VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Vaizdų klasterizavimas

Atliko: 2 kurso 9 grupės studentė
Dalia Martišiūtė _____

Darbo vadovas:
Doc. Algirdas Bastys _____

Darbo recenzentas:
Doc. Algimantas Juoazapavičius _____

Vilnius
2008

Turinys

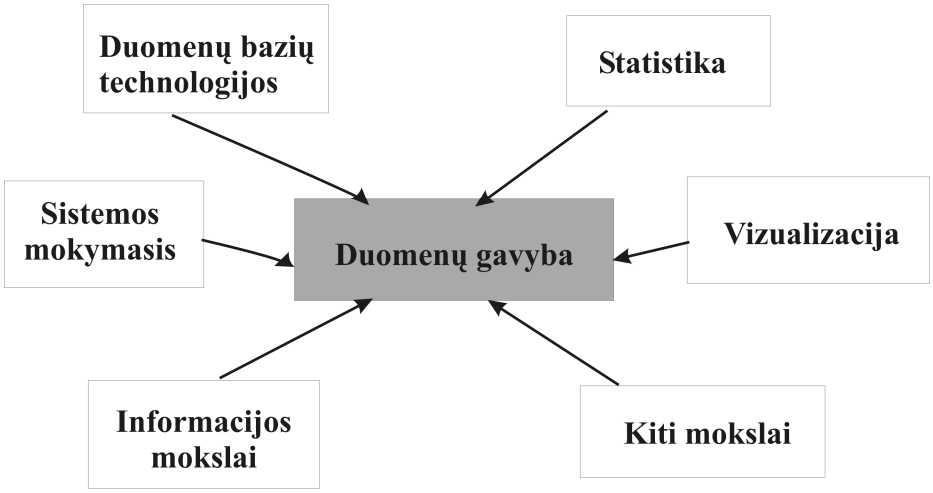
Sutartinių terminų sąrašas

Duomenų gavyba

Duomenų gavyba (angl. *data mining*) – duomenų analizės algoritmai, kurių tikslas – netrivialių, anksčiau nežinomų ir potencialiai vertingų žinių išgavimas iš didelių duomenų saugyklų. Dažnai duomenų gavyba tapatinama su žinių radimu duomenų bazėse (angl. *knowledge discovery from database*). Pastarasis terminas apibrėžia visą procesą nuo duomenų išrinkimo, apdorojimo, filtravimo, duomenų gavybos iki išgryninimo, vizualizacijos ir žinių gavimo. Šiuo atveju žinių gavimas suprantamas kaip žinių pateikimas žmogui suprantamomis priemonėmis.

Duomenų gavyba apjungia daugelį kitų disciplinų [VH01] (1 paveikslas), pavyzdžiui:

- Duomenų bazių technologijos
- Sistemos mokymasis (angl. machine learning)
- Informacijos mokslai (angl. information science)
- Statistika
- Vaizdavimas



1 paveikslas. Mokslo sritys, susijusios su duomenų gavyba.

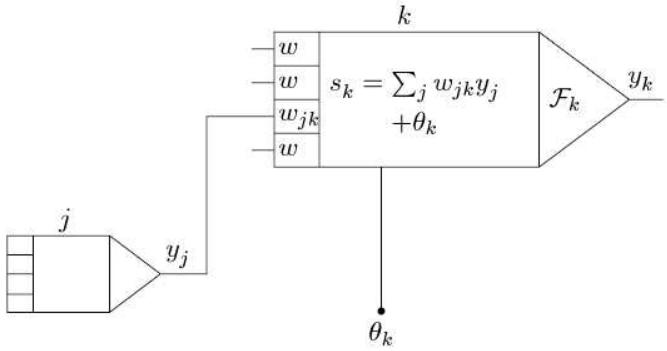
Klasterizavimas

Klasterizavimas (angl. *clustering*) - tai viena iš duomenų gavybos sričių. Klasterizavimo algoritmo užduotis – objektų suskirstymas į prasmingas grupes – klasterius, kai jokia papildoma informacija apie tas grupes (jų dydį, kiekį, grupavimo požymius) nėra iš anksto žinoma. Klasterizavimo algoritmas pats, pagal pasirinktus algoritmo parametrus, turi nurodyti, kokioms grupėms priklauso atitinkami įvesties duomenys.

Klasterizavimas dar vadinamas neprižiūrimu mokymusi, mokymusi be mokytojo, taip atskiriant jį nuo klasifikavimo – prižiūrimo mokymosi. Plačiau apie šiuos ir kitus duomenų analizės metodus rašoma [Ber02], [VHG03].

Dirbtinis neuronas

Dirbtinis neuronas – tai funkcija, turinti tiek teorinių, tiek praktinių panašumų su biologiniais neuronais. Dirbtinio neurono modelis pavaizduotas 2 paveiksle. Neuroną apibūdina jo aktyvacijos būsena y_k, turimi ryšiai w_{jk}, sklidimo taisyklė s_k bei aktyvacijos funkcija F_k, taip pat kiekvienam neuronui yra priskiriama išorinė įvestis (angl. *bias*, *offset*) θ_k. Svorio indeksas w_{jk} nurodo, kad ryšys jungia neuronus j ir k.



2 paveikslas. Dirbtinio neurono modelis.

Dirbtinis neuroninis tinklas

Dirbtinis neuroninis tinklas – tai duomenų apdorojimo sistema, sudaryta iš paprastų informacijos apdorojimo vienetų – neuronų; ši sistema remiasi paralelinio paskirstyto skaičiavimo idėjomis. Dirbtinis neuroninis tinklas dažnai lyginamas su žmogaus galvos smegenų žieve, nes jų abiejų pagrindas yra skaičiavimus atliekantys neuronai, turintys daug dendritų –įėjimo signalams ir vieną aksoną išvedimui. Kitas bendrumas – abu iš aplinkos gauna ir išsaugo gautą informaciją (mokosi) tarpneuroninių jungčių svoriuose.

Neuroninį tinklą apibūdina ([KS96]):

- Skaičiavimo ląstelių aibė (neuronai);
- Aktyvacijos būsena y_k kiekvienoje ląstelėje, t.y. tos ląstelės išvesties reikšmė.
- Ryšiai tarp ląstelių. Paprastai kiekvienas ryšys yra apibūdinamas svoriu w_{jk}, kuris nusako kokią įtaką ląstelės j signalas turės ląstelei k;
- Sklidimo taisyklė, kuri apibendrina į neuroną ateinančius įvesties duomenis iki faktinės įvesties s_k;
- Aktyvacijos funkcija, kuri pakeičia ląstelės būseną y_k priklausomai nuo faktinės įvesties s_k;
- Išorinė įvestis (angl. *bias*, *offset*) θ_k kiekvienai ląstelei;
- Informacijos rinkimo būdas (mokymosi taisyklė);
- Aplinka, kurioje veikia visa sistema; įvesties bei išvesties signalai, taip pat klaidų pranešimai.

Dažnai neuroninis tinklas kuriamas konkrečiai užduočiai spręsti, todėl mokymosi metu yra pateikiami pavyzdžiai, kaip sistema turėtų atsakyti į konkrečius įvesties vektorius. Tinklas adaptuoja savo svorius pagal pateiktus pavyzdžius (mokosi).

Po mokymosi fazės tinklas yra testuojamas, ar tikrai teisingai pasirinko savo svorius. Yra parenkamas tam tikras kiekis duomenų vektorių, nepateiktų tinklui mokymo fazės metu. Testavimui

yra patariama ([VHG03]) pasirinkti 2/3 visų duomenų vektorių, o mokymuisi skirti1/3, taip užtikrinamas testavimo rezultatų patikimumas.

Dažnai dirbtiniai neuroniniai tinklai naudojami klasifikavimo, funkcijų modeliavimo, atpažinimo uždaviniams spręsti.

Iteracija

Sąvoka vartojama kalbant apie dirbtinius neuroninius tinklus. Iteracija – toks laiko (ir skaičiavimų) tarpas, kurio metu vienas įvesties vektorius yra pateikiamas tinklui ir yra apdorojamas (kartu adaptuojami ir tinklo svoriai).

Epocha

Sąvoka vartojama kalbant apie dirbtinius neuroninius tinklus. Epocha – toks laiko (ir skaičiavimų) tarpas, kurio metu visi įvesties aibės vektoriai yra pateikiami tinklui ir yra apdorojami. Vieną epochą sudaro n iteracijų, čia n – duomenų aibės dydis.

Kohonen tinklas

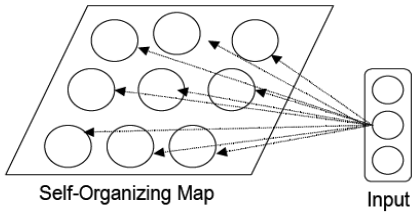
Žr. Savaime susitvarkantis tinklas

SOM tinklas

Žr. Savaime susitvarkantis tinklas

Savaime susitvarkantis tinklas

Savaime susitvarkantis tinklas, SOM tinklas – (angl. *Self-Organizing (Feature) Map* – savaime susitvarkantis (požymių) tinklas) neuroninio tinklo rūšis, pasiūlyta T. Kohonen [Koh82], [Koh95], [Koh98], [Koh99a], [Koh99b], kartais dar vadinamas Kohonen tinklu pagal kūrėjo pavardę. Šis tinklas dažnai yra naudojamas klasterizavimo uždaviniams spręsti, t.y. kai sistema, kurią modeliuoja tinklas nėra iš anksto žinoma, nėra apibrėžti tinklo atsakymai į konkrečias įvesties reikšmes ([PFL01]). SOM tinklą sudaro vienas įvesties vektoriaus sluoksnis ir vienas išvesties neuronų sluoksnis. Paprastai išvesties neuronai būna suskirstyti į dvimatę gardelę (rečiau vienmatę ar trimatę). Pavyzdinė SOM tinklo struktūra pavaizduota 3 paveiksle. Kiekvienas įvesties (angl. *input*) sluoksnio neuronas yra sujungtas su visais išvesties neuronais, tokiu būdu kiekvienas išvesties neuronas turi tiek svorio vektorių, kokia yra įvesties vektoriaus dimensija.



3 paveikslas. SOM tinklo struktūra.

Naujai atėjusiam įvesties vektoriui pirmiausia yra surandamas artimiausias neuronas (angl. BMU – *Best Matching Unit*); artimumas skaičiuojamas pagal pasirinktą atstumo funkciją, paprastai

naudojamas Euklidinis, Manheteno ar skirtumų kvadratų atstumas. Tuomet visi tinklo neuronai yra perskaičiuojami pagal mokymosi taisyklę:

$$W_{ki}(t+1) = W_{ki}(t) + \alpha(t) \times h(t) \times \left(X_i(t) - W_{ki}(t)\right),$$

čia W_{ik} – perskaičiuojamojo neurono svoris, t – pasirinkta iteracija, X(t) – įvesties vektorius iteracijoje t, o h(t) – kaimynystės (angl. *neigborhood*) funkcija:

$$h_{ci}(t) = \alpha(t) \times \exp\left(-\frac{\left\|r_i - r_c\right\|^2}{2\sigma(t)^2}\right),$$

kurioje α(t) – mažėjanti funkcija, nusakantį tinklo mokymosi greitį, σ(t) – mažėjanti funkcija, nusakanti kaimynystės plotį (kiek BMU neurono kaimynų bus paslenkami link įvesties vektoriaus), r_c – BMU neurono indeksas, r_i – perskaičiuojamojo neurono indeksas.

Tinklo mokymosi fazė baigiama kai tinklo svoriai beveik nesikeičia – jis pakankamai gerai prisitaikė prie įvesties duomenų erdvės –, arba po nustatyto epochų skaičiaus.

SOM tinklas pasižymi topologijos išsaugojimu (angl. *topology preserving*), t.y. tie vektoriai, kurių BMU neuronai yra gretimi, bus taip pat gretimi įvesties erdvėje, tačiau ne visada yra atvirkščiai.

BMU neuronas

Artimiausias neuronas, BMU neuronas (angl. *Best Matching Unit*) – toks dirbtinio neuroninio tinklo neuronas, kurio svoriai yra artimiausi įvesties vektoriui, pateiktam atitinkamoje mokymosi iteracijoje. Atstumas tarp įvesties vektoriaus ir tinklo neurono yra atstumas tarp įvesties vektoriaus ir jo BMU neurono, t.y. :

$$\left\|x(t) - W_s(t)\right\| = \min\left\|x(t) - W_k(t)\right\|,$$

čia x(t) yra įvesties vektorius iteracijoje t, W(t) – neurono svorių vektorius. Kiekvienoje iteracijoje t yra ieškomas toks k, kuris minimizuotų anksčiau pateiktą formulę.

U-Matrix

U-Matrix – (sutrumpinimas iš (angl.) *Unified distance Matrix* – suvienodinto atstumo matricos) SOM tinklo vizualizavimo metodas, pasiūlytas A. Ultsch [US90], [Ult93], [Ult03]. U-Matrix – tai matrica, kurios dydis sutampa su tinklo gardelės dydžiu, o laukeliuose įrašomi vidutiniai atstumai tarp atitinkamo tinklo neurono ir jo kaimynų (kvadratinės gardelės atveju - keturi).

Neuroną žymėsime $n_{i,j}$, kur i – tinklo gardelės eilutės indeksas, j – stulpelio indeksas.

$$\text{rytinisAtstumas}_{i,j} = \text{atstumas}(n_{i,j}, n_{i,j+1}),$$

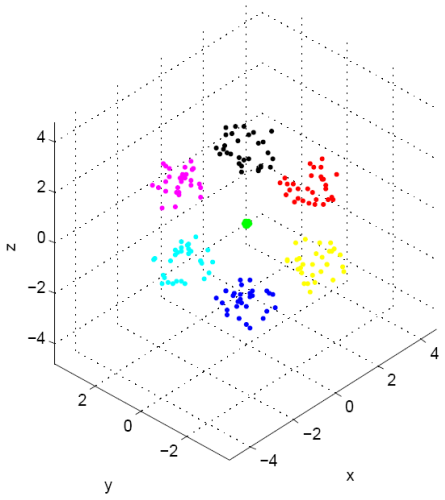
$$\text{pietinisAtstumas}_{i,j} = \text{atstumas}(n_{i,j}, n_{i+1,j}),$$

čia $\text{atstumas}(n_{i,j}, n_{k,l})$ – atstumas tarp neuronų svorių įvesties duomenų erdvėje; atstumas yra pasirenkamas pagal realiai tinklo naudojamą atstumo funkciją. Tada vidutinis atstumas tarp skaičiuojamojo neurono ir jo keturių kaimynų:

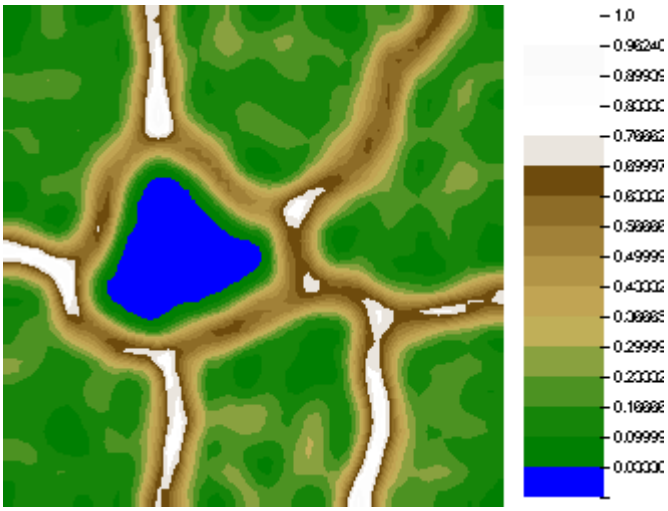
$$\text{vidutinisAtstumas}_{i,j} = (\text{rytinisAtstumas}_{i,j-1} + \text{rytinisAtstumas}_{i,j} + \text{pietinisAtstumas}_{i-1,j} + \text{pietinisAtstumas}_{i,j}) / 4 .$$

Taip visa matrica yra užpildoma vidutinėmis tinklo svorių skirtumų reikšmėmis. Iš tokios matricos nupiešę paviršių gautume reljefą, kurio įdubos, atskirtos kalnais, reprezentuoja klasterius, nes įdubose gretimų neuronų svoriai yra panašūs, o kalnai rodo, kad gretimų neuronų svoriai skiriasi.

4 paveiksle yra pavaizduota trimačių duomenų aibė – 212 taškų, sudarančių 7 gerai vienas nuo kito atskirtus klasterius. 50x50 dydžio tinklas buvo mokomas su tokiais duomenimis. Iš tinklo gauta U-Matrix pavaizduota 5 paveiksle, joje ryškiai matomi 7 klasteriai – mėlynas ežeras ir žali slėniai, atskirti rudais ir baltais kalnais.



4 paveikslas. Trimačių duomenų aibė.



5 paveikslas. U-Matrix, gauta iš ESOM tinklo, mokyto su ketvirto paveikslo duomenimis.

Taškinis požymis

Taškinių požymių radimas yra vienas iš objekto išskyrimo iš paveikslėlio metodų. Be taškinių požymių objekto išskyrimui taip pat gali būti naudojami trimačio vaizdo atstatymo bei dėmių išskyrimo algoritmai. Taškiniai požymiai remiasi linijomis, ribomis, kampais, ypatingomis struktūromis (pvz. ovalais, stačiakampiais); jų pagalba yra išskiriamas objektas. Taškiniams požymiams išskirti dažnai užtenka juodai balto paveikslėlio, o taškų radimui svarbūs yra pikselių intensyvumo pokyčiai.

Taškiniai požymiai dažnai naudojami objektų atpažinimo, tekstūros nustatymo, paveikslėlių išrinkimo, mobilių robotų lokalizacijos, duomenų gavybos iš video vaizdų, panoraminių vaizdų

suliejimo uždaviniuose. Taškiniai požymiai yra pakankamai charakteringi, saviti ir gerai veikia, net dalinai uždengus objektą.

SIFT

Vienas iš taškinių požymių radimo, aprašymo bei atpažinimo algoritmų, pateiktas [Low99] bei [Low04] straipsniuose. SIFT yra angliškų žodžių junginio „*Scale Invariant Feature Transform*“ sutrumpinimas. Plačiau šis algoritmas bus pristatytas 1.2.1. skyrelyje.

Detektorius

Detektoriaus (angl. *detector*) sąvoka vartojama kalbant apie ypatingų taškų paiešką paveikslėlyje; detektorius – algoritmas, kurio pagalba randami ypatingi taškai. Dažnai detektorius veikia ieškodamas ypatingų struktūrų paveikslėlyje, pavyzdžiui, linijų kampų, dėmių.

Deskriptorius

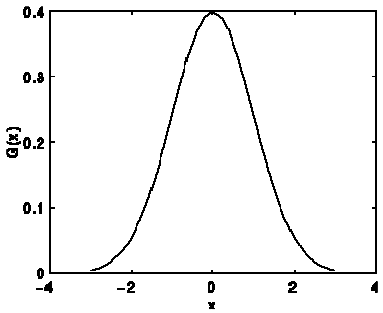
Deskriptorius (angl. *descriptor*) – algoritmas (ar algoritmo dalis), kurio pagalba yra aprašomi ypatingi taškai paveikslėlyje. Taip pat deskriptoriumi vadinami ir ypatingų taškų šablonai (angl. *template*), t.y. skaičių vektoriai, apibūdinantys ypatingą tašką paveikslėlyje. Dažniausiai deskriptorių skaičiavimui naudojama tam tikro dydžio aplinka šalia ypatingojo taško; ši aplinka normalizuojama ir aprašoma nustatyto ilgio vektoriumi.

Gauso filtras

Gauso filtras yra paveikslėlio filtravimo būdas, suteikiantis paveikslėliui išliejimo efektą (primenantį nesufokusuotos kameros vaizdą). Viematis gauso filtras aprašomas formule

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}},$$

čia σ yra gauso pasiskirstymo funkcijos standartinis nuokrypis; pasiskirstymo funkcijos vidurkis yra laikomas x=0. Gauso pasiskirstymas pavaizduotas 6 paveiksle.

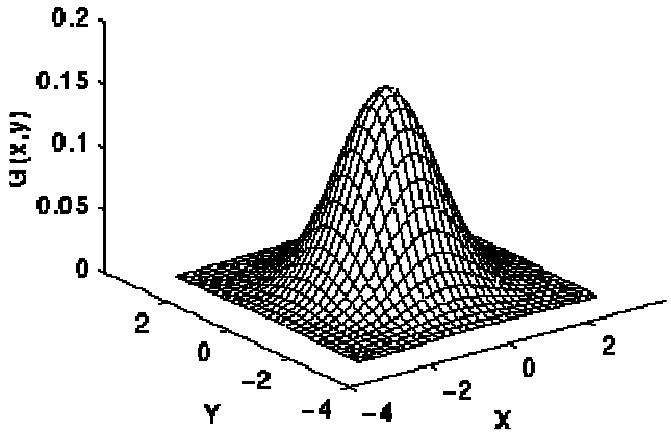


6 paveikslas. Vienmatis gauso pasiskistymas su vidurkiu 0 ir standartiniu nuokrypiu σ = 1.

Paveikslėlio filtravimui yra naudojamas dvimatis gauso filtras

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

Jo grafikas pavaizduotas 7 paveiksle.



7 paveikslas. Dvimatis gauso pasiskistymas su vidurkiu (0,0) ir standartiniu nuokrypiu σ = 1.

Gausinis pasiskirstymas yra visur teigiamas, taigi, begalinio dydžio. Kadangi paveikslėlis gali būti interpretuojamas kaip dvimatė pikselių matrica, tai atliekant filtravimą reikia diskretizuoti gauso filtrą. Jo dydis yra pasirenkamas toks, kad toliausiai nuo centro nutolusios reikšmės jau būtų pakankamai artimos nuliui.

ROC kreivės

ROC (angl. *Receiver Operating Characteristic*) kreivė buvo pasiūlyta [HM82]. Ji naudojama algoritmų palyginimui, nustatant, kuris iš algoritmų tiksliau atskiria „vienodus“ ir „skirtingus“ objektus. Kreivė vizualizuoja santykį tarp atmestų teisingų (FRR, angl. *False Rejection Rate*) bei paliktų klaidingų (FAR, angl. *False Acceptance Rate*) atitikimų tarp objektų.

1. Suskaičiuoti panašumus tarp visų objektų porų iš turimos duomenų aibės.
2. Surūšiuoti panašumus nuo didžiausio iki mažiausio. Braižysim grafiką nuo kairiojo krašto:
   a) jei poroje abu objektai yra vienodi, brėžti liniją vertikaliai viršun;
   b) jei poroje yra skirtingi objektai, brėžti liniją horizontaliai dešinėn.

Kuo statesnis gaunasi ROC grafikas, tuo geriau pasirinktas algoritmas identifikuoja objektus, atskiria vienodus objektus nuo skirtingų.

Kartu su ROC kreivėmis naudojamas EER (angl. *Equal Error Rate*) įvertis. Jis nurodo slenkstį, kurį pasirinkus FAR ir FRR klaidos bus lygios. Mažesnė EER reikšmė parodo geresnius atpažinimo algoritmo rezultatus.

Anotacija

Objektų klasterizavimas – tai viena iš duomenų gavybos (angl. data mining) sričių. Šių algoritmų pagrindinis privalumas – gebėjimas atpažinti grupavimo struktūrą be jokios išankstinės informacijos.

Magistriniame darbe yra pristatomas vaizdų klasterizavimo algoritmas, naudojantis savaime susitvarkančius neuroninius tinklus (angl. Self-Organizing Map). Darbe analizuojami vaizdų apdorojimo, ypatingųjų taškų radimo bei palyginimo metodai. Nustatyta, kad SIFT (angl. Scale Invariant Feature Transform) ypatingųjų taškų radimas bei aprašymas veikia patikimiausiai, todėl būtent SIFT taškiniai požymiai yra naudojami klasterizavime. Darbe taip pat analizuojamas atstumo tarp paveikslėlių radimo algoritmas, tiriami skirtingi jo parametrai. Algoritmų palyginimui yra naudojamos ROC (angl. Receiver Operating Characteristic) kreivės ir EER (angl. Equal Error Rate) rodiklis. Vaizdų klasterizavimui yra naudojamas ESOM (Emergent Self-Organizing Map) neuroninis tinklas, jis vizualizuojamas U-Matrix (angl. Unified distance Matrix) pagalba ir tinklo neuronai skirstomi į klasterius vandenskyros algoritmu su skirtingu aukščio parinkimu. Magistriniame darbe demonstruojami klasterizavimo rezultatai su pavyzdinėmis paveikslėlių duomenų bazėmis bei realiais gyvenimiškais vaizdais.

Summary

“Image Clustering”

Clustering algorithms – a field of data mining – aims at finding a grouping structure in the input data without any a-priori information.

The master thesis is dedicated for image processing and clustering algorithms. There are point-feature detection, description and comparison methods analyzed in this paper. The SIFT (Scale Invariant Feature Transform) by D. Lowe has been shown to behave better than the other ones; hence it has been used for image to image distance calculation and undirectly in clustering phase. Finding distances between images is not a trivial task and it also has been analysed in this thesis. Several methods have been compared using ROC (Receiver Operating Curve) and EER measurements. Image clustering process is described as: (1) training of ESOM (Emergent Self-Organizing Map), (2) its visualization in U-Matrix, (3) neuron clustering using waterflood algorithm, and (4) image grouping according to their best-matching unit neurons. The paper demonstrates the image clustering algorithm on public object image databases and real life images from the Internet as well.

Įvadas

Ankstesniame (bakalauro) darbe buvo pademonstruotas vaizdų klasterizavimo algoritmas. Šiame darbe yra siekiama jį pagreitinti pakeičiant taškelių intensyvumo reikšmes paveikslėliuose į taškinius požymius, nusakančius vaizduojamus objektus.

Klasterizavimas – tai duomenų gavybos algoritmų rūšis, kuri nustato duomenų grupavimo tvarką, kai ji nėra a priori žinoma. Šiame darbe yra analizuojami klasterizavimo algoritmai ir jie taikomi vaizdų suskirstymui į grupes. Tokio tipo algoritmai naudojami dirbtinio intelekto uždaviniuose – didelė paveikslėlių aibė skirstoma į kelias kategorijas. Taip pat algoritmas galėtų būti pritaikomas vaizdų rūšiavimo ir paieškos sistemose.

Vaizdų klasterizavimu siekiama nustatyti, kuriuose iš paveikslėlių vaizduojamas tas pats objektas. Tai yra gan sudėtinga užduotis, nes paveikslėliuose vaizduojami objektai gali būti skirtingai apšviesti, dalinai uždengti kokių nors kliūčių, nufotografuoti skirtingu kampu, jie gali būti skirtingų dydžių, spalvų, net iš dalies skirtingų formų.

Ankstesniame (bakalauro) darbe [Mar06a] buvo parodytas vaizdų klasterizavimo algoritmas, naudojantis juodai-baltų paveikslėlių intensyvumo reikšmes kaip klasterizavimo požymius. Šiame darbe analizuojami įvairūs taškinių požymių radimo algoritmai, lyginamos jų savybės. Taip pat yra tiriamas taškinių požymių algoritmų patikimumas atpažįstant įvairius objektus, algoritmų atsparumas triukšmams bei transformacijoms. Naudojantis ROC kreivėmis yra parenkamas taškinių požymių palyginimo algoritmas bei jo parametrai, patikimai veikiantys su pavyzdinėmis paveikslėlių duomenų bazėmis.

Galiausiai taškinių požymių radimo algoritmas yra pritaikomas klasterizavime neuroniniu tinklu. Paprastai klasterizavimo algoritmai naudoja tiesioginius objektų požymius (taško multidimensinėje erdvėje koordinates) ir juos lygina neišskirdami skirtingų atstumo metrikų skirtingiems objekto požymiams. Šiame darbe yra pademonstruotas klasterizavimo algoritmas, kuris naudoja pakeistą atstumo funkciją – įvesties paveikslėliai yra interpretuojami ne kaip pikselių masyvas, o kaip ypatingųjų taškų deskriptorių aibė. Reikia pažymėti, kad kiekviename paveikslėlyje gali ir dažniausiai yra skirtingas skaičius ypatingųjų taškų, todėl atstumų tarp paveikslėlių metrikos radimas nėra trivialus uždavinys.

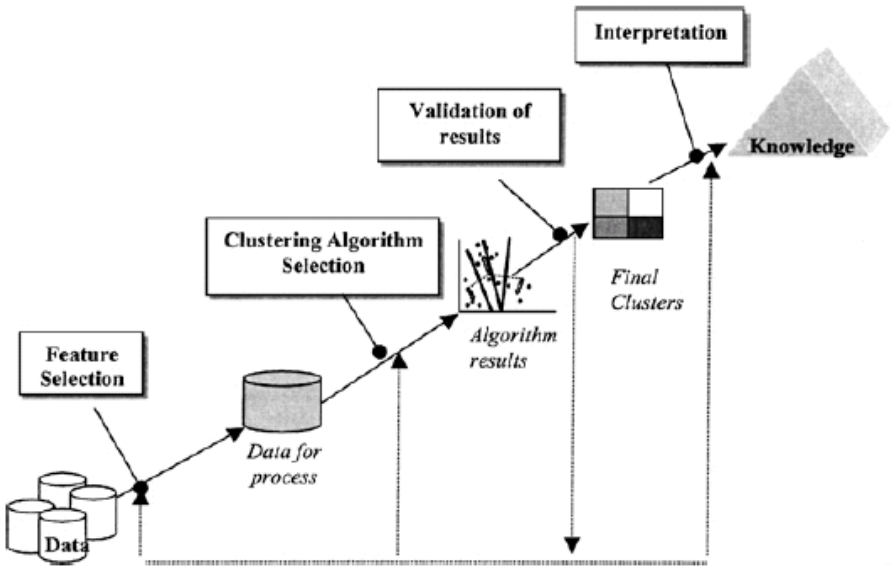
Klasterizavimui pasirinktas savaime susitvarkantis tinklas (SOM) bei jo modifikacijos aptartos ir išbandytos bakalauriniame darbe. Klasterizavimo rezultatai pademonstruoti su pavyzdinėmis duomenų aibėmis bei realiais gyvenimiškais paveikslėliais.

1. Literatūros apžvalga

1.1. Klasterizavimo algoritmai

Pagrindiniai klasterizavimo algoritmo etapai, kuriuos nurodo [FPS+96], parodyti 8 paveiksle.

