Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Úprava programového vybavení pro rekonstrukci povrchových modelů pro účely porovnání

Plzeň, 2003 **Petr Čížek**

Obsah

1. Úvod
1.1 Úvod do problematiky …… 2
1.2 Popis problémů ……... 3

2. Existující programy pro rekonstrukci ploch
2.1 Cocone …… 4
2.2 Powercrust …….. 5
2.3 CRUST ……... 6

3. Testovací data
3.1 Formát dat pro Cocone …….. 9
3.2 Formát dat pro Powercrust …… 9
3.3 Formát dat pro CRUST …… 10
3.4 Vybrané modely ……... 10

4. Dávkové spouštění
4.1 Popis …… 12
4.2 Použité techniky ……... 12
4.3 Ovládání ……... 12

5. Měření doby výpočtu
5.1 Možnosti ……... 14
5.2 Výsledky ……... 15

6. Vizuální porovnání
6.1 Pozorování …… 19
6.2 Výsledky ……... 19

7. Závěr …… 39

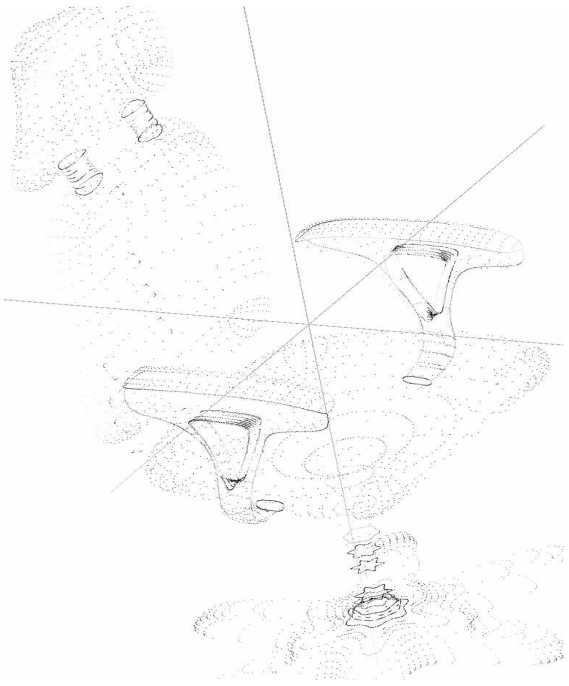
8. Zdroje …….. 40

1. Úvod

Na úvod si dovolím čtenáře seznámit se základními fakty a nastíněním problémů své práce.

1.1 Úvod do problematiky

V dnešní době vzrůstá potřeba digitální reprezentace objektů okolního světa, například ve strojírenství, zdravotnictví, architektuře, armádě, filmovém průmyslu a dalších odvětvích lidské činnosti. Jedním ze způsobů digitalizace je snímání povrchu pomocí 3D scanneru. Existuje několik způsobů snímání, důležité je však, že na výstupu získáme množinu bodů odpovídajících povrchu tělesa. Výstup vidíme např. na modelu židle[1] na obrázku 1-1.



Obr 1-1 : Množina bodů jako výstup z 3D scanneru

[1] Na opěrkách pro ruce splývají body zdánlivě v čáru. Zde vidíme nutnost jemnějšího vzorkování na ohybech nebo ostrých hranách.

Pro většinu dalších aplikací je tato reprezentace nedostačující, většinou potřebujeme znát povrch objektu, tedy nejčastěji body spojené do sítě polygonů. Právě to je předmětem metod rekonstrukce povrchu z bodů. Existuje mnoho metod, těmi se zde ale zabývat nebudeme. Podrobně jsou tyto metody popsány v [zdroje]. Účelem této práce je porovnání dostupných programů, které využívají některé z metod rekonstrukce, a zjištění rychlosti výpočtu a kvality výsledného povrchu. Hlavně se jedná o porovnání programu CRUST vytvořeného Ing. Varnuškou [1] pro Západočeskou univerzitu vůči dvěma konkurenčním programům zahraničních autorů.

1.2 Popis problémů

Jedním z problémů bylo zajistit správnou funkčnost konkurenčních programů v prostředí MS Windows, aby se dalo provést porovnání. A to po stránce kvality výstupu i rychlosti zpracování. K některým programům nebyl k dispozici zdrojový kód, ale jen spustitelný soubor. Z toho důvodu bylo žádoucí dávkové spouštění, neboť pro některé modely trval výpočet značnou dobu. Z toho také plyne nutnost zaznamenávat časové údaje s dobou výpočtu do logovacího souboru.
Hlavním problémem bylo vizuální porovnání výstupních dat všech tří programů mezi sebou a zhodnocení časové náročnosti rekonstrukce v jednotlivých programech. Přičemž bylo žádoucí, aby porovnání bylo spravedlivé a dalo se o něj opřít v dalším vývoji metod pro rekonstrukci ploch z oblaku bodů.

