KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

MULTIMEDIJOS INŽINERIJOS KATEDRA

ANDRIUS RIEPŠAS

ALGORITMŲ OPERACIJOMS SU PLOKŠČIOSIOMIS GEOMETRINĖMIS FIGŪROMIS SUDARYMAS IR TYRIMAS

MAGISTRO DARBAS

Vadovas doc. Aleksas Riškus

Kaunas, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

MULTIMEDIJOS INŽINERIJOS KATEDRA

ANDRIUS RIEPŠAS

ALGORITMŲ OPERACIJOMS SU PLOKŠČIOSIOMIS GEOMETRINĖMIS FIGŪROMIS SUDARYMAS IR TYRIMAS

MAGISTRO DARBAS

Vadovas .....
doc. Aleksas Riškus
2010 m. gegužės 31 d.

Recenzentas .....
prof. Eduardas Bareiša
2010 m. gegužės 31 d.

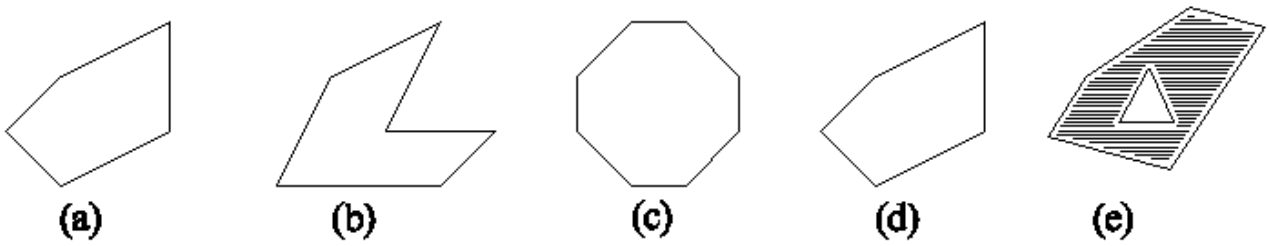
Atliko .....
IFM 4-1 gr. stud.
Andrius Riepšas
2010 m. gegužės 31 d.

Kaunas, 2010

Turinys

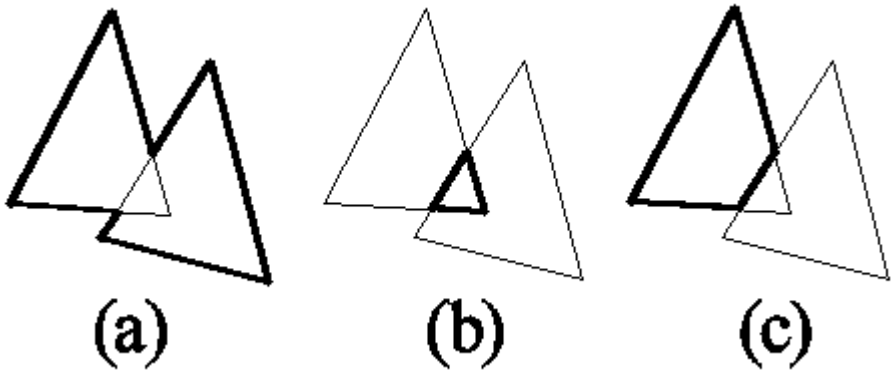
1 Įvadas

Plokščioji geometrinė figūra (terminas, naudojamas šitame darbe) – tai plokštumos dalis apribota kreivėmis. Šiame darbe nagrinėjamos tokios uždaros figūros, kurios sudarytos iš viršūnių (nemažiau nei 3 viršūnės) ir kurios gali būti jungiamos tiesėmis, lankais ar kvadratinėmis bei kubinėmis Bezjė (Bezier) kreivėmis. Figūros gali būti išgaubtos (*convex,* 1(a) pav.), įdubusios (*concave,* 1(b) pav.) , taisyklingos (*regural,* 1(c) pav.), netaisyklingos (*irregular,* 1(d) pav.), „save kertančios“ (self-intersecting) (pastarosios figūros nebus analizuojamos ir apdorojamos), turėti skyles (1(e) pav.).



1 pav. Plokščiųjų geometrinių figūrų tipai

Sujungimo (2(a) pav.), susikirtimo (2(b) pav.), atėmimo (2(c) pav.) operacijos su plokščiosiomis geometrinėmis figūromis yra svarbios ir labai plačiai naudojamos grafinėse projektavimo sistemose (*CAD – Computer Aided design*), todėl labai svarbu tobulinti jau esamus ir kurti naujus šių operacijų algoritmus.



2 pav. Operacijos su plokščiosiomis geometrinėmis figūromis - sujungimas, susikirtimas, atėmimas

Algoritmai operacijoms su plokščiosiomis geometrinėmis figūromis yra vertinami pagal

šiuos kriterijus:

- skaičiavimų laikas;
- rezultatų (gautų figūrų) kokybė.

2 Analitinė dalis

Labai plačiai paplitę algoritmai operacijoms su uždaromis geometrinėmis figūromis yra paremti tinklelio (*grid*), kuris dengia pradinius duomenis (figūras), paruošimu, skenavimo linijų (*scanline*) generavimu jame ir galutinių rezultatų skaičiavimu (vienas iš jų yra aprašytas „Fast scan conversion of arbitrary polygons“ [1] algoritme). Tačiau šis metodas yra pakankamai lėtas ir ne visada tikslus. Kokybė nukenčia dėl to, kad paruoštos skenavimo linijos ant tinklelio labai dažnai nekerta realių viršūnių koordinačių ir jų vietoje yra paskaičiuojamos naujos koordinatės, nukrypusios nuo pradinių. Norint padidinti tikslumą, galima didinti tinklelio tankį, tačiau tada labai stipriai auga skaičiavimų laikas (net ir su labai dideliu tikslumu, nėra pasiekiami pakankamai kokybiški rezultatai). Tokie yra esminiai skenavimo linijos algoritmų trūkumai.

Buvo peržiūrėti ir panagrinėti kiti algoritmai. Labai senai sukurtas Sutherland–Hodgman[10] algoritmas, tačiau naudojant šį algoritmą (įdubusioms geometrinėms figūroms) gautuose rezultatuose atsiranda papildomų kraštinių. Tai yra priimtina, kai šios operacijos naudojamos grafiniam duomenų atvaizdavimui (*graphic rendering),* bet netinka kitoms sistemoms (*CAD* ir t.t.), kurios atlieka geometrinius skaičiavimus.

Weiler [11] pristatytas metodas yra paremtas grafais, kurie užpildomi pradinių duomenų viršūnėmis ir susikirtimų taškais bei informacija apie taškų pozicijas vienas kito atžvilgiu. Tai sąlygoja labai sudėtingas duomenų struktūras bei sudėtingus veiksmus, kurie bendrame algoritmo aprašyme nėra išnagrinėti, o įvairios šio algoritmo realizacijos turi įvairių trūkumų ar apribojimų.

Liang-Barsky [5] ir Maillot [6] algoritmai yra apriboti sąlyga, kad viena iš figūrų turi būti stačiakampis. Tai gali būti naudinga grafiniam objektų vaizdavimo sprendimui, bet ne bendriems atvejams.

Reikia atkreipti dėmesį į tai, kad apžvelgti algoritmai yra nagrinėjami tik su daugiakampiais (viršūnės jungiamos tiesėmis), kai tuo tarpu šiame darbe pateikiamas algoritmas gali dirbti ir su tokiomis figūromis, kurių viršūnės yra jungiamos lankais bei bezjė kreivėmis.

Buvo išbandyta *Java2D API.* Tai viena iš *Java* bibliotekų, skirta darbui su dvimačiais geometriniais objektais (jos paskirtis yra kiek platesnė, bet mūsų atveju domina tik šita sritis). Naudojant *Java 2D API* galima sukurti įvairiausias geometrines figūras: pradedant paprasčiausiais trikampiais, apskritimais ir baigiant sudėtingesnėmis figūromis. Būtina akcentuoti ir išskirti tokią

galimybę, kad figūrų viršūnes galima jungti ne tik atkarpomis ar puslankiais, bet taip pat kvadratinėmis ir kubinėmis bezjė *(Bezier)* kreivėmis.

Atlikus geometrinių figūrų sujungimo testus naudojant *Java 2D API*, rezultatai buvo labai tikslūs. Tačiau vykdant tą patį sujungimo testą su didesniu figūrų skaičiumi (daugiau nei 1000 figūrų), *Java 2D* nebaigdavo savo darbo. Priežastis – labai neefektyvus atminties naudojimas *Java 2D API* atliekant geometrinius skaičiavimus. Dėl šios priežasties buvo atsisakyta toliau naudoti *Java 2D API. Java 2 API* bus naudojamas rezultatų palyginimui ir tikslumo įvertinimui, nes, kaip jau minėta anksčiau, po operacijų su figūromis gaunami labai tikslūs geometriniai objektai.

Šitame darbe bus nagrinėjamas algoritmas operacijoms su geometrinėmis figūromis, kuris paremtas šluojančios tiesės (*sweepline*) principu.

2.1 Darbo tikslas

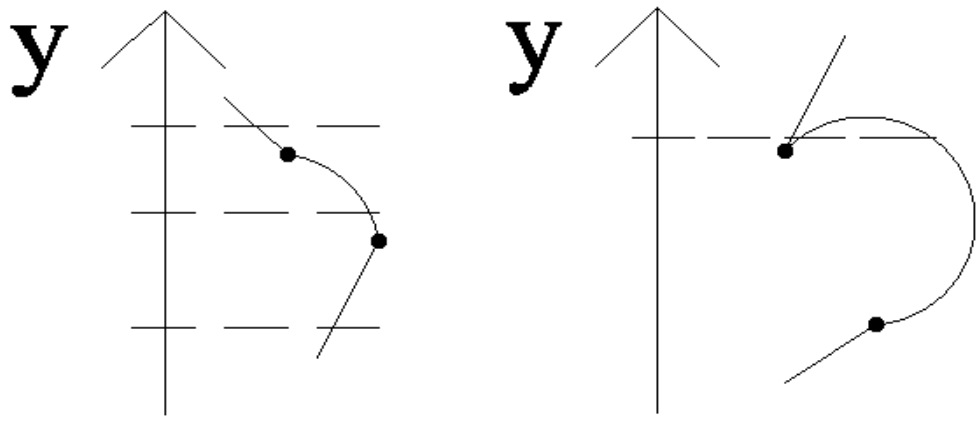
Sukurti ir ištirti operacijų (sujungimo, sankirtos, atėmimo) su plokščiosiomis uždaromis geometrinėmis figūromis algoritmus.

Pagrindiniai reikalavimai yra šie:

- Minimalus skaičiavimų laikas
- Skaičiavimų tikslumas
- C/C++ programavimo kalbos naudojimas

2.2 Bendras algoritmo aprašymas

Monotoninis segmentas y ašies atžvilgiu – tai toks segmentas, kuris su bet kokiu išvestu statmeniu kertasi nedaugiau kaip vieną kartą.

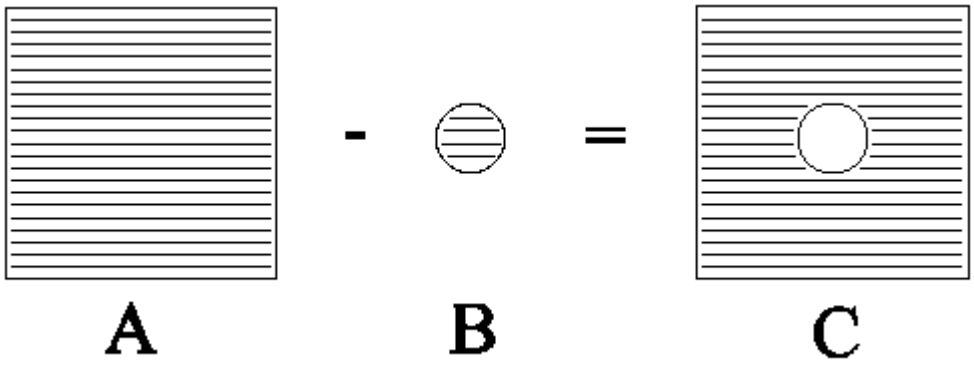


3 pav. Monotonis ir nemonotonis lanko segmentai

Kairėje 3 pav. pusėje visi segmentai yra monotoniniai bet kuriam išvestam statmeniui y ašies atžvilgiu, kai tuo tarpu dešinėje pusėje lanko segmentas yra nemonotoninis.

Vertikali šluojanti tiesė – tai tokia šluojanti tiesė, kuri pradeda judėti nuo įvykio taško, turinčio mažiausią x koordinatę, ir juda iki įvykio taško, turinčio didžiausią x koordinatę (lygiagreti y ašiai).

Horizontali šluojanti tiesė – tai tokia šluojanti tiesė, kuri pradeda judėti nuo įvykio taško, turinčio mažiausią y koordinatę, ir juda iki įvykio taško, turinčio didžiausią y koordinatę (lygiagreti x ašiai)



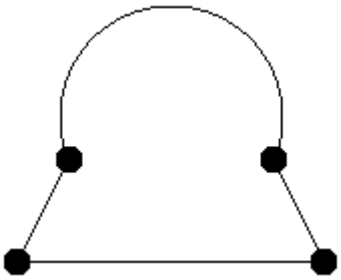
4 pav. Figūros B inversija

Figūros invertavimas – tai duotos figūros skirtumas su plokštuma. Tai iliustruojama pavyzdžiu parodytu 4 pav. Duota figūra *B* ir reikia ją invertuot. Norint tai padaryti, yra atliekama atimtis iš plokštumos *A* ir gaunamas rezultatas *C*.

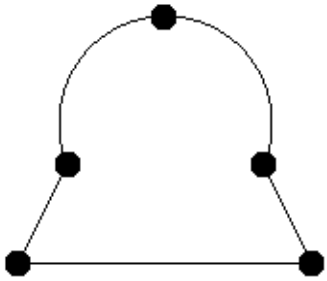
Pagrindinis algoritmas yra paremtas šiais etapais:

1. Pradinių duomenų paruošimas

   Gavus pradinius duomenis, tikrinama, ar plokštumos figūros ir jų skylės yra išdėstytos reikiama kryptimi (figūrų viršūnių kryptis – prieš laikrodžio rodyklę, o skylių atvirkščiai – pagal laikrodžio rodyklę). Tikrinami visi netiesiniai segmentai (lankai, kvadratinės bei kubinės bezjė kreives) kad nustatyti, ar jie yra monotoniniai y ašies atžvilgiu. Jei yra randami nemonotoniniai segmentai (5 pav.), jie verčiami į atitinkančius monotoninius (6 pav.).

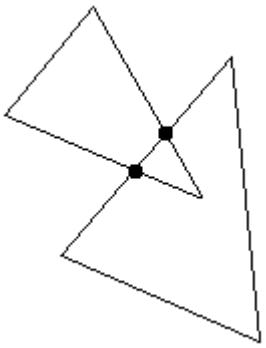


5 pav. Figūra su nemonotoniniu segmentu

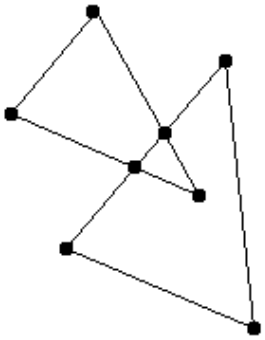


6 pav. Figūra su monotoniniais segmentais

2. Visų duomenų segmentų tarpusavio susikirtimų skaičiavimas bei jų apjungimas į bendrą visumą su segmentų viršūnėmis (įvykių taškai)



7 pav. Segmentų tarpusavio susikirtimai

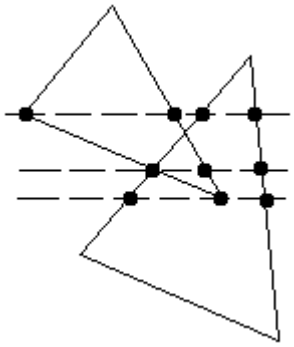


8 pav. Visi įvykio taškai

Tarpusavio segmentų susikirtimo taškams (7 pav.) skaičiuoti yra paruošti trys algoritmai. Visi jie yra paremti šluojančios tiesės principu ir turi tris skirtingas šluojančios tiesės statusų realizacijas: statusas, realizuotas binariniu medžiu, statusas, realizuotas masyvu su pilnu perrinkimu bei statusas, realizuotas masyvu su optimizuota paieška. Visos segmentų viršūnės ir gauti susikirtimo taškai (8 pav.) yra sudedami į vieną masyvą, kuris surūšiuojamas y koordinačių didėjimo tvarka.