- Požymių pasirinkimas. Šiame etape būtina pasirinkti esminius požymius iš duomenų aibės, taip kad juose būtų visa reikalinga konkrečiai užduočiai informacija ir kartu būtų kuo mažiau perteklinės informacijos.
- Klasterizavimo algoritmas. Šiame etape svarbu pasirinkti tokį algoritmą, kurio rezultatai būtų „geri“ pasirinktai duomenų aibei. Taip pat reikia tinkamai pasirinkti panašumo funkciją bei klasterizavimo kriterijus.
    - *Panašumo funkcija* nusako, koks atstumas yra tarp dviejų pasirinktų įvesties vektorių. Ši funkcija gali turėti svorius, t.y. vieni požymių skirtumai yra svarbesni už kitų, tačiau kartais būtent siekiama išvengti šio efekto.
    - *Klasterizavimo kriterijus* dažnai apibrėžiamas kainos funkcija (angl. *cost function*) arba tam tikrų taisyklių rinkiniu. Klasterizavimo kriterijus turėtų nurodyti, kokius klasterius tikimasi išskirti duomenų aibėje, taip pat – duomenų jungimo į vieną klasterį ir atskyrimo į kelis klasterius kriterijus.
- Rezultatų validavimas. Klasterizavimo schemos tinkamumas yra tikrinamas naudojantis tam tikrais kriterijais ir technikomis. Kadangi klasterių skaičius, jų savybės yra nežinomos a priori, dažniai jų validumui nustatyti yra reikalingas atskiras tyrimas.
- Rezultatų interpretavimas. Dažnai atitinkamos srities ekspertai turi klasterizavimo rezultatus sujungti su kitomis eksperimentinėmis ar teorinėmis žiniomis, kad priimtų teisingą sprendimą.



8 paveikslas. Pagrindiniai klasterizavimo proceso etapai. Iliustracija iš [FPS+96]

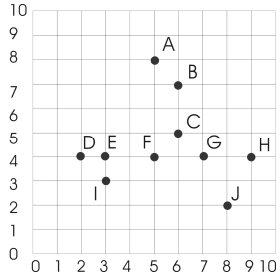
[VHG03] yra nurodyti pagrindiniai ir dažniausiai naudojami klasterizavimo algoritmai. Pagal naudojamą metodą jie grupuojami į [JMF99]:

- *Klasterizavimas skirstant* (angl. *partitional clustering*) – duomenų aibė yra padalinama į grupę atskirų klasterių. Dažnai yra stengiamasi surasti tokį klasterių skaičių, kuris optimizuotų klasterizavimo kriterijaus funkciją.
- *Hierarchinis klasterizavimas* (angl. *hierarchical clustering*) – algoritmas arba visą duomenų aibę panariui skaido į vis mažesnius klasterius (angl. *divisive clustering*), arba pradeda nuo kiekvieno elemento ir kiekviename etape sujungia panašiausius iš jų (angl. *agglomerative clustering*). Klasterizavimo rezultatas – klasterių medis, dendograma, rodanti, kaip klasteriai yra susiję. Pasirinktame lygyje nupjovus dendogramą gaunama pasirinkta klasterizavimo struktūra.
- *Klasterizavimas, paremtas tankiu* (angl. *density-based clustering*) – šio algoritmo panašumo funkcija naudoja tankio duomenų aibėje informaciją.
- *Klasterizavimas Kohonen neuroniniu tinklu* (angl. *Kohonen net clustering*) – šis metodas remiasi Kohonen neuroninio tinklo metodu.

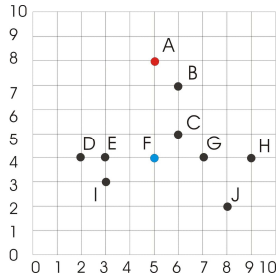
1.1.1. Klasterizavimas skirstant

Šio tipo algoritmai (statistiškai) suskirsto duomenų aibę į nurodytą klasterių skaičių (klasterių skaičius būna algoritmo parametras) optimizuodamas klasterizavimo kriterijų. Reprezentatyviausias šios grupės algoritmas – K-Means [Mac67], [Wei06].

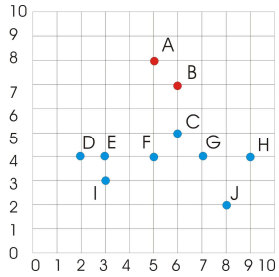
K-Means pagrindinis parametras yra klasterių skaičius k. Algoritmas prasideda pirmiausiai pasirenkant (atsitiktinai) klasterių centrus. 9 paveiksle pavaizduota duomenų aibė, o 10 paveiksle – pasirinkti du klasterių centrai (raudonas bei mėlynas taškai). Tada kiekvienam centrui yra priskiriami jam artimiausi taškai (kaip pavaizduota 11 paveiksle) ir perskaičiuojami nauji centrai (12 paveikslas). Toliau algoritmas kartojamas iteraciškai tol, kol klasterių centrai stabilizuojasi arba kol pasiekiamas maksimalus iteracijų skaičius.



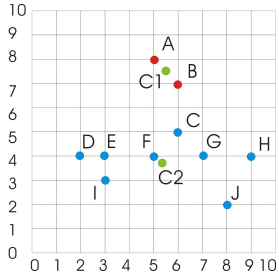
9 paveikslas. Dvimačių duomenų aibė.



10 paveikslas. K-Means algoritmo pirmasis etapas – centrų pasirinkimas. Raudona spalva žymi vieno klasterio centrą, mėlyna – kito.



11 paveikslas. Artimiausių taškų priskyrimas pasirinktiems klasterių centrams. Raudona spalva žymi vieną klasterį, mėlyna – kitą.



12 paveikslas. Naujų centrų paskaičiavimas pagal klasteriams priskirtus taškus. C1 žymi pirmojo (raudonojo) klasterio centrą, C2 – antrojo (mėlynojo).

K-Means klasterizavimo algoritmas siekia minimizuoti klaidos funkciją:

$$E_K = \sum_k \left\| x_k - m_{c(x_k)} \right\|^2,$$

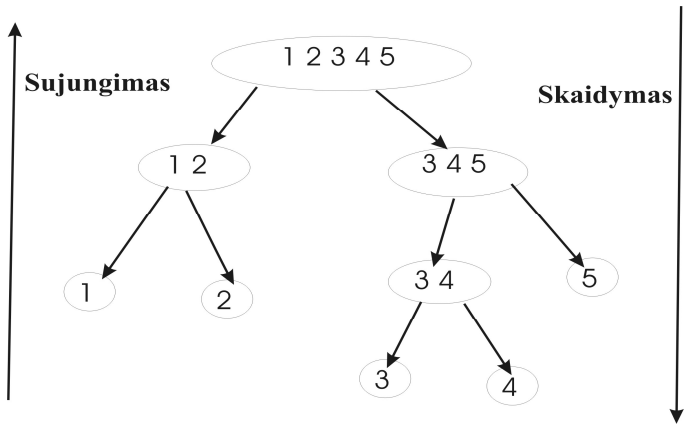
čia K – klasterių skaičius, x_k – įvesties duomuo, kuris priskirtas m_c klasteriui.

1.1.2. Hierarchinis klasterizavimas

Hierarchiniai klasterizavimo algoritmai kiekviename etape vis sujungia artimus klasterius, arba vis skaido mažai susijusius. Klasterizavimo algoritmo rezultatas – dendrograma. Konkreti klasterizavimo schema gaunama „nupjovus“ dendrogramą pasirinktame lygyje.

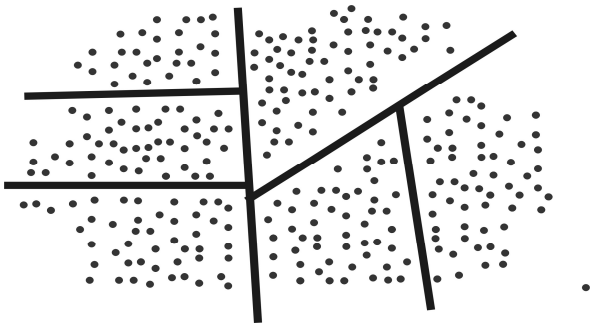
Pagal sujungimo ar skaidymo procesus algoritmai skirstomi į sujungiančius (angl. *agglomerative*) arba skaidančius (angl. *divisive*).

13 paveiksle pavaizduota pavyzdinė dendrograma. Skaidančio algoritmo atveju buvo pradedama nuo vieno klasterio, turinčio 1, 2, 3, 4 ir 5 taškus. Tada šis klasteris buvo suskaidytas į du klasterius, kurių viename buvo 1, 2 taškai, o kitame – 3, 4, 5 taškai. Trečiajame žingsnyje buvo išskirti 1 ir 2 taškai bei kitas klasteris išskaidytas į du – viename 3 ir 4, o kitame – 5 taškas. Paskutiniame žingsnyje buvo atskirti 3 ir 4 taškai. Sujungiantis algoritmas veiktų priešinga kryptimi, t.y. pradžioje sujungtų 3 ir 4 taškus į vieną bendrą klasterį, vėliau jungtų 1 ir 2 bei 3, 4 su 5 į bendrus klasterius, paskutiniame žingsnyje sujungtų visus likusius klasterius.

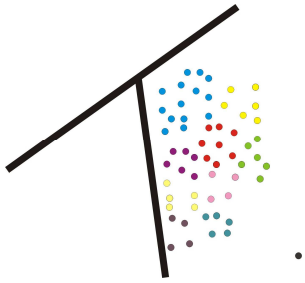


13 paveikslas. Hierarchinis klasterizavimas

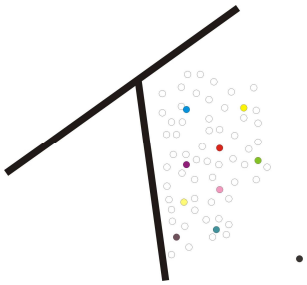
Hierarchinių algoritmų veikimas bus pademonstruotas pavyzdiniu CURE (angl. *Clustering Using REprezentatives*) algoritmu. 14 paveiksle pavaizduota dvimačių taškų aibė. Pirmajame klasterizavimo etape visa duomenų aibė yra suskirstoma į pasirinktą suskaidymų skaičių. 15 paveiksle pavaizduoti 6 suskaidymai (juodos linijos nurodo jų ribas). Toliau kiekvienas suskaidymas yra klasterizuojamas atskirai naudojant pasirinktą klasterizavimo algoritmą, pavyzdžiui K-Means. 13 paveiksle parodyta viena suskaidymo dalis suklasterizavus ją K-Means algoritmu su 10 klasterių centrų (klasterių centrai parodyti 16 paveiksle). Trečiajame etape kiekvieno klasterio centras yra paslenkamas link suskaidymo dalies centro, kaip parodyta 17 paveiksle. Taip kiekviena suskaidymo dalis yra reprezentuojama keliais klasterių centrais.



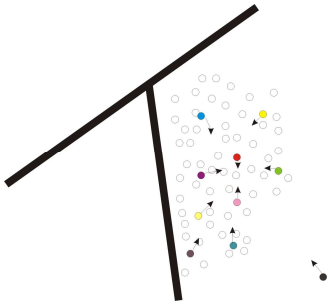
14 paveikslas. Duomenų aibė, padalinta į 6 grupes (suskaidymus).



15 paveikslas. 1 suskaidymo dalis suklasterizavus ją K-Means algoritmu, kai numatomas klasterių skaičius 10.



16 paveikslas. Vienos suskaidymo dalies „centrai“, gauti suklasterizavus ją K-Means algoritmu su parametru 10.



17 paveikslas. Klasterių centrai yra paslenkami link particijos vidurio.

1.1.3. Klasterizavimas, paremtas tankiu

Šios grupės algoritmai puikiai geba atpažinti įvairių formų klasterius kaip tankius objektų sambūrius, atskirtus retesniais regionais. Vienas šios grupės algoritmų DBSCAN ieško tokių klasterių, kurių kiekvieno taško aplinkoje (pagal pasirinktą spindulį) būtų bent jau minimalus taškų kiekis (jis yra nurodomas kaip algoritmo parametras). Taip yra atskiriami tankūs erdvės regionai nuo retesnių ir nustatoma klasterizavimo struktūra.

Tankiu parenti algoritmai yra atsparūs triukšmams.

1.1.4. Klasterizavimas Kohonen neuroniniu tinklu

Kohonen neuroninis tinklas – tai neuroninio tinklo rūšis, pasiūlyta T. Kohonen [Koh82]. Dažnai šis tinklas vadinamas savaime susitvarkančiu (požymių) tinklu (angl. *Self-Organizing (Feature) Map* – SOM).

Pasibaigus tinklo mokymosi fazei yra vykdomas klasterizavimas. Visi duomenų aibės vektoriai paduodami tinklui ir surandami BMU neuronai. Klasterių skaičių atitinka SOM tinklo neuronų skaičius; visi duomenų aibės vektoriai, kurių BMU neuronas yra tas pats yra priskiriami tam pačiam klasteriui.

Toks klasterizavimo būdas yra labai panašus į statistinį K-Means metodą, nes kiekviename etape yra ieškomi klasterių centrai (SOM tinkle juos atitinka neuronai) bei centrai paslenkami priklausomai nuo kitų klasteriui priklausančių taškų pozicijos. Vienintelis esminis skirtumas tarp šių klasterizavimo metodų yra SOM tinklo neuronų svorių perskaičiavime dalyvaujantys ne tik BMU neuronai (K-Means algoritme juos atitiktų klasterių centrai), bet ir jų kaimynai.

Deja, aprašytasis klasterizavimo būdas yra pakankamai neefektyvus, nes nėra išnaudojami visi SOM tinklo privalumai, t.y. topologijos išsaugojimo savybė.

Todėl buvo pasiūlytas [UV94], [Ult95] kitas klasterizavimo būdas, besiremiantis Kohonen neuroniniais tinklais. Šį kartą yra pilnai panaudojama topologijos išsaugojimo savybė ir vienam klasteriui priskiriami visų artimų BMU neuronų duomenų vektoriai. SOM tinklo dydis

pasirenkamas žymiai didesnis nei tikėtinas klasterių skaičius. Po mokymosi fazės tinklas yra vizualizuojamas U-Matrix metodu. Vėliau pagal pasirinktą aukštį yra nustatomi „slėniai“ – artimi neuronai – bei juos skiriantys „kalnai“ – vienas nuo kito nutolę neuronai – ir taip nustatomos klasterių ribos. Duomenys, kurių BMU neuronai pakliuvo į tą patį „ežerą“ yra priskiriami tam pačiam klasteriui.

Panašus klasterizavimo būdas yra pasiūlytas [VA00]. Jame pirmasis etapas taip pat yra SOM tinklo mokymas. Yra pasirenkama gan didelis išvesties neuronų skaičius, todėl SOM tinklas atlieka vektorių kvantizavimo funkciją – tinklo neuronai pasislenka į „tankias“ duomenų vietas, tuo tarpu retesnes aplenkdami. Antrajame etape yra vizualiai įvertinamas klasterių skaičius. Čia naudojami U-Matrix arba pataikymų skaičiaus metodai. Pataikymų skaičius kiekvienam neuronui nurodo, kiek kartų jis buvo pasirinktas kaip BMU neuronas duomenų vektoriui. Daugiadimensiniams duomenims vaizduoti naudojami specialūs metodai, pvz. Sammon žymėjimas (angl. *Sammon's mapping*) [Sam69], kreivalinijinių komponenčių analizė (angl. *curvilinear component analysis*) [DH97]. Trečiajame etape klasterizuojami neuronai naudojant klasterizavimo skirstant arba hierarchinio klasterizavimo algoritmus. Tokia klasterizavimo schema sumažina bendrą algoritmo sudėtingumą, taip pat rezultatai yra atsparesni triukšmams nei naudojant vien skaidymo ar hierarchinį klasterizavimo metodą.

SOM tinklo rūšis - ESOM (angl. *Emergent Self-Organizing Map*) buvo pasiūlytas [UM05]. Pagrindiniai jo skirtumai nuo įprastinio SOM tinklo yra žymiai didesnė gardelė bei U-Matrix vizualizacijos metodas, parodantis SOM tinklo artimus neuronus. Įprastiniuose klasterizavimo uždaviniuose naudojamas SOM tinklas turi būti sukuriamas tokio dydžio, kiek klasterių reikia surasti duomenų aibėje, t.y. vienas SOM tinklo neuronas atitinka vieno klasterio centrą. Tokiu būdu nėra išnaudojama SOM tinklo topologijos išsaugojimo savybė. ESOM autoriai pasiūlė naudoti daug didesnį neuroninį tinklą negu yra numanomų klasterių įvesties erdvėje – keliasdešimties neuronų aukščio ir pločio. Taip pat, siekiant išvengti klaidų gardelės pakraščiuose, buvo pasiūlyta naudoti gardelę be kraštų, t.y. turinčią sferinę arba toroidinę struktūrą. Po tinklo mokymosi yra sudaroma U-Matrix ir jos slėniai vaizduoja klasterius, t.y. vienas klasteris yra apibrėžiamas kaip neuronų (bei atitinkamų įvesties vektorių) grupė).

[Mar06a] darbe buvo aptarti įvairūs SOM tinklo variantai ir ESOM tinklas pasirodė kaip paprasčiausias, o kartu ir ganėtinai galingas įrankis klasterizavimo uždaviniams spręsti. Pagrindinis jo trūkumas – klasterizavimo etape nėra galimybės automatiškai nustatyti galutines klasterių ribas iš U-Matrix vaizduojamo paviršiaus.

1.2. Taškiniai požymiai

Paveikslėliuose esantiems taškiniams požymiams rasti bei aprašyti yra sugalvota daugybė algoritmų. Seniausieji jų, pvz. linijų radimas Canny metodu [Can86], kampų lokalizavimas Harris detektoriumi [HS88], jau vadinami klasikiniais. Pastaruosius du dešimtmečius ieškoma universalių taškinių požymių išskyrimo algoritmų. Pirmiausia, jie turėtų būti tinkami įvairių tipų objektams, jų dalims nustatyti – tas pats algoritmas turėtų patikimai veikti bei išskirti pakankamai požymių tiek iš natūralių gamtos, patalpų, įrengimų vaizdų, tiek iš dirbtinai sugeneruotų paveikslėlių. Antra, pageidaujama, kad algoritmas būtų atsparus afininėms transformacijos, t.y. posūkiui, postūmiui, tempimui bei iškraipymui. Tai reiškia, kad algoritmas turi rasti bei nustatyti dviejų taškų (iš skirtingų paveikslėlių) panašumą, net ir po afininių transformacijų. Trečia, siekiama sukurti algoritmus, atsparius apšvietimo pokyčiams, t.y. tiek dieną, šviečiant saulei, tiek apsiniaukusiu oru, tiek esant dirbtiniam apšvietimui arba tik prieblandai, algoritmas turi sutapatinti du taškus iš skirtingų paveikslėlių. Ketvirta, pageidaujama, kad algoritmas būtų atsparus net ir trimačiam posūkiui arba žiūrėjimo kampo pasikeitimui, t.y. taškai iš paveikslėlių, vaizduojančių tą patį objektą skirtingu kampu, turėtų būti sutapatinami. Šiuolaikiniai taškinių požymių radimo algoritmai geba atpažinti objektus, pasuktus iki 60° trimačiu kampu.

Plačiausiai naudojamas SIFT algoritmas bus apibūdintas 1.2.1. skyrelyje. Tada, 1.2.2. skyrelyje bus pateikta viena šio algoritmo modifikacija.

Daug invariantiškų regionų algoritmų buvo palyginta [MTS+05] straipsnyje. Tai Harris-Affine detektorius ([MS02] bei [MS03]), Hessian-Affine ([MS02] bei [MS04]), MSER (angl. *Maximally Stable Extremal Regions*) detektorius [MCU+02], EBR (angl. *Edge-based Region*) detektorius [TV99] bei [TV04], IBR (angl. *Intensity Extrema-based Region*) [TV04], entropija besiremiantis detektorius (dar vadinamas *salient region*) [KZB04]. Iš jų tik Harris-Affine bei Hessian-Affine detektoriai yra taškiniai, remiasi ypatingu tašku. Šie du algoritmai bus aptarti 1.2.6. ir 1.2.7. skyreliuose.

[MS03] straipsnyje minimi deskriptoriai – „shape context“ [BMP02], „steerable filter“ [FA91], PCA-SIFT [KS04], „differencial invariants“ [KD87], „spin images“[LSP03], „complex filters“ [SZ02], GLOH (sutrumpinimas iš angl. *Gradient Location-Orientation Histogram*) [MS03]. Jie bus aprarti 1.2.3. – PCA SIFT, 1.2.4. – „shape context“, 1.2.5. – GLOH, 1.2.8. – „Spin images“, 1.2.9. „Complex filters“.

1.2.1. SIFT

SIFT yra angliškų žodžių *Scale Invariant Feature Transform* trumpinys. Šį algoritmą paskelbė bei patentavo David G. Lowe 1999 metais [Low99], [Low04]. Tai yra vienas iš plačiausiai naudojamų taškinių požymių radimo, aprašymo bei atpažinimo algoritmų.

Autorius skelbia, kad algoritmas geba rasti taškus, atsparius didinimui/mažinimui, postūmiui, posūkiui, apšvietimo pasikeitimui ir net iki 60% trimačiam posūkiui.

SIFT algoritmo pagrindiniai etapai yra:

1) taškinių požymių išskyrimas iš paveikslėlio (radimas);
2) tų taškų aprašymas (juos aprašančio vektoriaus sudarymas);
3) taškų aprašų, paimtų iš skirtingų paveikslėlių, palyginimas (atitinkamų porų radimas).

Toliau bus išvardinti ir apibūdinti SIFT algoritmo etapai (iš [Mar06b]).

Pirmiausia iš pradinio juodai balto paveikslėlio yra sudaroma piramidė. Joje būna pasirinktas skaičius *oktavų* (angl. *octave*), kuriose yra pasirinktas skaičius *lygių* (angl. *level*).

Vieną *oktavą* sudaro vienodo dydžio paveikslėliai kiekvieną jų filtruojant Gauso filtru su vis didėjančia Gauso filtro parametro σ reikšme. Gauso filtras (angl. *Gaussian filter*) paveikslėlį daro mažiau kampuotą, kiek išlieja vaizdą; šio filtro parametras sigma nurodo, kiek smarkiai duotasis paveikslėlis turėtų būti „išliejamas“. Taip pat Gauso filtras turi ypatingą savybę – su šiuo filtru filtruoti paveikslėliai atrodo lyg būtų nufotografuoti iš toliau. Taigi, vienoje *oktavoje* esantys *lygių* paveikslėliai atrodo lyg būtų nufotografuoti vis tolstant nuo objekto.

Tada paskutinio reikšmingo *oktavos lygio* paveikslėlis yra sumažinamas du kartus ir jis bus pirmasis sekančios *oktavos* paveikslėlis.

Gauso filtro parametras σ parenkamas taip, kad kiekviename *lygyje* jis proporcingai didėtų, be to paskutinio reikšmingo lygio filtravimui būtų nurodomas σ artimas 2. Čia remiamasi Gauso filtro „tolinimo“ savybe – paveikslėlis, filtruotas su parametru σ = 2, gali būti mažinamas dvigubai neprarandant jokios informacijos.

Piramidės sudarymo pseudo kodas:

```
function buildPyramid(Image image)
  Image sample = image;
  Octaves octave;
  for o = 1 : octavesCnt
```

```
    for l = 1 : levelCnt
   if (l == 1)
     octave[o]->level[1] = sample;
   else
     octave[o]->level[l] = gaussBlur(octave[o]->level[l-1], sigma0 *
scaleFactor);
   end;
    end;
    sample = halfSize(octave[o]->level[levelCnt-3]);
  end;
end
```

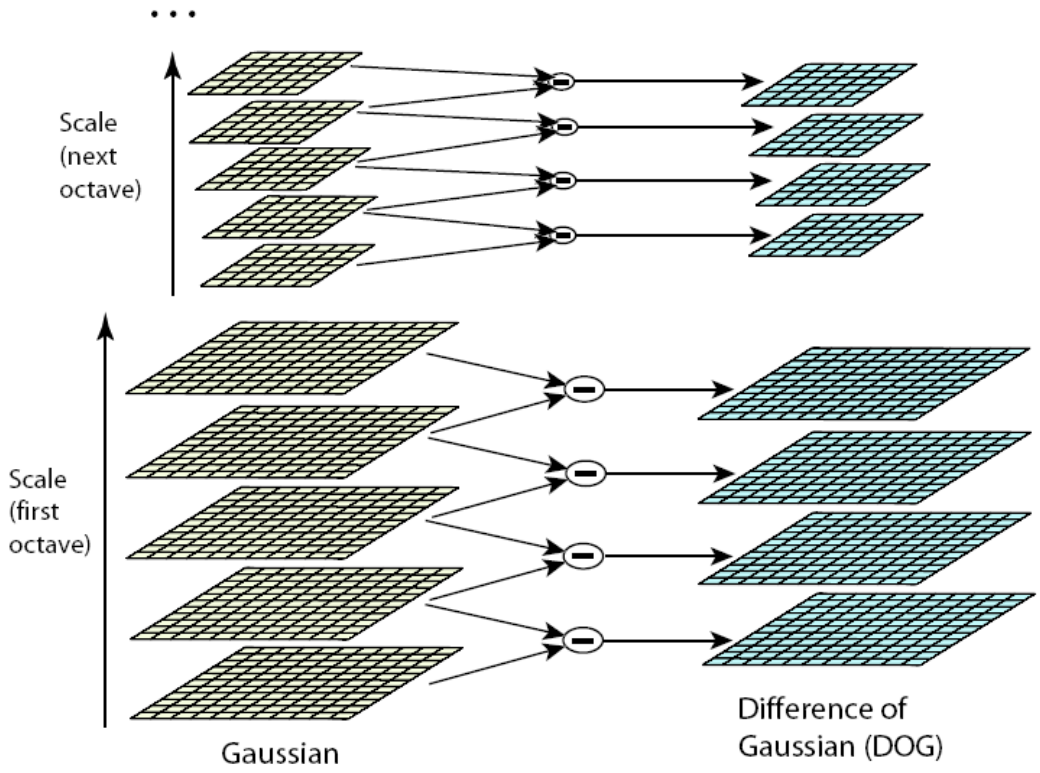
Kiekviename piramidės *lygyje* esantys filtruoti paveikslėliai yra atimami vieni iš kitų taip gaunant *DoG* (angl. *Difference of Gaussian* – Gausų skirtumas) paveikslėlius. Šie gan tiksliai aproksimuoja Laplaso filtrą – antros eilės Gauso filtro išvestinę. *DoG* paveikslėlių yra gaunama vienu mažiau negu yra *lygių* paveikslėlių.

DoG paveikslėlių gavimo pseudo kodas:

```
function calcDoG(Octaves octave)
   for o = 1 : octavesCnt
      for l = 1 : levelsCnt-1
         octave[o]->dog[l] = octave[o]->level[l+1] – octave[o]->level[l];
     end;
  end;
end
```

Kai kurie autoriai pataria *DoG* paveikslėlius normalizuoti pagal juose naudojamų *lygių* paveikslėlių filtravimo parametrų sigma reikšmės.

[Low04] straipsnyje pateikta iliustracija (18 paveikslas) vaizduoja piramidės sudarymą. Čia pateikiama 2 lygių piramidė (joje yra 5 Gauso filtru filtruoti paveikslėliai, iš jų gaunami 4 *DoG* paveikslėliai, tačiau ypatingų taškų ieškoma tik dviejuose viduriniuose *DoG* paveikslėliuose – likusieji naudojami tik taškų tikrinimo metu).

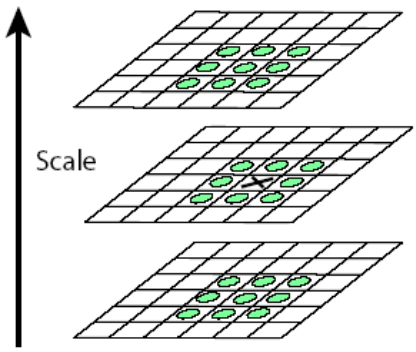


18 paveikslas. SIFT piramidės sudarymas (iš [Low04])

Ypatingų invariantiškų taškų ieškoma *DoG* paveikslėliuose.

Pradžioje kiekvienas *DoG* paveikslėlio taškelis yra tikrinamas, ar jis yra lokalus ekstremumas bei didesnis už pasirinktą „reikšmingumo“ parametrą. Kiekvienas taškelis yra lyginamas su savo artimiausiais 8 kaimyniniais taškeliais, taip pat su 9 atitinkamais taškeliais iš prieš šį paveikslėlį esančio *DoG* paveikslėlio bei su 9 taškeliais iš po šio esančio *DoG* paveikslėlio. Tinkamas kandidatas į ypatingus taškus turi būti didesnis už „reikšmingumo“ parametrą bei didesnis už 26 (9 + 9 + 8) kaimyninius taškelius, arba mažesnis už juos visus. Tada yra atmetami taškai, esantys ant linijų paveikslėlyje, o vėliau dar yra patikslinama ypatingų taškų pozicija – jiems surandamos tikslios taškelio koordinatės (racionalių skaičių išraiška). Kiekvienam rastam taškui yra randama jo lokali charakteringa orientacija, taško aplinkoje sudarant orientacijų histogramą bei išrenkant joje esančius pikus.

[Low04] straipsnyje pateikta taškų ekstremumų tikrinimo iliustracija parodyta 19 paveiksle.