2. Existující programy pro rekonstrukci

V této kapitole se pokusím přiblížit testované programy, co o nich autoři tvrdí, i jak se ovládají.

2.1 Cocone

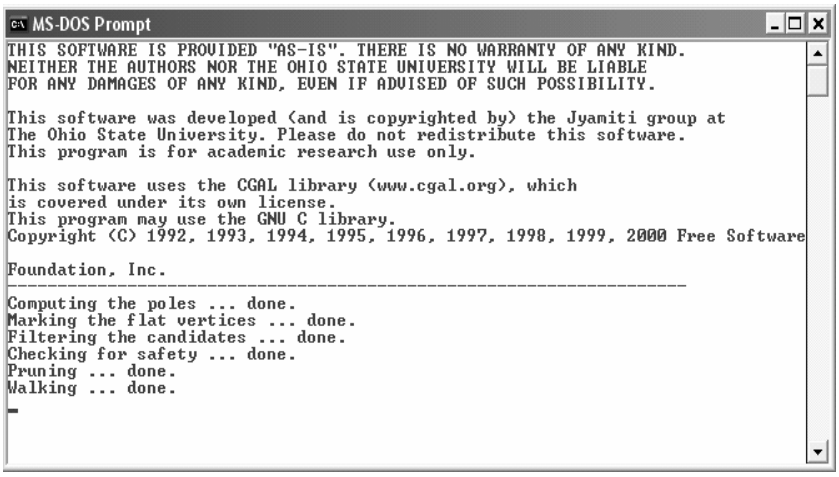
Program vytvořený skupinou kolem Thamala Krishna Deye [URL1] na univerzitě v Ohiu.
Cocone umí rekonstruovat povrch z oblaku bodů zadaných souřadnicemi ve 3D. Na výstupu obdržíme lineární aproximaci povrchu složenou z trojúhelníků s vrcholy pouze ve vstupních bodech. Tento software je založen na COCONE algoritmu, který používá jednoduchý Voronoi/Delaunay výpočet, vše je blíže popsáno v [3]. Program pracuje za předpokladu dostatečně hustých vstupních dat, která jsou navzorkována z hladké plochy. Podle tvrzení autorů umí Cocone zjistit hranice[1] v ploše, tedy je vhodný i pro rekonstrukci ploch, které mají hranice. Dokáže rovněž zjistit oblasti s nedostatečným počtem vzorků. Více o podvzorkování ve [2].

Jedná se o konzolovou aplikaci, spouští se z příkazové řádky a veškerá komunikace s uživatelem probíhá pouze v textovém režimu.
Spouští se s parametrem vstupních dat:

>cocone.exe input.pts

V průběhu výpočtu program nevypisuje téměř žádné informace, uživatel má pocit jakoby „zamrzl“. Nejprve se vypíší informace o programu, autorech a copyrightu. Poté probíhá výpočet a až ke konci program vypíše posledních šest řádek počínaje *Computing the poles.*

[1] Hranice: například ostré hrany, oblasti s velkou křivostí nebo detaily ve vstupních datech.



Obr. 2-1: Okno *Cocone.exe* za běhu

Výstupem programu je několik souborů, výsledná data jsou vždy uložena v souboru s příponou *.wrl*, v našem případě by se jednalo o *input.pts.wrl*. Formáty souborů jsou popsány v další kapitole.

2.2 Powercrust

Tento program je vytvořený skupinou kolem Niny Amenty [URL2] na univerzitě v Austinu. Autoři zaručují korektní výsledky, pokud jsou splněna kritéria hustoty vzorků vstupních dat. Tento software používá power crust algoritmus pro rekonstrukci plochy založený na transformaci střední osy. Více lze nalézt ve [4].

