Algoritmul de PIVOTARE a BILEI pentru reconstruirea SUPRAFEȚELOR

Mihai Scorțaru

Algoritmul de pivotare a bilei (Ball-Pivoting Algorithm - BPA) determină o plasă de triunghiuri interpolând un nor de pucte. De obicei, aceste puncte reprezintă eșantioane de puncte de pe suprafața unui obiect, obținute prin mai multe scanări ale acestuia.

Principiul care stă la baza *BPA* este relativ simplu: trei puncte formează un triunghi dacă o bilă având o rază specificată le atinge fără să mai conțină nici un alt punct.

Pornind de la un triunghi, bila pivotează în jurul unei laturi până în momentul în care atinge un alt punct, formând un alt triunghi.

Procesul continuă până în momentul în care sunt încercate toate laturile accesibile și apoi reîncepe pornind de la un alt triunghi.

Preliminarii

Dezvoltarea diferitelor tehnologii *hardware* pentru achiziționarea datelor tridimensionale au dus la răspândirea utilizării scanărilor pentru a obține descrierea geometriei unor obiecte fizice în scopul arhivării lor.

Procesul de achiziție a datelor tridimensionale constă, de obicei, în următoarele faze:

- ***scanarea:*** achiziționarea eșantioanelor pentru suprafețe folosind diferite dispozitive cum ar fi un *scanner* cu laser;
- ***înregistrarea datelor:*** alinierea mai multor scanări într-un sistem de coordonate unic;
- ***integrarea datelor:*** interpolarea eșantioanelor și a punctelor derivate din aceste eșantioane pentru a obține o reprezentare a suprafeței cum ar fi o plasă de triunghiuri;
- ***convertirea modelului:*** optimizarea plasei, compatibilitatea cu alte reprezentări etc.

În cadrul acestui articol ne vom ocupa de cea de-a treia fază: integrarea datelor. Vom prezenta un algoritm care poate fi utilizat pentru a determina o plasă de triunghiuri pentru o mulțime de puncte.

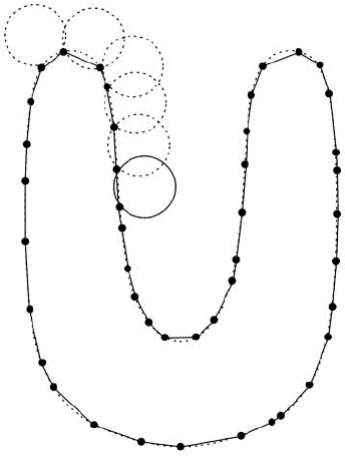

Figura 1: Bila nu poate trece prin suprafață (caz bidimensional)

