Univerzita Komenského
Fakulta Matematiky, Fyziky
a Informatiky
Katedra algebry, geometrie
a didaktiky matematiky
Odbor: informatika

Diplomová práca

Triangulácia implicitne definovanej plochy

Autor: Ondrej Jaborník
Školiteľ: RNDr. Pavel Chalmovianský, PhD.

Bratislava 2006

Týmto vyhlasujem, že som diplomovú prácu vypracoval samostatne s odbornou pomocou školiteľa.

Bratislava, Máj 2006 Ondrej Jaborník

Abstrakt

Táto práca sa zaoberá problémom triangulácie implicitne definovaných plôch. Popisujeme rôzne spôsoby získania triangulácie, výpočtu normál a detekcie hrán. Navrhli sme a implementovali algoritmus na trianguláciu implicitných plôch. Používame adaptívne upravenú metódu Marching Cubes na nájdenie aproximácie implicitnej plochy. Túto aproximáciu triangulujeme pomocou metódy Marching Triangles. Navrhli sme spôsob detekcie ostrých hrán na implicitnej ploche. Navrhli sme spôsob ako jednoducho rozšíriť náš algoritmus o detekciu hrán.

Kľúčové slová: Triangulácia, Implicitne definované plochy, Marching Cubes, Marching Triangles, Detekcia hrán

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam algoritmov

1 Úvod

V posledných rokoch sa stali implicitne definované plochy veľmi populárne. Oproti parametricky definovaným plochám majú niekoľko výhod. Poskytujú informáciu aj o objeme a nie len o povrchu. Sú schopné reprezentovať objekty ľubovoľnej topológie. Jednoduchá realizácia CSG a ďalších operácií ich robí ideálnymi na modelovanie. Implicitné plochy sú často používané na rekonštrukciu objektov z bodov.

Na rozdiel od parametricky definovaných plôch je určenie bodu na implicitne definovanej ploche zložitejšie. Na ich zobrazenie je nutné použiť metódy na hľadanie koreňa. Metódou ray-tracing sú dosahované fotorealistické obrázky implicitných plôch. Ray-tracingom ale nezískame informáciu o celej ploche. Vhodným riešením je aproximácia pomocou polygónov, najčastejšie trojuholníkov (triangulácia). Väčšina 3D aplikácií je schopná s takouto reprezentáciou ďalej pracovať. Zobrazovanie polygónov je štandardne akcelerované grafickými adaptérmi.

Primárnym cieľom našej práce je navrhnúť a implementovať efektívny algoritmus na trianguláciu implicitne definovaných plôch. Veľkosť trojuholníkov by sa mala adaptívne meniť podľa krivosti plochy. Vnútorné uhly trojuholníkov by mali byť $\approx 60°$. Ďalej navrhneme spôsob detekcie hrán a jednoduchý postup ako túto detekciu začleniť do nášho algoritmu.

Používajú sa dva základné princípy pri triangulácii, prehľadávanie priestoru a prehľadávanie povrchu. Prehľadávanie priestoru je založené na rozdelení priestoru na menšie oblasti, v ktorých je nezávisle riešená triangulácia. Typickými aplikáciami tohto prístupu sú metódy Marching Cubes a Marching Tetrahedras. Pri prehľadávanie povrchu sa generuje triangulácia postupne. Pozícia nového vrcholu je odhadnutá na základe už existujúcej triangulácie a následne upravená tak, aby vrchol ležal čo najbližšie k ploche.

Prvý prístup sa vyznačuje rýchlosťou (najmä modifikácia sledovania povrchu) a jednoduchosťou implementácie. Negeneruje však veľmi kvalitnú trianguláciu. Naproti tomu druhý prístup generuje veľmi kvalitnú trianguláciu. Avšak má viacero nevýhod. Treba riešiť prekrývanie triangulácie a možné zacyklenie.

Rozhodli sme sa spojiť výhody oboch prístupov, rýchlosť z prehľadávania priestoru a kvalitu z prehľadávania povrchu. Vytvoríme aproximáciu plochy pomocou metódy Marching Cubes a následne na túto aproximáciu použijeme metódu Marching Triangles.

Kapitola 2 obsahuje stručný prehľad metód používaných na trianguláciu,

odhad normál a na detekciu hrán. V kapitole 3 popisujeme nami navrhnutý algoritmus. V kapitole 4 popisujeme niektoré detaily implementácie a dosiahnuté výsledky. Kvalitu a rýchlosť nášho algoritmu porovnávame so známymi metódami a implementáciami. V kapitole 5 sme stručne zhrnuli obsah našej práce a načrtli sme oblasti ďalšieho možného vylepšenia nášho algoritmu.

2 Prehľad

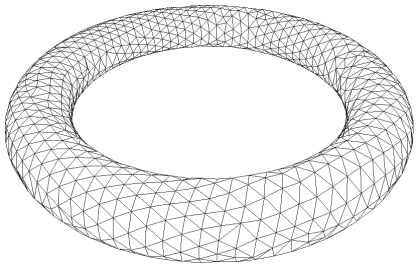
2.1 Implicitne definovaná plocha

Nech $f : \mathbb{R}^3 \to \mathbb{R}$ je funkcia. Budeme predpokladať, že f je $C^k, k \geq 0$ a po častiach polynomická. Implicitne definovaná plocha je množina bodov $Z(f) = \{\mathbf{p} \in \mathbb{R}^3 : f(\mathbf{p}) = 0\}$, teda množina koreňov alebo nulová hladina funkcie f.

Množina $Z(f)$ obvykle delí priestor na vonkajšiu a vnútornú časť. Vonkajšia časť je definovaná ako $Z_+(f) = \{\mathbf{p} \in \mathbb{R}^3 : f(\mathbf{p}) > 0\}$. Vnútorná časť je definovaná ako množina bodov $Z_-(f) = \{\mathbf{p} \in \mathbb{R}^3 : f(\mathbf{p}) < 0\}$. Teda vzťah

$$f_{torus}(x, y, z) = (x^2 + y^2 + z^2 + 3^2 - 2^2)^2 - 4 * 2^2(x^2 + y^2) \qquad (1)$$

popisuje torus v rovine $\overline{xy}$ s vonkajším polomerom 3, vnútorným polomerom 2 a stredom v $[0, 0, 0]$ (obr. 1).



Obrázok 1: Torus definovaný ako $Z(f_{torus})$ zo vzťahu 1.

Ak f je polynóm, $Z(f)$ nazývame algebraická plocha. Ak explicitné vyjadrenie f nepoznáme, môžme geometrické vlastnosti príslušnej implicitnej plochy, odhadnúť pomocou numerického vyhodnocovania f.

Hodnota $\frac{f(\mathbf{p})}{\|\nabla f(\mathbf{p})\|}$ je aproximáciou prvého rádu vzdialenosti bodu $\mathbf{p}$ od $Z(f)$.

Operácie zjednotenia, prieniku a rozdielu sú na telesách $Z_-(f)$ realizované veľmi jednoducho:

$$(f_1 \cup f_2)(\mathbf{p}) = min(f_1(\mathbf{p}), f_2(\mathbf{p}))$$
$$(f_1 \cap f_2)(\mathbf{p}) = max(f_1(\mathbf{p}), f_2(\mathbf{p}))$$
$$(f_1 \setminus f_2)(\mathbf{p}) = max(f_1(\mathbf{p}), -f_2(\mathbf{p})).$$

Výsledná funkcia však nemusí byť C^1 spojitá v bode $\mathbf{p}$, pre ktorý $f_1(\mathbf{p}) =$

$f_2(\mathbf{p})$. Preto sa niekedy používajú R-funkcie:

$$(f_1 \cup f_2)(\mathbf{p}) = f_1(\mathbf{p}) + f_2(\mathbf{p}) + \sqrt{f_1(\mathbf{p})^2 + f_2(\mathbf{p})^2}$$
$$(f_1 \cap f_2)(\mathbf{p}) = f_1(\mathbf{p}) + f_2(\mathbf{p}) - \sqrt{f_1(\mathbf{p})^2 + f_2(\mathbf{p})^2}$$
$$(f_1 \setminus f_2)(\mathbf{p}) = f_1(\mathbf{p}) - f_2(\mathbf{p}) - \sqrt{f_1(\mathbf{p})^2 + f_2(\mathbf{p})^2}.$$

Výsledná funkcia nemusí byť C^1 spojitá v bode $\mathbf{p}$, pre ktorý $f_1(\mathbf{p}) = f_2(\mathbf{p}) = 0$.

2.2 Triangulácia

Meš je definovaný ako usporiadaná trojica (V, E, F), kde $V \subset \mathbb{R}^3$ je množina vrcholov, E je množina hrán a F je množina stien spĺňajúcich štandardné požiadavky (napr. že meš neobsahuje izolované vrcholy a hrany). Presnú definíciu mešu je možné nájsť v [PS85]. Hranu $e = \{\mathbf{v_1}, \mathbf{v_2}\} \in E$ spájajúcu vrcholy $\mathbf{v_1}, \mathbf{v_2} \in V$ označujeme $\mathbf{v_1 v_2}$. V prípade, že uvažujeme orientovanú hranu, upozorníme na to. Stenu $\Pi = (\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, \ldots, \mathbf{v_n})$ označujeme $\overline{\mathbf{v_1 v_2 v_3 \ldots v_n}}$.

Meš $M = (V, E, F)$ nazývame trianguláciou, ak pre každú stenu $\Pi \in F$ platí, že Π je tvaru $\overline{\mathbf{v_1 v_2 v_3}}$.

Triangulácia implicitnej plochy $Z(f)$ je triangulácia, ktorá aproximuje $Z(f)$. Na meranie kvality aproximácie môžme použiť vzdialenosti vrcholov mešu od $Z(f)$, alebo vzdialenosť ťažísk trojuholníkov od $Z(f)$. Ak poznáme veľkosť povrchu S plochy $Z(f)$, kvalitu aproximácie môžme merať porovnaním S a súčtu obsahov trojuholníkov triangulácie. Pre lepšiu numerickú stabilitu a aj vizuálnu kvalitu mešu je výhodné používať trojuholníky, ktorých uhly sú $\approx 60°$.

2.3 Prehľadávanie priestoru

Techniky prehľadávania priestoru sú založené na vyhodnocovaní implicitnej funkcie vo vrcholoch mriežky. Jednotlivé bunky sú triangulované nezávisle. Triangulácia je obvykle priamo určená znamienkami funkcie f v jednotlivých vrcholoch bunky. Ako bunky sa najčastejšie používajú kocky (odsek 2.3.1). Použitie iných buniek je popísané napr. v [NB93] [CTM03]). Polygón spájajúci priesečníky hrán bunky a implicitnej plochy sa nazýva záplata.

Zo spojitosti funkcie f vyplýva, že ak sú znamienka funkcie v dvoch susedných vrcholoch mriežky opačné, tak na úsečke e spájajúcej tieto dva body leží bod $\mathbf{p}$, v ktorom má funkcia hodnotu nula. Teda ak označíme $\mathbf{p_1}$

a $\mathbf{p_2}$ krajné body úsečky e, platí:

$$f(\mathbf{p_1})f(\mathbf{p_2}) < 0 \Rightarrow$$
$$\exists s,t \in \mathbb{R}_0^+ : s+t = 1 \wedge \mathbf{p} = s\mathbf{p_1} + t\mathbf{p_2} \wedge f(\mathbf{p}) = 0.$$

Na približné určenie bodu $\mathbf{p}$ sa najčastejšie používa lineárna interpolácia

$$\mathbf{p} \doteq \frac{f(\mathbf{p_2})\mathbf{p_1} - f(\mathbf{p_1})\mathbf{p_2}}{f(\mathbf{p_2}) - f(\mathbf{p_1})}$$

alebo bisekcia (pozri alg. 1). Lineárna interpolácia je rýchla no nie vždy presná metóda, naproti tomu pri použití bisekcie získame výsledky s vopred určenou presnosťou.

Algoritmus 1: Bisekcia

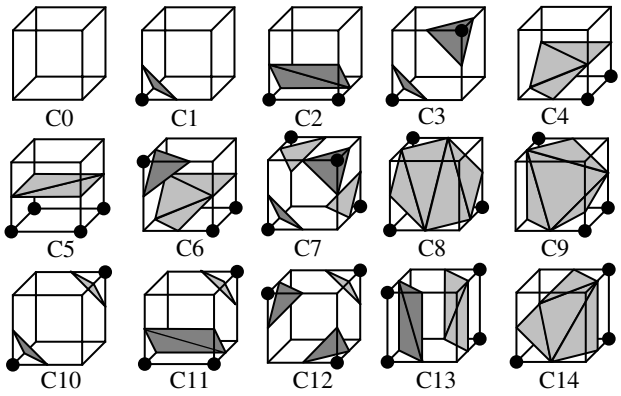
vstup : $\mathbf{p_1}$, $\mathbf{p_2}$, ϵ
výstup: $\mathbf{p}$ taký, že $f(\mathbf{p}) < \epsilon$
if `f(` $\frac{\mathbf{p_1}+\mathbf{p_2}}{2}$ `)` $< \epsilon$ **then**
  return $\frac{\mathbf{p_1}+\mathbf{p_2}}{2}$;
else
  if `f(`$\mathbf{p_1}$`)` `f(`$\frac{(\mathbf{p_1}+\mathbf{p_2})}{2}$`)` < 0 **then**
    return `Bisekcia(`$\mathbf{p_1}$, $\frac{\mathbf{p_1}+\mathbf{p_2}}{2}$, $=\epsilon$`)`;
  else
    return `Bisekcia(`$\frac{\mathbf{p_1}+\mathbf{p_2}}{2}$, $\mathbf{p_2}$, ϵ`)`;
  end
end

2.3.1 Marching Cubes

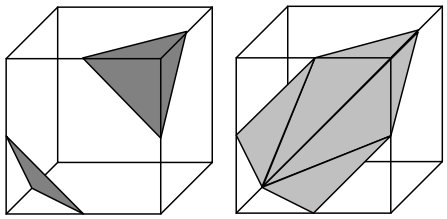
Metóda Marching Cubes (MC), ako názov napovedá, delí priestor na kocky. Pre kocky je $8^2 = 256$ rôznych konfigurácií znamienok vo vrcholoch. Využitím symetrií kocky sa tento počet dá znížiť na 15 (pozri obr. 2). Podrobnejšie o metóde MC a modifikáciách napr. v [LC87], [Nie03], [Blo88].

Problémom pri metóde MC je nejednoznačnosť niektorých konfigurácií. Napr. konfigurácia C3 môže byť triangulovaná dvoma spôsobmi (pozri obr. 3). Pri nesprávanom výbere môžu pri susedných kockách vznikať diery v triangulácii.

Jeden z častých spôsobov výberu konfigurácie používa hodnoty vo vhodne zvolenom bode v nejednoznačnej stene. Týmto bodom môže byť napr. stred steny, hodnota je určená nasledovane $v = \frac{B_{00}+B_{01}+B_{10}+B_{11}}{4}$, kde B_{00}, B_{01}, B_{10}, B_{11} sú hodnoty funkcie v príslušných vrcholoch steny. V [NH91] je použitý priesečník asymptot kontúr bilineárneho interpolantu.



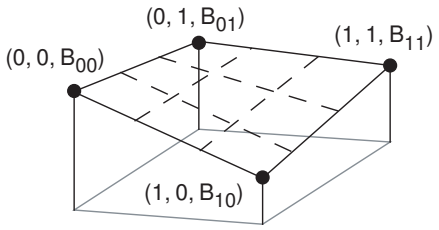
Obrázok 2: 15 základných konfigurácií znamienok vo vrcholoch kocky.



Obrázok 3: Nejednoznačná konfigurácia C3.

Bilineárny interpolant je určený vzťahom

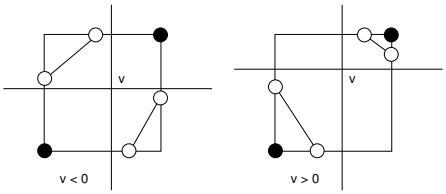
$$B(s,t) = (1-s, s) \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \begin{pmatrix} 1-t \\ t \end{pmatrix}.$$



Obrázok 4: Bilineárny interpolant.

Kontúry bilineárneho interpolantu sú hyperboly. Hodnota v priesečníku ich asymptot (t.j. hodnota, na základe ktorej rozhodujeme) je

$$v = \frac{B_{00}B_{11} + B_{10}B_{01}}{B_{00} + B_{11} - B_{01} - B_{10}}.$$



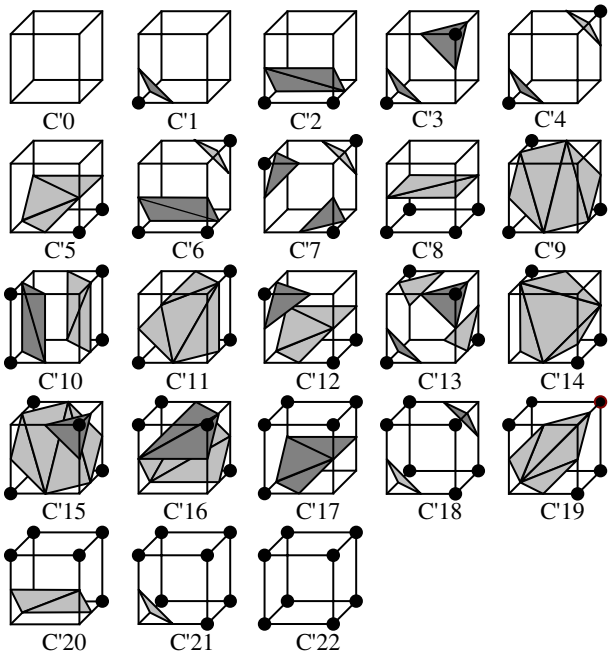
Obrázok 5: Asymptotické rozhodovanie.

Ak je hodnota v vo vybranom bode menšia ako 0, sú priesečníky spojené tak, aby tento ležal vo vnútri triangulovaného objektu, ak väčšia ako 0, potom sú spojené tak, aby ležal mimo triangulovaného objektu. Na obr. 5 je znázornené spojenie priesečníkov pre asymtotické rozhodovanie.

