KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Vaidas Velutis

Taškinių vaizdų vektorizavimas ploninimo metodu

Magistro darbas

Darbo vadovas
doc. dr. Antanas Lenkevičius

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Vaidas Velutis

Taškinių vaizdų vektorizavimas ploninimo metodu

Magistro darbas

Recenzentas
prof. Vacius Jusas
2007-05-23

Vadovas
doc. dr. Antanas Lenkevičius
2007-05-23

Atliko
IFM-1/2 gr. stud. Vaidas Velutis
2007-05-23

Kaunas, 2007

Turinys

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

SUMMARY

Vectorization of raster data using thining algorithm

Until the early 1980`s, computer graphics was a small, specialized field, largely because the hardware was expensive and graphics-based application programs that were easy to use and cost effective-were few. Then personal computers with built-in raster graphics displays popularized the use of graphics for user – computer interaction and as they become more affordable, an explosion of easy-to-use and inexpensive graphics-based applications soon followed.

Since then, computer graphics and image processing have experienced vigorous growth, having been subject to interdisciplinary study and research in such fields as engineering, computer science, physics, chemistry, biology and medicine.

One of the areas that have benefited a great deal was computer aided design (CAD). Several non-raster based graphics formats – vector formats – were introduced as more convenient and appropriate means of storage for CAD systems. It became necessary for print work produced before the appearance of CAD systems to be reinstated in electronic form. In the early days this work had to be done manually. Later, with the development of image processing, specialized tools were devised to tackle the problem of *vectorization*, or raster-to-vector conversion.

The aim of this thesis is to improve vectorization system, and to compare vectorization results with similar systems.

The main steps of raster data vectorization is discussed in this document. In project part of this thesis you will find the main technical and design issues of vectorization system that was developed during master course. New thining algorithm that uses Delaunay triangulation was implemented and experiments showed that it is about 75% faster than typical thining algorithm (Zhang and Suen). Nevertheless the thining result was cleaner than result of typical thing algorithm. Vectorization system performance and results were compared with other vectorization software.

ĮVADAS

Iki devyniolikto amžiaus devinto dešimtmečio pradžios kompiuterinė grafika buvo labai maža, specializuotas veiklos sfera. To priežastis buvo brangi techninė įranga ir mažas taikomųjų programų, kurios būtų ne brangios bei efektyvios, pasirinkimas. Tokių programų skaičius žymiai išaugo, kai pasirodė personaliniai kompiuteriai su vaizduokliais ir vartotojas galėjo tiesiogiai dirbti su grafiniais vaizdais.

Nuo to laiko kompiuterinė grafika ir taškinių vaizdų apdorojimas labai stipriai patobulėjo. Tai tapo inžinerinių, informatikos, fizikos, medicinos bei biologijos mokslo šakų disciplinomis universitetuose.

Viena iš sričių, kuri pažengė toliausiai šioje srityje yra kompiuterinis dizainas (CAD). Buvo pasiūlyta naudoti vektorinį duomenų formatą vietoje taškinių vaizdų, kadangi vektorinis formatas yra patogesnis ir tinkamesnis saugoti duomenims. Todėl atsirado būtinybė senus popierinius darbus, naudotus prieš pasirodant CAD sistemoms, paversti į vektorinį formatą. Pačioje pradžioje tai buvo atliekama rankiniu būdu, tačiau tobulinant taškinių vaizdų apdorojimo priemones buvo sukurti specializuoti įrankiai padedantys taškinius vaizdus versti į vektorinį formatą (vektorizuoti).

Šių tezių tikslas yra sukurti bei patobulinti sistemą gebančią vektorizuoti taškinius vaizdus. Palyginti naudojamus algoritmus bei sistemos veikimą su panašiomis sistemomis.

Šių tezių teorinėje dalyje yra detaliai aprašyti vektorizavimo procesą sudarantys veiksmai bei algoritmai. Projektinėje dalyje pateikti magistratūros studijų metu sukurtos programinės įrangos, gebančios vektorizuoti taškinius vaizdus, techninės-projektinės dokumentacijos esminiai aspektai. Tyrimo dalyje pasiūlytas taškinio vaizdo ploninimo algoritmo patobulinimas. Nustatyta, kad ploninimo algoritmas naudojantis Delaunay trianguliaciją veikia 75% greičiau už įprastą ploninimo algoritmą pasiūlytą Zhang and Suen. Eksperimentinėje dalyje sukurtoji vektorizavimo sistema yra palyginta su kitomis vektorizavimo sistemomis. Palyginti trys sistemų parametrai: vektorizavimo sparta, išskirtų objektų skaičius bei verteksų skaičius išskirtose polilinijose.

1 ANALITINĖ DALIS

1.1 Problemos aprašymas

Taškinių vaizdų vektorizavimas, kitaip žinomas kaip rastro konvertavimas į vektorių, yra procesas, kurio metu taškiniame vaizde yra randami vektoriai (tiesūs linijų segmentai). Vektorizavimas yra plačiai naudojamas dokumentų analizei atlikti: simbolių, teksto ar grafinių objektų atpažinimui. Paprasčiausio vektorizavimo metu taškinio vaizdo pikseliai yra apjungiami į primityvias linijas, kurios charakterizuojamos atributais (linijos pločiu ar charakteringas taškais), sudėtingesnio vektorizavimo metu gali būti išskiriamas tekstas ar grafiniai objektai tokia kaip apskritimai, lankai ir pan.

Šiuo metu yra sukurta nemažai komercinių sistemų, skirtų taškinių vaizdų (rastrų) vektorizavimui. Taip yra todėl, kad visos jos nėra tobulos. Kol bus būtinybė versti taškinius vaizdus į vektorinį formatą, tol bus kuriamos naujos, efektyvesnės taškinių vaizdų vektorizavimo sistemos.

1.2 Taškiniai ir vektoriniai vaizdai

Šio skyrelio tikslas yra supažindinti skaitytoją su sąvokomis bei žymėjimais, kurie bus naudojami šiame dokumente.

1.2.1 Taškinio vaizdo modelis

Formaliai terminas *taškinis vaizdas* (arba *rastrinis vaizdas*) reiškia dvimatę monochrominę šviesos intensyvumo funkciją. Ši funkcija aprašoma taip – *f(x, y)*, kur *f* reikšmė tam tikrose erdvės koordinatėse *(x, y)* grąžina vaizdo intensyvumą tame taške. Vaizdo intensyvumas gali būti 0 arba 1. Monochrominiai vaizdai taip pat gali būti vadinami *binariniais*. Kiekvienas taškinio vaizdo elementas *f(x, y)* yra vadinamas *vaizdo elementu*. Vaizdo elementai, kurių reikšmė yra 0 bus vadinami *foniniais* vaizdo elementais, o tie kurių reikšmė lygi 1 – *objekto* vaizdo elementais. Objekto vaizdo elementai yra žymimi juodais kvadratais. Vaizdo elementų poaibis *f(x, y)* bus žymimas *S*.

1 pav. Binarinio vaizdo pavyzdys

1.2.1.1 Gretimi vaizdo taškai

Vaizdo elementas *p* esantis koordinatėse *(x, y)* turi keturis gretimus vaizdo taškus – du *vertikalius* ir du *horizontalius*. Gretimų vaizdo taškų koordinatės apskaičiuojamos taip:

$$(x + 1, y), (x – 1, y), (x, y + 1), (x, y - 1)$$

Šis vaizdo elementų rinkinys, vadinamas *keturiais p kaimynais*, bus žymimas $N_4(p)$. Keturi *įstrižainėse* esantys *p* kaimynai turės tokias koordinates:

$$(x + 1, y + 1), (x + 1, y - 1), (x - 1, y + 1), (x - 1, y - 1)$$

Įstrižainėse esantys vaizdo elemento *p* kaimynai bus žymimi $N_D(p)$. Šie ir prieš tai minėti keturi taškai yra vadinami *aštuoniais p kaimynais* ir yra žymimi $N_8(p)$ [1].

1.2.1.2 Vaizdo taškų sąjunga

Vaizdo elementų sąjunga yra labai svarbi sąvoka, naudojama norint nustatyti objektų ribas taškiniame vaizde. Norint nustatyti ar vaizdo taškai yra sąjungoje reikia nustatyti ar jie yra kaimyniniai.

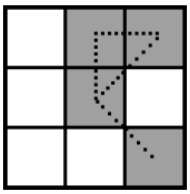
8	4/8	8
4/8	p	4/8
8	4/8	8

2 pav. Vaizdo elemento *p* sąjungos

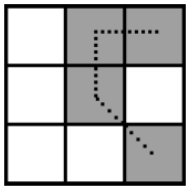
Aptarsime tris sujungimo tipus:

1. *4-sąjunga* – du vaizdo taškai *p* ir *q* yra 4-sąjungoje jeigu *q* priklauso aibei $N_4(p)$.
2. *8-sąjunga* – du vaizdo taškai *p* ir *q* yra 8-sąjungoje jeigu *q* priklauso aibei $N_8(p)$.
3. *m-sąjunga (maišyta sąjunga)* – du vaizdo taškai *p* ir *q* yra m-sąjungoje jeigu:
    a. *q* priklauso aibei $N_4(p)$, arba

b. q priklauso aibei $N_D(p)$ ir aibė $N_4(p) \cap N_4(q)$ yra tuščia (tai aibė taškų, kurie yra keturi p ir q kaimynai).



3 pav. Viduriniojo vaizdo elemento *8-sąjunga*



4 pav. Viduriniojo vaizdo elemento *m-sąjunga*

Maišyta sąjunga (m-sąjunga) yra modifikuota 8-sąjunga ir naudojama tam, kad eliminuoti pasikartojančių kelių sąjungas, kuriuos nuolat atsiranda naudojant 8-sąjungą.

Jeigu p ir q priklauso taškinio vaizdo poaibiui S, tai p yra sąjungoje su q poaibyje S tik tuomet kai egzistuoja kelias nuo p iki q. Kaimyninių vaizdo elementų seka yra vadinama vaizdo elementų *grandine*, kuri susideda tik iš poaibiui S priklausančių vaizdo elementų. Vaizdo elemento p, priklausančio poaibiui S, sąjungos elementai, taip pat priklausantys poaibiui S, yra vadinami *sujungtais poaibio S komponentais* [2].

Objektu vadinsime kiekvieną sujungtą taškinio vaizdo poaibio S komponentą (pagal nutylėjimą poaibis S yra visas taškinis vaizdas).

1.2.1.3 Taikomosios programos

Taškiniai vaizdai yra naudojami saugant skaitmenines fotografijas, piešinius bei nuskenuotus vaizdus. Faktas, kad šie vaizdai yra sudaryti iš individualių vaizdo elementų, gali būti kartu privalumas ir trūkumas. Tai priklauso nuo kokio detalumo vaizdo mes norime. Didesnės rezoliucijos vaizdas duoda didesnį detalumą, tačiau užima daugiau vietos diske bei pailgėja operacijų atlikimo laikas.

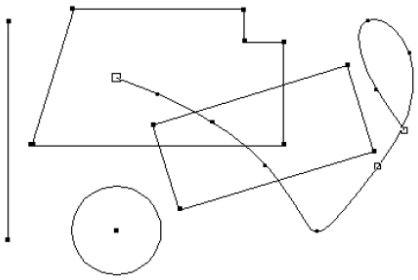
1.2.2 Vektorinio vaizdo modelis

Vektoriniu vaizdu galima vadinti aibę *vektorinių esybių* ir ryšius tarp tų esybių.

Vektorinės esybės dažniausiai yra geometriniai objektai, kurie gali būti išreikšti matematinėmis formulėmis. Pavyzdžiui linija gali būti išreikšta jos galų koordinatėmis arba pagrindo tašku, aukščiu ir ilgiu. Vektorines esybes galima lengvai valdyti bei pritraukti neprarandant vaizdo tikslumo kadangi jos nėra riboto skaičiaus vaizdo taškų kompozicijos.

Ryšiai tarp vektorinių esybių gali būti įvairūs, nuo paprasto grupavimo iki sudėtingesnių pavyzdžiui loginių operatorių.

Skirtingai negu taškiniuose vaizduose, kur rezoliucija yra iš anksto žinoma, vektoriniai vaizdai nepriklauso nuo rezoliucijos. Kitaip tariant vaizdo kokybė tiesiogiai priklauso nuo spausdintuvo ar kito išvesties įrenginio charakteristikų.



5 pav. Vektorinio vaizdo pavyzdys

1.2.2.1 Taikomosios programos

Vektorinės esybės yra pagrindiniai *GIS (Geografinė informacinė sistema)* ir *CAD (kompiuterinis dizainas)* sistemų elementai. Vektoriniai vaizdai yra plačiai naudojami visose srityse kur reikalingas *kompiuterinis vaizdas*.

1.3 Vektorizavimas

Vektorizavimu yra vadinamas procesas, kurio metu taškinis vaizdas yra paverčiamas vektoriniu vaizdu. Šis procesas, atliekamas rankiniu būdu, užima daug laiko ir yra labai brangus.

Vektorizavimas gali būti įsivaizduojamas kaip objektų (geometrinių primityvų: linijų, apskritimų ir pan.) atpažinimo ir išreiškimo matematinėmis formulėmis taškiniame vaizde procesas.

1.3.1 Taikomosios programos

Šiame skyrelyje apžvelgsime populiariausias vektorizavimo taikymo sritis.

1.3.1.1 Medicina

Medicinos technologijos stipriai patobulėjo per paskutinius 10 – 15 metų. Atsirado sudėtingos vaizdo pateikimo sistemos, pavyzdžiui: magnetinio rezonanso atvaizdavimo įrenginiai, rentgeno spindulių bei ultragarso skaitliai. Šių įrenginių pateikiamo vaizdo analizė galėtų būti žymiai paprastesnė jeigu jie būtų pateikiami paprastesnėje formoje, kur kiekviena vaizdo dalis egzistuotų kaip atskira esybė.

1.3.1.2 Geografinės informacinės sistemos (GIS)

Geografinės informacinės sistemos teikia aukšto detalumo žemėlapius, kadastrines bei topografines nuotraukas ir pan. Labai svarbu, kad šie duomenys būtų saugomi tinkamoje formoje. Tam labiausiai tinka vektorinis formatas kadangi šiose sistemose yra labai didelis duomenų kiekis.

1.3.1.3 Inžinerija (CAD)

Visuose inžineriniuose darbuose yra naudojami skaitmeniniai arba popieriniai brėžiniai, planai ir pan. Planavimo darbų kainą sumažino *CAD* sistemų, kurios stipriai palengvino šių darbų atlikimą, atsiradimas. *CAD* sistemų tikslas buvo visiškai atsisakyti popierinės duomenų saugojimo formos, o tai reiškia, kad dauguma popierinių planų turi būti paversti į elektroninę formą. Šis darbas reikalauja labai daug laiko trunkančio rankinio vektorizavimo. Šioje srityje naudojama daugiausia automatinių vektorizavimo priemonių.

1.4 Svarstomos problemos

1.4.1 Įėjimų charakteristikos

Kuriant vektorizavimo sistemą nebūtina palaikyti visus taškinių vaizdų formatus. Norint gauti gerus rezultatus reikia sumažinti naudojamų taškinių vaizdų formatų skaičių. Taipogi reikia nustatyti kokius geometrinius objektus mūsų sistema turi atpažinti. GIS ir CAD sistemų atveju objektai būtų linijos, lankai, apskritimai ir pan. Sudėtingesni objektai (rodyklės, specialūs simboliai, tekstas, brūkšninės linijos) taip pat gali būti atpažįstami.

1.4.2 Reikalavimai tikslumui ir efektyvumui

Taikomosios programos naudojamos tolimesniam vektorizavimo rezultatų apdorojimui nulemia reikalingą vektorizavimo tikslumą. Tikslumo reikalavimai turi būti įgyvendinti matematiniuose ir geometriniuose skaičiavimuose, kurie naudojami vektorizavimo procese. Didžiausias tikslumas reikalingas dirbant su CAD sistemomis.

Efektyvumas yra taip pat svarbus kaip ir tikslumas. Šie abu kriterijai turi būti optimalūs. Efektyvumą labiausiai įtakoja pasirinktas sąveikos su vartotoju modelis, aprašytas sekančiame skyrelyje.

1.4.3 Sąveikos su vartotoju modelis

Vektorizavimo metu yra sprendžiama daug dviprasmiškų situacijų bei atliekami atitinkami skaičiavimai. Jeigu sistemai leisime visus sprendimus priimti pačiai – tai vektorizavimo rezultatas ir jo tikslumas gali būti nepatenkinami. Šiame skyrelyje bus aptarti trijų sąveikos su vartotoju modelių pliusai ir minusai. Pirmi du modeliai *automatinis* ir *pusiau automatinis* yra euristiniai, o trečiajame – *besimokančios sistemos* yra naudojamas dirbtinis intelektas.

1.4.3.1 Automatinis

Automatinės sistemos visas operacijas atlieka pačios be jokio vartotojo įsikišimo. Pagrindinis šių sistemų privalumas yra veikimo sparta, tačiau rankinis rezultatų redagavimas daugeliu atvejų yra neišvengiamas.

1.4.3.2 Pusiau automatinis

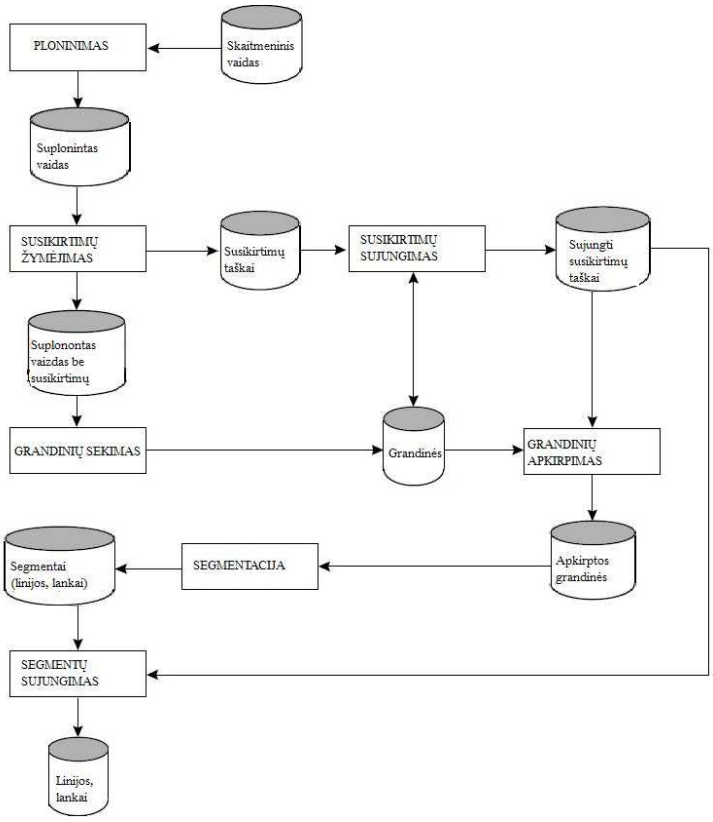
Pusiau automatinėse sistemose yra reikalingas vartotojo arba operatoriaus įsikišimas. Vartotojui gali tekti užpildyti keletą formų parenkant reikalingus parametrus ir taip padedant išspręsti neaiškias, dviprasmiškas situacijas. Šiose sistemose rečiau pasitaiko vektorizavimo klaidų, o rezultatas būna pakankamai tikslus.