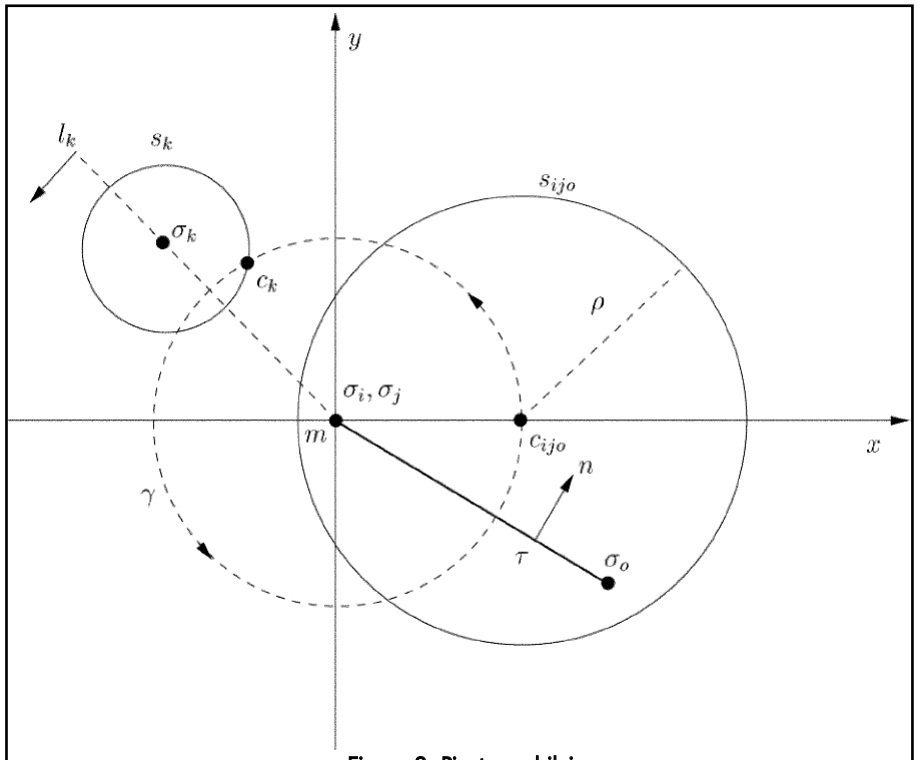

Figura 2: Pivotarea bilei

Bila care pivotează este în contact cu trei puncte ρ_i, ρ_j și ρ_0 care determină un triunghi a cărui normală este n. Latura $e_{(i,j)}$ care rămâne fixă în timpul pivotării se află pe aza Oz, iar mijlocul ei reprezintă orginea sistemului tridimensional de coordonate. Cercul s_{ij0} reprezintă intersecția dintre bilă și planul pentru care avem z = 0. Sistemul de coordonate este astfel ales încât centrul c_{ij0} al acestui cerc să se afle pe semiaxa pozitivă Ox. În timpul pivotării bila rămâne în contact cu punctele i și j, iar centrul bilei descrie o traiectorie circulară γ al cărei centru este m și a cărei rază este dată de distanța dintre c_{ij0} și m. În timpul mișcării de pivotare bila atinge un nou punct ρ_k. Vom nota prin s_k intersecția dintre o sferă de rază ρ al cărui centru este ρ_k și planul pentru care avem z = 0. Centrul c_k al bilei care pivotează este dat, în momentul în care bila atinge punctul ρ_k, de intersecția dintre γ și s_k. Acest punct se află în semiplanul stâng determinat de dreapta orientată l_k.

Reconstituire și pivotare

Conceptul care stă la baza algoritmului de pivotare a bilei este foarte simplu. Notăm prin *M* suprafața obiectului tridimensional și prin *S* o mulțime de puncte obținute prin eșantionarea suprafeței *M*.

Presupunem că *S* este suficient de densă pentru ca o bilă de raza ρ să nu poată trece prin suprafața *M* fără a atinge puncte din *S*. Un exemplu bidimensional este prezentat în figura 1.

Începem prin a plasa bila astfel încât să atingă trei puncte din *S*. Păstrând contactul cu două dintre aceste puncte efectuăm o pivotare a bilei până în momentul în care aceasta atinge un alt punct.

În figura 2 este prezentat un exemplu al pivotării.

Prezentarea algoritmului

Algoritmul propus utilizează *paradigma avansării frontului* pentru a determina o triangulație de interpolare.

El are ca parametrii o mulțime de puncte cu coordonate tridimensionale și o rază ρ. Fiecărui punct σ_i îi este asociată o normală n_i.

În principiu, algoritmul începe prin alegerea unui triplet de puncte care formează un așa numit ***triunghi rădăcină***. Acest triunghi (σ_i, σ_j, σ_k) este ales astfel încât o bilă de rază ρ care le atinge să nu conțină nici unul dintre celelalte puncte. În continuare se folosește operația de pivotare descrisă anterior și se adaugă câte un triunghi la fiecare pas.

Frontiera (sau frontul) *F* va fi reprezentată sub forma unei colecții de liste înlănțuite care conțin laturi. Inițial, această frontieră va fi formată doar dintr-o buclă care conține laturile care determină triunghiul rădăcină.

Pentru fiecare latură $e(i, j)$ vom păstra:

- cele două extremități ale laturii: σ_i și σ_j;
- vârful opus σ_0 în triunghiul obținut atunci când a fost adăugtă latura $e(i, j)$;
- centrul c_{ij0} al bilei care atinge cele trei puncte;
- o referință spre următoarea latură care face parte din bucla care reprezintă frontul;
- o referință spre precedenta latură care face parte din bucla care reprezintă frontul.

O latură poate fi *activă*, de *frontieră* sau *înghețată*.

O ***latură activă*** este o latură care va fi folosită penru pivotare. Dacă o latură nu poate fi folosită pentru pivotare, ea devine ***latură de frontieră***.

Conceptul de latură înghețată va fi prezentat atunci când vom descrie modalitățile de îmbunătățire a performanțelor algoritmului de bază.

Păstrarea tuturor acestor informații referitoare la o latură simplifică procesul de pivotare a bilei în jurul laturii respective.

