Klasifikacija ploskev

Rok Koleša, Domen Kren, Darko Janković

27. april 2015

Mentor: as. dr. Gregor Jerše

Kazalo

1 Cilj

Cilj projekta je bil ustvariti program za klasifikacijo ploskev z robovi.

1.1 Definicije

Ploskev Ploskev v matematiki pomeni dvorazsežno tvorbo v večrazsežnem prostoru. Tvorba mora biti kompaktna in povezana.

Robna komponenta Komponenta, ki predstavlja luknjo v triangulaciji. Sestavljajo jo robovi, ki so rob natanko enega trikotnika.

2 Vhodni podatki

Abstraktni simplicialni kompleksi, podani kot seznami trojic števil. Vsak element je celo število, ki predstavlja indeks oglišča, celotna trojica pa predstavlja trikotnik v kompleksu.

Primer vhoda za disk:

```
1 2 3
2 3 4
2 4 5
2 5 6
```

3 Rešitev

Program za klasifikacijo ploskev je sestavljen iz treh delov: prvi preveri, če vhodna triangulacija predstavlja ploskev, drugi del prešteje robne komponente, tretji pa jo klasificira.

3.1 Ali je triangulacija ploskev

Najprej bomo preverili ali podana triangulacija predstavlja ploskev. V našem primeru se to prevede na preverjanje ali triangulacija predstavlja več komponent in koliko sosedov ima vsak rob. Oba algoritma sta precej preprosta.
Pri prvem vzamemo nek trikotnik in mu dodamo sosede, nato dodamo njihove sosede, itd. Na koncu le pogledamo ali so v množici podanih trikotnikov ostali kakšni, ki jih s tem pregledovanjem nismo dosegli. Če obstajajo, imamo več komponent, česar ne moremo klasificirati.
Pri drugem pa se sprehodimo po vseh robovih trikotnikov in pogledamo ali imajo za soseda natanko enega (je del robne komponente), ali dva (rob je v ploskvi) trikotnika. Če najdemo več sosedov, triangulacija ne predstavlja ploskve.

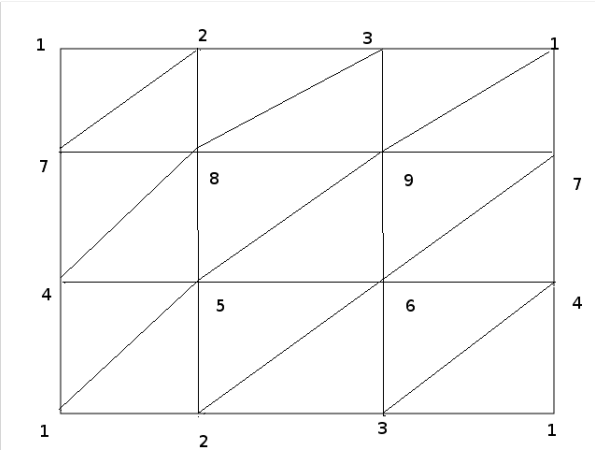
3.2 Število robnih komponent

Število robnih komponent oz. število lukenj smo poiskali s preprostim algoritmom, ki je v grobem sestavljen iz naslednjih korakov:

a. poišči robove, ki se v triangulaciji pojavijo le enkrat

b. preštej cikle, ki jih sestavljajo dobljeni robovi

Opisan algoritem bi na triangulaciji, prikazani na sliki 1, v primeru odstranjenih trikotnikov *(2, 6, 5)* in *(6, 7, 9)* našel naslednje robove, ki se pojavijo samo enkrat: *2 - 5*, *2 - 6* in *5 - 6*, ter *6 - 7*, *6 - 9* in *6 - 9*, iz česar bi potem zaznal dva cikla, ki sta identična odstranjenima trikotnikoma.



Slika 1: Primer triangulacije

3.3 Klasifikacija

Na koncu je prišla na vrsto še dejanska klasifikacija ploskve. Na vajah smo že pokazali kako se klasificira ploskve brez roba, in sicer lahko to naredimo na podlagi Eulerjeve karakteristike $\chi(T)$, kjer je T triangulacija, in orientabilnosti triangulacije. Videli smo že, da se vse ploskve razdelijo na 3 različne razrede:

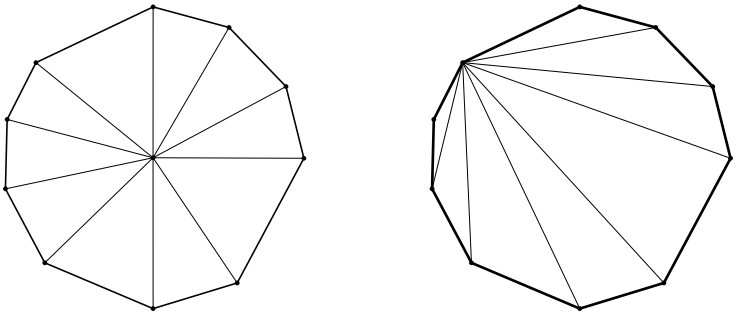
a. sfera - je orientabilna, $\chi(T) = 2$

b. n torusov - je orientabilna, $\chi(T) = 2 - 2n$

c. n projektivnih ravnin - ni orientabilna, $\chi(T) = 2 - n$

Tu pa imamo opravka s ploskvami, ki mogoče vsebujejo kakšno robno komponento. Ideja za klasifikacijo takšnih ploskev pa je, da zapolnimo vse robne komponente in dobimo novo ploskev T', ki pa ne vsebuje robov. Če vzamemo