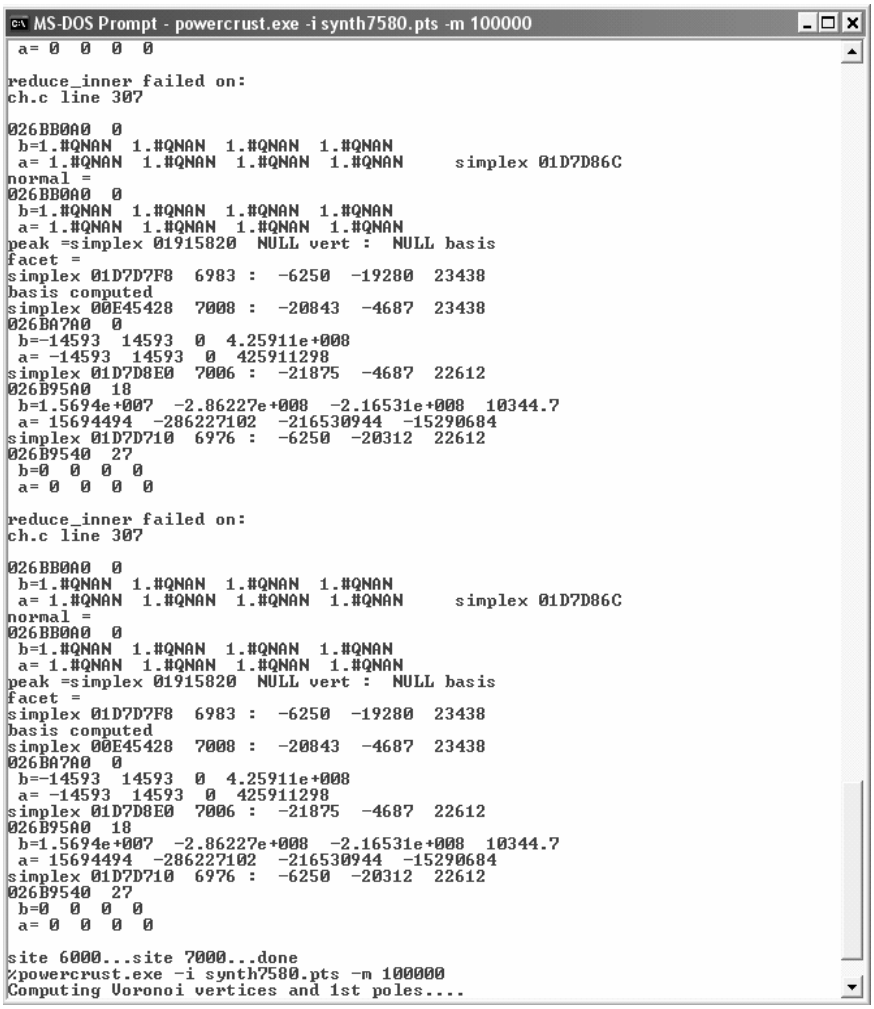
Jedná se opět o konzolovou aplikaci spouštěnou z příkazové řádky. Může být uvedeno několik různých parametrů:

-i <input> ... Specifikuje vstupní soubor.
-m <multip>... Určuje multiplikátor, tj. jakým číslem budou vstupní data násobena. Použijeme ho, jestliže jsou souřadnice vstupních dat reálná čísla menší než 1. Powercrust nejspíše počítá v celých číslech a pří zaokrouhlení by došlo ke ztrátě informace, proto je ve většině případů nutno uvést hodnotu multiplikátoru v řádu alespoň desítek tisíc.

Spuštění tedy vypadá následovně:

>powercrust.exe -i input.pts -m 100000

Za běhu programu dostává uživatel dostatek informací o probíhajícím výpočtu, jak lze vidět na obr 2.2.



