

## Zjednodušování povrchů s použitím chybové kvadriky

[xjahoda@fi.muni.cz](mailto:xjahoda@fi.muni.cz)  
[xkrcmar1@fi.muni.cz](mailto:xkrcmar1@fi.muni.cz)

## Úvod

- Mnoho aplikací produkuje či vyžaduje složité, vysoce detailní modely
- LOD však může kolísat – často se hodí, aby zbytečně detailní modely byly nahrazeny aproximacemi
- Představovaný algoritmus umí rychle vyrábět vysoce kvalitní aproximace trojúhelníkových modelů

## Úvod (2)

- Narozdíl od prvně ukázaného algoritmu (kontrakce hran) používá kontrakci dvojice vertexů
- Měl by být schopen spojit nezacelené oblasti modelu a poskytnout tak daleko lepší výsledky
- System pracuje i s non-manifoldy

## Úvod (3)

- Algoritmus je tedy založen na kontrakci dvojice vertexů, což je vlastně zobecnění kontrakce hran
- V průběhu algoritmu je získávána geometrická chyba pro každý vertex právě zpracovávaného modelu
- Tato chyba je reprezentována maticí kvadriky

## Slibované výhody

- Účinnost:
  - Algoritmus by měl být schopen zjednodušovat modely velice rychle
  - Vlastní implementace autorů uměla zjednodušit model se 70.000 plochami na model se 100 pl. do 15 sekund (SGI Indigo2, R10000 195MHz, 128MB RAM)
- Kvalita:
  - Algoritmus podle autorů produkuje vysoce věrné aproximace původních modelů, jsou zachovány primární rysy i po značném zjednodušení

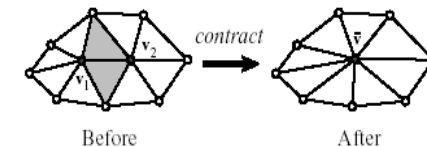
## Slibované výhody (2)

- Obecnost:
  - Na rozdíl od většiny zjednodušovacích algoritmů, tento je schopen spojit nezacelené oblasti modelu (agregace)
  - Takto je dosaženo mnohem lepších výsledků u modelů s mnoha rozpojenými částmi
  - Algoritmus je schopen překrýt díry i spojit roztržené části

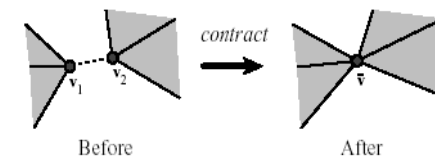
## Algoritmus

- Založen na iterativní kontrakci dvojic vertexů
- Dvojice vertexů  $v_1, v_2$  je přesunuta do pozice  $v$ , jsou spojeny všechny sousedící hrany do  $v_1$  a smazán bod  $v_2$
- Jestliže  $v_1, v_2$  je hrana, pak je alespoň jedna plocha odstraněna

## Algoritmus (2)



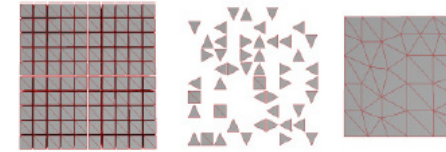
- V jiném případě jsou předtím oddělené části spojeny



## Algoritmus (3)

- Alg. Je založen ještě na obecnějším schématu: celá množina vertexů může být pohlcena do jednoho
- Kontrakce dvojice je však atomickou operací
- Na vstupu je tedy model a v každé iteraci je výstupem zjednodušenější model až je dosaženo výsledku

## Agregace



- V aplikacích jako rendering je často důležitější celkový tvar než topologie
- Na příkladu mřížka ze 100 krychlí, vprostřed zjednodušena kontrakcí hran, vpravo pak kontrakcí dvojic vertexů