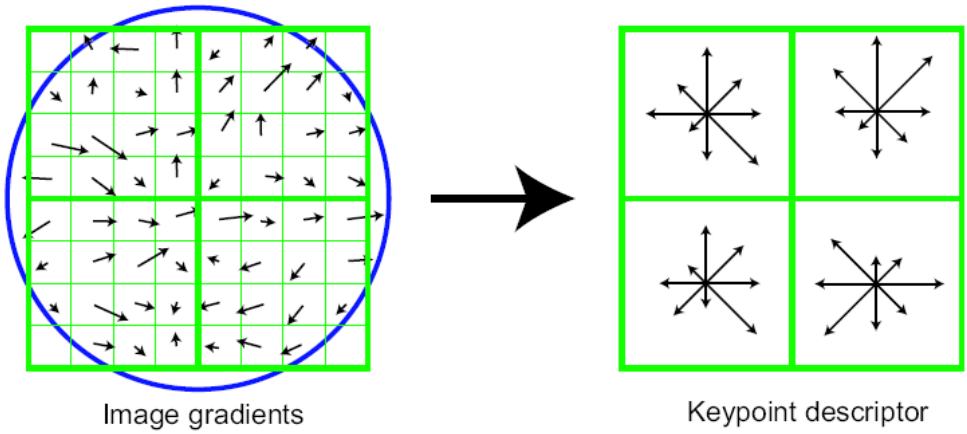


19 paveikslas. SIFT taškų ekstremumo tikrinimas. X pažymėtas taškas lyginamas su 26 kaimyniniais taškais. (iš [Low04])

Po šios – ypatingų taškų radimo fazės – yra išsaugomos rastų taškų koordinatės, orientacijos, o taip pat ir paveikslėlio piramidė – ji bus reikalinga skaičiuojant taškų aprašus.

Taškų aprašas yra skaičiuojamas iš to *lygio* piramidės paveikslėlio, kuriame *DoG* paveikslėlyje buvo rastas tas taškas. Imama pasirinkto dydžio kvadratinė aplinka, ji „pasukama“ pagal rastą to taško orientaciją. Tada aplinka padalijama į 2x2 arba 4x4 kvadratėlius ir kiekviename jų yra skaičiuojama 8 krypčių orientacijų histograma. Artimesni SIFT taškui gradientai yra skaičiuojami su didesniu svoriu, negu esantys kvadrato kraštuose. 20 paveiksle pateikta 2x2 kvadratėlių SIFT taško aprašo skaičiavimo iliustracija. Apskritimas didesniajame kvadrate rodo, kad artimesni SIFT taškui gradientai yra imami su didesniu svoriu. Iliustracija iš [Low04] straipsnio. SIFT taškų aprašai skaičiuojami iš *lygių* paveikslėlių.

SIFT taško aprašas yra apdorota visų kvadratėlių orientacijų histograma, t.y. jei buvo pasirinkti 4x4 kvadratėliai bei skaičiuojama 8 krypčių histograma, tai bus gautas 128 reikšmių SIFT taško aprašas.



20 paveikslas. SIFT taškų aprašo skaičiavimas (iš [Low04])

D.Lowe siūlo taškų lyginimui įsivesti reikšmingumo slenkstį, t.y. rodiklį, nurodantį, kad atstumas iki artimiausio taško turi būti nurodytą kiekį kartų mažesnis už atstumą iki antro artimiausio taško. Atstumui tarp taškų nustatyti yra naudojamas euklidinis atstumas. Kai kurie autoriai siūlo naudoti Mahalanobis atstumo funkciją arba skaliarinę sandaugą.

```
function findPairs(Descriptor des1, Descriptor des2)
  Pairs p;
  for i =  1 : des1->size()
     bestInd = 0;
     secondBestInd = 0;
     bestDist = Infinity;
     secondBestDist = Infinity;
     for j = 1 : des2->size()
        dis = 0;
        for k = 1 : des2[j]->vector->size()
          dis += (des1[i]->vector[k] – des2[j]->vector[k])^2;
       end;
        if (dis < bestDist)
          secondBestInd = bestInd;
          secondBestDist = bestDist;
          bestInd = j;
          bestDist = dis;
        end;
        if (dis < secondBestDist)
          secondBestInd = j;
          secondBestDist = dis;
```

```
        end;
      end;
      if (bestDist * threshold < secondBestDist)
         // reikia patikrinti, ar des1[i] taškas išrenkamas kaip
         //  artimiausias des2[bestInd] taškui
         bestJ = 0;
         secondBestJ = 0;
         bestDist = Infinity;
         secondBestDist = Infinity;
         for ii = 1 : des1->size()
            dis = 0;
            for k = 1 : des2[j]->vector->size()
               dis += (des1[i]->vector[k] – des2[j]->vector[k])^2;
            end;
            if (dis < bestDist)
               secondBestJ = bestJ;
               secondBestDist = bestDist;
               bestJ = ii
               bestDist = dis;
            end;
            if (dis < secondBestDist)
               secondBestJ = ii;
               secondBestDist = dis;
            end;
         end;
         if (bestDist * threshold < secondBestDist && bestJ == i)
            pairs->add(bestJ, bestInd, bestDist);
         end;
      end;
   end;
end
```

1.2.2. Greitas apytikslis SIFT

Šis SIFT algoritmo patobulinimas buvo paskelbtas [GGB06] straipsnyje. Greito apytikslio SIFT (angl. *Fast Approximated SIFT*, arba trumpiau, *ApproxSIFT*) autoriai pasiūlė dvi pagrindines naujoves – pakeisti piramidės sudarymą bei taškų aprašo skaičiavimui naudoti *integral histogram* vietoj įprastinės histogramos.

Pirmiausia, autoriai siūlo atsisakyti pradinio paveikslėlio didinimo, kaip tai daroma įprastinio SIFT algoritmo atveju. Straipsnio autoriai teigia, kad paprastai taškai, rasti šioje “padidintoje” *oktavoje*, nėra svarbūs taškų palyginimo fazėje, t.y. jie beveik niekada nebūna pasirenkami kaip sutampantys su ieškomaisiais taškais.

Antra, autoriai siūlo vietoj Gauso filtro naudoti stačiakampį filtrą (angl. *mean filter*) bei pakeisti paveikslėlio saugojimo struktūrą į *integral image*, taip iš dalies prarandant tikslumą galima labai pagreitinti algoritmą. Taškų suradimui yra naudojama *DoM* (angl. *Difference of Mean* – vidurkių skirtumas) erdvė vietoj originalios *DoG* erdvės.
Autorių pateikta SIFT algoritmų (piramidės sudarymo) palyginimo lentelė (išversta į lietuvių kalbą) yra 1 lentelė.

Įprastinis SIFT algoritmas	**ApproxSIFT algoritmas**
Dvigubai didinamas paveikslėlis	-
-	Skaičiuojamas “integral image”
DoG didinimo/mažinimo erdvė	*DoM* didinimo/mažinimo erdvė
Papildomas apdorojimas	

1 lentelė. Įprasto SIFT algoritmo bei ApproxSIFT piramidės sudarymo skirtumai

Duomenų struktūros *integral image* aprašymas yra pateiktas [VJ01] straipsnyje. Šiai struktūrai išskiriamas paveikslėlio dydžio buferis ir kiekvienoje (i,j) vietoje įrašoma visų stačiakampyje nuo (0,0) iki (i,j) esančių taškelių suma:

$$I(i, j) = \sum_{i`\le i, j`\le j} N(i`, j`),$$

čia I(i,j) žymimas *integral image* (i,j) elementas, o N(i,j) – originalaus paveikslėlio (i,j) taškelis. Dažnai *integral image* yra skaičiuojamas tokiu būdu:

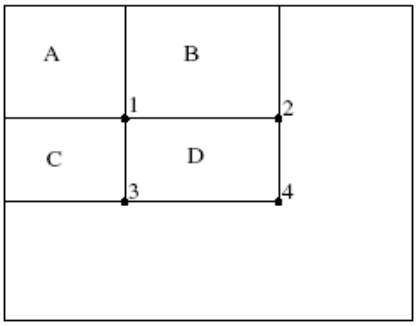
$$I(i, j) = I(i, j-1) + I(i-1, j) + N(i, j) - I(i-1, j-1),$$

imant I(-1, j) = I(i, -1) = I(-1, -1) = 0

Paprastai *integral image* skaičiavimui naudojama rekursinė procedūra ir visas *integral image* randamas tik vieną kartą perbėgant originalųjį paveikslėlį, kiekvienam elementui skiriant 2 sudėties operacijas. Rekursinės procedūros pseudo kodas:

```
function ii = integralImage(Image im)
   prevV = 0;
   // ii(1,0) := 0;
   for j = 1 : size(im,2)
       for i = 1 : size(im,1)
           ii(i,j) = prevV + im(i,j) – ii(i, j-1);
           prevV = ii(i,j);
       end;
   end;
end
```

ApproxSIFT piramidės sudarymui kiekvienoje oktavoje yra suskaičiuojamas *integral image*, tada jis filtruojamas su stačiakampiu filtru tiek kartų, kiek lygių turi būti kiekvienoje oktavoje. Filtruojant *integral image* stačiakampiu filtru pirmiausia yra surandamas reikiamo dydžio stačiakampis (tam prireikia 3 sudėties operacijų). Ši procedūra parodyta 21 paveiksle (iliustracija iš [VJ01]). Originalaus paveikslėlio plotas D randamas iš *integral image* reikšmės 1 pozicijoje (paveikslėlio plotas A) atimant *integral image* reikšmę 2 pozicijoje (paveikslėlio plotas A+B), dar atimant 3 pozicijos reikšmę (paveikslėlio plotas A+C) bei pridedant 4 pozicijos reikšmę (paveikslėlio plotas A+B+C+D). Filtruojant su stačiakampiu 9 dydžio filtru, pasirenkamas 9 taškelių plotas padalijamas iš 9 ir gauta reikšmė įrašoma į vidurinįjį taškelį.



21 paveikslas. *integral image* rekursinio skaičiavimo iliustracija (iš [VJ01]).

Vėliau filtruoti *lygių* paveikslėliai yra atimami vieni iš kitų – gaunami *DoM* paveikslėliai, kurie dar normalizuojami „jautrumo parametru“, čia s_1 bei s_2 yra mažesniojo ir didesniojo stačiakampio filtro dydžiai:

$$jautrumas * \left(1 - \frac{{s_1}^2}{{s_2}^2}\right).$$

Šis parametras turi įtakos randamų ypatingų taškų skaičiui.

Tolesnis ypatingų SIFT taškų radimo procesas yra labai panašus į originaliojo SIFT algoritmo, tik autoriai atsisako tikslaus taško radimo – taip sutaupo kiek laiko, kartu prarasdami požymio tikslumą, bet paprastuose dviejų paveikslėlių sutampančių taškų radimo užduotyse šis tikslumas nėra svarbus. Tiksli taško pozicija svarbi tik ieškant geometrinių požymių.

Ypatingų taškų aprašymo metu autoriai siūlo vietoj įprastinių 4x4 krypčių histogramų kiekvienam ypatingam taškui skaičiuoti *integral histogram* visam paveikslėliui, o tik paskui kiekvienam taškui išrinkti reikalingas šių histogramų dalis.

integral histogram yra *integral image* struktūros apibendrinimas. Jo aprašymas bei taikymo pavyzdžiai pateikti [Por05] straipsnyje.

Įprastinis SIFT algoritmas	**ApproxSIFT algoritmas**
Kiekvienam ypatingam taškui Charakteringo dydžio aplinkoje skaičiuojamos 4x4 orientacijos histogramos (8 krypčių).	Visame paveikslėlyje paskaičiuojamos 8 krypčių integralinės histogramos. Kiekvienam ypatingam taškui išrenkamos jų dalys.

2 lentelė. Įprasto SIFT algoritmo bei ApproxSIFT taškų aprašų skaičiavimų skirtumai

Taškui **x** = [x_1, x_2] (paveikslėlio atveju erdvė yra dviejų matavimų) p pozicijoje *integral histogram* apibrėžiama:

$$H(x^p, b) = \bigcup_{j=0}^{p} Q(f(x^j)).$$

Funkcija Q(*) nusako, į kurį histogramos stulpelį turi būti įrašytas taškas f(x), o sąjungos operatorius reiškia visų aplankytų taškų atitinkamų histogramos stulpelių reikšmių sumą.

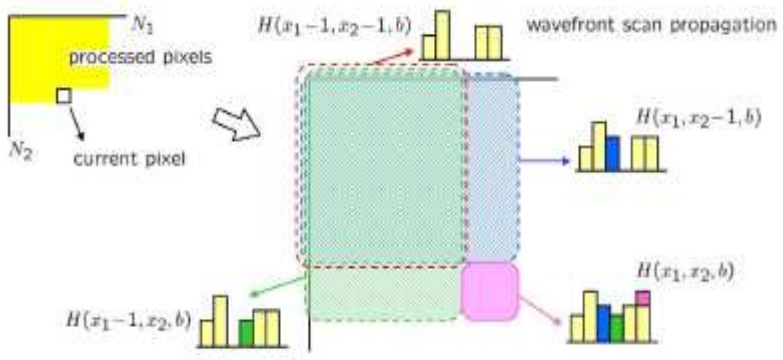
Taigi, H(x^p, b) reiškia viso regiono $0 \le x_1^j \le x_1^p, 0 \le x_2^j \le x_2^p$ histogramą.

Rekursinis *integral histogram* skaičiavimas (imant H(0,b) = 0, nes pradžioje visi histogramos stulpeliai yra nuliniai):

$$H(x^j, b) = H(x^{j-1}, b) \cup Q(f(x^j)).$$

22 paveiksle pavaizduotas *integral histogram* skaičiavimas 2D masyve (paveikslėlyje) naudojant „bangos“ perėjimo algoritmą. Histogramos reikšmė (x1,x2) pozicijoje:

$$H(x_1, x_2, b) = H(x_1, x_2 - 1, b) + H(x_1 - 1, x_2, b) - H(x_1 - 1, x_2 - 1, b) + Q(f(x_1, x_2)).$$



22 paveikslas. *integral histogram* skaičiavimas (iš [Por05])

1.2.3. PCA-SIFT deskriptorius

PCA-SIFT deskriptorius, paskelbtas [KS04] straipsnyje, remiasi SIFT algoritmu, tik vietoj tolydesnių histogramų su svoriais (angl. *smoothed weighted histogram*) naudojama PCA (sutrumpinimas iš angl. *Principal Components Analysis*) analizė normalizuotų gradientų gavimui.

PCA-SIFT algoritmas naudoja 41x41 erdvę aplink ypatingąjį tašką. SIFT deskriptoriaus dimensijos sumažinimui yra naudojama PCA analizė.

PCA-SIFT autoriai pateikė šio algoritmo realizacijos išeities kodus [PCA07] svetainėje.

1.2.4. „Shape context“ deskriptorius

Šis deskriptorius buvo pasiūlytas [BMP02] straipsnyje. Jo pagrindiniai etapai:

- rasti atitinkamus taškus tarp dviejų formų
- pagal atitinkamus taškus nustatyti formų transformaciją
- atstumas tarp dviejų formų nusakomas pagal bendrą atitinkamų taškų sutapatinimo klaidų skaičių

„Shape context“ deskriptorius skaičiuojamas labai panašiai kaip ir SIFT deskriptorius, tik jo pagrindas yra linijos, o ne taškai, kaip SIFT atveju. Linijos paveikslėlyje randamos Canny algoritmu. Aplink tašką, esantį ant rastos linijos, parenkamos 3 skritulio formos aplinkos 6, 11 bei 15 taškelių spinduliu, kaip parodyta 23 paveiksle. Kiekvienoje šių aplinkų sudaroma krypčių histograma bei padalinama į 4 stulpelius. Taigi, iš viso gaunamas 9*4 ilgio deskriptorius.

23 paveikslas. Artimiausios aplinkos apie ypatingą tašką padalijimas į sritis. Iliustracija iš [MS03].

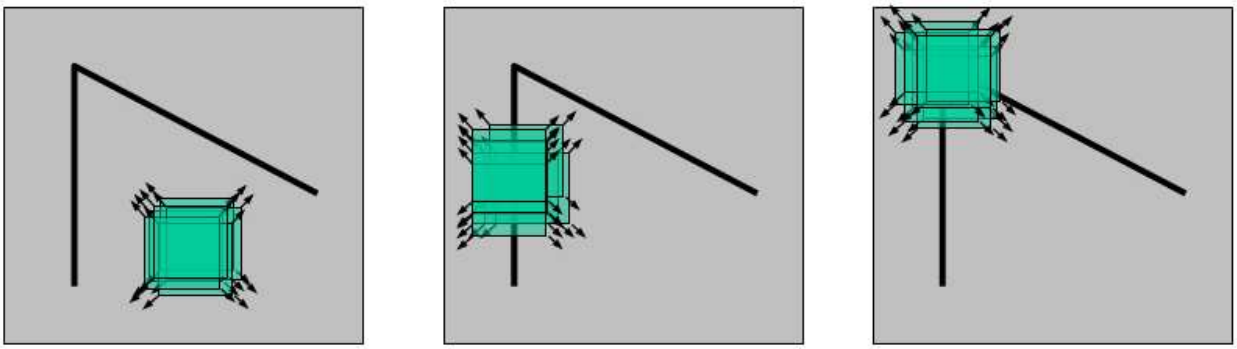
1.2.5. GLOH deskriptorius

Šis deskriptorius yra SIFT deskriptoriaus modifikacija, pristatyta [MS03] straipsnyje. Ypatingojo taško aplinka 6, 11 bei 15 taškelių spinduliu padalinama į 8 kryptis, taip gaunant 17 aplinkos dalelių (vidurinė 6 taškelių spindulio aplinka nėra dalijama į kryptis). Kiekvienoje iš šių erdvės vietų suskaičiuojama orientacijų histograma, padalinta į 16 stulpelių, panašiai kaip SIFT

deskriptoriaus atveju. Iš viso gaunamas 272 ilgio deskriptoriaus vektorius. Jis, taikant PCA analizę, sumažinamas iki 128 reikšmių (imant didžiausias tikrines reikšmes iš PCA analizės).

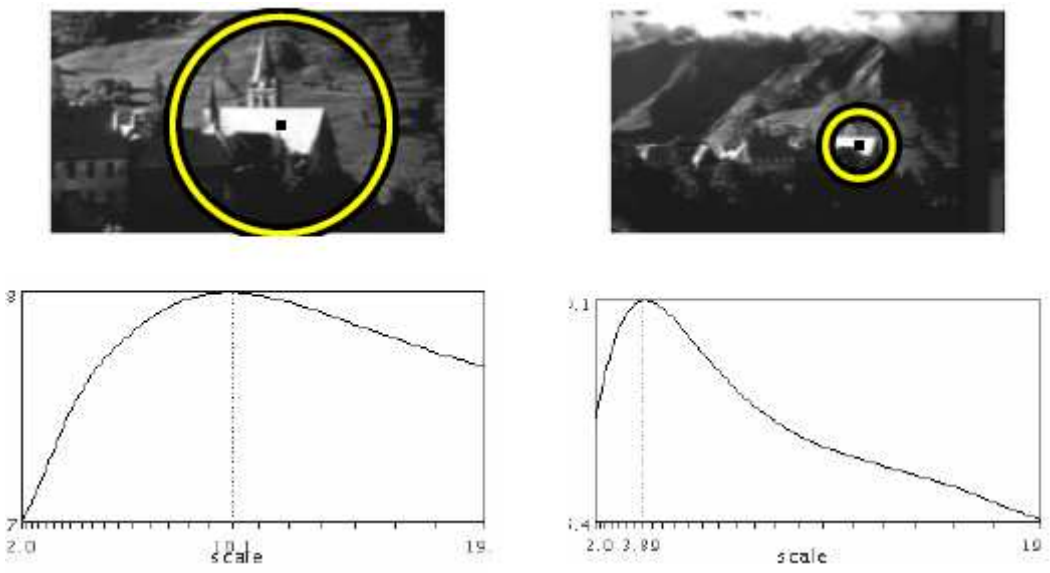
1.2.6. Harris-Affine detektorius

Šis detektorius buvo aprašytas [MS02] bei [MS04] straipsniuose. Algoritmas remiasi Harris kampų radimo paveikslėliuose funkcija ir prideda charakteringo dydžio radimo funkciją. Harris kampų radimo algoritmas jau tapo klasikiniu. Pagrindinė jo idėja – esant mažiems lango pozicijos pasikeitimams, ryškus paveikslėlio taškelių intensyvumo pasikeitimas nurodytų toje vietoje esant kampą. Šios idėjos iliustracija pateikta 24 paveiksle. Pirmasis paveikslėlis simbolizuoja lygią erdvę, t.y. stumdant langą visomis kryptimis taškelių intensyvumai praktiškai nesikeičia. Antrasis paveikslėlis rodo liniją, t.y. stumdant langą išilgai linijos taškelių intensyvumas nesikeičia. Trečiasis paveikslėlis vaizduoja kampą, nes lango judinimas bet kuria kryptimi sąlygoja ryškius taškelių intensyvumo pokyčius.



24 paveikslas. Kampo radimo algoritmo idėja. Pasirinkto dydžio langas slenkamas įvairiomis kryptimis. Pirmajame iš kairės paveikslėlyje nėra jokių esminių taškelių intensyvumo pasikeitimų, todėl nustatoma, kad toje vietoje yra plokščia erdvė. Viduriniajame paveikslėlyje intensyvumas nekinta slenkant langą aukštyn ir žemyn – toje vietoje yra linija. Trečiajame paveikslėlyje taškelių intensyvumas kinta slenkant langą visomis kryptimis, taigi, langas yra ant kampo. Iliustracija iš [FS04].

Harris kampų radimo algoritmas geba atpažinti pastumtus, pasuktus kampus, jis iš dalies yra atsparus apšvietimo (t.y. taškelių intensyvumo) pokyčiams, tačiau labai jautrus objekto didinimui ir mažinimui. Siekdami pašalinti šį trūkumą [MS02] autoriai naudoja [Lin98] straipsnyje T.Lindeberg pasiūlytą idėją: ieškomas požymis (sudarytas iš antro lygio išvestinių) įgis maksimumą tokiame paveikslėlio dydyje, kuris ir bus charakteringas bei atspindės tikrąją objekto struktūrą. Paveikslėlis yra filtruojamas Laplaso (angl. *Laplacian*) filtru ir ieškoma, koks filtro dydis duos maksimalų atsaką ieškomame taške. Tokiu būdu išrenkamas charakteringas to požymio dydis. 25 paveiksle yra parodytas tas pats objektas, matomas skirtingu atstumu. Žemiau pavaizduota kiekvieno iš tų taškų „dydžio“ funkcija, kurios maksimumas padeda nustatyti, kokio dydžio aplinką apilink rastąjį tašką reikėtų naudoti siekiant sutapatinti šiuos du paveikslėlius.



25 paveikslas. Pirmoji eilutė vaizduoja pastatą, nufotografuotą skirtingu atstumu. Antroje eilutėje parodyta Laplaso atsakai pažymėtame taške imant skirtingo dydžio paveikslėlius. Pirmosios funkcijos maksimumas 10.1, antrosios - 3.9; santykis tarp šių dydžių nurodo, kiek kartų pirmasis paveikslėlis turi būti sumažintas siekiant gauti antrąjį paveikslėlį. Geltoni apskritimai vaizduoja ypatingų taškų aplinką, kurios spindulys tris kartus didesnis už nustatytą charakteringą dydį. Iliustracija iš [MTS+05].

Kai jau rastas ypatingasis taškas ir jo charakteringas dydis, ieškoma charakteringa elipsės formos aplinka aplink šį tašką, t.y. tiriama, kokiu kampu yra pasisukęs ypatingas taškas. Tada ši aplinka normalizuojama į skritulio formos aplinką.

Harris-Affine detektorius nustato ypatingo taško koordinates, jo charakteringą dydį ir posūkį.

1.2.7. Hessian-Affine detektorius

Šis detektorius, aprašytas[MS02] bei [MS04] straipsniuose, iš esmės labai panašus į Harris-Affine detektorių, tik pradiniam požymių išrinkimui naudojama Hessian matrica:

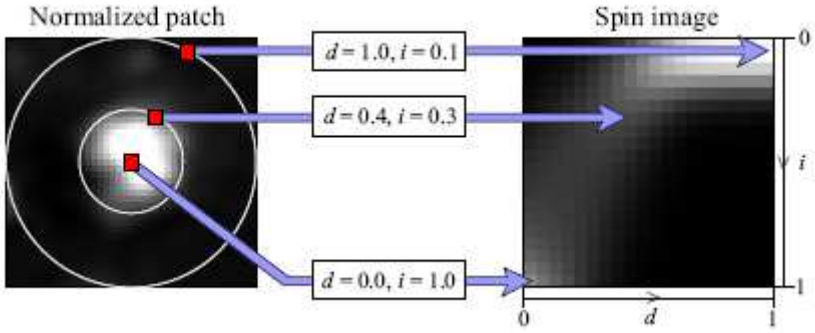
$$H = \begin{bmatrix} I_{xx}(x,\sigma_D) & I_{xy}(x,\sigma_D) \\ I_{xy}(x,\sigma_D) & I_{yy}(x,\sigma_D) \end{bmatrix},$$

čia I nurodo lokalias antros eilės paveikslėlio gradientų išvestines, skaičiuotas su Gauso filtro sigma reikšme σ_D.

Šios matricos determinantas bus lokaliai maksimalus toje vietoje, kur egzistuoja dėmė paveikslėlyje. Charakteringo požymio dydžiui bei posūkiui nustatyti naudojama tokia pati technika, kaip ir Harris-Affine detektoriuje.

1.2.8. „Spin images“ deskriptorius

Šis deskriptorius buvo pristatytas [LSP03] straipsnyje. „Spin image“ yra afininė normalizuoto regiono aplink ypatingąjį tašką dvimatė histograma iš taškelių intensyvumo reikšmių. Histogramos dimensijos yra d – atstumas nuo ypatingojo taško iki regiono krašto – ir i – intensyvumo reikšmės. Ši histograma pavaizduota 26 paveikslėlyje (paimtame iš [LSP03] straipsnio). Paveikslėlis kairėje vaizduoja normalizuotą regioną aplink ypatingąjį tašką, o paveikslėlis dešinėje – „spin image“ dvimatę histogramą.



26 paveikslas. „spin images“ deskriptoriaus sudarymo schema. Iliustracija iš [LSP03].

„Spin image“ deskriptorių palyginimui reikia skaičiuoti koreliaciją; siekdami to išvengti autoriai pasiūlė normalizuoti gautąją dvimatę histogramą ir tuomet naudoti euklidinę arba kvadratų sumų atstumo metriką.

1.2.9. „Complex filters“ deskriptorius

Šis deskriptorius buvo pristatytas [SZ02] straipsnyje. Jis skaičiuojamas skritulio formos normalizuotoje aplinkoje, pasirenkant charakteringą taško posūkį. Ši aplinka parodyta 27 paveiksle. Deskriptoriams palyginti naudojamas euklidinis arba kvadratų sumų atstumas.

27 paveikslas. „Complex filters“ deskriptorius. Viršutiniame paveikslėlyje apibrauktas ypatingas taškas, o apatiniame paveikslėlyje pavaizduota normalizuota jo aplinka. Iliustracija iš [SZ02].

2. Duomenų bazės

Algoritmo testavimui buvo ieškota duomenų bazių. Vienos jų yra tinkamos taškinių požymių testavimui, kitos – paveikslėlių klasterizavimui.

2.1. Affine Covariant Features

Šią duomenų bazę galima parsisiųsti iš Oksfordo universiteto tinklalapio [Aff07]. 28 paveiksle parodyti pavyzdiniai šios duomenų bazės paveikslėliai.

28 paveikslas. Affine Covariant Features duomenų bazės pavyzdiniai paveikslėliai.

Ši duomenų bazė buvo naudojama [MS04] bei [MTS+05] straipsniuose tikrinant įvairių taškinių požymių atpažinimo algoritmus.

Joje yra 6 grupės paveikslėlių su skirtingomis afininėmis transformacijomis: žiūrėjimo kampo pasikeitimas (trimatis posūkis), artinimas/tolinimas (trimatis poslinkis), paveikslėlio išsiliejimas, apšvietimo pasikeitimas (iš dalies pakito ir kontrastas) bei JPEG kompresija. Kai kurios scenos turi aiškiai atpažįstamus vienalyčius regionus su puikiai matomais kraštais (pavyzdžiui, grafiti piešinys, pastatai), tuo tarpu kitose scenose yra pasikartojančių ornamentų.

Žiūrėjimo kampo pasikeitimo sekoje kameros pozicija keičiasi beveik 60 laipsnių. Didinimo/mažinimo bei ryškumo pokyčiai buvo modeliuoti keičiant kameros parametrus: artinimą/tolinimą bei fokusavimą. Didinimo/mažinimo pokyčiai yra iki 4 kartų dydžio. Apšvietimo pokyčiai buvo modeliuoti keičiant kameros diafragmos dydį. Paveikslėliai, vaizduojantys JPEG kompresijos pasekmes buvo gauti spaudžiant paveikslėlius su skirtingais kokybės parametrais – nuo 40% iki 2%. Kiekvienoje paveikslėlių sekoje yra 800x640 dydžio šeši paveikslėliai. Visos paveikslėlių transformacijos yra žinomos, todėl galima tiksliai apskaičiuoti kiekvieno taško koordinates po transformacijos (sekančiame sekos paveikslėlyje).

Affine Covariant Features duomenų bazė yra tinkama patikrinti taškinių požymių algoritmams, nes taškų transformacijos paveikslėliuose yra žinomos ir lengva nustatyti ar algoritmas atpažįsta tuos pačius taškus ar ne. Duomenų bazė yra gan maža (8 sekos po 6 paveikslėlius), be to joje nėra objektų, tinkamų klasterizavimui.

2.2. Amsterdam Library of Object Images (ALOI)

Šią duomenų bazę galima rasti Amsterdamo universiteto tinklalapyje [Ams07]. 29 paveiksle parodyti pavyzdiniai duomenų bazės objektai.

29 paveikslas. ALOI duomenų bazės pavyzdiniai paveikslėliai.

Amsterdam Library of Object Images (angl. Amsterdamo objektų paveikslėlių duomenų bazė, sutrumpintai bus vadinama *ALOI*) sudaro 1000 mažų objektų, nufotografuotų moksliniais tikslais. Nuosekliai buvo keičiamas objektų apšvietimas (jo kryptis, spalva) bei kameros kryptis. Iš viso kiekvienam objektui užfiksuoti buvo padaryta virš 1000 kadrų, ir visoje duomenų bazėje yra 110250.

Duomenų bazėje yra daug objektų, jie yra surūšiuoti, todėl bus tinkami klasterizavimo algoritmo testavimui. Visi objektai yra nufilmuoti juodame fone, todėl fono pokyčiai neturės įtakos klasterizavimo algoritmo veikimui.

2.3. CSCLAB Image Database

Šią duomenų bazę galima rasti bei parsisiųsti iš [CSC07] tinklalapio. Pavyzdiniai duomenų bazės paveikslėliai yra parodyti 30 paveiksle.