Problém s nejednoznačnosťou možno odstrániť aj modifikáciou konfigurácií. Vybrané sú také základné konfigurácie, aby vrcholy ležiace v nejednoznačnej stene boli spojené vždy rovnako (pozri [NHS02]). Príklad takýchto konfigurácií je na obrázku 6. Pri výbere základnej konfigurácie sa využívajú iba rotácie kocky bez využitia symetrie, preto je tento výber jednoznačný.

Rozdelením kocky na štvorsteny a ich nezávislou trianguláciou sa rovnako môžme vyhnúť nejednoznačným konfiguráciám (pozri [NB93]) .

Veľkosť kociek určuje aké veľké detaily budú s istotou vo výslednej triangulácii zahrnuté. Napríklad valec s priemerom menším ako d, kde d je veľkosť použitých kociek, nemusí byť zachytený vo výslednej triangulácii.



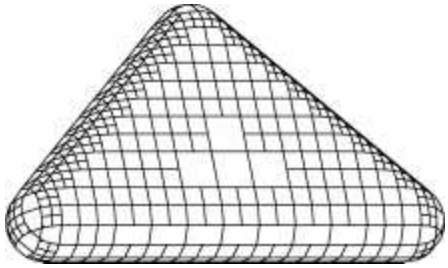
Obrázok 6: 23 základných konfigurácií znamienok vo vrcholoch kocky.

2.3.2 Adaptívna modifikácia MC

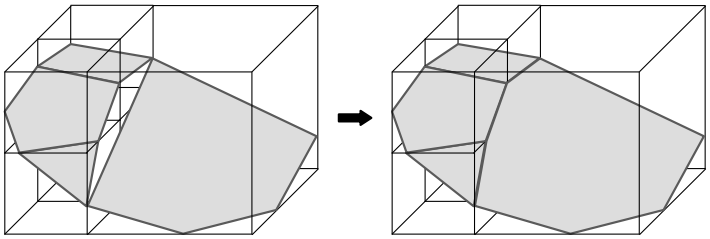
Pri delení priestoru na rovnako veľké kocky máme minimálnu kontrolu nad veľkosťou a tvarom výsledných trojuholníkov. Tento problém čiastočne rieši adaptívna prispôsobovanie veľkosti kocky. Adaptívna modifikácia spočíva v použití hustejšej mriežky "tam, kde je to treba". Na týchto miestach je kocka prerozdelená (t.j. budujeme v nej octree), čo vo výslednej triangulácii znamená vytvorenie menších trojuholníkov a lepšiu aproximáciu $Z(f)$. Kritériá na prerozdelenie sú najčastejšie tieto:

- planarita záplaty
- divergencia normál vrcholov záplaty
- dosiahnutie maximálneho alebo minimálneho prerozdelenia
- vzdialenosť od implicitnej plochy

Na spoločnej stene dvoch nerovnako prerozdelených kociek sa môžu vyskytnúť diery (pozri obr. 7). Preto treba do záplaty v menej prerozdelenej kocke pridať vrcholy zo spoločnej steny (pozri obr. 8). Podrobnejšie v [Blo88] alebo [BS01].



Obrázok 7: Adaptívne prerozdelenie [Blo].

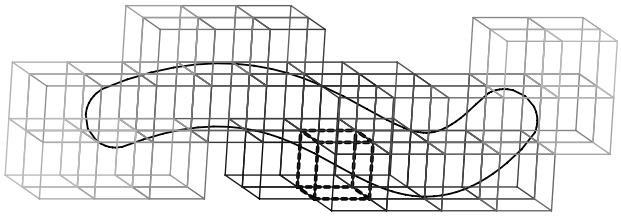


Obrázok 8: Spojenie pri nerovnakom prerozdelení susedných kociek.

2.3.3 MC - Sledovanie povrchu

Tento prístup pri vytváraní buniek sleduje povrch, teda vytvára iba bunky pretínajúce $Z(f)$ (pozri [Blo88]). Vytvárajú sa iba bunky susedné k už existujúcim a iba v smere stien pretínajúcich $Z(f)$. Počiatočná bunka môže byť vstupom pre algoritmus alebo ju je možné určiť iteráciou.

Vyhodnotené bunky sa najčastejšie ukladajú do grafu susedností, aby sa zabránilo viacnásobnému vyhodnocovaniu tej istej bunky. Pre použitie adaptívnej metódy je však výhodnejšie používať octree a metódy na hľadanie suseda v octree. Pridávaním nových buniek je treba vytvárať nové rodičovské uzly a korene.

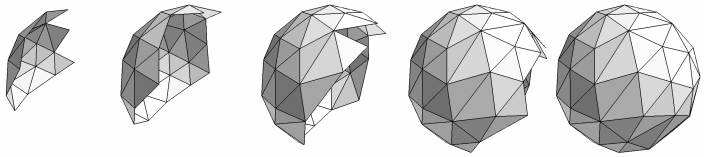


Obrázok 9: Sledovanie povrchu [Blo88].

Príklad metódy MC so sledovaním povrchu je na obr. 9. Počiatočná kocka je znázornená čiarkovane, kocky vyhodnocované skôr sú tmavšie.

Časová zložitosť metódy sledovania povrchu je $O(n^2)$, kde n je maximálny počet kociek v axiálnom smere. Čo je oproti zložitosti $O(n^3)$ pri prehľadávaní celého priestoru nespornou výhodou. Nevýhodou je schopnosť triangulovať iba jeden súvislý objekt.

2.4 Prehľadávanie povrchu technikou predpovede a opravy



Obrázok 10: Prehľadávanie povrchu technikou predpovede a opravy (obrázok z [MV04]).

Metódy prehľadávania povrchu postupne spájajú vrcholy patriace $Z(f)$ do polygónov najčastejšie trojuholníkov(pozri 2.4.1). Nový vrchol sa vypočíta posunutím známeho vrcholu $\mathbf{p_i}$ po dotykovej rovine k $Z(f)$ v bode $\mathbf{p_i}$. Pozícia nového vrcholu je upravená tak, aby vrchol ležal na implicitnej ploche (napr. použitím Newtonovej iterácie). Pri vytváraní nového polygónu treba kontrolovať, či nepríde k prekrytiu už existujúceho polygónu.

2.4.1 Marching Triangles

Metóda Marching Triangles (MT) je najčastejšie používanou aplikáciou prehľadávania povrchu. Popíšeme hlavnú myšlienku metódy z [Har03]. Front triangulácie je definovaný ako postupujúca lomená čiara na hranici už vytvorenej triangulácie. Základné kroky algoritmu sú:

Krok 0 Vychádzame z vrcholu $\mathbf{p_1} \in Z(f)$. Okolo $\mathbf{p_1}$ vytvoríme v dotykovej rovine pravidelný šesťuholník $\mathbf{q_2}, \ldots, \mathbf{q_7}$. Vrcholy $\mathbf{q_2}, \ldots, \mathbf{q_7}$ premietneme na plochu (vrcholy $\mathbf{p_2}, \ldots, \mathbf{p_7}$). Vrcholy $\mathbf{p_2}, \ldots, \mathbf{p_7}$ tvoria aktuálny front triangulácie(obr. 11 a).

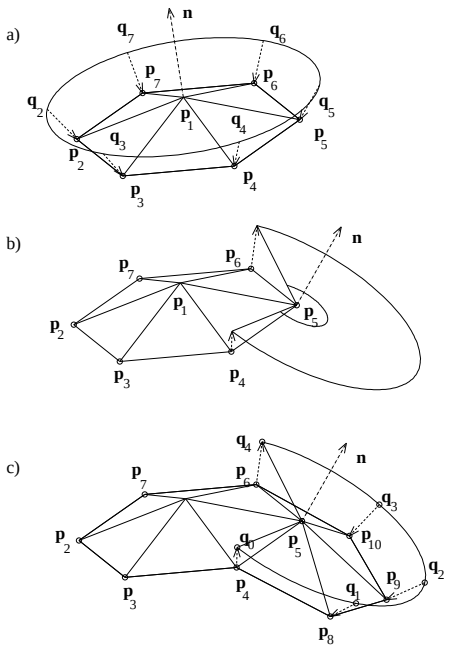
Krok 1 Pre každý vrchol frontu triangulácie spočítame uhol netriangulovanej časti (obr. 11 b).

Krok 2 Ak je niektorý vrchol $\mathbf{p_i}$ patriaci aktuálnemu frontu triangulácie "blízko":

- vrcholu $\mathbf{p_j}$, ktorý patrí tomu istému frontu triangulácie, potom pridaním hrany $\mathbf{p_i p_j}$ ho rozdelíme na dva menšie fronty triangulácie.
- vrcholu $\mathbf{p_j}$, ktorý nepatrí tomu istému frontu triangulácie, potom pridaním hrany $\mathbf{p_i p_j}$ spojíme tieto dva fronty do jedného.

Krok 3 Určíme vrchol $\mathbf{p_m}$, ktorý má najmenší uhol netriangulovanej časti. Okolo vrcholu $\mathbf{p_m}$ vytvoríme trojuholníky s vnútornými uhlami $\approx 60°$. Vrchol $\mathbf{p_m}$ odstránime z frontu triangulácie a pridáme do neho novovytvorené vrcholy (obr. 11 c).

Krok 4 Kroky 1-3 opakujem, až pokým aktuálny front triangulácie nebude mať iba 3 vrcholy, ktoré spojíme do nového trojuholníka. Ak existuje ešte nejaký front triangulácie, nastavíme ho ako aktuálny.



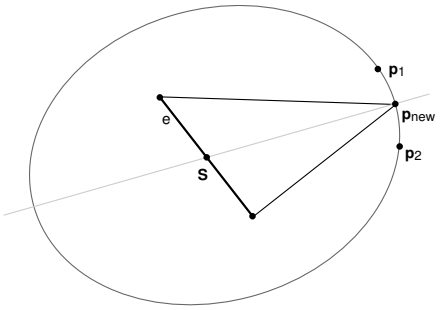
Obrázok 11: Metóda Marching Triangles a) vytvorenie prvého šesťuholníka; b) určenie uhla netriangulovanej časti; c) vytvorenie trojuholníkov [Har03].

Použitie Delaunayovej podmienky pri vytváraní triangulácie je demonštrované v [HI97]. Namiesto vrcholov sa spracúvajú hrany. Nespracované hra-

ny sú udržiavane v zozname. Aktuálne spracúvaná hrana sa nazýva aktívna hrana. $M' = \{T_0, \ldots, T_n\}$ je už vytvorená triangulácia implicitnej plochy. Nový trojuholník T_{new} je pridaný do triangulácie $M = M' \cup T_{new}$, ak spĺňa nasledujúcu podmienku:

3D Delaunayova podmienka Trojuholník $T_{new} = \overline{\mathbf{p_i p_j p_{new}}}$ môže byť pridaný do hranice mešu na hrane $e(\mathbf{p_i}, \mathbf{p_j})$, ak žiadna časť triangulácie M' s rovnakou orientáciou neleží vnútri gule opísanej trojuholníku T_{new} so stredom c_T. Bod c_T je stred opísanej kružnice vrcholov $\mathbf{p_i}, \mathbf{p_j}, \mathbf{p_{new}}$.

Modifikácia pri hľadaní nového vrcholu $\mathbf{p_{new}} \in Z(f)$ je predstavená v [ČS02]. Namiesto Newtonovej iterácie je použitá bisekcia po kružnici (pozri obr. 12). Stred kružnice leží v strede spracúvanej hrany $\mathbf{e}$ a leží v rovine kolmej na $\mathbf{e}$. Takto získaná triangulácia obsahuje trojuholníky s veľkosťami vnútorných uhlov $\approx 60°$.

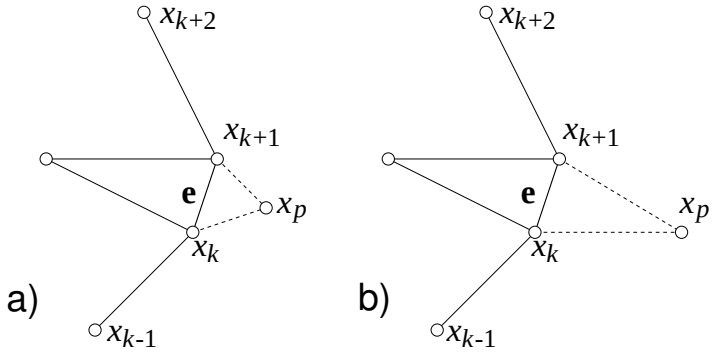


Obrázok 12: Metóda bisekcie po kružnici (Edge Spinning).

2.4.2 Adaptívne modifikácie

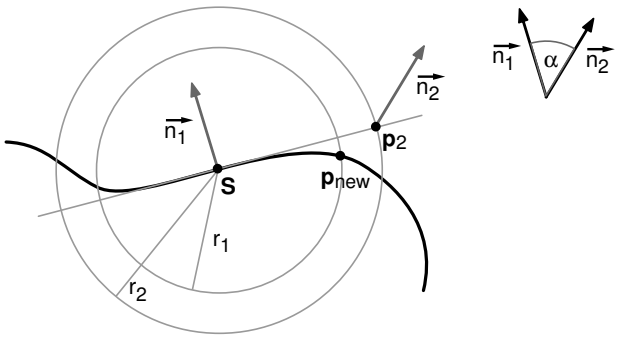
Adaptívna modifikácia metódy MT predstavená v [AG01] používa na odhad veľkosti nového trojuholníka veľkosť aktívnej hrany a jej susedov. Na obrázku 13 vľavo je zobrazený adaptívny výber nového vrcholu $\mathbf{x_p}$ a vpravo neadaptívny. Veľkosť trojuholníkov sa síce zmenšuje na miestach s väčšou krivosťou, ale nedochádza k ich zväčšovaniu pri prechode na miesta s menšou krivosťou.

Iný prístup k adaptívnej triangulácii založený na divergencii normál je opísaný v [ČS04]. Stred aktívnej hrany označíme S. Normálu v S označíme $\overrightarrow{n_1}$. Normálu v novom vrchole označíme $\overrightarrow{n_2}$. Uhol normál $\overrightarrow{n_1}$ a $\overrightarrow{n_2}$ označíme α. Pozíciu hľadaného vrcholu posunieme tak, aby ležal na kružnici so stredom v S a polomerom $r_2 = r_1 \dot{k}, k \in \langle 0, 1 \rangle$ (pozri obr. 14), kde polomer r_1 je vždy



Obrázok 13: Adaptívna metóda MT a) poloha $\mathbf{x_p}$ je upravená podľa veľkosti hrany e; b) poloha $\mathbf{x_p}$ nie je upravená.

rovnaký a určuje požadovanú veľkosť detailov zachytených v triangulácii. Ak je α menšia ako limitný uhol α_{lim} použijeme $k = \left(\frac{\alpha_{lim}-\alpha \cdot c}{\alpha_{lim}}\right)$, v opačnom prípade použijeme $k = k_{min}$, kde k_{min} minimálna prípustná hodnota k. Ak je $k < k_{min}$, použijeme $k = k_{min}$. Konštanta c určuje rýchlosť akou sa zmenšuje polomer vzhľadom na α. Autori použili nasledujúce hodnoty: $\alpha_{lim} = \pi/2$, k_{min} =0,2 a c =1,2.



Obrázok 14: Adaptívna metóda Edge Spinning.

2.5 Výpočet normál

Normála k $Z(f)$ v bode $\mathbf{p} = [x, y, z]$ je definovaná ako gradient funkcie f v bode $\mathbf{p}$:

$$\nabla f(x,y,z) = \begin{pmatrix} \frac{\partial f}{\partial x}(x,y,z) \\ \frac{\partial f}{\partial y}(x,y,z) \\ \frac{\partial f}{\partial z}(x,y,z) \end{pmatrix}.$$

Uvedený vzťah ale možno použiť iba v prípade, že vieme určiť parciálne derivácie. Ak nepoznáme vyjadrenie f, musíme použiť numerické metódy. Častým spôsobom výpočtu normály je použitie asymetrickej diferencie

$$\nabla f(x,y,z) = \begin{pmatrix} f(x,y,z) - f(x-\delta,y,z) \\ f(x,y,z) - f(x,y-\delta,z) \\ f(x,y,z) - f(x,y,z-\delta) \end{pmatrix} \cdot \frac{1}{\delta}$$

alebo symetrickej diferencie

$$\nabla f(x,y,z) = \begin{pmatrix} f(x+\delta,y,z) - f(x-\delta,y,z) \\ f(x,y+\delta,z) - f(x,y-\delta,z) \\ f(x,y,z+\delta) - f(x,y,z-\delta) \end{pmatrix} \cdot \frac{1}{2\delta}.$$

Uvedený prístup ale vyžaduje ďalšie 3 (resp. 6 pre symetrickú diferenciu) vyhodnotenia implicitnej funkcie.