1.4.3.3 Besimokančios sistemos

Žymiai sudėtingesnė sąveikos forma, paremta dirbtiniu intelektu, yra naudojama besimokančiose sistemose. Sistema iš pradžių išmokoma atpažinti paprastus objektus. Atsiradus dviprasmiškom situacijom sistema klausia operatoriaus ką daryti ir mokosi iš operatoriaus veiksmų. Didžiausias šių sistemų privalumas prieš euristines yra jų lankstumas ir galimybė pritaikyti jas daugeliui sričių.

1.5 Teorinis modelis

Automatinio vektorizavimo proceso duomenų srautų diagrama susideda iš devynių žingsnių (6 pav. **Vektorizavimo proceso duomenų srautų diagrama**).



6 pav. Vektorizavimo proceso duomenų srautų diagrama

1.5.1 Taškinio vaizdo ploninimas

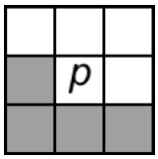
Originalus skaitmeninis vaizdas vektorizavimo procese atlieka įėjimo vaidmenį. Ploninimo (thining) tikslas yra sumažinti taškinio vaizdo objektų plotį iki vieno vaizdo taško. Ploninimo rezultatas turėtų atspindėti tikrųjų objektų, kurie turi būti išskirti iš taškinio vaizdo, formą, struktūrą bei sąjungas.

Egzistuoja du visiškai skirtingi ploninimo tipai:

- *Kontūrinių linijų* išskyrimas.
- *Centrinių linijų* išskyrimas.

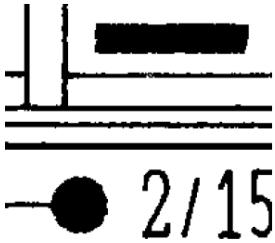
1.5.1.1 Kontūrinių linijų išskyrimas

Kontūrinių linijų (outline) išskyrimo idėja yra rasti taškinio vaizdo objektų kontūrus, jis taip pat žinomas kaip *kontūrų sekimas* [2].

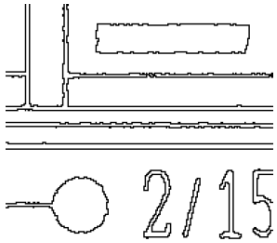


7 pav. Kontūro taškas *p*

Kontūro vaizdo elementas yra foninis vaizdo elementas, kuris turi bent vieną objekto vaizdo elementą savo aštuoniuose kaimynuose ($N_8(p)$).



8 pav. Originalus vaizdas



9 pav. Vaizdas pritaikius kontūrinių linijų išskyrimą

1.5.1.2 Centrinių linijų išskyrimas

Centrinių linijų (centerline) išskyrimo metu objektai yra suploninami iki jų *skeleto* (gali būti vadinama *skeletonizavimu*). Objekto skeletas gali būti aprašytas naudojant viduriniosios ašies transformaciją (*medial axis transformation MAT*), pasiūlytą Blum [2]. Objekto *R* viduriniosios ašies transformacija (MAT) su riba (arba kraštu) *B* yra randama taip: kiekvienam vaizdo taškui *p* objekte *R* reikia surasti artimiausią kaimyną riboje *B*. Jeigu *p* turi daugiau nei vieną tokį kaimyną, tai sakoma, kad vaizdo taškas priklauso objekto *R* MAT (skeletui).

Nors taikant MAT galima gauti norimą objekto skeltą, tačiau tai nėra optimaliausias būdas, kadangi jo metu reikia atlikti labai daug skaičiavimų. Reikėtų skaičiuoti atstumus nuo kiekvieno vidinio objekto taško iki kiekvieno objekto ribos taško. Buvo pasiūlyta keletas algoritmų pagerinančių skaičiavimų efektyvumą bei tuo pačiu gaunančių duoto objekto viduriniosios ašies transformaciją. Tai yra ploninimo algoritmai, kurie trina objekto ribos vaizdo taškus laikantis trijų sąlygų:

1. Negalima trinti galinių taškų.
2. Reikia išlaikyti sujungimus.
3. Nesukelti per didelės objekto erozijos.

Šiame skyrelyje bus aptartas algoritmas, kurį pasiūlė Zhang and Suen [5]. Algoritmas susideda iš dviejų žingsnių pritaikomų objekto kraštiniams vaizdo taškams. Toks pats algoritmas yra taikomas kraštinių vaizdo taškų radimui kontūrų sekime. Skirtumas tik tas, kad čia yra ieškoma vaizdo taškų turinčiu bent vieną foninį vaizdo tašką savo aštuoniuose kaimynuose ($N_8(p)$).

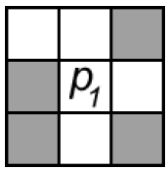
p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

10 pav. Kaimyninių vaizdo taškų išsidėstymas naudojant ploninimo algoritmą

Pagal aštuonių kaimynų išdėstymą (10 pav.) kur $p_i \in \{0, 1\}$, pirmasis žingsnis pažymi kraštinį vaizdo tašką p_1 trynimui tik tada kai tenkinamos šios sąlygos:

1. $2 \leq N(p_1) \leq 6$,
2. $S(p_1) = 1$,
3. $p_2 * p_4 * p_6 = 0$,
4. $p_4 * p_6 * p_8 = 0$.

Kur $N(p_1)$ yra kaimyninių vaizdo taško p_1 objektinių vaizdo taškų skaičius (kurių reikšmė nelygi nuliui). Taigi $N(p_1) = p_2 * p_3 * p_4 * p_5 * p_6 * p_7 * p_8 * p_9$. Tuo tarpu $S(p_1)$ yra perėjimų iš 0 į 1 (0 → 1) skaičius surikiuotoje sekoje $p_2, p_3, \ldots, p_8, p_9$. Pavyzdžiui jeigu $N(p_1) = 4$ ir $S(p_1) = 3$ tai turėtume tokį vaizdą kaip pavaizduota 11 pav..



11 pav. Vaizdas kai $N(p_1) = 4$ ir $S(p_1) = 3$

Antrajame žingsnyje pirmoji ir antroji sąlygos išlieka tokios pačios, tačiau keičiasi likusios dvi:

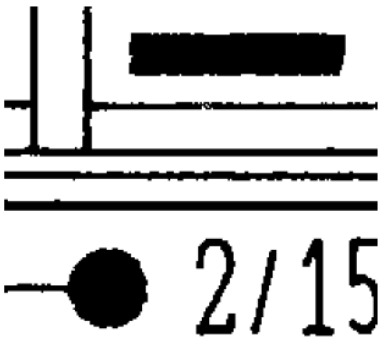
1. $2 \leq N(p_1) \leq 6$,
2. $S(p_1) = 1$,
3. $p_2 * p_4 * p_8 = 0$,
4. $p_2 * p_6 * p_8 = 0$.

Pirmasis žingsnis yra taikomas kiekvienam objekto krašto vaizdo taškui. Jeigu nors viena iš keturių sąlygų yra netenkinama tai vaizdo taško reikšmė yra nekeičiama. Jeigu visos sąlygos tenkinamos – tai vaizdo taškas yra pažymimas trynimui. Vaizdo taškas nėra trinamas tol, kol nėra apdoroti visi kraštiniai vaizdo taškai. Tai apsaugo nuo duomenų struktūros pasikeitimo algoritmo veikimo metu. Pažymėtieji trynimui vaizdo taškai yra ištrinami (pavyzdžiui jų reikšmė nustatoma į 0) tik tada kai pirmasis žingsnis yra baigtas. Antrasis žingsnis yra taikomas likusiems duomenims lygiai taip pat kaip ir pirmasis žingsnis.

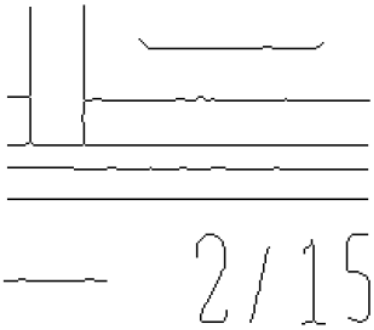
Remiantis ankstesniais samprotavimais galima daryti išvadą, kad viena ploninimo algoritmo iteracija susideda iš šių veiksmų:

1. Pirmojo žingsnio įvykdymas ir trinamų vaizdo taškų sužymėjimas.
2. Trynimui pažymėtų vaizdo taškų ištrynimas.
3. Antrojo žingsnio įvykdymas ir trinamų vaizdo taškų sužymėjimas.
4. Trynimui pažymėtų vaizdo taškų ištrynimas.

Ši procedūra yra vykdoma tol, kol neištrinamas nei vienas vaizdo taškas. Po algoritmo įvykdymo taškiniame vaizde lieka tik objektų skeletai.



12 pav. Originalus vaizdas



13 pav. Vaizdas pritaikius centrinių linijų metodą

Pirmoji sąlyga yra netenkinama tuomet kai kraštinis vaizdo taškas p_1 turi vieną arba septynis objektinius vaizdo taškus savo aštuoniuose kaimynuose. Jeigu p_1 turi tik vieną tokį kaimyną tai reiškia, kad jis yra galinis skeleto vaizdo taškas ir neturi būti trinamas. Jeigu p_1 turi septynis tokius kaimynus tai jo trynimas galėtų sukelti per didelę objekto eroziją.

Antroji sąlyga yra netenkinama tuomet kai ji yra taikoma taškams, kurie yra vieno vaizdo taško storio linijoje. Tokių vaizdo taškų trynimas atskirtų skeleto segmentus.

Trečia ir ketvirta sąlygos yra patenkinamos šiose situacijose:

$$(p_4 = 0) \vee (p_6 = 0) \vee (p_2 = 0 \wedge p_8 = 0)$$

Pagal aštuonių kaimynų išdėstymą (10 pav.) vaizdo taškas tenkinantis pirmas dvi bei šią sąlygas gali būti rytinis arba pietinis krašto taškas, arba šiaurės vakarų kampo vaizdo taškas. Bet kuriuo atveju vaizdo taškas p_1 nepriklauso skeletui ir turi būti trinamas.

Panašiai trečia ir ketvirta sąlygos yra patenkinamos šiose situacijose:

$$(p_2 = 0) \vee (p_8 = 0) \vee (p_4 = 0 \wedge p_6 = 0)$$

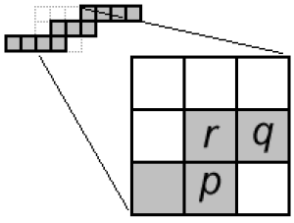
Čia vaizdo taškas tenkinantis šias sąlygas gali būti šiaurinis arba vakarinis krašto taškas, arba pietrytinio kampo taškas. Be to šiaurės rytų kampo vaizdo taškas, kuris turi $(p_2 = 0 \wedge p_4 = 0)$, bei pietvakarių kampo vaizdo taškas, kuris turi $(p_6 = 0 \wedge p_8 = 0)$, taip pat tenkina visas sąlygas.

Šis algoritmas atlieka labai daug skaičiavimų. Iteracijų skaičius priklauso nuo maksimalaus objekto storio. Laikas, per kurį vykdoma kiekviena iteracija, yra proporcingas rastų kraštinių vaizdo taškų skaičiui.

1.6 Susikirtimų žymėjimas

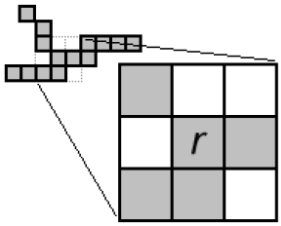
Nei vienas iš ploninimo tipų, nei centrinių linijų nei kontūrinių linijų, nenutraukia susikirtimų tarp objektų taškiniame vaizde. Rezultate liekantys vieno vaizdo taško pločio objektai yra struktūrinės 8-sąjungos grandinės, kurios gali būte sekamos nesudėtingu būdu. Tačiau reikia žinoti, kad kiekvienas objekto taškas grandinėje turi tik du savo kaimynus priklausančius ($N_8(p)$). Galiniai grandinės vaizdo taškai turi tik po vieną tokį kaimyną. Ši prielaida yra reikalinga tam, kad grandinės sekimo metu neiškiltų dviprasmybių ir kiekvienas sekantis vaizdo taškas galėtų būti unikaliai identifikuojamas. Taigi susikirtimų žymėjimo (junction mapping) funkcijos paskirtis yra garantuoti, kad grandinės sekimo metu bus galima identifikuoti kiekvieną sekantį vaizdo tašką.

Susikirtimų žymėjimas gali būti įsivaizduojamas kaip *taškinio vaizdo grafo* sudarymas. Tai nekryptingas grafas, kur susikirtimo taškai yra grafo viršūnes, o vaizdo taškų grandinės – grafo briaunos. Šis grafas, vaizduojantis taškinio vaizdo objektų susijungimų struktūrą, yra labai svarbus tolimesnėse vektorizavimo proceso fazėse.



14 pav. Vaizdo taškai *p* ir *q* yra 4-kaimynai bei 8-sąjungoje su vaizdo tašku *r*

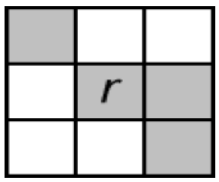
Bendrai kalbant susikirtimo taškas yra tas, kuris priklauso daugiau negu vienai grandinei. Susikirtimo taško apibrėžimas vaizdo taškų lygyje reikalauja daugiau apsvarstymų. Kiekvienas vaizdo taškas, turintis 3 arba daugiau kaimynų yra potencialus kandidatas buti susikirtimo tašku, tačiau 14 pav. rodo, kad jis gali tokiu ir nebūti. Figūroje grandinę sudaro trys vaizdo taškai *p*, *q* ir *r*. Vaizdo taškas *r* turi tris kaimynus priklausančius ($N_8(r)$), tačiau tai akivaizdžiai nėra susikirtimo taškas, kadangi vaizdo taškai *p* ir *q* yra 4-kaimynai bei 8-sąjungoje su vaizdo tašku *r*. Šioje situacijoje, kur vaizdo taškas *r* turi ne daugiau kaip 3 kaimynus priklausančius ($N_8(r)$), galima pašalinti vaizdo tašką *r* išlaikant ryšį tarp *p* ir *q*. Tokių vaizdo taškų pašalinimą galime vadinti pirmuoju susikirtimų žymėjimo algoritmo žingsniu. Reikia tiksliai nustatyti vaizdo taško *r* kaimynų skaičių, tam kad nebūtų per klaidą pašalinti susikirtimų taškai (15 pav.).



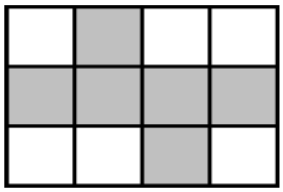
15 pav. Vaizdo taškas *r* negali būti pašalintas

Susikirtimų taškų radimo procesą galime išskaidyti į tris žingsnius:

1. *Pirmas žingsnis*. Bet kuris vaizdo taškas *r,* turintis tris arba daugiau kaimyninių vaizdo taškų priklausančių ($N_8(r)$) ir kur nei viena jų pora nėra 4-sąjungoje, yra įdedamas į susikirtimo taškų sąrašą (16 pav.). Galiausiai *r* ir visi jo ($N_8(r)$) priklausantys kaimynai yra pašalinami.
2. *Antras žingsnis*. Šis žingsnis yra reikalingas tam, kad surinkti tuos susikirtimo taškus, kurių neaptiko pirmasis žingsnis (17 pav.). Antrojo žingsnio procedūra yra panaši kaip ir pirmajame žingsnyje, tik panaikinama sąlyga dėl 4-sąjungos, tai reiškia, kad bet kuris vaizdo taškas *r*, turintis tris arba daugiau kaimyninių vaizdo taškų priklausančių ($N_8(r)$), bus traktuojamas kaip susikirtimo taškas bei pašalintas iš taškinio vaizdo kartu su jo kaimynais.



16 pav. Vaizdo taškas *r* yra susikirtimo taškas



17 pav. Situacija, kurios neišsprendžia pirmasis susikirtimų radimo algoritmo žingsnis

1.7 Grandinės sekimas

Grandinių sekimo (chain tracing) operacijos tikslas yra taškiniame vaizde išskirti atskiras vaizdo taškų atskiras ir paruošti jas segmentacijai. Operacija atliekama dviem žingsniais. Pirmuoju žingsniu yra išskiriamos visos atviros grandinės, antruoju – uždaros grandinės, kurių neaptiko pirmasis žingsnis, bei tos kurios buvo uždaros bet pasidarė atviros po pirmojo žingsnio įvykdymo.

1. *Pirmas žingsnis – atviros grandinės*.
    a. Reikia surasti objekto vaizdo tašką *p*, kuris turi tik vieną kaimyną priklausantį ($N_8(p)$). Radus tokį tašką reikią sukurti sąrašą *C*, kuris saugos šios grandinės taškus ir pradėti antrąjį etapą (*b*). Jeigu toks vaizdo taškas nerandamas procedūra yra nutraukiama.
    b. Įtraukti vaizdo tašką *p* į sąrašą *C* ir pašalinti jį iš taškinio vaizdo.
    c. Surasti vaizdo taško *p* kaimyną *q* priklausantį ($N_8(p)$). Jeigu toks rastas nustatyti, kad $p = q$ ir kartoti antrąjį etapą (*b*), kitu atveju kartoti pirmąjį žingsnį iš naujo.

18 pav. Atviros grandinės sekimas

2. *Antras žingsnis – uždaros grandinės*.
    a. Reikia surasti objekto vaizdo tašką *p*. Radus tokį tašką reikią sukurti sąrašą *C*, kuris saugos šios grandinės taškus ir pradėti antrąjį etapą (*b*). Jeigu toks vaizdo taškas nerandamas procedūra yra nutraukiama.
    b. Įtraukti vaizdo tašką *p* į sąrašą *C* ir pašalinti jį iš taškinio vaizdo.
    c. Surasti vaizdo taško *p* kaimyną *q* priklausantį ($N_8(p)$). Jeigu toks rastas nustatyti, kad $p = q$ ir kartoti antrąjį etapą (*b*), kitu atveju kartoti pirmąjį žingsnį iš naujo.