30 paveikslas. CSCLAB duomenų bazės pavyzdiniai paveikslėliai.

Šioje duomenų bazėje yra 50 objektų. Buvo naudojamos 10 skirtingų aplinkų, kuriose buvo padedami keli objektai (nuo 3 iki 5 objektų), uždengiantys vienas kitą. Kartu su paveikslėliais duomenų bazėje yra pateikiami ir jų aprašai, tiksliai nurodantys objekto koordinates paveikslėlyje.

Ši duomenų bazė vaizduoja realius gyvenimiškus vaizdus, kuomet daiktai gali būti skirtingose aplinkose, esant skirtingoms apšvietimo sąlygoms bei dalinai uždengti. Patikimas objektų atpažinimo algoritmas turėtų gebėti surasti šiuos objektus paveikslėliuose.

2.4. Caltech 101, Caltech 256

Šios dvi duomenų bazės buvo surinktos mokslininkų Fei-Fei Li, Marco Andreetto, Marc Aurelio Ranzato, Greg Griffin, Alex Holub, Pietro Perona iš Kalifornijos technologijos instituto.

Pirmoji pasirodė Caltech 101 [FFP04], [Cal07a] duomenų bazė 2003 m. Joje yra 9144 paveikslėliai suskirstyti į 102 kategorijas (101 objektas ir viena kategorija – fonas, triukšmas). Kiekvienoje iš kategorijų yra nuo 31 iki 800 paveikslėlių (vidutiniškai 90).

Antroji duomenų bazė Caltech 256 [GHP07],[Cal07b] buvo surinkta 2006 metais, joje yra 30608 paveikslėliai suskirstyti į 257 kategorijas (256 objektai ir viena kategorija – fonas, triukšmas). Kiekvienoje iš kategorijų yra nuo 80 iki 827 paveikslėlių (vidutiniškai 119). Pavyzdiniai šios duomenų bazės paveikslėliai parodyti 31 paveiksle.

31 paveikslas. Caltech 256 duomenų bazės pavyzdiniai paveikslėliai.

Šiose duomenų bazėse yra daiktų, priklausančių tai pačiai kategorijai, bet atrodančių skirtingai, pavyzdžiui, kategorijoje „lėktuvai“ yra ir karinių lėktuvų nuotraukų, ir keleivinių, ir taip pat žaislinių modelių bei stilizuotų piešinių. Taigi, šioje duomenų bazėse esančios kategorijos atspindi sąvokas, bet ne konkrečius daiktus.

Autoriai nerekomenduoja naudoti šios duomenų bazės lokalizacijos testavimui.

3. Analizė

Pirmiausia reikia pasirinkti taškinius požymius, kurie bus naudojami klasterizavimo algoritme. Taškinių požymių analizė yra aprašyta 3.1. skyrelyje. Tuomet paveikslėlių palyginimo algoritmas bus analizuojamas 3.2. skyrelyje, ir galiausiai, klasterizavimas demonstruojamas 3.3. skyrelyje.

3.1. Taškinių požymių analizė

Taškinius požymius sudaro jų detektoriai (algoritmas, lokalizuojantis ypatingus taškus paveikslėlyje) bei deskriptoriai (algoritmas, aprašantis tų taškų aplinką tam tikru būdu; lyginant du panašių taškų deskriptorius gauname mažą atstumą tarp jų, o lyginant dviejų visiškai skirtingų taškų deskriptorius gauname didelį atstumą tarp jų).

3.1.1. Ypatingų taškų detektoriai

Taškinių požymių detektoriai buvo pristatyti literatūros apžvalgoje, 1.2. skyrelyje. Straipsniuose [MS04] ir [MTS+05] buvo lyginami taškinių požymių detektoriai. Buvo parodyta, kad MSER, Hessian-Affine bei Harris-Affine detektoriai yra patikimiausi, t.y. šie detektoriai atpažįsta tuos pačius taškus pasuktuose, atitolintuose, kitaip apšviestuose paveikslėliuose. D.Lowe straipsnyje [Low04] teigia, kad SIFT algoritmas duoda patikimesnius rezultatus nei Harris-Affine (kitų dviejų detektorių dar nebuvo nagrinėta). Straipsnyje [MTS+05] tyčia nenagrinėjamas SIFT detektorius, nes, pasak autorių, jis randa ypatingąjį tašką tik tam tikrame paveikslėlio dydyje (SIFT detektorius pirmiausia suranda, kokiame dydyje buvo tas ypatingasis taškas ir tada jį lokalizuoja).

Straipsnio [MTS+05] autoriai sugalvojo tam tikrą taškinių požymių detektorių testavimo metodiką. Pasirinktu algoritmu yra randami ypatingi taškai duomenų bazės *Affine Covariant Features* paveikslėliuose. Imamos paveikslėlių sekos (nuo pirmojo iki šeštojo paveikslėlio), kuriose iš anksto yra suskaičiuotos afininės transformacijos, nusakančios kaip iš pirmojo paveikslėlio gauti kurį nors kitą. Žinodami tikslią afininę transformaciją bei algoritmo surastus ypatingus taškus (bei kartu tam tikro dydžio jų aplinką), galime nustatyti algoritmo daromos klaidos dydį, t.y. kiek nesutampa pirmojo paveikslėlio ir kito sekos paveikslėlio detektorių aplinkos.

Darbo metu buvo praktiškai analizuotas SIFT detektorius pagal [MTS+05] metodiką. [Aff07] tinklalapyje yra pateikta MATLAB programa, tikrinanti įvairių taškinių požymių algoritmų pakartojamumą. Veikianti SIFT algoritmo realizacija yra pateikta [Key07] tinklalapyje. SIFT programa duotame paveikslėlyje suranda ypatinguosius taškus (taškų koordinates, jų charakteringą dydį, taškų kryptį) bei pateikia taškų deskriptorius. SIFT algoritmas buvo testuotas parenkant autoriaus rekomenduotus parametrus: iš pradžių yra didinamas paveikslėlis, naudojama tiksli taško lokalizacija, ypatingų taškų kontrastas 0.04. SIFT deskriptorius buvo transformuotas pagal [Aff07] tinklalapyje nurodytą formatą:

$$u\ v\ a\ b\ c\ ,$$

čia u ir v yra taško koordinatės paveikslėlyje (koordinates (0,0) reiškia kairįjį viršutinį paveikslėlio kampą). O a, b ir c koeficientai yra iš lygties, aprašančios ypatingojo taško aplinkos dydį - elipsę:

$$a(x-u)(x-u)+2b(x-u)(y-v)+c(y-v)(y-v)=1\ .$$

SIFT aprašymo transformacija yra gan paprasta:

$$u = sift(x),$$
$$v = sift(y),$$

```
a = 1 / (4.5 * sift(scale)),
b = 0,
c = 1 / (4.5 * sift(scale)).
```

Naujo formato SIFT detektorius buvo naudotas tikrinant SIFT algoritmo pakartojamumą. Kiekvienoje paveikslėlių sekoje buvo imami pakartojamumo procentai su 40% klaidos galimybe (tokia pati klaidos galimybė buvo naudojama [MTS+05] straipsnyje).

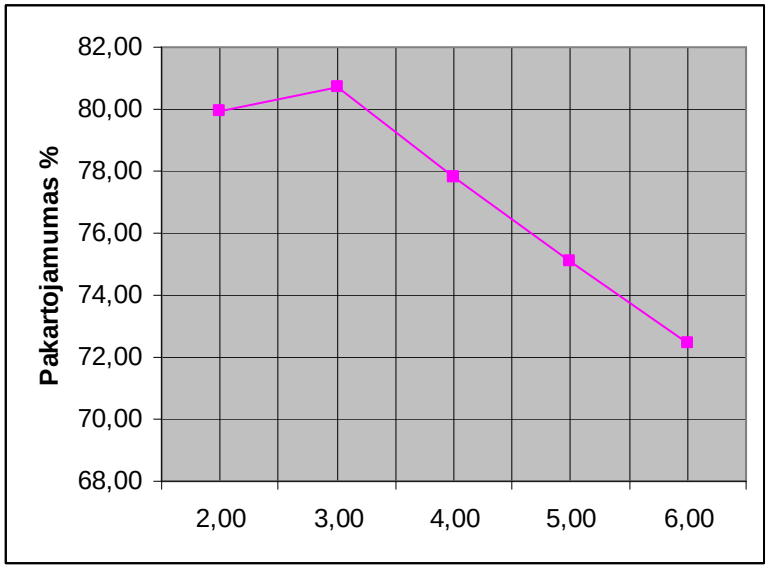
Lentelėje 3 yra parodyti SIFT algoritmo pakartojamumo procentai ir sutapusių taškų skaičiai *Blur (su motociklais)* paveikslėlių sekoje (32 paveikslas). Šie skaičiai yra imami lyginant image1.ppm paveikslėlį visais kitais paveikslėliais ir leidžiant deskriptorių aplinkos 40% persiklojimo klaidos galimybę.

Didėjantis išliejimas	**2,00**	**3,00**	**4,00**	**5,00**	**6,00**
Sutampančių taškų skaičius	1686,00	1135,00	641,00	434,00	308,00
Pakartojamumo procentas	79,94%	80,72%	77,79%	75,09%	72,47%

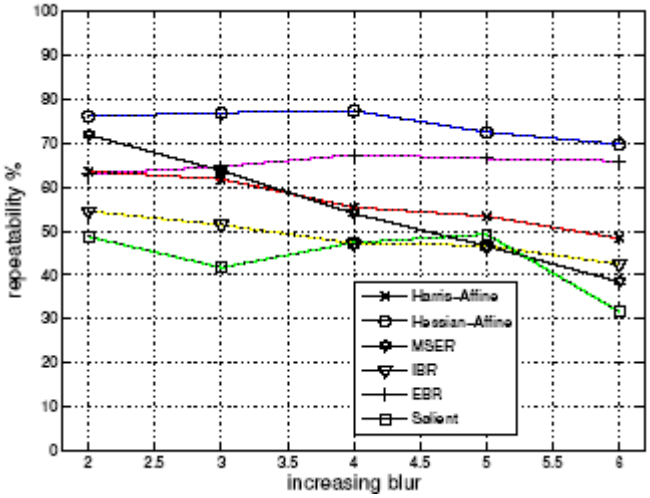
3 lentelė. SIFT detektoriaus rezultatai *Blur (su motociklais)* paveikslėlių sekoje.

32 paveikslas. Pavyzdinis paveikslėlis iš *Blur (su motociklais)* sekos.

33 paveiksle pavaizduotas pakartojamumo procentų grafikas SIFT algoritmui ir 34 paveiksle grafikas iš [MTS+05] straipsnio su kitų algoritmų rezultatais. Matyti, kad SIFT detektorius veikia taip pat gerai kaip tiksliausias straipsnyje analizuotas detektorius Hessian-Affine.

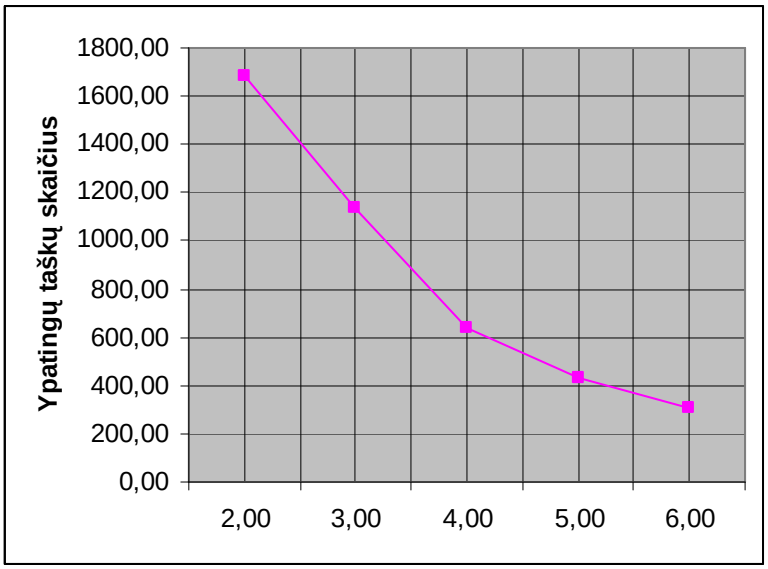


33 paveikslas. SIFT algoritmo pakartojamumo procentai *Blur (su motociklais)* sekoje.

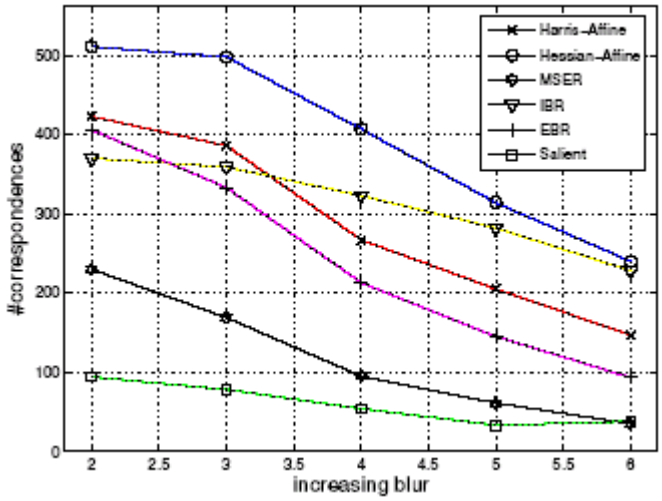


34 paveikslas. Įvairių detektorių pakartojamumo procentai *Blur (su motociklais)* sekoje. Paveikslėlis iš [MTS+05] straipsnio.

Paveiksluose 35 bei 36 pavaizduoti atitinkamai SIFT algoritmo bei straipsnyje pristatytų algoritmų ypatingųjų taškų skaičiai. SIFT algoritmas suranda dvigubai daugiau taškų nei Hessian-Affine, taigi taškų pakartojamumo procentai pirmajam jų yra patikimesni už Hessian-Affine.



35 paveikslas. SIFT algoritmo ypatingų taškų skaičiai *Blur (su motociklais)* sekoje.



36 paveikslas. Įvairių detektorių ypatingų taškų skaičiai *Blur (su motociklais)* sekoje. Paveikslėlis iš [MTS+05] straipsnio.

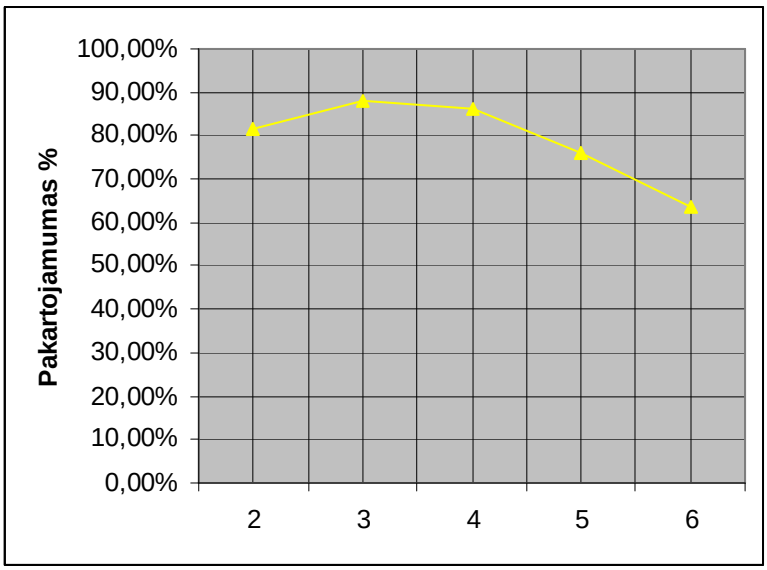
4 lentelėje yra parodyti SIFT algoritmo pakartojamumo procentai ir sutapusių taškų skaičiai *Blur (su medžiais)* paveikslėlių sekoje (37 paveikslas). Šie skaičiai yra imami lyginant image1.ppm paveikslėlį visais kitais paveikslėliais ir leidžiant 40% persiklojimo klaidos galimybę.

Didėjantis išliejimas	2	3	4	5	6
Sutampančių taškų skaičius	11647	17157	9791	4275	2024
Pakartojamumo procentas	81,38%	88,22%	86,07%	75,99%	63,65%

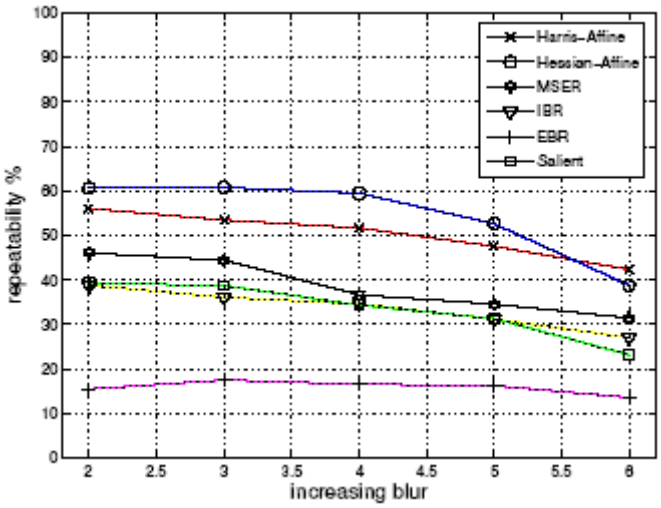
4 lentelė. SIFT detektoriaus rezultatai *Blur (su medžiais)* paveikslėlių sekoje.

37 paveikslas. Pavyzdinis paveikslėlis iš *Blur (su medžiais)* sekos.

38 paveiksle pavaizduotas pakartojamumo procentų grafikas SIFT algoritmui ir 39 paveiksle grafikas iš [MTS+05] straipsnio su kitų algoritmų rezultatais. SIFT detektoriaus pakartojamumas yra daug didesnis negu kitų algoritmų.

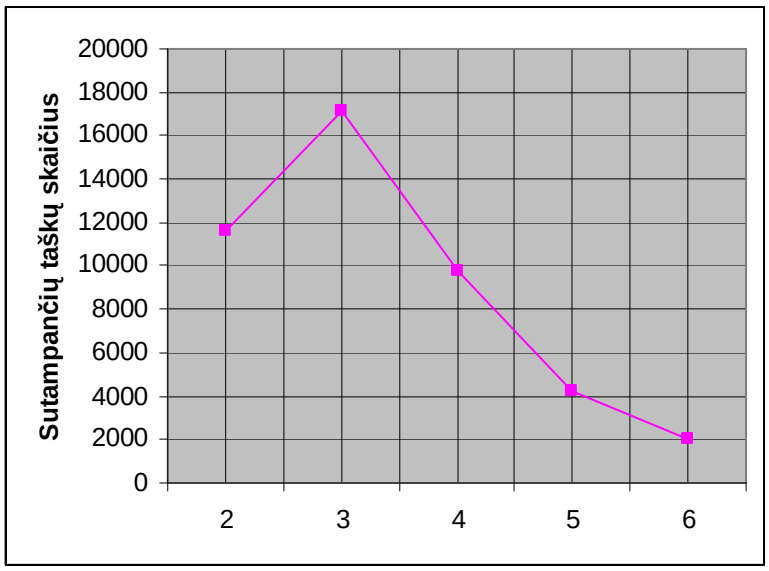


38 paveikslas. SIFT algoritmo pakartojamumo procentai *Blur (su medžiais)* sekoje.

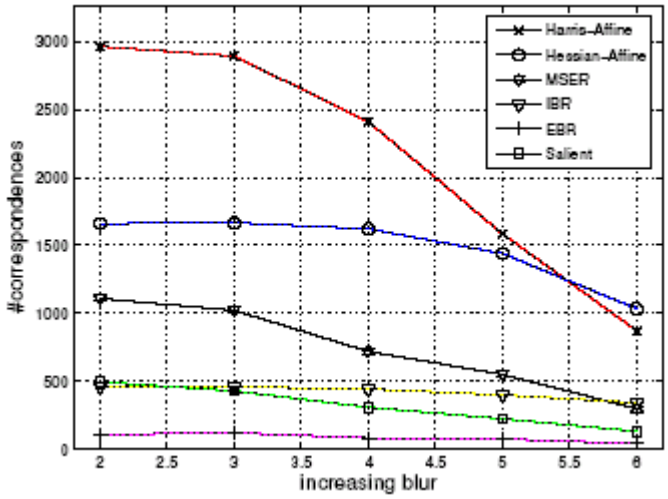


39 paveikslas. Įvairių detektorių pakartojamumo procentai *Blur (su medžiais)* sekoje. Paveikslėlis iš [MTS+05] straipsnio.

Paveiksluose 40 bei 41 pavaizduoti atitinkamai SIFT algoritmo bei straipsnyje pristatytų algoritmų ypatingųjų taškų skaičiai. Sutampantys taškai yra paimti su 40% persiklojimo klaida. SIFT detektorius randa nuo šešių iki dviejų kartų daugiau taškų nei kiti algoritmai. Taigi, jo pakartojamumo procentas yra daug patikimesnis už kitų detektorių, išskiriančių mažiau ypatingų taškų.



40 paveikslas. SIFT algoritmo ypatingų taškų skaičiai *Blur (su medžiais)* sekoje.



41 paveikslas. Įvairių detektorių ypatingų taškų skaičiai *Blur (su medžiais)* sekoje. Paveikslėlis iš [MTS+05] straipsnio.

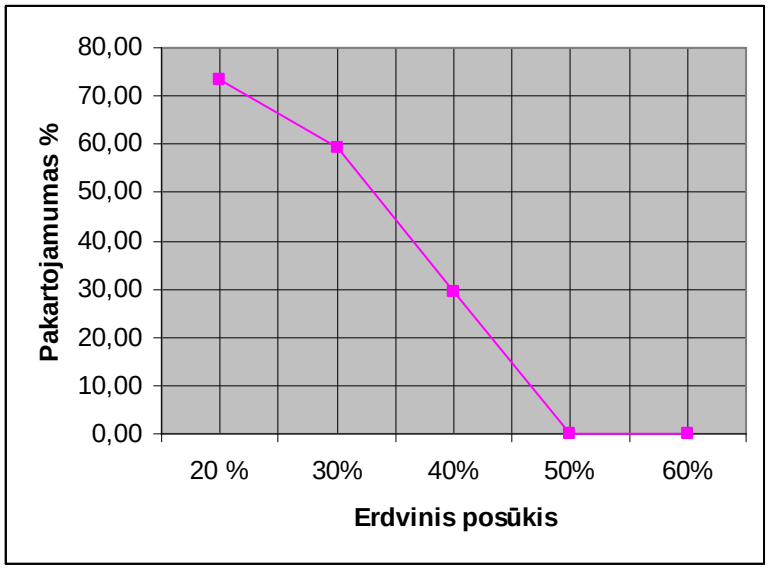
5 lentelėje yra parodyti SIFT algoritmo pakartojamumo procentai ir sutapusių taškų skaičiai *Viewpoint (su grafiti siena)* paveikslėlių sekoje (42 paveikslas). Šie skaičiai yra imami lyginant image1.ppm paveikslėlį visais kitais paveikslėliais ir leidžiant 40% persiklojimo klaidos galimybę.

Didėjantis žiūrėjimo kampas	**20 %**	**30%**	**40%**	**50%**	**60%**
Sutampančių taškų skaičius	2023,00	1572,00	708,00	0,00	0,00
Pakartojamumo procentas	73,54	59,10	29,57	0,00	0,00

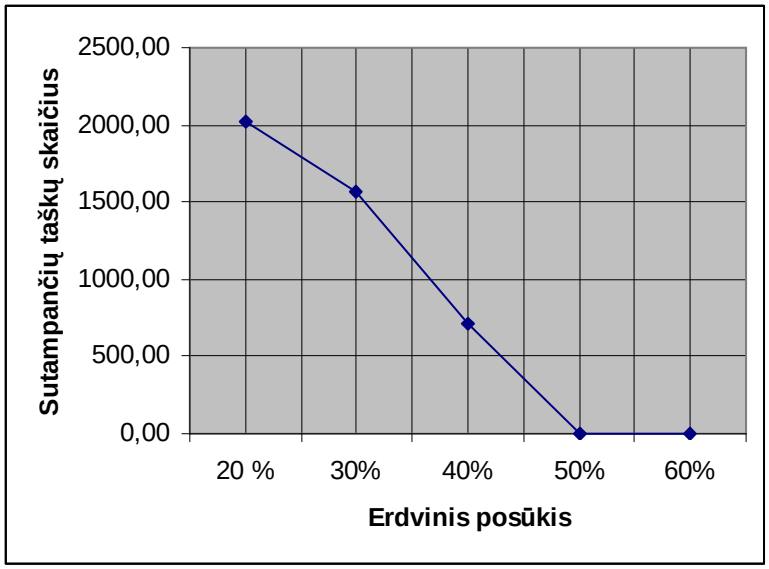
5 lentelė. SIFT detektoriaus rezultatai *Viewpoint (su grafiti siena)* paveikslėlių sekoje.

42 paveikslas. Pavyzdinis paveikslėlis iš *Viewpoint (su grafiti siena)* sekos.

43 bei 44 paveiksluose grafiškai pavaizduoti SIFT algoritmo pakartojamumo procentai bei sutampančių taškų skaičiai su *Viewpoint (su grafiti siena)* seka. Buvo imtas 40% sutapimo klaidos procentas. Straipsnyje [MTS+05] nėra pateikiami panašūs grafikai; yra pateikiami kitokie grafikai analizuojantys šią paveikslėlių seką skirtingais algoritmais, tačiau nėra paaiškinti šių grafikų gavimo metodai.

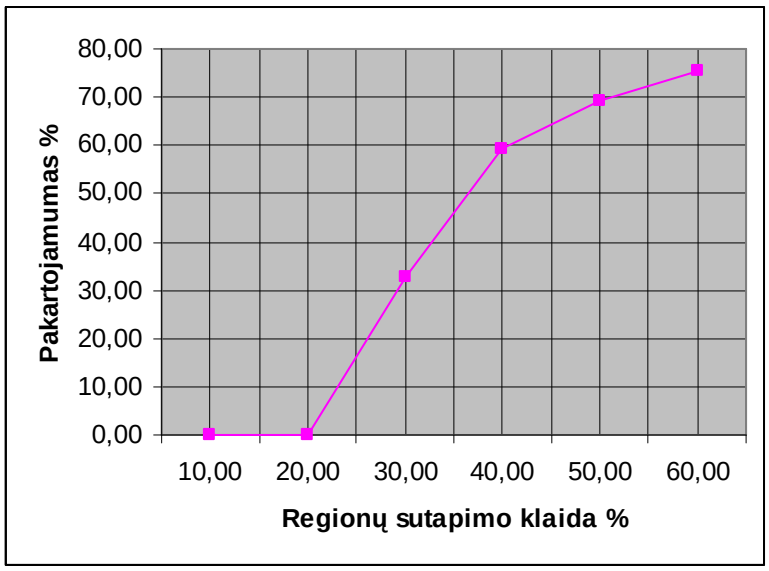


43 paveikslas. SIFT algoritmo pakartojamumo procentai *Viewpoint (su grafiti siena)* sekoje.

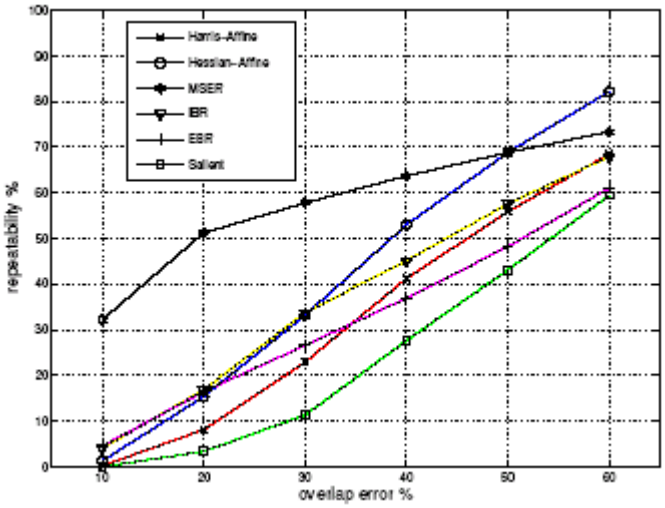


44 paveikslas. SIFT algoritmo ypatingų taškų skaičiai *Viewpoint (su grafiti siena)* sekoje.

45 paveiksle pavaizduotas pakartojamumo procentų grafikas SIFT algoritmui ir 46 paveiksle grafikas iš [MTS+05] straipsnio su kitų algoritmų rezultatais. SIFT algoritmo pakartojamumas (pirmojo ir trečiojo paveikslėlio) yra panašus į Hessian-Affine, ir gan daug nusileidžia Harris-Affine algoritmui, duodančiam geriausius pakartojamumo rezultatus šiame grafike.



45 paveikslas. SIFT algoritmo pakartojamumo procentai, lyginant *Viewpoint (su grafiti siena)* 1 paveikslėlį su 3 paveikslėliu. Pakartojamumo procentas pagal sutapimo paklaidas.



46 paveikslas. Įvairių detektorių pakartojamumo procentas *Viewpoint (su grafiti) 1 ir 3 paveikslėliuose*. Pakartojamumo procentas pagal sutapimo paklaidas. Grafikas iš [MTS+05] straipsnio.

[MTS+05] straipsnis nepateikia tikslesnės analizės su likusiomis paveikslėlių sekomis. SIFT algoritmas buvo testuotas su *Zoom + rotation (su žole)* bei *Light* sekomis

6 lentelėje yra parodyti SIFT algoritmo pakartojamumo procentai ir sutapusių taškų skaičiai *Zoom + rotation (su žole)* paveikslėlių sekoje (47 paveikslas). 7 lentelėje parodyti MSER detektoriaus rezultatai.

	pav.1 su pav.2	**pav.1 su pav.3**	**pav.1 su pav.4**	**pav.1 su pav.5**	**pav.1 su pav.6**
Sutampančių taškų skaičius	1387,00	510,00	163,00	46,00	6,00
Pakartojamumo procentas	81,25%	36,98%	13,69%	5,39%	1,26%

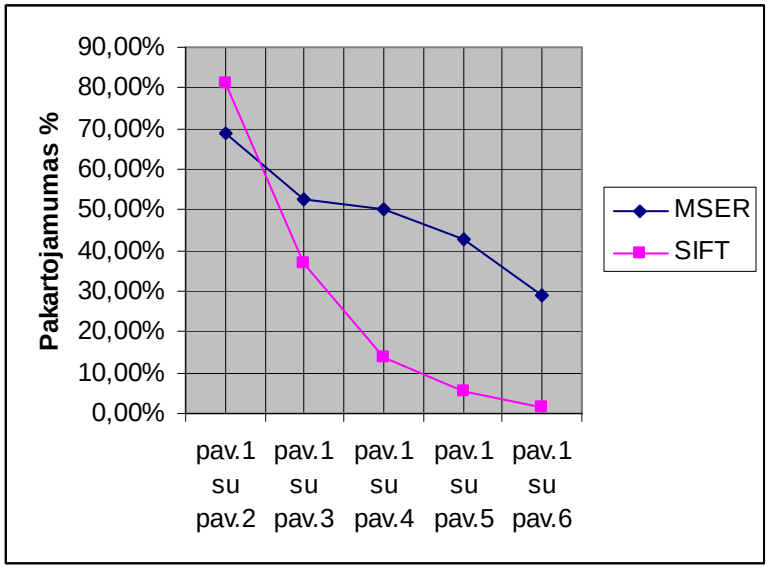
6 lentelė. SIFT detektoriaus rezultatai *Blur (su motociklais)* paveikslėlių sekoje.