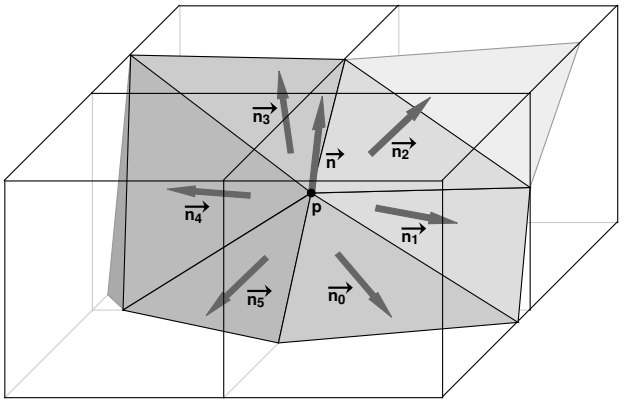
Ak poznáme hodnoty implicitnej funkcie f iba vo vrcholoch ležiacich na pravidelnej mriežke (metóda Marching Cubes), je možné použiť nasledujúci postup (pozri [LC87]). Najskôr vypočítame normálu vo vrcholoch mriežky podľa vzťahu

$$\nabla N(i,j,k) = \begin{pmatrix} \frac{F(i+1,j,k)-F(i-1,j,k)}{2\Delta x} \\ \frac{F(i,j+1,k)-F(i,j-1,k)}{2\Delta y} \\ \frac{F(i,j,k+1)-F(i,j,k-1)}{2\Delta z} \end{pmatrix},$$

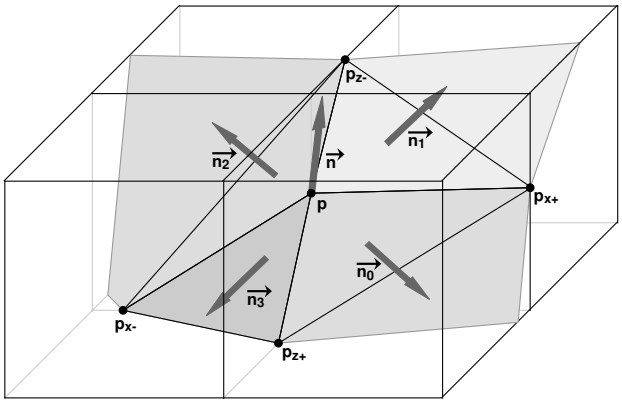
kde $N(i,j,k)$ je gradient vo vrchole $[i,j,k]$, $F(i,j,k)$ je hodnota implicitnej funkcie vo vrchole $[i,j,k]$ a Δx (resp. Δy, Δz) je veľkosť kocky v smere x (resp. y, z). Normála v priesečníku hrany kocky a implicitnej plochy sa vypočíta lineárnou interpoláciou.

Ďalší spôsob výpočtu normál pri použití metódy Marching Cubes využíva normály lokálnych lineárnych aproximácií plochy. Nech $\mathbf{p}$ je vrchol, v ktorom zisťujeme normálu. Normály trojuholníkov, ktoré obsahujú vrchol $\mathbf{p}$ označíme $\overrightarrow{n_1}, \overrightarrow{n_2}, \ldots, \overrightarrow{n_n}$ (pozri obr. 15). Normálu $\overrightarrow{n}$ vo vrchole $\mathbf{p}$ vypočítame ako vážený priemer normál $\overrightarrow{n_1}, \overrightarrow{n_2}, \ldots, \overrightarrow{n_n}$, teda $\overrightarrow{n} = \sum_{i=1}^{n} w_i \overrightarrow{n_i}$. Váhy w_i majú najčastejšie hodnoty $w_i = 1$ alebo $w_i = 1 - \frac{S_i}{S}$, kde S_i je obsah príslušného trojuholníka a $S = \sum_{i=1}^{n} S_i$. Nevýhodou tohto prístupu je nutná znalosť topológie triangulácie plochy a závislosť na konkrétnej triangulácii.

Uvedený postup možno modifikovať tak, aby znalosť topológie nebola potrebná (pozri [NHS02]). Popíšeme prípad, keď vyšetrovaný vrchol $\mathbf{p}$ leží na hrane rovnobežnej s osou y. Vrcholy ležiace v rovine $(\overline{xy})$ označíme ako $\mathbf{p_{x+}}$ (vrchol s väčšou súradnicou x ako $\mathbf{p}$) a $\mathbf{p_{x-}}$ (vrchol s menšou súradnicou x ako $\mathbf{p}$). Vrchol ležiace v rovine $(\overline{yz})$ označíme ako $\mathbf{p_{z+}}$ a $\mathbf{p_{z-}}$. Normálu



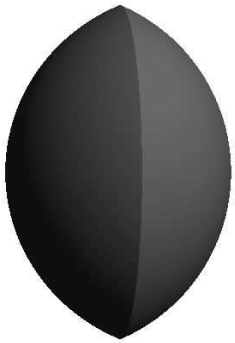
Obrázok 15: Výpočet normály v bode **p** s využitím topológie.



Obrázok 16: Výpočet normály v bode **p** bez využitia topológie.

vypočítame ako vážený priemer normál trojuholníkov $\overline{\mathbf{pp_{z+}p_{x+}}}$, $\overline{\mathbf{pp_{x+}p_{z-}}}$, $\overline{\mathbf{pp_{z-}p_{x-}}}$ a $\overline{\mathbf{pp_{x-}p_{z+}}}$ (pozri obr. 16). Prípady, keď $\mathbf{p}$ leží na hrane rovnobežnej s osou x alebo z sa riešia analogicky.

2.6 Detekcia hrán



Obrázok 17: Povrch prienik dvoch gúľ definovaný ako $Z(f_1 \cap f_2)$ zo vzťahu 2.

Ostrú hranu definujeme ako množinu bodov $SE(f) = \{\mathbf{p} : f(\mathbf{p}) = 0 \wedge \nabla f(\mathbf{p}) = \overrightarrow{0}\}$. Ak použijeme na funkcie definujúce dve hladké plochy niektorú zo skôr uvedených funkcií $\cup$, $\cap$ alebo $\setminus$, plocha definovaná výslednou funkciou bude obsahovať ostrú hranu. Na obrázku 17 je znázornená $Z(f_1 \cap f_2)$, kde

$$\begin{gathered} f_1(x,y,z) = (x-0.5)^2 + y^2 + z^2 - 1 \text{ a} \\ f_2(x,y,z) = x^2 + y^2 + z^2 - 1. \end{gathered} \tag{2}$$

Popíšeme niektoré možné metódy detekcie ostrých hrán.

Ak poznáme vyjadrenie čiastkových funkcií, na ktorých prieniku vzniká ostrá hrana, môžme použiť metódu z [Har03]. Vrchol $\mathbf{p_{k+1}} \in SE(f)$ je určený pomocou algoritmu 2. Vstupom je bod $\mathbf{q_0} = p_k + s\overrightarrow{t_k}$, kde $\overrightarrow{t_k} = \frac{\triangle f_1(\mathbf{p_k}) \times \triangle f_2(\mathbf{p_k})}{\| \triangle f_1(\mathbf{p_k}) \times \triangle f_2(\mathbf{p_k}) \|}$.

Ďalší spôsob spočíva vo využití duálnych mešov a minimalizácie energie (pozri [Hop94], [OB02]). Popíšeme metódu z [OB02]. Primárny meš často neposkytuje dostatočnú aproximáciu (pozri obr. 18 a), preto uvažujeme duálny meš. Vrcholy duálneho mešu premietneme na $Z(f)$ (pozri obr. 18 a, b). Zistíme dotykové roviny vo vrcholoch modifikovaného duálneho mešu (pozri obr. 18 c). Pozíciu vrcholov primárneho mešu upravíme minimalizáciou štvorcov vzdialeností od dotykových rovín v susedných vrcholoch duálneho

Algoritmus 2: Vrchol na hrane

vstup : $\mathbf{q_0}$, ϵ

výstup: $\mathbf{p_{new}}$

repeat

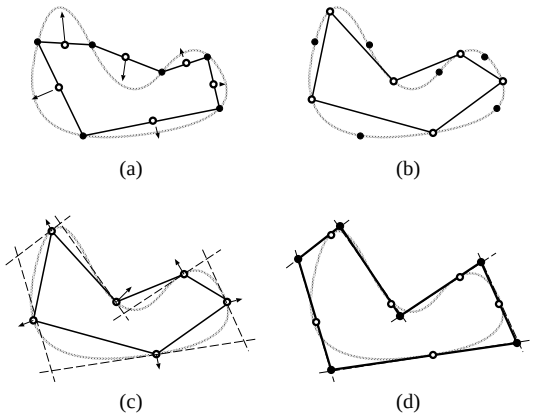
$\mathbf{q_{k+1}} = \mathbf{q_k} + \Delta_k$, kde $\Delta_k = \alpha_k \nabla f_1(\mathbf{q_k}) + \beta_k \nabla f_2(\mathbf{q_k})$ a

$\Delta_k \cdot \nabla f_j(\mathbf{q_k}) = -f_j(\mathbf{q_k}), j = 1, 2.$

until $\|\mathbf{p_{k+1}} - \mathbf{p_k}\| < \epsilon$;

return $\mathbf{p_{k+1}}$

mešu (pozri obr. 18 d). V spojení s adaptívnym prevzorkovaním a prerozdelením mešu a následnými iteráciami sú dosiahnuté lepšie výsledky. Nevýhodou tejto metódy je časté vyhodnocovanie funkcie f.

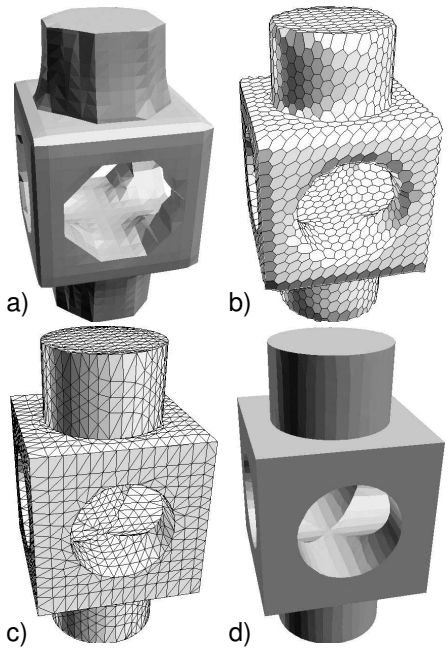


Obrázok 18: Optimalizácia primárneho mešu použitím duálneho mešu a) primárny meš a normály vo vrcholoch duálneho mešu; b) duálny meš; c) dotykové roviny vo vrcholoch duálneho mešu; d) optimalizovaný primárny meš [OB02].

Na obrázku 19 je zobrazený príklad využitia duálneho mešu.

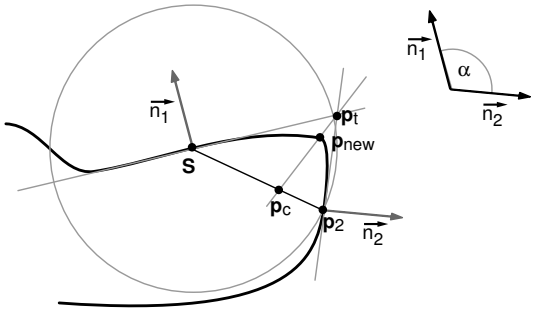
Ďalšia metóda, ktorá deteguje ostré hrany na už vytvorenom meši, využíva divergenciu normál (pozri [KBSS01], [AFRS03]). Ak sa normály vo vrcholoch záplaty líšia viac ako nastavená prahová hodnota, predpokladáme ostrú hranu. Na nájdenie bodu na hrane použijeme minimalizáciu štvorcov vzdialeností od dotykových rovín vo vrcholoch záplaty.

Detekcia ostrých hrán počas behu algoritmu MT (modifikácia Edge Spinning) je predstavená v [ČS03]. Ak je uhol normály $\overrightarrow{n_2}$ v novom bode $\mathbf{p_2}$ a normály $\overrightarrow{n_1}$ v strede $\mathbf{S}$ kružnice c väčší ako α_{min}, predpokladáme na ploche ostrú hranu. Nájdeme bod $\mathbf{p_t}$ ako priesečník dotykových rovín v bo-



Obrázok 19: Príklad optimalizácie primárneho mešu použitím duálneho mešu a) meš vygenerovaný metódou MC; b) duálny meš; c) a d) optimalizovaný primárny meš [OB02].

doch $\mathbf{S}$ a $\mathbf{p_2}$ a roviny kružnice c. Na úsečke $\overline{\mathbf{Sp_2}}$ zvolíme bod $\mathbf{p_c}$ tak, aby $\|\mathbf{Sp_c}\| : \|\mathbf{p_2p_c}\| = \|\mathbf{Sp_t}\| : \|\mathbf{p_2p_t}\|$. Ak je $f(\mathbf{p_t})f(\mathbf{p_c}) < 0$ použijeme $\mathbf{p_{new}}$ určený bisekciou, v opačnom prípade použijeme $\mathbf{p_{new}} = \mathbf{p_t}$. Takto vzniknuté trojuholníky v okolí ostrej hrany môžu mať zdeformovaný tvar.



Obrázok 20: Detekcia hrán počas behu MT (Edge Spinning).

3 Popis algoritmu

Vstupom algoritmu je spojitá funkcia f. Cieľom algoritmu je vytvoriť trianguláciu, ktorá aproximuje $Z(f)$. Veľkosť trojuholníkov sa adaptívne mení podľa krivosti plochy.

Na prvé priblíženie implicitnej plochy použijeme adaptívnu metódu Marching Cubes (odsek 3.1). Na takto vytvorenej aproximácii hľadáme ostré hrany (odsek 3.2). Metódou MT triangulujeme $Z(f)$ aproximovanú pomocou záplat (odsek 3.3), pričom zachovávame ostré hrany, ktoré sme našli v predchádzajúcom kroku. Výslednú trianguláciu premietneme na $Z(f)$ (odsek 3.4).

3.1 Marching cubes - prehľadávanie povrchu

Poznáme pozíciu a veľkosť kocky, ktorá pretína $Z(f)$. Veľkosť kocky zodpovedá veľkosti detailov, ktoré chceme mať zahrnuté v triangulácii.

Hranu kocky budeme označovať k-hrana. Na výpočet priesečníka k-hrany a plochy použijeme lineárnu interpoláciu

$$\mathbf{p} = t\mathbf{p_1} + (1-t)\mathbf{p_2}, \text{ kde } t = \frac{f(\mathbf{p_2})}{f(\mathbf{p_2}) - f(\mathbf{p_1})}.$$

Na miestach s väčším detailom (s väčšou krivosťou) dôjde k prerozdeleniu (odsek 3.1.2) a pozícia priesečníka sa spresní. V prerozdelených kockách budujeme octree.

3.1.1 Konfigurácie

Pretože kladieme dôraz na rýchlosť, nejednoznačnosť konfigurácií nebudeme riešiť, namiesto toho sa prikloníme k použitiu konfigurácií z [Nie04], ktoré sú navrhnuté tak, aby susedné záplaty na seba nadväzovali.

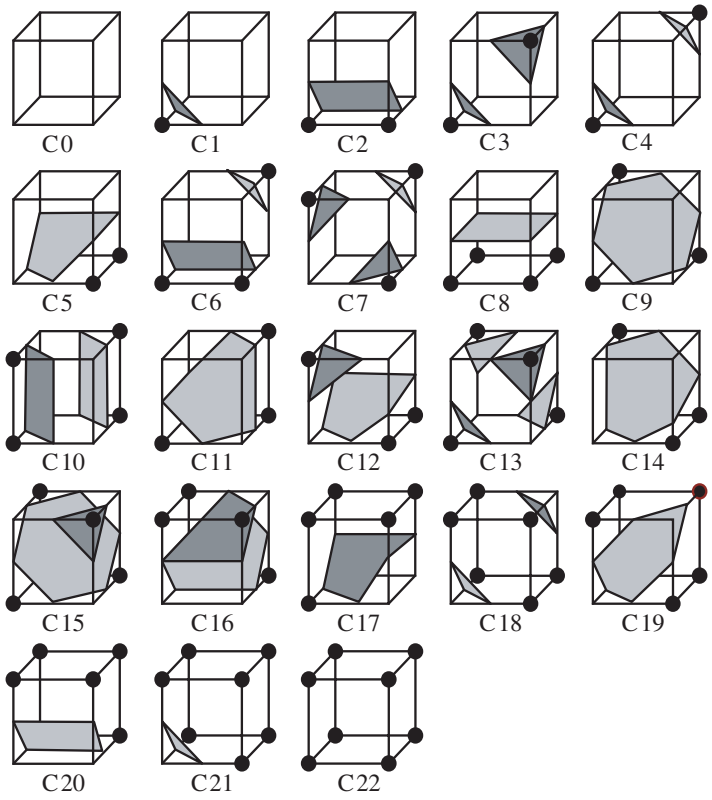
Použijeme základné konfigurácie na obr. 21. Rozdiel oproti konfiguráciám z obr. 6 je v tom, že z každej kocky použijeme iba záplaty a nie triangulácie. Zvyšné konfigurácie získame rotáciou vrcholov niektorej z uvedených základných konfigurácií (podrobnejšie v [NHS02]).

V prípade, že pre niektorý vrchol v platí $f(v) = 0$ položíme $f(v) = \epsilon_0$, kde ϵ_0 je malá kladná hodnota. Vzhľadom na to, že trianguláciu v poslednom kroku premietneme na $Z(f)$, kvalitu aproximácie ovplyvníme minimálne. Na druhej strane bude výber konfigurácie jednoznačný a tým zabránime vzniku degenerovaných záplat (t.j. záplat s viacnásobnými vrcholmi) a tým sa vyhneme riešeniu špeciálnych prípadov napr. pri výpočte normál.

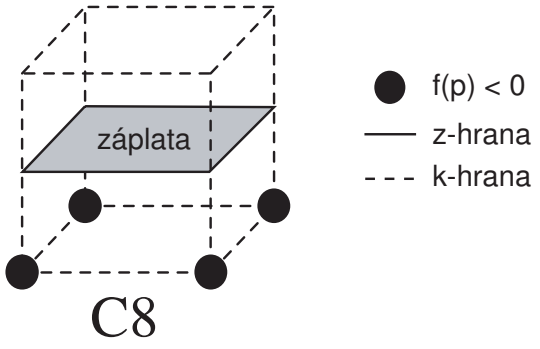
Algoritmus 3: Marching Cubes

```
vstup : C_0
append(active_cubes, C_0);
while active_cubes not empty do
    active_cube = top(active_cubes);
    if active_cube.evaluate is set then
        if active_cube.subdivision_level = 0 then
            foreach traversal face in active_cube do
                if neighbour_cube doesn't exist then
                    create neighbour_cube ;
                    append(active_cubes, neighbour_cube);
                    set neighbour_cube.evaluate;
                end
            end
        end
        foreach treversal edge e in active_cube do
            if subdivision needed then
                foreach cube cube containing edge e do
                    set cube.subdivide;
                end
            end
        end
    end
    if active_cube.subdivide is set then
        subdivide active_cube ;
        foreach child of active_cube do
            append(active_cubes, child);
            set child.evaluate;
        end
    end
end
```

Keďže plochu nebudeme v tejto fáze triangulovať, informácie o prieniku plochy s k-hranami postačujú. Takisto získame dostatok informácií na odhad normál (odsek 3.1.3).