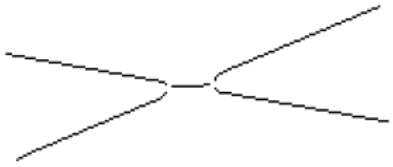
19 pav. Uždaros grandinės sekimas

1.8 Susikirtimų sujungimas

Susikirtimų sujungimo (junction merging) funkcijos tikslas yra koreguoti *taškinio vaizdo grafą*, sukurtą susikirtimų žymėjimo funkcijos metu. Funkcija suranda susikirtimo taškų, kurie yra labai arti vienas kito, poras ir pakeičia juos vienu susikirtimo tašku.

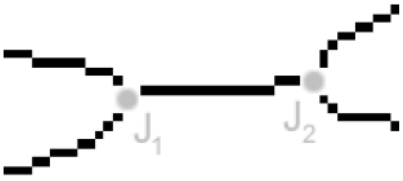
Dviejų susikertančių linijų (20 pav.) ploninimas gali duoti du susikirtimo taškus (21 pav.) nors iš tikro mes turime tik vieną susikirtimo tašką.

20 pav. Susikertančios linijos (8 vaizdo taškų storio)



21 pav. Vaizdas įvykužius ploninimo operaciją

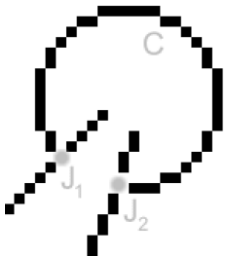
Grandinė tarp J_1 ir J_2 yra tuščia (22 pav.) ir gali būti pašalinta jei neviršija iš anksto nustatyto susikirtimo ilgio J_ε.



22 pav. Du susikirtimo taškai J_1 ir J_2

23 pav. Apjungtas susikirtimo taškas J

Minimalaus susikirtimų ilgio nustatymas yra būtinas tam, kad nebūtu pašalinti išoriniai susikirtimai (24 pav.).



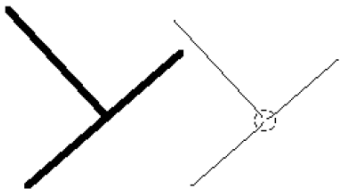
24 pav. Išorinė, du susikirtimus J_1 ir J_2 jungianti, grandinė C negali būti šalinama

Žemiau pateikiamas susikirtimų sujungimo algoritmas:

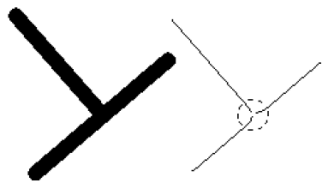
1. Patikrinti atstumus d tarp visų susikirtimo taškų J_1 ir J_2 porų.
2. Jeigu $d < J_\varepsilon$ tai taškai priklauso tam pačiam susikirtimui.
3. Jeigu tarp rastų susikirtimo taškų yra grandinė kurios ilgis $l < J_\varepsilon$ tai ją reikia pašalinti. Taipogi reikia atnaujinti *taškinio vaizdo grafą* pašalinant iš jo taškus J_1 ir J_2 ir vietoj jų įdedant naują tašką J. Grandinės priskirtos taškams J_1 ir J_2 turi būti priskirto taškui J.

1.9 Grandinės apkirpimas

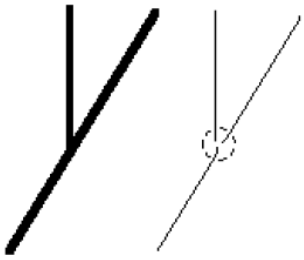
Grandinės apkirpimas (chain trimming) pašalina nepageidautinas deformacijas prie susikirtimo taškų. Deformacijos atsiranda ploninimo algoritmo metu, kadangi jis paprasčiausiai išskiria regiono viduriniąją ašį. Šios figūrų deformacijos bei iškraipymai yra paplitę tik prie susikirtimo taškų ir proporcingi susikirtimo plotui. Deformacijos apimtis priklauso nuo susikertančių objektų skaičiaus, jų pločio bei susikirtimo kampo. Figūros žemiau vaizduoja deformacijos efektą, kurį sukelia susikirtimai. Tokiu atveju pakaktų pašalinti keletą grandinės galo, kuris priklauso susikirtimui, taškų.



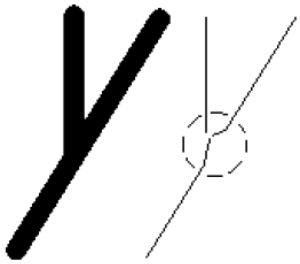
25 pav. τ-susikirtimas. 4 vaizdo taškų plotis



26 pav. τ-susikirtimas. 12 vaizdo taškų plotis



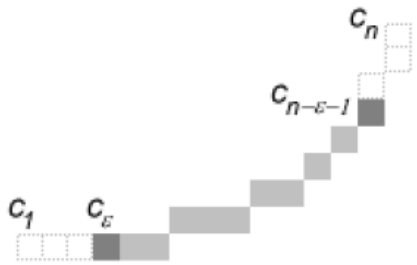
27 pav. λ -susikirtimas. 4 vaizdo taškų plotis



28 pav. λ-susikirtimas. 12 vaizdo taškų plotis

Žemiau pateiktas algoritmas aprašo procedūrą, kuri šalina grandinių galus. Reikia žinoti jog tokiu būdu gali būti pašalintos trumpos grandinės, jeigu abu jų galai yra pašalinami daugiau kaip per pusę grandinės ilgio.

1. Kiekvienai grandinei $C = c_1\ c_2\ ...\ c_{n-1}\ c_n$, kur c_i yra vaizdo taškai grandinėje, reikia vykdyti pirmą ir antrą žingsnius.
2. Išgauti du susikirtimo taškus artimiausius grandinės galo taškams c_1 ir c_n. Grandinė yra nepririšta nei prie vieno susikirtimo taško jeigu nors vienas iš rastų susikirtimo taškų yra nutolęs toliau kaip per tu vaizdo taškus nuo jam priklausančio grandinės galo. Nepririštos prie susikirtimo taškų grandinės yra nemodifikuojamos.
3. Jeigu randama pririšta prie susikirtimo taško grandinė ir iš anksto nustatytas parametras *apkirpimo ilgis* $\varepsilon \geq [^1/_2 n]$ tuomet grandinė C yra pašalinama iš *taškinio vaizdo grafo*. Jeigu $\varepsilon < [^1/_2 n]$ tuomet grandinė C yra pakeičiama tokia $C' = c_{\varepsilon+1}\ c_{\varepsilon+2}\ ...\ c_{n-\varepsilon-1}\ c_{n-\varepsilon}$. Kitaip tariant mes pašaliname ε vaizdo taškus iš grandinės galų (29 pav.).



29 pav. Grandinė $c_1\ c_2\ ...\ c_{n-1}\ c_n$ yra apkirpta abiejuose galuose kai $\varepsilon = 3$

1.10 Segmentacija

Segmentacijos tikslas yra suskaidyti grandines į mažesnius prasmingus *segmentus*, kurie gali būti paverčiami į geometrines primityvas. Kiekvienos grandinės segmentacijos rezultatas – sujungtų linijų ir lankų sąrašas.

30 pav. Vaizdo taškų grandinė

31 pav. Segmentuota grandinė

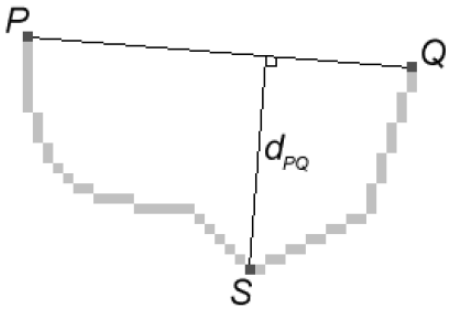
1.10.1 Keturių žingsnių segmentacija

Pirmieji du šio metodo žingsniai padalina grandinę į tiesias linijas. Kiti du žingsniai bando tas linijas paversti lankais. Jeigu norimam rezultatui pakanka tik linijų, tai paskutiniai du šio metodo žingsniai gali būti nevykdomi [3].

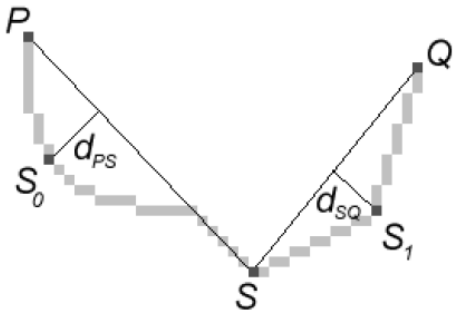
1.10.1.1 Linijų surinkimas

Linijų surinkimo (line fitting) tikslas yra rasti geriausią būdą kaip padalinti gandinę į sujungtas linijas. Tai yra daroma dalinant grandinę į dvi dalis tol kol iš anksto nustatytas kriterijus yra patenkinamas. Kriterijus vadinamas *išlenkimo tolerancija* ε, kuris nusako kokio tiesumo turi būti grandinė, kad ji būtų laikoma tiesia linija.

Reikia surasti ilgiausią statmenį tarp grandinės ir linijos jungiančios abu grandinės galus. Rastas grandinės taškas bus laikomas dalinimo tašku, kuris dalina grandinę į dvi dalis. Tarkim turim grandinę PQ (32 pav.) kur maksimalus statmuo yra d_{PQ} vaizdo taške S. Sakykime, kad $d_{PQ} > \varepsilon$ tuomet reikalingas dar vienas grandinės dalinimas vaizdo taške S. Taigi toliau bus dalinamos grandinės PS ir SQ. Jeigu $d_{SQ} \leq \varepsilon$, tai grandinė SQ yra laikoma tiesia linija ir jos toliau dalinti nebereikia. Linija SQ gali būti pridėta į sąrašą su *Lowe reikšmingumo atributu* σ, kuris apskaičiuojamas taip: $\sigma_{SQ} = d_{SQ}/||SQ||$. Grandinės PS atveju reikalingas tolimesnis dalinimas jeigu $d_{PS} > \varepsilon$.

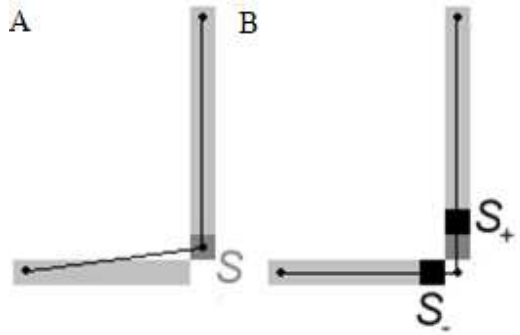


32 pav. Grandinės PQ dalinimo taškas S



33 pav. Grandinių PS ir SQ dalinimas

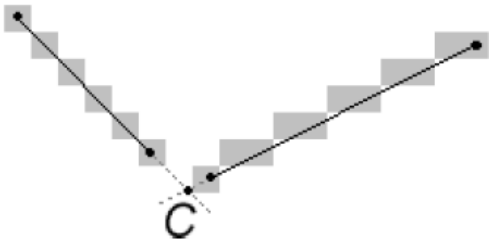
Sudėtingesnis linijų surinkimo metodas yra žinomas kaip *adaptyvus linijų surinkimas*. Šis metodas yra tikslesnis nei anksčiau aprašytas. Vietoj vieno grandinės dalinimo taško S čia yra naudojami du taškai S_- ir S_+. Šie taškai yra gretimi taškui S (34 pav.).



34 pav. Adaptyvaus linijų surinkimo privalumas. A – linijų surinkimas; B – adaptyvus linijų surinkimas.

Adaptyvus linijų surinkimo metodas gali suskaidyti linijas dėl to, kad išmeta tikruosius grandinės dalinimo taškus, kurie yra grandinės dalis. Sąjungos gali būti atstatytos projektuojant linijas į jų susikirtimus pvz.: 36 pav. vaizduoja linijų susikirtimo projekciją į tašką *C*.

35 pav. Adaptyvūs grandinės dalinimo taškai



36 pav. Atskirti linijos segmentai turi būti sujungti taške *C*

1.10.1.2 Linijų derinimas

Linijų derinimo (line blending) metu yra analizuojamas linijų sąrašas gautas pirmo žingsnio metu. Linijos, kurios tenkina *Lowe* reikšmingumo kriterijų, gali būti pakeistos ilgesnėmis. Žemiau pateiktas algoritmas aprašo linijų derinimo procesą.

1. Iš linijų sąrašo, gauto pirmojo žingsnio metu, pasirinkti visas gretimų linijų poras *PQ* ir QR.
2. Apskaičiuoti statmenį d_{PR} bei *Lowe* reikšmingumo atributą σ_{PR}.
3. Jeigu $d_{PR} \leq \varepsilon$ bei $w\sigma_{PR} < \sigma_{PQ}$ ir $w\sigma_{PR} < \sigma_{QR}$ kur $w \in (0,1]$, tuomet galima suderinti linijas *PQ* ir *PR* bei pakeičiant jas linija *PR*.
4. Vykdyti pirmąjį žingsnį tol kol nebesuderinama nei viena linija.

Parametras *w* yra vadinamas *derinimo svoriu* kontroliuoja linijų derinimą nuslopindamas originalių linijų, gautų pirmojo žingsnio metu, *Lowe* reikšmingumo atributų reikšmes.

37 pav. Linijų segmentai, gauti įvykužius antrąjį linijų derinimo algoritmo žingsnį

1.10.1.3 Lankų surinkimas

Lankų surinkimo procedūra (arc fitting) bando linijų, gautų antrojo žingsnio metu, poaibius paversti lankais taip sukurdamas sujungtų linijų ir lankų sąrašą. Kad rezultatas būtų patenkinamas pirmajame žingsnyje reikalinga žema kreivumo tolerancija ε, kadangi taip bus gauta daugiau linijų. Žemiau vartojamas terminas *linijų eilutė* reiškia sujungtų linijų sąrašą.

Lankų surinkimo metu linijų eilutė yra dalinama į dvi dalis, tol kol iš anksto žinomas kriterijus bus patenkintas. Kriterijus yra paremtas linijų, iš linijų eilutės, deviacija nuo įsivaizduojamo lanko. Žemiau pateiktas algoritmas yra taikomas kiekvienai antrame žingsnyje gautai linijų eilutei. Tarkime kad du galiniai linijų eilutės taškai yra S_1 ir S_n.

1. Pasukti linijų eilutę S_1S_n taip kad linija S_1S_n būtų statmena X ašiai. Tuomet perkelti linijų eilutę taip, kad linijos S_1S_n vidurinysis taškas M atsirastų koordinačių ašies pradžioje $[0, 0]$.
2. Surasti išlinkio centrą O. Tai atliekama remiantis dalinimo pusiau metodu. Tarkime kad $y_{min} = 0$, $y_{max} = \{y_i \mid S_i = [x_i, y_i], i = 2 .. n - 1\}$. Parametras δ yra iš anksto nustatyta reikšmė nusakanti dalinimo pusiau tikslumą.
    a. Tarkim $y = {}^1\!/_2 (y_{min} + y_{max})$, $O = [0, y]$, $O^+ = [0, y + \delta]$ ir $O^- = [0, y - \delta]$.
    b. Apskaičiuojam *radialinį nuokrypį* $\partial R^+ = \sum| R^+_1 + R^+_i|$ ir $\partial R^- = \sum| R^-_1 + R^-_i|$ $(i = 2 .. n)$ kur $R^+_i = ||O^+S_i||$ ir $R^-_i = ||O^-S_i||$.
    c. Jeigu $\partial R^+ \approx \partial R^-$ tuomet vykdome trečią žingsnį.
    d. Jeigu $\partial R^+ < \partial R^-$ tuomet nustatome $y_{max} = y$ ir kartojame iš pradžių (a).
    e. Nustatome $y_{min} = y$ ir kartojame iš pradžių (a).
3. Apskaičiuojame nuokrypį $d`$ tarp linijų eilutės $S_1S_n$ ir įsivaizduojamo lanko (centras $O$, spindulys $R = ||OS_1||$). $d`$ bus lygi ilgiausio statmens, tarp bent kurio linijos iš linijų eilutės taško ir bet kurio lanko taško, ilgiui .

4. Jeigu $d` > \varepsilon`$ tuomet daliname linijų eilutę taške $T = S_j$, kur $|R_1 - R_j| = max\{|R_1 - R_j| \mid j = 1 .. n\}$ ir taikome šį algoritmą linijoms S_1T bei TS_n. Jeigu $d` \leq \varepsilon`$ tai linijų eilutė laikoma lanku, kurio centras $O`$, o spindulys $R$. $O`$ yra apskaičiuojamas taikant inversinę pirmojo žingsnio transformaciją taškui O. Galiausiai lankas yra įdedamas į sąrašą kartu su *Lowe* reikšmingumo atributu, kuris apskai2iuojamas taip: $\sigma` = d`/w$, kur w yra lanko ilgis.

1.10.1.4 Lankų derinimas

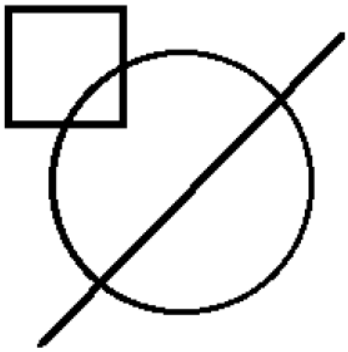
Lankų derinimo (arc blending) paskirtis yra tokia pati kaip ir linijų derinimo. Žemiau pateikiamas lankų derinimo algoritmas.

1. Iš lankų sąrašo, gauto trečiojo žingsnio metu, pasirinkti visas gretimų lankų poras PQ ir QR.
2. Apskaičiuoti nuokrypį $d`_{PR}$ bei *Lowe* reikšmingumo atributą $\sigma`_{PR}$.
3. Jeigu $d`_{PR} \leq \varepsilon$ bei $w`\sigma`_{PR} < \sigma`_{PQ}$ ir $w`\sigma`_{PR} < \sigma`_{QR}$ kur $w \in (0,1]$, tuomet galima suderinti lankus PQ ir PR pakeičiant juos lanku PR.
4. Vykdyti pirmąjį žingsnį tol kol nebesuderinamas nei vienas lankas.