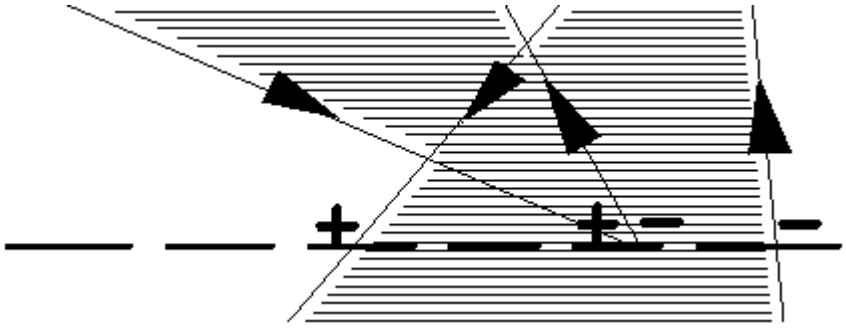
3. Šluojančios tiesės judėjimas per įvykių taškus bei jos susikirtimų su duomenų segmentais skaičiavimas

   Per kiekvieną įvykio tašką juda horizontali šluojanti tiesė. Kiekviename įvykio taške šluojanti tiesė yra kertama su duomenų segmentais, fiksuojama x koordinatė bei segmento pobūdis (kylantis ar krentantis)



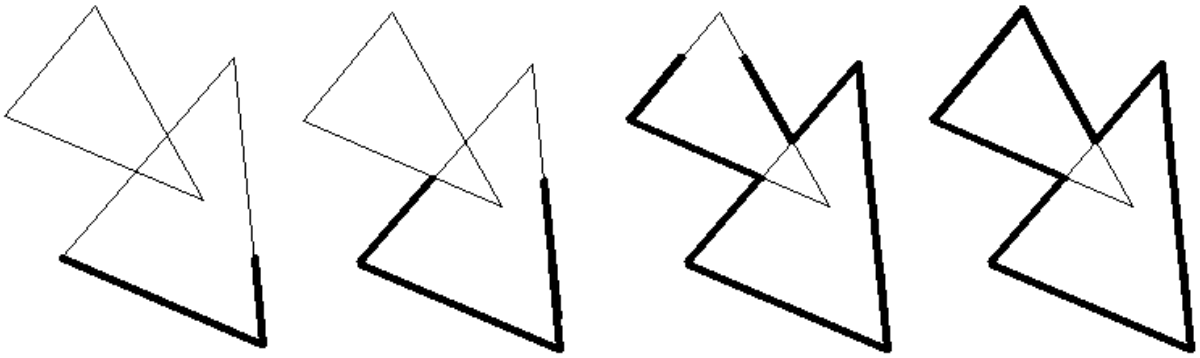
9 pav. Šluojanti tiesė ir jos susikirtimai su duomenų segmentais

4. Intervalų formavimas iš gautų šluojančios tiesės ir segmentų susikirtimų
   Kiekvienai šluojančios tiesės bei duomenų segmentų susikirtimų aibei yra formuojami intervalai. Intervalas fiksuojamas tada, kai kylančių ir krintančių segmentų suma (+1 krentantis segmentas, -1 kylantis segmentas, 10 pav.) yra lygi užduotam parametrui. Šio užduodamo parametro reikšmė priklauso nuo atliekamos operacijos – sujungimo atveju jis lygus „0“, sankirtos atveju - „1”.



10 pav. Intervalų formavimas

5. galutinio rezultato skaičiavimas iš suformuotų intervalų.
   Remiantis intervalais yra formuojama galutinė geometrinė figūra (-os). Atliekama intervalų analizė ir, priklausomai nuo intervalų, pridedami/šalinami rezultato segmentai (11 pav.). Detalus aprašymas pateikiamas skyriuje *3.4 Operacijų su plokščiosiomis geometrinėmis figūromis tyrimas*.



11 pav. Galutinių rezultatų generavimas

Šis pagrindinis algoritmas yra sudarytas sujungimo ir sankirtos operacijoms. Invertavimo funkcijos pagalba gaunamas figūrų atėmimas: viena iš figūrų yra invertuojama ir tada atliekamas abiejų figūrų sujungimas.

2.2.1 Pradinių duomenų paruošimas

Geometrinių figūrų viršūnių krypčių koregavimas

Prieš pradedant skaičiavimus, geometrinės figūros sutvarkomos taip, kad figūrų viršūnės būtų išdėstytos prieš laikrodžio rodyklę, o skylių – atvirkščiai.

Lankų, kvadratinių ir kubinių bezjė kreivių monotoniškumo tikrinamas ir nemonotoninių kreivių pakeitimas monotoninėmis

Duomenų paruošimo stadijoje svarbu atlikti netiesinių (lankų, bezjė kreivių) segmentų analizę bei modifikacijas. Šiame algoritme operacijoms su uždaromis geometrinėmis figūromis yra viena būtina sąlyga/apribojimas – visi segmentai, kurie nėra tiesės, turi būti monotoniniai (šis reikalavimas yra svarbus dėl to, kad viena šluojanti tiesė nekirstų to paties segmento daugiau nei vieną kartą).

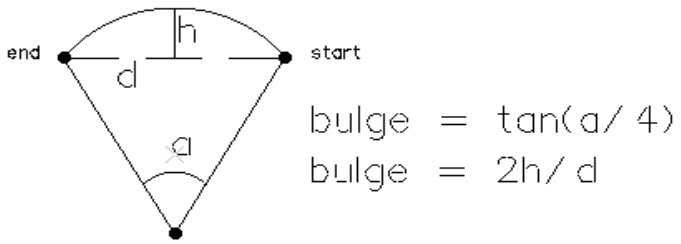
Lanko segmento analizė

Lanko aprašo formatas: *pradžios taškas (x0,y0) – lanko centro koordinatė (xC, yC) ir spindulys (r/-r) – pabaigos taškas (x1, y1).*

Jei spindulys yra neigiamas, tai lanko kryptis pagal laikrodžio rodyklę, jei teigiamas – prieš.

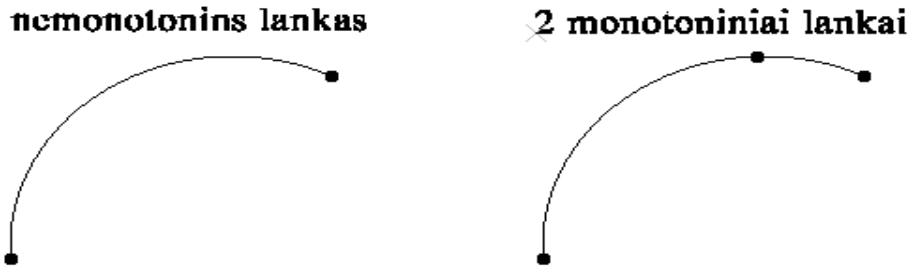
Norint nustatyti ar lankas yra nemonotoninis reikia atlikti tokius skaičiavimus:

1. paskaičiuoti lanko *bulge*



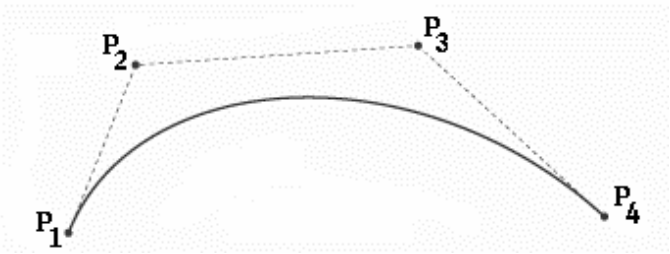
2. nustatyti, kuriuose ketvirčiuose yra lanko pradžios ir pabaigos taškai.

Nemonotoninio lanko pakeitimo monotoniniais lankais rezultatas parodytas paveikslėlyje:



Bezjė kreivių analizė.

Kubinė bezjė kreivė yra nusakoma keturiais taškais: pradžios taškas *(P_1),* pabaigos taškas *(P_4)* ir du kontroliniai taškai *(P_2),(P_3).* Kontroliniai taškai nėra kreivės taškai, jie tik nusako kreivės pobūdį (12 pav.).



12 pav. Kubinė Bezjė (Bezier) kreivė

Norint nustatyti, ar kubinė bezjė kreivė yra monotoninė, reikia paskaičiuoti ekstremumo taškus. Nemonotoninės kubinės bezjė kreivės atveju gali būti vienas arba du ekstremumo taškai.

Kubinės bezjė kreivės parametrinė lygtis yra:

$$B(t) = (1-t)^3 P1 + 3t(1-t)^2 P2 + 3t^2 (1-t)P3 + t^3 P4 \quad (1)$$

t parametras kinta intervale [0..1].

Skaičiuojama pirmos eilės išvestinė:

$$y_x' = \frac{y_t'}{x_t'}$$

Kai ši išvestinė lygi nuliui, gaunami ekstremumo taškai. Reikia suskaičiuoti ir išspręsti:

$$y_t' = 0$$

$$y_t' = ((1 - t)^3 y_1 + 3t(1 - t)^2 y_2 + 3t^2(1 - t)y_3 + t^3 y_4)'$$

$$y_t' = -3(1 - t)^2 y1 + 3((1 - t)^2 - 2t(1 - t))y2 + 3(2t(1 - t) - t^2)y3 + 3t^2 y_4 = 0$$

$$(-3y1 + 9y_2 - 9y_3 + 3y_4)t^2 + (6y_1 - 12y_2 + 6y_3)t + (-3y_1 + 3y_2) = 0 \quad (2)$$

Jei lygtis (2) neturi sprendinių, tai kubinė bezjė kreivė yra monotoninė, jei lygtis turi sprendinius, tai kubinė bezjė kreivė yra nemonotoninė.

Tuo atveju, kai kubinė bezjė kreivė yra nemonotoninė ir turint vieną ar dvi *t* reikšmes, atliekamas nemonotoninės bezjė kreivės suskaldymas į monotonines. Šiam tikslui pasiekti yra naudojamas *Kastelžo (Casteljau)* algoritmas.

Detaliai nagrinėjama situacija, kai kubinė bezjė kreivė turi vieną ekstremumo tašką, t.y. išsprendus (2) lygtį gauname vieną sprendinį. Gavus *t* reikšmę, paskaičiuojamas taškas *C*, kuris duotąją kubinę bezjė kreivę dalina į dvi nemonotonines kreives, kurių pradžios ir pabaigos taškai yra (P_1, *C*) ir *(C,* P_4). Naudojantis *Kastelžo* algoritmu, galima suskaičiuoti kontrolinius taškus dviems naujoms kreivėms.

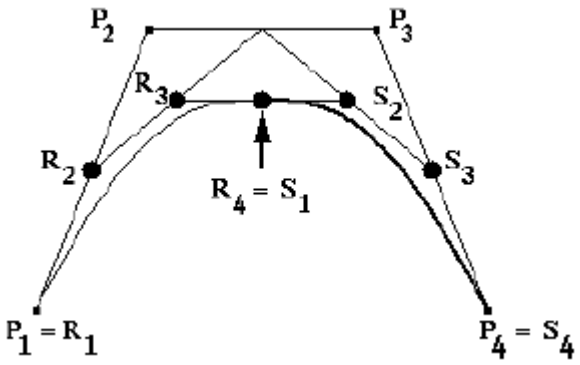
$$S_1 = C$$

$$R_2 = P_1 + t * (P_2 - P_1)$$

$$S_3 = P_3 + t * (P_4 - P_3)$$

$$R_3 = R_2 + t * ((P_2 + t * (P_3 - P_2)) - R_2)$$

$$S_2 = P_2 + t * (P_3 - P_2) + t * (S_3 - (P_2 + t * (P_3 - P_2))).$$



13 pav. Kastelžo (Casteljau) algoritmas

Gauname naujas monotonines kubines bezjė kreives (P1)(R2)(R3)(S1) ir (S1)(S2)(S3)(P4)

2.2.2 Visų duomenų segmentų tarpusavio susikirtimų skaičiavimas bei jų apjungimas į bendrą visumą su segmentų viršūnėmis (įvykių taškai)

Atlikus pradinį duomenų paruošimą, reikia surinkti visas viršūnių y koordinates bei visų pradinių segmentų tarpusavio susikirtimo taškų y koordinates į vieną masyvą – įvykio taškų masyvą (vėliau per kiekvieną įvykio tašką judės vertikali šluojanti tiesė).

Toliau pateikiamos skaičiavimų formulės ir algoritmai skirtingiems segmentų tipams:

1. Atkarpos
   - **atkarpa – atkarpa**

     Vienos atkarpos parametrinė lygtis:

     $$x(t)=x_0+t*(x_1-x_0)$$
     $$y(t)=y_0+t*(y_1-y_0)$$

     Kitos atkarpos parametrinė lygtis:

     $$x(t')=x_0+t'*(x_1-x_0)$$
     $$y(t')=y_0+t'*(y_1-y_0)$$

     Išsprendžiama lygčių sistema ir gaunamas *t* parametras.
2. Lankai
   - **atkarpa – lankas**

     Atkarpos parametrinė lygtis :

     $$x(t)=x_0+t*(x_1-x_0)$$
     $$y(t)=y_0+t*(y_1-y_0)$$

     čia (x_0, y_0) ir (x_1, y_1) atkarpos pradžios ir pabaigos koordinatės

     Apskritimo parametrinė lygtis:

     $$x(t')=x_c+r*(\frac{1-t'^2}{1+t'^2})$$
     $$y(t')=y_c+r*(\frac{2*t'}{1+t'^2})$$

     čia (x_c, y_c) – apskritimo centro koordinatės, r – apskritimo spindulys

     Sudaroma ir sprendžiama sekanti lygčių sistema:

     $$x_0+t*(x_1-x_0)=x_c+r*(\frac{1-t'^2}{1+t'^2})$$
     $$y_0+t*(y_1-y_0)=y_c+r*(\frac{2*t'}{1+t'^2})$$

- **lankas – lankas**

$$x(t)=x_c+r*(\frac{1-t^2}{1+t^2})$$
$$y(t)=y_c+r*(\frac{2*t}{1+t^2})$$

čia (x_c, y_c) – apskritimo centro koordinatės, r – apskritimo spindulys

Kito apskritimo lygtis identiška, tik kitas parametras t'.

Sudaroma ir sprendžiama sekanti lygčių sistema:

$$x0_c+r0*(\frac{1-t^2}{1+t^2})=x1_c+r1*(\frac{1-t'^2}{1+t'^2})$$
$$y0_c+r0*(\frac{2*t}{1+t^2})=y1_c+r1*(\frac{2*t'}{1+t'^2})$$

Lankų susikirtimo taškus galima paskaičiuoti ir kitaip.

Pirmiausia, paskaičiuojamas atstumas d tarp dviejų apskritimo centrų

$$d=\|P_1-P_0\|$$

- Jei $d>r_0+r_1$ tada sprendinių nėra, apskritimai yra atskiri
- Jei $d<|r_0-r_1|$ tada sprendinių nėra, nes vienas apskritimas yra kitame apskrime
- Jei $d=0$ ir $r_0=r_1$ tada apskritimai sutampa vienas su kitu ir yra begalinis sprendinių skaičius.

Nagrinėjant du trikampius $P_0P_2P_3$ ir $P_1P_2P_3$, mes galime užrašyti:

$$a^2+h^2=r_0^2 \text{ ir } b^2+h^2=r_1^2$$

Žinodami, kad $d=a+b$ mes galima suskaičiuoti *a:*

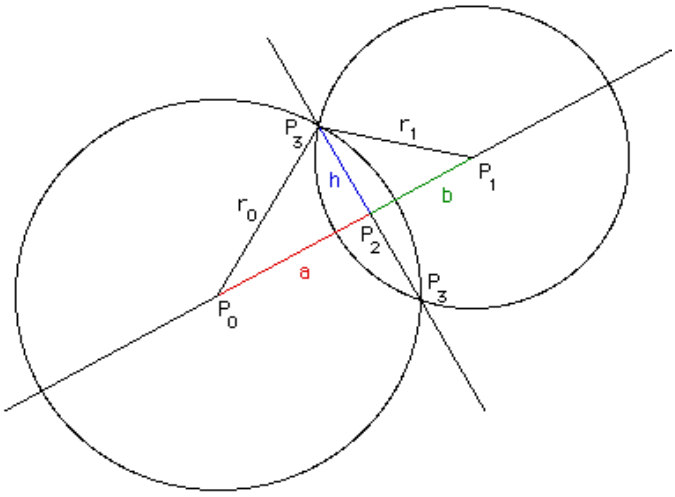
$$a=(r_0^2-r_1^2+d^2)/(2\mathrm{d})$$

Iš pirmos lygties galime išsireikši $h^2=r_0^2-a^2$ ir suskaičiuoti:

$$P_2=P_0+a(P_1-P_0)/d$$

Ir jei $P_3=(x_3, y_3)$, $P_0=(x_0, y_0)$, $P_1=(x_1, y_1)$ $P_2=(x_2, y_2)$, tai:

$$x_3=x_2\pm h(y_1-y_0)/d$$
$$y_3=y_2\pm h(x_1-x_0)/d$$



14 pav. Apskritimų tarpusavio susikirtimas

3. Kubinės bezjė kreivės
   - **kubinė bezjė kreivė – kubinė bezjė kreivė**

$$x(t)=(1-t_1)^3*x_0+3*(1-t_1)^2*t_1*x_1+3*(1-t_1)*t_1^2*x_2+t_1^3*x_3$$
$$y(t)=(1-t_1)^3*y_0+3*(1-t_1)^2*t_1*y_1+3*(1-t_1)*t_1^2*y_2+t_1^3*y_3$$

čia (x_0, y_0) - pradžios koordinatė, (x_1, y_1), (x_2 , y_2) - kontrolinių taškų koordinatės, (x_3 , y_3) - pabaigos koordinatė.