Motivul pentru care utilizăm o colecție de liste (și nu o singură listă) este dat de faptul că, pe măsură ce bila pivotează în jurul unei laturi, aceasta din urmă poate atinge un punct care a mai fost sau nu procesat. Configurația (topologia) frontului se modifică în funcție de situație.

Algoritmul de pivotare a bilei tratează toate situațiile folosind doi *operatori topologici*: ***unirea*** și ***lipirea***.

Acești operatori ne asigură că, în fiecare moment, frontul este reprezentat de către o colecție de liste înlănțuite.

Algoritmul BPA

Prezentăm în continuare versiunea în pseudocod a algoritmului de pivotare a bilei.

Pe baza acestei versiuni vom explica operațiile care au loc, precum și structurile de date care trebuie utilizate.

```
algoritm BPA(S, ρ)
  cât timp adevărat execută
                          // ciclu infinit
    e_ij ← latură_activă(F)
    cât timp e_ij ≠ nil execută
      σ_k ← pivotare(e_ij)
      dacă σ_k ≠ nil și
           (neutilizat(σ_k) sau
            pe_front(σ_k)) atunci
        scrie σ_i, σ_k, σ_j
        unește(e_ij, σ_k, F)
        dacă e_ki ∈ F atunci
          lipește(e_ik, e_ki, F)
        sfârșit dacă
        dacă e_jk ∈ F atunci
          lipește(e_kj, e_jk, F)
        sfârșit dacă
      altfel
        latură_frontieră(e_ij)
      sfârșit dacă
      e_ij ← muchie_activă(F)
    sfârșit cât timp
    (σ_i, σ_j, σ_k) ←
          triunghi_rădăcină()
    dacă (σ_i, σ_j, σ_k) ≠ nil
                          atunci
      scrie σ_i, σ_j, σ_k
      adaugă_latură(e_ij, F)
      adaugă_latură(e_jk, F)
      adaugă_latură(e_ki, F)
    altfel
      returnează
    sfârșit dacă
  sfârșit cât timp
sfârșit algoritm
```

Evident, pentru a mări lizibilitatea, nu am inclus diverse verificări suplimentare.

Laturile din frontul *F* vor fi accesate păstrând o structură de tip coadă care conține laturile active.

Operația de unire constă în adăugarea în front a două laturi active.

Operația de lipire constă în eliminarea a două laturi din front și modificarea topologiei frontului fie prin spargerea unei bucle în alte două bucle, fie prin combinarea a două bucle în una singură.

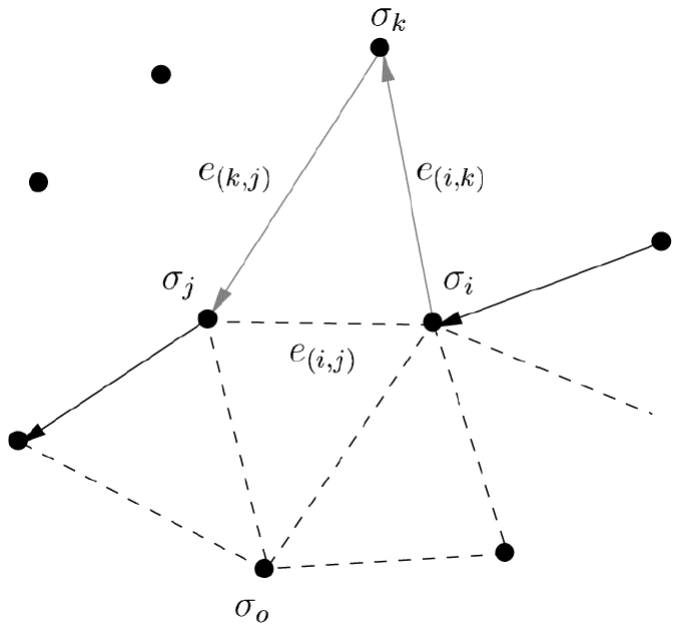

Figura 3: Operația de unire

O unire constă în simpla adăugare a unui nou triunghi, eliminarea laturii e(i, j) de pe front și adăugarea a două noi laturi e(i, k) și e(k, j).

Este utilizată și o operație care determină un triunghi rădăcină care poate fi utilizat.

Interogări spațiale

Atât operația de pivotare a bilei, cât și cea de deteminare a unui triunghi rădăcină, constau în căutarea rapidă a unei submulțimi de puncte care se află într-o vecinătate spațială relativ mică.

Această operație de interogare poate fi implementată folosindu-se o grilă de celule cubice (numite ***voxeli***). Latura unui voxel va fi întotdeauna $\delta = 2 \cdot \rho$.