47 paveikslas. Pavyzdinis paveikslėlis iš *Zoom + rotation (su žole)* sekos.

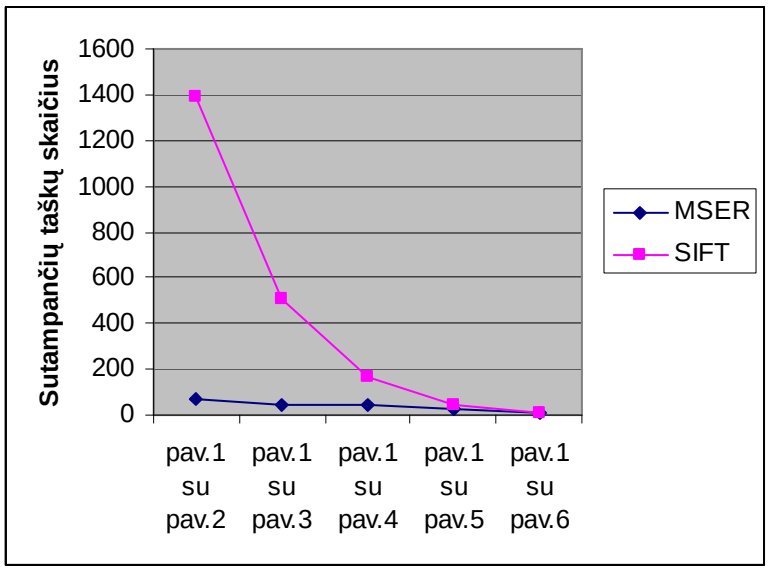
	pav.1 su pav.2	pav.1 su pav.3	pav.1 su pav.4	pav.1 su pav.5	pav.1 su pav.6
Sutampančių taškų skaičius	67	47	45	27	11
Pakartojamumo procentas	69,07%	52,81%	50,00%	42,90%	28,90%

7 lentelė. MSER detektoriaus rezultatai *Blur (su motociklais)* paveikslėlių sekoje.

48 paveiksle pavaizduotas pakartojamumo procentų grafikas, o 49 paveiksle sutapusių taškų skaičiai SIFT ir MSER algoritmams su *Zoom + rotation (su žole)* duomenų seka imant 40% sutapimo paklaidą. Iš paveikslų matyti, kad pakartojamumas smarkiai mažėja didėjant posūkiui bei keičiantis vaizdo dydžiui. Šioje sekoje vaizdo pokyčiai yra drastiški, todėl toks mažas pakartojamumo procentas iš dalies yra įtakojamas nedidelio sutampančio paveikslėlių ploto. Nors MSER detektoriaus pakartojamumas didesnis (1 paveikslėlį lyginant su 3, 4, 5 bei 6 paveikslėliais), tačiau MSER algoritmas išskiria daug mažiau ypatingųjų taškų, kurių skaičių dinamika parodyta 48 paveiksle.



48 paveikslas. SIFT bei MSER algortimų pakartojamumo procentai *Zoom + rotation (su žole)* sekoje.



49 paveikslas. SIFT bei MSER algoritmų ypatingų taškų skaičiai *Zoom + rotation (su žole)* sekoje.

8 lentelėje parodyti SIFT algoritmo pakartojamumo procentai ir sutapusių taškų skaičiai *Light* paveikslėlių sekoje (50 paveikslas). Šioje sekoje keičiant kameros parametrus yra gauti keli paveikslėliai vaizduojantys tą patį pastatą. Nei žiūrėjimo kryptis, nei atstumas ar posūkis nebuvo keisti. 9 lentelėje yra pateikti MSER algoritmo sutampančių taškų skaičiai bei pakartojamumo procentai su ta pačia paveikslėlių seka.

	pav.1 su pav.2	**pav.1 su pav.3**	**pav.1 su pav.4**	**pav.1 su pav.5**	**pav.1 su pav.6**
Sutampančių taškų skaičius	2280,00	1939,00	1660,00	1470,00	1195,00
Pakartojamumo procentas	78,06%	78,50%	77,97%	76,68%	76,31%

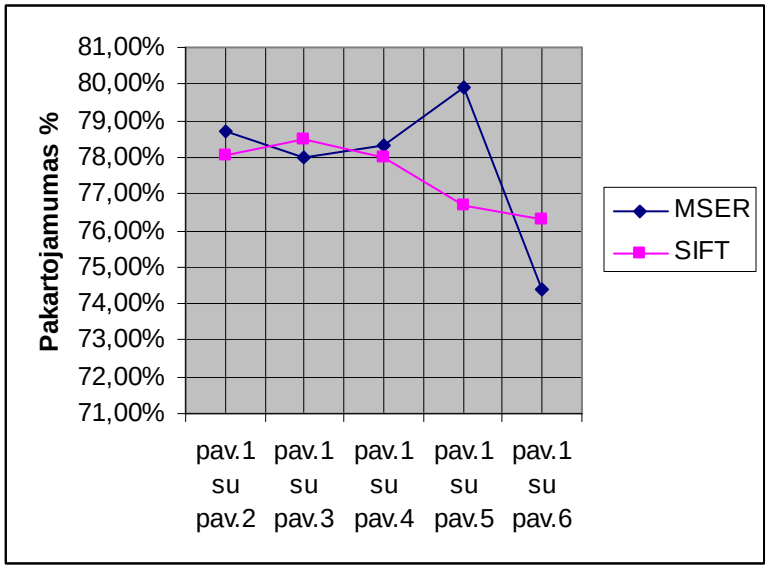
8 lentelė. SIFT detektoriaus rezultatai *Light* paveikslėlių sekoje.

50 paveikslas. Pavyzdinis paveikslėlis iš *Light* sekos.

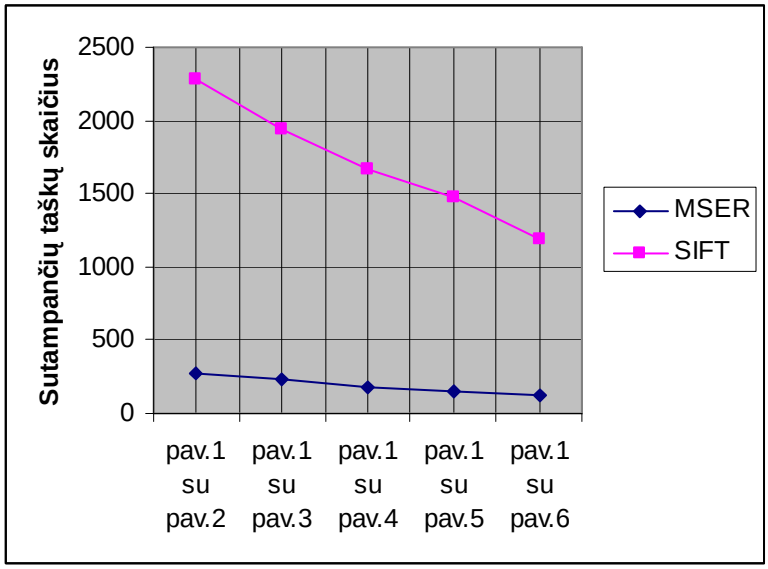
	pav.1 su pav.2	**pav.1 su pav.3**	**pav.1 su pav.4**	**pav.1 su pav.5**	**pav.1 su pav.6**
Sutampančių taškų skaičius	277	227	184	155	119
Pakartojamumo procentas	78,70%	78,00%	78,30%	79,90%	74,40%

9 lentelė. MSER detektoriaus rezultatai *Light* paveikslėlių sekoje.

51 paveiksle pavaizduotas pakartojamumo procentų grafikas, o 52 sutampančių taškų skaičiai SIFT bei MSER algoritmams su *Light* duomenų seka imant 40% sutapimo paklaidą. Šioje sekoje esantys paveikslėliai yra panašaus kontrasto, nors jų šviesumas ir skiriasi. SIFT algoritmas yra atsparus apšvietimo pokyčiams su sąlyga, kad kontrastas išlieka toks pat, taigi, šioje sekoje SIFT algoritmo pakartojamumo procentai yra gan dideli. SIFT detektoriaus pakartojamumas yra panašus į MSER detektoriaus, tačiau pastarasis randa žymiai mažiau ypatingų taškų paveikslėlyje, todėl SIFT algoritmo veikimas yra patikimesnis.



51 paveikslas. SIFT bei MSER algoritmų pakartojamumo procentai *Light* sekoje.



52 paveikslas. SIFT bei MSER algoritmų ypatingųjų taškų skaičiai *Light* sekoje.

Eksperimentiškai įsitikinta, kad SIFT detektorius duoda panašius, o kai kuriais atvejais ir geresnius pakartojamumo rezultatus nei Hessian-Affine algoritmas. Kadangi, [Low04] teigiamas SIFT pranašumas prieš Harris-Affine, o [MTS+05] parodomas Harris-Affine bei Hessian-Affine panašus veikimas, galima daryti išvadą, kad SIFT algoritmas nenusileidžia Harris-Affine, Hessian-Affine bei MSER detektoriams.

3.1.2. Ypatingų taškų deskriptoriai

Taškinių požymių deskriptoriai buvo nagrinėti [MS05] straipsnyje. Jame nurodoma, kad GLOH bei SIFT deskriptoriai duoda patikimiausius rezultatus, t.y. tas pats taškas įvairiai transformuotuose paveikslėliuose atpažįstamas kaip tas pats taškas. Tačiau [MTS+05] straipsnyje tvirtinama, kad SIFT deskriptorius yra geresnis ir stabilesnis už kitus deskriptorius, taip pat būtent SIFT deskriptorius yra naudojamas šio straipsnio detektoriams testuoti.

[MGC07] straipsnyje patvirtinamas SIFT algoritmo patikimumas, jo atsparumas posūkiams, didinimams/mažinimams. [MGC07] autoriai naudoja SIFT bei dar kelis algoritmus kurdami maisto produktų bei buities prekių atpažinimo sistemą, pagelbėsiančią silpnaregiams žmonėms.

3.2. Taškinių požymių palyginimas

Taškinių požymių palyginimas susideda iš dviejų etapų:

1. Atstumo skaičiavimas tarp dviejų ypatingų taškų (randamos taškų poros)
2. Atstumo skaičiavimas tarp dviejų ypatingų taškų aibių (randamas atstumas (angl. *distance*) arba panašumas (angl. *similarity*) tarp dviejų paveikslėlių).

Atstumu čia bus vadinamas dydis, kurio minimali reikšmė nurodo, kad du objektai yra sutampantys. Panašumas tuo tarpu yra dydis, kurio minimali reikšmė pasako apie dviejų objektų skirtingumą, o maksimali – apie vienodus objektus. Panašumas dažnai būna normuojamas iki intervalo [0;1] ir naudojamas kaip koreliacijos koeficientas.

3.2.1. Atstumas tarp dviejų taškų

Pirmojo etapo palyginimas yra aprašytas [Low04] straipsnyje.

Paveikslėlio A ypatingieji taškai bus žymimi a_n, o paveikslėlio B ypatingieji taškai – b_m. Paveikslėliuose A ir B gali būti skirtingas taškų skaičius, todėl ypatingieji taškai žymimi skirtingais indeksais. Kiekvienas ypatingasis taškas yra aprašomas jo koordinatėmis (u,v) paveikslėlyje, charakteringu posūkiu, postūmiu ir dydžiu, o taip pat ypatingojo taško deskriptoriumi – vektoriumi iš 128 reikšmių (x_1, x_2,...x_{128}) (šio vektoriaus – lokalios aplinkos – skaičiavimas buvo aprašytas 1.2.1 skyrelyje).

Atstumas tarp dviejų ypatingųjų taškų yra apibrėžiamas kaip euklininio atstumo kvadratas tarp tų taškų deskriptorių:

$$d(a_n(x_1, x_2,.., x_{128}), b_m(x_1, x_2,.., x_{128})) = \sum_{i=1}^{128} (a_n(x_i) - b_m(x_i))^2 .$$

Tokiu būdu yra lyginamas kiekvienas paveikslėlio A taškas su kiekvienu paveikslėlio B tašku, ir randamos tokios ypatingųjų tašku poros, tarp kurių atstumas yra mažiausias.

SIFT algoritmo autorius pasiūlė būdą kaip patobulinti artimiausio kaimyno parinkimą – jei atstumų santykis tarp artimiausio kaimyno ir sekančio kaimyno yra didesnis už numatytą reikšmingumo slenkstį (*distRatio*), tuomet artimiausias kaimynas yra pasirenkamas, priešingu atveju – ypatingasis taškas neturi savo poros.

Ši atstumo metrika yra asimetrinė, t.y. jei paveikslėlio A ypatingajam taškui a_4 buvo rastas artimiausias kaimynas iš paveikslėlio B – b_2, tai taškui b_2 artimiausias kaimynas nebūtinai bus a_4.

Visai nesunku tokio tipo atstumo metriką pertvarkyti į simetrinę – tereikia atmesti visas taškų poras, kurios yra asimetrinės, t.y. nesutampa artimiausias kaimynas ieškant iš paveikslėlio A pusės ir iš B pusės.

Algoritmo bandymuose bus analizuojamas tiek asimetrinis, tiek ir simetrinis porų radimo būdas bei testuojama su skirtingais *distRatio* parametrais.

3.2.2. Atstumas tarp dviejų paveikslėlių

Antrajame atstumo radimo etape naudojamos taškų poros suskaičiuotos ankstesniame žingsnyje. Atstumas tarp dviejų paveikslėlių yra nustatomas kaip suma visų atstumų tarp taškų porų padalinta iš absoliutaus maksimumo:

$$d(A,B) = \frac{\sum_{a_n \in A, b_m \in B} d(a_n, b_m)}{\max TheoreticalDist},$$

čia d(*, *) nurodo atstumo funkciją, A, B – paveikslėlių SIFT taškų aibės, a_n – n-tojo ypatingojo taško, priklausančio paveikslėliui A, deskriptorius, o b_m – m-tojo ypatingojo taško, priklausančio paveikslėliui B, deskriptorius.

Siekiant pagerinti atstumo funkciją, vietoj euklidinio atstumo tarp porų, yra pasirenkama panašumo metrika:

$$sim(a_n, b_m) = threshold - d(a_n - b_m),$$

čia d(*,*) nurodo euklidinį atstumą, o *threshold* – pasirinktą maksimalų atstumą. Teorinis maksimalus atsutmas tarp dviejų SIFT taškų yra 128, todėl parametras *threshold* yra pasirenkamas kaip pusė (64) arba trečdalis (43) to atstumo; laikoma, kad labiau nutolusios taškų poros yra visiškai nepanašios, todėl neverta jų nagrinėti. Tuomet panašumas tarp paveikslėlių bus skaičiuojamas:

$$sim(A,B) = \frac{\sum_{a_n \in A, b_m \in B} sim(a_n, b_m)}{\max TheoreticalDist},$$

čia max*TheoreticalDist* lygus taškų porų skaičiui padaugintam iš *threshold*.

Maksimalus atstumas tarp dviejų SIFT taškų yra 128, todėl minimalus leidžiamas nutolimas (slenkstis) parenkamas pusė arba trečdalis, t.y. 64 arba 43. Tuomet maksimalus teorinis atstumas (maxTheoreticalDistance) bus lygus slenksčiui padaugintam iš rastų taškų porų.

Naudojant panašumo metriką vietoj atstumo metrikos, yra atmetamos labiau nutolusios ypatingųjų taškų poros, o tai padidina algoritmo atsparumą triukšmams.

3.2.3. Praktiniai rezultatai

Testavimui buvo naudota ALOI paveikslėlių duomenų bazė. Kadangi SIFT algoritmas analizuoja juodai baltus paveikslėlius, iš duomenų bazės buvo pasirinkti tik tokie paveikslėliai.

ALOI paveikslėlių duomenų bazėje yra 1000 objektų, jie fotografuoti keturiomis skirtingomis sąlygomis:

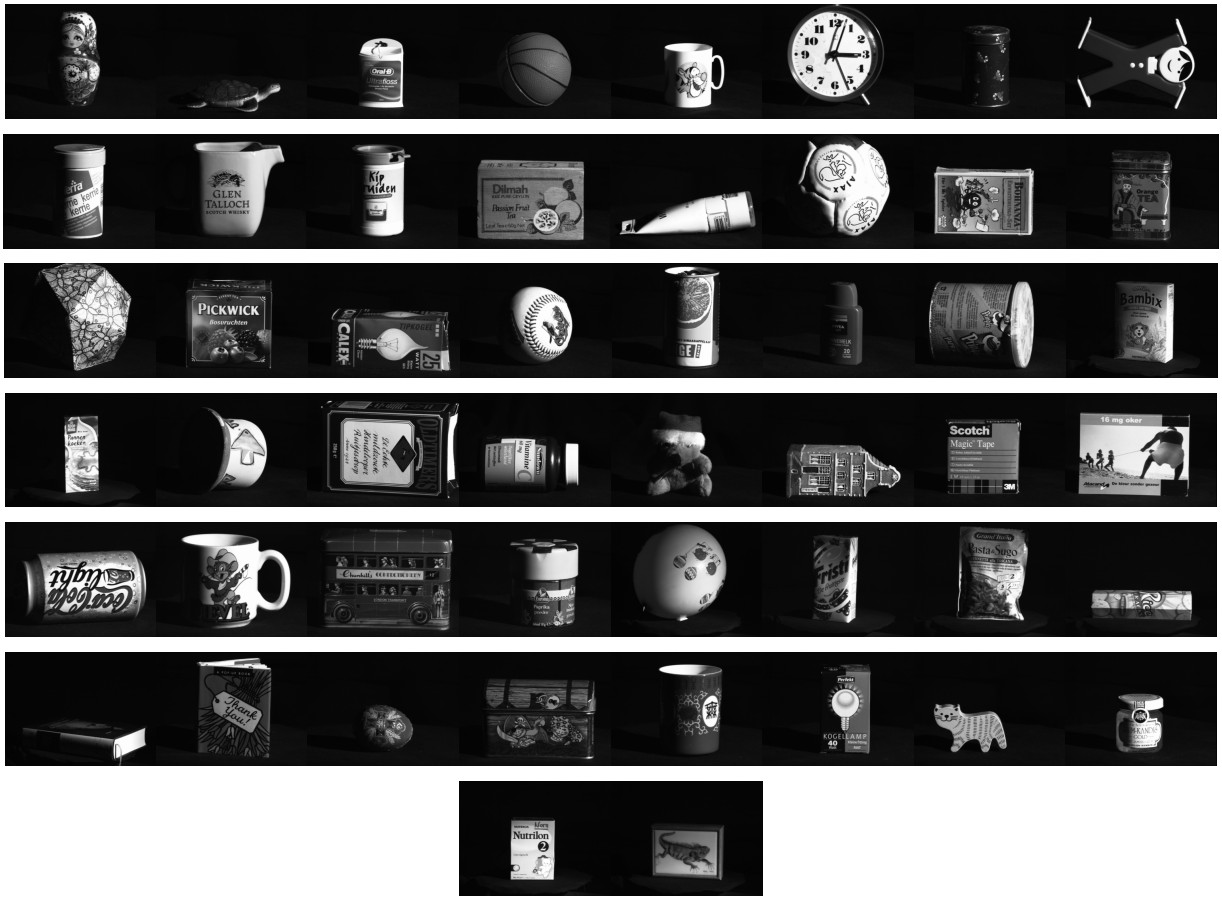
a) kai keičiasi apšvietimo spalva (objekto spalva taip pat pakinta,

b) kai keičiasi apšvietimo intensyvumas ir kampas,

c) kai keičiasi kameros žiūrėjimo kampas (kas 5 laipsnius),

d) trimačio vaizdo komponentai (objektas iš priekio, kairės pusės ir dešinės pusės).

Testavimui pasirinkta apšvietimo intensyvumo pasikeitimo aibė, iš jos paimti 50 objektų, pavaizduoti 53 paveiksle.



53 paveikslas. Testavimui naudoti objektai iš ALOI paveikslėlių duomenų bazės (vienas iš 24 objektų paveikslėlių).

Praktinių bandymų metu tirtos atstumo skaičiavimo algoritmo modifikacijos bei jų slenksčiai:

- asimetrinis bei simetrinis ypatingųjų taškų porų radimo būdai;
- skirtingos parametro *distRatio* reikšmės – 0,6 ir 1.0 (ypatingųjų taškų porų radime);
- skirtingos parametro *threshold* reikšmės – 0,3; 0,5 ir 0,8 (paveikslėlių palyginime).

Algoritmo modifikacijos tarpusavyje lyginamos naudojant ROC kreives tokiu būdu:

1. Pagal paveikslėlio numerius galime nuspręsti, ar du paveikslėliai vaizduoja tą patį objektą, ar skirtingus.
2. Suskaičiuoti panašumus tarp visų paveikslėlių porų iš turimos duomenų aibės.
3. Surūšiuoti panašumus nuo didžiausio iki mažiausio. Braižysim grafiką nuo kairiojo krašto:
   c) jei paveikslėliai vaizduoja tą patį objektą, brėžti liniją vertikaliai viršun;
   d) jei paveikslėliai vaizduoja skirtingus objektus, brėžti liniją horizontaliai dešinėn.

Kuo statesnis gaunasi ROC grafikas, tuo geriau pasirinktas algoritmas identifikuoja objektus, atskiria vienodus objektus nuo skirtingų.

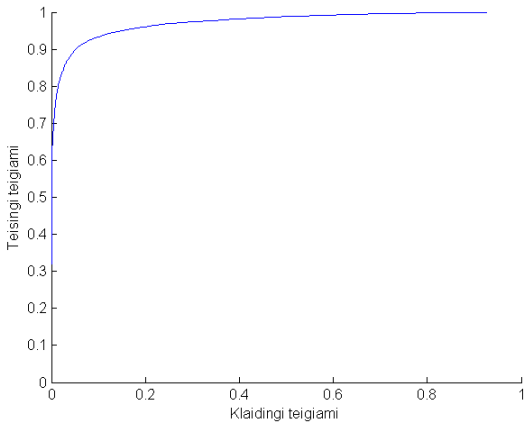
Atstumo tarp paveikslėlių skaičiavimo algoritmas susideda iš dviejų žingsnių – ypatingųjų taškų porų radimo ir atstumo tarp paveikslėlių radimas remiantis taškų poromis. Pirmajame etape – SIFT taškų lyginime turi būti nustatytas parametras – minimalus santykis tarp atstumo iki artimiausio kaimyno ir atstumo iki sekančio artimiausio kaimyno (toliau bus žymimas *distRatio*). Testavimui buvo pasirinktos dvi reikšmės:

a) *distRatio*=1 – nėra reikalavimo, kad atstumai tarp artimiausio kaimyno ir sekančio artimiausio skirtųsi;

b) *distRatio*=0.6 – minėtieji atstumai turi skirtis bent 60%.

Antrajame etape – paveikslėlių lyginime – svarbus parametras yra slenkstis, kuris nurodo, kad pora taškų yra pernelyg tolima, kad į ją būtų atsižvelgiama. Kadangi SIFT deskriptoriai yra normuoti iki vienetinio ilgio, todėl maksimalus atstumas tarp dviejų deskriptorių yra 1. Tikslinga minimalų slenkstį (toliau bus žymimas *threshold*) pasirinkti kaip „0,3“, „0,5“ bei „0,8“.

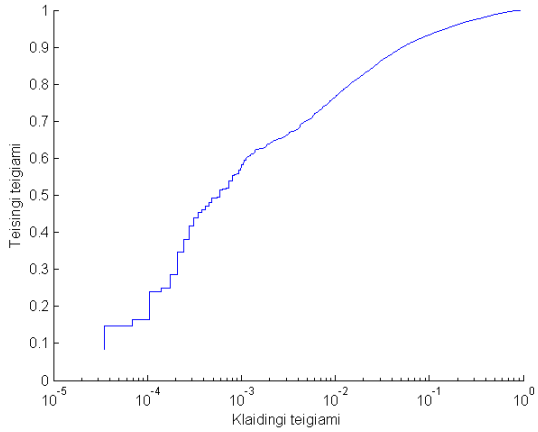
Pradžioje buvo pasirinktas SIFT taškų lyginimo parametras *distRatio*=1 (atstumas iki artimiausio kaimyno ir sekančio artimiausio kaimyno gali būti lygus) bei paveikslėlių panašumo rodiklis *threshold*=0,5 (jei atstumas tarp dviejų ypatingųjų taškų poroje yra didesnis už pusę maksimalaus įmanomo atstumo, tai reikia atmesti šią porą). Naudotas asimetrinis ypatingųjų taškų porų radimo algoritmas. Gautoji ROC kreivė pavaizduota 54 paveiksle.



54 paveikslas. ROC kreivė, naudojant parametrus *distRatio*=1 ir *threshold*=0,5.

Matome, kad iš pradžių kreivė kyla į viršų – panašumas tarp to paties objekto paveikslėlių (angl. *genuine*) yra didžiausias, vėliau vis labiau darosi horizontalesnė – pradeda maišytis atstumai tarp to paties objekto ir skirtingų objektų (angl. *impostor*) paveikslėlių.

Svarbiausia kreivės dalis yra prie pat taško (0;0). Ten galime pastebėti, ar algoritmas gerai atpažįsta vienodus objektus. 55 paveiksle pavaizduota ROC kreivė su logaritmine Ox ašimi.

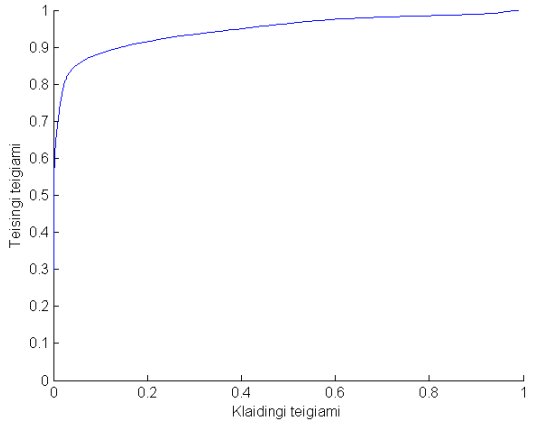


55 paveikslas. ROC kreivė, naudojant *distRatio*=1 ir *threshold*=0,5. Ox ašis logaritminėje skalėje.

ROC kreivės įvertinimo parametras EER (angl. *Equal Error Rate*) pasako, ties kokia reikšme FAR ir FRR klaidos yra lygios, t.y. atmetame tiek pat teisingų atvejų, kiek ir priimtume klaidingų atvejų. Kuo mažesnis EER dydis, tuo mažiau klaidų daro atpažinimo algoritmas. EER turėtų būti naudojamas tik kaip pirminis įvertis, tačiau jis neturėtų būti vienintelis algoritmo gerumo nustatymo įrankis.

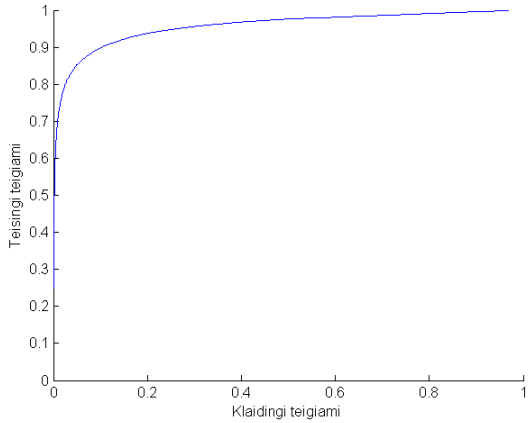
ROC kreivės, pavaizduotos 55 paveiksle, EER lygus 0,0793, arba 7,9 % atvejų algoritmo sprendimas yra klaidingas.

Sekantis algoritmo parametrų rinkinys – *distRatio* = 1, *threshold* = 0,8 – pavaizduotas 56 paveiksle. Vizualiai ši kreivė yra labai panaši į ankstesnę (parodytą 52 paveiksle); jos EER lygus 0,1125.



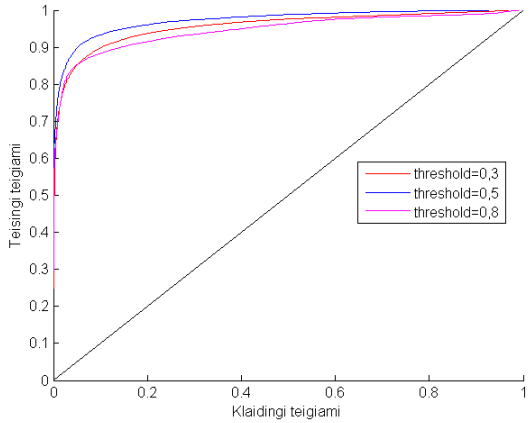
56 paveikslas. ROC kreivė, naudojant distRatio=1 ir threshold=0,8.

Siekiant labiau įtakoti skirtingų objektų klaidingus panašumus (FAR), *threshold* parametras buvo sumažintas iki 0,3, t.y. atstumas tarp ypatingųjų taškų porų turi būti ne didesnis nei maksimalus įmanomas. Gautoji ROC kreivė parodyta 57 paveiksle; jos EER lygus 0,1051.