Obrázok 21: Záplaty v základných konfiguráciach.



Obrázok 22: Podrobnejší popis konfugurácie C8.

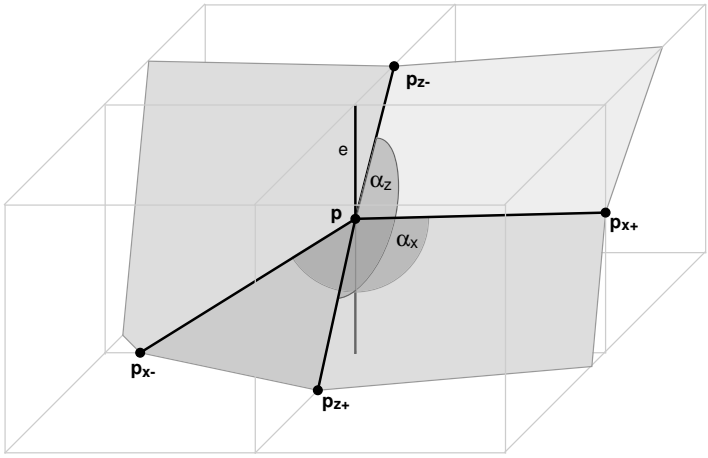
Hrany záplaty budeme označovať z-hrany (pozri obr. 22).

3.1.2 Prerozdelenie

Miesta plochy s väčšou krivosťou chceme pokryť hustejšie, aby bola aproximácia presnejšia. Na odhad krivosti použijeme uhly medzi susednými záplatami.

Pre každý priesečník k-hrany a plochy vypočítame vnútorné uhly príslušnej k-hrany a z-hrany. Ak sú tieto uhly príliš malé, kocky okolo príslušnej k-hrany prerozdelíme (budujeme octree).

Postup popíšeme pre prípad k-hrany e, ktorá je rovnobežná s osou y. Zvyšné prípady riešime analogicky. Priesečník hrany e a plochy si označíme ako **p**. Susedné priesečníky ležiace v rovine xy (resp. yz) si označíme ako p_{x+} a p_{x-} (resp. p_{z+} a p_{z-}). Vnútorné uhly označíme $\alpha_x = \angle\overrightarrow{pp_{x-}}, \overrightarrow{pp_{x+}}$ a $\alpha_z = \angle\overrightarrow{pp_{z-}}, \overrightarrow{pp_{z+}}$ (pozri obr. 23).



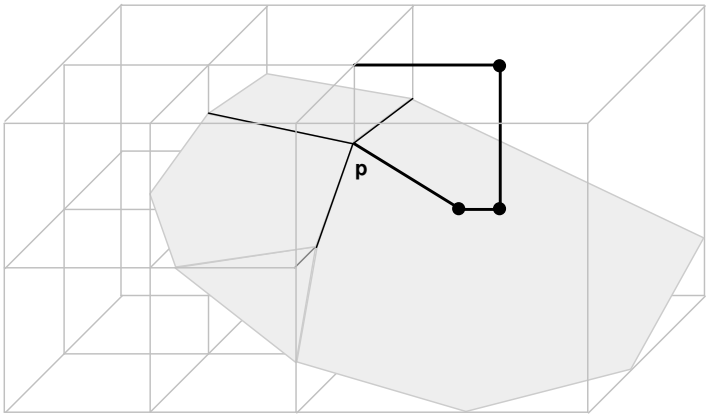
Obrázok 23: Uhly pre α_x, α_y odhad krivosti $Z(f)$ v bode **p**.

Označme $\alpha = \min(\alpha_x, \alpha_z)$. Uhol α_{min} definujeme ako prahovú hodnotu. Ak je $\alpha < \alpha_{min}$, kocky prerozdelíme. HÍbku octree ohraničíme hodnotou $depth_{max}$. Hodnotu α_{min} je vhodné voliť úmerne hĺbke prerozdelenia.

Ak na k-hrane e z úrovne prerozdelenia n, vznikne na úrovni $n+1$ priesečník, kocky obsahujúce hranu e prerozdelíme až do úrovne $n+1$. Podobne, ak stena Π nie je pretínajúca na úrovni n, a na úrovni $n+1$ v nej vznikne priesečník, susednú kocku obsahujúcu stenu Π prerozdelíme.

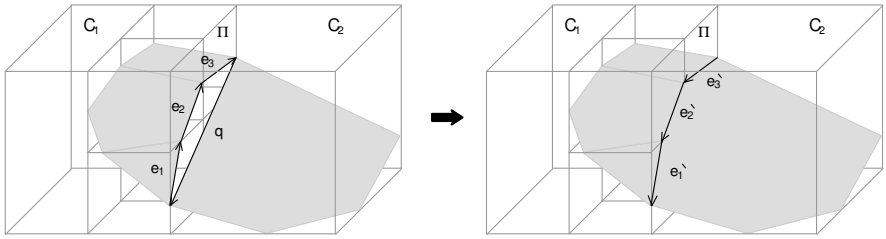
V prípade, že nie sú dve susedné kocky rovnako prerozdelené, nie je možné určiť uhol podľa uvedenej metódy. Preto ho neurčujeme a prerozdelenie

v tomto prípade nevykonáme. Chýbajúce údaje sú zvýraznené na obr. 24.



Obrázok 24: Chýbajúce údaje pri nerovnakom prerozdelení susedných kociek.

Označme Π spoločnú stenu dvoch kociek C_1 a C_2 na rovnakej úrovni prerozdelenia. Nech C_1 je prerozdelená a C_2 nie je vôbec prerozdelená (uvažujeme kocku, ktorá je listom v octree). Označme $e_1, e_2, \ldots, e_n$ orientované hrany záplat z kocky C_1 ležiace v stene Π. Označme q orientovanú hranu záplaty ρ z kocky C_2 ležiacu v stene Π. Hranu q v záplate ρ nahradíme hranami $e_n', e_{n-1}', \ldots, e_1'$, kde e_i' je hrana e_i s opačnou orientáciou, pre $i = 1, 2, \ldots, n$. Príklad pre jedno prerozdelenie a $n = 3$ je zobrazený na obr. 25.

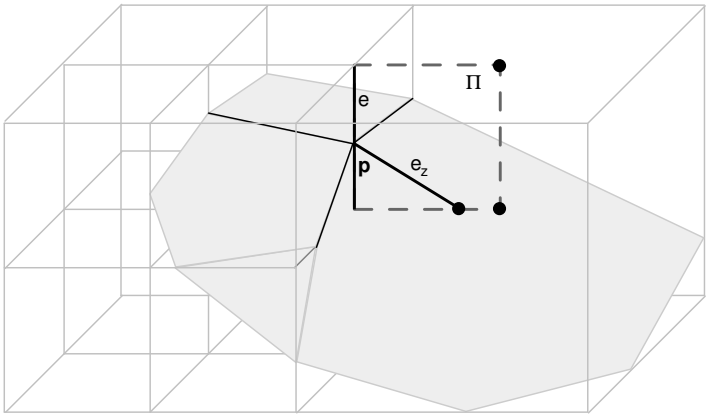


Obrázok 25: Spájanie záplat nerovnako prerozdelených susedných kociek.

3.1.3 Výpočet normály

Po skončení algoritmu MC je potrebné nájsť normály pre vrcholy záplaty. Každý vrchol leží na k-hrane. Z každého vrcholu vychádzajú buď práve 4 z-hrany alebo práve 3 z-hrany. Druhý prípad nastane, ak vrchol leží v stene medzi dvoma nerovnako prerozdelenými kockami.

Keďže sa snažíme minimalizovať počet volaní implicitnej funkcie f, vhodným spôsobom výpočtu normály je metóda z [NHS02].



Obrázok 26: Dopočítané údaje pri výpočte normály.

Ak z vyšetrovaného bodu **p** vychádzajú iba 3 z-hrany, uvedenú metódu nemôžeme použiť. Preto je treba chýbajúce údaje dopočítať (pozri obr. 26). Nech e je k-hrana, na ktorej leží **p**. Nech d je veľkosť najmenšej kocky obsahujúcej hranu e. Simulujeme vyhodnotenie steny Π, v ktorej by mala ležať chýbajúca z-hrana e_z. Veľkosť Π je rovná d. Takto vytvorenú fiktívnu z-hranu e_z použijeme iba na výpočet normály.

Vrcholy označíme ako pri výpočte uhlov pre odhad krivosti (odsek 3.1.2). Normála je vypočítaná ako vážený priemer normál z trojuholníkov $pp_{z+}p_{x+}$, $pp_{x+}p_{z-}$, $pp_{z-}p_{x-}$ a $pp_{x-}p_{z+}$ (pozri obr. 16).

3.2 Detekcia ostrých hrán

Predpokladáme, že hrany vznikajú na miestach s vysokým prerozdelením. Preto budeme vyšetrovať iba kocky na najnižšej úrovni $depth_{max}$ prerozdelenia (listy v octree), ktoré sú označené na prerozdelenie. V každej kocke poznáme záplatu.

Ostré hrany chceme detegovať ešte pred samotnou triangulációu. Tým zabránime vzniku deformovaných trojuholníkov pozdĺž hrán.

Nájdeme vrcholy vo vnútri záplaty, ležiace na ostrej hrane. Vrcholy v susedných záplatách pospájame a adaptívne upravíme. Okolo ostrých hrán vytvoríme trianguláciu, tak aby pri ďalšom kroku nášho algoritmu nebolo nutné uvažovať ostré hrany.

3.2.1 Detekcia

Hrany detegujeme podľa uhlov normál v jednotlivých vrcholoch záplaty. Ak je niektorý z uhlov príliš veľký, predpokladáme, že v záplate je ostrá hrana.

Vo vrcholoch vyšetrovanej záplaty Π nemôžeme určiť normály metódou, ktorú sme popísali v 3.1.3. Vypočítané normály by sa mohli výrazne líšiť od normál implicitnej plochy. Preto posunieme pozíciu vrcholov záplaty Π na $Z(f)$ pomocou bisekcie. Na určenie normál použijeme symetrickú diferenciu. Hodnotu δ volíme tak, aby bola výrazne menšia ako veľkosť kocky, ktorou je záplata Π tvorená.

Vrcholy záplaty Π označíme $\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_n}$. Normály vo vrcholoch označíme $\overrightarrow{n_0}, \overrightarrow{n_1}, \ldots, \overrightarrow{n_n}$. To, či záplatou Π prechádza ostrá hrana, určíme podľa uhla medzi normálami. Spočítame $\theta = \max_{i,j}(\angle \overrightarrow{n_i} \overrightarrow{n_j})$, kde $i, j = 0, 1, \ldots, n$ a $i \neq j$. Ak je $\theta \geq \theta_{edge}$ predpokladáme, že záplatou Π prechádza aspoň jedna ostrá hrana.

Nech $\overrightarrow{n_a}$ a $\overrightarrow{n_b}$ sú také normály $\overrightarrow{n_i}$, že $\angle \overrightarrow{n_a} \overrightarrow{n_b} = \theta$. Zistíme odklon normál $\overrightarrow{n_0}, \overrightarrow{n_1}, \ldots, \overrightarrow{n_n}$ od roviny určenej normálou $\overrightarrow{n^\star} = (\overrightarrow{n_a} \times \overrightarrow{n_b})/(\|\overrightarrow{n_a} \times \overrightarrow{n_b}\|)$. Spočítame $\varphi = \max_i(\angle \overrightarrow{n_i} \overrightarrow{n^\star})$, kde $i = 0, 1, \ldots, n$. Ak je $\varphi \geq \varphi_{corner}$ predpokladáme, že sa v záplate nachádza roh.

Hľadáme bod, ktorý leží na ostrej črte (ostrá hrana alebo roh). Vyšetrujeme záplatu Π, ktorá je na najnižšej úrovni prerozdelenia, jej vrcholy $\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_n}$ sú dostatočne blízko ostrej črty . Dotykové roviny vo vrcholoch $\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_n}$ určené príslušnými normálami $\overrightarrow{n_0}, \overrightarrow{n_1}, \ldots, \overrightarrow{n_n}$ dobre aproximujú $Z(f)$ v blízkosti záplaty Π. Preto hľadáme priesečník týchto dotykových rovín, teda riešime systém

$$\mathcal{N} \cdot \mathbf{p_\Pi} = \begin{pmatrix} \overrightarrow{n_0} \cdot \mathbf{p_0} \\ \overrightarrow{n_1} \cdot \mathbf{p_1} \\ \vdots \\ \overrightarrow{n_n} \cdot \mathbf{p_n} \end{pmatrix}, \text{ kde } \mathcal{N} = \begin{pmatrix} \overrightarrow{n_0} \\ \overrightarrow{n_1} \\ \vdots \\ \overrightarrow{n_n} \end{pmatrix}. \tag{3}$$

Ak sa v záplate nachádza ostrá hrana, mali by byť všetky normály z $\mathcal{N}$ lineárne závislé s $\overrightarrow{n_a}$ alebo $\overrightarrow{n_b}$ (teda riadová hodnosť matice $\mathcal{N}$ je 2) a riešením by bola celá priamka. Ale nami spočítané normály $\overrightarrow{n_1}$ sú len aproximáciami s istou nepresnosťou a ani samotná ostrá hrana nemusí byť lineárna, preto systém (3) nemusí mať vždy riešenie. Z rovnakých dôvodov nemusí mať systém riešenie ani v prípade, keď detegujeme roh. Aby sme sa vyhli riešeniu špeciálnych prípadov budeme systém 3 riešiť pomocou pseudoinverznej matice $\mathcal{N}^+$.

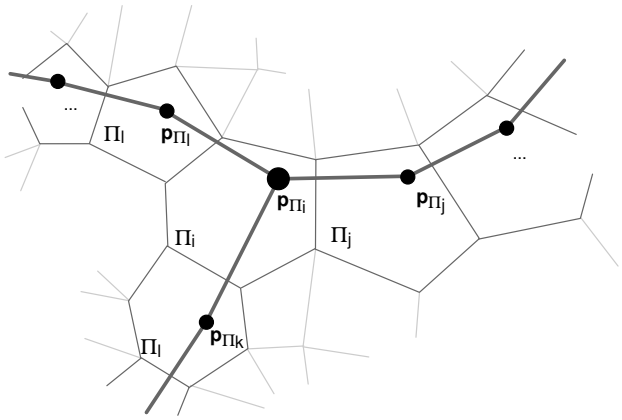
Riešením pomocou pseuodinverznej matice je bod, ktorého štvorce vzdia-

leností od dotykových rovín sú minimálne a leží čo najbližšie počiatku súradnicovej sústavy. Aby riešenie ležalo v záplate, posunieme stred súradnicovej sústavy do ťažiska záplaty ($\mathbf{0} = \Sigma_{i=0}^{n} \frac{\mathbf{p_i}}{n+1}$).

Maticu $\mathcal{N}^+$ vypočítame pomocou dekompozície podľa singulárnych hodnôt. Každá $m \times n$ matica $\mathcal{N}$ sa dá rozložiť podľa singulárnych hodnôt na súčin matíc $\mathcal{N} = \mathcal{U}\Sigma\mathcal{V}^T$, kde $\mathcal{U}$ je $m \times m$ ortogonálna matica, Σ je $m \times n$ diagonálna matica s nezápornými prvkami a $\mathcal{U}$ je $n \times n$ ortogonálna matica. Maticu $\mathcal{N}^+$ vypočítame ako $\mathcal{N}^+ = \mathcal{V}\Sigma^+\mathcal{U}^T$, kde Σ^T je transponovoná matica Σ s hodnotami na diagonále $s_{ii}^{,} = s_{ii}^{-1}$. Podrobnejšie o tom ako vypočítať dekompozíciu napr. v [PTVF92],[GL91].

Ak v záplate nedetegujeme roh ale iba hranu, matica $\mathcal{N}$ by mala mať iba dve singulárne hodnoty (teda riadková hodnosť matice $\mathcal{N}$ je 2). Ale ako sme už spomenuli, na reálnych dátach sa to udeje málokedy. Preto položíme najmenšiu singulárnu hodnotu matice $\mathcal{N}$ rovnú 0. Nájdený bod označíme $\mathbf{p_\Pi}$.

Každé dva takto získané vrcholy $\mathbf{p_{\Pi_i}}$ a $\mathbf{p_{\Pi_j}}$ pre susedné záplaty Π_i a Π_j spojíme (pozri obr. 27). Vrcholy, ktoré sú spojené s viac ako dvoma ďalšími vrcholmi, označíme ako rohové vrcholy.



Obrázok 27: Zdetegované ostré hrany.

3.2.2 Výpočet normál

Pre každý vrchol $\mathbf{p_\Pi}$ je nutné spočítať toľko normál, s koľkými vrcholmi je spojený. Hrany vychádzajúce z vrcholu $\mathbf{p_\Pi}$ označíme $e_0, e_1, \ldots, e_n, e_{n+1} = e_0$. Príslušné vrcholy označíme ako $\mathbf{p_i}$, kde $i = 0, 1, \ldots, n$. Vrchol $\mathbf{p_{n+1}}$ je totožný s vrcholom $\mathbf{p_0}$. Výsek $\angle e_i e_{i+1}$ je priestor medzi hranami $e_i e_{i+1}$. Vrcho-

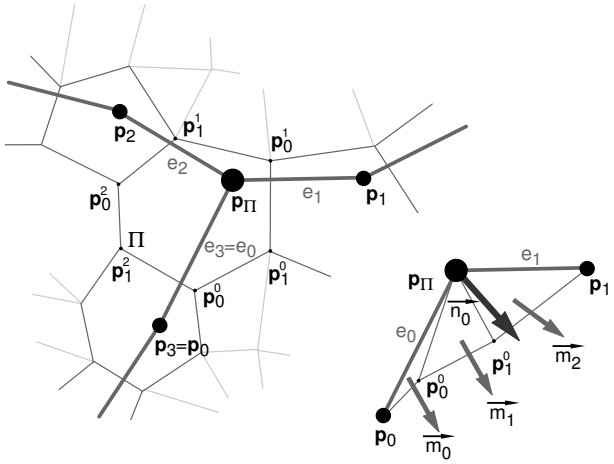
ly záplaty Π medzi hranami e_i a e_{i+1} postupne označíme ako $\mathbf{p_0^i}, \mathbf{p_1^i}, \ldots, \mathbf{p_{m_i}^i}$. Normálu $\overrightarrow{n_i}$ vypočítame ako vážený priemer normál trojuholníkov $\overline{\mathbf{p_\Pi p_i p_0^i}}$, $\overline{\mathbf{p_\Pi p_0^i p_1^i}}, \ldots, \overline{\mathbf{p_\Pi p_{m_i}^i p_{i+1}}}$.