Punctele sunt păstrate într-o listă și lista este organizată în așa fel încât punctele aflate în interiorul unui același voxel să se afle pe poziții consecutive.

Pentru fiecare voxel se va păstra o referință către începutul sublistei care îi corespunde (dacă sublista este vidă, referința va fi spre voxelul următor). Un voxel suplimentar aflat la capătul listei va păstra o referință vidă.

Pentru a parcurge punctele corespunzătoare unui voxel este suficient să traversăm punctele aflate în listă între referința corespunzătoare voxelului respectiv și referința corespunzătoare voxelului următor.

Pentru un punct *p* este foarte simplu să determinăm voxelul *V* căruia îi aparține dacă împărțim coordonatele sale la valoarea ρ.

De obicei va trebui să căutăm puncte aflate la o distanță de cel mult $2 \cdot \rho$ față de un punct dat *p*. Aceste puncte se află în voxelul *V* sau în cele 26 de voxeluri adiacente voxelului *V*.

Această grilă permite accesarea punctelor în timp constant. Dimensiunile sale af fi inacceptabile dacă trebuie să procesăm un set mare de date într-un singur pas.

Din acest motiv, vom modifica mai târziu algoritmul pentru a putea lucra doar cu o parte dintre puncte la un pas.

Alegerea triunghiului rădăcină

Pentru a determina un triunghi rădăcină valid este suficient să efectuăm următoarele operații:

- alegerea unui punct σ care nu apare în triangularea deja determinată;

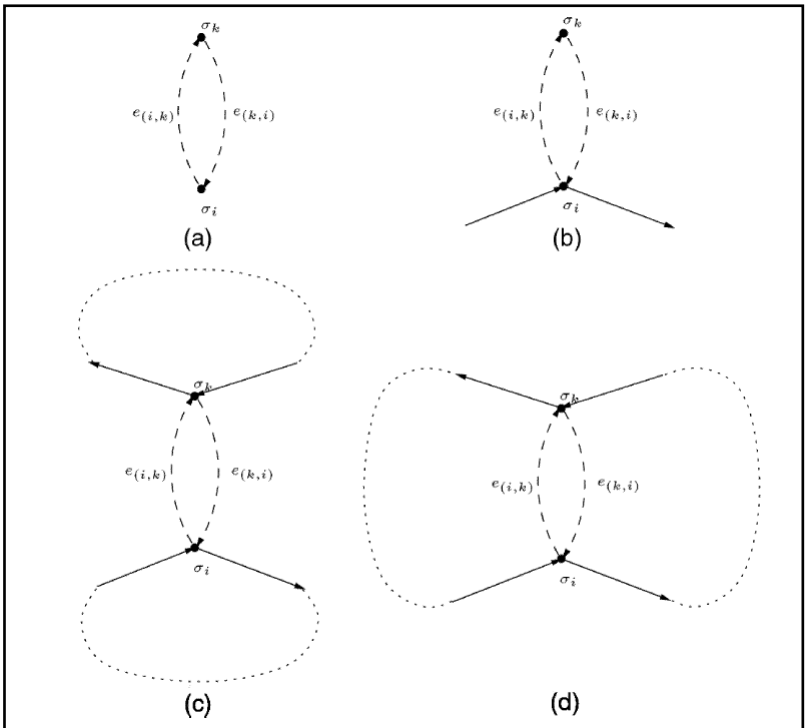

Figura 4: Operația de lipire

O lipire este efectuată atunci când printr-o unire obținem o latură care a mai fost obținută (orientată invers). Cele două laturi sunt eliminate și frontul este ajustat. Există patru posibile cazuri: (a) cele două laturi formează o buclă - bucla este eliminată din front; (b) cele două laturi fac parte din aceeași buclă și se află pe poziții consecutive - laturile sunt eliminate și bucla se scurtează; (c) laturile nu se află pe poziții consecutive, dar fac parte din aceeași buclă - bucla este împărțită în două bucle; (d) laturile nu se află pe poziții consecutive și nu fac parte din aceeași buclă - cele două bucle sunt unite în una singură

- determinarea tuturor perechilor de puncte (σ_a, σ_b) aflate în vecinătatea punctului σ, în ordinea distanțelor față de σ;
- construirea triunghiurilor rădăcină potențiale (σ, σ_a, σ_b);
- verificarea faptului că normala triunghiului respectă normalele punctelor (spre exterior);
- verificarea existenței (înspre exterior) a unei bile de rază ρ care atinge cele trei puncte, dar nu atinge nici un alt punct;
- întreruperea verificărilor în momentul identificării primului triunghi rădăcină valid.

