime, segmentavimo rezultatų keitimąsi mažinant bei didinant tam tikrų parametrų reikšmes. Segmentavimas priklausomai nuo parametrų reikšmių atrodys atitinkamai:



6.8 pav. Rezultatų paveiksliukas,
kai parametrų reikšmės yra atitinkamos:
μ=0.2, γ=1, λ_1=1, λ_2=2

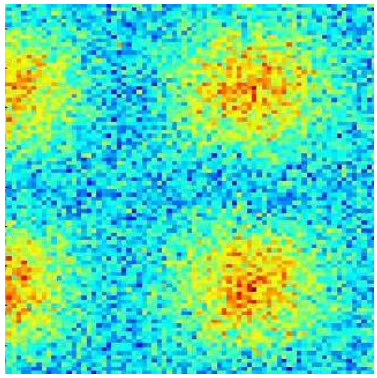


6.9 pav. Rezultatų paveiksliukas,
kai parametrų reikšmės yra atitinkamos:
$\mu = 0.3,\ \gamma = 0,\ \lambda_1 = 1,\ \lambda_2 = 1$



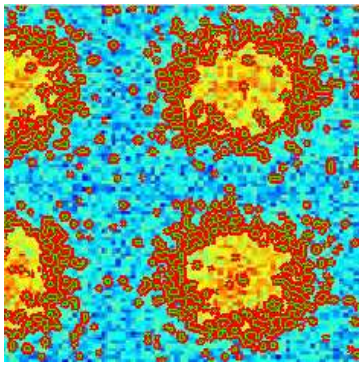
6.10 pav. Rezultatų paveiksliukas,
kai parametrų reikšmės yra atitinkamos:
$\mu = 0.5,\ \gamma = 0,\ \lambda_1 = 3,\ \lambda_2 = 1$

Kaip matome, paskutiniu atveju gavome geriausius segmentavimo rezultatus. Todėl parametrų reikšmes $\mu = 0.5,\ \gamma = 0,\ \lambda_1 = 3,\ \lambda_2 = 1$ galime laikyti tinkamiausiomis paveiksliukams su triukšmu. Reikėtų pastebėti, jog duotosios parametrų reikšmės yra bendros tiek pilkiems tiek spalvotiems triukšmingiems paveiksliukams. Tarkime, turime pradinį paveiksliuką:



6.11 pav. Pradinis paveiksliukas

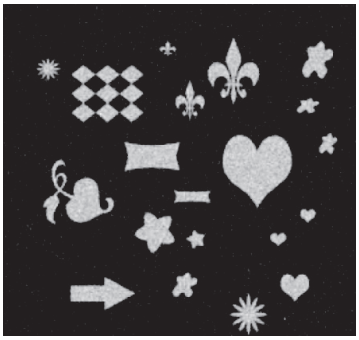
Po segmentavimo su duotomis parametrų reikšmėmis gauname rezultatą:



6.12 pav. Rezultatų paveiksliukas

Abejais atvejais baltų ir geltonų taškų klasteriai liko aptikti. Kadangi duotosios parametrų reikšmės visiškai netinka kitų tipų vaizdams, geriausia yra programoje atskirai realizuoti tokio tipo paveiksliukų segmentavimą, panaudojus tik jiems tinkamas parametrų reikšmes.

Dar vienas paveiksliukų tipas, kurį turėtume išnagrinėti, tai binariniai paveiksliukai. Kitaip paveiksliukai, kurių pikseliai gali turėti tik dvi reikšmes. Dažniausiai tai juodai balti paveiksliukai. Kadangi tokie paveiksliukai dažnai pasitaiko, svarbu kad ir šie iš pažiūros paprasti paveiksliukai būtų gerai susegmentuojami. Tarkime, turime pradinį paveiksliuką:



6.13 pav. Pradinis paveiksliukas

Naudojant spalvotiems ir pilkiems paveiksliukams skirtas parametrų reikšmes gauname rezultatus:

6.14 pav. Rezultatų paveiksliukas

Taigi binariniams paveiksliukams tinka bet kokios parametrų reikšmės, vadinasi juos galime segmentuoti naudojant bet kokiai vaizdo klasei skirta segmentavimą. Tokiu atveju šių paveiksliukų segmentavimo galime nerealizuoti programoje, kadangi bet kokiu atveju gausime gerus rezultatus.

Sunkiausiai susegmentuojamos yra nuotraukos. Segmentuodami nuotraukas dažniausiai gauname tik objektus atskirtus nuo fono, o vidiniai kraštai/linijos nėra aptinkami.

Nuotraukų segmentavimui yra siūlomi sudėtingesni algoritmai – kaip, pavyzdžiui, multifazinis segmentavimas. Tačiau tokių algoritmų veikimas yra pakankamai lėtas, kas yra nemažu trūkumu. Todėl jeigu mums svarbu, kad būtų segmentuota kažkokia paveiksliuko dalis, geru sprendimu būtų pažymėti / apkarpyti rūpimą vaizdo dalį ir ją susegmentuoti. Tokia galimybė taip pat yra realizuota programoje. Tokiu atveju dažniausiai gauname gerus teisingus rezultatus.

Kadangi jau atskyrėme kokios parametrų reikšmės geriausiai tiktų pilkiems, spalvotiems, binariniams bei su triukšmu paveiksliukams, kiekvieno tipo paveiksliuko segmentavimą realizavome programoje atskirai. Programa yra realizuota taip, kad vartotojas įkėlęs paveiksliuką gali pats pasirinkti tinkamą segmentavimo būdą pagal paveiksliuko tipą. Esant mišriems

paveiksliukams – pavyzdžiui spalvotiems su triukšmu, yra siūloma išbandyti skirtingus segmentavimo būdus ir tuomet pasirinkti vizualiai mums tinkamiausius. Kiekvieno vaizdo tipo segmentavimą realizavome atskirai norėdami optimizuoti segmentavimo rezultatus. Pasirenkant kiekvienam vaizdo tipui atskirą jam tinkantį segmentavimą, gauname optimaliausius segmentavimo rezultatus.

7. Algoritmo realizacija programoje

Kaip jau anksčiau minėjome, skirtingiems vaizdų tipams tinka skirtingos parametrų reikšmės. Ankstesniuose skyriuose atskyrėme tinkamiausius parametrus spalvotiems, pilkiems bei triukšmingiems paveiksliukams. Kadangi bendroji funkcionalo formulė neduoda optimaliausių segmentavimo rezultatų, kiekvieno vaizdo tipo segmentavimą programoje realizavome atskirai. Programa yra realizuota taip, kad vartotojas įkėlęs vaizdą pats galėtų programos meniu pasirinkti segmentavimą pagal vaizdo tipą.

Algoritmo realizacijai pirmiausia nustatome pradinę kreivę. Pradinės kreivės formą dažniausiai pasirenkame atsižvelgdami į apdoruojamų paveiksliukų struktūrą bei objektų padėtį paveikslėlyje. Tuomet skaičiuojame Mumford Shah funkcionalą (4) kreivės minimizavimui. Kreivės nauja padėtis yra atnaujinama atitinkamai pagal Mumford Shah funkcionalo reikšmę. Minimizuojamos kreivės taškus perkeliame ten, kur funkcionalo reikšmė būtų mažesnė.