Įvedame pažymėjimus:

$$x_1=x_1(t_1)$$
$$y_1=y_1(t_1)$$

Kitos kubinės bezjė kreivės lygtis:

$$x(t)=(1-t_2)^3*x_0+3*(1-t_2)^2*t_2*x_2+3*(1-t_2)*t_2^2*x_2+t_2^3*x_3$$
$$y(t)=(1-t_2)^3*y_0+3*(1-t_2)^2*t_2*y_2+3*(1-t_2)*t_2^2*y_2+t_2^3*y_3$$

Įvedame pažymėjimus:

$$x_2=x_2(t_2)$$
$$y_2=y_2(t_2)$$

Sudaroma lygčių sistema:

$$\begin{cases} x_1(t_1)=x_2(t_2) \\ y_1(t_1)=y_2(t_2) \end{cases}$$
$$\begin{cases} x_1(t_1)-x_2(t_2)=0 \\ y_1(t_1)-y_2(t_2)=0 \end{cases}$$

Įvedame pažymėjimus:

$f_1 = x_1(t_1) - x_2(t_2)$
$f_2 = y_1(t_1) - y_2(t_2)$

Kubinių bezjė kreivių sankartai skaičiuoti naudojamas Niutono metodo algoritmas. Pirmiausia apibrėžiama, kaip Niutono metodas taikomas dviejų lygčių sistemai spręsti. Turime lygčių sistemą:

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases} \quad (1)$$

Turime pradinį jos sprendinį $x_0 = (x_1^{(0)}; x_2^{(0)})$, be to, tokias Δx_1 ir Δx_2 reikšmes, su kuriomis $x_1^{(0)} + \Delta x_1$ ir $x_2^{(0)} + \Delta x_2$ yra (1) lygties sprendiniai. Norint suskaičiuoti tokias Δx_1 ir Δx_2 reikšmes, reikia funkcijas $f_1(x_1, x_2) = 0$ ir $f_2(x_1, x_2) = 0$ išskleisti Teiloro eilute taško $(x_1^{(0)}; x_2^{(0)})$ aplinkoje ir palikę pirmus tris eilutės narius, turime tokią lygčių sistemą

$$\begin{cases} \frac{\partial f_1}{\partial x_1}\Delta x_1 + \frac{\partial f_1}{\partial x_2}\Delta x_2 = -f_1 \\ \frac{\partial f_2}{\partial x_1}\Delta x_1 + \frac{\partial f_2}{\partial x_2}\Delta x_2 = -f_2 \end{cases}$$

kurią galima užrašyti matricine išraiška:

$J \cdot \Delta x = -f$

čia J – antrosios eilės Jakobio matrica, Δx ir f – vektoriai stulpeliai, kurių komponentės yra atitinkamai Δx_1 ir Δx_2 bei f_1 ir f_2 .

Taigi, pradžioje užduodamas x_0 - pradinis sistemos sprendinys. Reikia rasti sprendinį *x* su užduotu tikslumu *eps,* per *k* iteracijų.

Sprendinio ieškoma tokia seka:

1. apskaičiuojama $f(x_k)$ ir nusprendžiama, ar reikia baigti, ar tęsti;
2. apskaičiuojama Jakobio matrica $J(x_k)$;
3. išsprendžiama lygtis $J(x_k) \cdot \Delta x_k = -f(x_k)$;
4. randamas $x_{k+1} = x_k + \Delta x_k$

Skaičiavimo pabaigos sąlyga. Skaičiavimas baigiamas, kai $max\left|\frac{\Delta x_i}{x_i}\right| < eps$, $1 \leqslant i \leqslant n$

arba $max|f_i(x)| < eps$, $1 \leqslant i \leqslant n$

- **atkarpa – kubinė bezjė kreivė**

$x(t) = (1-t)^3 * x_0 + 3*(1-t)^2 * t * x_1 + 3*(1-t) * t^2 * x_2 + t^3 * x_3$
$y(t) = (1-t)^3 * y_0 + 3*(1-t)^2 * t * y_1 + 3*(1-t) * t^2 * y_2 + t^3 * y_3$

čia (x_0, y_0) - pradžios koordinatė, (x_1, y_1), (x_2, y_2) - kontrolinių taškų koordinatės, (x_3, y_3) - pabaigos koordinatė.

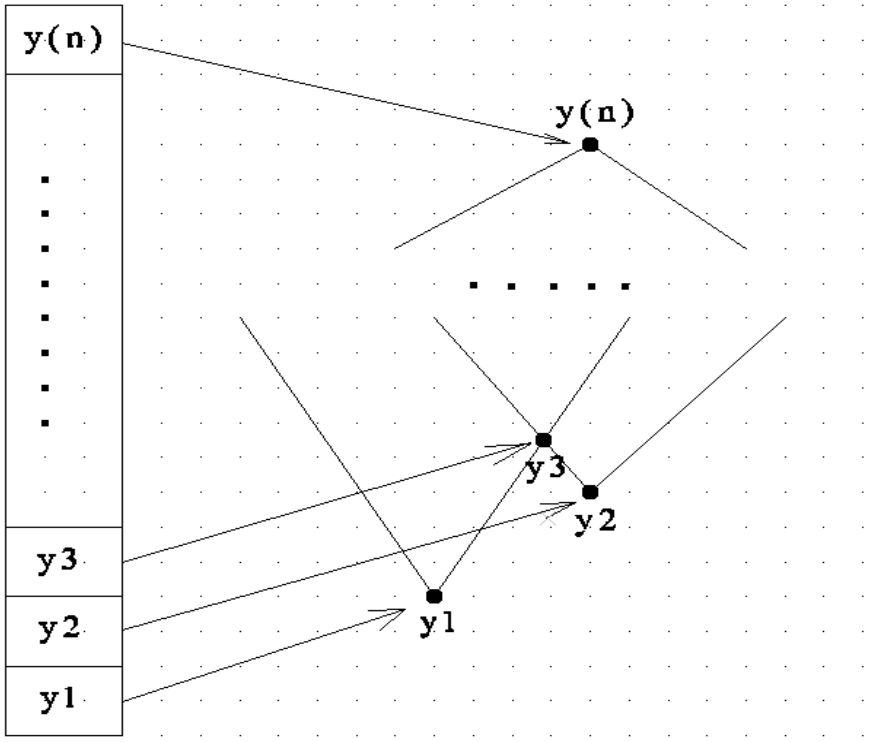
Atkarpos parametrinė lygtis :

$$x(t') = x_0 + t' * (x_1 - x_0)$$
$$y(t') = y_0 + t' * (y_1 - y_0)$$

čia (x_0, y_0) ir (x_1, y_1) atkarpos pradžios ir pabaigos koordinatės

Norint nustatyti tiesės ir kubinės bezjė kreivės susikirtimo taškus taip pat naudojamas Niutono metodas, kuris buvo detaliai išnagrinėtas ieškant susikirtimų tarp dviejų kubinių bezjė kreivių.

Atlikus segmentų susikirtimų analizę, gaunamas bendras įvykio taškų y koordinačių masyvas.



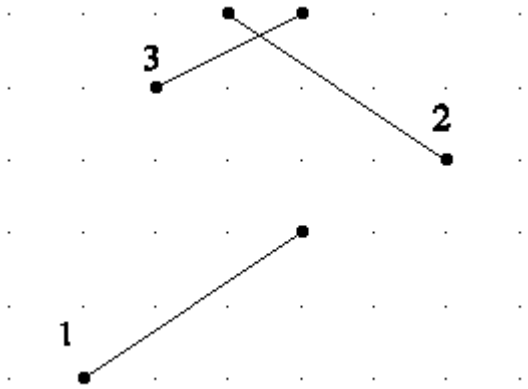
15 pav. Visų įvykio taškų masyvas

Segmentų tarpusavio susikirtimams rasti naudojami sekantys algoritmai:

- vertikali šluojanti tiesė, kurios statusas realizuojamas masyvu su pilnu perrinkimu
- vertikali šluojanti tiesė, kurios statusas realizuojamas masyvu su optimizuotu perrinkimu
- vertikali šluojanti tiesė, kurios statusas realizuojamas binariniu medžiu

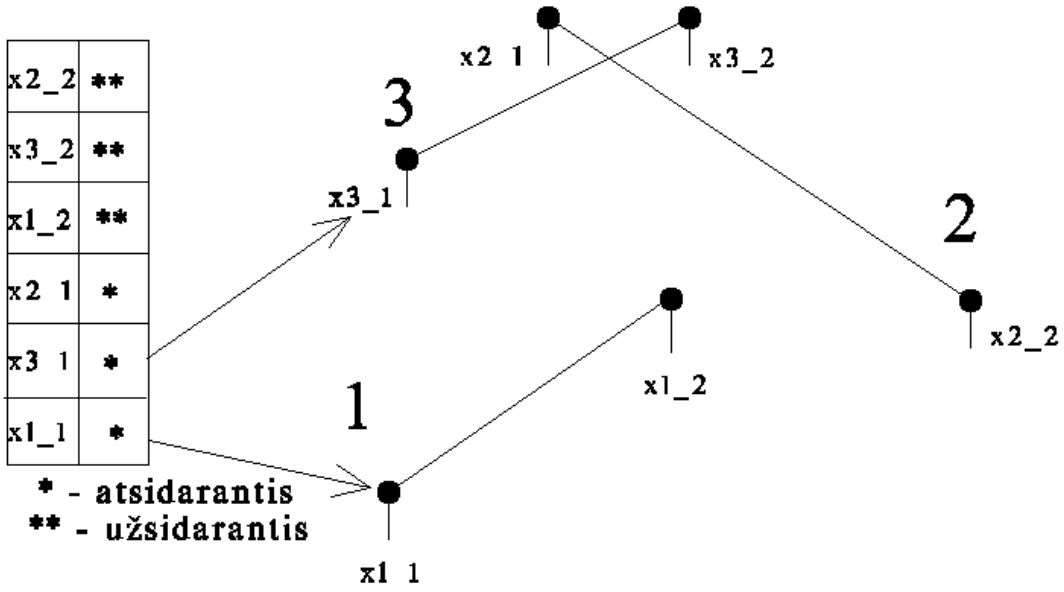
Šluojančios tiesės algoritmas susikirtimams skaičiuoti

Visi segmentai surūšiuojami *y* koordinatės (mažesnė segmento y koordinatė) didėjimo tvarka (16 pav.).



16 pav. Surūšiuoti pradiniai duomenys

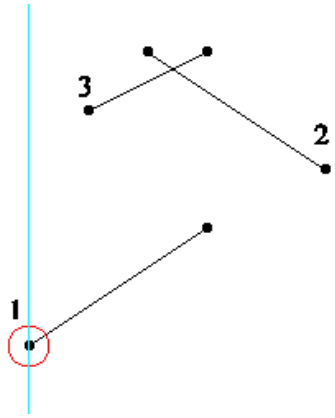
Į atskirą masyvą (*x* informacinis masyvas) sudedamos kiekvieno segmento pradžios ir pabaigos x koordinatės (prie kiekvienos *x* koordinatės taip pridedamas požymis – ar tai segmento pradžios/pabaigos *x* koordinatė).



17 pav. x informacinis masyvas

Algoritmas peržiūri visus *x* informacinio masyvo elementus. Jei elementas iš *x* informacinio masyvo yra duomenų segmento atsidarymo *x* koordinatė, tada tas segmentas įdedamas į darbinį

masyvą. Darbiniame masyve laikomi segmentai, kurių uždarymo koordinačių dar nepasiekė šluojanti tiesė.



18 pav. Vertikalios šluojančios tiesės pirmas įvykio taškas

Jei elementas iš x informacinio masyvo yra duomenų segmento užsidarymo x koordinatė, tai tas segmentas surandamas darbiniame masyve ir pašalinamas iš jo.

Susikirtimų skaičiavimas yra vykdomas tada, kai naujas segmentas yra įdedamas į darbinį masyvą.

Vertikali šluojanti tiesė, kurios statusas realizuojamas masyvu su pilnu perrinkimu

Šiuo atveju, susikirtimų paieška vykdoma su visais masyve tuo metu esančiais segmentais

V*ertikali šluojanti tiesė, kurios statusas realizuojamas masyvu su optimizuotu perrinkimu*

Darbiniame masyve elementai yra surūšiuoti didėjimo tvarka pagal mažesnę segmento y (y-low) koordinatę. Gavus naują segmentą, yra pradedama peržiūra nuo darbinio masyvo pradžios iki tol, kol pasiekiamas segmentas, kurio mažesnė y (y-low) koordinatė yra didesnė už naujo segmento didesnę y (y-high) koordinatę. Naujai atėjusį segmentą yra bandoma kirsti su visais prieš tai buvusiais segmentas.

Vertikali šluojanti tiesė, kurios statusas realizuojamas binariniu medžiu

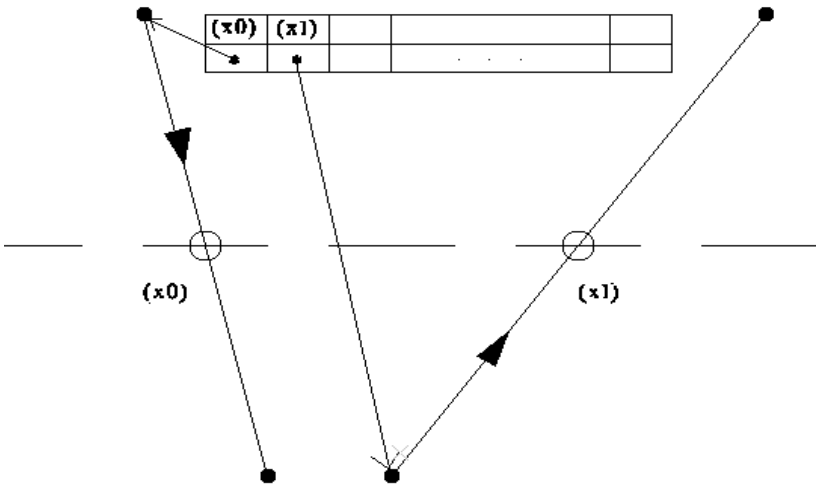
Naudojamas *red-black* save balansuojantis dvejetainis medis. Jis užpildomas segmentais iš pradinių duomenų. Šito tipo medis turi vieną esminį skirtumą lyginant su kitais dvejetainiais medžiais – kelias nuo šaknies iki be kurio įrašo medyje nėra ilgesnis už dvigubą atstumą iki artimiausio įrašo nuo šaknies. Pritaikius šią savybę, gauname stipriai balansuotą medį, todėl paieška jame atliekama labai greit. Tačiau dėl to įterpimo, išmetimo operacijos tampa sudėtingesnėmis.

1 lentelė. Binarinio medžio operacijų sudėtingumas

	Vidutiniškai	Blogiausiu atveju
Paieška	O(log n)	O(log n)
Įterpimas	O(log n)	O(log n)
Šalinimas	O(log n)	O(log n)

2.2.3 Intervalų formavimas iš gautų šluojančios tiesės ir segmentų susikirtimų

Per kiekvieną įvykio tašką juda horizontali šluojanti tiesė ir skaičiuojami šluojančios tiesės susikirtimai su duomenų segmentais. Atlikus skaičiavimus, gaunamas x koordinačių masyvas, kuriame kiekvienas x koordinatės įrašas turi nuorodą į duomenų segmentą, su kuriuo sukirtus šluojančią tiesę buvo gauta ši x koordinatė.



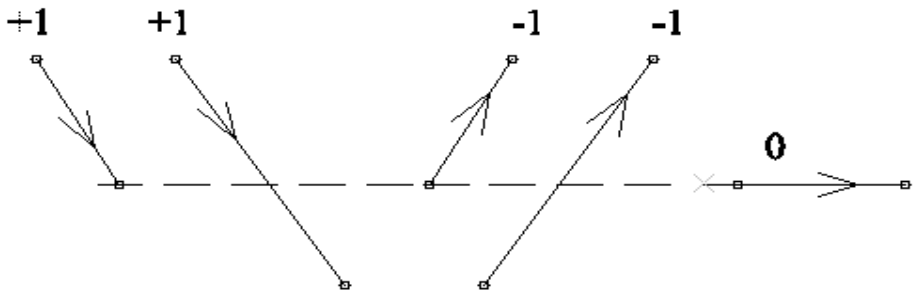
19 pav. x koordinačių masyvas

Yra formuojami du masyvai: virš ir po horizontalia šluojančia tiese.