Postup demonštrujeme na príklade z obrázku 28. Spočítame normálu $\overrightarrow{n_0}$ vo vrchole $\mathbf{p_\Pi}$ pre výsek $\angle e_0 e_1$. Normálu trojuholníka $\overline{p_\Pi p_0 p_0^0}$ (resp. $\overline{p_\Pi p_0^0 p_1^0}$ a $\overline{p_\Pi p_1^0 p_1}$) vypočítame ako

$$\overrightarrow{m_0} = \frac{\overrightarrow{\mathbf{p_\Pi p_0^0}} \times \overrightarrow{\mathbf{p_\Pi p_0}}}{\|\overrightarrow{\mathbf{p_\Pi p_0^0}} \times \overrightarrow{\mathbf{p_\Pi p_0}}\|}$$

$$(\text{resp. } \overrightarrow{m_1} = \frac{\overrightarrow{\mathbf{p_\Pi p_1^0}} \times \overrightarrow{\mathbf{p_\Pi p_0^0}}}{\|\overrightarrow{\mathbf{p_\Pi p_1^0}} \times \overrightarrow{\mathbf{p_\Pi p_0^0}}\|} \text{ a } \overrightarrow{m_2} = \frac{\overrightarrow{\mathbf{p_\Pi p_1}} \times \overrightarrow{\mathbf{p_\Pi p_1^0}}}{\|\overrightarrow{\mathbf{p_\Pi p_1}} \times \overrightarrow{\mathbf{p_\Pi p_1^0}}\|}).$$

Normálu $\overrightarrow{n_0}$ vypočítame ako priemer týchto normál $\overrightarrow{n_0} = \Sigma_{i=0}^{2} \frac{m_i}{3}$.



Obrázok 28: Výpočet normály na ostrej hrane.

Normálu $\overrightarrow{n_i}$, ktorú použijem pre vrchol konkrétneho trojuholníka, určíme podľa toho, do ktorého výseku $\angle e_i e_{i+1}$ trojuholník patrí.

3.2.3 Úprava hrán

Nájdené ostré hrany svojou veľkosťou nemusia vyhovovať zadaným podmienkam na trianguláciu. Preto ich musíme upraviť. Postup je analógiou metódy MT pre 2D.

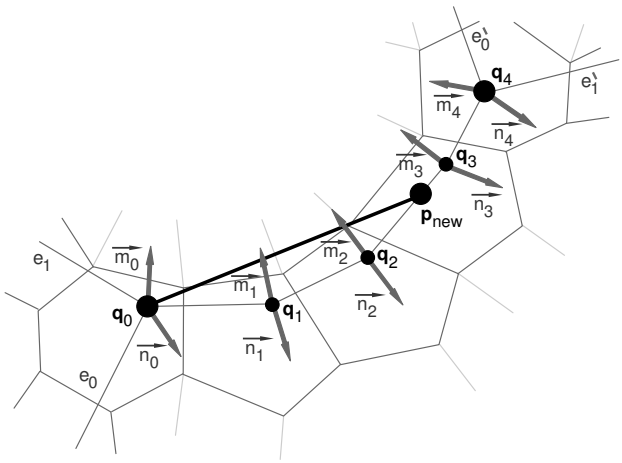
Všetky rohové vrcholy označíme ako aktívne. Ak nemáme žiadne rohové vrcholy, vyberieme ľubovoľný vrchol $\mathbf{p_\Pi}$ a označíme ho ako aktívny. Postupne prechádzame lomené čiary spájajúce aktívne vrcholy a susedné segmenty,

v ktorých sa normály nie príliš líšia (t.j. uhol je menší ako α_{lim}) spájame. Vrcholy, ktoré týmto spojením odstránim označíme ako spracované.

Aktuálne spracúvaný vrchol označíme $\mathbf{q_0}$. Ak vrchol $\mathbf{q_0}$ susedí iba so spracovanými vrcholmi, odstránime ho zo zoznamu aktívnych vrcholov. V opačnom prípade vykonáme úpravu hrán.

Vrcholy na lomenej čiare spájajúcej $\mathbf{q_0}$ s ďalším aktívnym vrcholom, označíme nasledovne. Nech $\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_n}$ je postupnosť vrcholov medzi $\mathbf{q_0}$ a $\mathbf{q_n}$, kde $\mathbf{q_1}$ je nespracovaný vrchol, $\mathbf{q_n}$ je aktívny vrchol a žiadny iný $\mathbf{q_i}$ nie je aktívny vrchol, pre $i = 1, 2, \ldots, n-1$.

Normály vo vrcholoch $\mathbf{q_i}$, pre $i = 1, 2, \ldots, n-1$ určené výsekmi $\angle \mathbf{q_{i-1}q_i}, \mathbf{q_iq_{i+1}}$ (resp. $\angle \mathbf{q_{i+1}q_i}, \mathbf{q_iq_{i-1}}$) označíme $\overrightarrow{n_i}$ (resp. $\overrightarrow{m_i}$). V aktívnych vrcholoch musíme normály určiť inak, pretože tieto vrcholy môžu byť aj rohové. Normála $\overrightarrow{n_0}$ (resp. $\overrightarrow{m_0}$) je normála vo vrchole $\mathbf{q_0}$ prislúchajúca výseku $\angle e_0, \mathbf{q_0q_1}$ (resp. $\angle \mathbf{q_0q_1}, e_1$). Normála $\overrightarrow{n_n}$ (resp. $\overrightarrow{m_n}$) je normála vo vrchole $\mathbf{q_n}$ prislúchajúca výseku $\angle \mathbf{q_{n-1}q_n}, e_1'$ (resp. $\angle e_0', \mathbf{q_{n-1}q_n}$) (pozri obr. 29).



Obrázok 29: Úprava ostrých hrán.

Uhly medzi normálami $\overrightarrow{n_0}$ a $\overrightarrow{n_i}$ (resp. $\overrightarrow{m_0}$ a $\overrightarrow{m_i}$) označíme ako α_i (resp. β_i).

Hľadáme prvý taký vrchol q_i, v ktorom sa niektorá z normál $\overrightarrow{n_i}$, resp. $\overrightarrow{m_i}$ príliš líši od $\overrightarrow{n_0}$, resp. $\overrightarrow{m_0}$, pričom dbáme na to, aby novovzniknutá hrana nebola príliš dlhá a ani príliš krátka. Teda hľadáme najmenšie $i \in \{1, 2, \ldots, n\}$ také, že pre vrchol $\mathbf{q_i}$ platí aspoň jedna z nasledujúcich podmienok:

(1) $\|q_0q_i\| \geq \epsilon_{min}$ a pre niektoré $j \in \{1, 2, \ldots, i\}$ platí, že $\alpha_j > \alpha_{lim}$ alebo $\beta_j > \alpha_{lim}$

(2) $\|q_0 q_i\| \geq \epsilon_{max}$,

kde α_{min} je hraničný uhol použitý pri triangulácii (odsek 3.3) a ϵ_{min} (resp. ϵ_{max}) je najmenšia (resp. najväčšia) prípustná dĺžka hrany trojuholníka. Ak taký vrchol neexistuje, použijeme vrchol q_n.

Potrebujeme vypočítať koeficient t pre lineárnu interpoláciu, pričom berieme do úvahy prahový uhol (α_{lim}), minimálnu (ϵ_{min}) a maximálnu (ϵ_{max}) dĺžku hrany. Spočítame koeficient $t_1 = (\|q_0 q_i\| - \epsilon_{min})/(\|q_0 q_{i-1}\| - \|q_0 q_i\|)$ pre výpočet podľa minimálnej dĺžky hrany a koeficient $t_2 = (\|q_0 q_i\| - \epsilon_{max})/(\|q_0 q_{i-1}\| - \|q_0 q_i\|)$ pre výpočet podľa maximálnej dĺžky hrany. Ak $\alpha_i > \beta_i$, použijeme $\gamma_i = \alpha_i$ a $\gamma_{i-1} = \alpha_{i-1}$, inak použijeme $\gamma_i = \beta_i$ a $\gamma_{i-1} = \beta_{i-1}$. Určíme koeficient $t_3 = (\gamma_i - \alpha_{lim})/(\gamma_{i-1} - \gamma_i)$ pre výpočet podľa prahového uhla. Koeficient $t = \max(t_2, \min(t_3, t_1))$ vyberieme ohraničením t_3 v intervale $\langle t_2, t_1 \rangle$ a následne ohraničíme $t = \max(0, \min(t, 1))$ intervalom $\langle 0, 1 \rangle$.

Nový vrchol $\mathbf{p_{new}}$ vypočítame ako $\mathbf{p_{new}} = t\mathbf{q_{i-1}} + (1-t)\mathbf{q_i}$. Podobne určíme normály $\overrightarrow{n_{new}} = t\overrightarrow{n_{i-1}} + (1-t)\overrightarrow{n_i}$ a $\overrightarrow{m_{new}} = t\overrightarrow{m_{i-1}} + (1-t)\overrightarrow{m_i}$. Hranu $q_{i-1}q_i$ rozdelíme na $q_{i-1}p_{new}$ a $p_{new}q_i$. Vrcholy q_j, kde $j = 1, \ldots, i-1$, označíme ako spracované. Vrchol $\mathbf{p_{new}}$ označíme ako aktívny.

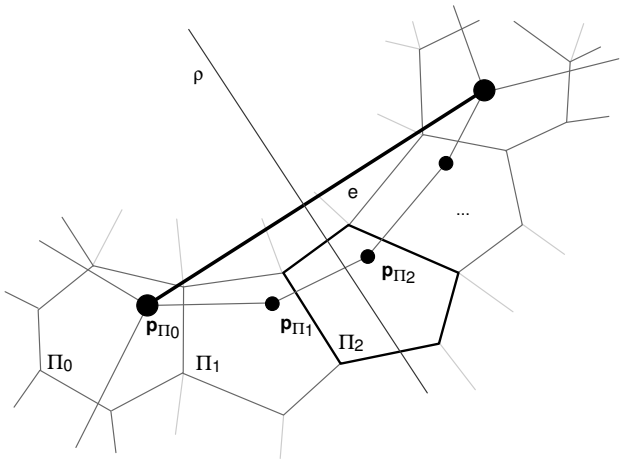
Úpravu ostrých hrán ukončíme ak sú všetky vrcholy označené ako spracované.

3.2.4 Triangulácia okolo ostrých hrán

Pri vlastnej triangulácii treba rozlišovať vrcholy ležiace na ostrej hrane a vyberať vhodnú normálu. Aby sme zbytočne nekomplikovali algoritmus MT, rozhodli sme sa najskôr vyriešiť tieto špeciálne prípady. Vrchol patriaci ostrej hrane nazývame ostrý vrchol. Okolo ostrých hrán vytvoríme trianguláciu na okraji ktorej sa nenachádzajú ostré vrcholy. Urobíme to modifikáciou algoritmu MT.

Algoritmus 4 modifikujeme nasledovne. Vrchol hľadáme iba pre aktívnu hranu, ktorá obsahuje ostrý vrchol. V prípade, že e je ostrá hrana zmeníme aj spôsob hľadania záplaty pre jej stred (odsek 3.3.4). Prechádzame záplaty, v ktorých ležali pôvodné vrcholy $\mathbf{p_{\Pi_i}}$ prislúchajúce hrane e (pozri obr. 30). Podmienka pre hľadanú záplatu zostáva nezmenená a teda výsledná záplata bude preťatá rovinou kolmou na e a prechádzajúcou jej stredom.

Ak nová hrana e_{new} obsahuje ostrý vrchol $\mathbf{p}$, normálu $\overrightarrow{n_i}$ v ňom určíme podľa výseku $\angle e_i e_{i+1}$, do ktorého hrana e_{new} patrí.



Obrázok 30: Záplata pre stred ostrej hrany.

3.3 Marching triangles

Po predchádzajúcich krokoch algoritmu máme celú plochu $Z(f)$ adaptívne aproximovanú záplatami. V prípade, že sa na $Z(f)$ nachádzali ostré hrany je okolo nich vytvorená triangulácia. Pre každý vrchol na okraji triangulácie poznáme prislúchajúcu záplatu a žiaden z týchto vrcholov neleží na ostrej hrane. Hrany na okraji triangulácie uložíme v zozname aktívnych hrán $\mathcal{L}$, ktorý je vstupom pre algoritmus MT (pozri alg. 4). Prípadnú trianguláciu okolo ostrých hrán zahrnieme do výslednej triangulácie.

Hľadanie bodu na implicitnej ploche redukujeme na hľadanie priesečníkov záplat a roviny (odsek 3.3.1). Nové trojuholníky vytvárame tak, aby ich vnútorne uhly boli $\approx 60°$ (odsek 3.3.2) a aby neprekrývali už vytvorenú trianguláciu (odsek 3.3.3). Postupne spracúvame hrany v $\mathcal{L}$. Práve spracúvanú hranu nazývame aktívna hrana.

3.3.1 Nový vrchol

Rovina $\rho : ax + by + cz + d = 0$ pretína úsečku $\mathbf{p_0 p_1}$ práve vtedy, keď $(ax_0 + by_0 + cz_0 + d)(ax_1 + by_1 + cz_1 + d) \leq 0$ (kde $\mathbf{p_0} = [x_0, y_0, z_0]$ a $\mathbf{p_1} = [x_1, y_1, z_1]$). Rovina pretína záplatu práve vtedy, keď pretína niektorú jej hranu. Rovinu, ktorá prechádza stredom aktívnej hrany a je na ňu kolmá, nazývame pretínajúca rovina.

Pretínajúca rovina je kolmá na aktívnu hranu, ktorá dobre aproximuje $Z(f)$, a preto je v blízkosti aktívnej hrany takmer kolmá aj na dotykové roviny k $Z(f)$ vo vrcholoch záplat. Priemet záplat do dotykovej roviny k $Z(f)$

Algoritmus 4: Marching Triangles

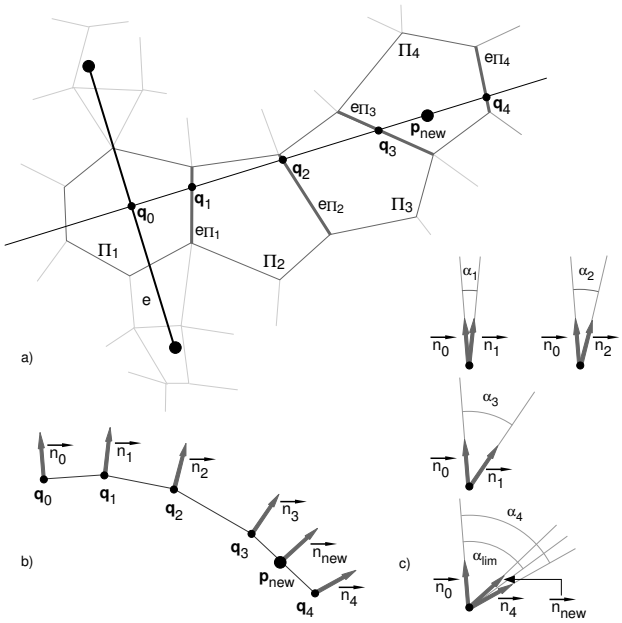
```
vstup: L
if empty(L) then
    find_first_edge(L);
end
while not empty(L) do
    active_edge = head(L);
    find patch for active_edge ;
    new_vertex = find_new_vertex();
    check_shape(active_edge, new_vertex);
    if FAILED then
        new_vertex = existing vertex;
    end
    check_overlap(active_edge, new_vertex);
    if FAILED then
        new_vertex = nearest vertex;
    end
    create new triangle;
    add new edges at the end of L ;
end
```

v ľubovoľnom z vrcholov záplaty je konvexný polygón. Preto pretínajúca rovina pretína záplatu v okolí aktívnej hrany maximálne v dvoch vrcholoch. Toto tvrdenie nemusí platiť pre záplaty obsahujúce ostré hrany, no tie boli v predchádzajúcom kroku už spracované a v tomto kroku ich už neuvažujeme.

Ak bola hrana záplaty prerozdelená (t.j. ležala v spoločnej stene dvoch nerovnako prerozdelených kociek), môže mať s pretínajúcou rovinou viac ako dva priesečníky. V takomto prípade treba špeciálne upraviť prechádzanie záplat. Špeciálny prípad môžme detegovať napr. spočítaním priesečníkov. Možným riešením je prerozdelenie záplaty. V našej implementácii túto situáciu detegujeme a ak nastane, aktívnu hranu presunieme na koniec L a pokračujeme ďalšou hranou.

Nový vrchol budeme hľadať v pretínajúcej rovine, aby novovzniknutý trojuholník mal čo najlepší tvar. Vzdialenosť hľadaného vrcholu od aktívnej hrany adaptívne prispôsobíme krivosti plochy.

Vychádzame zo záplaty prislúchajúcej k stredu aktívnej hrane. Postupne prehľadávame záplaty preťaté pretínajúcou rovinou. Hľadanie ukončíme, ak je uhol normály v novom vrchole a normály v strede aktívnej hrany väčší ako zadaná prahová hodnota α_{lim} (pozri alg. 5).