Pivotarea bilei

O operație de pivotare are ca parametru o latura (aceasta conține și o referință spre vârful opus din triunghiul din care face parte).

Latura face parte dintr-un triunghi $\tau = (\sigma_i, \sigma_j, \sigma_0)$, va fi notată prin $e(i, j)$, iar bila se află în poziția sa inițială (centrul este punctul c_{ij0}) și nu conține în interiorul ei nici unul dintre punctele date.

Aceasta se datorează faptului că τ este fie un triunghi rădăcină, fie a fost obținut printr-o operație de pivotare anterioară.

În principiu, pivotarea este o mișcare continuă a bilei în timpul căreia bila rămâne în contact cu extremitățile laturii $e(i, j)$.

Datorită acestei mișcări centrul c_{ij0} al bilei descrie un cerc γ care se află într-un plan perpendicular pe latura $e(i, j)$ și al cărui centru este mijlocul m al laturii $e(i, j)$. Ca urmare, raza cercului γ va fi dată de distanța dintre centrul bilei ρ și mijlocul laturii $e(i, j)$.

În timpul acestei mișcări bila poate atinge un alt punct σ_k. În cazul în care nu este determinat nici un astfel de punct, atunci latura $e(i, j)$ este o latură de frontieră.

Dacă a fost întâlnit un punct, atunci triunghiul (σ_i, σ_k, σ_j) este un nou triunghi valid și bila, în poziția în care a ajuns, nu atinge nici un alt punct, deci poziția sa va fi validă pentru următoarea operație de pivotare.

În practică, vom determina punctul σ_k astfel: se iau în considerare toate punctele aflate la o distanță cel mult egală cu $2 \cdot \rho$ față de m.

Pentru fiecare astfel de punct σ_x determinăm centrul c_x al bilei care atinge punctele σ_i, σ_j și σ_x (dacă există o astfel de bilă).

Fiecare astfel de punct c_x se află pe circumferința cercului γ și poate fi determinat prin intersectarea unui sfere de rază ρ cu centrul în punctul c_x cu cercul γ.

Dintre aceste puncte c_x îl vom alege pe primul care se află pe circumferința cercului γ pornind dinspre m.

Operațiile de unire și lipire

Aceste două operații generează triunghiuri în timp ce se adaugă sau se elimină laturi în/din buclele frontului F.

Cea mai simplă operație este cea de unire; ea este utilizată în momentul în care bila care pivotează atinge un punct σ_k neutilizat anterior (σ_k nu face încă parte din plasă).

După cum se poate vedea în figura 3, în acest moment scriem triunghiul (σ_i, σ_k, σ_j) și modificăm frontul prin adăugarea laturilor $e(i, k)$, respectiv $e(k, j)$.

Dacă punctul face parte deja din plasă, distingem două situații:

- σ_k este un punct interior al plasei (nu există pe front nici o latură care să aibă una dintre extremități în punctul σ_k); în acest caz triunghiul obținut nu poate fi valid, iar latura $e(i, j)$ devine astfel o latură de frontieră.
- punctul σ_k face parte din front; în acest caz trebuie să verificăm existența și orientarea laturilor incidente punctului σ_k; vom aplica apoi o operație de unire și vom scrie noul

triunghi (σ_i, σ_k, σ_j); operația de unire ar putea duce la obținerea a una sau două perechi de laturi identice (cu orientări diferite, de exemplu $e(y, z)$ și $e(z, y)$) care vor fi eliminate prin operația de lipire.

Operația de lipire elimină din front perechi de laturi identice cu orientări diferite (algoritmul nu duce niciodată la crearea de laturi identice cu aceeași orientare).

Pot fi identificate patru cazuri care sunt prezentate în figura 4.

În figura 5 este ilustrată o secvență de operații de unire și lipire.

Înghețarea

Datorită numărului mare de voxeli care ar fi necesari, structura de date necesară ar putea fi prea mare pentru a fi utilizabilă.

Datorită faptului că algoritmul operează asupra unor zone de dimensiuni reduse la fiecare pas, putem împărți datele de intrare în ***felii***.

Practic, vom utiliza două plane care sunt paralele cu două axe dintre cele trei axe de coordonate.