## Volba dvojice

- Při inicializaci algoritmu je vybrána množina *platných* dvojic, dále se uvažují jen tyto (úvaha založena na tom, že v dobré aproximaci se body neposunou daleko ze svých pozic)
- Pár vertexů je *platný* jestliže
  - Tvoří hranu nebo
  - Jejich vzdálenost je menší než nějaký práh  $t$

## Volba dvojice (2)

- Práh musí být zvolen citlivě, je-li příliš vysoký, pak jsou spojeny i vzdálené oddělené plochy modelu
- Při volbě  $t=0$  pak dostáváme jednoduchý algoritmus kontrakce hran
- Když provedeme kontrakci  $v_1, v_2 \rightarrow v$ , tak dochází k tomu, že  $v_1$  získává nejen hrany vedoucí do  $v_2$ , ale přidává všechny dvojice vertexů, jejichž je  $v_2$  členem, do své vlastní množiny

## Chyba pomocí kvadriky

- K definici chyby, která vznikne při kontrakci dvojice použijeme symetrickou matici  $Q$   $4 \times 4$  asociovanou s každým vertexem a definujeme chybu jako kvadratickou formu získanou jako  $\Delta(v) = v^T Q v$
- Pro danou kontrakci  $v_1, v_2 \rightarrow v$  musíme vždy odvodit novou matici  $Q = Q_1 + Q_2$ .

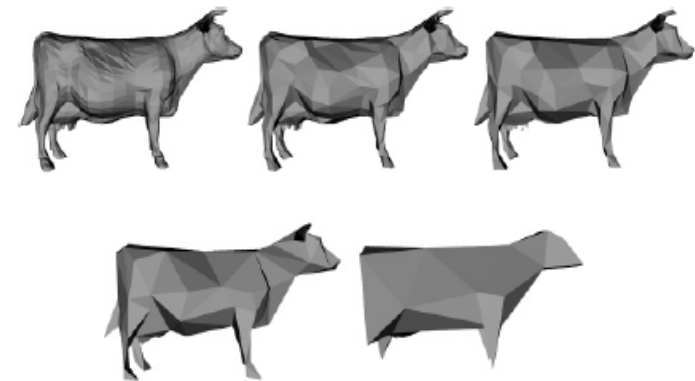
## Chyba pomocí kvadriky (2)

- K provedení kontrakce musíme také získat pozici pro  $v$
- Můžeme buď jednoduše zvolit  $v_1$  nebo  $v_2$  anebo  $v_1 + v_2 / 2$  v závislosti na tom, která volba vyprodukuje nejmenší chybu ( $\Delta(v)$ )
- Algoritmus však vyhledá pozici  $v$  tak, aby  $\Delta(v)$  bylo co nejmenší (chybová funkce  $\Delta(v)$  je kvadratická – nalezení minima)

## Shrnutí algoritmu

1. Spočítej matice  $Q$  pro všechny počáteční vertexy
2. Vyber platné páry
3. Spočítej optimální cíl kontrakce  $v$  pro každý platný pár  $v_1, v_2$ . Chyba  $v^T(Q_1 + Q_2)v$  je cenou kontrakce takové dvojice.
4. Ulož všechny dvojice na haldy, kde budou odshora dvojice s nejmenší cenou kontrakce
5. Iterativně odebírej dvojice vertexů  $v_1, v_2$  z haldy, proved' kontrakci a aktualizuj ceny všech platných párů obsahujících  $v_1$

## Příklad účinnosti algoritmu



- 5.804 ploch, 994,532,248 a 64

## Odvození chybové kvadriky

- Chyba v každém vertexu definována jako suma čtvercových vzdáleností všech ploch potkávajících se v daném vertexu:

$$\Delta(\mathbf{v}) = \Delta([v_x \ v_y \ v_z \ 1]^T) = \sum_{p \in \text{planes}(\mathbf{v})} (\mathbf{p}^T \mathbf{v})^2$$