Kiekvienas kirstas duomenų segmentas šluojančios tiesės atžvilgiu yra skirstomas taip :

- *įeinantis ir pasibaigiantis šluojančios tiesės kirstame taške*
- *įeinantis ir kertantis šluojančios tiesės kirtimosi taške*
- *kolinearūs šluojančiai tiesei linijai*
- *išeinantis ir prasidedantis šluojančios tiesės kirstame taške*

- *įeinantis ir kertantis šluojančios tiesės kirtimosi taške*



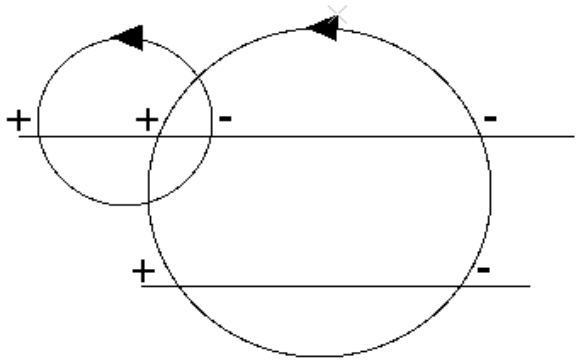
20 pav. Duomenų segmentų analizė šluojančios tiesės atžvilgiu

„+“ - įeinantis pasibaigiantis/kertantis duomenų segmentas šluojančios tiesės kirstame taške arba dar vadinamas krentančių

„-“ - išeinantis prasidedantis/kertantis duomenų segmentas šluojančios tiesės kirstame taške arba dar vadinamas kylančiu

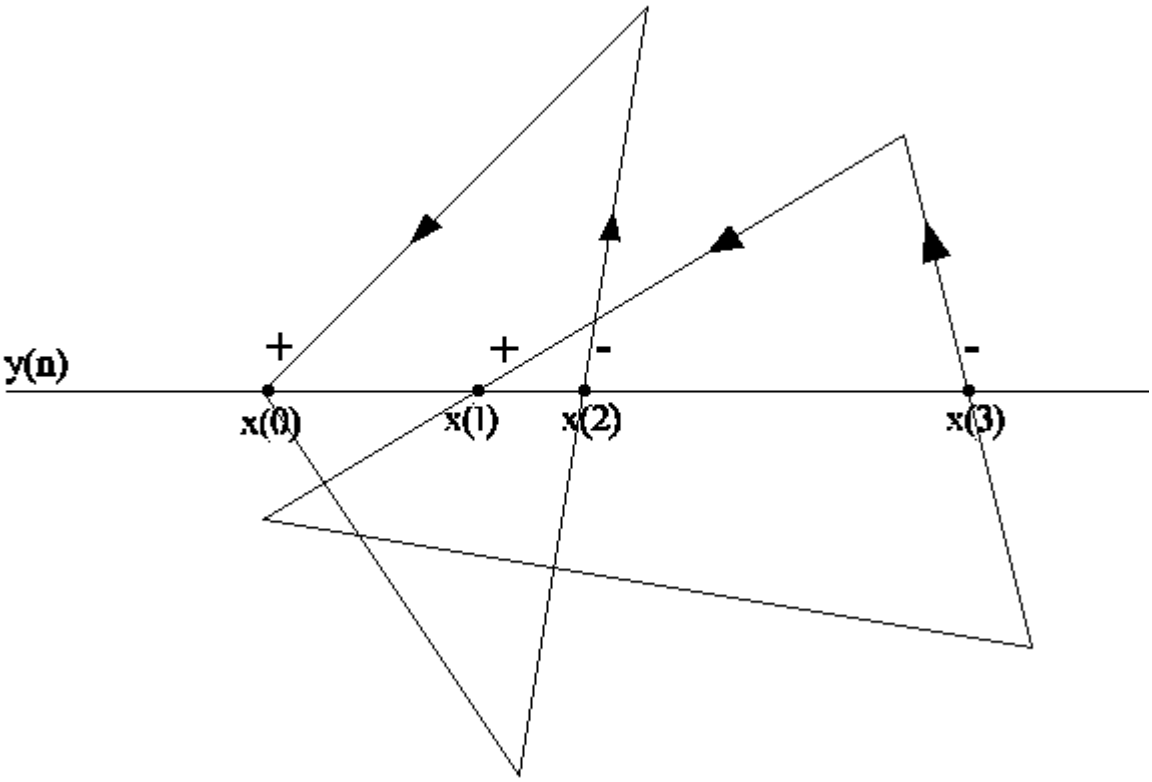
Turint šituos duomenis, yra formuojami intervalai virš ir po šluojančia tiese. Skaičiavimai pradedami nuo kairės pusės. Yra naudojamas skaitliukas, kuris priklausomai nuo to koks intervalas (kylantis ar krentantis), yra didinamas arba mažinamas.

Intervalų skaičiavimo algoritmui pradžioje yra užduodamas skaičius, kuri pasiekus fiksuojama intervalo pradžia ar pabaiga. Sujungimo atveju skaičius yra „0“, sankirtos - „1“.



21 pav. Duomenų segmentų pobūdis
šluojančios tiesės atžvilgiu kirstuose taškuose

Toliau bus išnagrinėjami du skaičiavimo pavyzdžiai – intervalų formavimas virš šluojančios tiesės, atliekant plokščiųjų geometrinių figūrų sujungimą ir sankirtą (abiem atvejais, skaičiavimai bus atlikti vienam pavyzdžiui pavaizduotam 22 pav.).



22 pav. Šluojančios tiesės susikirtimai y(n) koordinatėje

Kaip buvo minėta, skaičiuojant plokščiųjų geometrinių figūrų sujungimą, intervalo pradžia ar pabaiga fiksuojama tada, kai sumuojant bei atiminėjant krentančius ir kylančius segmentus gaunamas nulis.

Algoritmas analizę pradeda nuo susikirtimo, turinčio mažiausią x koordinatę (22 pav., x(0) koordinatėje). Algoritmo darbinis skaitliukas, skaičiuojantis kylančius ir krentančius segmentus, pradžioje yra lygus nuliui. Šluojanti tiesė koordinatėje x(0) kerta krentantį segmentą. Gavus krentantį segmentą, algoritmas pirmiausia patikrina ar darbinis skaitliukas nėra lygus užduotam parametrui, nurodančiam kada fiksuot intervalus. Kadangi skaitliukas ir parametras yra lygūs, x(0) koordinatėje suformuojama intervalo pradžia (kadangi kirstas segmentas yra krentantis) bei darbinis skaitliukas yra padidinamas (jo reikšmė lygi „1“).

Koordinatėje x(1) kertamas vėl krentantis segmentas, atliekamas patikrinamas ar darbinis skaitliukas lygus užduotam skaičiui, nurodančiam fiksuot intervalo pradžią/pabaigą. Kadangi skaitliuko ir parametro reikšmės nelygios, šis taškas praleidžiamas, o darbinis skaitliukas padidinamas (jo reikšmė lygi „2“).

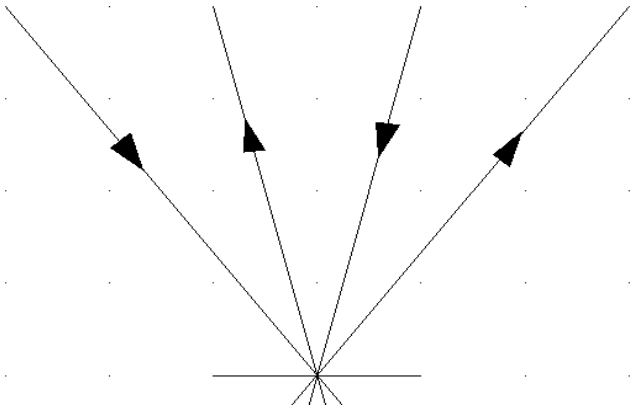
Koordinatėje x(2) šluojanti tiesė kerta kylantį segmentą. Kirtus kylantį segmentą, pirmiausia yra sumažinamas darbinis skaitliukas (jo reikšmė pasidaro lygi „1“) ir tada daromas lyginimas su

parametru, nurodančiu ar fiksuoti intervalo pradžią ir pabaigą. Kadangi jie nelygūs, šis taškas yra praleidžiamas.

Koordinatėje x(3) šluojanti tiesė kerta kylantį segmentą, todėl darbinis skaitliukas vėl yra sumažinamas (jo reikšmė pasidaro lygi „0“). Šiame etape jie yra lygūs ir, kadangi kirstas segmentas yra kylantis, čia fiksuojama intervalo pabaiga. Taigi, įvykio taške y(n) yra gaunamas toks intervalas – [x(0), x(3)].

Identiški skaičiavimai atliekami ir skaičiuojant plokščiųjų geometrinių figūrų sankirtą, tik intervalų pradžios ir pabaigos fiksavimo parametras yra lygus „1“. Atlikus skaičiavimus gaunamas intervalas – [x(1), x(2)], kuris toliau bus naudojamas formuoti galutinį sankirtos rezultatą.

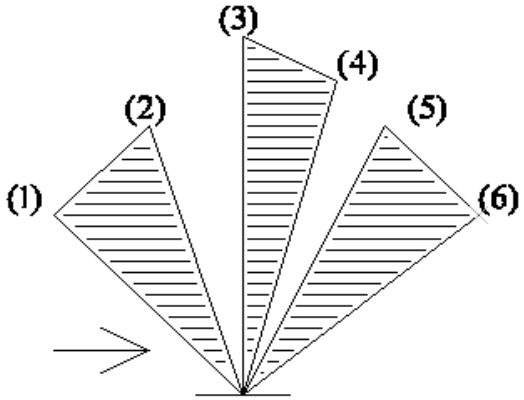
Kritinė situacija gaunasi tada, kai vienoje x koordinatėje kertasi daugiau nei vienas segmentas. Nors ir yra vienoda x koordinatė, kiekvienam iš tokių kirtimų yra formuojamas atskiras įrašas rezultatų masyve.



23 pav. Kelių segmentų susikirtimas viename taške

Surinkus visus duomenis, atliekamas papildomas jų apdorojimas:

- nustatomos vietos, kuriose vieną x koordinatę kerta daugiau nei vienas segmentas;
- duomenys tose vietose yra specialiai surūšiuojami: į priekį dedami tie segmentai, kurie einant iš kairės pusės būtų sutinkami pirmiausia

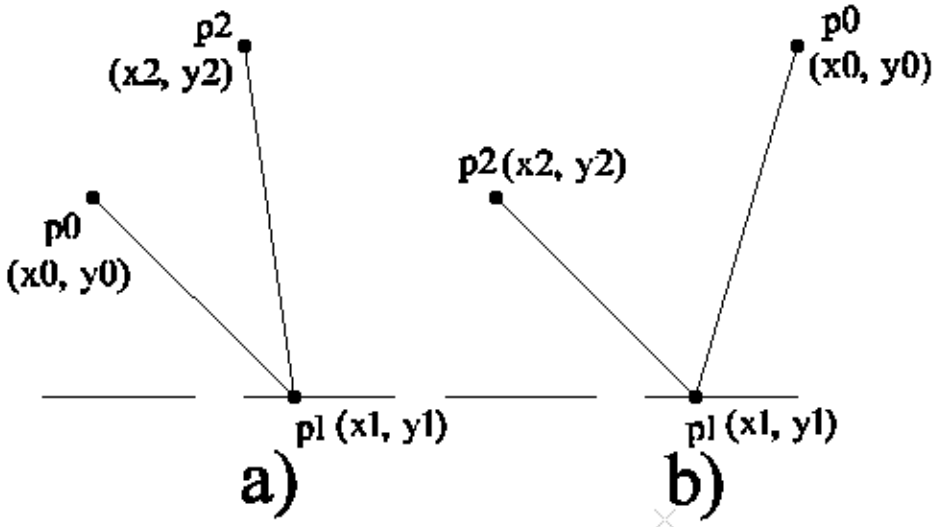


24 pav. Teisingas segmentų surūšiavimas

Pateiktame 24 pav. parodyta, kokia tvarka turi būti surūšiuoti šeši segmentai, besikertantys/besiliečiantys viename taške. Visos kitokios segmentų išsidėstymo tvarkos neleistų teisingai suformuoti intervalus, o vėliau ir galutinį rezultatą.

„Kairiau“ stovinčiam segmentui nustatyti naudojamas sekantis algoritmas.

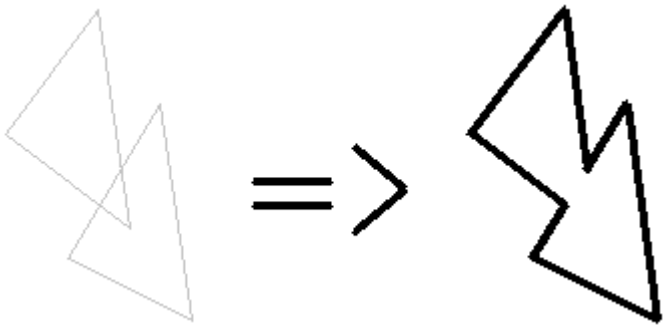
Norint nustatyti, kokį posūkį sudaro trys taškai („posūkį į kairę“ ar „posūkį į dešinę“) visai nereikia skaičiuoti kampo reikšmės tarp dviejų linijos segmentų, o pakanka panaudoti paprasčiausius aritmetinius veiksmus. Trims taškams (x0,y0), (x1,y1) ir (x2,y2) reikia suskaičiuoti dviejų vektorių (x0,y0), (x1,y1) ir (x0,y0), (x2,y2) *cross product* kryptį, aprašomą išraiškos (x1 - x0)(y2 - y0) - (y1 - y0)(x2 - x0) rezultatu ir jo ženklu. Jei rezultatas yra lygus nuliui, tai trys taškai yra kolinearūs, jei teigiamas – trys taškai sudaro „posūkį į kairę“, jei neigiamas - trys taškai sudaro „posūkį į dešinę“. 25 pav. pateikti 2 pavyzdžiai iliustruojantys „posūkį į kairę“ (25a pav.) ir „posūkį į dešinę“ (25b pav.).



25 pav. Segmentų pozicijos vienas kito atžvilgiu

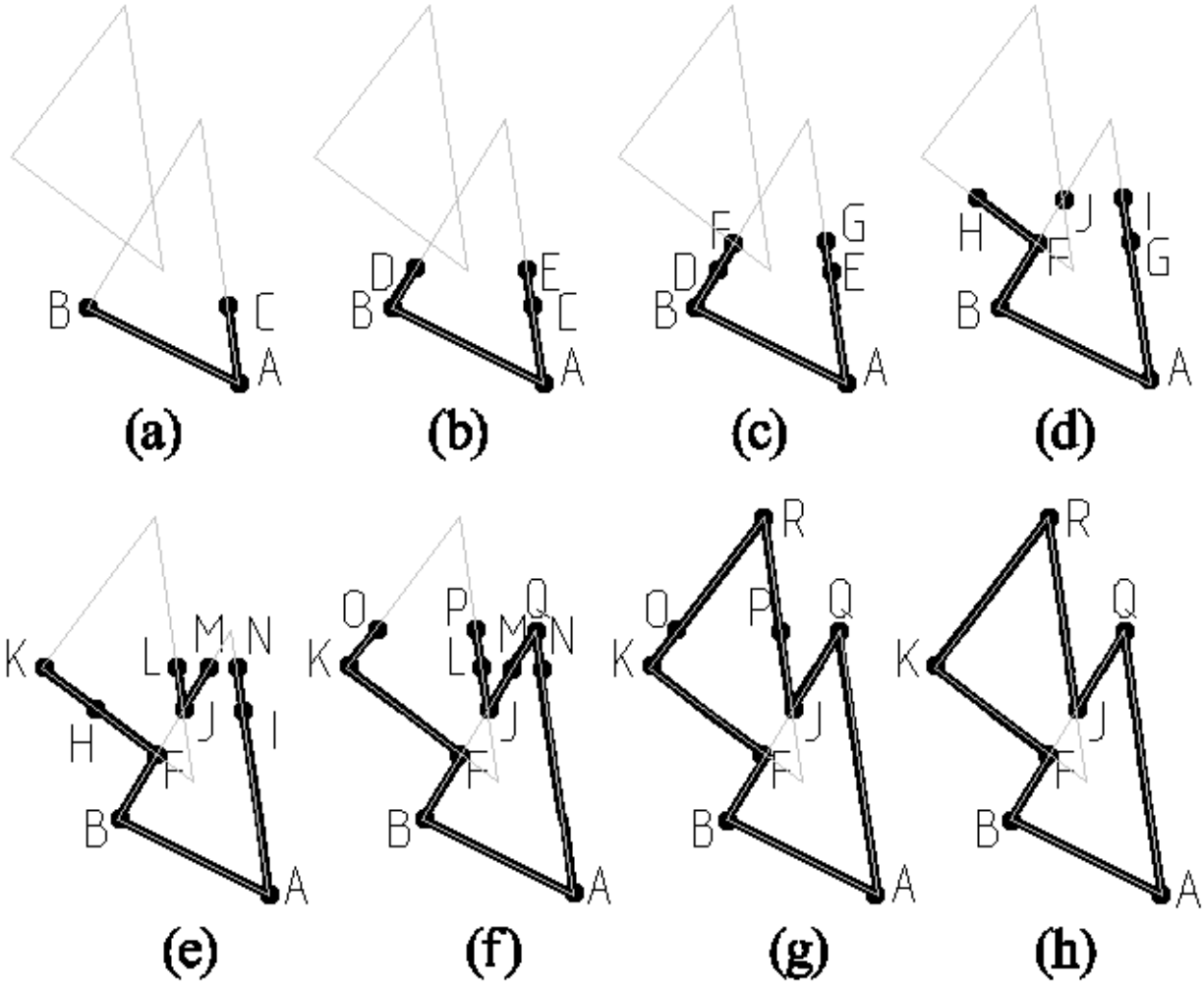
2.2.4 Galutinio rezultato skaičiavimas

Algoritmas, naudojamas galutinio rezultato formavimui, remiasi paskaičiuotų intervalų analize. Intervalai yra paskaičiuojami kiekviename įvykio taške.



26 pav. Pradiniai duomenys ir gauti rezultatai

Toliau pavaizduojama galutinių rezultatų dinamika.



