Inkrementalni 3D konveksni omotač

Seminarski rad

Konstrukcija i analiza algoritama 2

Miroslav Mišljenović 1110/2018

Studijski program Informatika

Nastavnik: dr Vesna Marinković

Asistent: Mirko Spasić

Matematički fakultet

Univerziteta u Beogradu

Februar 2019., Beograd

1. Osnovne karakteristike softverskih rešenja

Osnovni rezultati su dobijeni krajem 20. veka.

Većina softvera u javnom vlasništvu je pisana u starijim verzijama Jave i C++ - a.
Isti softver služi i za druge svrhe (dijagrami Voronjeva i Delanijeva trijangulacija) pa ima znatnog viška koda.

Posebno treba obratiti pažnju na robusnost softvera, jer se tačke uglavnom zadaju u pokretnom zarezu. Tada se dobijaju strane koje se skoro ne razlikuju što izaziva razne probleme (kolinearnost i komplanarnost), ali to neće biti razmatrano.

2. Algoritmi za određivanje 3D konveksnog omotača

Za skup od N tačaka u trodimenzionalnom prostoru, potrebno je pronaći neku njihovu funkcionalnu prezentaciju sa manjim brojem tačaka, recimo u problemima utvrđivanja kolizije nekih objekata, što se može primeniti u raznim video igrama. Jedna takva prezentacija je konveksni omotač i za njegovo određivanje postoji nekoliko vrsta algoritama.
Neki od njih su:

- Brzi algoritam za dobijanje 3D konveksnog omotača (eng. Quick 3D hull algorithm) – pohlepni algoritam, postiže odlične performanse kada postoji mnogo unutrašnjih tačaka.
- Čanov (eng. Chan) algoritam – kombinacija nekog algoritma za određivanje 3D konveksnog omotača i „uvijanja poklona“. Prvi algoritam koji ima zagarantovano vreme izvršavanja O(n log n).

- Merge hull algoritam – za dva skupa sa po N/2 tačaka se odrede omotači i spoje u finalni konveksni omotač postupkom „uvijanja poklona“.
- Inkrementalni kojim se dodaje jedna po jedna tačka u tekući konveksni omotač.

3. Inkrementalni algoritam

Prvi korak je da se odrede četiri nekoplanarne tačke, koje čine tetraedar – najmanji konveksni skup u trodimenzionom prostoru. Zatim se dodaje jedna po jedna od preostalih N-4 tačaka u tekući konveksni omotač.

Da bi se obezbedio slučajni položaj tačaka koje se dodaju u omotač, preporučuje se da se tačke izmešaju (eng. shuffle). Time se značajno smanjuje verovatnoća da su polazne tačke u takvom rasporedu da obrazuju konveksni omotač, čime bi se dobile veoma loše performanse.[1]

Za svaku dolazeću tačku se određuje da li je u unutrašnjosti ili na granici tekućeg konveksnog omotača (i u tom slučaju se ona odbacuje) ili se tačka nalazi van omotača. U drugom slučaju se uklanjaju vidljive strane omotača iz te tačke i kreiraju nove strane od horizonta do te tačke.

[1] Neka N/2 tačaka čini pravilan mnogougao u ravni, sa centrom u koordinatnom početku i neka je N/2 tačaka raspoređeno u tačkama na pozitivnom delu z-ose. Najpovoljniji raspored dodavanja tačaka na pravilni poligon je da se doda tačka sa najvećom z-koordinatom. U tom slučaju je potrebno formirati N/2 ivičnih strana. Sve preostale tačke na z-osi su u unutrašnjosti dobijenog konveksnog skupa, pa je njihovo procesiranje jednostavno i brzo. U najnepovoljnijem slučaju, tačke se uključuju od one sa najmanjom z-koordinatom, pa naviše. Za svaku od njih se briše N/2 prethodno dobijenih bočne strana i dodaje N/2 bočnih strana, pa se složenost $O(n^2)$.