Spręsdami segmentavimą šiuo metodu, pastebėkime, kad paveiksliukas yra suvokiamas kaip viena sritis – algoritmo pradžioje nevykdome paveiksliuko padalijimo į atskirus regionus. Kas palengvina ir pagreitina algoritmo vykdymą. Tik vėliau įvykdžius segmentavimą paveiksliuke yra atskiriami objektų kraštai. Funkcionalo formulė (4) programoje buvo realizuota panaudojus atitinkamą Matlab kodą. Kadangi į funkcionalo formulę įeina ne vieną sudėtinga formulė, patariama skirtingas funkcijas aprašyti skirtingose klasėse, kas buvo realizuota programoje. Tokiu atveju išvengiame painumo bei klaidų programoje. Realizuojant algoritmą buvo sukurtos šešios Matlab klasės: programos paleidimo klasė, pagrindinė klasė – skaičiuojanti funkcionalą, sustojimo sąlygas tikrinanti klasė, klasė atsakinga už kreivės evoliuciją bei dvi likusios, skirtos kitoms funkcionalo formulės funkcijoms – Heaviside ir gradientui apskaičiuoti. Taip pat reikėtų pastebėti, jog esant tokiai programos struktūrai, žymiai lengviau yra taisyti algoritmo realizavimą programoje. Bendrai programos kodą sudaro 532 eilutės

Bendrosios funkcionalo formulės (taip pat tinkančios spalvotiems vaizdams) realizacija programoje atrodytų atitinkamai:

```
1.   % Formulės δ0(φ)[μdiv(∇φ(x,y)/|∇φ(x,y)|) − γ − λ1(u0(x,y) − c1)^2 + λ2(u0(x,y) − c2)^2], skirtos
2.   % funkcionalo skaičiavimui realizacija programoje. Pagrindiniame
3.   % cikle pirmiausia paskaičiuojame vidurkius c1 ir c2 – atitinkamai
4.   % kreivės viduje ir kreivės išorėje. Turėdami vidurkius
5.   % skaičiuojame funkcionalą pagal aukščiau išvardytą formulę.
6.
7.
8.
9.    for n=1:iteraciju_skaicius
10.      vid_indeksas = find(phi0>=0);
11.      isor_indeksas = find(phi0<0);
12.      paveiksliuko_energija = 0;
13.      for i=1:sluoksnis
14.          L = im2double(P(:,:,i));           % įvesties paveiksliukas
15.
16.          c1 = sum(sum(L.*Heaviside(phi0)))/   %vidurkis kreivės viduje
17.              /(length(vid_indeksas)+eps);
18.
19.          c2 = sum(sum(L.*(1- Heaviside(phi0))))/  %vidurkis kreivės
20.               /(length(isor_indeksas)+eps);       %išorėje
21.
22.          paveiksliuko_energija=-(L-c1).^2+(L- c2).^2+
23.               +paveiksliuko_energija;       %skaičiuojame funkcionalą
24.
25.      end
26.
27.      energija = mu*gradientas(phi0)./max(max(abs(gradientas(phi0))))+
28.           +paveiksliuko_energija;         % skaičiuojame funkcionalą
29.
30.
31.                                .......
32.
33.   end
```

Reikėtų pastebėti, jog šis programos kodo fragmentas kažkiek skiriasi skirtingoms vaizdų klasėms. O tiksliau skiriasi skaičiuojamo funkcionalo formulės parametrų reikšmės.

Kaip matome, vykdydami kreivės minimizavimo ciklą, pirmiausia paveiksliuko pikselius pakeičiame į double skaičius, kad skaičiavimas būtų detalesnis. Vėliau skaičiuojame vidurkius c_1 ir c_2 atitinkamai kreivės viduje ir išorėje. Tuomet apskaičiuojame pilną funkcionalo formulę. Pagal gautą funkcionalo formulę vykdome kreivės energijos minimizavimą.

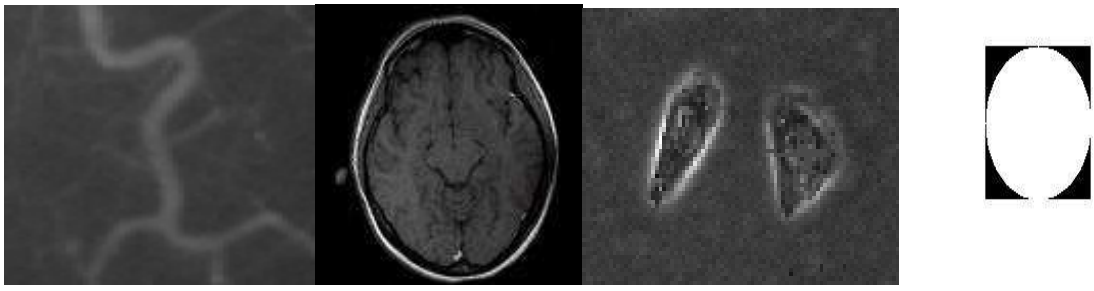
Svarbi vieta programoje yra sustojimo sąlygos tikrinimas. Programa veikia per užduotą skaičių iteracijų, tačiau po kiekvienos iteracijos yra tikrinama ar nėra tenkinamos sustojimo sąlygos – tai yra, ar dar turime kur minimizuoti kreivės energiją.

8. Statistiniai rezultatai ir algoritmo efektyvumas

Išnagrinėję visas funkcionalo formulės parametrų reikšmių konfigūracijas bei parinkę tinkamas kiekvienai vaizdų klasei, rezultatų pagerėjimu turime įsitikinti atlikę detalius skaičiavimus. Tam tikrų parametrų reikšmių tinkamumą geriausia pagrįsti statistiniu tyrimu, skaičiuojant segmentavimo paklaidas. Tokiu atveju turime galimybę kažkiek dar pataisyti funkcionalo formules arba įsitikinti esamų parametrų reikšmių tinkamumu tam tikrai vaizdų klasei. Taip pat galėsime įsitikinti kai kurių parametrų įtaka segmentavimo rezultatams.

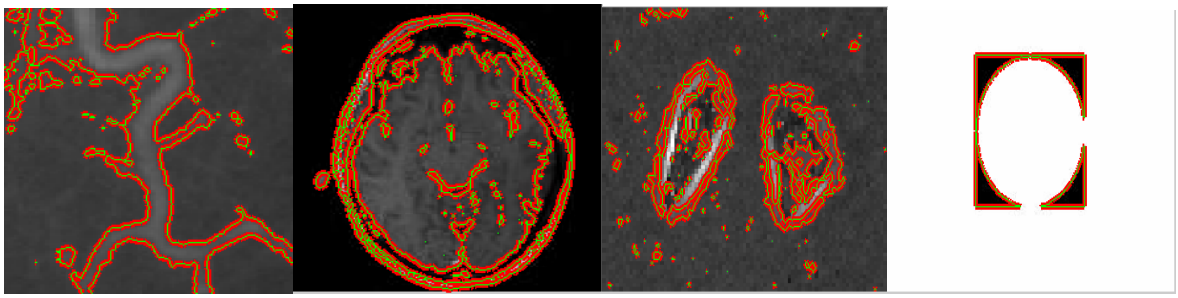
Norėdami segmentavimo rezultatus patikrinti statistiniu būdu, gautus rezultatus lyginsime su pavyzdiniais paveiksliukais, kur iš anksto tiksliai žinome kurioje vietoje bus kraštai. Pavyzdinius paveiksliukus galime pagaminti su paveiksliukų generatoriumi (Matlab programavimo kalba rašyta programa) arba grafine programa. Tokiu atveju tiksliai žinosime, kur turi būti kraštai. Turėdami du paveiksliukus – segmentavimo rezultatą bei paveiksliuką su žinomais kraštais, skaičiuojame euklidinį atstumą tarp šių paveiksliukų. Kadangi gautas euklidinis atstumas dažniausiai yra gana didelis skaičius, todėl norėdami normalizuoti gautą rezultatą, padaliname jį iš pikselių skaičiaus. Jeigu vis dar gauname nemažą skaičių, galime jį padalinti iš pikselių skaičiaus pakelto kvadratu. Taip gausime segmentavimo paklaidas naudojant tam tikras parametrų reikšmes. Skaičiuodami euklidinį atstumą, galime pagrįsti tam tikrų parametrų tinkamumą duotajai paveiksliukų klasei. Taip galime tiksliai nustatyti kokios parametrų reikšmės būtų tinkamiausios kokio tipo vaizdams bei jas pakeisti. Statistiniam tyrimui buvo panaudota po keliasdešimt paveiksliukų. Keli pavyzdiniai rezultatai pateikti darbe.