27 pav. Galutinio rezultato kūrimas

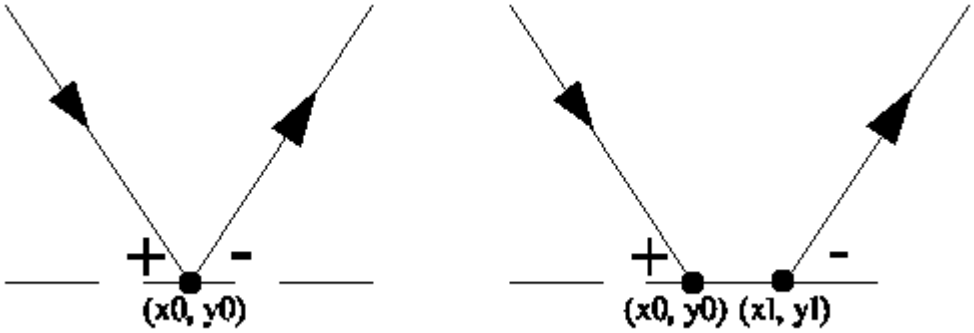
Gavus susikirtimo tašką, yra nagrinėjama informacija virš ir po šluojančia tiese (kiekvienas gautas susikirtimo taškas yra ant šluojančios tiesės). Detaliai kūrimo eiga parodyta 27 pav.:

- šluojanti tiesė darbą pradeda nuo taško *A* (27 pav.), ir kertant šluojančią tiesę su pradinių duomenų segmentais gaunami kirtimai taške *A* ir šituo atveju į galutinius rezultatus įtraukiami 2 segmentai su koordinatėmis *[(∞,A), (A,∞)]* (pirmo segmento pradžios ir antro segmento pabaigos koordinatės bus tikslinamas vėliau – šioje stadijoje algoritmas dar negali nustatyti tikslių jų koordinačių)
- šluojanti tiesė toliau keliauja iki sekančio įvykio taško, bei gauna 2 kirtimus *B* ir *C* taškuose (27(*a*) pav.). Galutinis rezultatas yra papildomas 2 naujais segmentais ((*B,A*) ir (*A,C)*) ir gaunamas rezultatas [(∞,*B*),(*B,A*), (*A,C)* (*C*,∞)]
- šluojančiai tiesei pasiekus sekantį įvykio tašką, gaunami 2 kirtimai taškuose *D* ir *E* (27(*b*) pav.). Taškas *E* ir galutiniam rezultate jau įtrauktas taškas *C* priklauso tam pačiam duomenų segmentui, todėl taškai nėra dubliuojami, ir galutiniame rezultate

taškas *C* yra pakeičiamas tašku *E*. Į galutinių rezultatų masyvą įtraukiamas naujas segmentas *(D, B)*. Po šito etapo gauname 5 segmentus - *[(∞,D),(D, B),(B,A),* (*A,E)* (*E*,∞)]

- toliau gaunami kirtimo taškai *F* ir *G* (27(*c*) pav.). Kirtimosi taškas *F* ir prieš tai įdėtas taškas *D* priklauso tam pačiam duomenų segmentui, tai taškas *D* galutiniam rezultate yra pakeičiamas tašku *F* (identiška situacija gaunasi ir su taškais *G* ir *E*). Šiame šluojančios tiesės etape naujais segmentais galutinis rezultatas nėra papildomas. Atnaujinamos tik taškų koordinatės - *[(∞,F),(F, B),(B,A),* (*A,G)* (*G*,∞)]
- šluojančiai tiesei nukeliavus į sekantį įvykio tašką, gaunami kirtimai taškuose – *H, J* ir *I* (27(*d*) pav.). Taškas *I* ir galutiniuose rezultatuose taškas *G* priklauso tam pačiam duomenų segmentui, tai taškas *G* galutiniam rezultate yra pakeičiamas tašku *I.* Taške *J* gauti susikirtimai suformuoja 2 segmentus, kurie dedami į atskirą masyvą. Po šio apdorojimo papildomas pirmasis galutinių rezultatų masyvas vienu segmentu bei sukuriamas naujas masyvas su dviem segmentais - *[(∞,H),(H,F),(F, B),(B,A),* (*A,I)* (*I*,∞)] (pirmasis masyvas), [(∞,*J*), (*J*,∞)] (antrasis masyvas)
- sekančiame žingsnyje (27(*e*) pav.) šluojanti tiesė gauna susikirtimus taškuose *K, L, M* ir *N.* Kirtimosi taškas *K* ir prieš tai įdėtas taškas *H* pirmajame galutinių rezultatų masyve priklauso tam pačiam duomenų segmentui, tai taškas *H* galutiniam rezultate yra pakeičiamas tašku *K* (identiška situacija gaunasi ir su taškais *N* ir *I*). Tuo tarpu antrasis masyvas yra papildomas dviem naujais segmentais – *(L, J)* ir *(J, M)*. Galutinių rezultatų masyvai - *[(∞,K),(K,F),(F, B),(B,A),* (*A,N)* (*N*,∞)], [(∞,*L*), (*L*,*J*), (*J,M*), (*M*,∞)].
- kai šluojanti tiesė pasiekia sekantį įvykio tašką, gaunami šluojančios tiesės ir duomenų segmentų susikirtimai taškuose – *O, P, Q* (27(*f*) pav.). Pirmasis masyvas iš galutinių rezultatų yra papildomas segmentu – *(O,K).* Taškas *N* iš pirmojo masyvo, yra pakeičiamas tašku *Q*, nes abu taškai gauti kirtus tą patį duomenų segmentą. Tokie pat pakeitimai yra padaromi su taškais *L* ir *P* bei *M* ir *Q* iš galutiniuose rezultatuose turimo antrojo masyvo. Šio etapo pabaigoje yra padaromas abiejų masyvų apjungimas į vieną. Gautas rezultatas yra - *[(∞,O),(O,K)(K,F),(F, B),(B,A), (A,Q), (Q,J),(J,P), (P,∞)]*
- toliau šluojanti tiesė pasiekia paskutinį įvykio tašką, ir gauna kirtimus taške R (27(g) pav.). Kirtimosi taškas *R* ir prieš tai įdėtas taškas O priklauso tam pačiam duomenų segmentui, tai taškas *O* galutiniam rezultate yra pakeičiamas tašku *R* (identiška

situacija gaunasi ir su taškais *P* ir *R)*. Šiame etape segmentai iš galutinio rezultato masyvo yra sujungiami į vieną uždarą grandinėlę ir gaunamas toks rezultatas - *[(R,K)(K,F),(F, B),(B,A), (A,Q), (Q,J),(J,R)]*



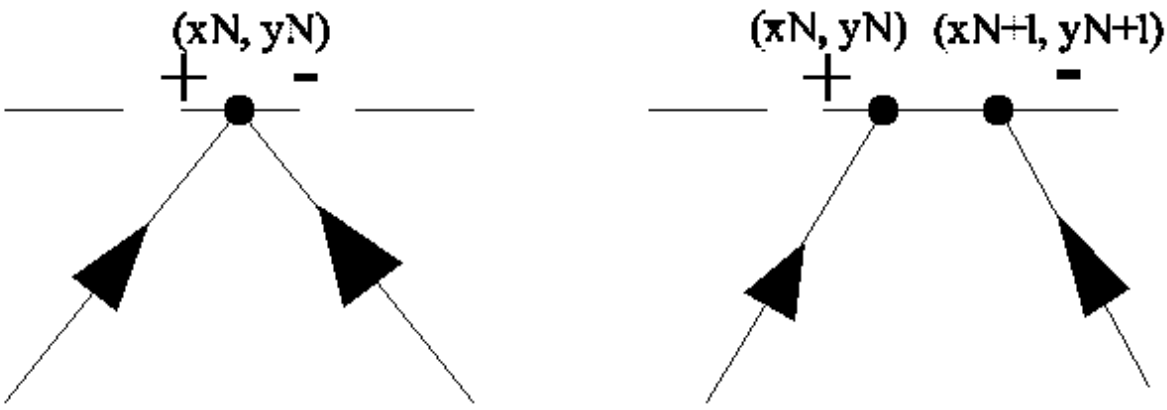
28 pav. Intervalų analizė virš šluojančios tiesės

Šiuo atveju (28 pav.):

- virš šluojančios tiesės yra suformuojamas vienas intervalas (vienu atveju $x_0 = x_1$, o kitu atveju $x_0 \neq x_1$).
- po šluojančia tiese jokios informacijos nėra suskaičiuojama

Abiem pavaizduotais atvejais, paruošti intervalai yra vienodi, tačiau algoritmas į galutinių rezultatų masyve pirmu atveju įdeda du segmentus *[(∞,(x0, y0)), ((x0, y0),∞)]*, o antru atveju (kai $x_0 \neq x_1$) - tris segmentus *[(∞,(x0, y0)), ((x0, y0), (x1, y1)), ((x1, y1),∞)]*.

Kitas atvejas (29 pav.) yra gaunamas tada, kai virš šluojančios tiesės nėra fiksuojama jokių intervalų, kai tuo tarpu po šluojančia tiese yra suformuojamas.

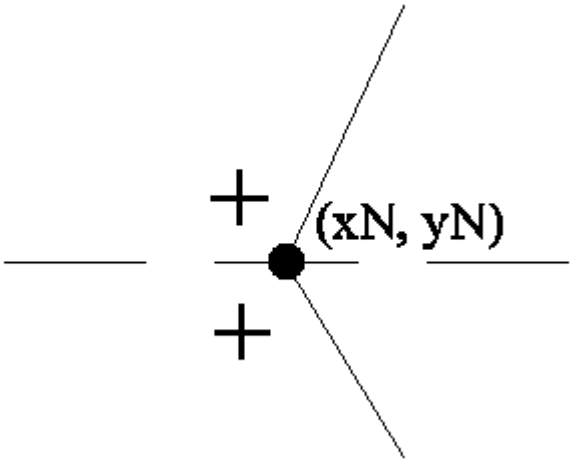


29 pav. Intervalų analizė po šluojančia tiese

Galutiniam rezultate, kai $xN \neq xN+1$, algoritmas įdės vieną papildomą segmentą – *(xN, yN)(xN+1, yN+1)*, o priešingu atveju – naujų segmentų nekuriama ir atskiri segmentai sujungiami į uždarą grandinę.

Išskiriamas dar vienas atvejas (30 pav.):

- virš šluojančios tiesės yra intervalo atsidarymas
- po šluojančia tiese yra intervalo atsidarymas
- šluojanti tiesė kerta dviejų segmentų susijungimo tašką



30 pav. Intervalų analizė virš ir po šluojančia tiese

Šitoje situacijoje galutiniam rezultate įdedamas vienas naujas segmentas.

3 Eksperimentinė dalis

3.1 Eksperimento aprašymas

Šiame darbe eksperimentinę dalį sudaro:

1. Tarpusavio segmentų susikirtimų sąrašo skaičiavimo tyrimas

   Šiame tyrime bus analizuojami ir lyginami trijų skirtingų algoritmų kokybiniai rodikliai ir jų greitis.Segmentų tarpusavio susikirtimams naudojami sekantys algoritmai:

   - vertikali šluojanti tiesė, kurios statusas realizuojamas masyvu su pilnu perrinkimu jame
   - vertikali šluojanti tiesė, kurios statusas realizuojamas masyvu su optimizuotu perrinkimu jame
   - vertikali šluojanti tiesė, kurios statusas binariniu medžiu

   Pagrindinis šio tyrimo tikslas – nustatyti ir ištirti skaičiavimų greičio priklausomybę nuo

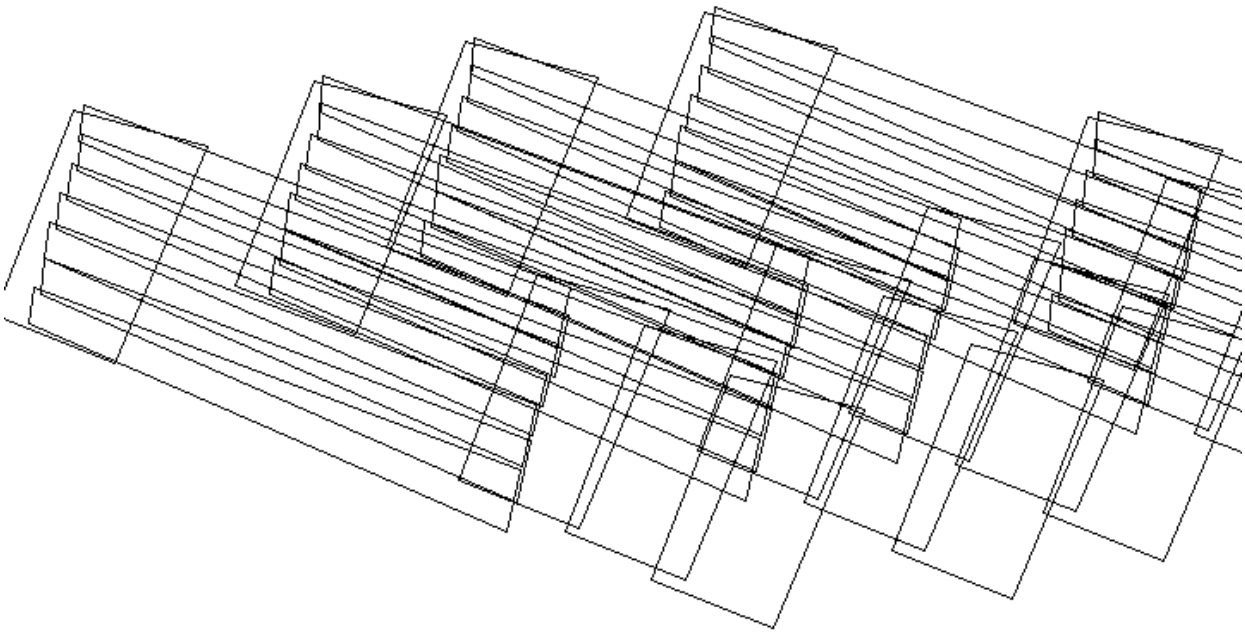
pradinių duomenų segmentų kiekio. Taip pat ištirti skaičiavimų greičio priklausomybę nuo pradinių duomenų segmentų išsidėstymo vienas kito atžvilgiu, t.y. nustatyti įtaką skaičiavimo greičiui, kai pradinių duomenų segmentai visai nesikerta tarpusavyje, kai pradinių duomenų segmentai kertasi tarpusavyje po kelis ir daugiau kartų.

2. Duomenų segmentų ir šluojančios tiesės susikirtimų tyrimas
   Kaip ir aukščiau aprašytame tyrime, šiame tyrime bus nagrinėjami trys algoritmai – binarinio medžio, šluojančios tiesės ir pilno perrinkimo algoritmai.
3. Plokščiųjų geometrinių figūrų sujungimo, sankirtos ir atėmimo skaičiavimų tyrimas
   Šiame tyrime bus lyginama darbe pateikto algoritmo rezultatai ir darbo greitis su algoritmo, naudojančio skenavimo liniją, rezultatais ir skaičiavimu laiku. Taip pat palyginama abiejų algoritmų skaičiavimų greičio priklausomybė nuo pradinių duomenų kiekio,

3.2 Tarpusavio segmentų susikirtimų skaičiavimų tyrimas

Tyrimams atlikti bus naudojami trys skirtingi pradinių duomenų rinkiniai:

1) pirmas rinkinys



31 pav. Pradinių duomenų fragmentas iš pirmojo rinkinio

Pirmąją pradinių duomenų aibę sudaro:

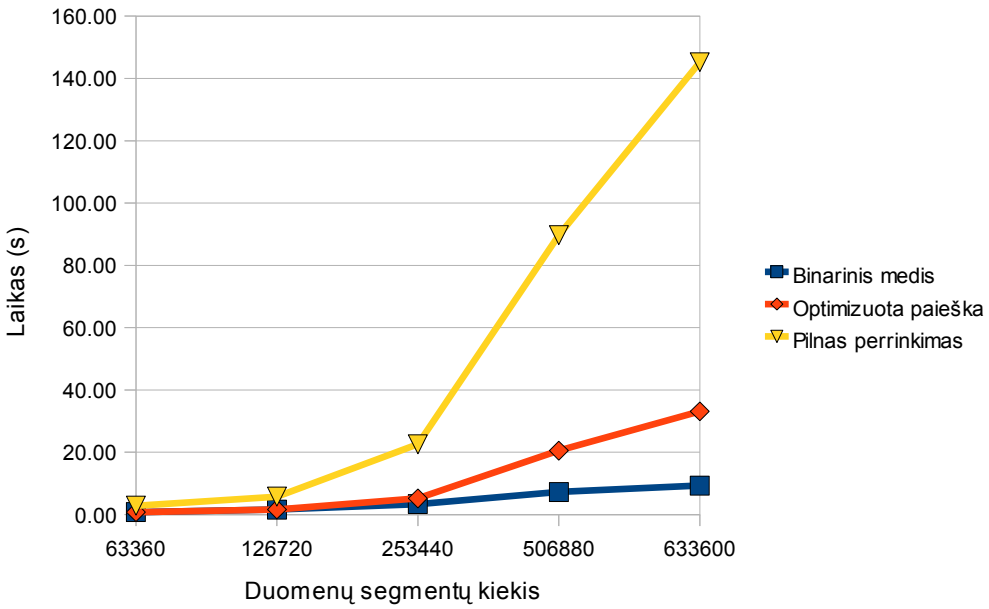
- 63 360 segmentai
- 186 943 segmentų tarpusavio susikirtimai.