Obrázok 31: Hľadanie nového vrcholu a) priesečníky záplat s pretínajúcou rovinou; b) normály v priesečníkoch; c) uhly medzi normálami.

Aktívnu hranu označíme ako e. Stred hrany e označíme ako $\mathbf{q_0}$. Normálu vo vrchole $\mathbf{q_0}$ označíme $\overrightarrow{n_0}$ (vypočítame ju lineárnou kombináciou normál vo vrcholoch hrany e). Záplatu prislúchajúcu ku $\mathbf{q_0}$ označíme ako Π_1 (spôsob ako ju nájdeme je popísaný v 3.3.4). Pretínajúcu rovinu označíme ako ρ. Postupne prechádzame záplaty Π_i pre $i = 1, 2, \ldots$ preťaté rovinou ρ nasledovne. Nájdeme z-hranu e_{Π_i} záplaty Π_i takú, že e_{Π_i} je preťatá rovinou ρ a $e_{\Pi_i} \neq e_{\Pi_{i-1}}$. Priesečník na hrane e_{Π_i} označíme ako $\mathbf{q_i}$ (pozri obr. 31). Normálu v bode $\mathbf{q_i}$ označíme ako $\overrightarrow{n_i}$ a vypočítame ju lineárnou interpoláciou normál vo vrcholoch hrany e_{Π_i}. Uhol, ktorý zvierajú normály $\overrightarrow{n_0}$ a $\overrightarrow{n_i}$ označíme ako α_i.

Nový vrchol $\mathbf{p_{new}}$ a normálu $\overrightarrow{n_{new}}$ v ňom určíme podobne ako v odseku 3.2.3 na strane 33, ale budeme uvažovať iba uhly α_i a konštanty ϵ_{min} a ϵ_{max} určujú minimálnu, resp. maximálnu povolenú výšku trojuholníka.

Jednoznačnosť výberu orientovanej z-hrany e_{Π_i} môžme zabezpečiť nasledovne. Nech $e_{\Pi_i} = \mathbf{p_0 p_1}$. Nech $\rho : ax + by + cz + d = 0$. Hranu e_{Π_i} vyberieme tak, aby $ax_0 + by_0 + cz_0 + d \leq 0$, kde $\mathbf{p_0} = [x_0, y_0, z_0]$ a $ax_1 + by_1 + cz_1 + d > 0$, kde $\mathbf{p_1} = [x_1, y_1, z_1]$.

Algoritmus 5: Nájdenie nového vrcholu v blízkosti $Z(f)$.

vstup : $\mathbf{q_0}$, $\overrightarrow{n_0}$, Π, ρ

výstup: $\mathbf{p_{new}}$, $\overrightarrow{n}$, Π_{actual}

$\mathbf{q} = \mathbf{q_0}$;

$\overrightarrow{n} = \overrightarrow{n_0}$;

min_length_reached = FALSE ;

$\Pi_{actual} = \Pi$;

while $\mathbf{p_{new}}$ *not found* **do**

  $\mathbf{q_{old}} = \mathbf{q}$;

  $\overrightarrow{n_{old}} = \overrightarrow{n}$;

  $\mathbf{q}, \overrightarrow{n}$ = `find_intersection`(Π_{actual}, ρ);

  if *not* min_length_reached $\wedge$ $\|\mathbf{q_0 q_{old}}\| \geq \epsilon_{min}$ **then**

    min_length_reached = TRUE ;

  end

  if min_length_reached $\wedge$ $\angle\overrightarrow{n_0}\overrightarrow{n} \geq \alpha_{lim}$ **then**

    $t_1 = (\|\mathbf{q_0 q}\| - \epsilon_{min})/(\|\mathbf{q_0 q_{old}}\| - \|\mathbf{q_0 q}\|)$;

    $t_2 = (\|\mathbf{q_0 q}\| - \epsilon_{max})/(\|\mathbf{q_0 q_{old}}\| - \|\mathbf{q_0 q}\|)$;

    $t_3 = (\angle\overrightarrow{n_0}\overrightarrow{n} - \alpha_{lim})/(\angle\overrightarrow{n_0}\overrightarrow{n_{old}} - \angle\overrightarrow{n_0}\overrightarrow{n})$;

    t = `max`(t_2, `min`(t_3, t_1));

    t = `max`(*0*, `min`(t, *1*));

    $\mathbf{p_{new}} = t\ \mathbf{q} + (1 - t)\ \mathbf{q_{old}}$;

    $\overrightarrow{n} = t\ \overrightarrow{n} + (1 - t)\ \overrightarrow{n_{old}}$;

  end

  else if $\|\mathbf{q_0 q}\| \geq \epsilon_{max}$ **then**

    $t = (\|\mathbf{q_0 q}\| - \epsilon_{max})/(\|\mathbf{q_0 q_{old}}\| - \|\mathbf{q_0 q}\|)$;

    $\mathbf{p_{new}} = t\ \mathbf{q} + (1 - t)\ \mathbf{q_{old}}$;

    $\overrightarrow{n} = t\ \overrightarrow{n} + (1 - t)\ \overrightarrow{n_{old}}$;

  end

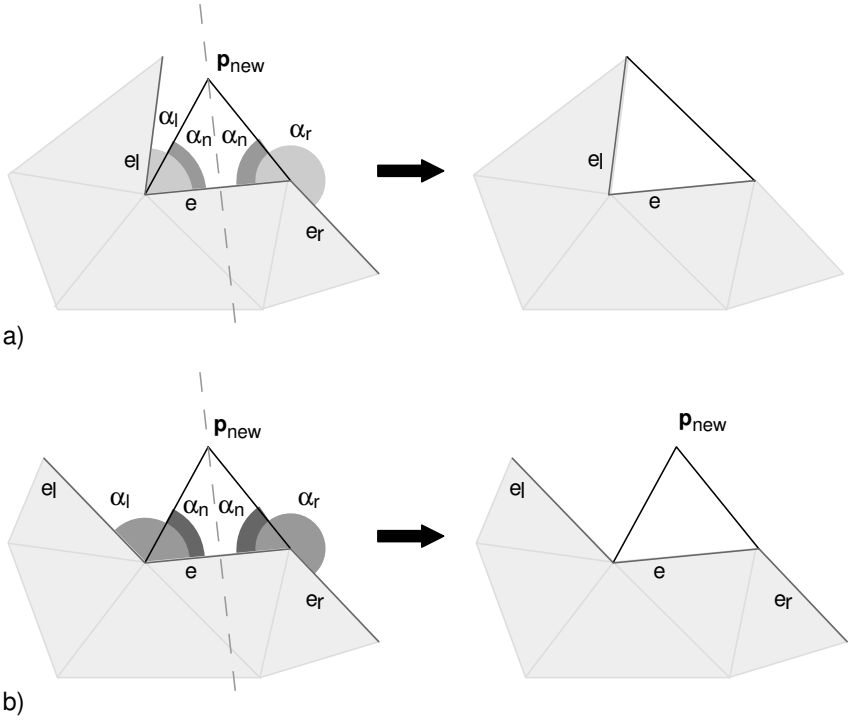
  Π_{actual} = `find_neighbour_patch`(Π_{actual}, ρ);

end

3.3.2 Riadenie tvaru nového trojuholníka

Keďže chceme, aby tvar trojuholníkov bol čo najlepší, musíme zaistiť, aby medzi susednými hranami z $\mathcal{L}$ nevznikali príliš malé uhly, čo by malo za následok vytváranie priúzkych trojuholníkov.

Aktívnu hranu označíme $e = \mathbf{p_0 p_1}$. Uhol $\angle \mathbf{p_{new} p_0 p_1}$ označíme α_n. Susednú aktívnu hranu naľavo (resp. napravo) od e označíme e_l (resp. e_r). Uhol medzi hranou e a e_l (resp. e_r) označíme α_l (resp. α_r). Položíme $\alpha = \min(\alpha_l, \alpha_r)$. Zisťujeme, či je uhol α dostatočne veľký, t.j. porovnávame ho s $k_{lim}\alpha_n$, kde k_{lim} určuje, ako veľký musí byť uhol α. Ak je $\alpha < k_{lim}\alpha_n$, vytvoríme trojuholník spojením aktívnej hrany s príslušnou susednou hranou (ak $\alpha_l < \alpha_r$ je to e_l, inak e_r) (pozri obr. 32 a). Ak je $\alpha \geq k_{lim}\alpha_n$ použijeme trojuholník $\overline{\mathbf{p_0 p_{new} p_1}}$ (pozri obr. 32 b).



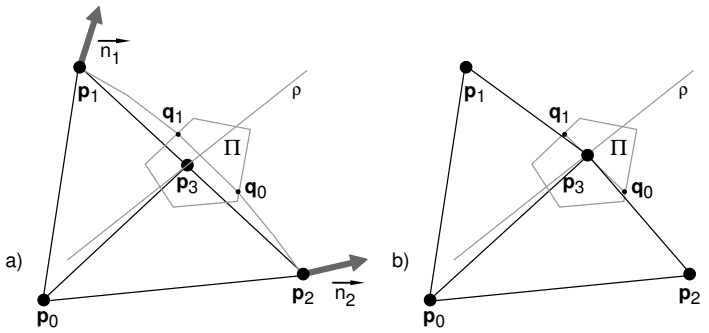
Obrázok 32: Porovnanie uhlov a) $\min(\alpha_l, \alpha_r) < k_{lim}\alpha_n$; b) $\min(\alpha_l, \alpha_r) \geq k_{lim}\alpha_n$.

V našej implementácii volíme hodnotu konštanty k_{lim} =1,7. Tým zabezpečíme, aby susedný trojuholník mal minimálny uhol $\geq$0,7α.

V prípade, že spojíme susedné aktívne hrany, musíme overiť, či nový trojuholník dobre aproximuje implicitnú plochu. Nech nový trojuholník vznik-

ne spojením hrán e a e_r. Vrchol na hrane e_r rôzny od $\mathbf{p_0}$ označíme $\mathbf{p_2}$. Normálu v bode $\mathbf{p_1}$ (resp. $\mathbf{p_2}$) označíme $\overrightarrow{n_1}$ (resp. $\overrightarrow{n_2}$). Ak je uhol medzi normálami $\overrightarrow{n_1}$ a $\overrightarrow{n_2}$ väčší ako α_{lim}, hranu $\mathbf{p_1p_2}$ rozdelíme. Vytvoríme nový vrchol $\mathbf{p_3} = \frac{\mathbf{p_1}+\mathbf{p_2}}{2}$. Vytvoríme dva trojuholníky $\overline{\mathbf{p_0p_2p_3}}$ a $\overline{\mathbf{p_0p_3p_1}}$ (pozri obr. 33 a).

Vrchol $\mathbf{p_3}$ posunieme tak aby ležal v záplate. Nech Π je záplata prislúchajúca k stredu hrany $\mathbf{p_2p_1}$ (odsek 3.3.4). Nech $\mathbf{q_0}$ a $\mathbf{q_1}$ sú priesečníky na záplate Π, ktoré vznikli pri jej hľadaní. Nech ρ je rovina kolmá na hranu $\mathbf{p_2p_1}$ a prechádza vrcholom $\mathbf{p_3}$. Vzdialenosť bodu $\mathbf{q_0}$ (resp. $\mathbf{q_1}$) od roviny ρ označíme d_0 (resp. d_1). Novú pozíciu vrcholu určíme nasledovne $\mathbf{p_3} = t\mathbf{q_0} + (1-t)\mathbf{q_1}$, kde $t = d_1/(d_0+d_1)$. Normálu vo vrchole $\mathbf{p_3}$ vypočítame ako $\overrightarrow{n_3} = t\overrightarrow{n_{q_0}} + (1-t)\overrightarrow{q_1}$, kde $\overrightarrow{n_{q_0}}$ (resp. $\overrightarrow{n_{q_0}}$) je normála v bode $\mathbf{q_0}$ (resp. $\mathbf{q_1}$) (pozri obr. 33 b).



Obrázok 33: Rozdelenie trojuholníka pri veľkej divergencii normál vo vrcholoch novej hrany a) pred rozdelením; b) po rozdelení.

Prípad, keď trojuholník vnikne spojením hrán e a e_l riešime analogicky.

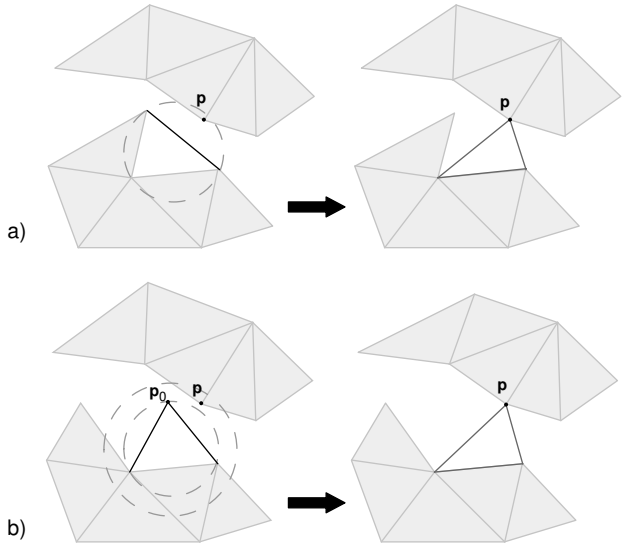
3.3.3 Overenie prekrytia trojuholníkov

Nový trojuholník môžeme pridať to triangulácie iba vtedy, ak neprekrýva nejaký už existujúci trojuholník. Na detekciu prekrytia použijeme analógiu Delaunayovej podmienky.

Nový trojuholník označíme $T_{new} = \overline{\mathbf{p_0p_1p_2}}$, kde $\mathbf{p_1p_0}$ je aktívna hrany. Nech $\mathbf{c}$ je stred a r je polomer minimálnej guľovej plochy opísanej trojuholníku T_{new}. Hľadáme hrany existujúcej triangulácie, ktoré pretínajú guľovú plochu $(\mathbf{c}, r)$, pričom neberieme do úvahy aktívnu hranu a hrany susediace s aktívnou hranou (pozri obr. 34 a). V prípade, že trojuholník nevznikol spojením susedných aktívnych hrán, použijeme $r = k_r\|\mathbf{cp_0}\|$ (pozri obr. 34 b).

Konštantu k_r používame preto, aby sme zaručili, že nový trojuholník je dostatočne ďaleko od existujúcich hrán a teda zostane dostatok priestoru na vytvorenie nového trojuholníka s uhlami $\approx 60°$. V našej implementácii sme použili k_r =1,5.

Nech vrchol $\mathbf{p}$ je taký, že $\|\mathbf{cp}\| = \min \|\mathbf{cq}\|$, kde $\mathbf{q}$ sú vrcholy patriace nájdeným hranám.



Obrázok 34: Test vzdialenosti a) pre trojuholník, ktorý vznikol spojením susedných aktívnych hrán; b) pre trojuholník, ktorý vznikol použitím nového vrcholu.

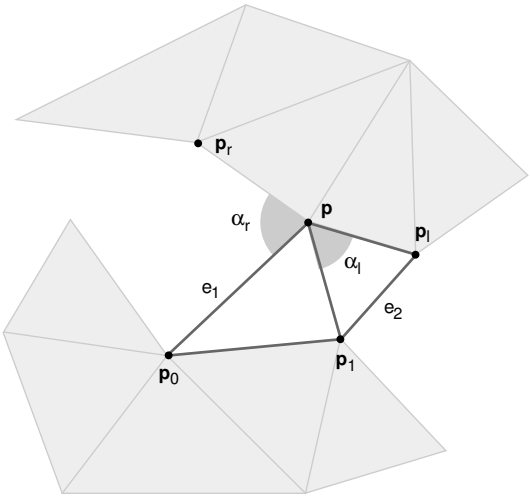
V prípade a) trojuholník T_{new} pridáme do výslednej triangulácie. V prípade b) zistíme, či je splnená Dealunayova podmienka pre trojuholník $\overline{\mathbf{p_0 p_1 p}}$. Ak nie je, žiadny trojuholník do triangulácie nepridáme, aktívnu hranu necháme nespracovanú a pridáme ju na koniec zoznamu $\mathcal{L}$. Algoritmus sa môže zacykliť v prípade, že nie je možné vytvoriť trojuholník spĺňajúci Delaunayovu podmienku. Preto treba detegovať zacyklenie a následne trianguláciu riešiť špeciálne. Zmenou parametrov (napr. zmenšením hodnoty maximálnej výšky trojuholníka) alebo počiatočnej konfigurácie sa môžme (nie nutne musíme) vyhnúť zacykleniu.

My sme sa prikloníme k druhej možnosti a zacyklenie iba detegujeme, no neriešime následnú trianguláciu. Označíme si vždy iba aktívnu hranu, ktorá bola spracúvaná hneď po vytvorení nevého trojuholníka. Ak hranu, ktorá bola takto označená ako posledná,spracúvame druhý krát (t.j. po prejdení

celého $\mathcal{L}$ nebol vytvorený ani jeden trojuholník), nastalo zacyklenie.

Ak je splnená Delaunayova podmienka, musíme zistiť, či je možné vo vrchole $\mathbf{p}$ rozdeliť hranicu triangulácie. To robíme preto, lebo vrcholom $\mathbf{p}$ by prechádzali 4 hrany z $\mathcal{L}$ a to by spôsobovalo nejednoznačnosti pri výbere susedných hrán. Nech $e_l, q_r \in \mathcal{L}$ sú orientované hrany prechádzajúce vrcholom $\mathbf{p}$ také, že $e_l = \mathbf{pp_l}$ a $e_r = \mathbf{p_r p}$. Spočítame uhly $\alpha_r = |\angle\overrightarrow{pp_r}\overrightarrow{pp_0}|$ a $\alpha_l = |\angle\overrightarrow{pp_l}\overrightarrow{pp_1}|$. Ak $\min(\alpha_r, \alpha_l) > \alpha_{divide}$ nie je možné hranicu triangulácie rozdeliť, preto žiadny trojuholník nevytvoríme a aktívnu hranu pridáme na koniec $\mathcal{L}$.