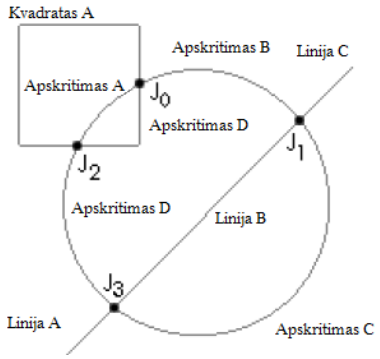
1.11 Segmentų sujungimas

Susikirtimų žymėjimo metu grandinės buvo išskaidytos. Kadangi objektų vientisumas yra labai svarbus tai gretimi segmentai, gauti segmentacijos proceso metu, turi būti sujungti. Segmentų sujungimas niekaip nekeičia *taškinio vaizdo grafo*.

Panagrinėkime paprastą pavyzdį (38 pav.) su trimis besikertančiais geometriniais objektais. Po susijungimų žymėjimo ir grandinės sekimo šie objektai yra išskaidomos (39 pav.).



38 pav. Originalus vaizdas

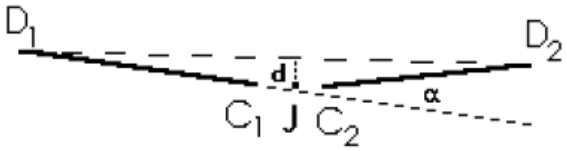


39 pav. Struktūriniai vaizdo komponentai

Aptarsime du segmentų sujungimo metodus. Teisingas segmentų sujungimo rezultatas pasiekiamas tik tuomet kai naudojami abu metodai. Pirmiau taikomas *perdengimas (spanning)* po kurio seka *sutraukimas (contracting)*. Segmentų sujungimo procedūra taikysime tik linijiniams segmentams.

1.11.1 Perdengimas

Tarkime, kad turime dvi gretimas linijas C_1D_1 ir C_2D_2, kurios yra gretimos susikirtimo tašui J. Gretimi šių linijų taškai bus C_1 ir C_2 (40 pav.).



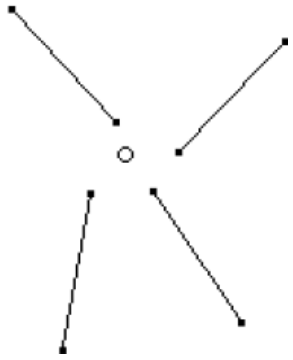
40 pav. Susikirtimo tašui J gretimos linijos C_1D_1 ir C_2D_2

Ši linijų pora yra sujungiama į *perdengiančią (span) liniją* D_1D_2, jeigu statmens d ir kampo α nuokrypiai yra mažesni už iš anksto nustatytus slenksčius. Žemiau yra aprašytas perdengimo algoritmas.

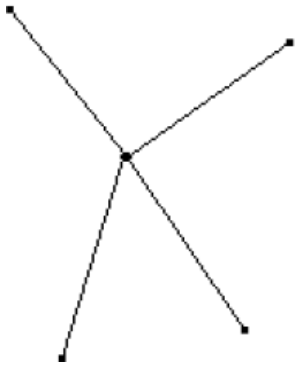
1. Tarkime $S_j = \emptyset$ (perdengiančių linijų aibė) bei vykdome antrąjį žingsnį kiekvienam susikirtimo taškui J bei jam gretimų linijų aibei L_j.
2. Sujungiam labiausiai gretimų linijų poras $C_1D_1 \in L_j$, $C_2D_2 \in L_j$ kurios tenkina statmens ir kampo nuokrypius. Į aibę S_j įdedame perdengiančią liniją D_1D_2 (tariant, kad susikirtimo taškui J gretimi taškai yra C_1 ir C_2) ir kartojame antrą žingsnį tol kol nebesukuriame nei viena nauja perdengianti linija.

1.11.2 Sutraukimas

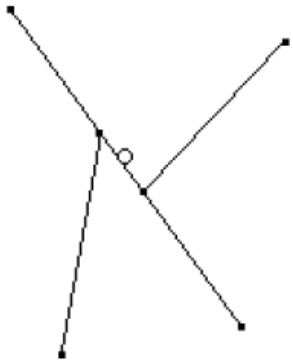
Išskirsime du sutraukimo (contracting) atvejus. Pirmuoju atveju, kai nerandama nei viena perdengianti linija arba kai perdengimas nebuvo naudojamas, linijos yra sujungiamos jų susikirtimo taškuose. Antruoju atveju, kai buvo rasta nors viena perdengianti linija likusios linijos yra pratęsiamos iki jų susikirtimo su perdengiančia linija.



41 pav. Originalus išdėstymas



42 pav. Sutraukimas be perdengimo



43 pav. Sutraukimas su perdengimu

Žemiau pateiktas algoritmas aprašo abu sutraukimo atvejus.

1. Kiekvienam susikirtimo taškui J vykdyti antrą žingsnį.
2. Jeigu $S_j = \emptyset$ tuomet vykdyti trečią žingsnį (pirmasis atvejis), kitu atveju (antrasis atvejis) kiekvienai linijai $CD \in L_j$, reikia surasti linijų CD ir $s \in S_j$ susikirtimo tašką P taip, kad atstumas $||PC||$ būtų minimalus (taškas C yra gretimas susikirtimo taškui J). Pakeisti liniją CD į liniją PD.
3. Pakeisti kiekvieną liniją $CD \in L_j$ į liniją JD (taškas C yra gretimas susikirtimo taškui J).

1.12 Išvados

1. Šiame skyriuje apžvelgtos problemos susijusios su taškinių vaizdų vektorizavimu.
2. Apžvelgti taškinių ir vektorinių vaizdų modeliai, jų privalumai bei trūkumai.
3. Vektorizavimo procesas susideda iš šių veiksmų: taškinio vaizdo ploninimo, susikirtimų žymėjimo, grandinės sekimo, susikirtimų sujungimo, grandinės apkirpimo, segmentacijos bei segmentų sujungimo.
4. Taškinio vaizdo ploninimas gali būti atliekamas dviem būdais: kontūrinių arba centrinių linijų išskyrimas. Ploninimo tikslas yra sumažinti taškinio vaizdo objektų plotį iki vieno vaizdo taško.
5. Susikirtimų žymėjimo funkcijos paskirtis yra garantuoti, kad grandinės sekimo metu bus galima identifikuoti kiekvieną sekantį vaizdo tašką.
6. Grandinės sekimo operacijos tikslas yra taškiniame vaizde išskirti vaizdo taškų atskiras grandines ir paruošti jas segmentacijai.
7. Susikirtimų sujungimo funkcija suranda susikirtimo taškų, kurie yra labai arti vienas kito, poras ir pakeičia juos vienu susikirtimo tašku.
8. Grandinės apkirpimas pašalina nepageidautinas deformacijas prie susikirtimo taškų.
9. Segmentacijos tikslas yra suskaidyti grandines į mažesnius prasmingus *segmentus*, kurie gali būti paverčiami į geometrines primityvas. Segmentaciją sudaro keturi žingsniai: linijų surinkimas, linijų derinimas, lankų surinkimas bei lankų derinimas.
10. Segmentų sujungimas sujungia objektų segmentus gautus segmentacijos metu. Sujungimo procedūrą sudaro du žingsniai: perdengimas ir sutraukimas.

2 PROJEKTINĖ DALIS

Magistratūros studijų metu sukurtos programinės įrangos techninė-projektinė dokumentacija. Joje pateikiamas pasirinkto sprendimo realizacijos kelias.

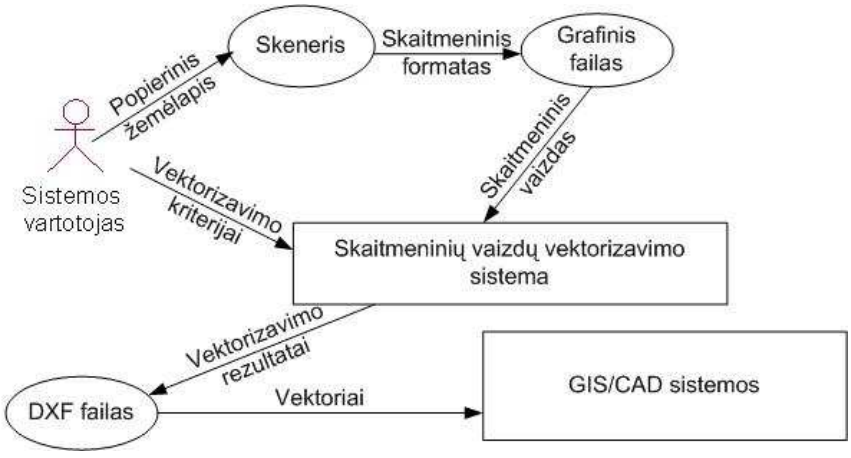
2.1 Sistemos apibūdinimas

2.1.1 Programų sistemos funkcijos

Sistema teikia šias funkcijas:

- Automatinis taškinių vaizdų vektorizavimas.
- Kontūrinių arba centrinių linijų išskyrimas.
- Taškinio vaizdo regiono vektorizavimas.
- Vektorizavimo rezultatų redagavimas realiu laiku.
- Vektorizavimo rezultatų išsaugojimas DXF formate.
- Taškinio vaizdo apdorojimas (kraštų radimo, triukšmo pašalinimo ir kiti filtrai) bei redagavimas (veidrodžio efektas, vaizdo pasukimas ir pan.).
- Palaikomi populiariausi taškinių vaizdo failų formatai (.tif, .tiff, .bmp, .jpg, .jpeg, .pcx, .pct, .png, ir kt.).

2.1.2 Sistemos kontekstas



44 pav. Sistemos kontekstas

Sistema duomenis gauna iš grafinio failo, kuris savo ruožtu yra sukuriamas skenavimo metu. Sistemos sukurti rezultatai (vektoriai), gali būti panaudoti kompiuterinio dizaino (CAD) sistemose (pvz. *Autodesk* šeimos produktai.) tolimesniam apdorojimui.

Veiklos padalinimas:

1 lentelė. Sistemos resursai

Eil. Nr.	Įvykio pavadinimas	Įvesties / Išvesties informacijos srautai
1	Sistemos vartotojas skenuoja popierinį žemėlapį	Vaizdas popieriniame formate (in) Taškinis vaizdas (Grafinis failas) (out)
2	Vektorizuojamas taškinis vaizdas	Taškinis vaizdas (in) Vektorizavimo kriterijai (in) Vaizdas vektoriniame formate (out)
3	Saugojimas į DXF formatą	Vaizdas vektoriniame formate (in) DXF formato failas (out)

2.1.3 Vartotojo problemos

Sukurtas produktas palengvina GIS/CAD specialistų darbą šiais aspektais:

- Sumažina taškinių vaizdų vektorizavimo laiko sąnaudas.
- Leidžia minimaliai redaguoti vektorizavimo rezultatus (vektorius) realiu laiku, nenaudojant didelių GIS/CAD sistemų.

2.2 Architektūros pateikimas

Šis dokumentas pateikia architektūrą keliais skirtingais požiūriais į sistemą : panaudojimo atvejų požiūris, loginis požiūris, procesų požiūris, paskirstymo požiūris. Šie požiūriai pateikti naudojant *Enterprise Architect* programinę įrangą.

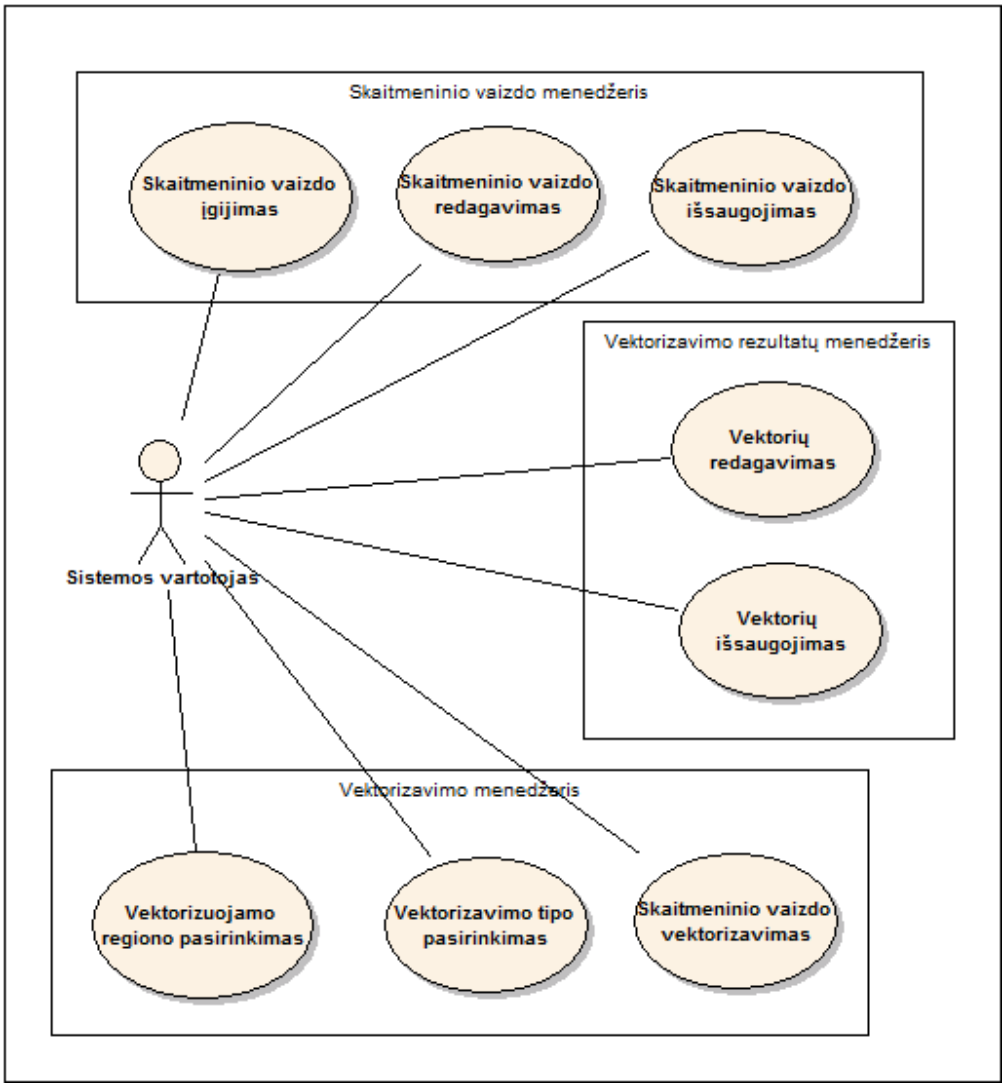
2.2.1 Architektūros tikslai ir apribojimai

Sprendimai buvo priimti remiantis šiais apribojimais:

1. Sistema kuriama naudojant *C++* programavimo kalbą.
2. Sistemos komponentai bendraus naudojant *COM* objektus.
3. Sistema veiks tik su *Windows 2000/XP* operacine sistema.
4. Sistema kuriama norint apsiginti KTU magistrinį darbą.

5. Į sistemą turi įeiti tokie nepriklausomi moduliai : taškinio vaizdo, vektorizavimo, vektorizavimo rezultatų (vektorių) bei vartotojo sąsajos.
6. Vartotojo sąsaja bus kuriamas naudojant *MFC (Microsoft Foundation Classes).*

2.2.2 Panaudojimo atvejų požiūris



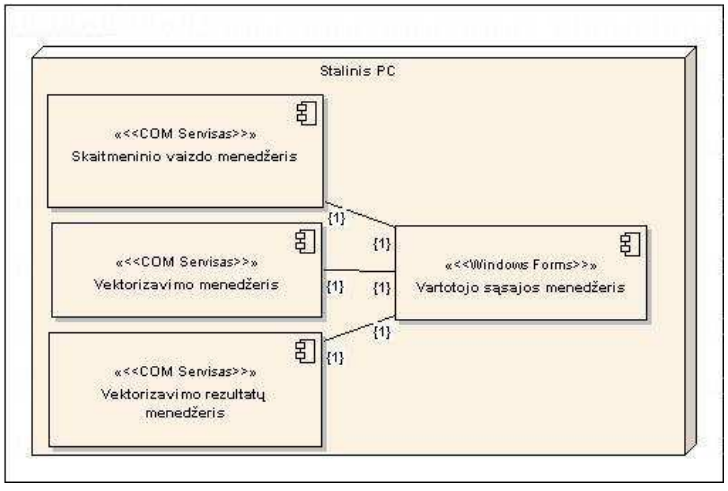
45 pav. Architektūrai reikšmingi panaudojimo atvejai

Detalizuotus panaudojimo atvejus galima rasti reikalavimų specifikacijoje.

1. *Taškinio vaizdo įgijimas. Š*is panaudojimo atvejis leidžia vartotojams sukurti norimo formato grafinį failą taškinio vaizdo skenavimo metu, bei jį atsidaryti.
2. *Taškinio vaizdo išsaugojimas* – šis panaudojimo atvejis leidžia vartotojams išsaugoti pakoreguotą taškinį vaizdą į norimo formato grafinį failą.

3. *Taškinio vaizdo redagavimas* – šis panaudojimo atvejis leidžia vartotojams koreguoti taškinį vaizdą taikant linijinius bei nelinijinius filtrus, arba keičiant spalvų gamą.
4. *Taškinio vaizdo vektorizavimas.* Šis panaudojimo atvejis apima taškinio vaizdo vektorizavimo procesą, kurio metu yra sukuriami vektorizavimo rezultatai (vektoriai).
5. *Vektorizavimo tipo pasirinkimas.* Suteikia galimybę vartotojams pasirinkti norimą vektorizavimo metodą. Galima rinktis centrinių arba kontūrinių linijų vektorizavimą bei kitus kriterijus.
6. *Vektorizuojamo regiono pasirinkimas*. Šis panaudojimo atvejis suteikia galimybę vartotojui pasirinkti taškinio vaizdo regioną, kuris bus vektorizuojamas.
7. *Vektorių redagavimas.* Šis panaudojimo atvejis suteikia galimybę vartotojui redaguoti vektorizavimo rezultatus (vektorius), keičiant jų spalvą, sluoksnį ar taškų koordinates.
8. *Vektorių Išsaugojimas.* Šis panaudojimo atvejis suteikia galimybę vartotojui išsaugoti vektorizavimo rezultatus (vektorius) DXF formato faile.

2.2.3 Paskirstymo požiūris



46 pav. Paskirstymo požiūris

Stalinis *PC* – tai vartotojo kompiuteris.

Sistemoje išskiriami tokie komponentai:

Taškinio vaizdo menedžeris - šis komponentas yra atsakingas už panaudojimo atvejų susijusių su taškinio vaizdo apdorojimu įgyvendinimą:

- Taškinio vaizdo įgijimas.
- Taškinio vaizdo išsaugojimas.
- Taškinio vaizdo koregavimas.