Takođe, prilikom izvršavanja algoritma, održava se konfliktni graf. To je bipartitni graf u kome jedna particija sadrži tačke, a druga strane. Grane grafa određuju koje strane se vide iz kojih tačaka. U početku se graf sastoji od svih tačaka osim onih koje čine početni tetraedar, kao i strana početnog tetraedra. Inicijalno konfliktni graf nema grane. One se inicijalizuju sa svim vidljivim parovima (p_t, f), gde je t > 4, a f je strana konveksnog omotača.

Tokom rada, održavaju se dva skupa:

- $P_{conflict}(f)$ – skup tačaka koje vide stranu f
- $F_{conflict}(p_R)$ – skup strana vidljivih iz p_R

Na kraju algoritma, konfliktni graf sadrži samo strane koje obrazuju konveksni omotač, nema čvorova grafa koji odgovaraju tačkama, niti grana grafa.

Postupci dodavanja nove tačke, određivanja horizonta, kao i koncept vidljivih strana iz neke tačke, su prikazani u pratećoj .ppt prezentaciji.

4. Složenost algoritma

Poznato je da Ojlerova formula za povezani planarni graf sa n čvorova, n_e grana i n_f strana zadovoljava relaciju:

$n - n_e - n_f = 2$

Ako su strane trouglovi, onda je:

$2 * n_e = 3 * n_f$

Odatle dobijamo da je:

$n_f = 2 * n - 4$

$n_e = 3 * n – 6$

Znači, za n čvorova imamo O(n) grana i O(n) strana.

Prva faza inicijalizacije zahteva O(n) operacija, jer imamo 4 strane i n-5 tačaka.
Kreiranje i brisanje strana: svaka strana se bar jednom pravi/briše, pa je složenost ovog koraka O(n).

Ažuriranje konfliktnog grafa: najveći deo vremena se provede u procesiranju ivica horizonta.

Lema 11.6 je dokazana u referenci 1 i glasi:
Očekivana vrednost za $\sum_e$ card(P(e)),
gde je sabiranje po ivicama horizonta
koje se javljaju u nekom koraku algoritma,
iznosi O(n log n).

P(e) je unija tačaka koje vide strane f_1 i f_2 koje su incidentne sa ivicom horizonta e, samim tim vide i e.

Ovaj poslednji korak ima najveću složenost,
pa je ukupna složenost algoritma O(n log n).