Kaip jau anksčiau minėjome, segmentavimo rezultatai skirtingiems paveiksliukų tipams labiausiai priklauso nuo nustatytų funkcionalo formulės parametrų reikšmių. Galime keisti keturių parametrų reikšmes: λ_1, λ_2, μ, γ taip įtakodami segmentavimo rezultatus. Ankstesniame skyriuje paveiksliukus suskaidėme į pilkų, spalvotų bei triukšmingų vaizdų klases. Pilkų paveiksliukų segmentavimas geriausiai vyksta, kai parametrams priskiriame reikšmes: λ_1=1, λ_2=2, μ=0.2, γ=1. Esant išvardintoms parametrų reikšmėms, gauname vizualiai geriausius pilkų paveiksliukų segmentavimo rezultatus. Pabandykime tą pagrįsti statistiniu tyrimu. Tarkime, turime pilkos skalės pradinius paveiksliukus:



8.1 pav. Pradiniai paveiksliukai

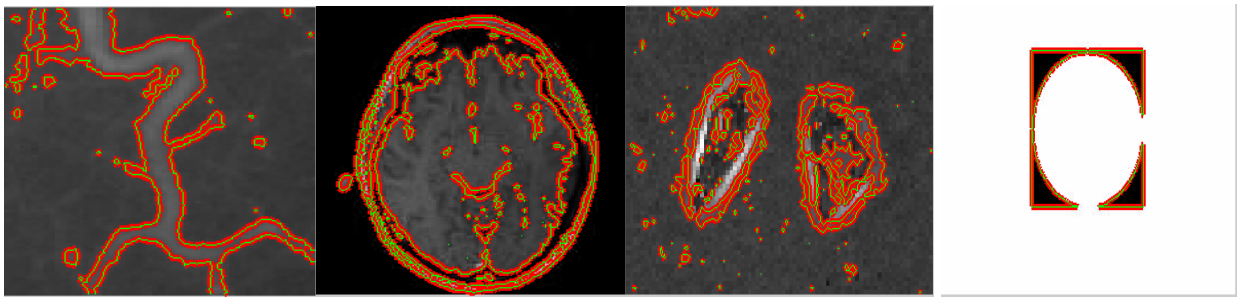
Panaudojus bendrosios formulės parametrų reikšmes $\lambda_1=1$, $\lambda_2=1$, $\mu=0.2$, $\gamma=0$, gauname rezultatus:



8.2 pav. Rezultatų paveiksliukai

Euklidinio atstumo vidurkis yra lygus: 0.59 .

Segmentavimo rezultatai naudojant parametrų reikšmes $\lambda_1=1$, $\lambda_2=2$, $\mu=0.2$, $\gamma=1$ atrodo atitinkamai:



8.3 pav. Rezultatų paveiksliukai

Euklidinio atstumo vidurkis yra lygus 0.43.

Geresnius segmentavimo rezultatus gauname padidinę λ_2 ir γ parametrų reikšmes. Tačiau reikėtų pabrėžti, jog dar labiau didinant išvardintų parametrų reikšmes geresnių rezultatų

negauname – rezultatų pagerėjimas tarsi sustoja. Todėl šias parametrų reikšmes galėtume laikyti tinkamiausiomis. Kaip matome, šiuo atveju segmentavimo rezultatai labiausiai priklauso nuo λ_2 ir γ parametrų reikšmių. Todėl galime sudaryti segmentavimo rezultatų priklausomybės nuo šių parametrų reikšmių lentelę. Kuo mažesnė segmentavimo paklaida, tuo geresni segmentavimo rezultatai. Parametrų reikšmių ir segmentavimo rezultatų lentelė atrodytų atitinkamai:

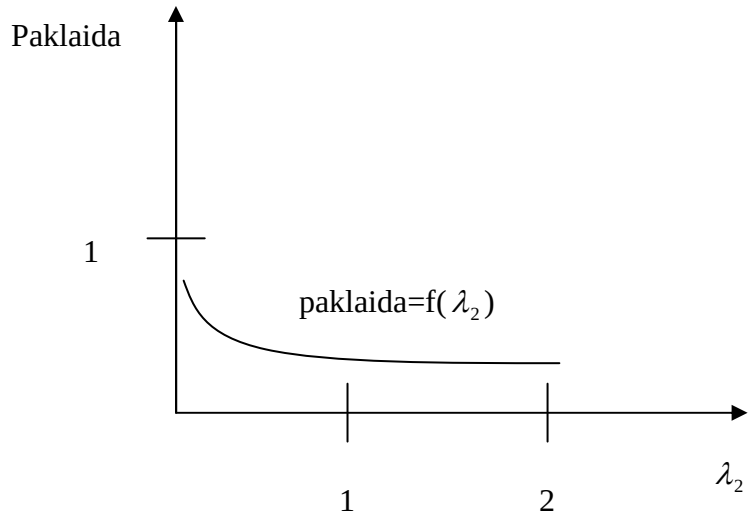
λ_2	**Segmentavimo paklaidos vidurkis**
0	0.732
0.1	0.742
0.2	0.73
0.5	0.68
1	0.59
2	0.43
2.5	0.43

8.1 lent. Segmentavimo paklaidos vidurkiai priklausomai nuo λ_2 parametro

γ	**Segmentavimo paklaidos vidurkis**
0	0.46
0.1	0.4001
0.2	0.3002
0.3	0.3002
0.5	0.3
1	0.18
2	0.18
2.5	0.18

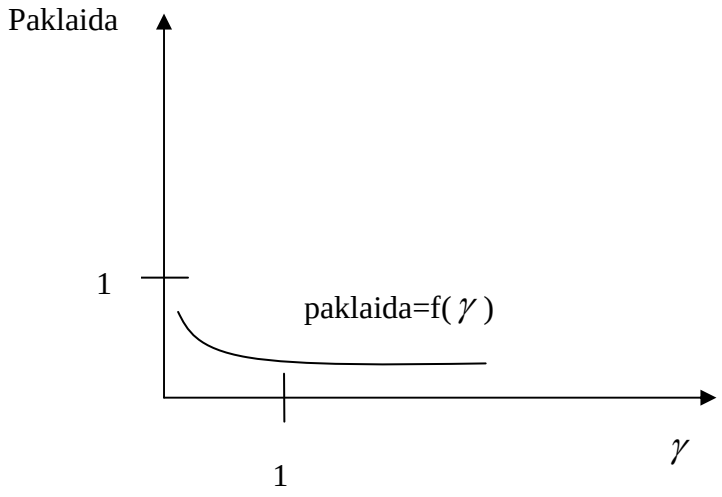
8.2 lent. Segmentavimo paklaidos vidurkiai priklausomai nuo γ parametro

Pilkų paveiksliukų segmentavimo rezultatų priklausomybę nuo parametrų λ_2 ir γ galime pavaizduoti grafiškai:



8.4 pav. Segmentavimo paklaida priklausomai nuo parametro λ_2

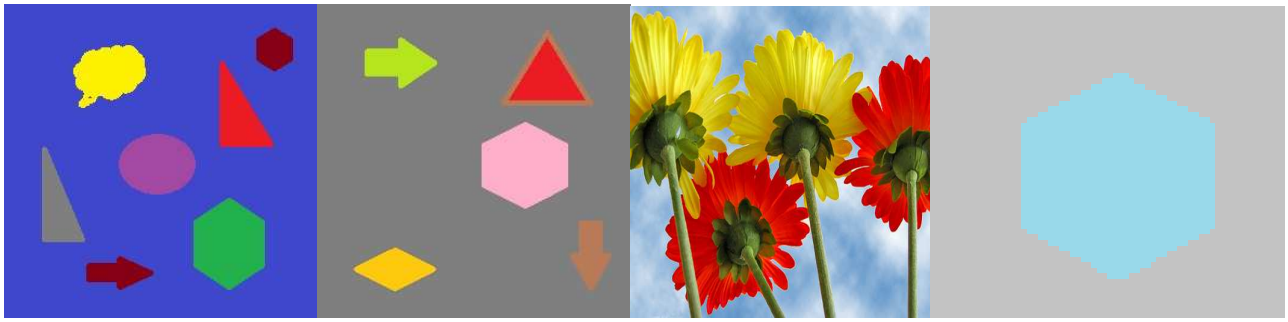
Panašų grafiką gauname parametro γ įtakai:



8.5 pav. Segmentavimo paklaidos priklausomybė nuo parametro γ

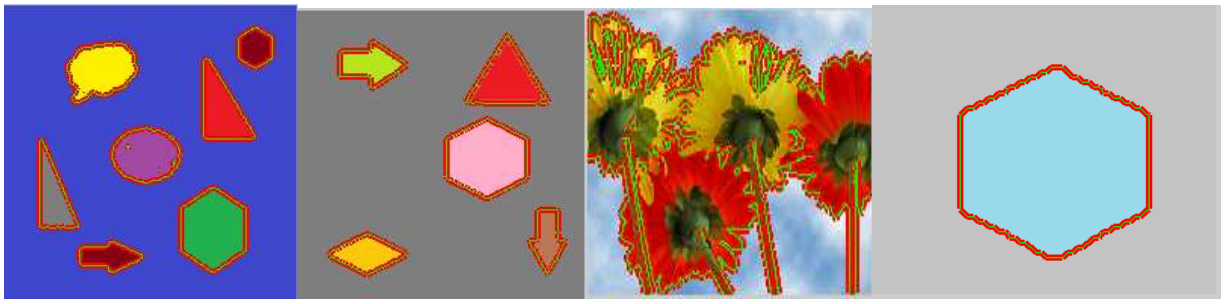
Kaip matome, padidinus parametrų γ ir λ_2 reikšmes mažėja segmentavimo paklaida.

Panašiai galime išanalizuoti spalvotų vaizdų klasę. Tarkime, turime pradinius paveiksliukus:



8.7 pav. Pradiniai paveiksliukai

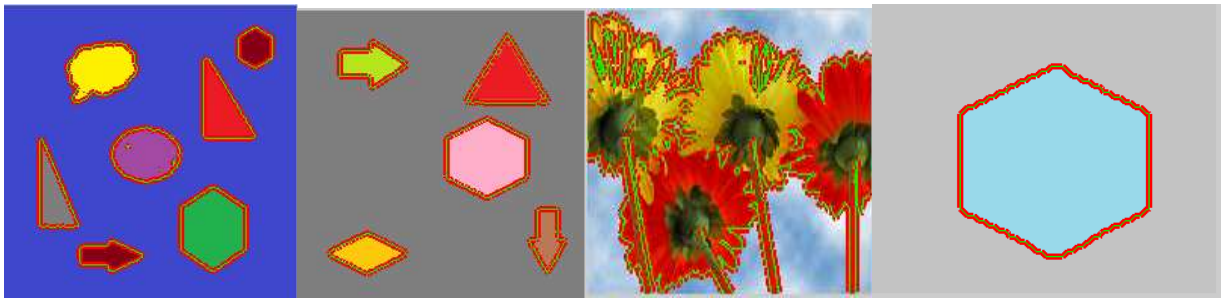
Segmentavimo rezultatai panaudojus bendrosios funkcionalo formulės parametrus λ_1=1, λ_2=1, μ=0.2, γ=0, atrodo atitinkamai:



8.8 pav. Rezultatų paveiksliukai

Euklidinio atstumo vidurkis yra lygus 0.52. Naudojant pilkiems paveiksliukams skirtus parametrus gauname euklidinio atstumo vidurkį lygu 0.58.

Taip pat galime pabandyti spalvotus vaizdus susegmentuoti su pilkiems paveiksliukams skirtomis parametrų reikšmėmis. Tokiu atveju gauname segmentavimo rezultatus:



8.9 pav. Rezultatų paveiksliukai

Vizualiai gauname panašius rezultatus. Tačiau euklidinių atstumų vidurkis yra lygus 0.599. Taigi kaip matome, spalvotiems paveiksliukams labiausiai tinka bendrosios funkcionalo formulės parametrų reikšmės: λ_1=1, λ_2=1, μ=0.2, γ=0. Taip pat galime pateikti statistinių tyrimų lenteles:

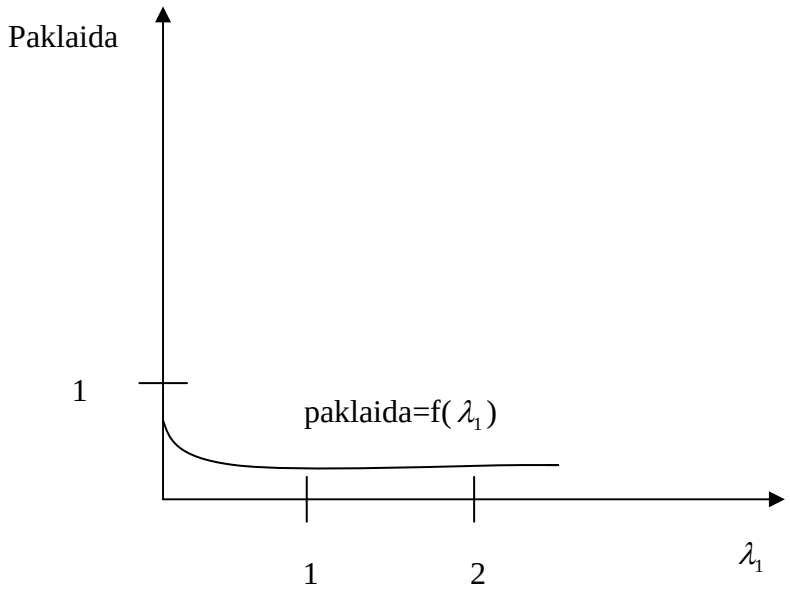
λ_1	**Segmentavimo paklaidos vidurkis**
0	0.32
0.1	0.3002
0.2	0.3002
0.5	0.2801
1	0.19
2	0.19
2.5	0.19