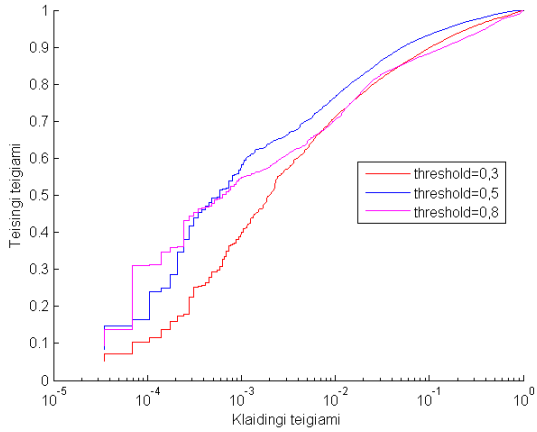


57 paveikslas. ROC kreivė, naudojant *distRatio*=1 ir *threshold*=0,3.

Visos trys ROC kreivės, naudojant skirtingus *threshold* parametrus, vizualiai atrodo vienodos, todėl siekiant pastebėti nedidelius skirtumus jos nubraižytos viena ant kitos. Grafikai tiesinėje ir pusiau logaritminėje skalėse pavaizduoti atitinkamai 58 ir 59 paveiksluose.



58 paveikslas. ROC kreivės, naudojant *distRatio*=1 ir skirtingus *threshold* (*threshold* = 0,3 raudona linija; *threshold* = 0,5 mėlyna linija; *theshold* = 0,8 rožinė linija).



59 paveikslas. ROC kreivės, naudojant *distRatio*=1 ir skirtingus *threshold* (*threshold* = 0,3 raudona linija; *threshold* = 0,5 mėlyna linija; *theshold* = 0,8 rožinė linija). Logaritminė Ox ašis.

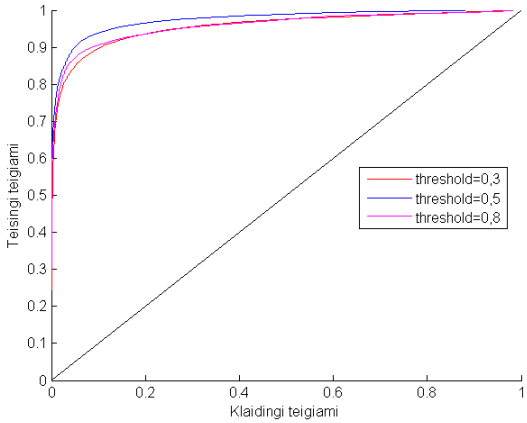
Tiesinėje ašyje visų trijų kreivių pradžia sutampa, o toliau matosi nežymūs skirtumai. Mėlyna kreivė (*threshold*=0,5) yra stačiausia, todėl šis parametras būtų tinkamiausias paveikslėlių palyginimo algoritmui. Logaritminėje Ox ašyje matosi, kad rožinės linijos pradžia yra stačiausia

(*threshold* = 0,8), todėl šis parametras labiau tiktų paveikslėlių palyginimo algoritmui. Analizuojant ROC kreivę didžiausias dėmesys turi būti kreipiamas į jos pradžią, t.y. (0;0) tašką ir toliau kylančią kreivę – kuo kreivė yra statesnė pradžioje, tuo geresnis yra algoritmas, nes jis geba atpažinti *impostor* paveikslėlius netgi kai panašumas tarp jų yra didelis bei artimas *genuine* paveikslėlių atstumui. Lyginant ROC kreivių EER (10 lentelė) akivaizdu, jog algoritmas su *threshold* = 0,5 veikia geriau už kitus.

Algoritmo parametras *threshold*	**EER reikšmė**
0,3	0,1051
0,5	0,0793
0,8	0,1125

10 lentelė. Paveikslėlių palyginimo algoritmų su *distRatio* = 1 ir skirtingais *threshold* parametrais EER reikšmės.

Toliau patikrinsime *distRatio* parametro įtaką algoritmo rezultatams. ROC kreivių grafikai, naudojant *distRatio*=0,6 ir tas pačias *threshold* reikšmes parodyti 60 paveiksle. Šių kreivių EER reikšmės parodytos 11 lentelėje.



60 paveikslas. ROC kreivė, naudojant *distRatio*=0,6 ir skirtingus *threshold* (*threshold*=0,3 raudona linija; *threshold* = 0,5 mėlyna linija; *threshold* = 0,8 rožinė linija).

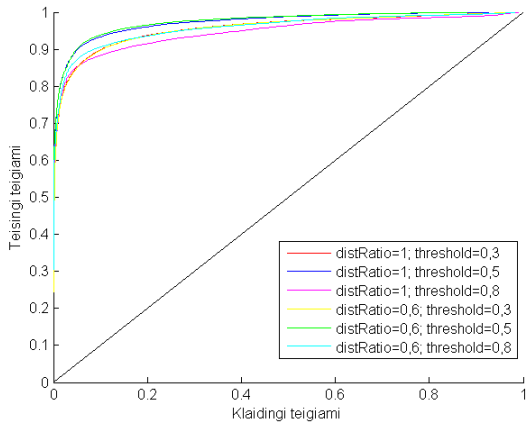
Algoritmo parametras *threshold*	**EER reikšmė**
0,3	0,1063
0,5	0,0766
0,8	0,0969

11 lentelė. Paveikslėlių palyginimo su parametru *distRatio* = 0,6 ir skirtingais *threshold* parametrais, EER reikšmės.

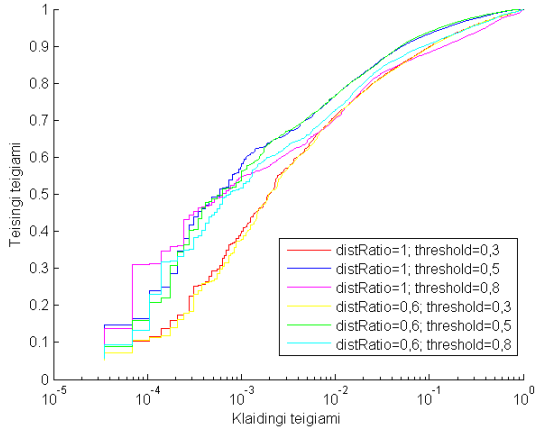
60 paveiksle pavaizduotos trys ROC kreivės su tuo pačiu *distRatio* parametru ir skirtingais *threshold* parametrais. Matome, kad mėlyna kreivė (*threshold*=0,5) kairėje pusėje labiausiai pakilusi, taigi, šie algoritmo parametrai labiausiai tinkami atskirti vienodus ir skirtingus objektus. EER rezultatai taip pat rodo, kad *threshold* = 0,5 yra tinkamiausiais parametras iš šių trijų.

Siekiant nustatyti, kuri *distRatio* parametro reikšmė yra geresnė, buvo nubraižytas dar vienas grafikas vaizduojantis 6 ROC kreives – atstumo tarp paveikslėlių skaičiavimo algoritmas su

parametrais *distRatio* = 0,6 ir 1,0 bei *threshold* = 0,3; 0,5; 0,8. Šis grafikas tiesinėje skalėje pavaizduotas 61 paveiksle, o pusiau logaritminėje skalėje – 62paveiksle.



61 paveikslas. ROC kreivė, naudojant skirtingus algoritmo parametrus: *distRatio*=1 ir *threshold* = 0,3 – raudona linija; *distRatio* = 1 ir *threshold* = 0,5 – mėlyna linija; *distRatio* = 1 ir *threshold* = 0,8 – rožinė linija; *distRatio* = 0,6 ir *threshold* = 0,3 – geltona linija; *distRatio* = 0,6 ir *threshold* = 0,5 – žalia linija; *distRatio* = 0,6 ir *threshold* = 0,8 – žydra linija.



62 paveikslas. ROC kreivė, naudojant skirtingus algoritmo parametrus: *distRatio*=1 ir *threshold* = 0,3 – raudona linija; *distRatio* = 1 ir *threshold* = 0,5 – mėlyna linija; *distRatio* = 1 ir *threshold* = 0,8 – rožinė linija; *distRatio* = 0,6 ir *threshold* = 0,3 – geltona linija; *distRatio* = 0,6 ir *threshold* = 0,5 – žalia linija; *distRatio* = 0,6 ir *threshold* = 0,8 – žydra linija. Ox ašis logaritminėje skalėje.

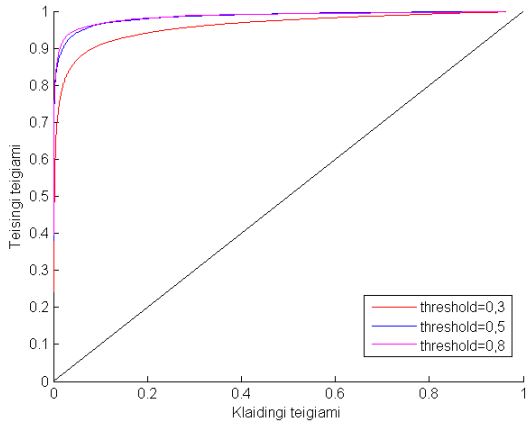
61 paveiksle matyti, kad parametro *threshold*=0,5 reikšmė duoda geriausius atpažinimo rezultatus. Tas pats grafikas su logaritmine Ox ašimi parodo, kad geriausi algoritmo parametrai yra *distRatio* = 1 ir *threshold* = 0,8 (rožinė linija). EER įverčiai patvirtina parametro *threshold* reikšmę 0,5, tačiau vos geresnis EER rezultatas buvo gautas su *distRatio* = 0,6 parametru.

Paveikslėlių palyginimo algoritmas buvo patobulintas – įvesta simetrija skaičiuojant ypatingų taškų poras tarp paveikslėlių. Skyrelyje 3.2.1. buvo pažymėta, kad klasikinis SIFT algoritmas yra asimetriškas ypatingų taškų skaičiavimo atžvilgiu – ypatingųjų taškų pora (a_n, b_m) ieškant paveikslėlio A taškui poros, gali nesutapti su pora, rasta ieškant iš B paveikslėlio pusės:

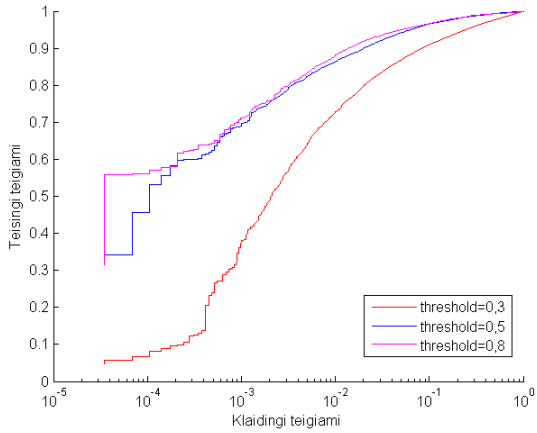
asimetrinė pora: $\min(d(a_n, b_m)) \neq \min(d(b_m, a_n))$;

simetrinė pora: $\min(d(a_n, b_m)) = \min(d(b_m, a_n))$.

Simetriniu algoritmu buvo tikrinti parametrai *distRatio* ir *threshold*, siekiant palyginti rezultatus su asimetriniu algoritmo variantu. 63 paveiksle pavaizduota ROC kreivė imant *distRatio* = 1 ir skirtingas *threshold* reikšmes (0,3; 0,5; 0,8); 64 paveiksle pavaizduota ta pati kreivė pusiau logaritminėje skalėje. EER reikšmės parodytos 12 lentelėje.



63 paveikslas. ROC kreivė, naudojant simetrinį ypatingųjų taškų porų radimą bei parametrus *distRatio*=1 ir skirtingus *threshold* (*threshold*=0,3 raudona linija; *threshold* = 0,5 mėlyna linija; *threshold* = 0,8 rožinė linija).



64 paveikslas. ROC kreivė, naudojant simetrinį ypatingųjų taškų porų radimą bei parametrus *distRatio*=1 ir skirtingus *threshold* (*threshold*=0,3 raudona linija; *threshold* = 0,5 mėlyna linija; *threshold* = 0,8 rožinė linija). Ox ašis logaritminėje skalėje.

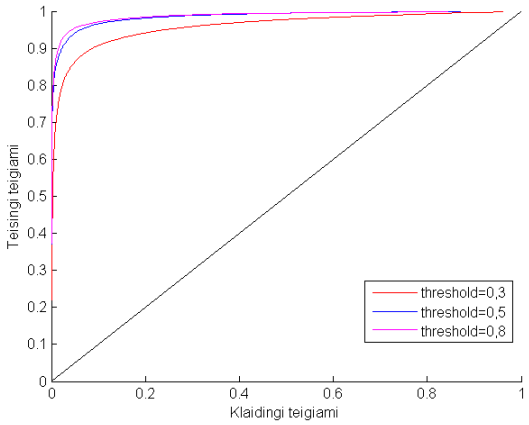
Algoritmo parametras *threshold*	EER reikšmė
0,3	0,0980
0,5	0,0563
0,8	0,0513

12 lentelė. Paveikslėlių palyginimo algoritmo, naudojančio simetrinį SIFT taškų porų radimą, su parametru *distRatio* = 1 ir skirtingais *threshold* parametrais, EER reikšmės.

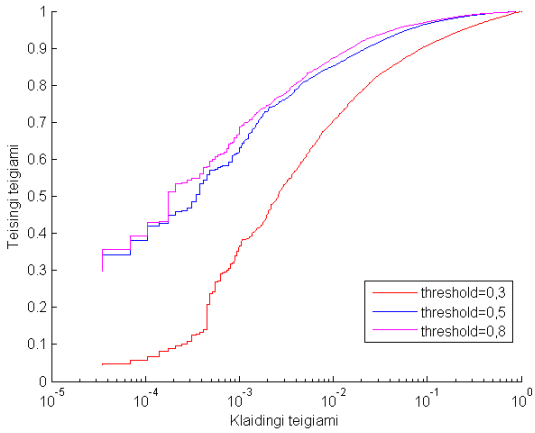
Vizualiai analizuojant 58 ir 63 paveikslų grafikus pastebime, kad simetrinis porų radimo būdas veikia kur kas geriau: esant *threshold* = 0,3 rezultatai atrodo panašūs, su *threshold* = 0,5 simetrinis porų radimo algoritmas duoda kiek geresnius rezultatus (ROC kreivė yra statesnė), o su *threshold* = 0,8 simetrinis porų radimas smarkiai lenkia asimetrinį būdą. Simetrinis porų radimo algoritmas tiksliausiai veikia su parametrų rinkiniu *distRatio* = 1 ir *threshold* = 0,8 (kaip matome iš 62 paveikslo – rožinė kreivė yra stačiausia, eina aukščiau kitų). Asimetrinio porų radimo algoritmo

geriausi parametrai yra tie patys (kaip matyti iš 59 paveikslo), tačiau rožinė kreivė kertasi su mėlyna kreive ir net atsiduria žemiau jos, o tai rodo, kad su *threshold* = 0,5 parametru gaunamas geresnis atpažinimas esant mažesnėms panašumo (angl. *similarity*) reikšmėms. EER rodikliai pavirtina vizualiai gautas išvadas – simetrinis porų radimas smarkiai įtakojo algoritmo su parametru *threshold* = 0,8 veikimą, nežymiai pagerino rezultatus su parametru *threshold* = 0,3 bei kiek ženkliau pagerino algoritmo su parametru *threshold* = 0,5 atpažinimą. Geriausias *threshold* parametras yra 0,8, nagrinėjant algoritmą su simetriniu SIFT taškų porų radimu ir *distRatio*=1 parametru.

Toliau buvo analizuoti kitas algoritmo parametrų rinkinys: *distRatio* = 0,6 ir *threshold* = 0,3; 0,5; 0,8. Gautieji ROC grafikai parodyti 65 ir 66 paveiksluose, o EER reikšmės - 13 lentelėje.



65 paveikslas. ROC kreivė, naudojant simetrinį ypatingųjų taškų porų radimą bei parametrus *distRatio*=0,6 ir skirtingus *threshold* (*threshold*=0,3 raudona linija; *threshold* = 0,5 mėlyna linija; *threshold* = 0,8 rožinė linija).



66 paveikslas. ROC kreivė, naudojant simetrinį ypatingųjų taškų porų radimą bei parametrus *distRatio*=0,6 ir skirtingus *threshold* (*threshold*=0,3 raudona linija; *threshold* = 0,5 mėlyna linija; *threshold* = 0,8 rožinė linija). Ox ašis logaritminėje skalėje.

Algoritmo parametras *threshold*	EER reikšmė
0,3	0,0994
0,5	0,0572
0,8	0,0488

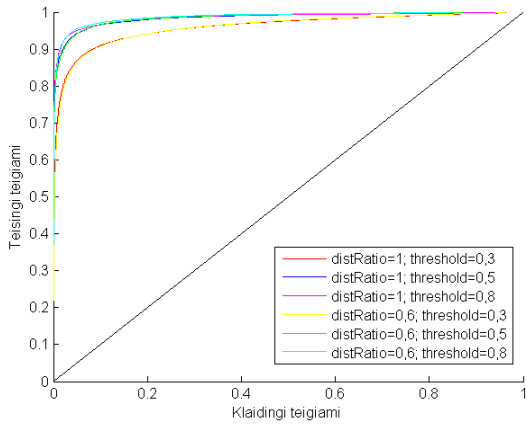
13 lentelė. Paveikslėlių palyginimo algoritmo su simetriniu SIFT porų radimu bei parametru *distRatio* = 0,6 ir skirtingais *threshold* parametrais, EER reikšmės.

63 bei 65 paveiksluose esančios ROC kreivės atrodo labai panašios, t.y. algoritmo rezultatai beveik nepriklauso nuo parametro *distRatio* reikšmės. Tačiau pažiūrėjus į atitinkamus pusiau

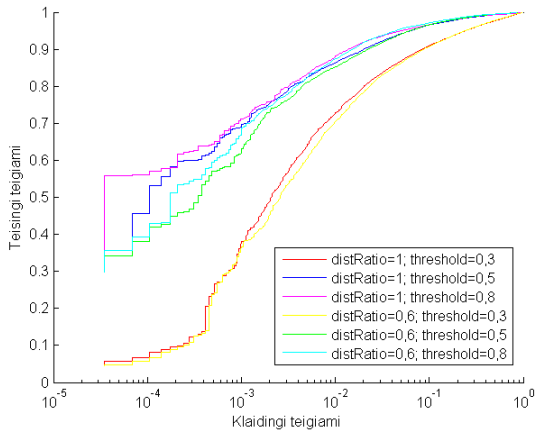
logaritminius grafikus (64 bei 66 paveikslai) pastebime, kad algoritmas geriau atpažįsta objektus naudojant parametrus *distRatio* = 1 ir *threshold* = 0,8 negu kitais atvejais. Tą patį pastebime analizuodami EER reikšmes.

Lyginant asimetrinį ir simetrinį ypatingųjų taškų porų radimo būdą pastebime, kad algoritmo rezultatai priklauso nuo parametro threshold pasirinkimo: kai *threshold* = 0,3, asimetrinis porų radimo algoritmas veikia kiek geriau negu simetrinis, tuo tarpu kai *threshold* = 0,5 abu porų radimo būdai duoda panašius atpažinimo rezultatus, o kai *threshold* = 0,8, simetrinis porų radimo būdas smarkiai pralenkia asimetrinį.

Siekiant įsitikinti, kuris parametrų rinkinys veikia geriausiai naudojant simetrinį ypatingųjų taškų radimo būdą, buvo nubraižytos ROC kreivės su visais 6 parametrų rinkiniais. Grafikas tiesinėje skalėje pavaizduotas 67 paveiksle, o pusiau logaritminėje skalėje – 68 paveiksle.



67 paveikslas. ROC kreivė, naudojant skirtingus algoritmo parametrus: *distRatio*=1 ir *threshold* = 0,3 – raudona linija; *distRatio* = 1 ir *threshold* = 0,5 – mėlyna linija; *distRatio* = 1 ir *threshold* = 0,8 – rožinė linija; *distRatio* = 0,6 ir *threshold* = 0,3 – geltona linija; *distRatio* = 0,6 ir *threshold* = 0,5 – žalia linija; *distRatio* = 0,6 ir *threshold* = 0,8 – žydra linija.



68 paveikslas. ROC kreivė, naudojant skirtingus algoritmo parametrus: *distRatio*=1 ir *threshold* = 0,3 – raudona linija; *distRatio* = 1 ir *threshold* = 0,5 – mėlyna linija; *distRatio* = 1 ir *threshold* = 0,8 – rožinė linija; *distRatio* = 0,6 ir *threshold* = 0,3 – geltona linija; *distRatio* = 0,6 ir *threshold* = 0,5 – žalia linija; *distRatio* = 0,6 ir *threshold* = 0,8 – žydra linija. Ox ašis logaritminėje skalėje.

ROC kreivės su visais 6 parametrų rinkiniais rodo, kad geriausi atpažinimo rezultatai gaunami naudojant *distRatio* = 1 ir *threshold* = 0,8. Blogiausi atpažinimo rezultatai gaunami naudojant *threshold* = 0,3 parametro reikšmę.

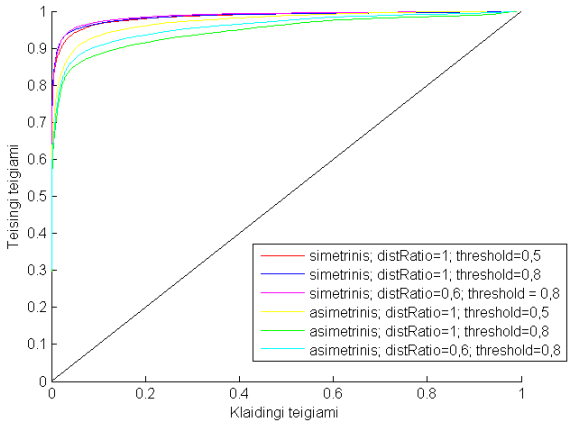
Lyginant simetrinį ir asimetrinį ypatingųjų taškų radimo būdą, matyti, jog abiem atvejais *threshold* = 0,3 pablogina atpažinimo rezultatus lyginant su kitomis parametro *threshold*

reikšmėmis (0,5 bei 0,8). Taip pat abiem atvejais geriausi atpažinimo rezultatai pasiekiami su parametrų rinkiniu *distRatio* = 1 ir *threshold* = 0,8.

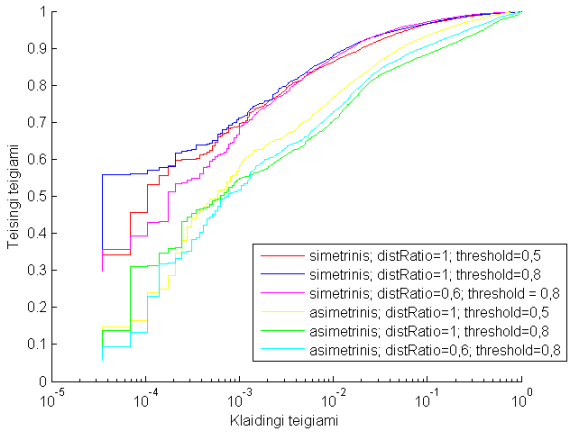
Siekiant nustatyti, koks panašumo tarp paveikslėlių skaičiavimo algoritmo variantas yra tinkamiausias, buvo pasirinkti keli geriausi parametrų rinkiniai iš simetrinio ir asimetrinio ypatingųjų taškų porų ieškojimo rezultatų. Taigi, buvo lygintos tokios ROC kreivės:

- simetrinis porų radimo būdas; parametrai *distRatio* = 1 ir *threshold* = 0,5;
- simetrinis porų radimo būdas; parametrai *distRatio* = 1 ir *threshold* = 0,8;
- simetrinis porų radimo būdas; parametrai *distRatio* = 0,6 ir *threshold* = 0,8;
- asimetrinis porų radimo būdas; parametrai *distRatio* = 1 ir *threshold* = 0,5;
- asimetrinis porų radimo būdas; parametrai *distRatio* = 1 ir *threshold* = 0,8;
- asimetrinis porų radimo būdas; parametrai *distRatio* = 0,6 ir *threshold* = 0,8;

Bendras grafikas pavaizduotas 69 (tiesinėje skalėje) ir 70 (pusiau logaritminėje skalėje) paveiksluose.



69 paveikslas. ROC kreivė, naudojant skirtingus algoritmo parametrus: simetrinis ypatingųjų taškų porų radimo metodas bei parametrai *distRatio*=1 ir *threshold* = 0,5 – raudona linija; simetrinis porų radimo metodas bei parametrai *distRatio* = 1 ir *threshold* = 0,8 – mėlyna linija; simetrinis porų radimo metodas bei parametrai *distRatio* = 0,6 ir *threshold* = 0,8 – rožinė linija; asimetrinis porų radimo metodas bei parametrai *distRatio* = 1 ir *threshold* = 0,5 – geltona linija; asimetrinis porų radimo metodas bei parametrai *distRatio* = 1 ir *threshold* = 0,8 – žalia linija; asimetrinis porų radimo metodas bei parametrai *distRatio* = 0,6 ir *threshold* = 0,8 – žydra linija.



70 paveikslas. ROC kreivė, naudojant skirtingus algoritmo parametrus: simetrinis ypatingųjų taškų porų radimo metodas bei parametrai *distRatio*=1 ir *threshold* = 0,5 – raudona linija; simetrinis porų radimo metodas bei parametrai *distRatio* = 1 ir *threshold* = 0,8 – mėlyna linija; simetrinis porų radimo metodas bei parametrai *distRatio* = 0,6 ir *threshold* = 0,8 – rožinė linija; asimetrinis porų radimo metodas bei parametrai *distRatio* = 1 ir *threshold* = 0,5 – geltona linija; asimetrinis porų radimo metodas bei parametrai *distRatio* = 1 ir *threshold* = 0,8 – žalia linija; asimetrinis porų radimo metodas bei parametrai *distRatio* = 0,6 ir *threshold* = 0,8 – žydra linija. Ox ašis logaritminėje skalėje.

69 paveiksle matyti, kad simetrinis porų radimo metodas duoda patikimesnius atpažinimo rezultatus. 70 paveiksle parodytos tos pačios ROC kreivės pusiau logaritminėje skalėje; ji nurodo, kad simetrinis porų radimo metodas kartu su parametrais *distRatio* = 1 bei *threshold* =0,8 duoda geriausius rezultatus.

Geriausi parametrai atstumo algoritmui būtų *distRatio*=1 ir *threshold*=0,8, nes juos taikant gautoji ROC kreivė buvo stačiausia. EER rodiklis taip pat patvirtina parametrų tinkamumą. Šie parametrai ir simetrinis porų radimo būdas bus naudojami klasterizavimo metu.

3.3. Klasterizavimas ESOM neuroniniu tinklu

Klasterizavimui pasirinktas algoritmas, kuris buvo plačiau aptartas [Mar06a] darbe.

Klasterizavimą sudaro 3 etapai:

- Neuroninio tinklo mokymas;
- U-Matrix sudarymas;
- Duomenų suskirstymas į klasterius remiantis U-Martix vaizduojamomis struktūromis.

Pirmajame etape yra naudojamas ESOM neuroninis tinklas, turintis plokščią gardelę. Ši suskirstyta į langelius taip, kad kiekvienas neuronas turi 4 tiesioginius kaimynus (išskyrus gardelės kraštuose esančius neuronus). Tinklo atstumo funkcija pakeista taip, kad atitiktų paveikslėlių palyginimo algoritmą, aptartą 3.2.3. skyriuje:

1. Kiekvienai konkrečiai duomenų aibei iš anksto suskaičiuojami panašumai tarp paveikslėlių ir išsaugomi matricoje. Paveikslėlių panašumų matricą vadinsime *D*.
2. Įkeliant duomenų aibę į neuroninio tinklo mokymosi algoritmą, kartu yra pakeičiama tinklo mokymosi funkcija taip, kad ji naudotų jau suskaičiuotus atstumus tarp paveikslėlių.
3. Duomenų aibė yra sudaryta iš vektorių, kurių nenuliniai indeksai nurodo, kurį paveikslėlį atitinka šis vektorius.
4. Panašumas tarp dviejų duomenų vektorių $input_1$=(0; 0; 1; 0) ir $input_2$= (0; 1; 0; 0) yra lygus panašumui tarp 3-ojo ir 2-ojo paveikslėlio, arba panašumų matricos *D(3,2)* celės reikšmei.
5. Atstumo funkcija, remdamasi jau žinomais panašumais tarp konkrečių paveikslėlių, gali interpoliuoti ir nurodyti panašumus tarp tinklo neuronų. Pavyzdžiui, du neuronai neu_1 = (0,1; 0,9; 0; 0) ir neu_2=(0; 0,5; 0; 0,5). Panašumas tarp jų bus skaičiuojamas

$$sim = neu_1 * D * neu_2 = \begin{pmatrix}0{,}1 & 0{,}9 & 0 & 0\end{pmatrix} * \begin{pmatrix} 1 & 0{,}3 & 0{,}4 & 0{,}7 \\ 0{,}3 & 1 & 0{,}65 & 0{,}43 \\ 0{,}4 & 0{,}65 & 1 & 0{,}2 \\ 0{,}7 & 0{,}43 & 0{,}2 & 1 \end{pmatrix} * \begin{pmatrix}0 & 0{,}5 & 0 & 0{,}5\end{pmatrix},$$

   t.y. vektorių koordinačių tiesinė kombinacija.

Antrajame etape tinklas yra vizualizuojamas U-Matrix pagalba. Sudaroma tinklo gardelės dydžio matrica, į kurios laukelius įrašomi vidutiniai atstumai tarp atitinkamo tinklo neurono ir jo kaimynų.

Matrica yra vizualizuojama reljefiniu paviršiumi – mėlyna („ežeras“) ir žalia („slėnis“) spalvos nurodo, kad gretimų neuronų svoriai yra panašūs, ruda spalva („aukštumos“, „kalnai“) – kad tolimesni, galiausiai balta spalva („sniegynas“) parodo, kad gretimų neuronų svoriai yra labai skirtingi.

Trečiajame klasterizavimo etape naudojamas užliejimo vandeniu, vandenskyros algoritmas, pasiūlytas [OM04] ir išbandytas [Mar06a] darbe. Užliejimo vandeniu algoritmas suklasterizuoja ESOM tinklo neuronus – U-Matrix vaizduojamus slėnius sujungia, atskirdamas kalnus ir sniegynus. Po to nustatoma, kokių įvesties duomenų BMU neuronai buvo priskirti tam pačiam klasteriui ir tokiu būdu nurodoma įvesties duomenų klasterizavimo struktūra.

