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 (kolinearnost i komplanarnost) što izaziva razne probleme, ali to ovde 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 video igrama). Jedna takva prezentacija je konveksni omotač i za njegovo određivanje postoji nekoliko 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

Algorithm CONVEXHULL(P)
Input. A set P of n points in three-space.
Output. The convex hull $\mathcal{CH}(P)$ of P.

1. Find four points p_1, p_2, p_3, p_4 in P that form a tetrahedron.
2. $\mathcal{C} \leftarrow \mathcal{CH}(\{p_1, p_2, p_3, p_4\})$
3. Compute a random permutation $p_5, p_6, \ldots, p_n$ of the remaining points.
4. Initialize the conflict graph $\mathcal{G}$ with all visible pairs (p_t, f), where f is a facet of $\mathcal{C}$ and $t > 4$.
5. **for** $r \leftarrow 5$ **to** n
6. **do** ($*$ Insert p_r into $\mathcal{C}$: $*$)
7. **if** $F_{\text{conflict}}(p_r)$ is not empty ($*$ that is, p_r lies outside $\mathcal{C}$ $*$)
8. **then** Delete all facets in $F_{\text{conflict}}(p_r)$ from $\mathcal{C}$.
9. Walk along the boundary of the visible region of p_r (which consists exactly of the facets in $F_{\text{conflict}}(p_r)$) and create a list $\mathcal{L}$ of horizon edges in order.
10. **for** all $e \in \mathcal{L}$
11. **do** Connect e to p_r by creating a triangular facet f.
12. **if** f is coplanar with its neighbor facet f' along e
13. **then** Merge f and f' into one facet, whose conflict list is the same as that of f'.
14. **else** ($*$ Determine conflicts for f: $*$)
15. Create a node for f in $\mathcal{G}$.
16. Let f_1 and f_2 be the facets incident to e in the old convex hull.
17. $P(e) \leftarrow P_{\text{conflict}}(f_1) \cup P_{\text{conflict}}(f_2)$
18. **for** all points $p \in P(e)$
19. **do** If f is visible from p, add (p, f) to $\mathcal{G}$.
20. Delete the node corresponding to p_r and the nodes corresponding to the facets in $F_{\text{conflict}}(p_r)$ from $\mathcal{G}$, together with their incident arcs.
21. **return** $\mathcal{C}$

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) kako ne bismo dobili veoma loše performanse.

Primer:
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 bočnih strana i dodaje N/2 novih bočnih strana, pa je složenost $O(n^2)$.

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 ivica horizonta do te tačke.

Horizont čini izlomljena linija (u prostoru) koja se dobija od ivica čija je jedna incidentna strana vidljiva a druga nevidljiva.

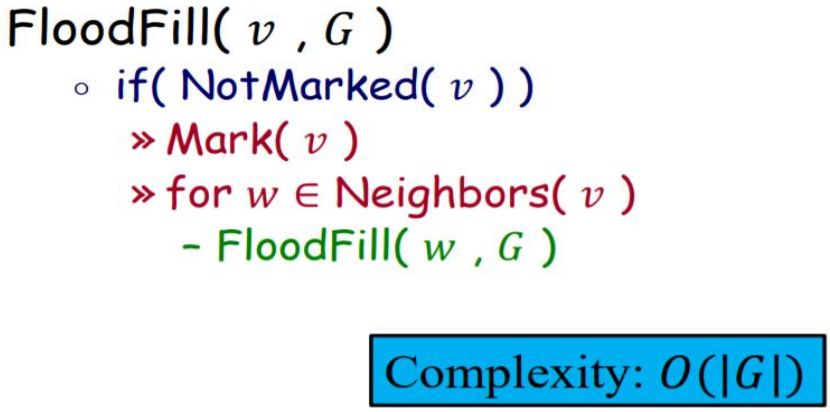
Algoritamski, dobijanje horizonta se može uraditi na sledeći način: Za svaku vidljivu stranu tekućeg konveksnog omotača, ivice te strane treba ažurirati u listi ivica horizonta (inicijalno prazna) na sledeći način:

ako ivica postoji u listi, obrisati je;

ako ivica ne postoji u listi, dodati je u listu.

Kada se iscrpe sve vidljive strane, lista će sadržati sve ivice horizonta.

Ovaj algoritam je poznat kao Algoritam plavljenja. Primer plavljenja je prikazan u pratećem dokumentu (FindingTheHorizon, ref. 3) počev od strane 36 i biće prikazan na odbrani ovog seminarskog rada.



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. Grane konfliktnog grafa povezuju vidljive parove tačaka i strana (p_t, f), gde je t > 4, a f je strana konveksnog omotača.