Obr. 2-2 Okno *Powercrust.exe* za běhu

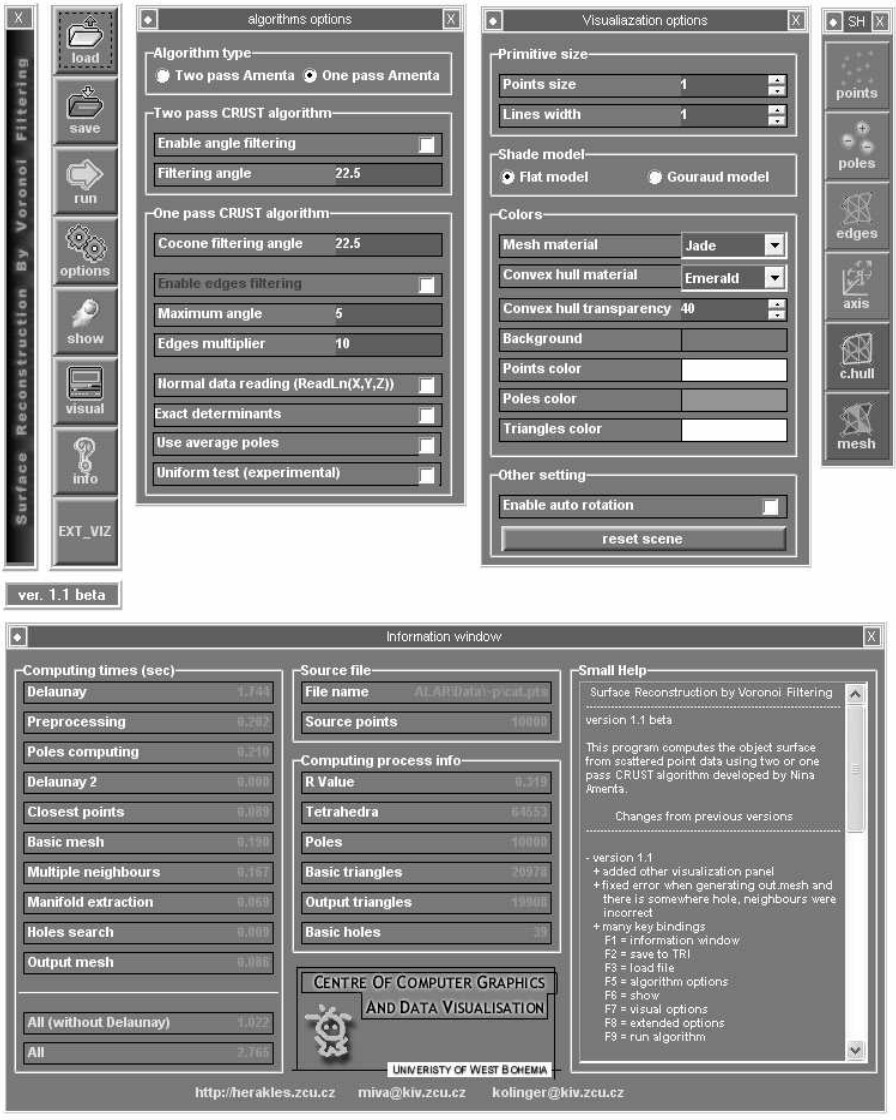
Po dokončení výpočtu program ještě provádí poněkud zdlouhavé generování výsledného souboru. Ten má příponu *.off*, více se lze dozvědět v následující kapitole.

K vygenerování výsledného souboru program potřebuje externí programy. Jedná se o „unixovské“ *cat.exe* a *rm.exe* sloužící pro spojování souborů (*cat*) a odstraňování souborů (*rm*). Ty lze získat například z prostředí CygWin [URL3]. Pokud tyto soubory nejsou k dispozici, Powercrust vypíše chybovou hlášku a skončí aniž by poskytl výsledný soubor s rekonstruovanou plochou.

2.3 CRUST

Program byl vytvořen Ing. Michalem Varnuškou na Západočeské univerzitě v Plzni. Jako hlavní algoritmus používá Crust vytvořený Ninou Amentou [URL2], a to dvouprůchodovou i novější jednoprůchodovou variantu. Více o implementaci v [1].

Jedná se o okenní aplikaci spustitelnou v prostředí Windows rodiny NT. Minimální požadavky na počítač jsou 128 MB operační paměti, procesor třídy Pentium a grafická karta podporující OpenGL. Od velikosti paměti se odvíjí maximální počet zpracovatelných bodů. Program potřebuje pro svůj běh ještě knihovnu pro přesnou aritmetiku *exact.dll* a knihovnu vizuálních komponent *vcl50.bpl*. Pro všechny testy jsem používal verzi *1.1 beta*, která obsahuje některá vylepšení oproti verzi *1.0* prezentované v [1].