## Odvození chybové kvadriky (2)

- Chybová metrika může být přepsána jako kvadratická forma ->

$$\begin{aligned} \Delta(\mathbf{v}) &= \sum_{p \in \text{planes}(\mathbf{v})} (\mathbf{v}^T \mathbf{p})(\mathbf{p}^T \mathbf{v}) \\ &= \sum_{p \in \text{planes}(\mathbf{v})} \mathbf{v}^T (\mathbf{p} \mathbf{p}^T) \mathbf{v} \\ &= \mathbf{v}^T \left( \sum_{p \in \text{planes}(\mathbf{v})} \mathbf{K}_p \right) \mathbf{v} \end{aligned}$$

- Kde  $\mathbf{K}_p$  je matice ->
- (plocha  $p$  je zapisována jako  $ax+by+cz+d=0$ )

$$\mathbf{K}_p = \mathbf{p} \mathbf{p}^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

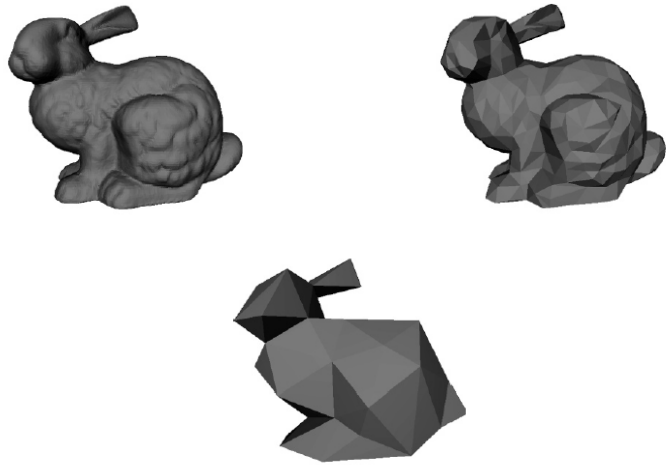
## Odvození chybové kvadriky (3)

- Základní chybová matice  $\mathbf{K}_p$  může být použita k nalezení čtvercové vzdálenosti jakéhokoli bodu v prostoru vzhledem k ploše  $p$ . Můžeme sečíst tyto kvadriky dohromady a reprezentovat celou množinu ploch pomocí jediné matice  $\mathbf{Q}$ .
- Počáteční chyba pro každý vertex je tedy 0, protože leží přímo v rovině všech v něm se potkávajících trojúhelníků

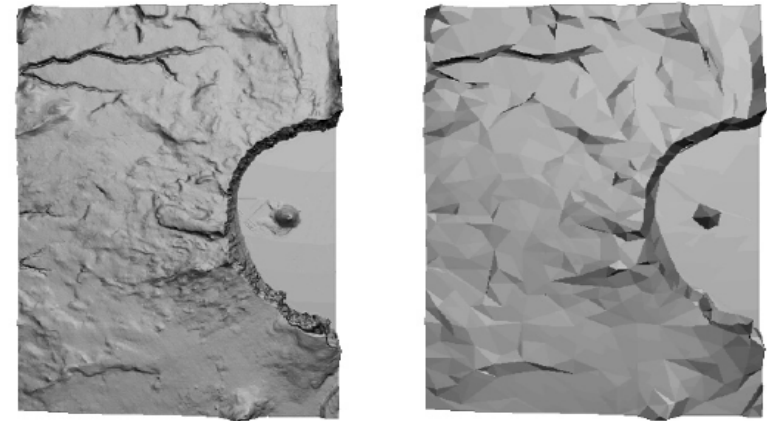
## Odvození chybové kvadriky (4)

- Takto spočítaná chybová kvadrika má také geometrický význam
- Takto aproximované povrchy jsou téměř vždy elipsoidy
- Výsledné  $\mathbf{v}$  je pak střed takového elipsoidu

## Příklady



## Příklady (2)



## Příklady (3)



Figure 14: **Original**. Bones of a human's left foot (4,204 faces). Note the many separate bone segments.



Figure 15: **Uniform Vertex Clustering**. 262 face approximation ( $11 \times 4 \times 4$  grid). Indiscriminate joining destroys approximation quality.