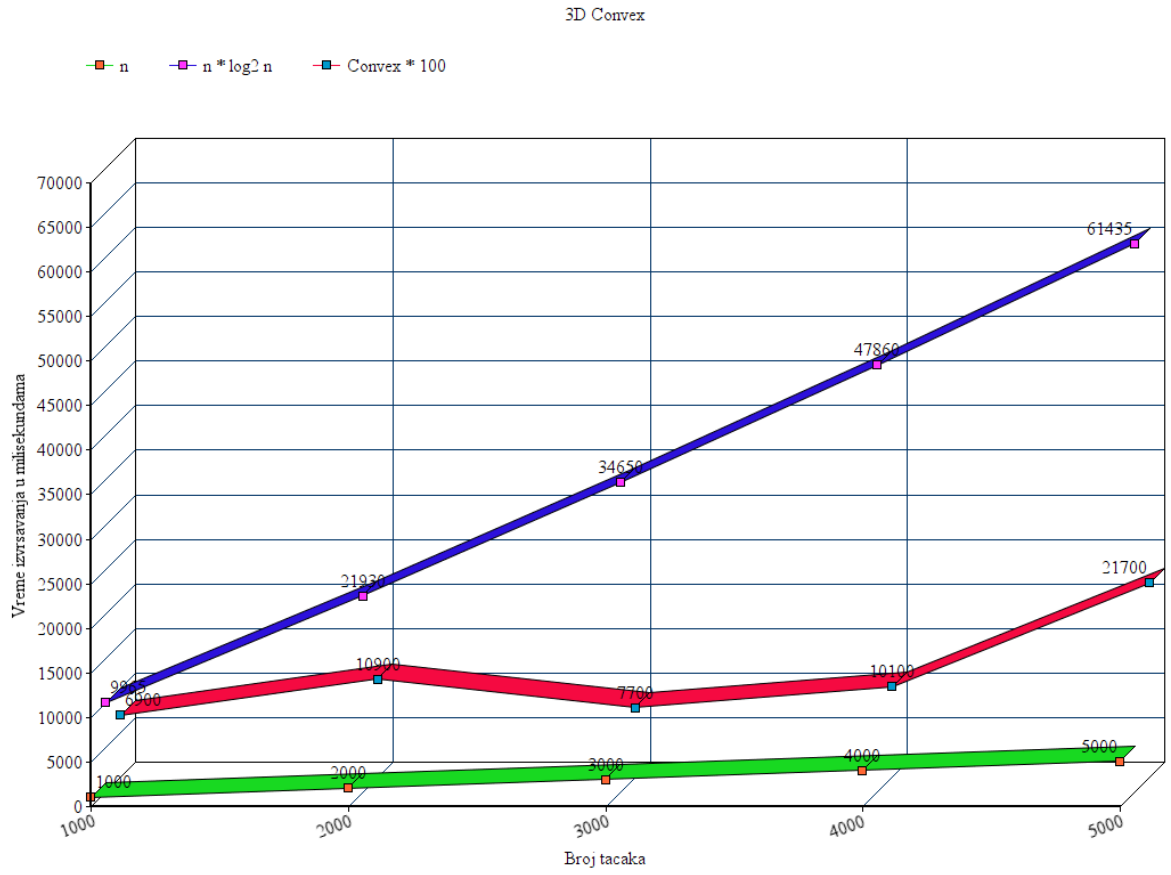
5. Implementacija i test primeri

Implementacija algoritma je obrađena u Javi. Kao osnova je služio otvoreni kod Krisa Pudnija (ref. 1). U njemu je bilo znatnog viška koda, koji je trebalo prilagoditi potrebama ovog algoritma.

Prirodno je da se rezultati mogu videti u nekom grafičkom okruženju, pa je izabran grafički softver Geomview (ref. 5).

Za generisanje slučajnih tačaka napisan je poseban program.

Program je formirao izlazne skupove od 1000, 2000, 3000, 4000 i 5000 slučajnih tačaka, sa uniformnom raspodelom
na intervalu [0-10000]. Ti skupovi koristili su se kao ulaz u glavni program. Rezultati izvršavanja su prikazani na sledećem grafikonu.



Zelenom bojom označena je linearna zavisnost.
Plavom bojom označena je n log n zavisnost.
Crvenom bojom su prikazani rezultati izvršavanja, koji su zbog potreba upoređivanja i tumačenja pomnoženi konstantom 100.
Sama vremena izvršavanja algoritma su bila u milisekundama.

Dobijene vrednosti između plave i zelene linije su u saglasnosti sa ulaznim podacima, jer su dobijeni skupovi podataka bili sa velikim brojem unutrašnjih tačaka, pa je dobijen mali broj strana

konveksnog skupa. Za 1000, 2000, 3000, 4000, 5000 tačaka je dobijeno redom 81, 93, 96, 105 i 108 tačaka
odnosno 158, 182, 188, 206 i 212 strana konveksnog omotača.

Ako bi tačke bile birane na neki drugi slučajan način (npr. normalnom raspodelom), očekujemo da bi dobili znatno različit rezultat od prethodnog.
Da bismo dobili rezultat saglasan sa složenošću O(n log n), jedan pouzdan pristup bi bio da uzmemo skup tačaka koje čine konveksni omotač kao ulazni skup. U dokazu leme 11.6 je dobijena ocena složenosti od 96 * n * log n, što jeste O(n log n), ali za ekstremno veliki broj tačaka. Stoga bi očekivali da za naše skupove podataka i u ovom slučaju ne bismo mogli dokazati asimptotsku ocenu.

6. Literatura

1. Chris Pudney, The University of Western Australia, 1998. – open source Java code
2. Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars,
   Computational Geometry – Algorithms and Applications, Third edition, Springer, 2008.
3. David M. Mount, Department of Computer Science, University of Maryland, Fall 2002.
   Computational Geometry
4. Dirk Gregorius, The 3D Quick Hull Algorithm, Valve Software
5. Geomview 1.9.5, Mart 2014. – http://www.geomview.org softver za vizualizaciju