Obr. 2.3 : Okna programu CRUST

Po startu aplikace se objeví hlavní okno s osmi tlačítky, odtud lze ovládat celý program. Tlačítkem *Load* zvolíme soubor se vstupními daty, *Run* spustí vlastní výpočet. Během výpočtu se zobrazuje jeho průběh. *Info* zobrazí informace o výpočtu - podrobně časy strávené na

jednotlivých fázích výpočtu. *Save* provede export rekonstruované plochy do souboru *.tri*. Po dokončení výpočtu se spustí vizualizace, parametry zobrazování můžeme měnit pomocí *Show*,*Visual* a *EXT_VIZ.*

Tlačítkem *Options* si můžeme zvolit typ a parametry algoritmu. Objeví se okno *algorithms options*. V něm se dá kliknutím zvolit variantu algoritmu, tedy dvouprůchodovou (Two pass Amenta) nebo jednoprůchodovou (*One pass Amenta*). Pro dvouprůchodovou verzi lze nastavit filtrování trojúhelníků pomocí jejich normál a úhlů (*Enable angle filtering*), pak určíme kritický úhel filtrování (*Filtering angle*). U jednoprůchodové verze lze obdobně nastavit *Cocone filtering angle*. Dá se zapnout i další úhlo-hranové filtrování (*Enable edges filtering*) a nastavit jeho dva parametry, maximální úhel filtrování (*Maximum angle*) a hranový multiplikátor (*Edges multiplier*). Ačkoliv se doporučuje nechat je na výchozích hodnotách, všechny nastavitelné parametry se dají měnit v rozahu <0, 90>.

3. Testovací data

Nyní popíšu formáty vstupních i výstupních souborů a vyberu modely pro testování.

3.1 Formát dat pro Cocone

Vstupní data musejí být uložena v textovém souboru. V každé řádce se nachází vždy pouze trojice reálných čísel odpovídající souřadnicím bodu v prostoru. V souboru se nenachází žádná hlavička, jinak dojde k ukončení programu s upozorněním na nekonzistentní vstupní data. Formát lze vidět na následujícím příkladu, kde by rekonstrukce probíhala ze čtyř bodů:

0.534901 0.582131 0.60291
0.479588 0.581171 0.366055
0.461174 0.571788 0.577064
0.5 0.639942 0.59103

Výstupní soubor má příponu *.wrl*, jedná se tedy o soubor jazyka VRML. Lze ho zobrazit například mnou používaným VRMLView [URL4].

3.2 Formát dat pro Powercrust

Powercrust používá stejný formát vstupních dat jako Cocone, tedy čistě textový soubor bez hlavičky. Pokud by soubor obsahoval hlavičku, Powercrust skončí chybovým výpisem o nekonzistentnosti vstupních dat.

Výstupní soubor má příponu *.off* a je ve formátu GeomView. Pro zobrazení jsem používal aplikaci RoffViewGL [URL6].

3.3 Formát dat pro CRUST

Vstupní soubor je opět textový, ale na rozdíl od obou předchozích programů musí obsahovat hlavičku indikující počet načítaných bodů (řádek souboru):

4
0.534901 0.582131 0.60291
0.479588 0.581171 0.366055
0.461174 0.571788 0.577064
0.5 0.639942 0.59103

Pokud v programu zvolíme tlačítko *Save*, provede se export do souboru ve formátu *.tri*. Jedná se o jednoduchý textový formát používaný hlavně v systému MVE [URL5], ale lze ho poměrně jednoduše konvertovat na jiné formáty. Prohlížet lze buď v samotném programu nebo v mnou napsané jednoduché aplikaci.

3.4 Vybrané modely

K experimentům jsem měl k dispozici zhruba padesát modelů. Na skupině některých modelů se algoritmy chovaly podobně, proto jsem z každé skupiny vybral typické představitele, na kterých pak probíhalo vlastní porovnávání. Vybíral jsem podle nejrůznějších kritérií:

- První model pro testování jsem zvolil *cow*. A to z důvodu, že je všeobecně známý, také neobsahuje příliš mnoho bodů a na některých místech modelu proto mohou vznikat oblasti podvzorkování, například na ocase, rohách, kolem očí a uší, tedy oblasti s vysokou křivostí.

- Další skupinu tvoří 3D grafy funkcí složených z pěti tisíc bodů. Volba padla na *poly*. Je v něm obsažena jak hladká plocha, tak i zlomová oblast. Právě tam je zajímavé pozorovat chování jednotlivých programů.

- Dále byl vybrán model *oilpump*. Ten lze zařadit mezi objekty se zhruba dvaceti tisíci body, kde je dostatek detailů na odhalení chyb v rekonstrukci.

- Z modelů postav nebo hlav jsem vybral datový set *baba*. Je náchylný na tvorbu nechtěných polygonů v blízkosti dvou povrchů (třeba nohy) a také obsahuje oblast podvzorkování.

- Pak jsem vybral kost stehenní, *bone,* díky značnému počtu bodů. Tak můžeme sledovat časové nároky programů.