Inițial planul π_0 se va afla sub toate punctele date, iar planul π_1 se va afla deasupra acestuia la o distanță aleasă convenabil pentru ca structura de date să nu aibă dimensiuni considerabile. În momentul creării unei noi laturi, verificăm dacă extremitățile sale se află deasupra planului π_1. În acest caz latura este marcată ca fiind ***înghețată***.

În momentul în care nu mai avem decât laturi înghețate, mutăm planurile (π_1 devine π_0, iar noul π_1 se va afla deasupra noului π_0 la aceeași distanță arbitrară aleasă). Totodată, toate laturile înghețate vor deveni laturi active. Singurele modificări aduse algoritmului prezentat constau în introducerea unui bucle exterioare care se ocupă de deplasarea planurilor și în adăugarea instrucțiunilor de înghețare a laturilor.

Parcurgeri succesive

Algoritmul poate fi modificat pentru a utiliza mai multe bile de raze diferite.

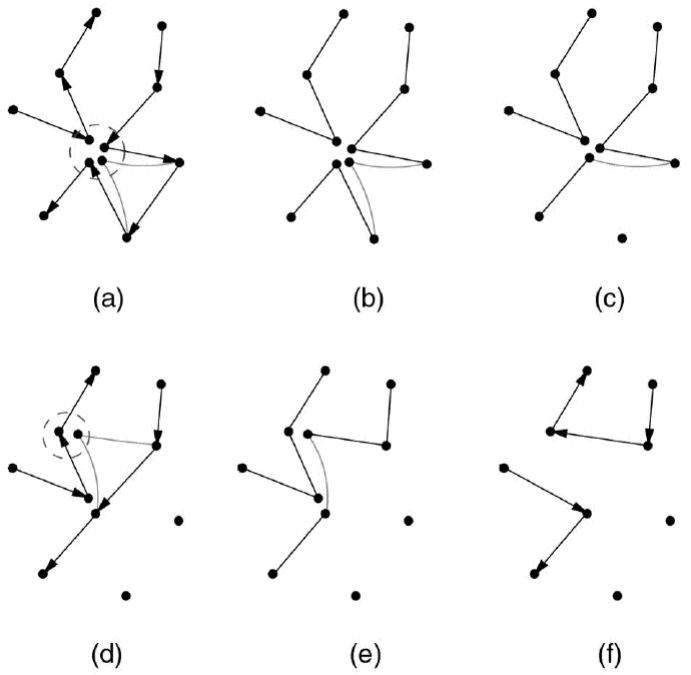

Figura 5: Secvență de uniri și lipiri

(a) trebuie adăugat frontului curent un nou triunghi; cele patru puncte ale frontului aflate în interiorul cercului punctat reprezintă, de fapt, un singur punct; (b) printr-o unire este eliminată o latură și sunt create două noi laturi care sunt identice cu alte două laturi deja existente; (c) prin două lipiri cele două perechi de laturi identice sunt eliminate; (d) un nou triunghi trebuie adăugat; (e) doar una dintre laturile create în urma acestei uniri este identică cu o latură existentă; (f) o operație de lipire elimină perechea de laturi identice

Este specificată o listă de raze ordonate crescător. Pentru fiecare felie vom considera succesiv voxeli de dimensiune $\delta = 2 \cdot \rho_i$, unde ρ_i este valoarea unei raze.

Algoritmul va rula până în momentul în care nu mai există nici o latură activă.

În acest moment trecem la următoarea rază din lista aleasă de utilizator, parcurgem toate laturile de pe front și verificăm dacă o latură împreună cu vârful opus formează un triunghi rădăcină valid pentru o sferă având noua rază.

În acest caz această nouă rază devine activă și pivotarea continuă folosind noua rază.

Concluzii

Am prezentat un algoritm care, folosind metoda avansării frontului, determină o triangulare de interpolare a unei mulțimi de puncte din spațiul tridimensional.

Algoritmul de pivotare a bilei are câteva proprietăți care îl recomandă a fi utilizat pentru recunoașterea suprafețelor:

- este intuitiv: triangularea se realizează folosind o bilă care se "plimbă" pe suprafață; în plus, utilizatorul trebuie să aleagă un singur parametru (raza bilei);
- este flexibil, eficient și robust;
- se bazează pe rezultate teoretice cunoscute.

Bibliografie

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, *The Ball-Pivoting Algorithm for Surface Reconstruction*, IEEE Transactions on Visualisation and Computer Graphics, Vol. 5, No. 4, october-december 1999, pp. 349 - 359