Vektorizavimo menedžeris – šis komponentas yra atsakingas už panaudojimo atvejų susijusių su taškinio vaizdo vektorizavimu įgyvendinimą:

- Taškinio vaizdo vektorizavimas.
- Vektorizavimo nustatymų pasirinkimas.

Vektorizavimo rezultatų menedžeris - šis komponentas yra atsakingas už panaudojimo atvejų susijusių su vektorizavimo rezultatais (vektoriais) įgyvendinimą:

- Vektorių redagavimas.
- Vektorių išsaugojimas.

Vartotojo sąsajos menedžeris – tai nepriklausomas sistemos komponentas, kuris atsakingas už vartotojo sąsają. Jo pagalba vartotojas gali vykdyti visas sistemos funkcijas.

2.3 Išvados

1. Šiame skyriuje apžvelgti magistratūros studijų metu sukurtos programinės įrangos, gebančios vektorizuoti taškinius vaizdus, techninės-projektinės dokumentacijos esminiai aspektai.
2. PĮ gebanti vektorizuoti taškinius vaizdus realizuoja 7 pagrindinius panaudojimo atvejus: taškinio vaizdo įgijimas, taškinio vaizdo išsaugojimas, taškinio vaizdo redagavimas, taškinio vaizdo vektorizavimas, vektorizavimo tipo pasirinkimas, vektorizuojamo regiono pasirinkimas, vektorių redagavimas bei vektorių išsaugojimas.
3. PĮ gebančią vektorizuoti taškinius vaizdus sudaro keturi komponentai: taškinio vaizdo menedžeris, vektorizavimo menedžeris, vektorizavimo rezultatų menedžeris bei vartotojo sąsajos menedžeris.

3 TYRIMO DALIS

Kadangi, programų sistemoje naudojamas ploninimo algoritmas atlieka labai daug skaičiavimų, vektorizavimo procesas užima nemažai laiko. Be to rezultate lieka nemažai ploninimo algoritmo veikimo metu atsiradusių šiukšlių. Bandysime pakeisti algoritmą siekiant sumažinti laiko sąnaudas bei pagerinti rezultato kokybę.

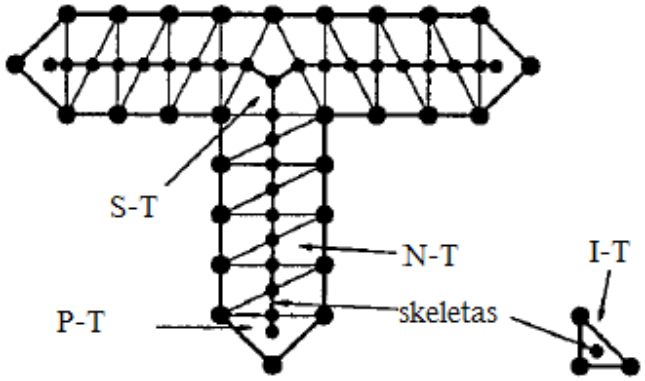
Šiame skyriuje apžvelgiamas ploninimo algoritmas neparemtas vaizdo taškais. Algoritme yra naudojama Delaunay trianguliacija. Taškinis vaizdas yra padalinamos į aibę nepersidengiančių trikampių. Tuomet tam tikri trikampiai yra sujungiami pašalinant šiukšles. Figūros skeletas gaunamas sujungiant jos sudedamųjų dalių (trikampių) vidurio taškus. Eksperimento rezultatai rodo, jog šis algoritmas yra kur kas greitesnis, nei tipinis ploninimo algoritmas Zhang and Suen [6].

3.1 Trianguliacija

Taškiniam vaizdui sukuriama Delaunay trianguliaciją. Tarkime, kad objektų kontūrai yra poligonai, kurie savo ruoštu gali būti aprašyti naudojant tiesios linijos grafą (plokštumoje) $G = (E, V)$, kur kiekviena viršūnė yra kontūro vaizdo taškas, o kiekviena briauna atitinka kontūro segmentą. Grafo G trianguliacija yra lygi $G' = (V, E')$, kur $E \subseteq E'$ taip, kad nei viena briauna, jungianti dvi viršūnes iš aibės V ir kertanti jau egzistuojančią briauną aibėje G', nebūtų įdėta į G'. Grafo G Delaunay trianguliacija (DT) yra tokia aibės G trianguliacija, kurioje apie kiekvieną trikampį apibrėžti apskritimai savo viduje neturi tokių grafo G viršūnių, kurios yra matomos iš trikampio viršūnėms. Dvi grafo viršūnes yra matomos viena iš kitos jeigu jas jungianti tiesi linija nekerta grafo briaunos.

Taškinio vaizdo Delaunay trianguliacija sudaro *vidiniai* ir *išoriniai* trikampiai. *Vidiniai* trikampiai dengia tik objektų vaizdo taškus, o *išoriniai* – taškinio vaizdo foną (vaizdo taškus lygius 0). *Vidinio* trikampio briauna vadinama *išorine briauna* tuomet kai ji yra kontūro segmentas, kitu atveju ji vadinama – *vidine briauna*. Kiekviena *vidinio* trikampio *vidinė briauna* reiškia, kad šalia jos esantis trikampis taip pat yra *vidinis*. Yra keturi vidinių trikampių tipai (47 pav.).

1. Vidinis trikampis, kuris neturi nei vienos vidinės briaunos yra vadinamas *izoliuotu trikampiu (I-T)*.
2. Vidinis trikampis turintis tik vieną vidinę briauną reiškia šakos pabaigą, kadangi jis jungiasi tik su vienu vidiniu trikampiu. Toks trikampis vadinamas *pabaigos trikampiu (P-T)*.
3. *Normaliu trikampiu (N-T)* vadinamas toks vidinis trikampis, kuris turi dvi vidines briaunas. Toks trikampis jungiasi su dviem vidiniais trikampiais.
4. Vidinis trikampis turintis tris vidines briaunas vadinamas *susikirtimo trikampiu (S-T)*. Toks trikampis yra trijų šakų santaka.



47 pav. Vidiniai Delaunay trianguliacijos trikampiai bei jų skeletai. I-T – izoliuotas trikampis; S-T – susikirtimo trikampis; N-T – normalus trikampis; P-T – pabaigos trikampis.

3.2 Ploninimas

Po trianguliacijos nubrėžimo kiekvienas taškinio vaizdo objektas tampa padalintas į nepersikertančius trikampius. Iš to seka, kad objekto skeletas gali būti gautas iš jį sudarančių vidinių trianguliacijos trikampių skeletų (47 pav.). *Pabaigos trikampių (P-T)* skeletas yra tiesi linija jungianti trikampio centroidą su viduriniuoju trikampio vidinės briaunos tašku. *Normaliųjų trikampių (N-T)* skeletas yra tiesi linija jungianti dviejų trikampio vidinių briaunų vidurio taškus. *Susikirtimo trikampių (S-T)* skeletas yra trys tiesios linijos jungiančios trikampio centroidą su vienos trikampio vidinės briaunos vidurio tašku. *Izoliuotų trikampių (I-T)* skeletas yra jų centroidas.

3.3 Šiukšlių šalinimas

Figūros skeletas gautas iš Delaunay trianguliacijos gali turėti šiukšlių, kurios skirstomos į dvi grupes: pakraščių ir susikirtimų šiukšles.

Pakraštinė skeleto šaka yra ribojama galinio ir susikirtimo taškų. Tokia šaka yra vadinama *pakraščio šiukšle*, tuomet kai ji yra nereikšminė objekto šaka. Išsikišimo reikšmingumas yra išsikišimo ilgio ir pločio santykis: $\gamma = l/w$. Jeigu $\gamma < \gamma_{th}$ tai pakraštinė skeleto šaka yra pašalinama, o ją sudarantys trikampiai – sujungiami. Rekomenduojama γ_{th} reikšmė yra *0,5*.

Objektų susikirtimai, po trianguliacijos, dažniausiai yra padalinami į du ar daugiau *susikirtimo trikampių*. Klaidinga skeleto šaka yra sukuriama tarp dviejų *susikirtimo trikampių* centroidų. Tokios klaidingos skeleto šakos yra vadinamos *susikirtimų šiukšlėmis*. Jos yra aptinkamos lyginant susikertančių šakų orientacijas, o pašalinamos sujungiant susijusius trikampius.

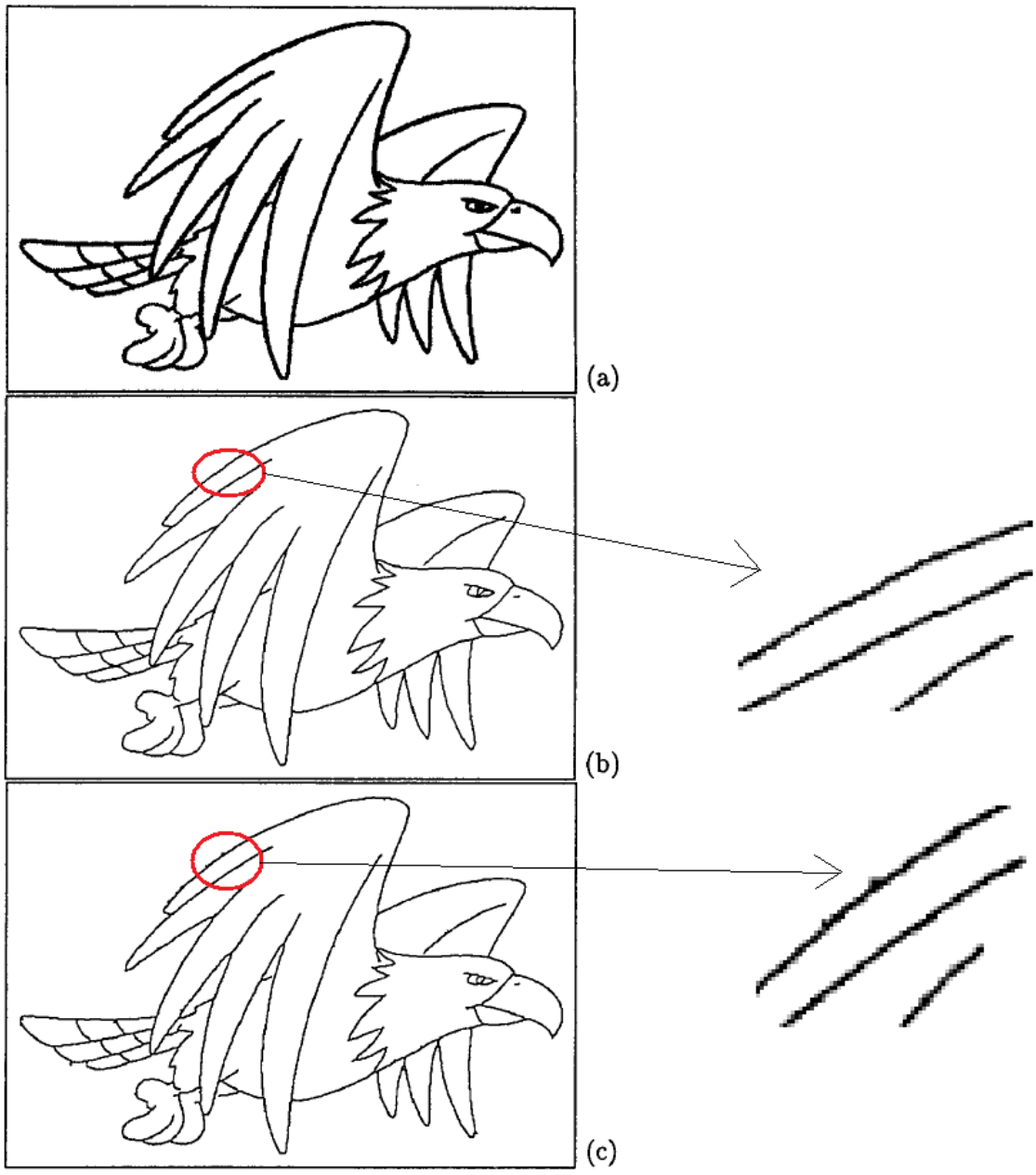
3.1 Algoritmas

Pasiūlytas algoritmas susideda iš šių žingsnių:

1. Pasirenkamas taškinis vaizdas.
2. Nustatomas objektų pločio mediana – apskaičiuojant objektų pločio medianą *w*, taškinis vaizdas yra „skaitomas“ horizontaliai ir vertikaliai. Tuomet sudaroma skaitymo trukmės histograma. *w* reikšmė bus maksimali histogramos reikšmė..
3. Išskiriami kontūrai bei jiems sudaromi poligonai – Kiekvieno objekto *S* kontūrui yra sudaromas poligonas, kurio visos briaunos, išskyrus vieną, yra lygios $l = 0,3w$, kur *w* yra objekto *S* pločio mediana. Vienos poligono kraštinės ilgis bus trumpesnis..
4. Sudaroma Delaunay trianguliacija kontūrams (skyrelis 3.1).
5. Pašalinamos šiukšlės (skyrelis 0).
6. Išskiriami skeletai (skyrelis 3.2).

3.2 Veikimo rezultatai

Pasiūlytas ploninimo algoritmas duoda švaresnį rezultatą nei tipinis ploninimo algoritmas (48 pav.). Paveikslėlyje matome, jog naudojant tipinį ploninimo algoritmą, suplonintas vaizdas turi nereikalingų objektinių vaizdo taškų. Tokios šiukšlės apsunkina tolimesnį vektorizavimo procesą (vektorių išskyrimą iš suploninto vaizdo), tuo pačiu padidėja tikimybė jog galutinis vektorizavimo rezultatas turės daugiau nereikalingų vektorių.

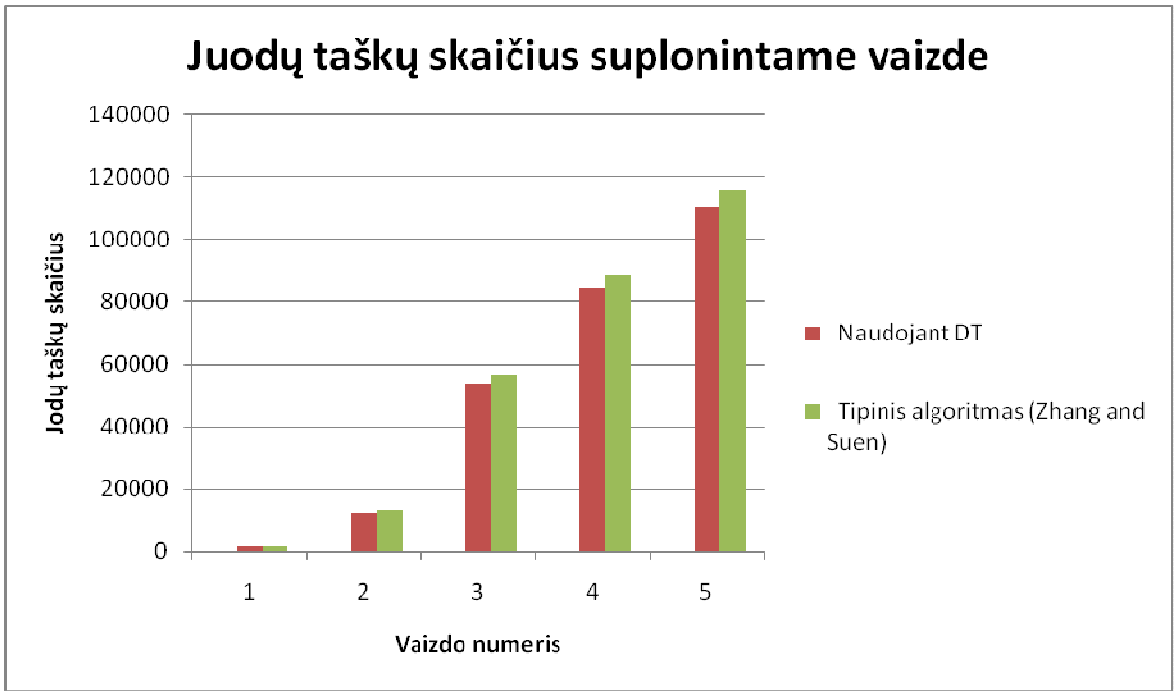


48 pav. (a) Pradinis vaizdas (b) Ploninimas naudojant Delaunay trianguliaciją (c) Ploninimas naudojant tipinį algoritmą (Zhang and Suen)

Žemiau esančioje lentelėje yra pateikti tipinio ploninimo algoritmo ir Delaunay trianguliacija paremto algoritmo apdorotuose taškiniuose vaizduose likusių objektinių vaizdo taškų skaičiai. Tyrime naudoti pirmieji penki eksperimentiniai paveiksliukai pateikti priede.

2 lentelė. Objektinių taškų skaičiai suplonintuose taškiniuose vaizduose

Taškinio vaizdo numeris	Objektinių taškų skaičius	
	Naudojant DT	Tipinis algoritmas (Zhang and Suen)
1	1569	1647
2	12541	13168
3	53917	56612
4	84693	88927
5	110405	115925

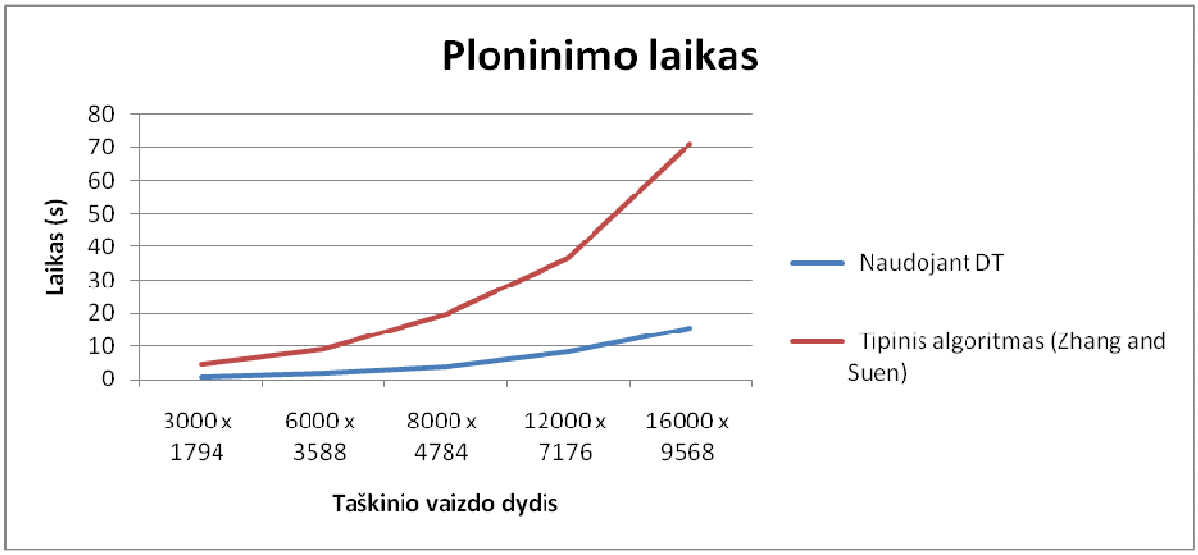


49 pav. Objektinių vaizdo taškų skaičiai suplonintame vaizde

Tyrimo metu nustatyta, kad pasiūlytas algoritmas veikia apie ~75% greičiau už tipinį algoritmą (3 lentelė).