- Jako další jsem zvolil model židle *chair* pro jeho velkou složitost.

- Zástupce jednoduchých modelů *hotdog* poskytla Nina Amenta společně s Powercrustem.
- Dále byla vybrána známá čajová konvička *teapot*, kde jsou data zadána uniformně s dostatečným počtem vzorků. Obsahuje i hranice, pravé úhly.
- Pro doplnění ještě přidám data *cat*, aby byl rozsah počtu bodů pravidelný.
- Rovněž tak jsem přidal model *mummy*.

Samostatnou kapitolou byla data „Vídeň“. Ta jsou extrémně zašuměná a obsahují velký počet bodů – více než dvě stě padesát tisíc. Pokud program nemá „rozumné“ paměťové nároky, nejspíše i s gigabajtem operační paměti uvízne v nekonečném odkládání do stránkovacího souboru.

V následující tabulce uvedu všechny zvolené modely i s počtem bodů.

	Název souboru	Počet bodů	Popis
1	*baba*	22625	Socha ženy
2	*bone*	68537	Kyčelní kloub
3	*cat*	10000	Model kočky
4	*chair*	17284	Kancelářská židle
5	*cow*	2905	Cannonical cow
6	*hotdogs*	1750	Dva párky
7	*mummy*	46970	Sarkofág
8	*oilpump*	30937	Pumpa
9	*poly*	5000	3D graf
10	*teapot*	26130	Konvička
11	*wien*	263575	Budova

Tab. 3.4 Modely vybrané k testování

4. Dávkové spouštění

Na následujících řádcích popíšu mnou vytvořený program pro dávkové spouštění.

4.1 Popis

Cílem bylo připravit „aparát“ pro dávkové spouštění programů Cocone a Powercrust. Zároveň bylo nutné měřit dobu výpočtu těchto programů. Pro tyto účely jsem vytvořil okenní aplikaci Batcher. Jedná se vlastně o grafické uživatelské rozhraní k uvedeným programům. Poskytuje intuitivní ovládání a možnost zápisu časových údajů do logovacího souboru.

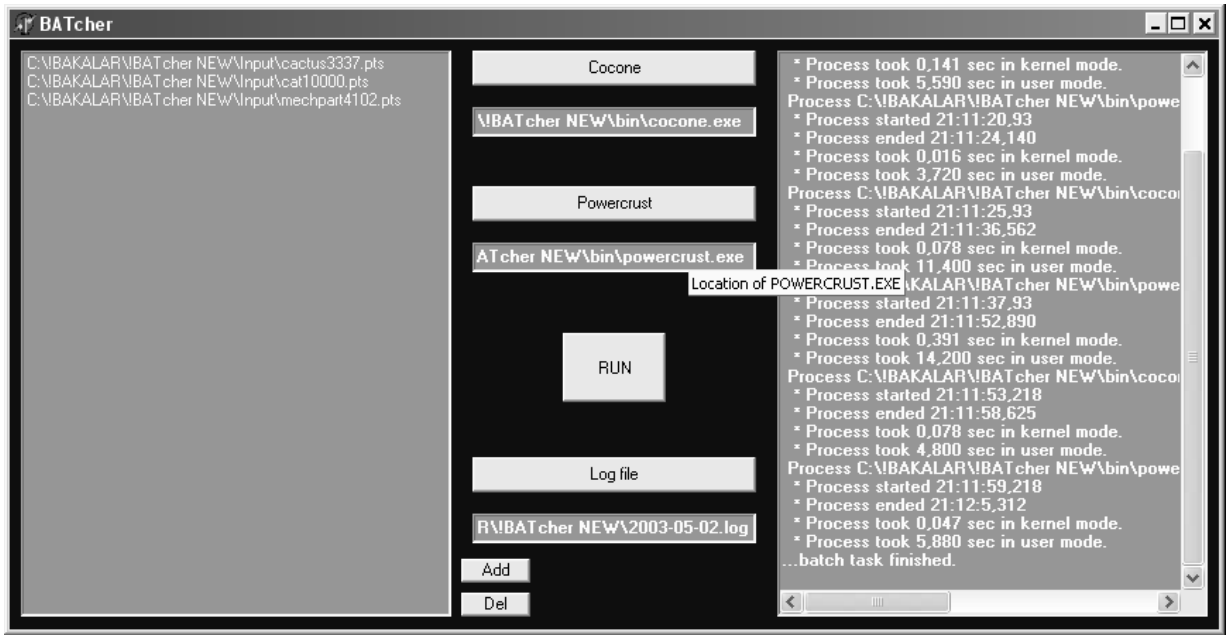
Vzhledem k provázanosti programu CRUST s vizuálními komponentami, nebylo možné bez velkých zásahů do programu vytvořit z něj jednoduchou konzolovou aplikaci, aby šla rovněž spouštět z příkazové řádky. Vše by se pak usnadnilo – všechny tři programy bychom spouštěli z jednoho prostředí. To ovšem není tolik na závadu, neboť v programu jsou přítomny poměrně kvalitní prostředky pro měření času.