Vaizdinis fragmentas parodytas 31 pav. Kitos duomenų aibės šiame tyrime paruošiamos naudojantis pirmąja duomenų aibe – visi duomenys iš pirmosios duomenų aibės yra pakartojami 2,

4 ir t.t. kartus (pradinių duomenų kopijos yra daromos tokios, kurios rezultate kopijos nepersidengia su prieš tai buvusiais duomenų segmentais).

2 lentelė. Segmentų susikirtimo skaičiavimo laikai gauti naudojant pirmąjį duomenų rinkinį

Duomenų segmentų kiekis	Binarinis medis, laikas(s)	Optimizuota paieška, laikas(s)	Pilnas perrinkimas, laikas(s)
63360	0,78	0.8	2,84
126720	1,63	1,65	5,75
253440	3,36	5,21	22,59
506880	7,28	20.53	89,66
633600	9,33	33,15	145,22



32 pav. Susikirtimų skaičiavimo laikai naudojant pirmąjį duomenų rinkinį

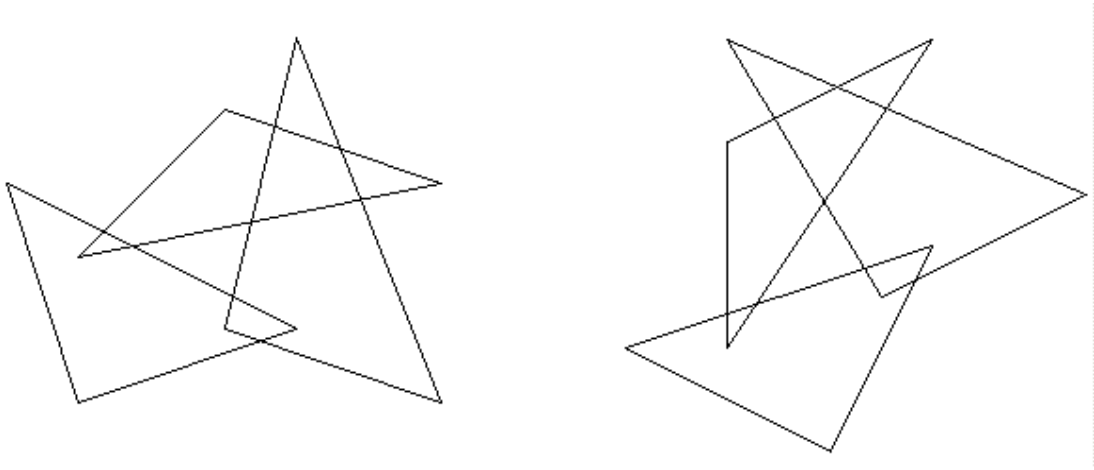
Visi trys algoritmai suskaičiuoja vienodą susikirtimų skaičių. Lentelėje 2 pateikti skaičiavimų laikai. Didėjant pradinių duomenų kiekiui išryškėja algoritmo, naudojančio binarinį medį, greičio pranašumas prieš kitus du algoritmus.

2) antras rinkinys

Antrąją pradinių duomenų aibę sudaro:

- 90 000 segmentai
- šie pradiniai duomenys turi 80 000 tarpusavio susikirtimų.

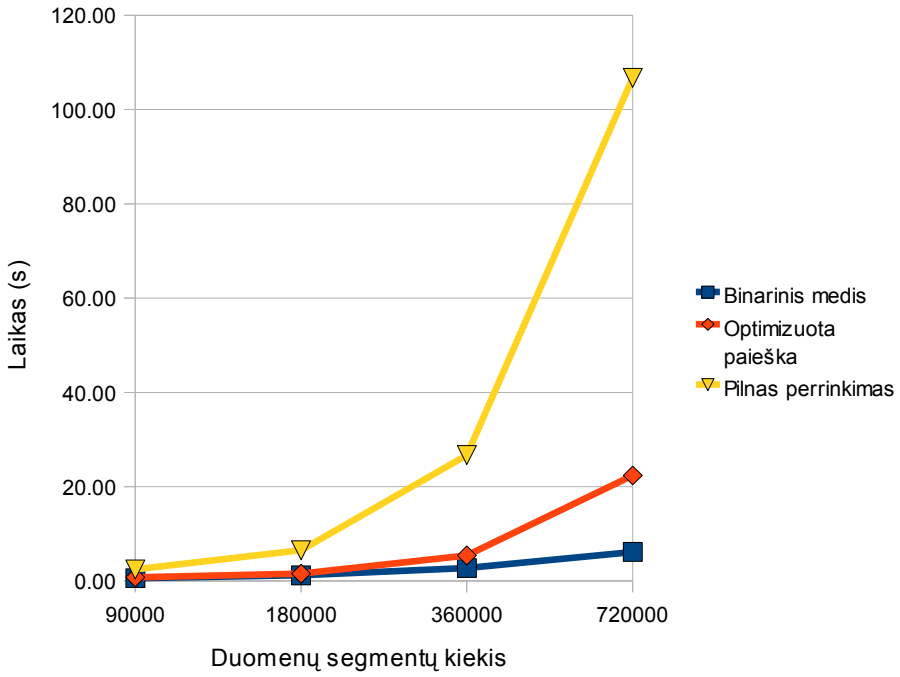
Kaip ir eksperimentinėje dalyje su pirmuoju rinkiniu, šiame tyrime padaromos duomenų kopijos, siekiant stebėti algoritmų skaičiavimų priklausomybę nuo pradinių duomenų kiekio



33 pav. Pradinių duomenų fragmentas iš antrojo rinkinio

3 lentelė. Segmentų susikirtimo skaičiavimo laikai gauti naudojant antrąjį duomenų rinkinį

Duomenų segmentų kiekis	Binarinis medis, laikas(s)	Optimizuota paieška, laikas(s)	Pilnas perrinkimas, laikas(s)
90000	0.59	0,76	2,45
180000	1,19	1,53	6,51
360000	2,73	5,41	26,70
720000	6,15	22,37	106,68

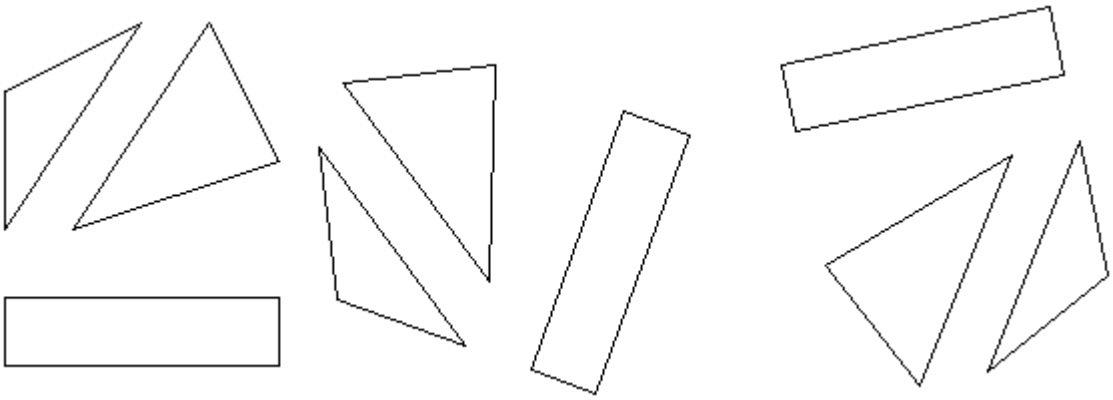


34 pav. Susikirtimų skaičiavimo laikai naudojant antrąjį duomenų rinkinį

3) trečiasis rinkinys

Trečiąją pradinių duomenų aibę sudaro:

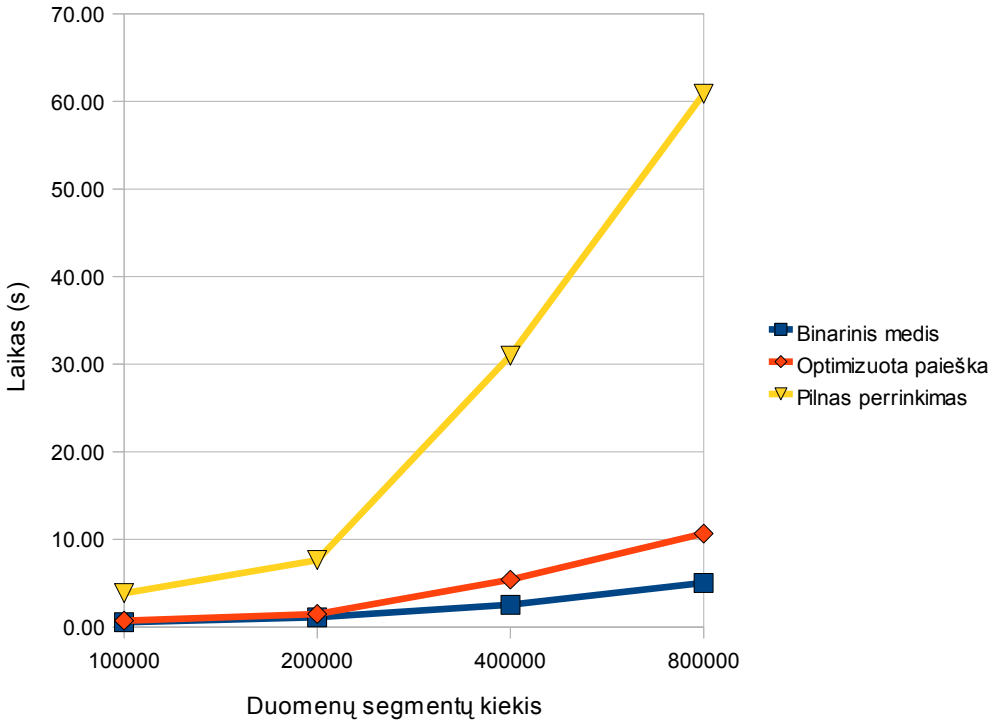
- 100 000 segmentai
- šie pradiniai duomenys tarpusavyje nesikerta.



35 pav. Pradinių duomenų fragmentas iš trečiojo rinkinio

4 lentelė. Segmentų susikirtimo skaičiavimo laikai gauti naudojant trečiąjį duomenų rinkinį

Duomenų segmentų kiekis	Binarinis medis, laikas(s)	Optimizuota paieška, laikas(s)	Pilnas perrinkimas, laikas(s)
100000	0,54	0,73	3,86
200000	1,10	1,49	7,65
400000	2,54	5,41	30,98
800000	5,03	10,66	60,89



36 pav. Susikirtimų skaičiavimo laikai naudojant trečiąjį duomenų rinkinį

3.3 Duomenų segmentų ir šluojančios tiesės susikirtimų tyrimas

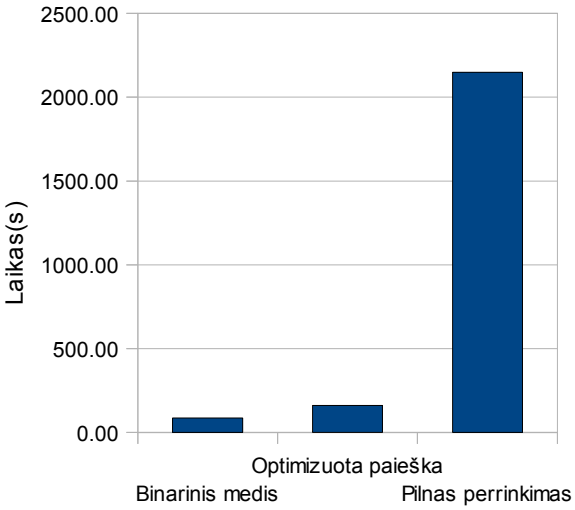
Šiame tyrime bus naudojami skyriuje „*Tarpusavio segmentų susikirtimų skaičiavimų tyrimas*“ aprašytos trys pradinių duomenų aibės:

- pirmas rinkinys su 63 360 segmentų, kurie tarpusavyje kertasi 186 943 kartus (vaizdinis fragmentas parodytas 31 pav.)
- antrasis rinkinys su 90 000 segmentų, kurie turi 80 000 tarpusavio susikirtimų (vaizdinis fragmentas parodytas 33 pav.)

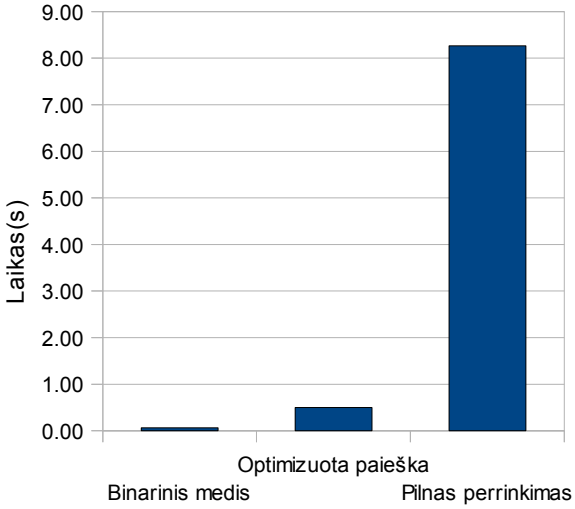
- trečiasis rinkinys su 100 000 segmentų, kurie tarpusavyje nesikerta (vaizdinis fragmentas parodytas 35 pav.)

Kiekvienam iš pradinių duomenų rinkinių yra suskaičiuojami visi įvykių taškai (viršūnės ir susikirtimo taškai), per kuriuos judės šluojanti tiesė ir ieškos susikirtimų su duomenų segmentais.

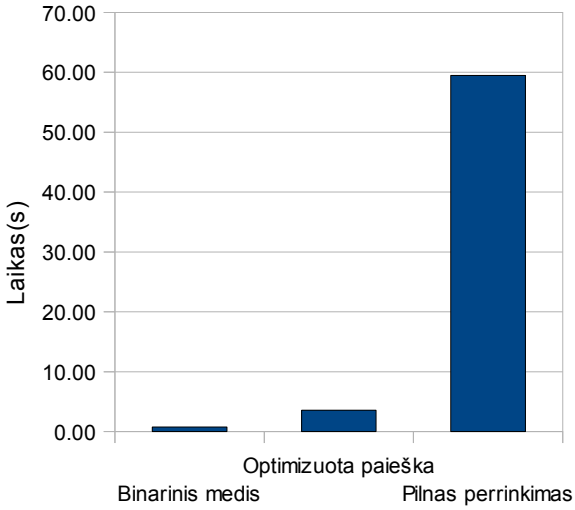
Tyrimo eigoje nustatyta, kad visi trys algoritmai pateikia vienodus kirtimų skaičius kiekvienam įvykio taške. Tačiau labai skiriasi jų suskaičiavimo laikai – didėjant įvykio taškų kiekiui kitus algoritmus stipriai lenkia šluojančios tiesės algoritmas, kurios statusas realizuotas binariniu medžiu. Toliau pateikiami skaičiavimų greičių grafikai bei bendra lentelė su skaičiavimų laikais.



37 pav. Duomenų segmentų ir šluojančios tiesės susikirtimų skaičiavimų laikai su pirmuoju duomenų rinkiniu



38 pav. Duomenų segmentų ir šluojančios tiesės susikirtimų skaičiavimų laikai su antruoju duomenų rinkiniu



39 pav. Duomenų segmentų ir šluojančios tiesės susikirtimų skaičiavimų laikai su trečiuoju duomenų rinkiniu

5 lentelė. Bendras algoritmų skaičiavimų laikų sąrašas

Duomenų segmentų kiekis	Binarinis medis, laikas(s)	Optimizuota paieška, laikas(s)	Pilnas perrinkimas, laikas(s)
63360	86,27	160,98	2148,32
100000	0,06	0,50	8,27
90000	0,73	3,55	59,49

3.4 Operacijų su plokščiosiomis geometrinėmis figūromis tyrimas

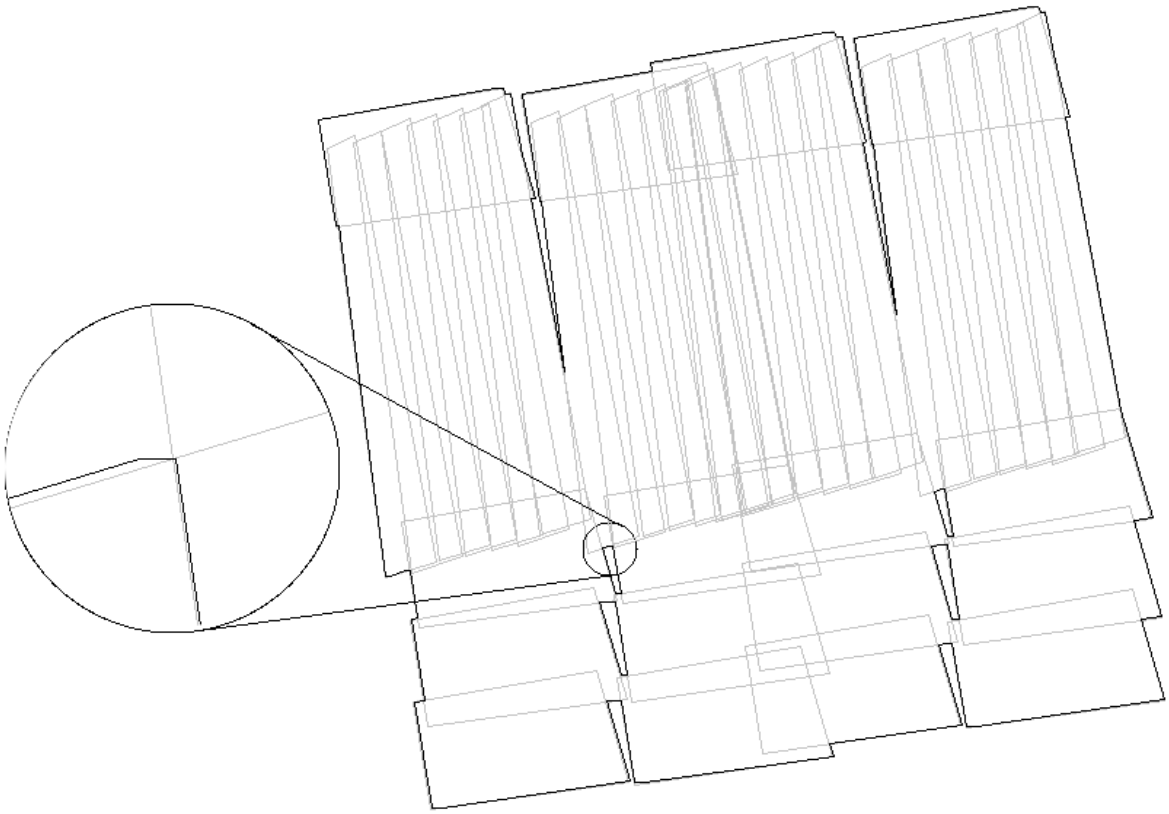
Šiame tyrime bus naudojamos skyriuje „Tarpusavio segmentų susikirtimų skaičiavimų tyrimas“ aprašytos trys pradinių duomenų aibės:

- pirmas rinkinys su 63 360 segmentų, kurie tarpusavyje kertasi 186 943 kartus (vaizdinis fragmentas parodytas 31 pav.)
- antrasis rinkinys su 90 000 segmentų (vaizdinis fragmentas parodytas 32 pav.)
- trečiasis rinkinys su 100 000 segmentų (vaizdinis fragmentas parodytas 33 pav.)