Standartiniai ESOM tinklo mokymo parametrai išrašyti 14 lentelėje.

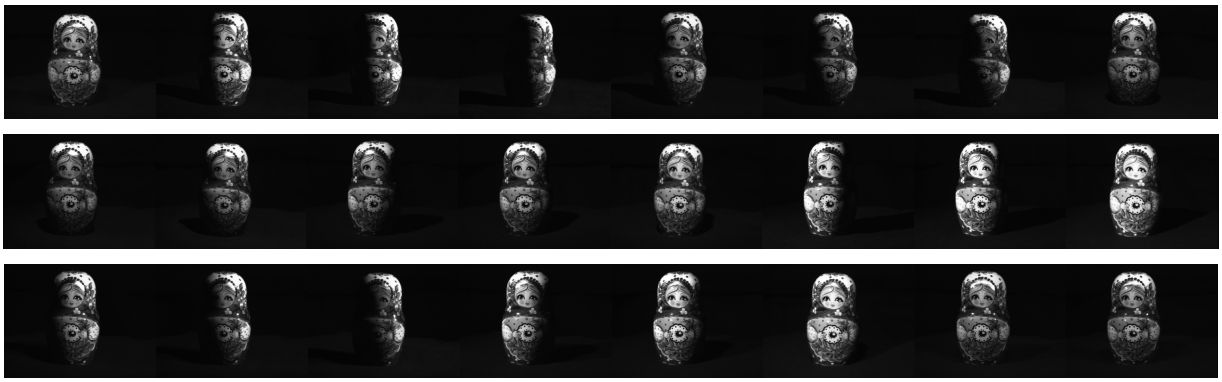
Pradinis mokymosi greitis (alpha)	0,8
Mokymosi greitis (alpha) paskutinėje epochoje	0,1
Pradinis kaimynystės plotis (sigma)	25
Kaimynystės plotis (sigma) paskutinėje epochoje	0
Mokymosi greičio kitimo funkcija	Tiesinė
Kaimynystės pločio kitimo funkcija	Tiesinė
Mokymosi epochų skaičius	20
Tinklo dydis (aukštis x plotis)	50x50

14 lentelė. ESOM tinklo mokymosi parametrai.

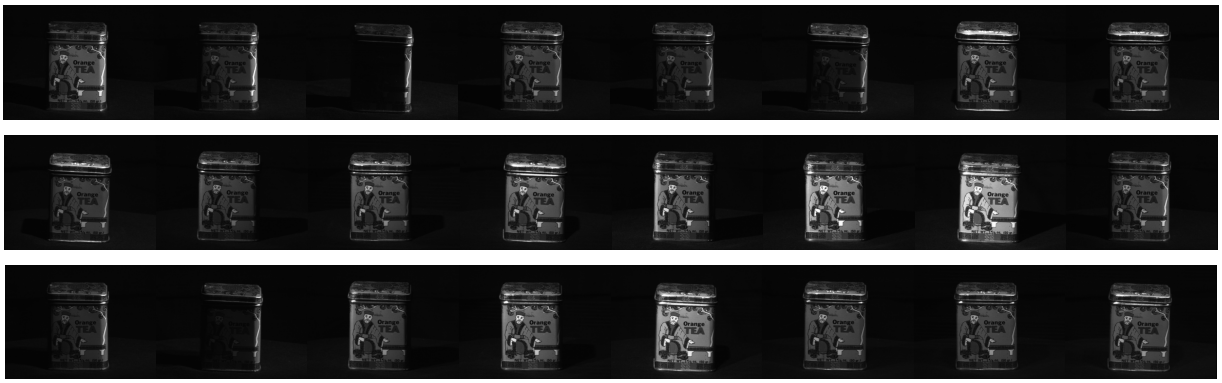
Praktiniai bandymai bus atlikti naudojant aprašytus ESOM tinklo parametrus, taip pat bus ieškoma kitų tinklo parametrų, kurie pagerintų paveikslėlių klasterizavimo algoritmą, padėtų lengviau atskirti į klasterius skirtingus objektus.

3.3.1. Dviejų objektų klasterizavimas

Pradiniam ESOM tinklo testavimui pasirinkti du objektai iš ALOI paveikslėlių duomenų bazės, naudoti 3.2. skyriuje. Šie objektai – matrioška ir arbatos dėžutė – pavaizduoti 71 bei 72 paveiksluose.

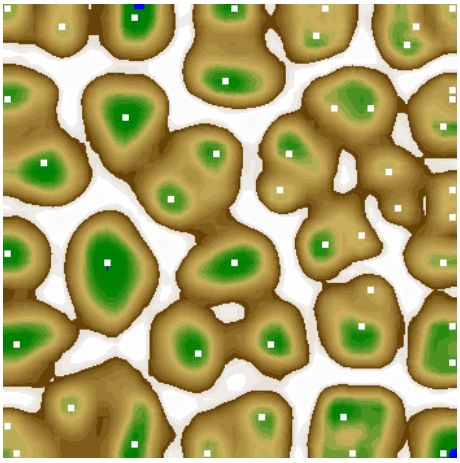


71 paveikslas. Pirmojo objekto – matrioškos – paveikslėliai.

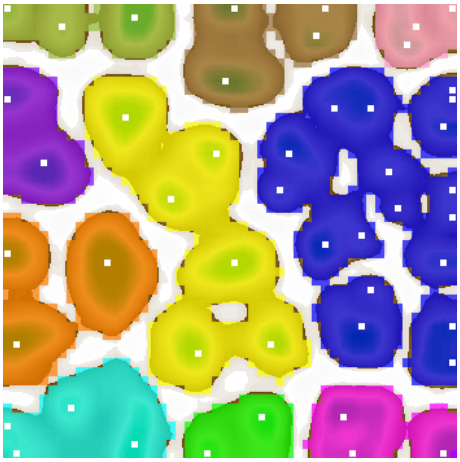


72 paveikslas. Antrojo objekto – arbatos dėžutės – paveikslėliai.

Iš pradžių buvo pasirinkti standartiniai ESOM tinklo parametrai – Online tinklo mokymosi būdas, tinklo gardelės dydis 50x50 neuronų, 20 mokymosi epochų. Iš ESOM tinklo neuronų svorių buvo suskaičiuota U-Matrix, pavaizduota 73 paveiksle. U-Matrix žalia spalva nurodo, kad gretimų neuronų svoriai yra panašūs, ruda – kad svoriai skiriasi, o balta spalva – gretimi neuronai yra visiškai skirtingi. Balti kvadratiniai taškeliai žymi BMU neuronus, t.y. tokius neuronus, kuriems vienas iš įvesties paveikslėlių yra pats artimiausias. Klasterizavimas vandenskyros algoritmu su 0,7 atskyrimo slenksčiu parodytas 74paveiksle. Vandenskyros algoritmo slenkstis nurodo, iki kokio aukščio gali pakilti vanduo, arba koks galimas maksimalus atstumas tarp gretimų ESOM tinklo neuronų, kad abu neuronai priklausytų tam pačiam klasteriui.



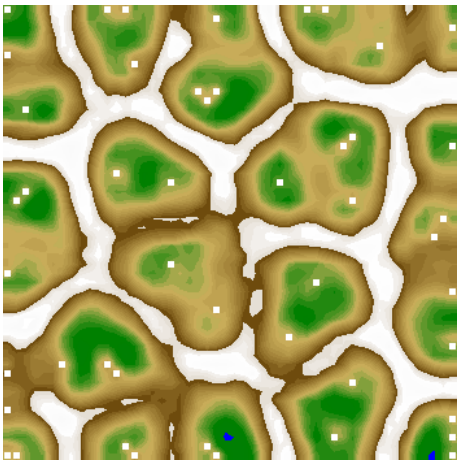
73 paveikslas. ESOM neuroninio tinklo U-Matrix. Klasterizavimui naudoti 2 objektų paveikslėliai, tinklo dydis 50x50, mokymasis truko 20 epochų.



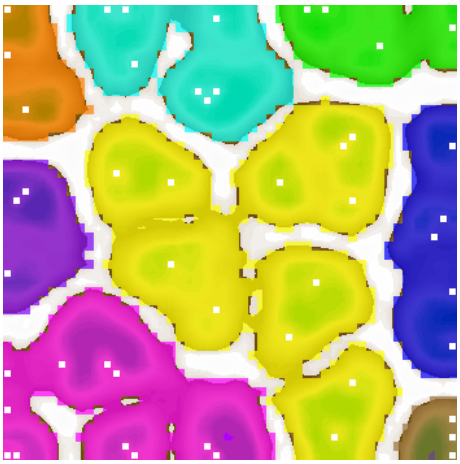
74 paveikslas. U-Matrix klasterizavimas naudojant vandenskyros algoritmą su atskyrimo slenksčiu 0,7.

Neuronų klasterizavime matome daug klasterių su trim ir daugiau BMU neuronų. Matriošką (pirmąjį objektą) atitinka šie klasteriai: geltonas, ryškiai žalias, žydras, oranžinis, violetinis, pastelinis žalias bei rudas klasteriai. Antrąjį objektą – arbatos dėžutę – atitinka 3 klasteriai: rožinis, mėlynas ir ryškiai rožinis.

Neuronų klasterių gausa gali būti įtakota tinklo persimokymo – tokios situacijos, kuomet tinklo mokymosi epochų yra per daug, todėl galiausiai kiekvienas įvesties duomuo turi savo „klasterį“ – neuronų grupę, smarkiai nutolusią nuo kitų neuronų. Siekiant išvengti tinklo persimokymo, epochų skaičius buvo sumažintas iki 10. Gauti rezultatai pavaizduoti 75 (tinklo U-Matrix) bei 76 paveiksluose (neuronų klasterizavimas, naudojant vandenskyros atskyrimo slenkstį 0,7).



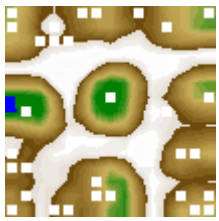
75 paveikslas. ESOM neuroninio tinklo U-Matrix. Klasterizavimui naudoti 2 objektų paveikslėliai, tinklo dydis 50x50, mokymasis truko 10 epochų.



76 paveikslas. U-Matrix klasterizavimas, naudojant vandenskyros algoritmą su nupjovimo parametru 0,7.

Matome, kad esant mažiau ESOM tinklo mokymo epochų, gauname mažesnį klasterių skaičių. 76 paveiksle žalias, mėlynas, geltonas ir rudas klasteriai vaizduoja matriošką, o rožinis, violetinis, oranžinis ir žydras – arbatos dėžutę.

ESOM tinklas taip pat buvo testuotas su mažesne gardele 15x15 neuronų. Gautoji U-Matrix bei jos klasterizavimas pavaizduoti 77 ir 78 paveiksluose.



77 paveikslas. ESOM neuroninio tinklo U-Matrix. Klasterizavimui naudoti 2 objektų paveikslėliai, tinklo dydis 15x15, mokymasis truko 20 epochų.

78 paveikslas. U-Matrix klasterizavimas, naudojant vandenskyros algoritmą su 0,7 nupjovimo reikšme.

Dabar ESOM tinklas įvesties duomenis suskirstė į 5 klasterius, iš kurių žalias ir žydras vaizduoja matriošką, likusieji – arbatos dėžutę. Galime pastebėti, kad sumažinus tinklo gardelės dydį, gavome mažiau klasterių, puikiai atskiriančių dviejų objektų paveikslėlius.

3.3.2. Vizualiai panašių bei skirtingų objektų klasterizavimas

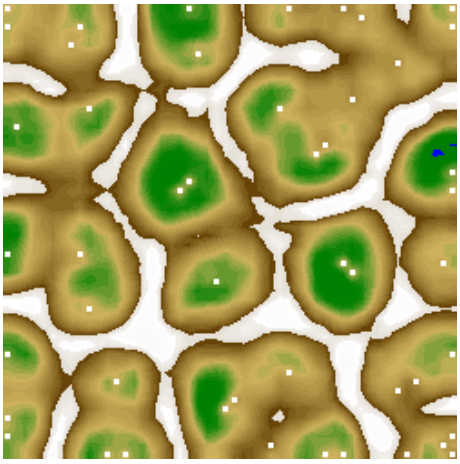
Šiame skyriuje bus aprašyti 3 bandymai. Iš ALOI duomenų bazės pasirinkti 3 objektai – Dilmah arbatos dėžutė, kūdikių maisto dėžutė bei puodelis. Pirmieji du objektai (arbatos dėžutė ir kūdikių maisto dėžutė) vizualiai yra panašūs, taigi, klasterizavimo algoritmas turėtų parodyti, kad sunkiai geba šių objektų paveikslėlius atskirti į skirtingus klasterius. Pirmasis bandymas – klasterizuoti arbatos dėžutės ir kūdikių maisto dėžutės paveikslėlius. Antrasis bandymas – klasterizuoti Dilmah arbatos dėžutės bei puodelio paveikslėlius. Šie objektai vizualiai yra skirtingi, todėl galime tikėtis, jog klasterizavimo algoritmas lengvai juos išskirs į atskirus klasterius. Trečiasis bandymas – klasterizuoti visų trijų objektų paveikslėlius. ESOM tinklas turėtų sumaišyti arbatos dėžutės ir kūdikių maisto paveikslėlius bei atskirti puodelio paveikslėlius.

Pirmajam bandymui iš ALOI paveikslėlių duomenų bazės buvo pasirinkti 2 panašūs objektai – Dilmah arbatos dėžutė (79 paveikslas) ir kūdikių maisto dėžutė (80 paveikslas).

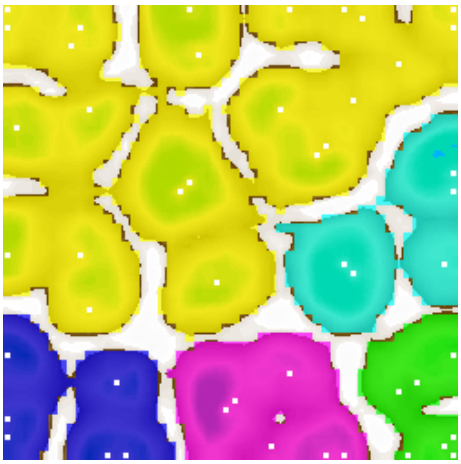
79 paveikslas. Dilmah arbatos pakelis.

80 paveikslas. Bambix kūdikių maisto pakelis.

Objektai buvo klasterizuoti ESOM tinklu naudojant 50x50 tinklo dydį, 10 mokymosi epochų. Gautoji U-Matrix ir jos klasterizavimas, imant 0,7 užliejimo vandeniu slenkstį, pavaizduoti 81 ir 82 paveiksluose.

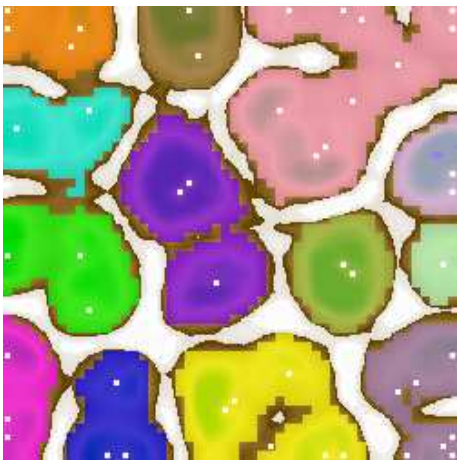


81 paveikslas. ESOM tinklo U-Matrix. Tinklo gardelės dydis 50x50 neuronų, mokymasis truko 10 epochų. Mokymosi duomenys – Dilmah arbatos pakelio ir kūdikių maisto pakelio paveikslėliai.



82 paveikslas. Neuronų klasterizavimas, naudojant 0,7 užliejimo vandeniu slenkstį.

Klasterizavimo paveiksle matome, jog susidarė tik 4 klasteriai. Naudojant 0,7 užliejimo algoritmo parametrą, geltoname klasteryje pateko ir arbatos pakelio, ir kūdikių maisto pakelio paveikslėlių BMU. Matosi, kad neuroninis tinklas sunkiai atskiria objektus į skirtingus klasterius. Pabandysime sumažinti užliejimo parametrą ir tokiu būdu išskirti daugiau klasterių (83 paveikslas).

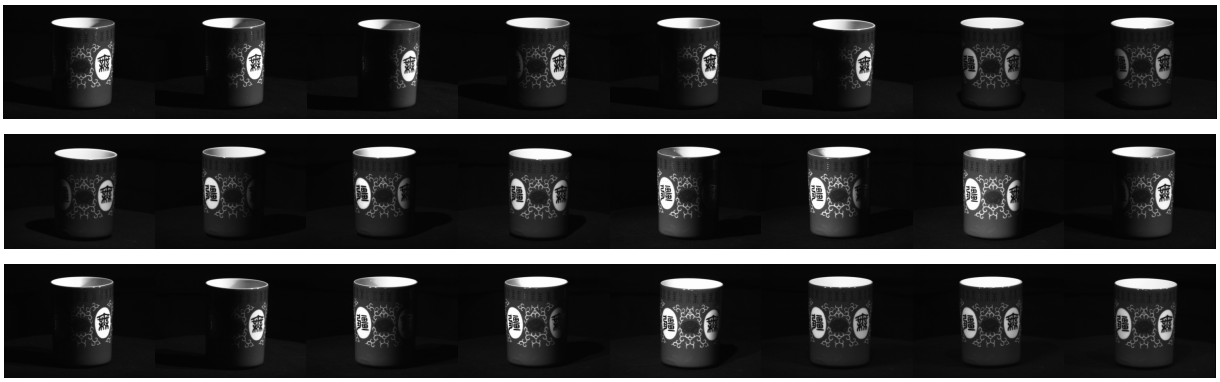


83 paveikslas. Neuronų klasterizavimas, naudojant 0,6 užliejimo vandeniu slenkstį.

Antruoju klasterizavimo būdu susidaro klasteriai, į kuriuos patenka tik 1-3 BMU neuronai, taigi, keli paveikslėliai turi savo atskirą klasterį. Toks klasterizavimo būdas yra nekorektiškas – jis vos ne kiekvienam paveikslėliui priskiria po klasterį.

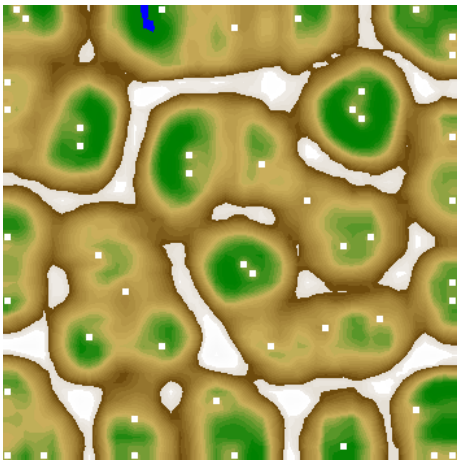
Galime daryti išvadą, kad klasterizavimo algoritmas sunkiai atskiria pasirinktų objektų – arbatos pakelio ir kūdikių maisto pakelio – paveikslėlius.

Antrajam bandymui iš ALOI paveikslėlių duomenų bazės pasirinkti 2 skirtingi objektai – tas pats Dilmah arbatos pakelis (79 paveikslas) ir puodelis (84 paveikslas).

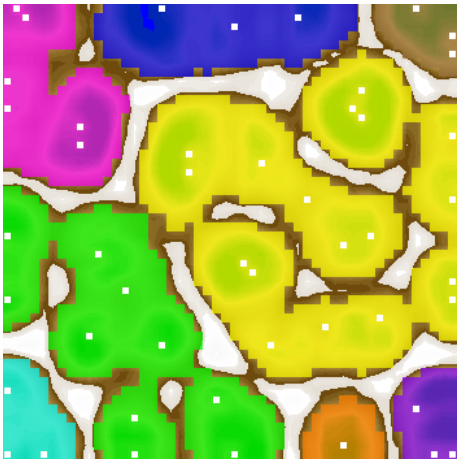


84 paveikslas. Puodelis.

Klasterizavimui pasirinkti tokie pat parametrai, kaip ir pirmuoju bandymu – tinklo gardelės dydis 50x50, ir 10 mokymosi epochų. Gautoji U-Matrix bei jos klasterizavimas naudojant užliejimo algoritmo parametrą 0,6, pavaizduoti atitinkamai 85 bei 86 paveiksluose.



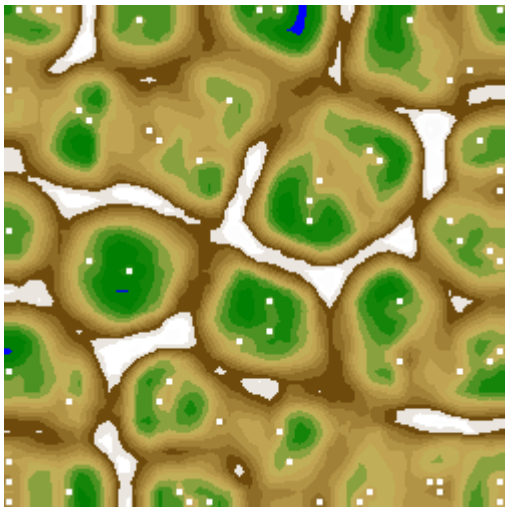
85 paveikslas. U-Matrix, gauta iš 50x50 gardelės dydžio ESOM tinklo, mokyto 10 epochų. Mokymosi aibė – Dilmah arbatos pakelio ir puodelio paveikslėliai.



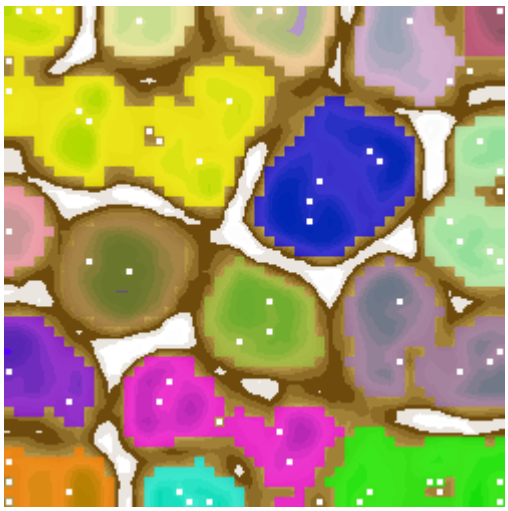
86 paveikslas. Neuronų klasterizavimas naudojant vandenskyros algoritmą su slenksčiu 0,6.

Neuronų klasterizavime 3 klasteriai – geltonas, rudas ir mėlynas – atitinka arbatos pakelio BMU neuronus, o likusieji klasteriai – rožinis, žalias, žydras, oranžinis bei violetinis – puodelio BMU neuronus. Matome, kad ESOM tinklas gerai atskiria šiuos du objektus.

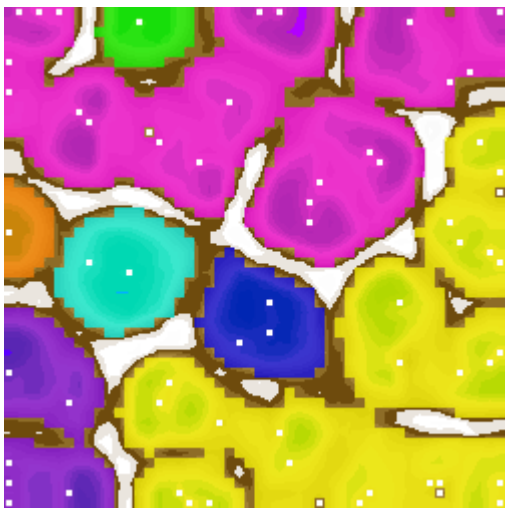
Trečiuoju bandymu buvo klasterizuoti visi 3 objektai – arbatos pakelis (79 paveikslas), kūdikių maisto pakelis (80 paveikslas) ir puodelis (84 paveikslas). Pasirinkti tokie pat ESOM tinklo mokymo parametrai, kaip ir ankstesniuose bandymuose – 50x50 tinklo gardelės dydis, 10 mokymosi epochų. Gautoji U-Matrix bei jos klasterizavimo būdai, naudojant 0,6 bei 0,5 vandenskyros slenksčius pavaizduoti atitinkamai 87, 88 bei 89 paveiksluose.



87 paveikslas. U-Matrix, gauta iš ESOM neuroninio tinklo su 50x50 neuronų gardele, mokyto 10 epochų. Mokymosi duomenys – Dilmah arbatos pakelio, kūdikių maisto pakelio ir puodelio paveikslėliai.



88 paveikslas. U-Matrix klasterizavimas, naudojant 0,6 atskyrimo slenkstį vandenskyros algoritme.

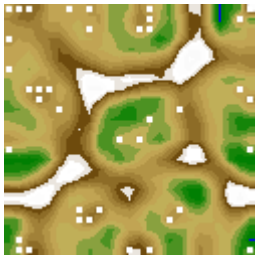


89 paveikslas. U-Matrix klasterizavimas, naudojant 0,5 atskyrimo slenkstį vandenskyros algoritme.

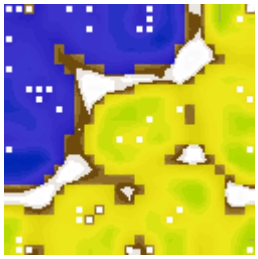
86 paveiksle matyti, kad su 0,6 atskyrimo parametru susidarė labai daug mažų klasterių. Sumažinus vandenskyros algoritmo parametrą iki 0,5 susidarė mažiau klasterių, tačiau skirtingų objektų BMU neuronai pateko į tuos pačius klasterius – arbatos pakelio ir kūdikių maisto

paveikslėlių BMU neuronai pateko į geltonąjį klasterį, o kūdikių maisto ir puodelio paveikslėlių BMU neuronai pateko į rožinį klasterį.

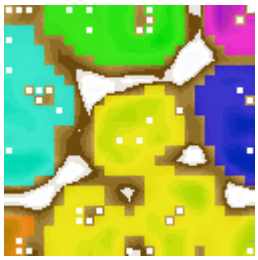
ESOM gardelės dydis buvo sumažintas siekiant minimizuoti klasterių skaičių. Tinklo mokymui naudota 25x25 dydžio gardelė, tinklas mokytas 10 epochų. Gautoji U-Matrix pavaizduota 90 paveiksle, o jos klasterizavimas naudojant 0,6 atskyrimo slenkstį - 91 paveiksle, naudojant slenkstį 0,5 - 92 paveiksle.



90 paveikslas. ESOM tinklo U-Matrix. Tinklo gardelės dydis 25x25, mokymosi epochų 10. Mokymosi aibė – Dilmah arbatos pakelio, kūdikių maisto pakelio ir puodelio paveikslėliai.



91 paveikslas. Neuronų klasterizavimas, naudojant vandenskyros algoritmą su slenksčiu 0,6.



92 paveikslas. Neuronų klasterizavimas, naudojant vandenskyros algoritmą su slenksčiu 0,5.

Galime pastebėti, kad esant mažesnei tinklo gardelei susidarė homogeniškesni klasteriai. Naudojant vandenskyros algoritmą su parametru 0,6 susiformavo 2 klasteriai (91 paveikslas). Geltonajame klasteryje yra arbatos pakelio bei kūdikių maisto BMU neuronai, o mėlynajame klasteryje – puodelio BMU neuronai. Taigi, ESOM klasterizavimo algoritmas neatskiria arbatos dėžutės ir kūdikių maisto paveikslėlių, jie atrodo homogeniški.

Esant 0,5 slenksčiui (92 paveikslas) kai kurie BMU neuronai nepatenka į jokį klasterį, todėl toks slenkstis yra netinkamas įvesties paveikslėlių klasterizavimui.

Šiame skyriuje buvo parodyti skirtingi klasterizavimo rezultatai naudojant trijų objektų iš ALOI duomenų bazės paveikslėlius. Vizualiai panašūs objektai klasterizavimo algoritmui taip pat pasirodė panašūs, tuo tarpu vizualiai skirtingi objektai – skirtingi. Taigi, pasirinktas klasterizavimo algoritmas geba atskirti paveikslėlius į skirtingas grupes neprasčiau nei žmogus.

3.3.3. Paveikslėlių iš Interneto klasterizavimas

Šiame skyriuje bus aprašytas gyvenimiškų paveikslėlių klasterizavimas.

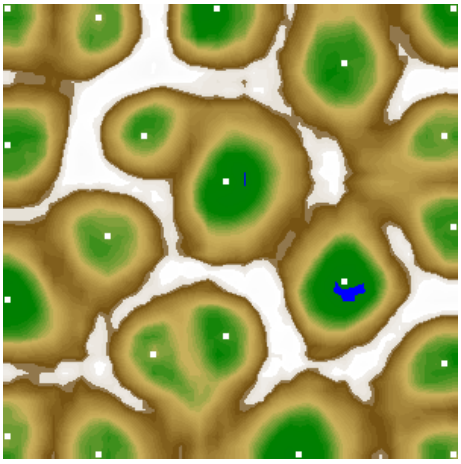
Interneto paieškos sistemoje http://images.google.lt surasti dviejų grupių paveikslėliai – miško (užklausa „forest“) bei miesto (užklausa „town“). Kiekvienai šių grupių surasta po 10 paveikslėlių. Jie pavaizduoti atitinkamai 93 bei 94 paveiksluose.

93 paveikslas. Miško paveikslėliai.

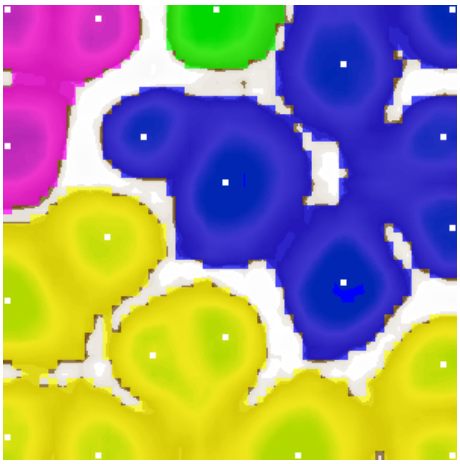
94 paveikslas. Miesto paveikslėliai.

Kiekviename iš paveikslėlių buvo išskirti SIFT požymiai, po to apskaičiuoti panašumai tarp paveikslėlių ir ši informacija naudojama klasterizavimui. Gyvenimiški paveikslėliai yra skirtingų dydžių, beveik visi jie didesni už testavimui naudotus paveikslėlius iš ALOI duomenų bazės. Kartu ir SIFT taškų skaičius paveikslėlyje yra didesnis – ALOI duomenų bazės vaizduose SIFT taškų yra apie 200 – 1000, tuo tarpu paveikslėliuose iš Interneto – apie 1000-10000 SIFT taškų.