Nech $\alpha_l \leq \alpha_r$ a $\alpha_l \leq \alpha_{divide}$. Ak trojuholník $\overline{\mathbf{pp_1p_l}}$ spĺňa Delaunayovu podmienku, pridáme trojuholníky $\overline{\mathbf{p_0p_1p}}$ a $\overline{\mathbf{pp_1p_l}}$ do triangulácie. Hrany $\mathbf{p_1p_0}$ a $\mathbf{pp_l}$ odstránime z $\mathcal{L}$. Hrany $\mathbf{pp_0}$ a $\mathbf{p_1p_l}$ pridáme do $\mathcal{L}$ (pozri obr. 35) . Ak trojuholník $\overline{\mathbf{pp_1p_l}}$ nespĺňa Delaunayovu podmienku, žiadny trojuholník nevytvoríme a aktívnu hranu $\mathbf{p_1p_0}$ pridáme na koniec $\mathcal{L}$. Prípad, keď $\alpha_r > \alpha_l$ riešime analogicky.



Obrázok 35: Rozdelenie hranice triangulácie.

Pri pridávaní hrany $e_1 = \mathbf{pp_0}$ (resp. $e_2 = \mathbf{p_1p_l}$) do $\mathcal{L}$ je nutné overiť, či hrany susedné k e_1 (resp. e_2) nie sú totožné. Ak sú totožné, tak hranu e_1 (resp. e_2) do zoznamu $\mathcal{L}$ nepridáme a susedná hrany k e_1 (resp. e_2) zo zoznamu $\mathcal{L}$ odstránime.

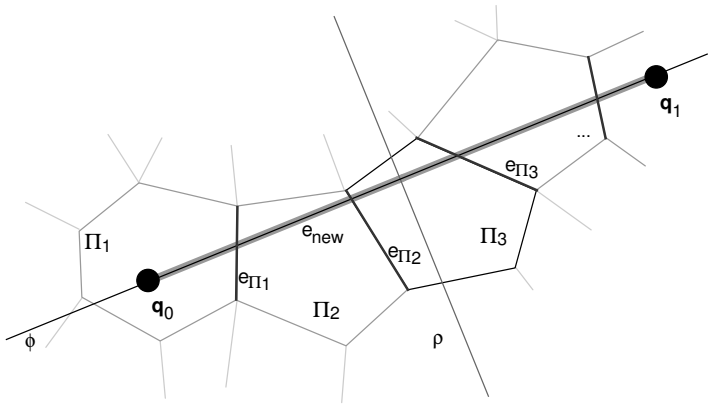
Vzhľadom na spôsob, akým vznikajú trojuholníky pri metóde MT stačí, ak testujeme iba vrcholy patriace aktívnym hranám (teda vrcholy na hranici už existujúcej triangulácie).

3.3.4 Nájdenie záplaty pre novú hranu

Aby sme mohli nájsť vrchol pre aktívnu hranu, potrebujeme nájsť počiatočnú záplatu. Táto záplata musí byť kolmá na pretínajúcu záplatu. Poznáme záplaty, v ktorých ležia vrcholy aktívnej hrany a poznáme normály v týchto vrcholoch.

Použijeme podobný princíp ako pri hľadaní vrcholu. Postupne budeme prechádzať záplaty preťaté rovinou, pokým nenarazíme na rovinu preťatú pretínajúcou rovinou.

Novú hranu označíme $e_{new} = q_0q_1$. Pretínajúcu rovinu hrany e_{new} označíme ρ . Teda ρ je kolmá na e_{new} a prechádza jej stredom. Záplatu príslúchajúcu vrcholu $\mathbf{q_0}$ označíme Π_0. Rovinu určenú normálovým vektorom $\overrightarrow{n} = (\overrightarrow{n_0} + \overrightarrow{n_1}) \times (\overrightarrow{\mathbf{q_0}\mathbf{q_1}})$ a vrcholom $\mathbf{q_0}$, označíme ϕ, kde $\overrightarrow{n_0}$ a $\overrightarrow{n_1}$ sú normály vo vrcholoch $\mathbf{q_0}$ a $\mathbf{q_1}$ (pozri obr. 36).



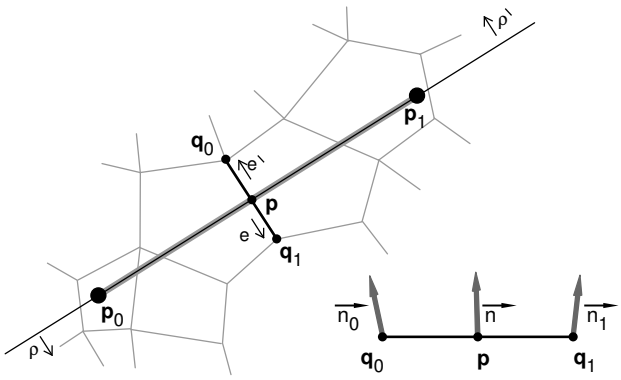
Obrázok 36: Nájdenie záplaty pre novú hranu.

Postupne prechádzame záplaty Π_i pre $i = 0, 1, \ldots$ preťaté rovinou ϕ nasledovne. Nájdeme z-hranu e_{Π_i} záplaty Π_i takú, že e_{Π_i} je preťatá rovinou ρ a $e_{\Pi_i} \neq e_{\Pi_{i-1}}$. Hľadanie ukončíme, ak je záplata Π_i preťatá rovinou ρ.

3.3.5 Prvá hrana

V prípade, že implicitná plocha $Z(f)$ neobsahuje ostré hrany, zoznam aktívnych hrán $\mathcal{L}$ je prázdny a teda nemáme žiadny vstup pre algoritmus MT. Preto potrebujeme nájsť hranu na $Z(f)$, z ktorej by sme mohli začať trianguláciu.

Budeme simulovať hľadanie nového vrcholu (odsek 3.3.1) pre fiktívnu hranu. Následne použijeme tú istú hranu ale s opačnou orientáciou na nájdenie



Obrázok 37: Nájdenie prvej hrany v prípade, že nemáme ostré hrany.

druhého vrcholu. Tieto dva vrcholy nám určia dvojicu orientovaných hrán, ktoré môžme použiť pre algoritmus MT.

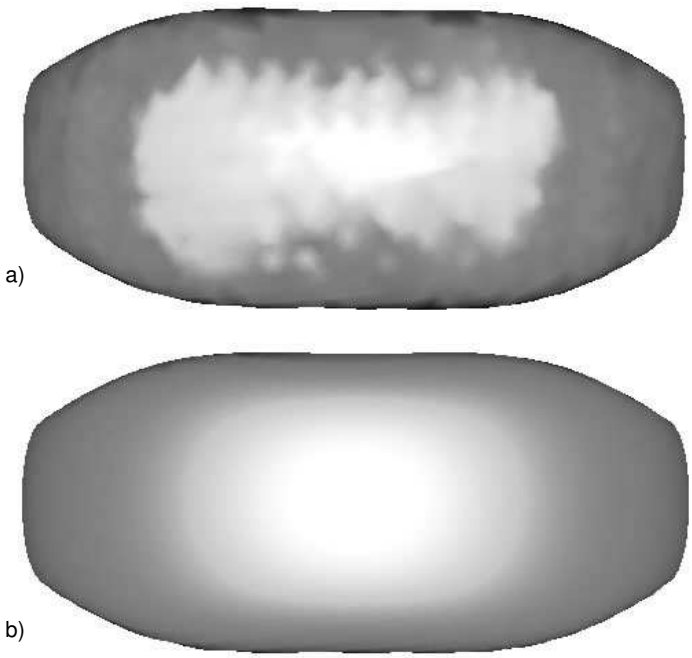
Ako fiktívnu hranu použijeme ľubovoľnú orientovanú z-hranu $e = \mathbf{q_0 q_1}$. Záplatu, ktorej patrí hrana e označíme Π_e. Hranu e s opačnou orientáciou označíme e' (teda $e' = \mathbf{q_1 q_0}$) a príslušnú záplatu $\Pi_{e'}$. Stred hrany e označíme $\mathbf{p}$. Normálu $\overrightarrow{n}$ v bode $\mathbf{p}$ vypočítame ako $\overrightarrow{n} = \frac{(\overrightarrow{n_0}+\overrightarrow{n_1})}{(\|\overrightarrow{n_0}+\overrightarrow{n_1}\|)}$, kde $\overrightarrow{n_0}$ (resp. $\overrightarrow{n_1}$) je normála vo vrchole q_0 (resp. q_1). Rovinu s normálovým vektorom $\overrightarrow{e}$ (resp. $\overrightarrow{e'}$) a bodom $\mathbf{p}$ označíme ρ (resp. ρ'). Na nájdenie vrcholu $\mathbf{p_1}$ použijeme záplatu Π, pretínajúcu rovinu ρ a počiatočnú normálu $\overrightarrow{n}$. Na nájdenie vrcholu $\mathbf{p_2}$ použijeme záplatu Π, pretínajúcu rovinu ρ' a počiatočnú normálu $\overrightarrow{n}$. Vytvoríme dve orientované hrany $e_1 = \mathbf{p_1 p_2}$ a $e_2 = \mathbf{p_2 p_1}$ (pozri obr. 37). Hrany e_1 a e_2 pridáme do zoznamu aktívnych hrán $\mathcal{L}$.

3.4 Projekcia triangulácie na $Z(f)$

Takto získaná triangulácia je adaptívna, no nemusí byť príliš dobrou aproximáciou (pozri obr. 38 a). To je spôsobené častým využitím lineárnej interpolácie, pričom f nemusí byť (a málokedy je) lineárna. Preto vrcholy výslednej triangulácie premietneme na $Z(f)$ (pozri obr. 38 b).

Smer premietania je určený normálou $\overrightarrow{n_i}$ v danom vrchole $\mathbf{p_i}$ a znamienkom $f(\mathbf{p_i})$. Prvé dva vrcholy $\mathbf{q_{j-1} q_j}$ pre ktoré platí $f(\mathbf{q_{j-1}})f(\mathbf{q_j}) <= 0$ použijeme ako vstup pre algoritmus bisekcie, kde $\mathbf{q_j} = \mathbf{q_{j-1}} - \delta \text{sign}(f(\mathbf{q_{j-1}}))\overrightarrow{n_i}$, pre $j = 1, 2, ...$ a $\mathbf{q_0} = \mathbf{p_i}$. Hodnotu δ volíme úmerne hustote mriežky pri metóde MC.

Vrchol $\mathbf{p_i}$ posunieme na pozíciu získanú bisekciou. Normály vo vrcholoch premietnutej triangulácie určíme asymetrickou diferenciou.



Obrázok 38: Projekcia na $Z(f)$.

Vrcholy ležiace na ostrých hranách neupravujeme, tie sme riešili špeciálne a ich pozícia je dostatočne presná.

4 Výsledky

4.1 Implementácia

Uvedený algoritmus (bez detekcie ostrých hrán) sme implementovali v prostredí Borland C++ Builder 6. Kocky sú reprezentované ako štruktúry s odkazmi na vrcholy, hrany a prípadné dcérske kocky pri prerozdelení. Záplata je reprezentovaná ako cyklický zoznam z-hrán. Aktívne hrany sú reprezentované ako okrídlené hrany. Na urýchlenie vyhľadávaní sme použili hešovacie tabuľky. Výstup nášho programu je popis triangulácie vo formáte VRML.

4.2 Výsledné triangulácie

Kvalitu tvaru trojuholníkov budeme merať pomerom najkratšej a najdlhšej hrany trojuholníka a pomer najmenšieho a najväčšieho vnútorného uhla trojuholníka. Žiaduce je, aby vnútorné uhly boli $\approx 60°$, preto chceme aby tieto pomery boli čo najbližšie 1.

Merania sme vykonali a počítači s procesorom Intel Celeron M 1,5 GHz a s veľkosťou pamäte 192 MB.

Uvedieme triangulácie plôch definované nasledujúcimi funkciami. Guľa:

$$f_1(x,y,z) = x^2 + y^2 + z^2 - 9, \tag{4}$$

objekt Genus3:

$$\begin{aligned} f_2(x,y,z) \quad = \quad & 256 * z^2 - \\ & (1 - x^2/36 - y^2/12, 25)[(x-3.9)^2 + y^2 - 1.44] \\ & [(x+3.9)^2 + y^2 - 1.44](x^2 + y^2 - 1.44), \end{aligned} \tag{5}$$

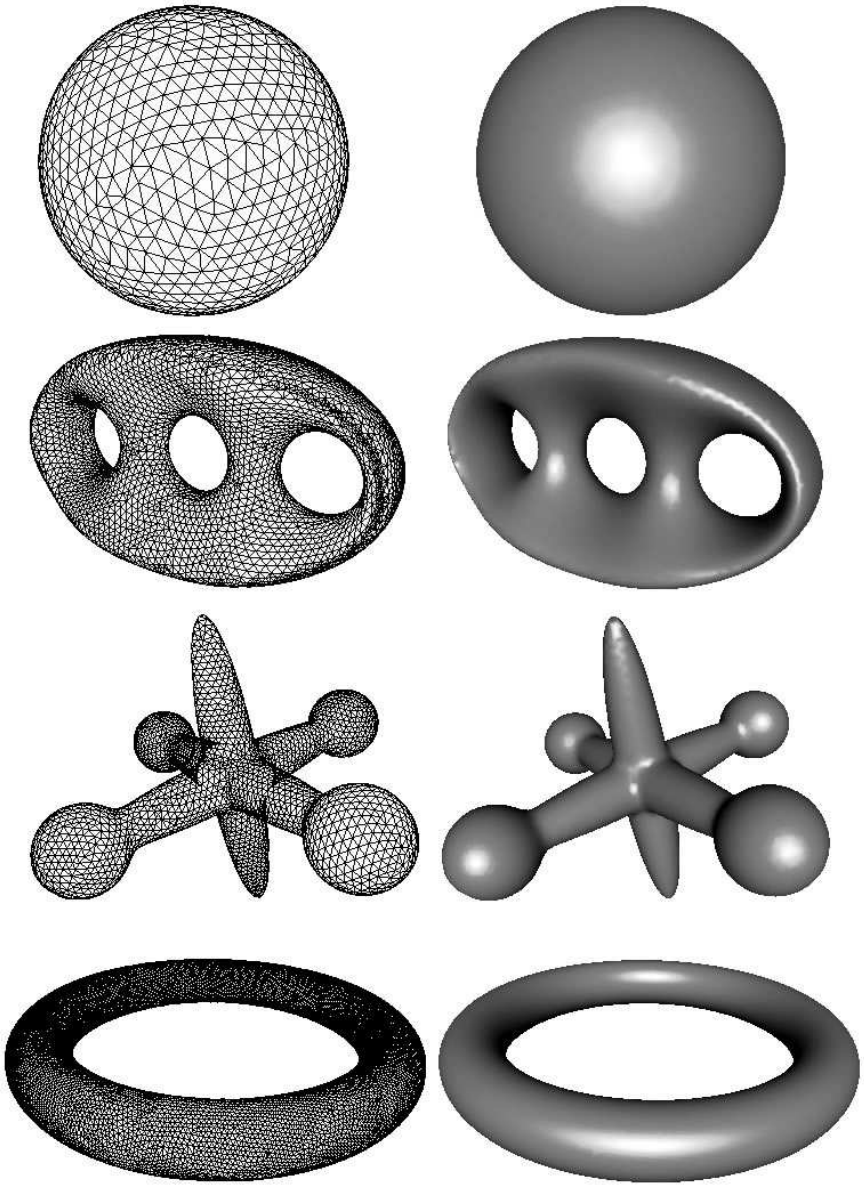
objekt Jack:

$$\begin{aligned} f_4(x,y,z) \quad = \quad & (x^2/9 + 4y^2 + 4z^2)^{-4} + (y^2/9 + 4x^2 + 4z^2)^{-4} + \\ & (z^2/9 + 4y^2 + 4x^2)^{-4} + \\ & [(4x/3 - 4)^2 + 16y^2/9 + 16z^2/9]^{-4} + \\ & [(4x/3 + 4)^2 + 16y^2/9 + 16z^2/9]^{-4} + \\ & [(4y/3 - 4)^2 + 16x^2/9 + 16z^2/9]^{-4} + \\ & [(4y/3 + 4)^2 + 16x^2/9 + 16z^2/9]^{-4} - 1 \end{aligned} \tag{6}$$

a torus:

$$f_3(x,y,z) = (x^2 + y^2 + z^2 + 3^2 - 2^2)^2 - 4 * 2^2(x^2 + y^2). \tag{7}$$

Výsledné triangulácie sú zobrazené na obrázku 39 a namerané hodnoty kvality triangulácie a čas triangulácie sa nachádzajú v tabuľke 1.



Obrázok 39: Triangulácie povrchu objektov: a) guľa ($Z(f_1)$ zo vzťahu 4); b) objekt Genus3 ($Z(f_2)$ zo vzťahu 5); c) objekt Jack ($Z(f_3)$ zo vzťahu 6); d) torus ($Z(f_4)$ zo vzťahu 7).

Objekt	$Z(f_1)$	$Z(f_2)$	$Z(f_3)$	$Z(f_4)$
Veľkosť kocky	0,512			
Max. hĺbka prerozdelenia	4			
Prahový uhol prerozdelenia	arccos(0,7) ≈45,6°			
Min. výška trojuholníka	0	0,045	0,05	0,05
Max. výška trojuholníka	0,25	0,175	0,1	0,05
Prahový uhol α_{lim}	arccos(0,95) ≈18,2°			180°
Konštanta k_{lim}	1,7			
Počet trojuholníkov	2968	13726	9592	35464
Počet vrcholov	1486	6858	4798	17732
Priemerný pomer strán	0,87	0,79	0,83	0,84
Priemerný pomer uhlov	0,8	0,68	0,74	0,74
Čas kroku MC (ms)	10	140	181	30
Čas kroku MT (ms)	60	431	430	1582
Celkový čas (ms)	80	611	691	1702

Tabuľka 1: Parametre triangulácie povrchu objektov: guľa ($Z(f_1)$ zo vzťahu 4); objekt Genus3 ($Z(f_2)$ zo vzťahu 5); objekt Jack ($Z(f_3)$ zo vzťahu 6); torus ($Z(f_4)$ zo vzťahu 7).