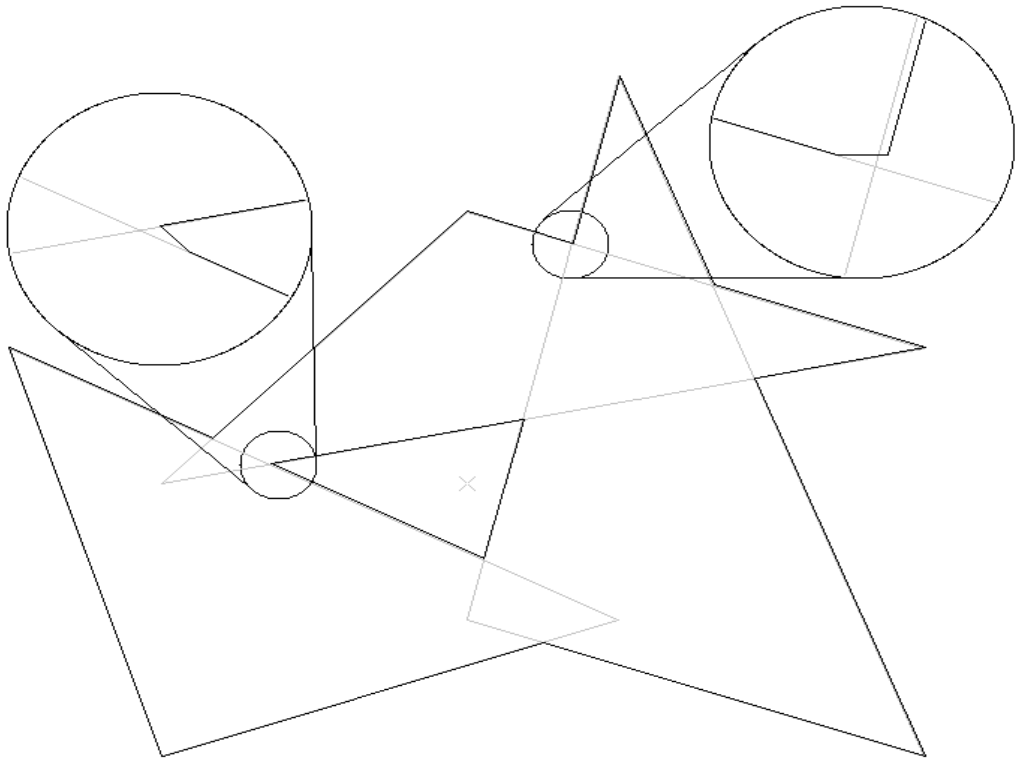
Eksperimentas bus atliekamas su dviem algoritmais – algoritmas, paremtas skenavimo linija, ir algoritmas, sudarytas ir aprašytas šiame darbe (paremtas horizontalia šluojančia tiese). Eksperimento eigoje bus tiriami operacijų su plokščiosiomis geometrinėmis figūromis rezultatai bei skaičiavimų greitis.

40 ir 41 pav. pavaizduoti galutiniai plokščiųjų geometrinių figūrų sujungimo rezultatai, gauti naudojant skenavimo algoritmą. Abiejuose paveiksluose yra išdidintos kritinės rezultatų zonos tikslumui įvertinti (pilka spalva parodyti pradiniai duomenys, o juoda spalva – galutinis rezultatas). Abiem atvejais galima pastebėti, kad galutinis rezultatas nėra visiškai tikslus palyginus su pradinių duomenų segmentais.

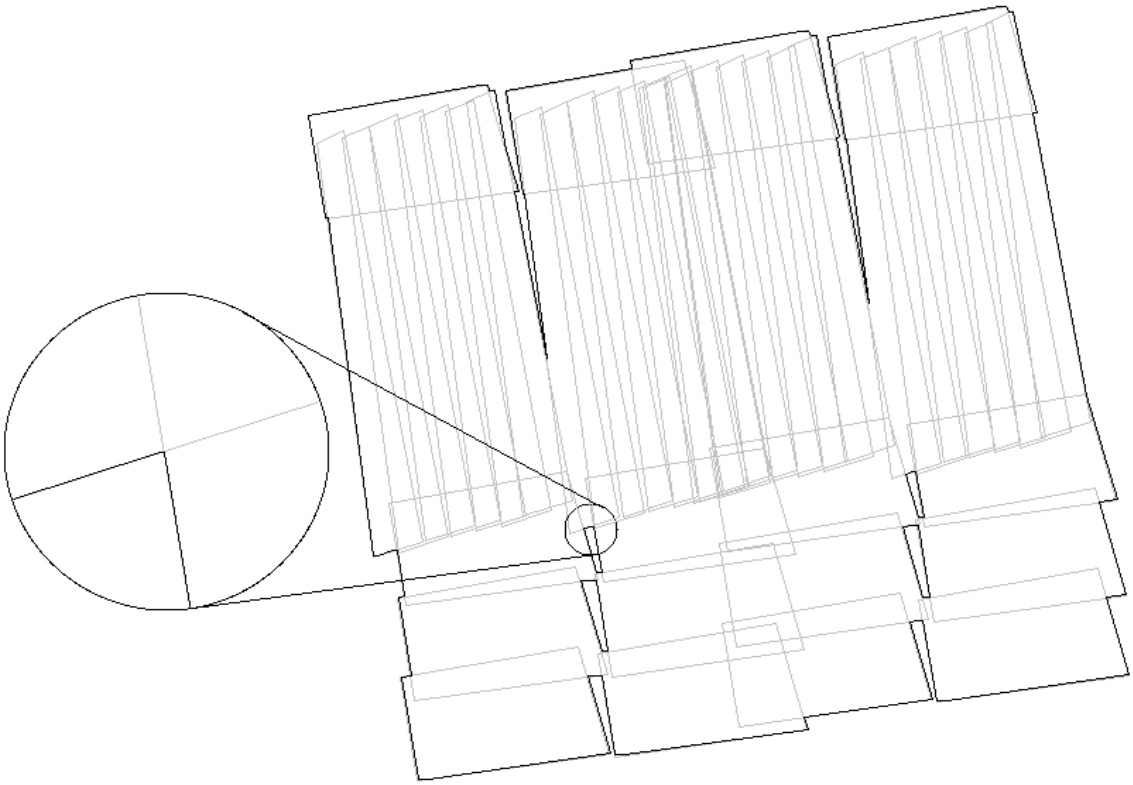
Tuo tarpu 42 ir 43 pav. pavaizduoti galutiniai plokščiųjų geometrinių figūrų sujungimo rezultatai, gauti naudojant šluojančios tiesės algoritmą. Šiuose paveiksluose išdidintos tos pačios vietos – iš rezultatų galima nustatyti, kad gauti rezultatai yra tikslūs.



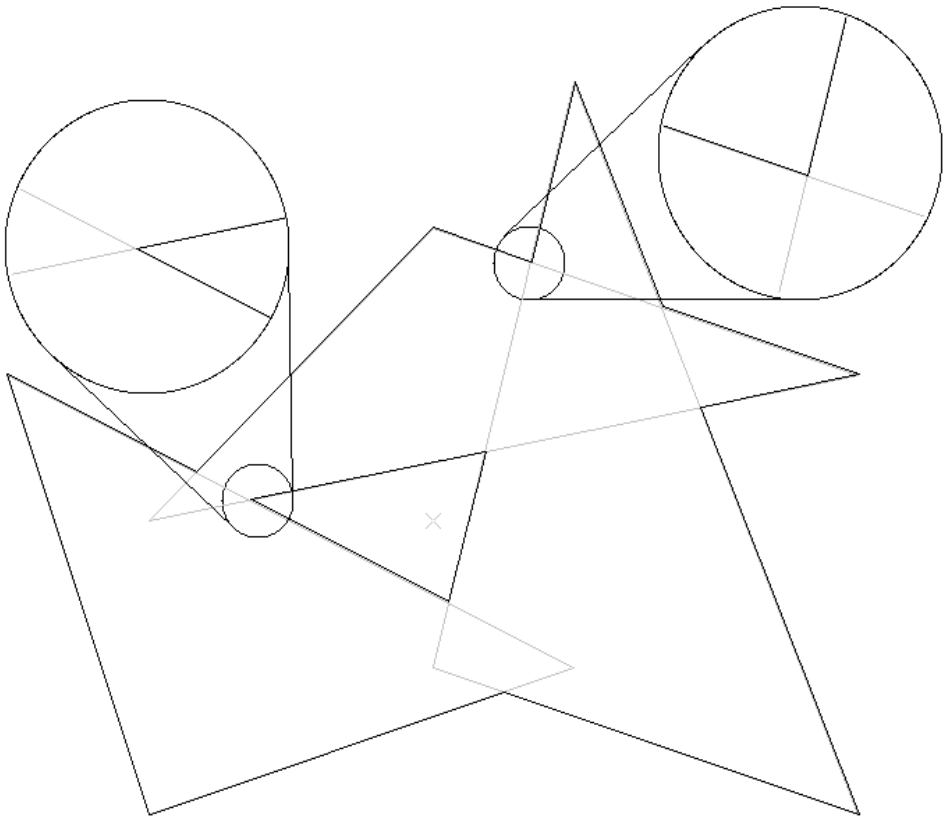
40 pav. Rezultatas sugeneruotas naudojant skenavimo algoritmą (63 360 segmentų)



41 pav. Rezultatas sugeneruotas naudojant skenavimo algoritmą(90 000 segmentų)



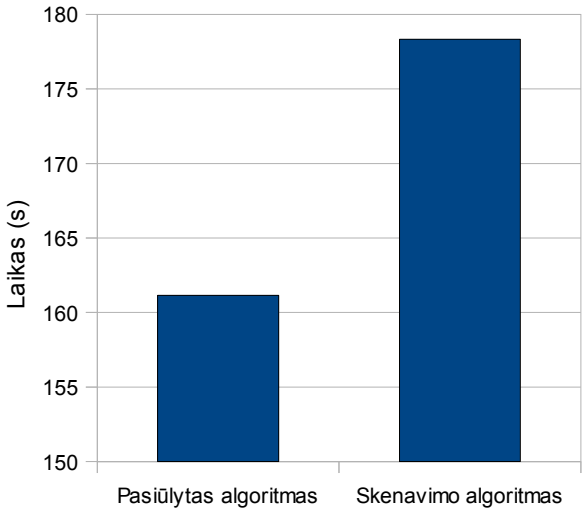
42 pav. Rezultatas sugeneruotas naudojant šluojančios tiesės algoritmą(63 360 segmentų)



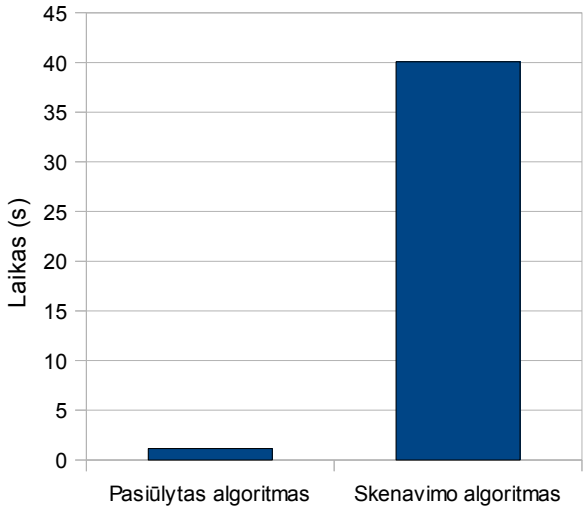
43 pav. Rezultatas sugeneruotas naudojant šluojančios tiesės algoritmą(90 000 segmentų)

Toliau šiame tyrime matuojamas skaičiavimų greitis. Diagramose 45 ir 46 galima

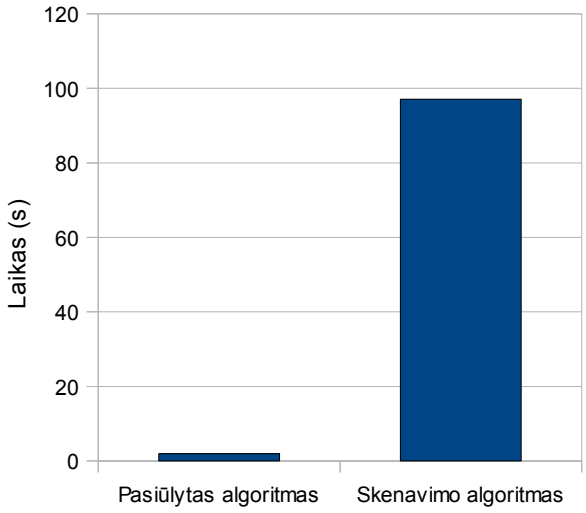
akivaizdžiai pastebėti algoritmo paremto šluojančia tiese pranašumą lyginant su skenavimo algoritmu. Diagramoje 44 jau tokio aiškaus skirtumo nėra – tai paaiškinama tuo, kad pirmam duomenų rinkinyje (63 360 segmentų) yra suskaičiuojama labai daug įvykio taškų per kuriuos turi judėti šluojanti tiesė. Ir tokiu atveju, jau nebėra tokio akivaizdaus pranašumo prieš skenavimo linijos algoritmą, kuris dirba ant suformuoto tinklelio (grid).