Tokom rada, održavaju se dva skupa:

- $P_{conflict}(f)$ – skup tačaka iz kojih je vidljiva strana f
- $F_{conflict}(p_t)$ – skup strana vidljivih iz p_t

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

Jednostavan način za ažuriranje skupa od N tačaka i skupa od O(N) strana zahteva $O(N^2)$ operacija. Da bi se održala složenost algoritma O(n log n), formira se struktura podataka koja to omogućuje. Ova struktura - konfliktni graf - se ažurira tokom rada algoritma, a u ažuriranju učestvuje mali broj tačaka i strana. Pokazuje se da se to stvarno može uraditi u O(n log n) operacija.

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. Cilj je da se pokaže da je složenost ovog dela algoritma O(n log n).
P(e) je unija tačaka iz kojih se vide strane f1 i f2 koje su incidentne sa ivicom horizonta e, samim tim vide ivicu e.
card(P(e)) je kardinalnost (broj elemenata) prethodno definisane unije tačaka;
očekivana vrednost za Σ_e card(P(e)), gde je sabiranje po ivicama horizonta koje se javljaju u nekom koraku algoritma,
iznosi O(n log n). (to je pokazano u referenci 2, lema 11.6)

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

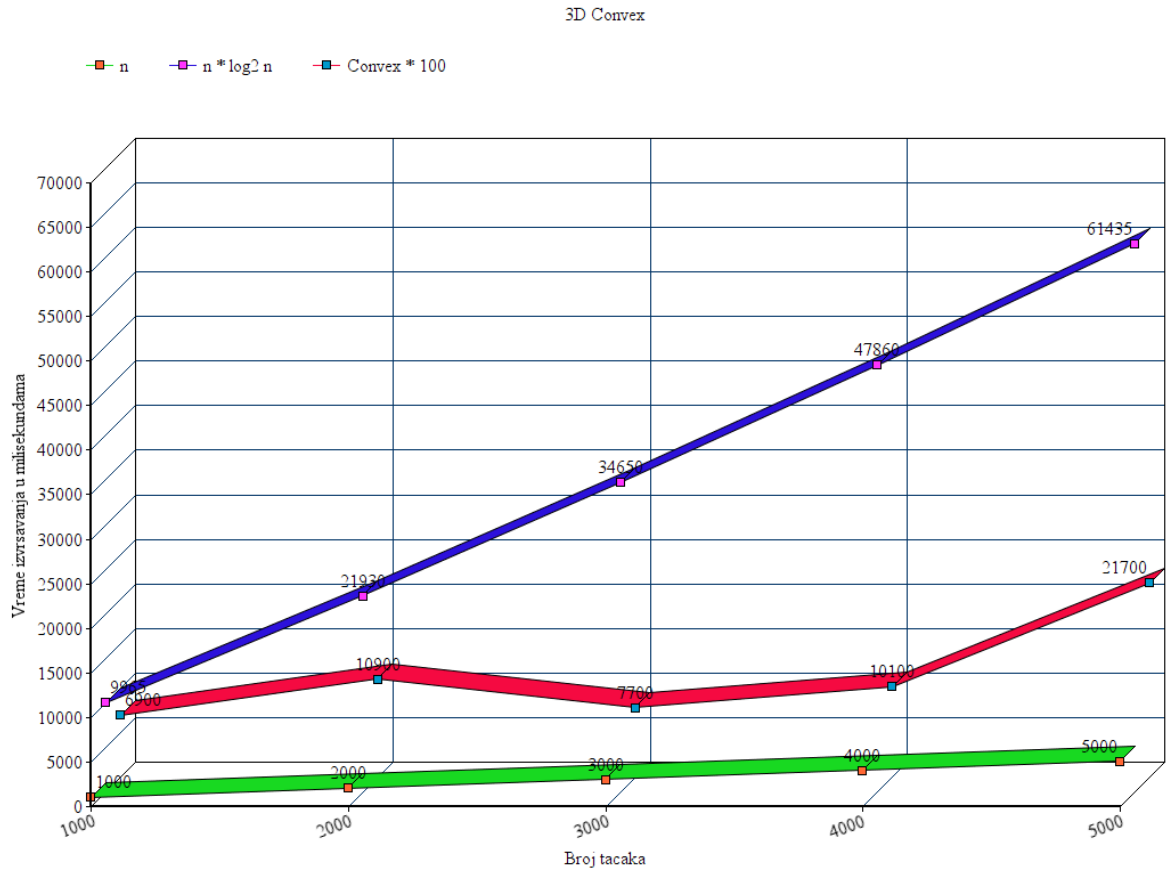
5. Implementacija i test primeri

Implementacija algoritma je realizovana 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. 4).

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 pristup bi bio da radimo sa najnepovoljnijim skupom tačaka, a to su one koje čine konveksni omotač, kao ulazni skup. U dokazu leme 11.6 iz reference 2, dobijena je 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 male skupove podataka i sa ovim pristupom ne bismo mogli dokazati/potvrditi 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. Dirk Gregorius, The 3D Quick Hull Algorithm, Valve Software
4. Geomview 1.9.5, Mart 2014. – http://www.geomview.org softver za vizualizaciju