8.3 lent. Segmentavimo paklaidos vidurkiai priklausomai nuo λ_1 parametro

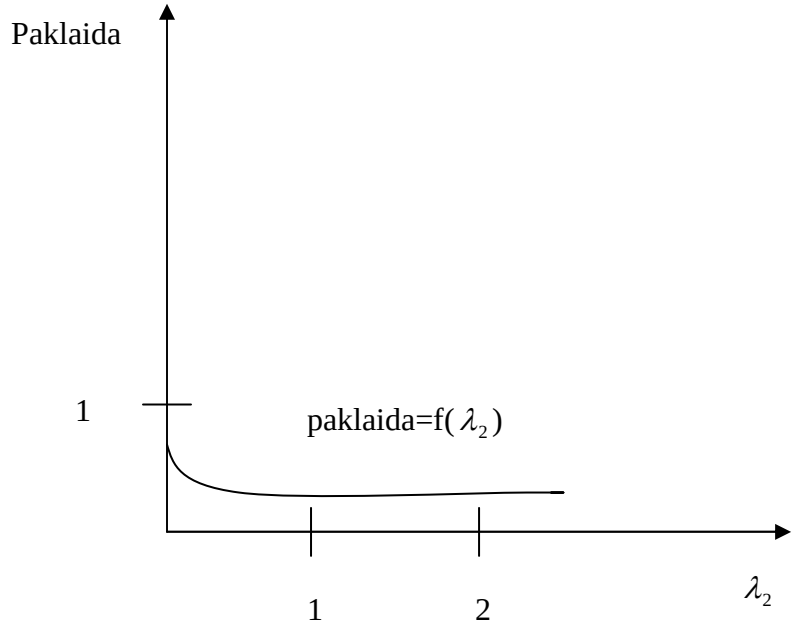
λ_2	**Segmentavimo paklaidos vidurkis**
0	0.32
0.1	0.3002
0.2	0.3002
0.5	0.2801
1	0.19
2	0.19
2.5	0.19

8.4 lent. Segmentavimo paklaidos vidurkiai priklausomai nuo λ_2 parametro

Taigi paklaidos grafikai atrodys atitinkamai:

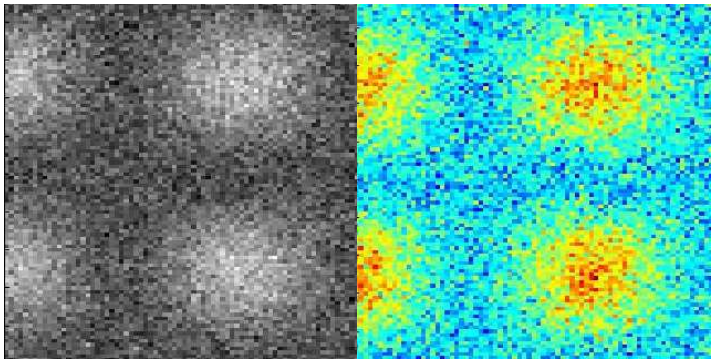


8.10 pav. Segmentavimo kokybės priklausomybė nuo parametro λ_2



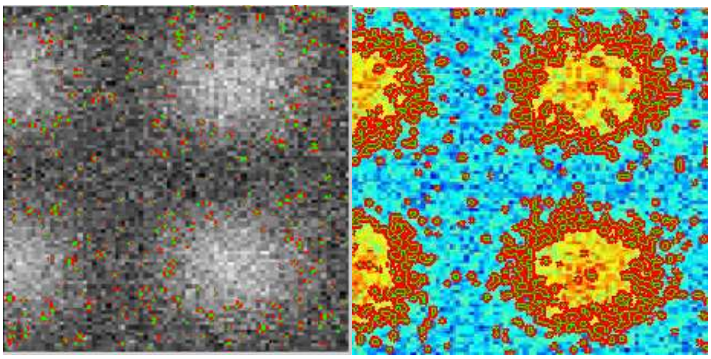
8.11 pav. Segmentavimo paklaido priklausomybė nuo parametrų λ_2

Atskirai turėtume išnagrinėti triukšmingų paveiksliukų segmentavimą, kadangi juos priskyrėme skirtingai vaizdų klasei. Turėdami tiek pilką tiek spalvotą paveiksliuką su triukšmu, negalime naudoti segmentavimo skirto pilkiems arba spalvotiems paveiksliukams, kadangi negausime gerų rezultatų. Kaip apibrėžėme praeitame skyriuje, triukšmingiems paveiksliukams labiausiai tinkančios parametrų reikšmės yra: $\lambda_1=2,\ \lambda_2=1,\ \mu=0.5,\ \gamma=0$. Tarkime, turime pradinius paveiksliukus su triukšmu:



8.12 pav. Pradiniai paveiksliukai

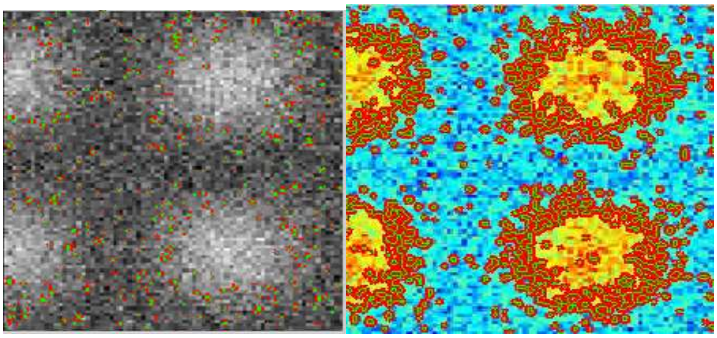
Pirmiausia, segmentavimui išbandome parametrų reikšmes skirtas pilkiems paveiksliukams: $\lambda_1=1,\ \lambda_2=2,\ \mu=0.2,\ \gamma=1$. Gauname rezultatus:



8.13 pav. Rezultatų paveiksliukai

Euklidinio atstumo vidurkis yra lygus 0.78.

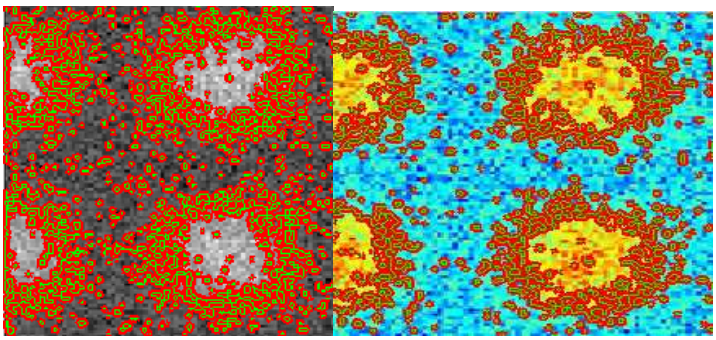
Panaudojus parametrų reikšmes skirtas spalvotiems paveiksliukams: $\lambda_1=1,\ \lambda_2=1,\ \mu=0.2,\ \gamma=0$ gauname rezultatus:



8.14 pav. Rezultatų paveiksliukai

Euklidinių atstumų vidurkis yra lygus: 0.71.

Nustačius parametrų reikšmes λ_1=2, λ_2=1, μ=0.5, γ=0 gauname rezultatus:



8.15 pav. Segmentavimo rezultatai

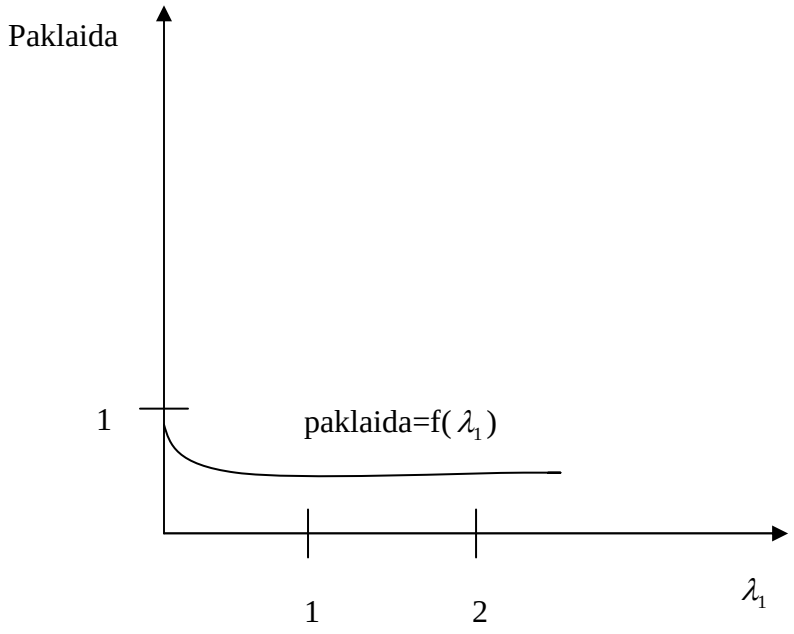
Euklidinių atstumų vidurkis yra lygus 0.511. Ką galime laikyti pagerėjusiais rezultatais. Kaip matome, gavome ryškiai apibrėžtus klasterius, ką galime laikyti sėkmingais rezultatais.