še orientacijo originalne triangulacije T in Eulerjevo karakteristiko nove, $\chi(T')$, lahko torej dokončno klasificiramo ploskev, ki jo določa T.
Ostane nam torej samo še premislek za koliko se spremeni Eulerjeva karakteristika, ko tako polnimo luknje v triangulaciji. Luknjo lahko zapolnimo na 2 preprosta načina. Prvi način je ta, da na sredini neke luknje naredimo novo oglišče in iz te točke napeljemo rob na vsako oglišče na robu luknje. Drug način pa je, da iz poljubne točke napeljemo rob do vsake še ne sosednje točke. Oba načina sta pokazana na sliki 2 na primeru, ko ima rob 10 oglišč.



Slika 2: Dva načina polnjena luknje

Poglejmo sedaj koliko smo "pokvarili"Eulerjevo karakteristiko. Recimo, da ima robna komponenta m oglišč. Pri prvem načinu smo dodali 1 oglišče in za vsako oglišče robne komponente en rob, torej m robov. S tem smo za vsak rob robne komponente (ki jih je enako kot oglišč, torej m) tudi dodali en trikotnik. Obstoječih simpleksov nismo kvarili in zato se Eulerjeva karakteristika glasi:

$$\chi(T') = \chi(T) + 1 - m + m = \chi(T) + 1$$

Pri drugem načinu pa dodamo vse diagonale iz neke točke kot robove. Teh je tako $m - 3$, saj ne moremo napeljati diagonale z direktnimi sosedi (ta rob v T namreč že obstaja) in tudi ne sam s sabo. Ker ima robna komponenta tudi m robov, bomo dobili $m - 2$ novih trikotnikov, saj samo z dvema robovoma (sosednima) ne moremo dobiti trikotnika. Novh točk nismo dodali in zato je Eulerjeva karakteristika enaka

$$\chi(T') = \chi(T) + 0 - (m - 3) + m - 2 = \chi(T) + 3 - 2 = \chi(T) + 1.$$

V obeh primerih smo pričakovano dobili enak rezultat, torej se $\chi(T)$ pri polnjenju ene luknje spremeni za 1.
Zaključimo torej, da je dovolj če izračunamo $\chi(T)$ in tej številki dodamo število robnih komponent, da dobimo $\chi(T')$. Sedaj lahko popolnoma klasificiramo vse ploskve z robom in brez.

4 Rezultati

Algoritem smo pognali na podanih testnih primerih. Rezultati testov so naslednji:

a. Klasifikacija triangulacije iz datoteke SyrfaceK.txt
Podana ploskev je 2 projektivnih ravnin s/z 0 luknjami

b. Klasifikacija triangulacije iz datoteke space_stationSurface.txt
Podana sta dva identična trikotnika: Trikotnik: (6366, 6367, 6368) in Trikotnik: (6368, 6367, 6366). Podan vhod ni ploskev.

c. Klasifikacija triangulacije iz datoteke space_stationSurface_no_duplicates.txt
Triangulacija ima več komponent!
Podana triangulacija ne predstavlja ploskve

d. Klasifikacija triangulacije iz datoteke disc.txt
Podana ploskev je sfera s/z 1 luknjami

e. Klasifikacija triangulacije iz datoteke SurfaceT.txt
Podana ploskev je 1 torusov s/z 0 luknjami

f. Klasifikacija triangulacije iz datoteke SurfaceGJ1.txt
Podana ploskev je 1 torusov s/z 0 luknjami

g. Klasifikacija triangulacije iz datoteke SurfaceGJ2.txt
Podana ploskev je 2 projektivnih ravnin s/z 0 luknjami

5 Zaključek

S projektnim delom smo dosegli klasifikacijo ploskve z morebitnim robom. Seveda mora podan abstraktni simplicialni kompleks sploh predstavljati ploskev. Nato je treba poiskati le orientacijo ploskve in robne komponente. Za oboje se lahko uporabi preprosta algoritem (za orientacijo smo že spoznali na vajah), ostalo pa tudi ni težko implementirati. Luknje se da zapolniti tudi na drugačne načine, opisana pa sta dva najbolj intuitivna.
Implementacija tega projekta je potekala v Javi verzije 8. Vsa koda vključno s testnimi primeri pa so dostopni tudi na GitHub-u:
`https://github.com/dokren/SurfaceClassification/`