44 pav. Sujungimo operacijos skaičiavimų greitis tarp 63 360 segmentų



45 pav. Sujungimo operacijos skaičiavimų greitis tarp 100 000 segmentų

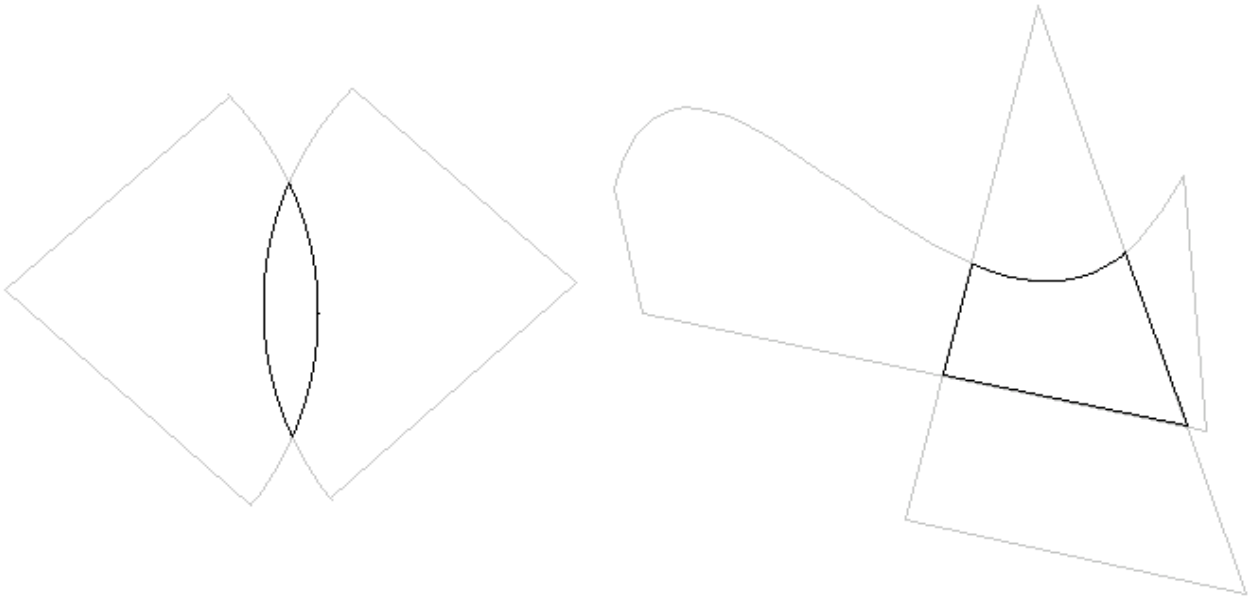


46 pav. Sujungimo operacijos skaičiavimų greitis tarp 90 000 segmentų

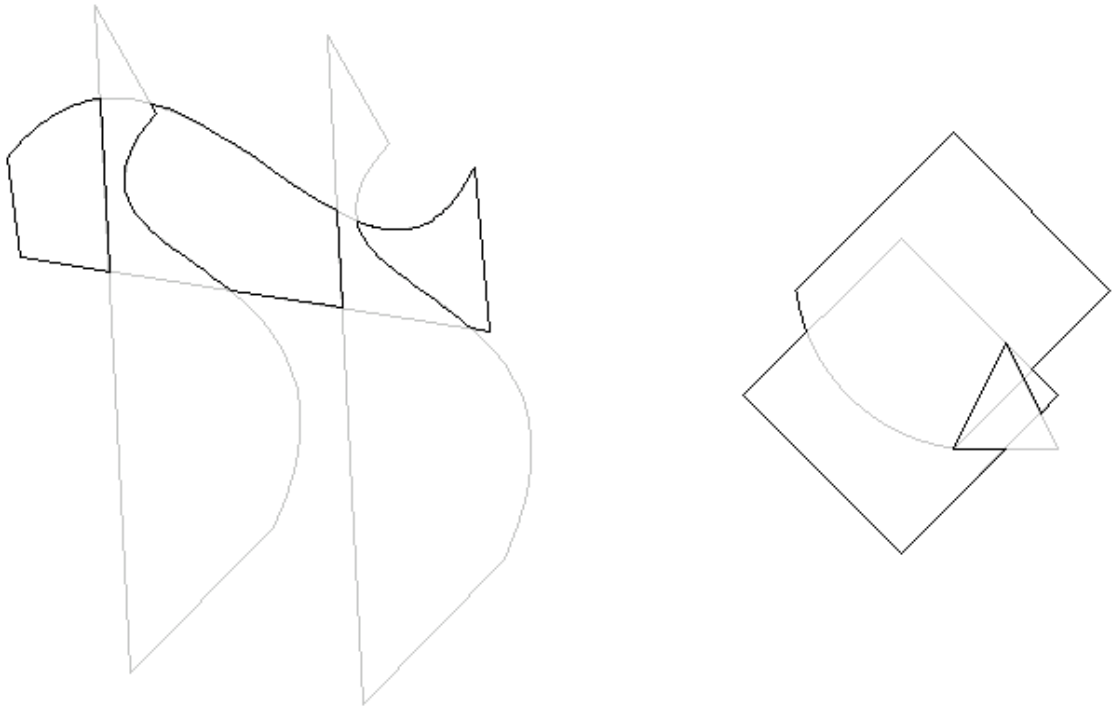
6 lentelė. Sujungimo operacijos atlikimo laikai

Duomenų segmentų kiekis	Binarinis medis, laikas(s)	Šluojanti tiesė, laikas(s)
63 360	161,15	178,34
90 000	1,96	97,12
100 000	1,14	40,06

Sekančiuose paveiksluose pateikti operacijų rezultatai naudojant šiame darbe pasiūlytą algoritmą.



47 pav. Sankirta



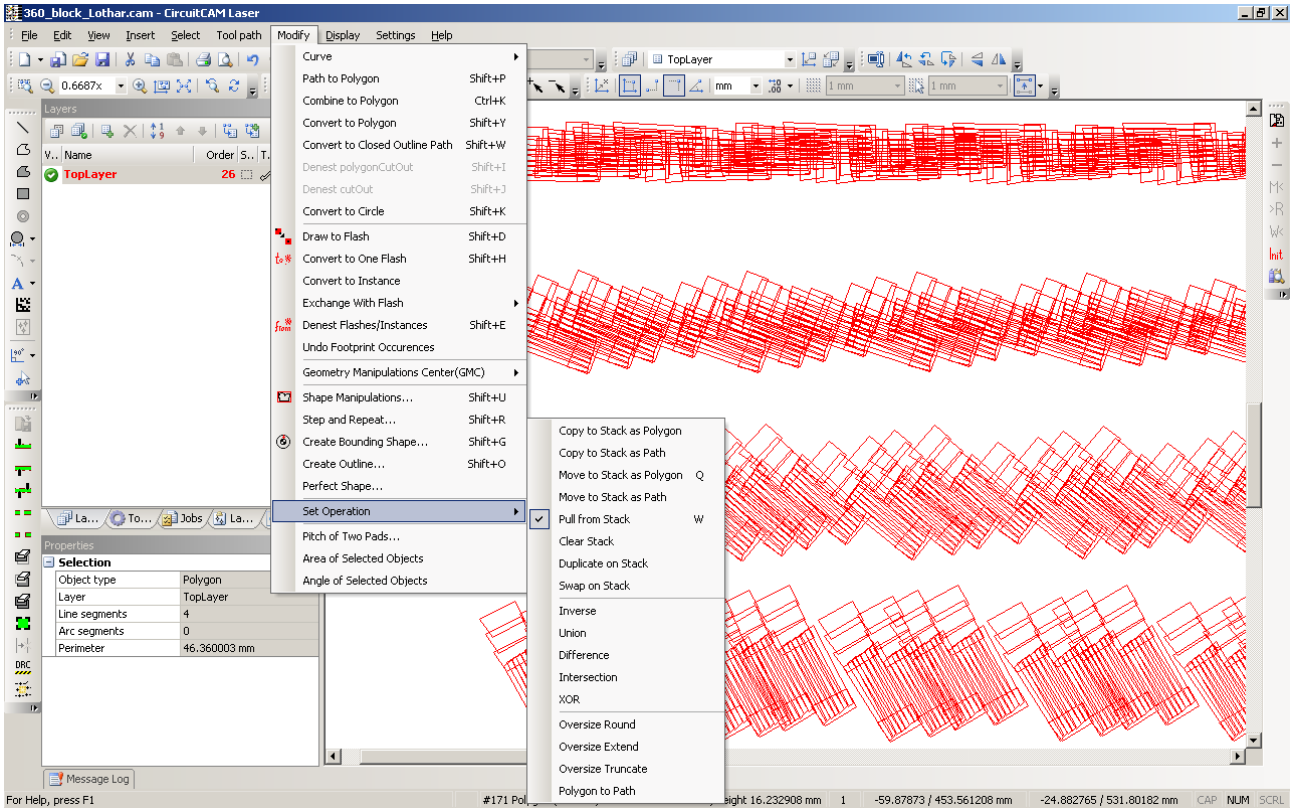
48 pav. Atėmimas

3.5 Algoritmo įdiegimas CircuitCAM programinėje įrangoje

Sudarytas algoritmas operacijoms su plokščiosiomis geometrinėmis figūromis buvo įdiegtas *CircuitCAM* programoje.

CircuitCAM – tai daugiau nei 10 metų rinkoje gyvuojanti programinė įranga (realizuota *C++* programavimo kalba), kurios pagrindinės funkcijos yra:

- nuskaityti įvairius projektavimų (*CAD/CAM)* sistemų duomenų formatus (*Gerber, GerberX, DXF, HPGL, ODB++* ir t.t.)
- redaguoti schemų topologijas
- atlikti įvairius skaičiavimus (sukurti izoliacinius takelius (*insulate),* paruošti trafaretus pjaustymui lazeriu (*stencil)* ir t.t.)
- eksportuoti duomenis įvairiais duomenų formatais.



49 pav. CircuitCAM programos bendras vaizdas

Su *CircuitCAM* programa buvo atliktas eksperimentinis sudaryto algoritmo tyrimas. Magistrinio darbo rezultatai panaudoti šiose operacijose (49 pav.):

- *Inverse* – invertuoti figūrą
- *Union* – atlikti 2 figūrų, esančių steke, sujungimą

- *Difference* – atlikti 2 figūrų, esančių steke, atėmimą
- *Intersection* – atlikti 2 figūrų, esančių steke, sankirtą

Pradiniai duomenys yra surenkami iš vidinio *CircuitCAM* duomenų formato ir tada užpildomos algoritmo duomenų struktūros, kurios saugomos steke (*stack)*. Tai galima atlikti kopijuojant duomenis į steką *(Copy to Stack as Polygon).* Po operacijų (sujungimo, sankirtos, atėmimo) gauti algoritmo rezultatai yra konvertuojami į vidinį *CircuitCAM* duomenų formatą – tai atliekama perkeliant duomenis iš steko į pagrindinį *CircuitCAM* dokumentą (*Pull from Stack)*.

Be to, visos šios operacijos naudojamos izoliacinių takelių skaičiavime, trafaretų paruošime ir eilėje kitų CircuitCAM funkcijų.

4 Išvados

1. Sudaryti ir ištirti algoritmai operacijoms su plokščiosiomis geometrinėmis figūromis – figūrų sujungimas, sankirta ir atėmimas.
2. Segmentų tarpusavio susikirtimams nustatyti buvo sudaryti ir ištirti trys algoritmai. Visi jie buvo paremti šluojančios tiesės principu, tik su skirtingu šluojančios tiesės statuso realizavimu – statusas, realizuotas binariniu medžiu, statusas, realizuotas masyvu su pilnu perrinkimu, statusas, realizuotas masyvu su optimizuota paieška. Pilnas perrinkimas yra visiškai netinkamas segmentų tarpusavio susikirtimams – didėjant duomenų apimčiai, skaičiavimų apimtys auga labai sparčiai. Visais atvejais, geriausią laiką parodė šluojančios tiesės algoritmas, kurios statusas aprašytas binariniu medžiu.
3. Visoms operacijoms yra naudojamas vienas bendras mechanizmas – skiriasi tik parametras, nusakantis intervalų, gautų sukirtus šluojančią tiesę su duomenų segmentais, skaičiavimų būdą.
4. Eksperimentais patvirtinta, kad sudarytasis algoritmas yra žymiai pranašesnis prieš skenavimo algoritmą, kai lyginami skaičiavimo atlikimo laikai. Didėjant segmentų susikirtimų skaičiui šis skaičiavimo laikų skirtumas mažėja, bet vistiek išlieka gana ženklus.
5. Nagrinėjant gautų rezultatų tikslumą buvo pastebėta, kad gautų rezultatų kokybė naudojant šiame darbe pateiktą algoritmą yra daug didesnė nei skenavimo algoritmo. Rezultatų kokybė buvo vertinama pagal tai, kokį segmentų kiekį turi gautos geometrinės figūros bei lyginant teoriškai paskaičiuotų viršūnių ir gautų figūrų viršūnių koordinates.
6. Pasiūlytas algoritmas įdiegtas ir išbandytas *CircuitCAM* pakete.
7. Pateikti ir realizuoti algoritmai, leidžiantys viršūnes jungti kvadratinėms ir kubinėmis *bezjė* kreivėmis. Ruošiamasi praplėsti *CircuitCAM* funkcionalumą įdiegiant *bezjė* kreivių palaikymą kuriant geometrines figūras ir atliekant sujungimo, sankirtos ir atėmimo operacijas.

5 Literatūra

1. Andrew, S. Glassner. *Graphics Gems*. Academic Press, 1995
2. Bartels, R. H.; Beatty, J. C.; Barsky, B. A. *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*. San Francisco, CA: Morgan Kaufmann, Ch. 10, p. 211-245, 1998.
3. Bourke, P. *Intersection of two circles,* April 1997*,* http://local.wasp.uwa.edu.au/~pbourke/geometry/2circle/. Žiūrėta 2010 03 11
4. Greiner G., Hormann K. *Efficient Clipping of Arbitrary Polygons,* Computer Graphics Group, University of Erlangen*,* *http://www.inf.usi.ch/hormann/papers/Greiner.1998.ECO.pdf*. Žiūrėta 2010 04 05
5. Liang Y., Barsky B. *An Analysis and Algorithm for Polygon Clipping* CACM, vol. 26, p. 868-877, 1983.
6. Maillot, Patrick-Gilles *A New*, *Fast Method For 2D Polygon Clipping: Analysis and Software Implementation*. *http://pmaillot.chez.com/2dpclip/2dpclip.htm*. Žiūrėta 2010 04 15
7. Plukas, K. *Skaitiniai metodai ir algoritmai*. Kaunas:Naujasis lankas, 2001.
8. Riškus, A. *Approximation of a cubic bezier curve by circular arcs and vice versa. ISSN 1392 – 124x information technology and control*, 2006, Vol.35, No.4, p. 371-377
9. Sederberg, T. W. *An Introduction to Bezier Curves*, January 6, 2003, *http://tom.cs.byu.edu/~455/bz455.pdf*. Žiūrėta 2010 05 02
10. Sutherland, I.E., Hodgman, G.W., *Reentrant Polygon Clipping*, Communications of the ACM, Vol. 17, No. 1, (Jan. 1974), pp. 32-42.
11. Weiler K., Atherton P. *Hidden Surface Removal Using Polygon Area Sorting*, Computer Graphics, vol. 11, p. 214-222, 1977.

6 Summary

An Algorithm for polygon set-operations (union, intersection, difference) was introduced. In CAD system these operations are very important and widely used. The most important aspects are speed and quality of results. Previously used and analyzed polygon operations was based on grid calculations. Thus this strategy takes big amount of time to perform operation and generates not exact results.

The major idea in presented algorithm for polygon operations was to use sweeplines through calculated event points (all vertexes and intersection points between data segments). For each sweepline the intervals information is generated (depending on amount of up and down directions from crossed segments). Afterward these intervals are connected between each other to construct the final polygon. As additional feature of this algorithm is support of arcs, quadratic and cubic bezier curves, which connects polygon vertexes.

The results of this algorithm were compared with results of scanline (grid) based algorithm and Java2D API (this API is not able to process big amount of data, thus this is used only to check the correctness of result for polygon operations).

After implementation and experiments it is obvious that this algorithm works and provides better results (speed and quality of resulting polygons). The detailed comparison and analysis of polygons set-operations is presented in experimental part of this paper.

7 Priedai

Įdiegimo aktas.

LKSoft Baltic

UŽDAROJI AKCINĖ BENDROVĖ „LKSOFT BALTIC"

Magistranto **Andriaus Riepšo** sukurto programinio paketo
loginėms operacijoms su geometrinėmis sritimis atlikti

ĮDIEGIMO AKTAS

2010-05-25
Kaunas

Jau daugelį metų UAB LKSoft Baltic plėtoja ir tobulina rinkoje paplitusią programų sistemą CircuitCAM. Tai programų sistema spausdintų plokščių topologijų projektavimui, technologiniam paruošimui ir spausdinimui. Laikui bėgant, sistemos funkcionalumas buvo išplėtotas, suteikiant įvairias schemų topologijų redagavimo galimybes. Vykdant šiuos darbus, jau penkis metus prie jų dalyvauja magistrantas Andrius Riepšas, kurio pasirinkta magistrinio darbo tema „Algoritmų operacijoms su plokščiosiomis geometrinėmis figūromis sudarymas ir tyrimas".

Andrius Riepšas į CircuitCAM programų sistemą įdiegė jo sukurtą programinį paketą, kuris atlieka logines operacijas su geometrinėmis sritimis – sujungimą, sankirtą ir atėmimą. Šios operacijos naudojamos izoliacinių takelių, trafaretų kūrime ir taip gaunami tikslesni takeliai bei trafaretai. Darbas atliktas naudojant C++ ir Microsoft Visual Studio 2005. Paketo funkcijos buvo ištestuotos CircuitCAM aplinkoje, kai pradiniai duomenys buvo sukurti CircuitCAM redagavimo priemonėmis arba importuoti iš įvairių duomenų formatų (*GerberX*, *DXF* ir kt.).

Dėl uždavinio specifika grįsto geometrinės sritis ribojančių analizuojamų taškų skaičiaus sumažėjimo paketu gaunamas skaičiavimo laiko sutrumpėjimas, lyginant su tipinėmis šių operacijų realizacijomis kitose sistemose. Be to, šis paketas yra paruoštas dirbti su Bezjė kreivėmis ir tai leidžia praplėsti CircuitCAM funkcionalumą (dabar CircuitCAM geometrinių figūrų viršūnės galima jungti tik tiesėmis ir lankais).

Paketas platinamas su programų sistema CircuitCAM kaip jos sudėtinė dalis.

Direktorius Vaidas Nargėlas

Įmonės kodas 1108 33187
PVM mokėtojo kodas LT108331811
Registro tvarkytojas VĮ Registrų centras

Vičiūnų g. 13
LT-45325 Kaunas
Lietuva

Tel. +370 37 407063
Faks. +370 407063
www.lksoft.lt

AB bankas Hansabankas
Atsiskaitomoji sąskaita
LT367300010002508701