Statistinių tyrimo rezultatus galime pavaizduoti lentelėje:

λ_1	**Segmentavimo paklaidos vidurkis**
0	0.894
0.1	0.892
0.2	0.8901
0.5	0.724
1	0.71
2	0.511
2.5	0.511

8.4 lent. Segmentavimo paklaidos vidurkiai priklausomai nuo λ_2 parametro

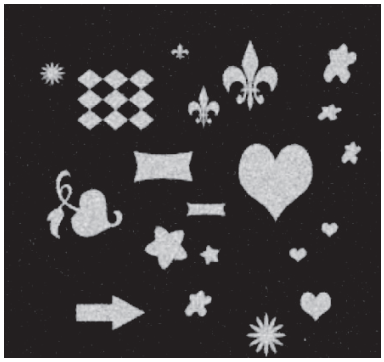
Parametrų įtaką galime pavaizduoti grafike:



8.16 pav. Segmentavimo paklaidos priklausomybė nuo parametro λ_1

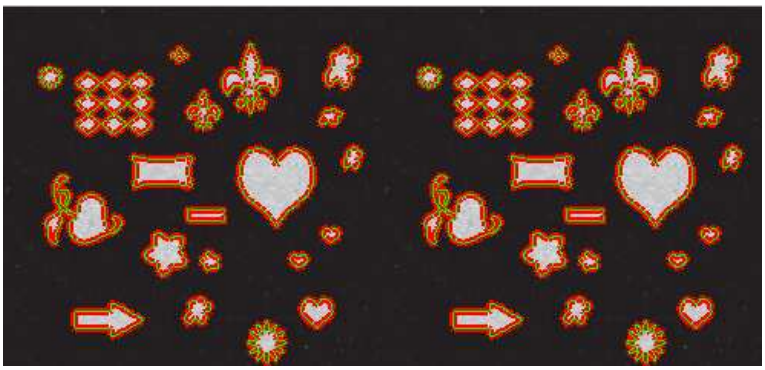
Taip pat turime dar vieną vaizdų klasę - binarinius paveiksliukus, kurių pikseliai gali įgyti tik dvi reikšmes. Dažniausiai tai būna baltai juodi paveiksliukai. Reikėtų pastebėti jog šių gana paprastų paveiksliukų segmentavimo rezultatai yra mažiau priklausomi nuo parametrų reikšmių. Kaip jau anksčiau minėjome, šio tipo vaizdai gali būti sėkmingai segmentuojami ir su kitoms

vaizdų klasėms tinkamomis parametrų reikšmėmis. Įsitikinkime tuo apžvelgdami pavyzdį. Tarkime, turime pradinį paveiksliuką:



8.17 pav. Pradinis paveiksliukai

Naudojant spalvotiems ir pilkiems paveiksliukams skirtas parametrų reikšmes gauname rezultatus:

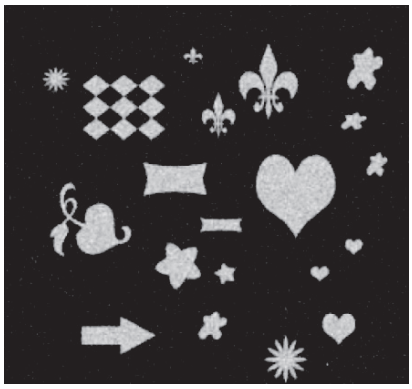


8.18 pav. Rezultato paveiksliukai

Kaip matome, gavome beveik vienodus segmentavimo rezultatus. Ką taip pat patvirtino euklidinio atstumo rezultatai. Tokio tipo vaizdai yra gerai susegmentuojami naudojant nepriklausomai ar spalvotiems ar pilkiems paveiksliukams skirtas funkcionalo formules.

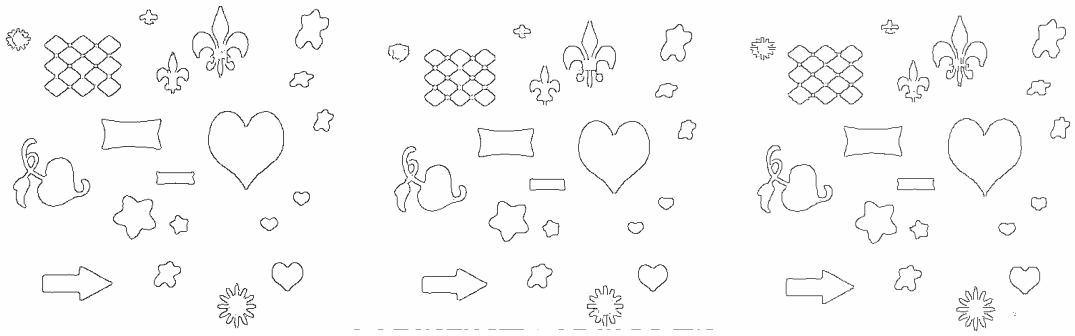
Taip pat svarbiu aspektu yra algoritmo veikimo greitis. Dažniausiai algoritmo įvykdymo laikas priklauso nuo užduotų iteracijų skaičiaus. Programoje naudojame dvidešimt iteracijų, kas yra užtektinai kiekvienam paveiksliuko tipui. Kai kurie paveiksliukai yra susegmentuojami netgi greičiau, tačiau detalumo dėlei galime įvykdyti pora iteracijų daugiau. Reikėtų paminėti, jog naudojant per daug iteracijų taip pat galime gauti blogesnius rezultatus. Šiuo atveju dvidešimt yra optimaliausiu iteracijų skaičiumi.

Norėdami pagrįsti algoritmo efektyvumą, galime jo gautus rezultatus palyginti su kitų algoritmų rezultatais. Tarkime, turime pradinį paveiksliuką:



8.19 pav. Pradinis paveiksliukai

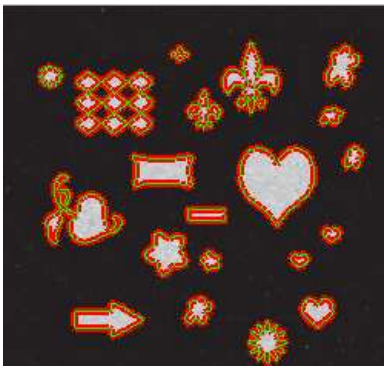
Naudojant kitus algoritmus objektų aptikimas atrodo atitinkamai:



Bessel filtras Jacob-Unser filtras Rao-Ben-Arie filtras

8.20 pav. Rezultatų paveiksliukai

Segmentavimas rezultatas naudojant aktyvių kontūrų metodą ir Mumfod-Shah funkcionalą atrodo atitinkamai:



8.21 pav. Rezultato paveiksliukai

Kaip matome, naudojant nagrinėjamą metodą vizualiai gauname panašius ir net kažkiek detalesnius rezultatus negu naudojant kitus kraštų aptikimo filtrus. Turint uomenyje, kad algoritmas veikia gana greitai bei galime jį pritaikyti bet kokio tipo paveiksliukui, algoritmą galime vadinti efektyviu.