3 lentelė. Algoritmų veikimo laikas

Taškinio vaizdo Dydis (vaizdo taškais)	Veikimo laikas (sek)	
	Naudojant DT	Tipinis algoritmas (Zhang and Suen)
3000 x 1794	0,9	4,5
6000 x 3588	2,1	8,89
8000 x 4784	4,0	19,6
12000 x 7176	8,2	36,6
16000 x 9568	15,4	71,26



50 pav. Algoritmų veikimo laikas

3.3 Išvados

1. Šiame skyriuje pasiūlytas naujas taškinių vaizdų ploninimo algoritmas paremtas Delaunay trianguliacija.
2. Pasiūlytas ploninimo algoritmas duoda švaresnį rezultatą nei tipinis ploninimo algoritmas. Tai įrodo mažesnis objektinių vaizdo taškų kiekis suplonintame vaizde. Rezultatų Lyginimas buvo atliekant ploninant identiškus taškinius vaizdus.
3. Palyginus veikimo laikus naudojant tą patį, skirtingų dydžių, taškinį vaizdą (57 pav.) nustatyta, kad algoritmas naudojantis Delaunay trianguliaciją veikia ~75% sparčiau už tipinį Zhang ir Suen pasiūlytą algoritmą.

4 EKSPERIMENTINĖ DALIS

4.1 Vektorizavimo metodai

Šiame skyriuje aprašytas vektorizavimo metodų eksperimentinis tyrimas. Egzistuojantys vektorizavimo (grafinių objektų atpažinimo) metodai gali būti skirstom i į vieno ir dviejų žingsnių metodus. Pirmos klasės metodai objektus atpažįsta tiesiai iš taškinio vaizdo, kai tuo tarpu antroji iš pradžių paverčia taškinį vaizdą į vektorinį formatą, o vėliau, taikant vektorinius algoritmus, atpažįsta objektus. Kiekviena iš šių klasių turi savo pliusų ir minusų.

4.1.1 Dviejų žingsnių metodai

Pirmasis šio tipo metodų etapas yra taškinio vaizdo pavertimas į vektorinį formatą (vektorizavimas). Šio etapo rezultatas yra trumpi vektoriai, kurie atitinka taškinio vaizdo objektus kaip galima tiksliau. Tačiau šie vektoriai neatitinka tikrųjų grafinių objektų tipo [8]. Dėl šios priežasties yra reikalingas papildomas gautų vektorių apdorojimas.

4.1.1.1 Vektorizavimas

Beveik visi dviejų žingsnių metodai taškinį vaizdą vektorizuoja remdamiesi kokia nors ploninimo koncepcija. Vėliau yra pritaikomi segmentacijos metodai bei gaunamos linijų bei lankų atkarpos [9].

Lam pasiūlė ploninimo metodus skirstyti į dvi kategorijas: iteracinius ir neiteracinius. Pirmasis metodų tipas yra lengvai realizuojamas, tačiau skaičiavimai užima daug laiko. Be to pirmojo tipo metodai sukelia deformacijas susikirtimo taškuose, yra jautrūs triukšmui, bei neišsaugo linijų storio. Antrojo tipo metodai apima Delaunay trianguliaciją (*DT*) naudojančius, kontūrais paremtus, bei grafo struktūra paremtus metodus. *DT* paremti metodai yra žymiai greitesni už iteracinius. Pagrindinis *DT* metodų trūkumas yra iškraipymu atsiradimas susikirtimo taškuose. Kontūrais paremti metodai išgaunant ašines objektų linijas lygina jų kontūrų poras, todėl jiems tereikia tik vieno perėjimo. Šie metodai yra labai greiti ir nesukelia iškraipymų. Tačiau konkūrų lyginimas, daugeliu atvejų, yra labai sudėtingas procesas [9]. Grafo struktūra paremti metodai naudoja grafus, kurie atspindi linijinių figūrų bei jų ryšius originaliame vaizde. Pats populiariausias metodas sudarant tokį grafą yra skaitymo ilgio šifravimas *(Run Length Encoding RLE*). Šie metodai gerai veikia su mažai linijų, kurios pagrinde yra horizontalios ir vertikalios, turinčiais vaizdais. Pagrindinis šių metodų minusas yra tai, kad rezultato kokybė stipriai priklauso nuo skaitymo krypties.

Išmėtytų vaizdo taškų sekimo metodas (*Sparse pixel tracking*) yra žymiai greitesnis už anksčiau paminėtus todėl, kad jis palanko tik mažą dalį vaizdo taškų. Tačiau šis metodas turi pagrindinius du trūkumus. Pirmasis tai, kad sekimas būna nutraukiamas tuomet kai susikirtimų skaičius yra didesnis negu iš anksto numatytas. Antrasis trūkumas yra tas, kad taikant šį metodą yra sunku išlaikyti objektų vientisumą.

4.1.1.2 Tolimesnis apdorojimas

Tolimesnio apdorojimo metu turi būti išspręstos visos problemos atsiradusios vektorizavus taškinį vaizdą. Problemos gali būti tokios: objektų vientisumo atstatymas, susikirtimų, galo taškų, tekstų segmentavimas. Šiame etape dažniausiai yra naudojami linijų bei lankų surinkimo ir derinimo metodai. Šių metodų rezultatas nėra labai tikslus, kadangi jie paremti atskirų linijų segmentų sujungimu. Yra metodų paremtų transformacija *Hough Transform (HT)*, šie metodai nėra jautrūs vidutiniams linijos iškraipymams ar duomenų trūkumui. Didžiausias *HT* paremtų metodų trūkumas yra tas, kad jie atlieka daug skaičiavimų ir reikalauja didelio atminties kiekio.

4.1.1 Vieno žingsnio metodai

Kovalevsky yra pasiūlęs vaizdo taškų sekimo algoritmą tiesioms linijoms ir lankams išskirti. Jis pasirenka siauriausią ruožą kaip sekimo kryptį, algoritmo rezultato neįtakoja susikirtimai ir triukšmai, jeigu pirmasis sekimo krypties pasirinkimas yra teisingas [13].

HT metodas yra taip pat naudojamas tiesioms linijoms, apskritimams bei elipsėms išskirti tiesiai iš taškinio vaizdo. Didžiausias *HT* metodų privalumas yra tai, kad jie gali išskirti norimus objektus iš gan triukšmingo taškinio vaizdo. Šio tipo metodai dažniausiai yra taikomi mažo dydžio vaizdams pvz.: kraujo ląstelių mikroskopiniams vaizdams.

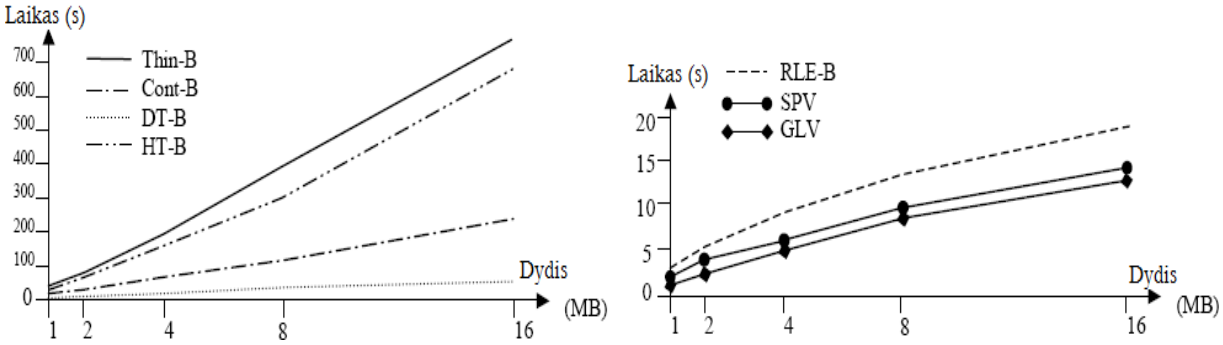
Chiang pasiūlė regionu paremtą tiesių linijų išskyrimo tiesiai iš taškinio vaizdo metodą. Šis metodas naudoja didžiausią įbrėžtinį apskritimą aptinkant tiesios linijos charakteristikas. Metodas išskiria linijas kartu su susikirtimo taškais, taip išvengiant iškraipymų tuose taškuose. Metodo trūkumas tas, kad jis nėra tikslus, bei negali išskirti lankų bei brūkšninių linijų [10].

Naujausias vieno žingsnio metodas yra globalios linijos metodas atpažįstantis tiesias linijas (vientisas ir brūkšniuotas), lankus ir apskritimus. Šis metodas yra labai greitas kadangi taškinis vaizdas yra palaipsniui supaprastinamas. Taipogi yra atstatomas išskirtų linijų plotis. Metodo trūkumas tas, kad jis ne visada atpažįsta trumpas linijas [11].

4.1.1 Eksperimentiniai rezultatai

Nors visi dviejų žingsnių metodai vykdo vektorizavimą bei tolimesnius apdorojimo veiksmus jų skaičiavimo sudėtingumai yra skirtingi. Buvo palyginti penkių dviejų žingsnių bei dviejų vieno žingsnio metodų veikimo laikai. Eksperimente buvo vykdomas tik pirmasis dviejų žingsnių metodų etapas – vektorizavimas. Testavimo duomenims buvo naudojami penki skenuoti brėžiniai, kurių dydis kinta nuo A4 (1 MB) iki A0 (16 MB). Eksperimentas buvo atliekamas naudojant AMD Athlon 64 1800+ procesorių bei 1024 MB RAM. Kiekvieno metodo veikimo laikai yra pavaizduoti 51 pav. Kairiame grafike yra vaizduojami metodų, kurie yra imlūs laikui, veikimo laikai, dešiniajame grafike – metodų, kurie nėra labai imlūs laikui. Žemiau esančiose lentelėse ir paveikslėliuose naudojami tokie žymėjimai:

- *Thin-B* – iteraciniu ploninimu paremtas metodas.
- *DT-B* – Delaunay trianguliacija (*DT)* paremtas metodas.
- *Cont-B* – kontūrais paremtas metodas.
- *RLE-B* – grafo struktūra paremtas metodas (*RLE* skaitymo kryptis - horizontali).
- *HT-B* – tiesios linijos išskyrimo *HT* metodas.
- *SPV* – išmėtytų vaizdo taškų sekimo metodas.
- *GLV* – globalios linijos vektorizavimo metodas.



51 pav. Vektorizavimo metodų veikimo laikai

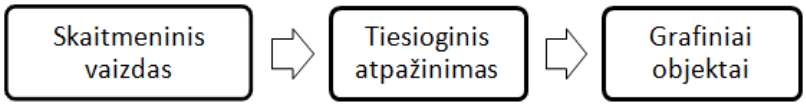
Iš 51 pav. matome jog ilgiausiai veikia iteracinis ploninimo metodas bei *HT* paremtas metodas. Kontūrais ir *DT* paremti metodai veikia greičiau todėl, kad nėra iteraciniai. *RLE* metodas veikia greičiausiai tačiau jis vykdo tik horizontalų skaitymą, todėl horizontalios (arba beveik horizontalios) linijos nėra gerai išskiriamos. Greičiausias iš dviejų žingsnių metodų yra išmėtytų vaizdo taškų sekimo (*SPV*) metodas, kadangi jis aplanko tik mažą vaizdo taškų dalį. *GLV* (globalios linijos vektorizavimo metodas) yra pats greičiausias dėl keturių priežasčių:

1. Jis neaplanko visų vaizdo taškų.
2. Sekimo kelias sudaromas naudojant Bresenham algoritmą.
3. Taškinis vaizdas yra palaipsniui supaprastinamas.
4. Metodas negeneruoja labai daug viduriniosios ašies vektorių, kaip tai daro dviejų žingsnių metodai.

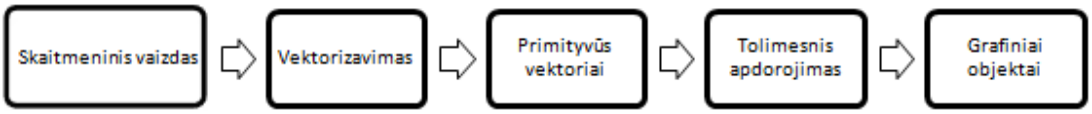
Iteracinio ploninimo bei kontūrais paremtų metodų laiko kreivės yra tiesinės kadangi yra aplankomi visi vaizdo taškai. *HT* metodo laiko kreivė yra tiesinė tuomet kai apdorojamo taškinio vaizdo dydis kinta nuo 1 MB iki 8 MB, tačiau kai dydis pasiekia 16MB veikimo laikas žymiai pailgėja, kadangi labai išauga saugomų reikšmių, reikalingų skaičiavimams, skaičius. *DT, RLE, SPV* bei *GLV* metodų veikimo laikų kreivės nėra linijinės, jos priklauso nuo linijų skaičiaus.

4.1.1 Apibendrinimas

Šiame skyrelyje yra pateikti vieno ir dviejų žingsnių metodų pliusai ir minusai bei veikimo principai.



52 pav. Vieno žingsnio vektorizavimo metodų veikimo principas



53 pav. Dviejų žingsnių vektorizavimo metodų veikimo principas

Vieno žingsnio metodų pliusai:

- Atpažįsta pilną objekto geometriją, nepažeistą susikirtimo taškuose.
- Nejautrūs triukšmams.
- Gerai apdoroja susikirtimo taškus.

Vieno žingsnio metodų minusai:

- Neatskiria didelio spindulio lankų nuo apskritimų.
- Neišskiria per trumpų/plonų linijų bei lankų.
- Gerai apdoroja susikirtimo taškus

Dviejų žingsnio metodų pliusai:

- Atpažįsta visų tipų geometrinius objektus.
- Nejautrūs linijos ilgiui/pločiui.
- Gerai išskiria izoliuotus objektus.

Dviejų žingsnio metodų minusai:

- Vektorizavimas sukelia iškraipymus susikirtimo taškuose ir triukšmingose vietose.
- Vektorizavimo rezultatų apdorojimas užima daug laiko.

4.2 Vektorizavimo sistemos

Šiame skyriuje aprašytas sukurtos sistemos eksperimentinis tyrimas. Eksperimentuose naudojami tie patys taškiniai vaizdai, kurie buvo naudoti vektorizavimo metodų tyrime. Visi gauti rezultatai pateikti lentelių bei diagramų forma.

Bus tiriamos trys vektorizavimo sistemos, skirtos tik taškiniams vaizdams vektorizuoti:

1. *WiseImage* 8.2
2. *Vextractor* 3.61
3. *Ras2Vec* 1.2
4. *Vektorizavimas* (mūsų sistema)

Žemiau esančioje lentelėje pateikiami sistemų gamintojai bei jų kainos:

4 lentelė. Vektorizavimo sistemų gamintojai ir kainos

Sistema	Gamintojas	Kaina (Lt)
WiseImage	Consistent software (www.easytrace.com)	3500
Vextractor	VextraSoft (www.vextrasoft.com/vextractor.htm)	250
Ras2Vec	Davide Libenzi (http://xmailserver.org/davide.html)	Nemokamas
Vektorizavimas	Vaidas Velutis (http://soften.ktu.lt/~s66927/Vektorizavimas.zip)	1200

Eksperimento metu bus lyginami tokie vektorizavimo metu gauti rezultatai: vektorizavimo laikas, vektorių skaičius bei verteksų skaičius.

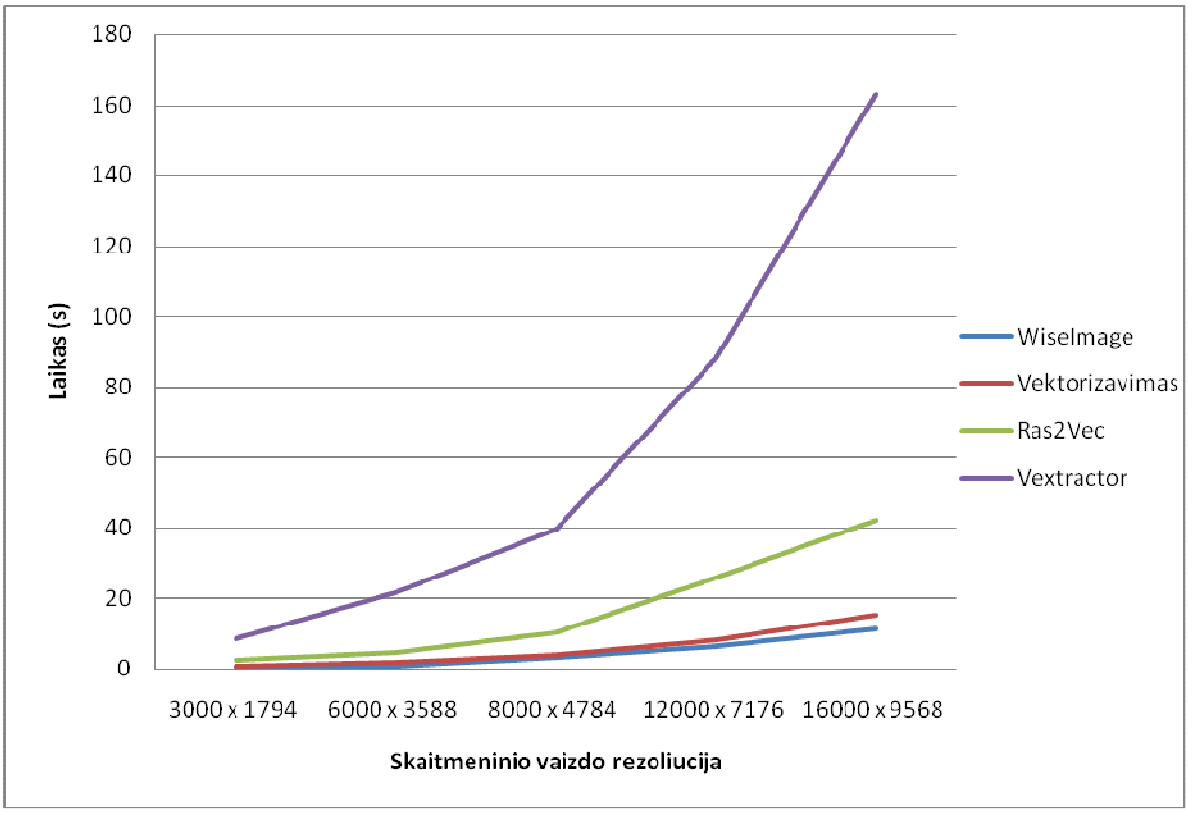
Sistemų *WiseImage, Vextractor* bei *Ras2Vec* naudojami vektorizavimo metodai nėra žinomi. Vektorizuojant vaizdus visose sistemose buvo išskiriamos centrinės objektų linijos, naudojant tokius pačius parametrus.