Miško ir miesto paveikslėliai buvo klasterizuoti naudojant 50x50 dydžio neuroninį tinklą, kurio mokymas truko 10 epochų. Gautoji U-Matrix pavaizduota 95 paveiksle, o jos klasterizavimas, naudojant vandenskyros algoritmą su slenksčiu 0,7, 96 paveiksle.



95 paveikslas. U-Matrix, gauta iš ESOM tinklo, kurio gardelė 50x50 neuronų, mokyto 10 epochų. Mokymosi aibė – miško ir miesto paveikslėliai.



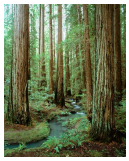
96 paveikslas. U-Matrix klasterizavimas, naudojant vandenskyros algoritmą su slenksčiu 0,7.

Klasterizavimo metu atsiskyrė 4 klasteriai – geltonas, mėlynas, rožinis bei žalias. Į šiuos klasterius pateko skirtingų kategorijų paveikslėliai: į geltonąjį, mėlyną ir rožinį pateko ir miesto, ir miško vaizdai, o į žalią klasterį (kuriame tėra vienas BMU neuronas) pateko miško paveikslėlis. Klasteriams priklausantys paveikslėliai pavaizduoti atitinkamai 97, 98, 99 ir 100 paveiksluose.

97 paveikslas. Geltonajam klasteriui priklausantys paveikslėliai.

98 paveikslas. Mėlynajam klasteriui priklausantys paveikslėliai.

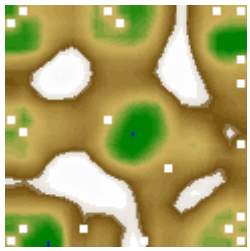
99 paveikslas. Rožiniam klasteriui priklausantys paveikslėliai.



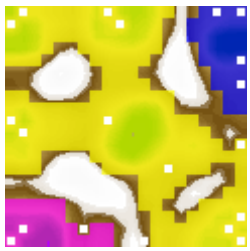
100 paveikslas. Žaliajam klasteriui priklausantys paveikslėliai.

Į tuos pačius klasterius patekę paveikslėliai vizualiai atrodo yra skirtingi.

Siekiant sumažinti klasterių skaičių ir homogenizuoti duomenis pačiuose klasteriuose buvo bandoma klasterizuoti naudojant mažesnę ESOM tinklo gardelę 25x25 neuronų, tinklą mokant 10 epochų. Gautoji U-Matrix pavaizduota 101 paveiksle, o jos klasterizavimas, naudojant 0,6 slenkstį vandenskyros algoritme, parodytas 102 paveiksle.



101 paveikslas. U-Matrix, gauta iš ESOM tinklo su 25x25 neuronų gardele, mokyto 10 epochų. Mokymo aibė – miško ir miesto paveikslėliai.



102 paveikslas. Neuronų klasterizavimas naudojant vandenskyros algoritmą su slenksčiu 0,6.

Klasterizuojant neuronus buvo atskirti 3 klasteriai – geltonas, mėlynas ir rožinis. Į pirmąjį ir paskutinįjį klasterį kartu pateko mažiau neuronų negu į vien geltonąjį klasterį. Tačiau apskritai klasterizavimas gavosi homogeniškesnis – rožiniame klasteryje yra vien tik miesto paveikslėliai, mėlynajame – vien tik miško, o geltonajame klasteryje paveikslėliai yra susimaišę. Paveikslėlių klasterizavimo rezultatai parodyti 103 (mėlynasis klasteris), 104 (geltonasis klasteris) ir 105 (rožinis klasteris) paveiksluose.

103 paveikslas. Mėlynajam klasteriui priklausantys paveikslėliai.

104 paveikslas. Geltonajam klasteriui priklausantys paveikslėliai.

105 paveikslas. Rožiniam klasteriui priklausantys paveikslėliai.

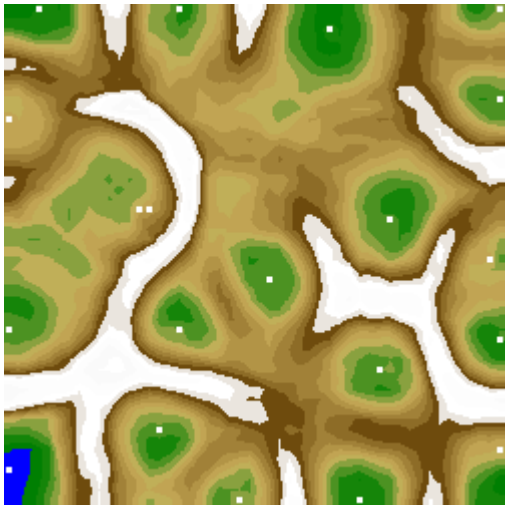
Realių vaizdų klasterizavimas pasirodė kiek blogesnis negu naudojant ALOI duomenų bazės paveikslėlius, tačiau šis klasterizavimas veikia kur kas geriau negu atsitiktinis skirstymas į grupes.

Nors miško ir miesto paveikslėliai vizualiai atrodo gan skirtingi, tačiau naudojant SIFT taškinius požymius yra analizuojama tik kiekvieno ypatingojo taško lokali aplinka, todėl smulkūs lapeliai miške gali būti panašūs į sienų plytas ar mažesnius langelius miesto vaizduose.

Klasterizavimo algoritmas buvo testuotas su kita paveikslėlių aibe – kaip pirmas objektas pasirinkti miesto vaizdai (94 paveikslas), o antrasis objektas – kalnų vaizdai (užklausa „mountain“) (106 paveikslas).

106 paveikslas. Kalnų paveikslėliai

Miesto ir kalnų paveikslėliai buvo klasterizuojami ESOM neuroniniu tinklu, pasirinkus gardelės dydį 50x50 ir 10 mokymosi epochų. Gautoji U-Matrix parodyta 107 paveiksle, o jos klasterizavimas vandenskyros algoritmu su 0,6 užliejimo aukščiu – 108 paveiksle.



107 paveikslas. U-Matrix gauta iš 50x50 neuronų ESOM tinko mokyto 10 epochų su miesto ir kalnų paveikslėliais.

108 paveikslas. Neuronų klasterizavimas, naudojant vandenskyros algoritmą su slenksčiu 0,6.

Klasterizuojant neuronus susidarė 8 klasteriai – vienas labai didelis, ir keli klasteriai tik su vienu BMU neuronu. Geltonajame, oranžiniame ir mėlynajame klasteriuose atsirado ir miesto, ir kalnų vaizdų, į žaliąjį klasterį pateko vien miesto vaizdai, o į rožinį – vien kalnai. Paveikslėlių klasterizavimas parodytas 109 (geltonasis klasteris), 110 (mėlynasis klasteris), 111 (rožinis klasteris), 112 (žaliasis klasteris), 113 (žydrasis klasteris), 114 (oranžinis klasteris), 115 (violetinis klasteris), 116 (rudasis klasteris) paveiksluose.

109 paveikslas. Geltonajam klasteriui priklausantys paveikslėliai.

110 paveikslas. Mėlynajam klasteriui priklausantys paveikslėliai.

111 paveikslas. Rožiniam klasteriui priklausantys paveikslėliai.

112 paveikslas. Žaliajam klasteriui priklausantys paveikslėliai.

113 paveikslas. Žydrajam klasteriui priklausantis paveikslėlis.

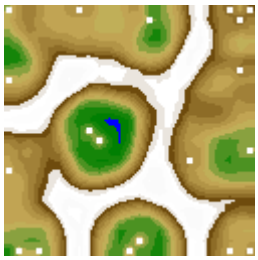
114 paveikslas. Oranžiniam klasteriui priklausantys paveikslėliai.

115 paveikslas. Violetiniam klasteriui priklausantis paveikslėlis.

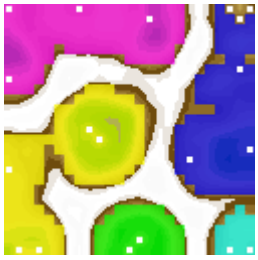
116 paveikslas. Rudajam klasteriui priklausantis paveikslėlis.

Geltonajam klasteriui priklauso daugiausia paveikslėlių, kartu jis ir pats nehomogeniškiausias klasteris. Taip pat yra du klasteriai po du paveikslėlius priklausančius skirtingoms kategorijoms – mėlynas ir oranžinis. Rožiniame klasteryje yra 4 kalnų paveikslėliai, taip pat ir žaliajame klasteryje yra 2 miesto vaizdai. Likusieji klasteriai turi tik po 1 paveikslėlį, todėl juos reikėtų interpretuoti kaip triukšmą.

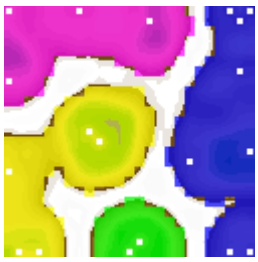
Ta pati duomenų aibė buvo klasterizuojama, naudojant kitus algoritmo parametrus: ESOM tinklo dydis 25x25 neuronai, 10 mokymosi epochų. Gautoji U-Matrix parodyta 117 paveiksle. Tinklo neuronai buvo klasterizuojami vandenskyros algoritmu taikant 0,6 ir 0,7 aukščius, rezultatai parodyti atitinkamai 118 ir 119 paveiksluose.



117 paveikslas. U-Matrix, gauta iš ESOM neuroninio tinklo su 25x25 neuronų gardele, mokyto 10 epochų. Mokymosi aibė – miesto ir kalnų vaizdai.

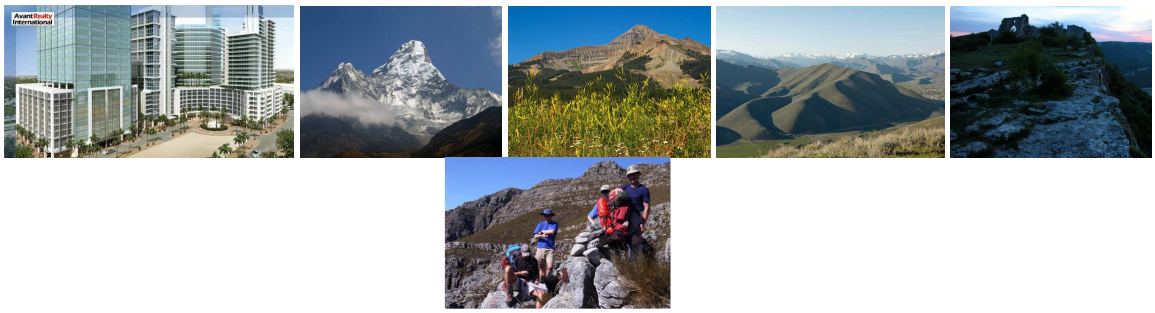


118 paveikslas. Neuronų klasterizavimas vandenskyros algoritmu su slenksčiu 0,6.



119 paveikslas. Neuronų klasterizavimas vandenskyros algoritmu su slenksčiu 0,7.

Neuronų klasterizavimas taikant 0,6 ir 0,7 slenksčius yra labai panašūs, tik taikant didesnį slenkstį mėlynajam klasteriui yra priskiriami žydrojo klasteriu neuronai. Taip pat taikant 0,6 slenkstį mėlynojo klasterio pakraštyje esantys kai kurie BMU neuronai nepatenka į klasterį, taigi, jie būtų priskiriami triukšmui. Klasterizavimas su 0,7 aukščiu atrodo geresnis. Paveikslėlių susiskirstymas parodytas 120 (geltonasis klasteris), 121 (mėlynasis klasteris), 122 (rožinis klasteris), 123 (žaliasis klasteris) ir 124 (triukšmas) paveiksluose.



120 paveikslas. Geltonojo klasterio paveikslėliai.

121 paveikslas. Rožiniam klasteriui priklausantys paveikslėliai.

122 paveikslas. Mėlynojo klasterio paveikslėliai.

123 paveikslas. Žaliojo klasterio paveikslėliai.

124 paveikslas. Jokiam klasteriui nepriklausantis paveikslėlis.

Naudojant mažesnę ESOM tinklo gardelę gauname mažiau klasterių, be to jie yra homogeniškesni – geltonajame ir mėlynajame klasteriuose tėra tik po 1 paveikslėlį iš kitos kategorijos, o rožiniame klasteryje visi paveikslėliai priklauso kalnų vaizdams.

Šiame skyriuje buvo parodyti realių gyvenimiškų vaizdų klasterizavimo rezultatai. Klasterizavimo algoritmas veikia kiek prasčiau negu su kontroliuojamoje aplinkoje užfiksuotais vaizdais, tačiau parodė gan neblogus grupavimo sugebėjimus.

Išvados

Šiame skyriuje bus pateiktos išvados, susijusios su taškinių požymių išskyrimo, aprašymo, paveikslėlių palyginimo ir klasterizavimo algoritmais.

Magistriniame darbe buvo palyginti 9 taškinių požymių išskyrimo bei aprašymo algoritmai. Praktiškai buvo parodyta, kad SIFT detektorius nenusileidžia kitiems algoritmams, tokiems kaip Hessian-Affine ar Harris-Affine. Teoriškai buvo analizuojamas jo veikimas ir nustatyta, jog jis duoda tikslesnius ir stabilesnius rezultatus už kitus detektorius. Todėl buvo padaryta išvada, jog SIFT algoritmas labiausiai tinkamas paveikslėlio ypatingiems taškams surasti bei aprašyti.

Darbe buvo tiriamas paveikslėlių palyginimo algoritmas, naudojantis SIFT ypatinguosius taškus. Remiantis ROC kreivėmis ir EER rodikliu eksperimentiškai rasti optimaliausi palyginimo algoritmo parametrai – maksimalus reikšmingas atstumas tarp SIFT taškų sudarančių porą, minimalus atstumų santykis tarp SIFT taško ir jo artimiausio kaimyno bei sekančio artimiausio kaimyno, bei simetrinis SIFT porų radimas, kuris pasirodė veikiantis patikimiau už asimetrinį. Tolimesnė paveikslėlių palyginimo algoritmo analizė, tikėtina, galėtų pasiūlyti dar geresnius atpažinimo rezultatus, nors rastasis algoritmas taip pat veikia pakankamai patikimai.

Darbe buvo įgyvendintas klasterizavimas neuroniniu tinklu, analogiškas naudotam bakalauriniame darbe. Nustatyta, kad naujasis algoritmas geba apdoroti daug didesnius paveikslėlius negu ankstesnis. Bakalauriniame darbe naudotas algoritmas galėjo apdoroti maksimaliai 32x32 dydžio paveikslėlius, tuo tarpu naujasis algoritmas sėkmingai veikia su 384x288 pikselių dydžio bei didesniais paveikslėliais.

Be to, naujasis klasterizavimo algoritmas veikia kur kas greičiau negu ankstesnis. Bakalauriniame darbe klasterizavimui buvo naudotos paveikslėlių pikselių intensyvumo reikšmės, o šiame darbe iš paveikslėlių yra išskiriamos įdomios sritys ir vėliau klasterizavimui naudojami tik tų sričių aprašymai.

Dar vienas magistriniame darbe aprašyto algoritmo privalumas – jis didžiąją dalį paveikslėlių palyginimų atlieka dar prieš klasterizavimą – ESOM tinklo mokymosi metu atstumas skaičiuojamas naudojant iš anksto suskaičiuotą panašumų tarp paveikslėlių matricą. Tokiu būdu pats klasterizavimo procesas yra labai trumpas.

Taigi, šiame darbe įgyvendinta klasterizavimo metodika veikia greičiau bei geba analizuoti didesnius paveikslėlius lyginant su bakalaurine darbe naudotu klasterizavimo algoritmu.

Šiame darbe klasterizavimui buvo panaudoti ne tiesioginiai paveikslėlio požymiai (pikseliai), o išvestinės savybės ir parodyta, jog jos veikia greičiau ir universaliau negu tiesioginiai požymiai.

Rekomendacijos

Tęsiant šį darbą galima būtų toliau tirti paveikslėlių palyginimo algoritmą. Jis susideda iš dviejų pagrindinių etapų: (1) SIFT taškų porų radimo ir (2) atstumo ar panašumo tarp paveikslėlių apskaičiavimo.

Kadangi SIFT taškų porų radimo yra jau apibrėžtas ir plačiai taikomas, todėl šioje dalyje reikėtų tik optimizuoti SIFT porų parinkimo slenkstį – dydį nusakantį kokiu santykiu turėtų skirtis atstumas tarp vieno paveikslėlio SIFT taško ir jo artimiausio taško kitame paveikslėlyje nuo atstumo tarp to paties SIFT taško iki sekančio geriausio taško kitame paveikslėlyje. Standartinėje algoritmo realizacijoje patariamas santykis yra 0,6, t.y. atstumas iki artimiausio taško turėtų būti bent 60% mažesnis už atstumą iki sekančio artimiausio taško. Magistriniame darbe buvo nustatyta, kad su pavyzdinėmis duomenų aibėmis, optimaliausias santykis yra 1, t.y. atstumai gali nesiskirti. Tęsiant šį tyrimą galima būtų automatizuoti santykio parinkimą priklausomai nuo vyraujančių atstumų tarp SIFT taškų porų – jei paveikslėlyje yra daug panašių SIFT taškų, tuomet didesnis atstumų santykis atmestų potencialiai teisingas SIFT taškų poras, iš kitos pusės, jei paveikslėlio SIFT taškai yra pakankamai skirtingi, tuomet būtų tikslinga parinkti didesnį atstumų santykių parametrą, kad būtų išvengta triukšmingų SIFT taškų porų atsiradimo.

Antrajame paveikslėlių palyginimo etape reikėtų ištirti kai kurių SIFT taškų porų atmetimo galimybes. Pavyzdžiui, remiantis SIFT taško papildoma informacija – taškų koordinatėmis paveikslėlyje, charakteringu posūkiu bei charakteringu dydžiu – galima būtų atmesti arba sumažinti svorį tų SIFT taškų porų, kurios atrodo išskirtinės. Pavyzdžiui, jei pirmajame paveikslėlyje vyraujantis charakteringas posūkis yra 30^0, o antrajame 125^0, bei atsiranda tokia SIFT taškų pora, kurios pirmojo taško (iš pirmojo paveikslėlio) charakteringas posūkis yra 95^0 , o antrojo taško (iš antrojo paveikslėlio) – 250^0, tuomet galbūt būtų tikslinga neįskaičiuoti šios poros atstumo į bendrą atstumo tarp paveikslėlių skaičių. Taip pat tiriant paveikslėlių palyginimo algoritmą galima būtų parinkti tikslesnį normalizacijos dydį.

Tobulinant klasterizavimo algoritmą būtų įdomu analizuoti automatinį parametrų parinkimą, tiek neuroninio tinklo mokymui, tiek jo neuronų klasterizavimui vandenskyros metodu. Bakalauro darbe buvo parodyta, kad klasterizavimo indeksai veikia nepatikimai, todėl reikėtų ištirti kitokias parametrų parinkimo galimybes.

Tęsiant šį darbą ir SIFT taškinių požymių analizę, šiuos požymius galima taikyti ne tik klasterizavimo, bet ir klasikiniams klasifikavimo algoritmas. Tikėtina, jog naudojant SIFT taškinius požymius paveikslėlių aprašymui, galima būtų pagerinti įvairių nespecializuotų klasifikavimo algoritmų veikimą. Išvestinių objekto požymių naudojimas yra gan nauja ir nedaug ištirta sritis duomenų gavybos algoritmų kontekste.

Literatūros sąrašas

[Aff07] Affine Covariant Features paveikslėlių duomenų bazė. Aprašymas, naudojimas bei paveikslėliai: http://www.robots.ox.ac.uk/~vgg/research/affine/ , 160 KB, 2007.

[Ams07] Amsterdam Library of Object Images duomenų bazė. Aprašymas bei paveikslėliai: http://staff.science.uva.nl/~aloi/

[Ber02] P. Berkhin. Survey of Clustering Data Mining Techniques. In: Accrue Software Inc., 2002.

[BMP02] S.Belongie, J.Malik, J.Puzicha. Shape Matching and Object Recognition Using Shape Contexts. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 2, no. 4, 2002, pages 509-522.

[Cal07a] Caltech 101 duomenų bazė. Aprašymas ir paveikslėliai: http://www.vision.caltech.edu/Image_Datasets/Caltech101/, 15,6 MB, 2007.

[Cal07b] Caltech 256 duomenų bazė. Aprašymas ir paveikslėliai: http://www.vision.caltech.edu/Image_Datasets/Caltech256/

[Can86] Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

[CSC07] CSCLAB duomenų bazė. Aprašymas ir paveikslėliai: http://csclab.ucsd.edu/labeledimages.php,

[DH97] P. Demartines and J. Herault. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. In IEEE Trans. Neural Networks, vol. 8, pp. 148-154, 1997.

[FA91] W.Freeman, E.Adelson. The Design and Use of Steerable Filters. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, no. 9, 1991, pages 891-906.

[FFP04] L. Fei-Fei, R. Fergus, P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision, 2004

[FPS+96] U.M. Fayyad, G. Piatesky-Shapiro, P. Smuth and R. Uthurusamy. Advances in Knowledge Discovery and Data Mining. AAAI Press, 1996.

[FS04] D.Frolova, D.Simakov. Matching with Invariant Features. The Weizmann Institute of Science, www.wisdom.weizmann.ac.il/~deniss/vision_spring04/files/InvariantFeatures.ppt, 2,5 MB, 2004.

[GGB06] M.Grabner, H.Grabner, H.Bischof. Fast Approximated SIFT. In Proc. ACCV 2006, Hyderabad, India Publisher Springer, LNCS 3851, pages 918-927.

[GHP07] G.Griffin, A.Holub, P.Perona. Caltech-256 Object Category Dataset, California Institute of Technology, http://vision.caltech.edu/Image_Datasets/Caltech256/paper/256.pdf, 1,2 MB, 2007.

[HM82] J.A.Hanley, B.J.McNeil. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve, Radiology 143, 1982, p. 29-36

[HS88] C.Harris, M.Stephens. A Combined Corner and Edge Detector. In Proceedings of the 4th Alvey Vision Conference,1988, pages 147-151.

[JMF99] A.K. Jain, M.N. Murty, P.J. Flyn. Data Clustering: A Review. ACM Computing Surveys, Vol. 32, No. 3, September 1999.

[KD87] J.Koenderink, A.van Doorn. Representation of Local Geometry in the Visual System. Biological Cybernetics, vol. 55, 1987, pages 367-375.

[Key07] Keypoint detector tinklalapis. SIFT algoritmo aprašymas, straipsniai bei realizacija. http://www.cs.ubc.ca/~lowe/keypoints/

[Koh82] T. Kohonen. Self-organizing formation of topologically correct feature maps. Biol. Cyb., 43(1): 59-69, 1982.

[Koh95] T. Kohonen. Self-Organizing Maps, Springer, Berlin, Heidelberg, 1995, (Third Extended Edition).

[Koh98] T.Kohonen. The Self-Organizing Map. Neurocomputing, 21(1-3): 1-6, 1998

[Koh99a] T. Kohonen. Data mining by the self-organizing map method, Uncertainty in Intelligent and Information Systems, World Scientific, 1999.

[Koh99b] T.Kohonen. Fast evolutionary learning with batch-type self-organizing maps, in Proc. Of 52nd Session of the International Statistical Institute (ISI'99), Helsinki, Finland, August 10-18, 1999

[KS04] Y.Ke, R.Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. Proceedings Conference Computer Vision and Pattern Recognition, 2004, pages 511-517.

[KS96] B.J.A. Kröse , P.P.van der Smagt. An Introduction to Neural Networks, The University of Amsterdam, 1996.

[KZB04] T.Kadir, A.Zisserman, M.Brady. An Affine Invariant Salient Region Detector. In Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic, 2004, pages 345-457.

[Lin98] T.Lindeberg. Feature detection with automatic scale selection. In International Journal of Computer Vision, 30(2), 1998, pages 79-116.

[Low04] D. Lowe "Distinctive Image Features from Scale-Invariant Keypoints", IJCV 60, 2004, pages 91-110

[Low99] D.Lowe. Object recognition from local scale-invariant features, In Proc. ICCV. Volume 2, 1999, pages 1150-1157.

[LSP03] S.Lazebnik, C.Schmid, J.Ponce. Sparse Texture Representation Using Affine-Invariant Neighborhoods. Proceedings Conference Computer Vision and Pattern Recognition, 2003, pages 319-324.

[Mac67] J.B. MacQueen. Some methods for Classification and Analysis of Multivariate Observations. In: Proceedings of 5th Berkley Symposium on Mathematical Statistics and Probability, Volume I: Statistics, University of California Press, pp. 281-297, 1967.

[Mar06a] D.Martišiūtė. Vaizdų klasterizavimas savaime susitvarkančiais tinklais. Baigiamasis bakalauro darbas, 2006

[Mar06b] D.Martišiūtė. Straipsnio „Fast Approximated SIFT" analizė. Kurso Duomenų struktūros ir algoritmai Praktinis darbas, 2006

[MCU+02] J.Matas, O.Chum, M.Urban, T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. In Proceedings of the British Machine Vision Conference, Cardif, UK, 2002, pages 384-393.

[MGB07] M.Merler, C.Galleguillos, S.Belongie. Recognizing Groceries in situ Using in vitro Training Data, *to appear*, SLAM 2007, Minneapolis, MN,

http://grozi.calit2.net/files/grozi_slam.pdf, 697 KB, 2007.

[MS02] K.Mikolajczyk, C.Schmid. An Affine Invariant Interest Point Detector. In Proceedings of the 7th International Conference on Computer Vision, Copenhagen, Denmark, 2002.

[MS03] K.Mikolajczyk, C.Schmid. A Performance Evaluation of Local Descriptors. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA, 2003

[MS04] K.Mikolajczyk, C.Schmid. Scale and Affine Invariant Interest Points Detectors. In International Journal on Computer Vision, 60(1), 2004, pages 63-86.

[MS05] K.Mikolajczyk, C.Schmid. A Performance Evaluation of Local Descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, 2005

[MTS+05] K.Mikolajczyk, T.Tuytelaars, C.Schmid, A.Zisserman, J.Matas, F.Schaffalitzky, T.Kadir, L.Van Gool. A Comparison of Affine Region Detectors. In International Journal of Computer Vision, 65(1-2), 2005, pages 43-72.

[PCA07] PCA-SIFT tinklalapis http://www.cs.cmu.edu/~yke/pcasift/ . 3,1 KB, 2007

[PF98] P. Perona and W. T. Freeman “A Factorization Approach to Grouping” Proceedings of the 5th European Conference on Computer Vision, Volume I, 1998, pages 655–670.

[PFL01] X. Polanco, C. Francois, J.C. Lamirel. Using Artificial Neural Networks for Mapping of Science and Technology: A Multi Self-Organizing Maps Approach. Scientometrics, 01389130, 2001, pages 267-292.

[Por05] F.Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In Proc. CVPR. Volume I, 2005, pages 829-836.

[Sam69] J.W. Sammon Jr.. A nonlinear mapping for data structure analysis. In IEEE Trans. Comput. Vol. C-18, pp. 401-409, 1969.

[SH90] G.L. Scott, H.C.Longuet-Higgins. Feature grouping by ‘relocalisation’ of eigenvectors of the proximity matrix. In Proc. British Machine Vision Conference, Oxford, UK, 1990, pages 103–108.

[SM00] J. Shi, J. Malik. Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 2000, pages 888-905.

[SZ02] F.Schaffalitzky, A.Zisserman. Multi-View Matching for Unordered Image Sets. In Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark, 2002, pages 414-431.

[TV04a] T.Tuytelaars, L.Van Gool. Wide Baseline Stereo Matching Based on Affine Invariant Regions. International Journal on Computer Vision, 59(1), 2004, pages 61-85.

[TV04b] T.Tuytelaars, L.Van Gool. Matching widely separated views based on affine invariant regions , In IJCV 59(1), 2004, pages 61-85.

[TV99] T.Tuytelaars, L.Van Gool. Content-based Image Retrieval Based on Local Affinelly Invariant Regions. In Internation Conference on Visual Information Systems, 1999, pages 414-431.

[Ult03] A. Ultsch. Maps for the Visualization of high-dimensional Data Spaces, In Proceedings Workshop on Self-Organizing Maps (WSOM 2003), Kyushu, Japan, 2003, p. 225-230

[Ult93] A. Ultsch. Knowledge Extraction from Self-Organizing Neural Networks, In Opitz,

O., editor, Information and Classification, Springer, 1993.

[Ult95] A. Ultsch, Self Organizing Neural Networks perform different from statistical k-means clustering, Gesellschaft für Klassifikation, Basel 8th – 10th March, 1995.

[UM05] A. Ultsch, F. Mörchen. ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM. Technical Report Dept. Of Mathematics and Computer Science, University of Marburg, Germany, No. 46, 2005

[US90] A.Ultsch, H.P. Siemon. Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis, in Proc. Int. Neural Network Conf. Dordrecht, The Netherlands, 1990, p. 305-308.

[UV94] A. Ultsch, C. Vetter. Self-Organizing-Feature-Maps versus Statistical Clustering Methods: A Benchmark, Research report No 90194, Department of Computer Science, September 1994, University of Marburg.

[VA00] J. Vesanto and E. Alhoniemi. Clustering of the Self-Organizing Map. IEE Transactions on Neural Networks 11, 2000, pp. 586-600.

[VH01] M.Vazirgiannis, M. Halkidi. Introduction to Data Mining. (http://heldinet.dbnet.ece.ntua.gr/), Thessaloniki, Greece, 2001. http://www.db-net.aueb.gr/index.php/corporate/content/download/262/1047/file/slides_HELDiNET2001.zip 1.94MB.

[VHG03] M. Vazirgianis, M. Halkidi, and D. Gunopoulos. Uncertainty Handling and Quality Assessment in Data Mining. Springer-Verlag, LNAI Series, 2003, 226 pages.

[VJ01] P.Viola, M.Jones. Rapid object detection using a boosted cascade of simple features. In Proc. CVPR. Volume I, 2001, pages 511-518.

[Wei06] E. W. Weisstein. K-Means Clustering Algorithm. MathWorld-A Wolfram Web Resource: http://mathworld.wolfram.com/K-MeansClusteringAlgorithm.html. 39,8 KB, 2006.