Išvados

Darbe yra išanalizuotas vaizdų segmentavimas aktyvių kontūrų metodu. Metode naudojamas Mumford Shah funkcionalas, kuris laikomas efektyviu vaizdų segmentavimo įrankiu. Vienu didžiausiu funkcionalo privalumu yra galimybė pritaikyti skirtingų vaizdų klasių segmentavimui, parenkant atitinkamas parametrų reikšmes. Darbe išsamiai išnagrinėtas parametrų reikšmių parinkimas skirtingiems paveiksliukų tipams, taip siekiant gauti geriausius segmentavimo rezultatus. Pagal pateiktą algoritmo pritaikymo skirtingo tipo vaizdams idėją yra realizuota vaizdų segmentavimo programa.

Norėdami gauti gerus bei kokybiškus segmentavimo rezultatus, pirmiausia segmentavimo metodą turime pritaikyti bet kokio tipo vaizdams. Duotojo metodo segmentavimo rezultatus galime įtakoti keisdami funkcionalo formulės parametrų reikšmes. Tą galime panaudoti metodo efektyvumo pagerinimui bei pritaikymui tam tikrai vaizdų klasei. Tam tikslui pirmiausia vaizdus suskaidome į skirtingas klases – spalvotus, pilkus, triukšmingus. Atrinkę optimaliausias parametrų reikšmes kiekvienam paveiksliuko tipui, skirtingo tipo vaizdus segmentuojame atskirai, naudodami tinkamiausias jiems parametrų reikšmes. Svarbu yra turėti uomenyje, jog kai kurios parametrų reikšmės gali būti tinkamos keliems paveiksliukų tipams. Į tai turėtume atsižvelgti realizuodami metodą programoje. Taip pat svarbu yra parametrų reikšmių tinkamumą ištirti statistiniu būdu, kad tiksliai žinotume kokias parametrų reikšmes turėdami gauname geriausius segmentavimo rezultatus bei mažiausias paklaidas. Kas buvo išnagrinėta darbe.

Metodo efektyvumą taip pat įtakoja pradinės kreivės inicializacija. Pradinę kreivę galime nustatyti bet kokios formos. Tačiau geriausia kreivę nustatyti stačiakampio formos palei paveiksliuko kraštus – juosiančia visą paveiksliuką. Tokiu atveju didesnė tikimybė, kad net smulkiausi objektai esantys paveiksliuko kraštuose bus aptikti.

Palyginus išnagrinėto algoritmo segmentavimo rezultatus su kitų populiarių vaizdų segmentavimo algoritmų rezultatais, matome, jog duotasis algoritmas yra ne mažiau efektyvus. Kai kuriais atvejais gauname vizualiai net detalesnius rezultatus. Taip pat algoritmas išsiskiria greitu įvykdymu bei galimybe pritaikyti jį skirtingoms vaizdų klasėms. Vienu didžiausiu algoritmo pranašumu yra galimybė pritaikyti jį skirtingiems paveiksliukų tipams.

Metodas realizuotas programiškai taip, kad vartotojas pats galėtų pasirinkti įkelto vaizdo segmentavimą pagal jo tipą. Taip pat reikėtų paminėti, jog kartais gali pasitaikyti mišrūs paveiksliukai – tarkime, spalvotas paveiksliukas su triukšmu. Tokiu atveju galime pabandyti vaizdą susegmentuoti ir kaip spalvotą paveiksliuką ir kaip paveiksliuką su triukšmu. Tuomet galėsime pasirinkti geriausius mūsų manymu rezultatus.

Realizuojant algoritmą programiškai buvo sukurtos šešios Matlab klasės: programos paleidimo klasė, pagrindinė klasė – skaičiuojanti funkcionalą, sustojimo sąlygas tikrinanti klasė, klasė atsakinga už kreivės evoliuciją bei dvi likusios, skirtos kitoms funkcionalo formulės funkcijoms – Heaviside ir gradientui apskaičiuoti. Taip pat reikėtų pastebėti, jog esant tokiai programos struktūrai, žymiai lengviau yra taisyti algoritmo realizavimą programoje. Bendrai programos kodą sudaro 532 eilutės.

Ištyrinėję atskirus segmentavimo atvejus ir suradę optimaliausias parametrų reikšmes kiekvienai vaizdų klasei, galime teigti, jog įgyvendinome šio darbo tikslus. Realizavus programiškai darbe pateiktas tezes apie tam tikrų parametrų reikšmių tinkamumą atskiroms vaizdų klasėms, gauname optimaliausius segmentavimo rezultatus. Darbas galėtų būti geru pagrindu norint algoritmą dar labiau optimizuoti – sutrumpinti jo vykdymo laiką bei pritaikyti dar sudėtingesnės struktūros vaizdams. Kadangi algoritmo efektyvumas liko išsamiai išnagrinėtas, darbas galėtų būti naudojamas norint duotąjį segmentavimo algoritmą palyginti su kitais.

Literatūra

[1] Tony F. Chan, Luminita A. Veese. *Actine Contours Without Edges.* 2001

[2] Ryo Takei. *Active Contours Without Edges And Image Segmentation.* Final Project APMA, 2001

[3] Chen Xu-Feng, Guan Zhi-Cheng. *Image segmentation based on Mumford-Shah functional*. Department of Mathematics, Zhejiang University, China 2002, p. 124-128

[4] Russell Valentine. *Image segmentation with the Mumford-Shah functional*. 2007, p. 1-14

[5] Vladimir Kluzner, Gershon Wolansky, Yehoshua Y. Zeevi. *A Geometric-Functional Based Image Segmentation and Inpainting*. Mathematics Department, Technion 2002, p.166-177

[6] Yingjie Zhang. *Fast Segmentation for the Piecewise Smooth Mumford-Shah Functional*. International Journal of Information and Communication Engineering, 2006, p. 245-250

[7] Ying Zhang, Li-ling Ge. *Improving Image Segmentation Performance via Edge Preserving Regularization.* World Academy of Science, Engineering and Technology 2006, p. 169-175

[8] Andy Tsai, Anthony Yezzi. *Curve Evolution Implementation of the Mumford–Shah Functional for Image Segmentation.* Transactions on Image Processing , No. 8, August 2001

[9] Jung-ha An and Yunmei Chen. *Region Based Image Segmentation using a Modified Mumford-Shah Algorithm.* Institute for Mathematics and its Applications (IMA), University of Minnesota, USA, Department of Mathematics, University of Florida, USA

[10] Thomas Brox and Joachim Weickert. *Level Set Based Image Segmentation with Multiple Regions.* In Pattern Recognition, Springer LNCS 3175, Germany 2004, p. 415-423

[11] Yingjie Zhang. *Fast Segmentation for the Piecewise Smooth Mumford-Shah Functional.* International Journal of Information and Communication Engineering, p. 2-4, 2006