4.2 Techniky

Vlastní spouštění programů je implementováno pomocí funkce Winows API *CreateProcess()*. Ta poskytuje naprostou kontrolu nad spouštěným procesem. Do parametrů se vloží jméno spouštěného souboru včetně plné cesty. Jako jeden z parametrů této funkce je také struktura typu *PROCESS_INFORMATION*. Ta vrací *handle* procesu. Díky tomuto *handle* pak můžeme získávat nejrůznější informace v průběhu procesu. Pro nás nejdůležitější je přesné měření času. Z funkce Windows API *GetProcessTimes()* lze získat nejen začátek a konec procesu, ale i počet milisekund, které si proces vyžádal v uživatelském režimu a v režimu jádra.

4.3 Ovládání

Jedná se o klasickou aplikaci operačního systému Windows. Po spuštění se objeví hlavní okno programu, z kterého jsou dostupné všechny ovládací prvky. Většina ovládacích prvků obsahuje tzv. hint, tedy po podržení myši nad prvkem se objeví nápověda. Vše nejlépe ilustruje obrázek 5-3.



Obr. 5-3: Okno programu Batcher

Na ploše okna nalevo je *ListBox* obsahující jména souborů s daty k rekonstrukci, vpravo potom *Memo*, do kterého se vypisují informace o spuštěném programu. V *ListBoxu* můžeme jména souborů libovolně mazat nebo přidávat nová, k tomu slouží dvě malá tlačítka *Del* a *Ins*. Pomocí tlačítek uprostřed zvolíme cestu k programům Cocone a Powercrust a případně také jméno souboru pro výpis (logovací soubor). Velkým tlačítkem *Run* se spustí zvolená úloha.

Po spuštění programu se prohledá aktuální adresář a pokud obsahuje podadresář s názvem *Input* a v něm soubory pro rekonstrukci s příponou *.pts*, automaticky se jména těchto souborů doplní do *ListBoxu*. Rovněž pokud je v aktuálním adresáři podadresář *bin* a v něm spustitelné soubory *Cocone.exe* a *Powercrust.exe*, automaticky se doplní cesta k nim.

Do podadresáře *Output* se ukládají výsledky. Ke jménům vstupních souborů se přiřadí přípona *.wrl* pro rekonstruovaná data z Cocone a *.off* pro data z Powercrustu:

Input.pts
Input.pts.wrl
Input.pts.off

Po dokončení výsledku můžeme provést sběr časových údajů, ať již z logu a nebo ze samotného programu (komponenta *Memo*).

5. Měření doby výpočtu

Zde se budu zabývat měřením doby rekonstrukce výsledné plochy.

5.1 Možnosti

Jelikož jsem měl od Cocone k dispozici pouze spustitelný soubor, nezbývalo mi, než se uchýlit k možnosti externího měření času v nějakém dalším programu. K tomu jsem si tedy vytvořil Batcher. V něm je implementováno dvojí měření běhu spustitelných souborů. Zaprvé je to klasické měření start-stop, zadruhé pak získání dodatečných časových informací o procesu, který odpovídá spouštěnému programu. Mám tedy k dispozici jednak hrubý čas běhu (dále t_1) programu. Ten odpovídá době strávené uživatelem od spuštění programu po obdržení výstupního souboru. A na druhé straně dostanu ještě čistý čas, který si proces vyžádal (dále t_2). To znamená bez jakýchkoliv vstupně-výstupních operací. Tím lze do jisté míry zamezit zkreslení výsledného času, pokud dojde ke swapování. Eliminuje se také čas pro generování výstupního souboru. Tato operace trvá zvláště u rozsáhlé datové množiny značnou dobu. Dalším důvodem pro dvojí měření bylo i to, že původní verze CRUSTu nezaznamenává dobu načítání vstupních dat ani generování výstupního souboru. To ale již ve verzi 1.1, kterou jsem používal na testování, bylo doplněno. Načítání oproti ostatním úkonům potřebným pro rekonstrukci sice zabere zanedbatelný čas, zvláště u větších datových množin, ale takto můžeme naměřené časy lépe porovnat s konkurenčními programy. Proto ve všech měřeních budu uvádět oba dva časy. Vše lze snadno vyčíst z grafů v následující podkapitole.