4.2.1 Vektorizavimo laikas

Eksperimento duomenims buvo naudojamas vienas skirtingų rezoliucijų taškinis vaizdas (57 pav. **Eksperimento paveiksliukas nr. 1**). Šio eksperimento metu yra tiriama kaip sistemos susidoroja su dideliais duomenų kiekiais. Žemiau esančioje lentelėje yra pateikti vektorizavimo laikai. Laikas lentelėje pateikiamas sekundėmis.

5 lentelė. Vektorizavimo sistemų veikimo laikai

Vaizdo nr./Sistema	*WiseImage*	*Vextractor*	*Ras2Vec*	*Vektorizavimas*
3000 x 1794	0,5	8,6	2,5	0,9
6000 x 3588	1,0	21,6	4,6	2,1
8000 x 4784	3,2	39,7	10,3	4,0
12000 x 7176	6,2	88,1	25,9	8,2
16000 x 9568	11,2	162,9	42,3	15,4



54 pav. Vektorizavimo sistemų veikimo laikai

Iš lentelės (5 lentelė) bei grafiko (54 pav.) matome, kad greičiausiai vektorizavimo procesą įvykdė *WiseImage* bei *Vektorizavimas*. Mūsų sistemos greitą veikimą galima pagrįsti tuo, jog ji vektorizavimą vykdo naudodama Delaunay trianguliacija pagrįstą ploninimo metodą, šis metodas aprašytas tyrimo dalyje. trečioje vietoje esančios nemokamos sistemos *Ras2Vec* viena iš pagrindinių jos greito veikimo priežasčių yra tai, kad ji neturi grafinės vartotojo sąsajos, o visos komandos vykdomos per konsolę. Paskutinėje vietoje liko *Vextractor* sistema, tai gali būti todėl, kad jos generuojamų vektorių skaičius, kaip pamatysime tolimesniuose eksperimentuose, yra pats didžiausias.

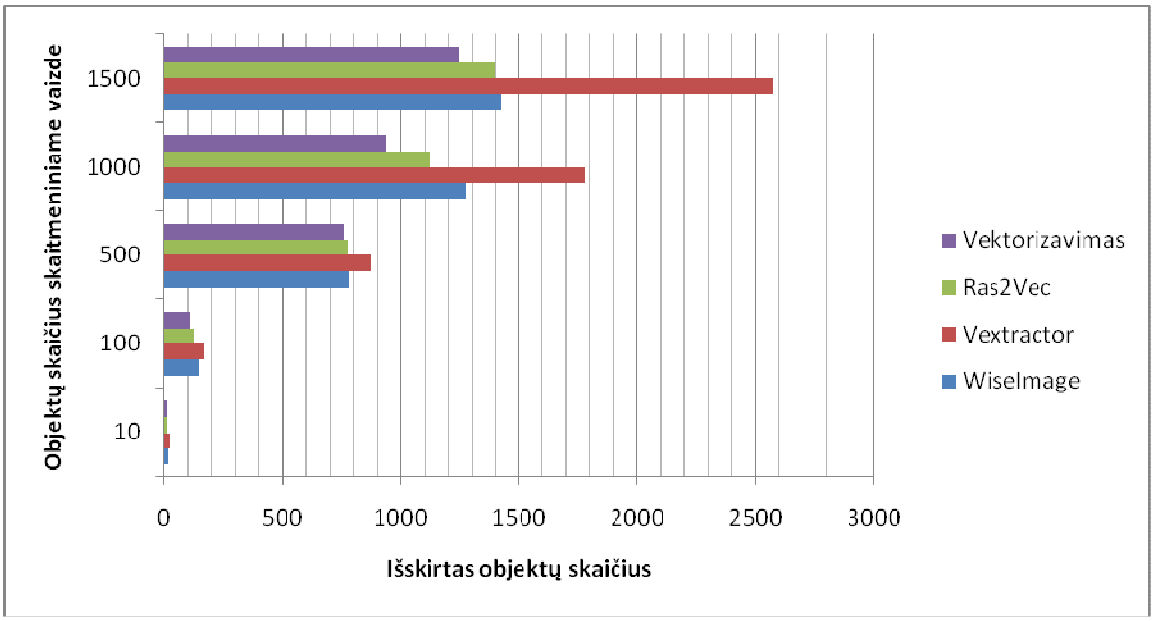
4.2.2 Vektorių skaičius

Eksperimento duomenims buvo naudojami penki skirtingi taškiniai vaizdai, kuriuose esantys objektai kertasi tarpusavyje bei jų skaičius yra iš anksto žinomas. Šio eksperimento metu yra tiriama kaip sistemos sugeba išlaikyti objektų vientisumą. Tirsime tik polilinijų atpažinimą, kadangi nei *Ras2Vec* nei *Vektorizavimas* neatpažįsta kitų objektų. Palyginimui bus vektorizuojami taškiniai vaizdai, kuriuose objektai nesikerta.

Eksperimente naudotus paveiksliukus galima rasti priede.

6 lentelė. Vektorizavimo sistemų išskirtų objektų skaičiai

Vaizdo nr./Sistema		*WiseImage*	*Vextractor*	*Ras2Vec*	*Vektorizavimas*
1 (10 obj.)	Objektai kertasi	16	22	13	10
	Objektai nesikerta	10	10	10	10
2 (100 obj.)	Objektai kertasi	148	168	123	105
	Objektai nesikerta	100	100	100	100
3 (500 obj.)	Objektai kertasi	782	872	777	760
	Objektai nesikerta	500	500	500	500
4 (1000 obj.)	Objektai kertasi	1272	1779	1123	938
	Objektai nesikerta	1000	1000	1000	1000
5 (1500 obj.)	Objektai kertasi	1425	2575	1396	1245
	Objektai nesikerta	1500	1500	1500	1500



55 pav. Vektorizavimo sistemų išskirtų objektų skaičius

Iš lentelės (6 lentelė) bei grafiko (55 pav.) matome, kad visos sistemos išskyrė vienodą bei teisingą objektų skaičių vaizduose, kuriuose objektai nesikerta. Sudėtingesniuose taškiniuose vaizduose, kur objektai tarpusavyje kertasi, sistemos išskyrė skirtingus skirtingą objektų skaičių.

Mažiausią objektų skaičių visuose vaizduose išskyrė dvi sistemos: *Ras2Vec* bei *Vektorizavimas*, kadangi Šios sistemos visas besikertančias linijas apjungė į polilinijas. Didžiausią objektų skaičių išskyrė *Vextractor* sistema, kuri nesujungė daugumos linijų į polilinjas. Be to *Vextractor* sistemos rezultate buvo rasta nemažai dubliuotų objektų.

Eksperimento metu pastebėta, kad objektų skaičiui peržengus 1000 vienetų visos sistemos, išskyrus *Vextractor,* generavo mažiau objektų negu yra iš tikrųjų. Taip atsitiko todėl, kad suprastėjo vaizdo kokybė ir kai kurie objektai nebuvo išskirti arba buvo apjungti su kitais.

Tiksliausi rezultatai esant, mažam objektų skaičiui taškiniame vaizde, gauti su *Vektorizavimas* sistema. Objektų skaičiui pasiekus 1000 ribą tiksliau veikė *WiseImage* sistema.

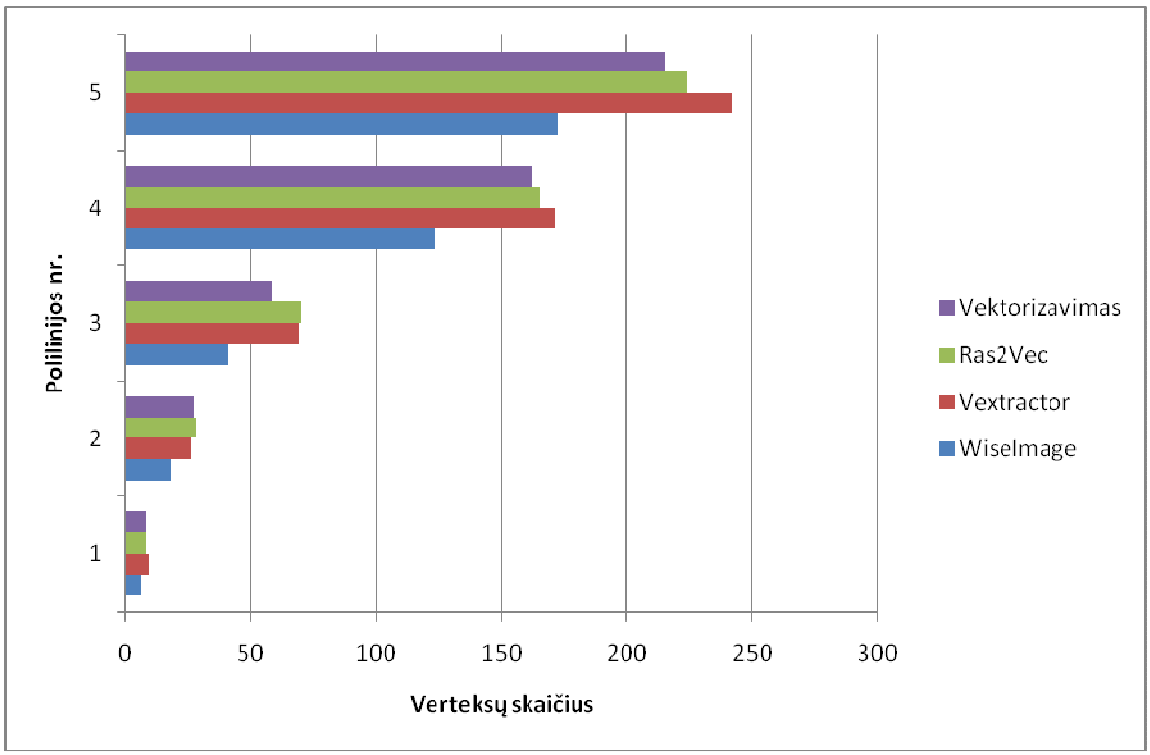
4.2.3 Polilinijos verteksų skaičius

Eksperimento duomenims buvo naudojamas vienas taškinis vaizdas turintis penkias nesikertančias polilinijas. Šio eksperimento metu yra tiriama kaip sistemos sugeba sumažinti išskirtų polilinijų sudėtingumą.

Eksperimente naudotus paveiksliukus galima rasti priede.

7 lentelė. Vektorizavimo sistemų išskirtas polilinijos verteksų skaičius

Polilinijos nr./Sistema	*WiseImage*	*Vextractor*	*Ras2Vec*	*Vektorizavimas*
1	6	9	8	8
2	18	26	28	27
3	41	69	70	58
4	123	171	165	162
5	172	242	224	215



56 pav. Vektorizavimo sistemų išskirtas polilinijos verteksų skaičius

Iš lentelės (7 lentelė) bei grafiko (56 pav.) matome, kad mažiausią verteksų skaičių išskyrė *WiseImage* sistema. Jos rezultate gautos polilinijos yra paprastesnės nei kitų sistemų, tačiau per mažas taškų skaičius gali reikšti jog yra prarandama svarbi informacija. *WiseImage* sugeneruotos polilinijos atrodo kampuotos bei neatitinka originalo formos. *Vextractor* sistemos išskirtos polilinijos turi daugiausiai verteksų, kurie gali būti pertekliniai, todėl tolimesnis rezultatų apdorojimas gali būti sudėtingesnis. Optimaliausi rezultatai gauti naudojant *Ras2Vec* bei *Vektorizavimas* sistemas.

4.2.4 Tirtų sistemų privalumai ir trūkumai

Sistemų lyginamoji analizė apibendrinta 7 lentelėje.

8 lentelė. Tirtų sistemų privalumai ir trūkumai

Savybė/Sistema	*WiseImage*	*Vextractor*	*Ras2Vec*	*Vektorizavimas*
Kontūrinių linijų išskyrimas	Taip	Taip	Taip	Taip
Centrinių linijų išskyrimas	Taip	Taip	Taip	Taip
Geometrinių objektų išskyrimas	Taip	Taip (tik apskritimai)	Ne	Ne
Lankų išskyrimas	Taip	Taip	Ne	Ne
Taškinio vaizdo apdorojimo funkcijos (filtrai)	Taip	Taip	Ne	Taip
Rezultatų redagavimas	Taip	Taip	Ne	Taip
Vektorizuojamo regiono pasirinkimas	Ne	Ne	Ne	Taip
Rezultatas DXF formate	Taip	Taip	Ne	Taip
Teksto atpažinimas	Taip	Ne	Ne	Ne

Mūsų sistemos didžiausias minusas yra tai, jog nėra atpažįstami geometriniai objektai. Tačiau nemažas pliusas yra tai jog galimas tik taškinio vaizdo regiono vektorizavimas.

4.3 Išvados

1. Šiame skyriuje aptarti ir palyginti taškinių vaizdų ploninimo algoritmai, bei ištirtos keturios vektorizavimo sistemos: *WiseImage*, *Vextractor*, *Ras2Vec* ir *Vektorizavimas*.
2. Taškinių vaizdų ploninimo metodai skirstomi į dvi grupes: vieno ir dviejų žingsnių metodai. Atlikus eksperimentą nustatyta, kad greičiausiai veikia Delaunay trianguliacija bei kontūrais paremti metodai, lėčiausiai tipinio ploninimo metodas.
3. Ištirtos penkios sistemos tiriant tris parametrus: vektorizavimo greitį, išskirtų vektorių skaičių bei išskirtų polilinijų verteksų skaičių.
4. Greičiausiai vektorizavimą vykdė *WiseImage* bei *Vektorizavimas* sistemos, lėčiausiai – *Vextractor*. Daugiausiai objektų išskyrė *Vextractor* sistema, mažiausiai *Vektorizavimas*. Sudėtingiausios polilinijos gautos naudojant V*extractor* sistema, mažiausiai verteksų turinčios polilinijos gaunamos su *WiseImage* sistema.

5 IŠVADOS

1. Darbe aptarti du duomenų formatai: taškinis vaizdas bei vektorinis. Išanalizuotas poreikis versti taškinius vaizdus į vektorinį formatą. Aptarta taškinio vaizdo automatinio vertimo į vektorinį formatą (vektorizavimo) probleminė sritis bei jo panaudojimo galimybės. Aptartos esminės, su vektorizavimu susijusios, problemos bei kiti svarbūs šios srities aspektai.
2. Aprašytas vektorizavimo procesas kurį sudaro šie veiksmai: taškinio vaizdo ploninimas, susikirtimų žymėjimas, grandinės sekimas, susikirtimų sujungimas, grandinės apkirpimas, segmentacija bei segmentų sujungimas.
3. Išanalizuotos pagrindinės esamų vektorizavimo metodų grupės. Aptarti jų privalumai ir trūkumai. Rinkoje egzistuojantys panašūs produktai vis dar pakankamai brangūs. Dėl to, buvo nuspręsta sukurti vektorizavimo sistemą, kuri vektorizuotų taškinius vaizdus ploninimo metodu.
4. Remiantis atlikta srities analize buvo dokumentuoti galimi vartotojų reikalavimai, atliktas sistemos specifikavimas pagal Volere šabloną. Remiantis vartotojų iškeltais reikalavimais bei sukauptomis šios srities žiniomis, buvo suprojektuota vektorizavimo sistema.
5. Pasiūlytas ploninimo algoritmas duoda švaresnį rezultatą nei tipinis ploninimo algoritmas. Tai įrodo mažesnis objektinių vaizdo taškų kiekis suplonintame vaizde. Rezultatų lyginimas buvo atliekamas ploninant identiškus taškinius vaizdus.
6. Palyginus veikimo laikus naudojant tą patį, skirtingų dydžių, taškinį vaizdą (57 pav.) nustatyta, kad algoritmas naudojantis Delaunay trianguliaciją veikia ~75% sparčiau už tipinį Zhang ir Suen pasiūlytą algoritmą.
7. Atliktas realizuotos sistemos dalies testavimas. Esminių klaidų neaptikta, tačiau dėl sprendžiamos problemos sudėtingumo, modulį dar reikia tobulinti.
8. Atliktas eksperimentinis tyrimas, kurio metu buvo palyginti vektorizavimo laikai, vektorių skaičiai bei polilinijų verteksų skaičiai, gauti su *Ras2Vec*, *WiseImage*, *Vextractor* bei su mūsų sistema *Vektorizavimas*.
9. Sukurtos sistemos vektorizavimo sparta ir rezultatų tikslumas yra patenkinamas. Ateityje sistema gali būti praplėsta teksto bei geometrinių objektų atpažinimu bei pusiau automatinio vektorizavimo, kai vartotojas gali pats išspręsti dviprasmiškas situacijas, galimybe.