Figure 16: **Edge Contractions**. 250 face approximation. Bone segments at the ends of the toes have disappeared; the toes appear to be receding back into the foot.

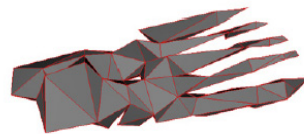


Figure 17: **Pair Contractions**. 250 face approximation ( $t = 0.318$ ). Toes are being merged into larger solid components. No receding artifacts. This model now contains 61 non-manifold edges.

## Příklady (4)

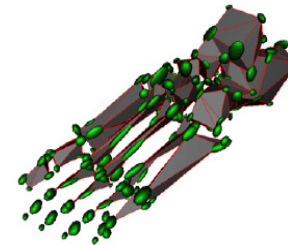


Figure 18: Level surfaces of the error quadrics at the vertices of the approximation shown in Figure 16.



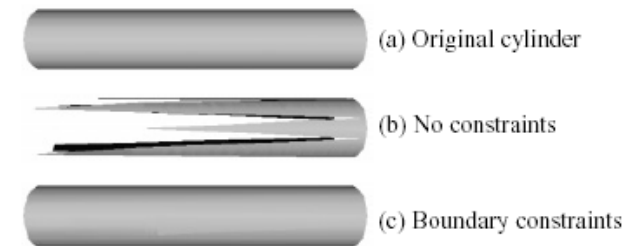
Figure 19: Pairs selected as valid during initialization (for Fig. 17). Red pairs are edges; green pairs are non-edges.

## Rozšíření algoritmu

- Autoři se pokusili algoritmus rozšířit tak, aby dokázal korektně zpracovat i vertexy nesoucí nějakou informaci o barvě (textuře)
- Dalším rozšířením je práce s křivkami na okrajích ploch

## Rozšíření algoritmu (2)

- Nahoře válec s neohraničenými konci (7960 ploch)
- Pod ním zjednodušení základním algoritmem (2460 ploch)
- Dole pak rozšířeným algoritmem (taky 2460 ploch)



## Rozšíření algoritmu (3)

- Během inicializace algoritmu označíme všechny okrajové hrany
- Ke každé přiléhající hraně k této okrajové pak dopočítáme plochu procházející touto přiléhající hranou a kolmou na povrch modelu (face)
- Toto je pak tzv. *constraint plane*
- Pro tuto plochu je vytvořena kvadrika a ta je vynásobena velkou penalizací (váha)

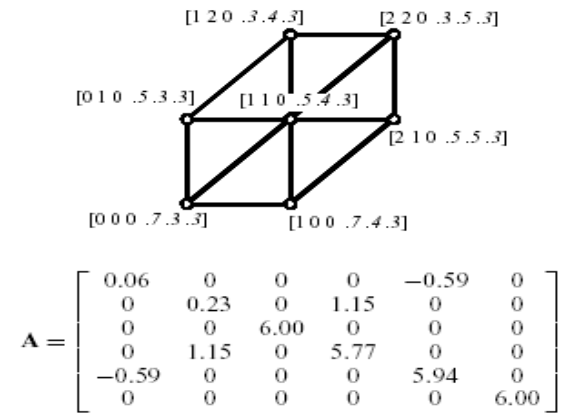
## Rozšíření algoritmu (4)

- Rozšíření základní metriky, které zahrne vlastnosti povrchu definované jako atributy vertexu
- Potřebujeme tedy měřit chyby, jestliže povrch má nějaké vlastnosti (např. barvu)
- Mohli bychom ke každému atributu vertexu spočítat zvlášť chybovou kvadriku a chybu měřit jako součet dvou kvadrik  $Q(v_{pos}) + R(v_{rgb})$
- Tento způsob nepočítá se vzájemným vztahem mezi pozicí a barvou (souřadnice textury)