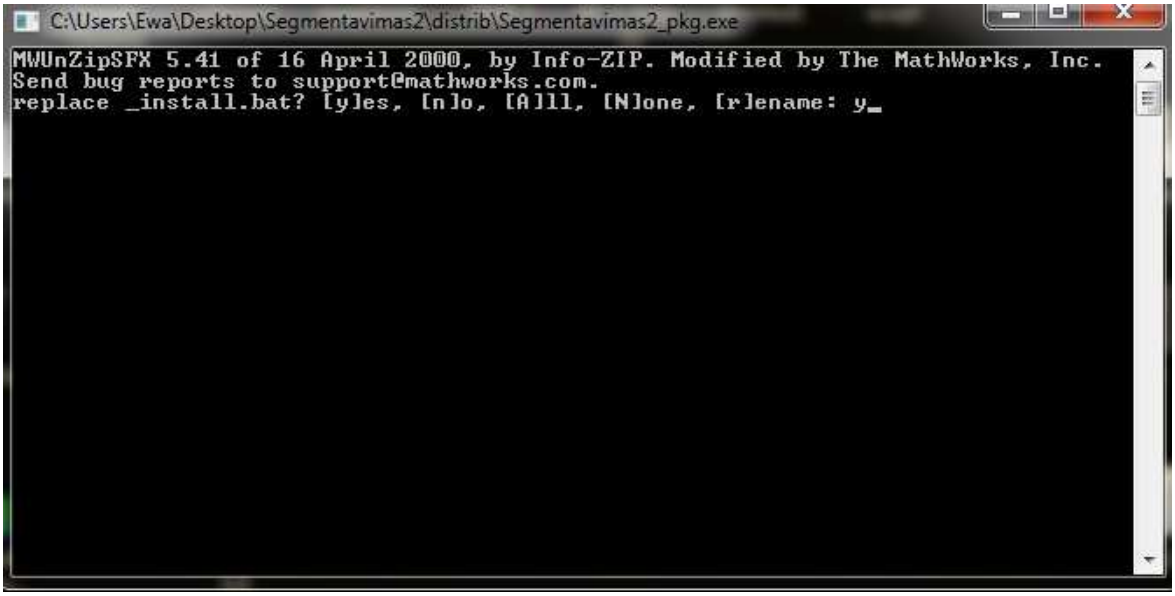
[12] Robert Crandall. *Image Segmentation Using the Chan-Vese Algorithm.* ECE 532 Project Fall, 2009, p. 1-23

Priedas

Algoritmas yra realizuotas Matlab programavimo kalba. Programą sudaro paleidimo klasė, pagrindinio ciklo klasė, taip pat kelios Matlab klasės papildomoms funkcijoms skaičiuoti – Heaviside žingsninei funkcijai, gradientui bei sustojimo sąlygai. Visos šios Matlab klasės yra sudėtos į vieną .exe failą patogesniam vartotojo naudojimui.

Kadangi programa yra realizuota Matlab programavimo kalba, norint paleisti .exe failą kitame kompiuteryje, pirmiausia turime užinstaliuoti Matlab‘o kompiliatoriaus aplinką MCR. Instaliavimas ir programos paleidimas atrodo atitinkamai:

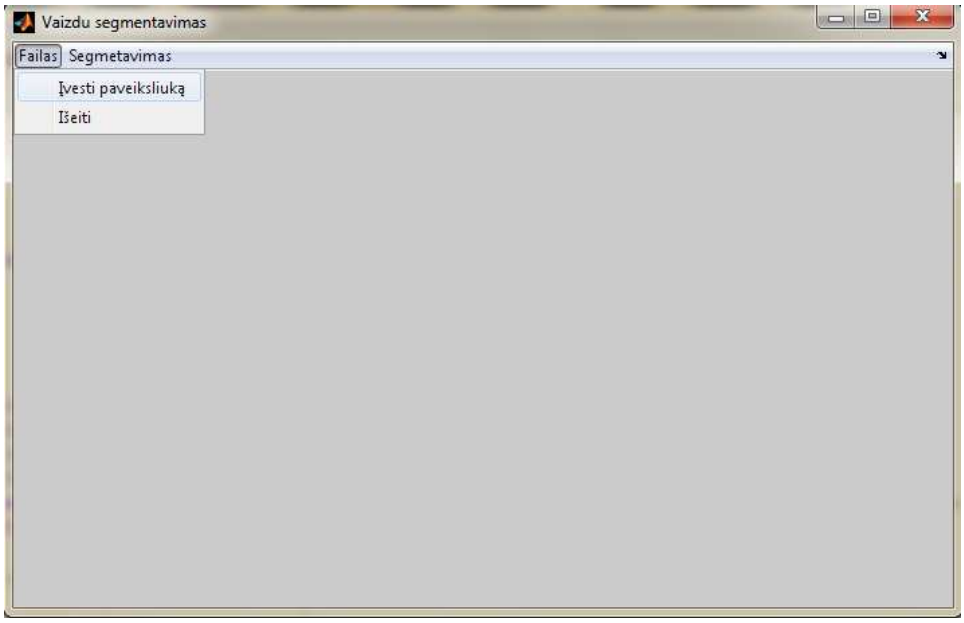
1) Spaudžiame ant „Segmentavimas_pkg“ failo, esančio kataloge Segmentavimas/distrib. Pasirodžiusiame lange spaudžiame „y“:



1 pav. Instaliavimo langas

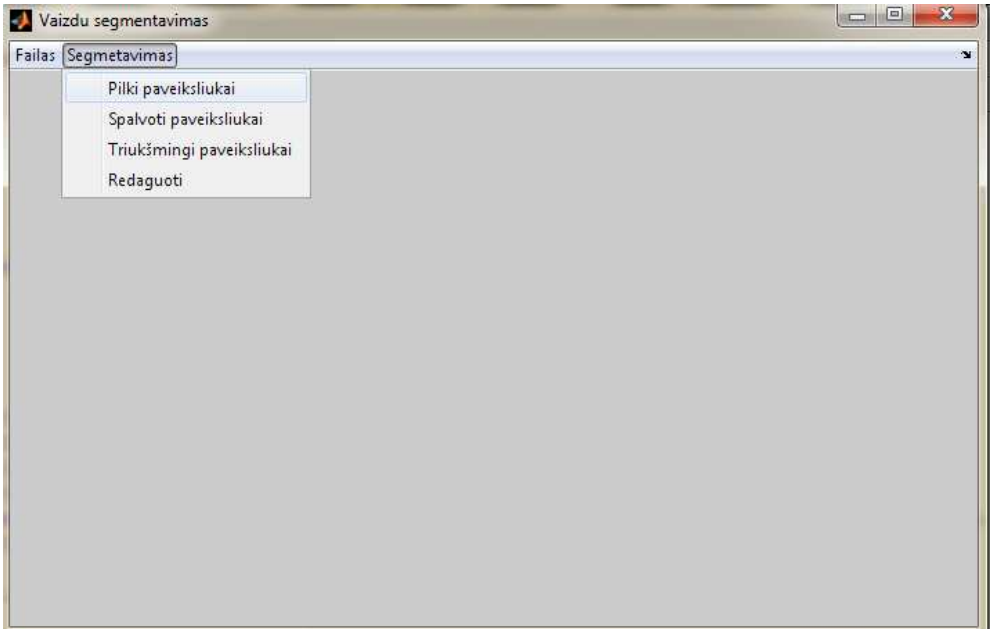
2) Atsakymą „y“ spaudžiame porai sekančių klausimų, po kurių prasideda Matlab kompiliatoriaus instaliacija.
3) Pasibaigus instaliacijai paspaudžiame ant toje pačioje direktorijoje esančio failo „Segmentavimas.exe“ ir paleidžiame programą.

Vaizdų segmentavimą galime kontroliuoti vartotojo lange, kuris pasirodo paleidus programą. Pirmiausia turime įvesti pradinį paveiksliuką. Tam tiksliui meniu „Failas“ išrenkame „Įvesti paveiksliuką“:



2 pav. Paveiksliuko pasirinkimas

4) Pasirodžiusiame lange išrenkame paveiksliuko direktoriją. Įvedus paveiksliuką nustatome segmentavimą pagal paveiksliuko tipą. Tam tikslui meniu „Segmentavimas“ išrenkame paveiksliuko tipą, pagal kurį bus vykdomas segmentavimas:



3 pav. Segmentavimo tipo nustatymas

Programai atlikus 20 iteracijų tame pačiame vartotojo lange pasirodo segmentavimo rezultatai.