LITERATŪRA

1. R. C. Gonzalez, P. Wintz, Digital Image Processing“ (2nd edition), Addison Wesley, 1987, p. 105 – 123.
2. F. van Dam, F. Hughes, Computer Graphics Principles and Practice“ (2nd edition), Addison Wesley, 1992, p. 567 – 580.
3. P. L. Rosin, G. A. W. West, Nonparametric segmentation of curves into various Representations, Pattern Analysis and Machine Intelligence, 1995, Nr. 17, p.1140 – 1153.
4. J. Zou, H. Yah, Extracting strokes from static line images based on selective searching, Pattern Recognition Proceedings, 1999, Nr. 2, p. 1142 – 1144.
5. T. Zhang, and C. Suen, A fast parallel algorithm for thinning Digital patterns, Communications of the ACM, 1984, Nr. 27, p. 236 – 239..
6. J. J. Zou, H. Yan, Vectorization of Cartoon Drawings, ACM International Conference Proceeding Series, 2002, Nr. 9, p. 77 – 78.
7. K. Tombre, Analysis of engineering drawing: state of the art and challenges, Graphics Recognition -- Algorithms and Systems, 2000, p. 257 – 264.
8. A. Chhabra, Graphic Symbol Recognition: An Overview, 1 Lecture Notes In Computer Science, 1998, Nr. 1389, p . 68 – 79.
9. K. Tombre, and A.K. Chhabra, Graphics Recognition - Algorithms and Systems, Lecture Notes in Computer Science, 1998, Nr. 1389, p. 420 – 422.
10. J. Chiang, S. Tue, Y. Leu, A new algorithm for line image vectorization, PR, 1998, Nr. 39, p. 1541-1549
11. J. Song, F. Su, C. Tai, J. Chen, S. Cai, Line Net Global Vectorization: an Algorithm and Its Performaance Evaluation, CVPR`00, 2000, Nr. 1, p. 1383 – 1390.
12. V. A. Kovalevsky, New definition and fast recognition of digital straight segments and arcs, Pattern Recognition Proceedings, 1990. Nr. 2, p. 31 - 34.
13. E. Bodansky, G. Alexander, P. Morakot, Post-processing of lines obtained by raster-to-vector conversion, Vision (machine Vision Association of SME), 2002, Nr. 18, p. 45 - 52.
14. L. Gorman, Basic Techniques and Symbol-Level Recognition An Overview, Graphic Recognition. Methods and Application, 1996, Nr. 1, p. 1 – 12.
15. S. Levachkine, E. Polchkov, Integrated Technique for Automated Digitization of Raster Maps, Revista Digital Universitaria, 2000, Nr. 1, p. 10 – 13.

16. D. Dori, Orthogonal Zig-Zag: an Algorithm for Vectorizing Engineering Drawings Compared with Hough Transform, Advances in Engineering Software, 1997, Nr. 28, p. 11 – 24.
17. Y. Wu, Raster, Vector, and Automated Raster-to-Vector Conversion, Moving Theory into Practice: Digital Imaging for Libraries and Archives, 2000, p. 725 – 769.
18. L.R. Poos and Y. Wu, Digitizing History: GIS and Historical Research, GIS World, 1995, p. 48 – 51.
19. GIS programinė įranga [interaktyvus] [žiūrėta 2007-05-01]. Prieiga per Internetą: http://www.hnit-baltic.lt/DesktopDefault.aspx?tabID=3428&alias=hnit-baltic&lang=lt-LT
20. Raster Vector Conversion Software [interaktyvus] [žiūrėta 2007-05-01]. Prieiga per Internetą: http://www.trixsystems.com/tractrix.html
21. Comparison of raster to vector conversion software [interaktyvus] [žiūrėta 2007-05-01]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Comparison_of_raster_to_vector_conversion_software
22. Compare of the R2V and Easy Trace 7.5 PRO software packages' performance capabilities at automatic raster to vector conversion of color isoline rasters. [interaktyvus] [žiūrėta 2007-05-01]. Prieiga per Internetą: http://www.easytrace.com/site/english/easytracepro/ET_vs_R2V.html

TERMINŲ IR SANTRUMPŲ ŽODYNAS

GIS – Geografinė informacinė sistema.

CAD – Kompiuterinio dizaino sistema.

DXF (Drawing eXchange Format) – Vektorinis duomenų formatas.

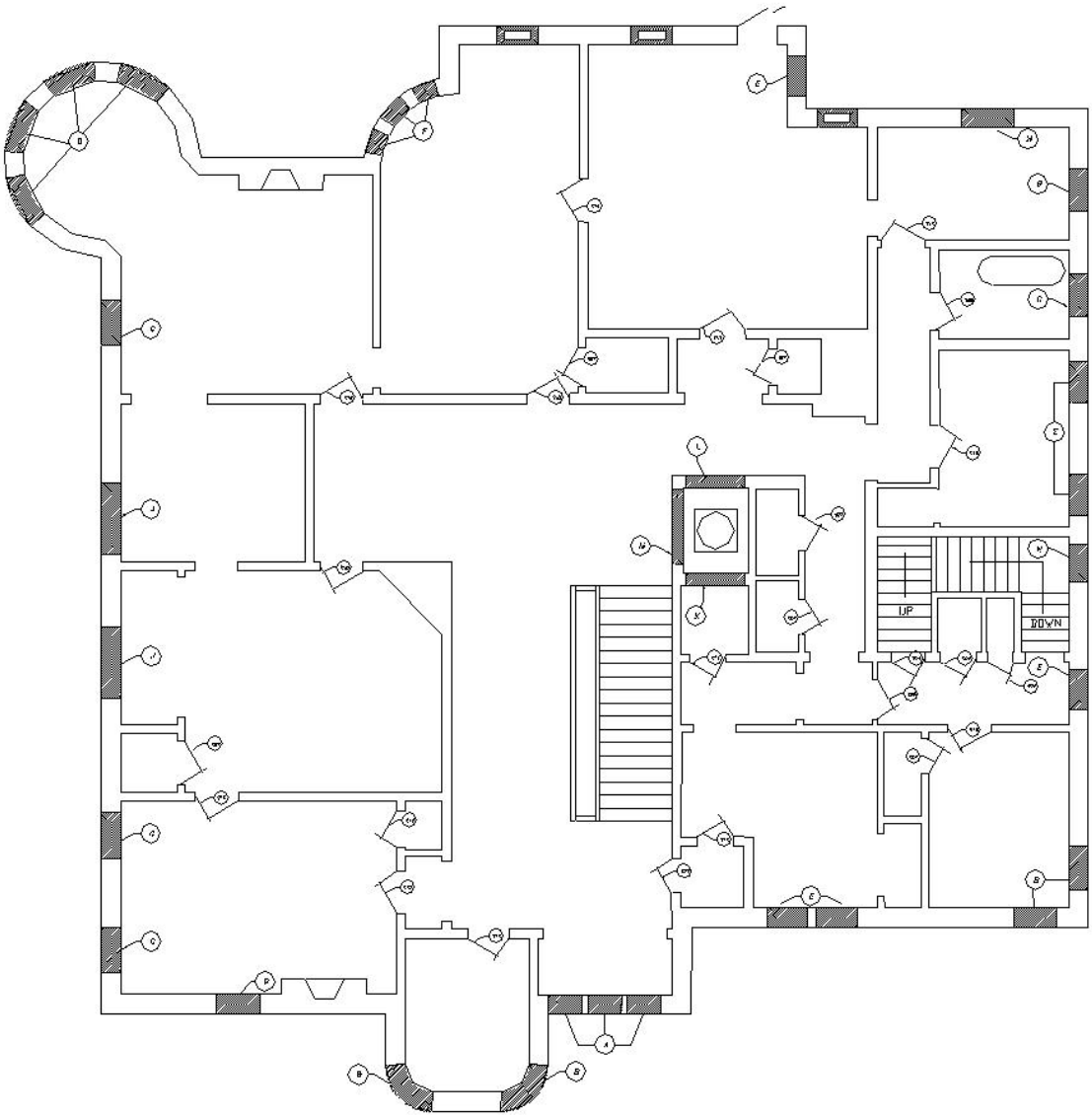
UML (Unified modeling language) - Unifikuota modeliavimo kalba.

MFC *(Microsoft Foundation Classes)* – Microsoft programinė biblioteka.

COM (Component Object Model) - Komponentinis objektų modelis.

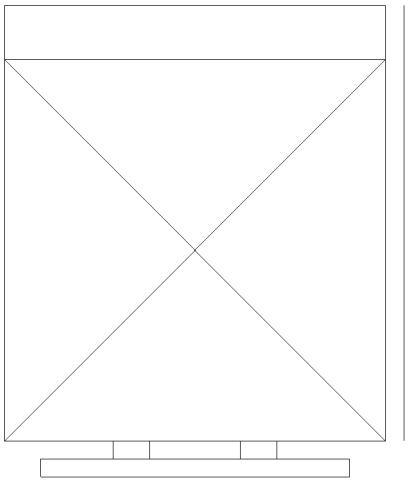
PRIEDAI

Vektorizavimo greičio eksperimente naudoti taškiniai vaizdai:

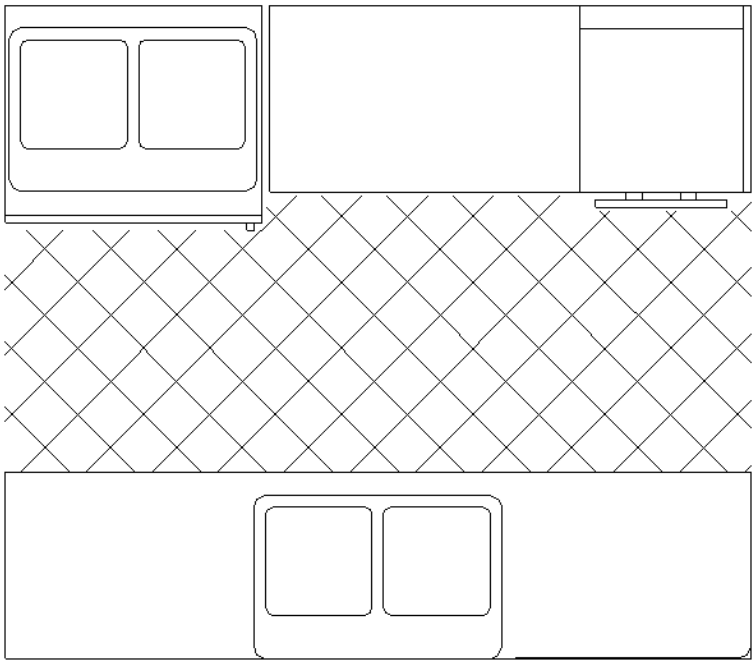


57 pav. Eksperimento paveiksliukas nr. 1

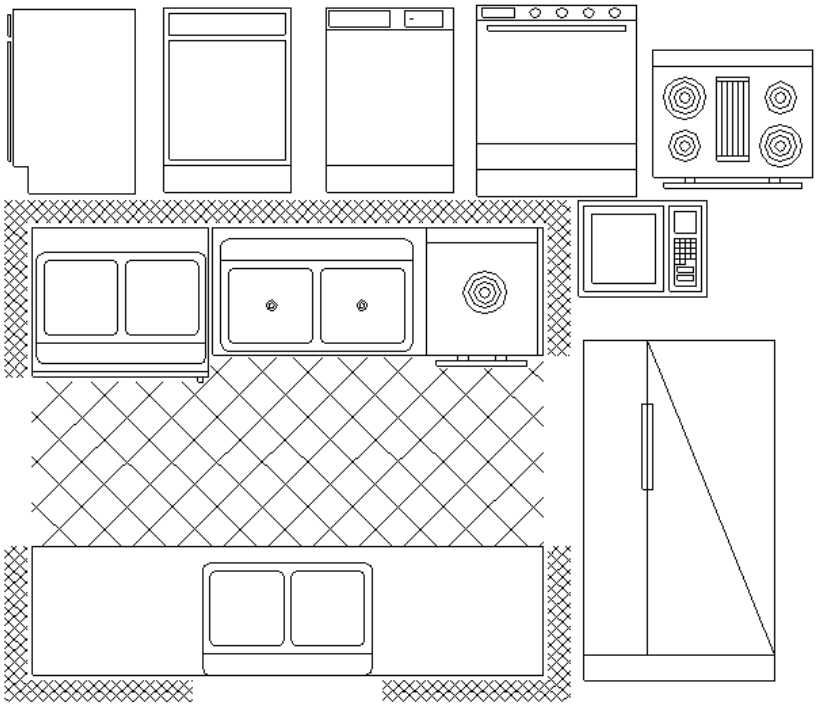
Vektorių skaičiaus eksperimente naudoti taškiniai vaizdai:



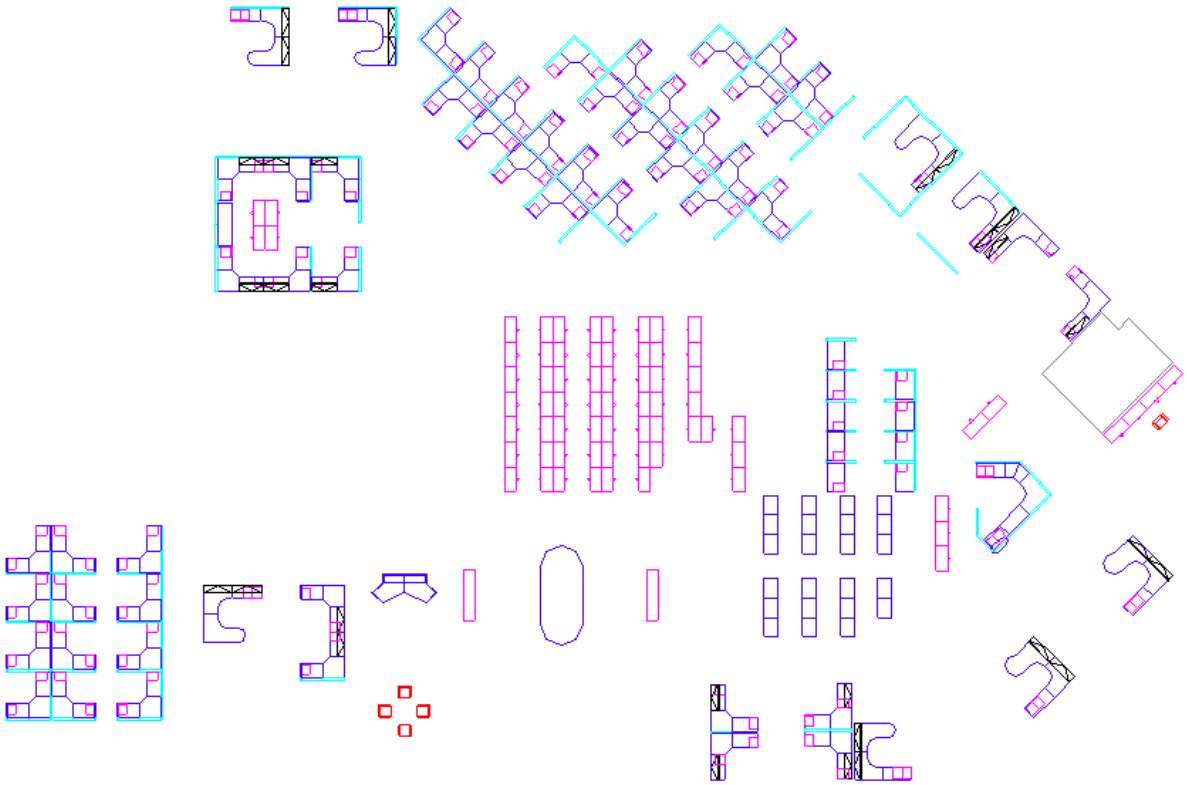
58 pav. Eksperimento paveiksliukas nr. 2



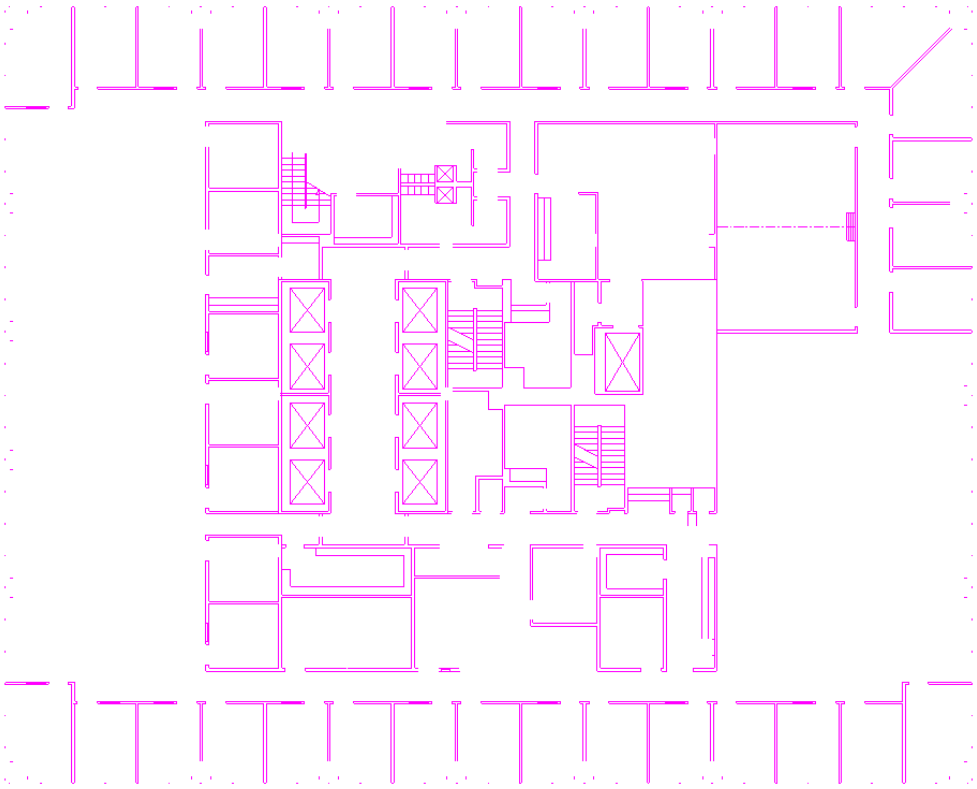
59 pav. Eksperimento paveiksliukas nr. 3



60 pav. Eksperimento paveiksliukas nr. 4

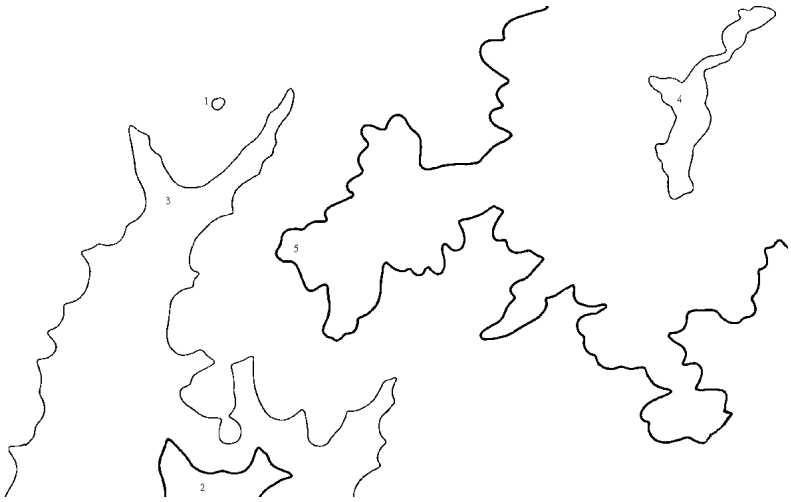


61 pav. Eksperimento paveiksliukas nr. 5



62 pav. Eksperimento paveiksliukas nr. 6

Polilinijos verteksų skaičiaus eksperimente naudoti taškiniai vaizdai:



63 pav. Eksperimento paveiksliukas nr. 7