## Rozšíření algoritmu (5)

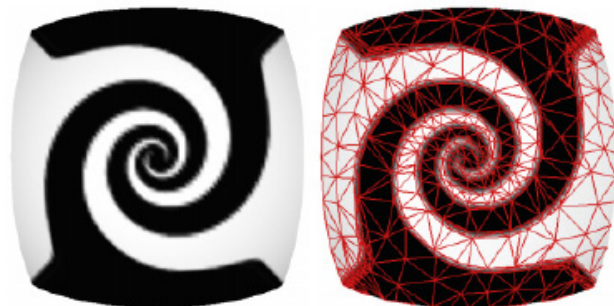
- Použije se tedy kvadrika vyšší dimenze, v jejíž koeficientech je zapsán vzájemný vztah mezi různými vlastnostmi
- V našem příkladě s barvou se tedy použijí vektory z  $R^6 \dots [x \ y \ z \ r \ g \ b]$ .
- Každé tři body každého troj. Modelu definují dvourozměrnou plochu, pro niž zkonstruujeme kvadriku, jež bude měřit čtvercovou vzdálenost jakéhokoli bodu z  $R^6$  k této ploše

## Rozšíření algoritmu (6)



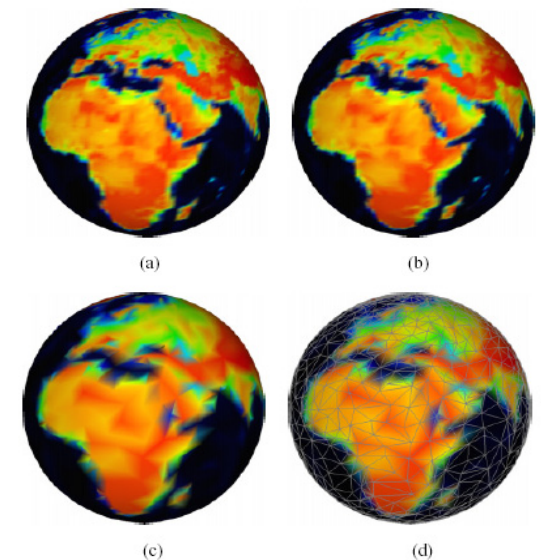
## Příklad použití 6D kvadriky

- Vlevo povrch s 18 tisíci plochami s barvou v každém vertexu
- Vpravo aproximace s 1000 plochami



## Další příklady

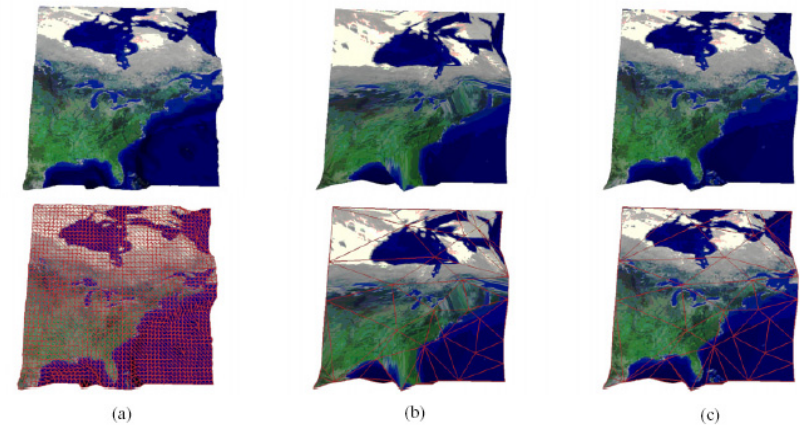
- 73.728
- 20.000
- 3.000



## Další příklady

- Na následujícím obrázku je zjednodušení modelu povrchové mapy z 3872 ploch na 53
- V prvním případě je zjednodušen způsobem neberoucím v potaz souřadnice textury

## Další příklady



## Zdroje

- <http://www.cs.cmu.edu/~garland/>
- <http://www.cs.cmu.edu/~ph/>