Prekrytie trojuholníkov a zacyklenie (pri vytváraní triangulácie alebo pri prechádzaní záplat) sú dosť častým javom (pozri tabuľku 2) a vyžadujú si vylepšenia. Prekrytie nastáva pri spájaní frontov triangulácie.

Veľkosť kocky	0,512					
Max. hĺbka prerozdelenia	4					
Prah. uhol prerozdelenia	arccos(0,7) ≈45,6°					
Min. výška trojuholníka	0,3	0,3	0,03	0,02	0,01	0,04
Max. výška trojuholníka	0,3	0,4	0,2	0,2	0,2	0,2
Prahový uhol α_{lim}	arccos(0,95) ≈18,2°					arccos(0,98) ≈11,5°
Konštanta k_{lim}	1,7					
Úspešnonsť	0	1	0	2	0	2

Tabuľka 2: Vplyv parametrov na stabilitu riešenia. Triangulácia povrchu objektu Genus3 ($Z(f_2)$ zo vzťahu 5). Úspešnosť: 0-triangulácia ukončená bez chýb, 1-zacyklenie, 2-prekrytie triangulácie.

4.3 Porovnanie kvality triangulácie

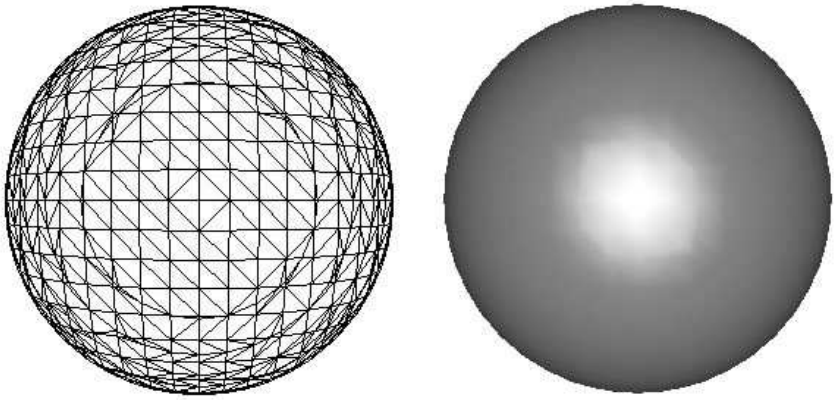
Kvalitu aproximácie budeme merať algebraickou vzdialenosťou ťažísk trojuholníkov od $Z(f)$, teda použijeme hodnotu $|f(\mathbf{T_c})|$, kde $\mathbf{T_c}$ je ťažisko vyšetrovaného trojuholníka. Takéto meranie nie je veľmi smerodajné, no na porovnanie rozdielnych triangulácií postačuje.

Pri prvom meraní použijeme funkciu definovanú vzťahom 4. Našu triangluláciu porovnávame s triangluláciou vygenerovanou programom HyperFun (HF) (pozri [hyp]). Parametre pre našu triangluláciu sú uvedené v tabuľke 1. Definícia funkcie f_1 zo vzťahu (4) pre HyperFun je (súbor genus3.hf)

```
my_model(x[3], a[1]){
my_model =9-x[1]^2-x[2]^2-x[3]^2;
}
```

a program je spúšťaný s nasledujúcimi parametrami:
`hfp.exe sphere.hf -b 3 -g 18 -d 6 -t -wrl sphere.wrl`.

Naša triangulácia je zobrazená na obr. 39 a), triangulácie programu HyperFun sú zobrazené na obrázku 40. V tomto prípade sme sa rozhodli zmerať kvalitu aproximácie aj pomocou pomeru povrchov. Ideálny povrch $Z(f_1)$ je $P = 4\pi r^2 = 36\pi$. Namerané výsledky sú zhrnuté v tabuľke 3.



Obrázok 40: Povrch gule definovaný ako $Z(f_1)$ zo vzťahu (4) vygenerovaný programom HyperFun.

	Náš algoritmus	HF
Počet trojuholníkov	2968	3008
Počet vrcholov	1486	1512
Priemerná odchýlka	0,03	0,06
Pomer povrchov	0,99	0,95
Priemerný pomer strán	0,87	0,52
Priemerný pomer uhlov	0,8	0,37

Tabuľka 3: Porovnanie kvality triangulácie povrchu gule definovaného ako $Z(f_1)$ zo vzťahu (4).

Ďalšie meranie sme vykonali na objekte Genus 3, ktorý bol určený funkciou zo vzťahu 5. Výsledky sme porovnávali s trianguláciou vygenerovanou programom HyperFun a s výsledkami algoritmu Edge Spinning (ES) a algoritmu MT publikovanými v [ČS04]. Parametre pre našu trianguláciu sú uvedené v tabuľke 1. Definícia funkcie f_2 zo vzťahu (5) pre HyperFun je (súbor genus3.hf)

```
my_model(x[3], a[1]){
my_model = (-1)*
           (256.0*x[3]^2-
            (1-x[1]^2/36.0-x[2]^2/12.25)*
            ((x[1]-3.9)^2+x[2]^2-1.44)*
            ((x[1]+3.9)^2+x[2]^2-1.44)*(x[1]^2+x[2]^2-1.44)
           );
}
```

a program je spúšťaný s nasledujúcimi parametrami:
`hfp.exe genus3.hf -b 6 -g 53 -d 3 -t -wrl genus3.wrl`.

Naša triangulácia je zobrazená na obr. 39 b). Triangulácie programu HyperFun a triangulácie získané algoritmom ES a MT sú zobrazené na obrázku 41. Namerané hodnoty sú zhrnuté v tabuľke 4.

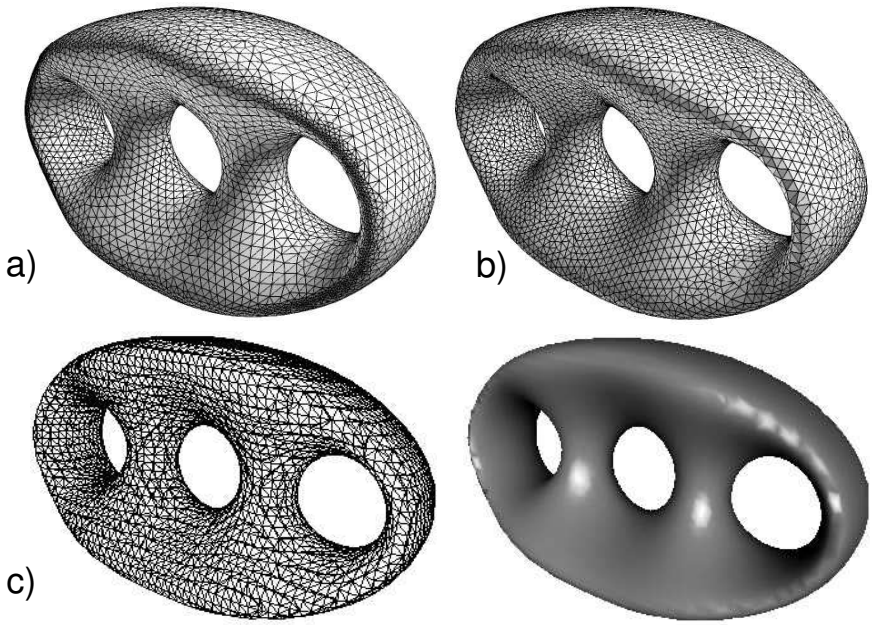
	Náš algoritmus	ES	MT	HF
Počet trojuholníkov	13726	13802	13695	13464
Počet vrcholov	6858	6897	6843	6728
Priemerná odchýlka	4,03	3,79	3,89	17,4
Priemerný pomer strán	0,79	0,81	0,83	0,53
Priemerný pomer uhlov	0,68	0,69	0,73	0,37

Tabuľka 4: Porovnanie kvality triangulácie povrchu objektu definovaného ako $Z(f_2)$ zo vzťahu (5).

Z uvedených meraní je vidno, že kvalitou aproximácie je náš algoritmus lepší ako HyperFun a porovnateľný s adaptívnym algoritmom ES. Oproti algoritmu MT dosahujeme horšie výsledky pri kvalite tvaru trojuholníkov, to je ale spôsobené tým, že algoritmus MT nie je adaptívny.

4.4 Porovnanie rýchlosti algoritmu

Rýchlosť algoritmu sme porovnávali s programom HyperFun, s adaptívnym algoritmom ES a algoritmom MT.



Obrázok 41: Povrch objektu Genus3 definovaný ako $Z(f_2)$ zo vzťahu (5): a) adaptívna metóda ES; b) MT; c) HyperFun.

Použili sme výsledky adaptívneho algoritmu ES a algoritmu MT publikované v [ČS04], ktoré boli namerané na počítači s procesorom AMD Athlon XP 1500+ a s veľkosťou pamäte 1GB.

Triangulovali sme $Z(f_2)$ zo vzťahu 5. Použili rovnaké nastavenia ako v odseku 4.3 na strane 50.

Výsledné triangulácie sú zobrazené na obrázku 41 a namerané výsledky v tabuľke 5.

	Náš algoritmus	ES	MT	HF
Počet trojuholníkov	13726	13802	13695	13464
Počet vrcholov	6858	6897	6843	6728
Celkový čas (ms)	611	861	101	2123

Tabuľka 5: Porovnanie času triangulácie povrchu objektu Genus3 definovaného ako $Z(f_2)$ zo vzťahu (5).

Program HyperFun používa metódu Marching Cubes a prehľadáva celý zobrazovaný priestor. Preto je jeho časová zložitosťou $O(n^3)$. Časová zložitosť nášho algoritmu rovnako ako algoritmu ES je $O(n^2)$. Do času triangu-

lácie programom HyperFun sme nezapočítavali čas potrebný na spracovanie súboru so zadaním funkcie.

Z nameraných hodnôt vidno, že náš algoritmus dosahuje nie len asymptoticky aj reálne lepší čas ako HyperFun. Dokonca dosahujeme aj lepší čas ako algoritmus ES, avšak autori algoritmu ES priznávajú, že implementáciu neoptimalizovali na rýchlosť. Algoritmus MT je mierne rýchlejší, ale to je spôsobené tým, že nevytvára adaptívnu trianguláciu.

5 Záver

Primárnym cieľom práce bolo navrhnúť a implementovať efektívny algoritmus na trianguláciu implicitne definovaných plôch. Veľkosť trojuholníkov sa mala adaptívne meniť podľa krivosti plochy. Vnútorné uhly trojuholníkov mali byť $\approx 60°$. Sekundárnym cieľom bolo navrhnúť rozšírenie algoritmu o detekciu ostrých hrán.

Merania potvrdili, že náš algoritmus generuje trianguláciu, ktorá dobre aproximuje $Z(f)$. Náš algoritmus sme porovnali s existujúcimi algoritmami ES a MT (pozri [ČS04]) a s programom HyperFun (pozri [hyp]). Kvalita a rýchlosť našej triangulácie je lepšia ako u programu HyperFun a porovnateľná s algoritmami ES a MT.

Navrhli sme spôsob detekcie ostrých hrán a aj spôsob ako jednoducho rozšíriť náš algoritmus.

Kritickým miestom algoritmu je detekcia prekrytia trojuholníkov a zacyklenie. Pre stabilitu algoritmu je nutné navrhnúť implementovať riešenia týchto problémov.

5.1 Budúca práca

Riešiť zacyklenie kroku MT. V našom algoritme detegujeme možné zacyklenie pri kroku Marching triangles. Bolo by vhodné navrhnúť spôsob riešenia triangulácie pri zacyklení.

Zlepšiť overovanie prekrytia trojuholníkov. Úspešnosť detekcie prekrytia je momentálne vysoko závislá na voľbe parametrov. Pre zvýšenie stability algoritmu je nutné navrhnúť robustnejšiu metódu.

Implementácia detekcie ostrých hrán. Navrhli sme spôsob ako detegovať ostré hrany. Naším cieľom je implementovať a experimentálne overiť tento algoritmus. Keďže detekcia hrán je v našom algoritme samostatným krokom, chceli by sme vyskúšať aj iné spôsoby detekcie, napr. pomocou duálnych mešov.

Redukcia počtu parametrov. V súčasnej podobe má náš algoritmus veľa parametrov. Naším cieľom je skúmať ich vzájomné závislosti a prípadné spôsoby ako ich určiť automaticky.

Nájdenie prvej kocky. Prvá kocka je momentálne vstupným parametrom nášho algoritmu. Zvažujeme implementovať rôzne spôsoby hľadania prvej kocky (napr. Exhaustive Search, Random Search, ...).

Spracovanie vstupu. Momentálne používame vkompilované funkcie definujúce $Z(f)$, teda nie je možné ich priamo modifikovať ani zadávať. Preto by sme chceli rozšíriť náš program o import funkcií. Vhodným formátom sa zdá byť Hyperfun modeling language.

Zlepšiť riadenie tvaru trojuholníkov. Kritérium minimálnej a maximálnej výšky trojuholníka sa zdá byť vhodnou metódou na riadenie tvaru trojuholníkov. No pre lepšie výsledky by bolo žiaduce tieto hodnoty upravovať podľa veľkosti aktívnej hrany. Zároveň na zlepšenie kvality aproximácie by bolo vhodné postupne triangulovať kocky od najviac prerozdelených (t.j. od kociek s najväčšou hĺbkou prerozdelenia).

Pamäťová optimalizácia. Momentálne sme sa sústredili hlavne na rýchlosť algoritmu, pričom sme nebrali ohľad na pamäťovú optimalizáciu. Pre väčšie triangulácie (rádovo $> 10^5$ trojuholníkov) je nutné kvôli vysokej pamäťovej náročnosti našu implementáciu optimalizovať aj týmto smerom.

Paralelizácia. V prvom kroku pokryjeme celú $Z(f)$ kockami, teda rozdelíme na menšie nezávislé časti. Preto by bolo dobré pouvažovať nad možnosťami paralelného spracovania týchto častí.

Zlepšiť kvalitu aproximácie MC. Pri prerozdelení kociek používame jednoduché kritérium uhlov, čím dosahujeme vysoké rýchlosti. Použitím zložitejších kritérií by sme mohli zlepšiť kvalitu aproximácie kroku MC a tým aj kvalitu výslednej aproximácie.

Referencie

[AFRS03] Marco Attene, Bianca Falcidieno, Jarek Rossignac, and Michela Spagnuolo. Edge-sharpener: recovering sharp features in triangulations of non-adaptively re-meshed surfaces. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 62–69, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[AG01] Samir Akkouche and Eric Galin. Adaptive implicit surface polygonization using marching triangles. *Computer Graphic Forum*, 20(2):67–80, June 2001.

[Blo] Jules Bloomenthal. Implicit surfaces.

[Blo88] Jules Bloomenthal. Polygonisation of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988.

[BS01] Ronald J. Balsys and Kevin G. Suffern. Visualisation of implicit surfaces. *Computer and Graphics*, 25(1):89–107, February 2001.

[ČS02] Martin Čermák and Václav Skala. Polygonization by the edge spinning. In *Algoritmy 2002*, pages 245–252, Slovakia, 2002. Univ.of Technology.

[ČS03] Martin Čermák and Václav Skala. Detection of sharp edges during polygonization of implicit surfaces by the edge spinning. In *Szczyrk 2003*, June 2003.

[ČS04] Martin Čermák and Václav Skala. Adaptive edge spinning algorithm for polygonization of implicit surfaces. In *Computer graphics international 2004*, pages 36–43, Los Alamitos, CA, USA, june 2004. IEEE Computer Society Press.

[CTM03] Hamish Carr, Thomas Theußl, and Torsten Möller. Isosurfaces on optimal regular samples. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 39–48, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[GL91] G. H. Golub and C. F. Van Loan. *Matrix Computation*. John Hopkins University Press, 1991. Second Edition.

[Har03] Erich Hartmann. Geometry and algorithms for computer aided design. Technical report, Department of Mathematics, Darmstadt

University of Technology, 2003. http://www.mathematik.tu-darmstadt.de/ ehartmann/cdgen0104.pdf.

[HI97] A. Hilton and J. Illingworth. Marching triangles: Delaunay implicit surface triangulation, 1997.

[Hop94] Hugues Hoppe. *Surface reconstruction from unorganized points*. PhD thesis, Department of Computer Science and Engineering, University of Washington, June 1994. http://research.microsoft.com/

[hyp] http://www.hyperfun.org.

[KBSS01] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 57–66, New York, NY, USA, 2001. ACM Press.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[MV04] Čermák Martin and Skala Václav. Surface curvature estimation for edge spinning algorithm. In *ICCS 2004*, Poland, June 2004.

[NB93] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *IEEE Comput. Graph. Appl.*, 13(6):33–41, 1993.

[NH91] Gregory M. Nielson and Bernd Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 83–91, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[NHS02] Gregory M. Nielson, Adam Huang, and Steve Sylvester. Approximating normals for marching cubes applied to locally supported isosurfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 459–466, Washington, DC, USA, 2002. IEEE Computer Society.

[Nie03] Gregory M. Nielson. Mc*: Star functions for marching cubes. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003*

(VIS'03), page 9, Washington, DC, USA, 2003. IEEE Computer Society.

[Nie04] Gregory M. Nielson. Dual marching cubes. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 489–496, Washington, DC, USA, 2004. IEEE Computer Society.

[OB02] Yutaka Ohtake and Alexander G. Belyaev. Dual/primal mesh optimization for polygonized implicit surfaces. In *SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 171–178, New York, NY, USA, 2002. ACM Press.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

Zoznam príloh

CD-ROM obsahujúci:

- Predkompilovanú verziu našej implementácie pre win32 (vyžaduje run-time knižnice prostredia Borland C++ Builder 6).
- Zdrojové súbory našej implementácie.
- Súbory VRML s výslednými trianguláciami.
- Elektronickú verziu textu diplomovej práce.