Z důvodu velké spotřeby paměti nemohla být stejná měření provedena i na datech „Vídeň“. Například pro dvouprůchodovou verzi algoritmu programu CRUST by bylo zapotřebí asi 3 GB operační paměti. V době testování nebyl k dispozici takto výkonný stroj. Výsledky se podařilo získat pouze pro jednoprůchodový CRUST a pro Cocone. Výsledky se podařilo získat pouze pro jednoprůchodový CRUST a pro Cocone. Výsledný čas uvedu pouze orientačně: nepřesáhl 1000 sekund u Cocone a zhruba čtvrtinu z tohoto času potřeboval CRUST. Jak vypadala rekonstrukce těchto dat je vidět v kapitole 6.

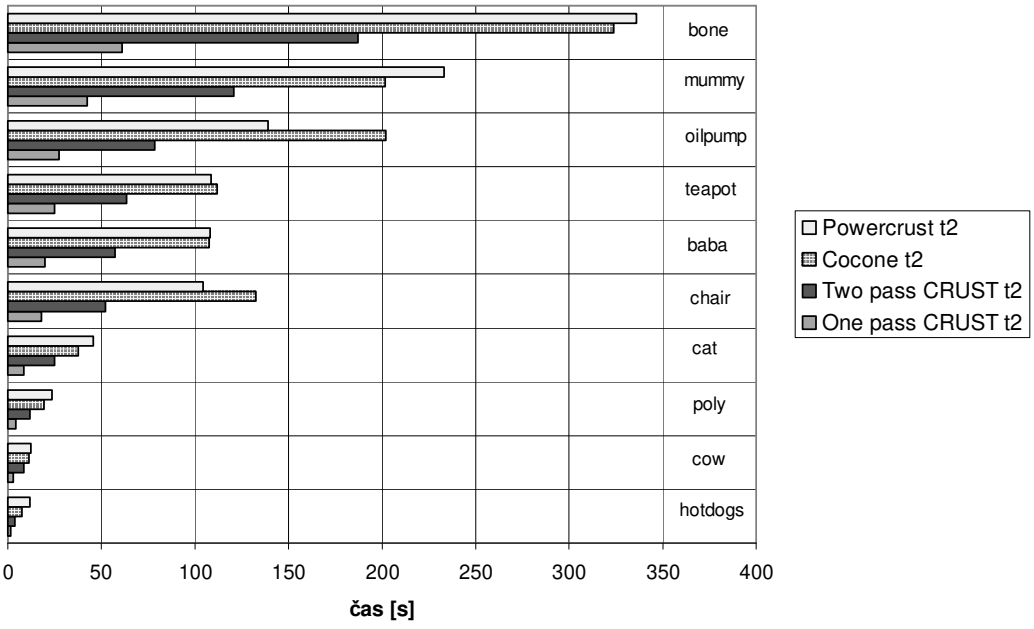
5.2 Výsledky

Následující testy probíhaly na počítači s procesorem AMD Athlon 500 MHz s 1 GB operační paměti. V tabulce 5-2-1 můžeme vidět dosažené časy každého programu pro jednotlivé modely. Časy t_1 a t_2 se od sebe příliš neliší, vyjma rozsáhlejších vstupních dat (*bone, mummy*). Snažil jsem se také omezit vliv ostatních programů běžících na pozadí jejich ukončením. Rovněž nedocházelo k odkládání do virtuální paměti.

model	Cocone t_1	Cocone t_2	Powercrust t_1	Powercrust t_2	CRUST 1 pass t_1	CRUST 1 pass t_2	CRUST 2 pass t_1	CRUST 2 pass t_2
baba	108,5	107,8	115,3	107,9	22,5	19,8	59,5	57,3
bone	325,4	324	456,5	336,1	66,3	61,1	201,3	186,9
cat	38,6	37,8	48,2	45,6	9,2	8,4	24,8	24,8
chair	11,8	11,3	12,6	12	3	2,7	8,7	8,3
cow	7,7	7,4	12,3	11,7	1,8	1,6	3,9	3,8
hotdogs	133,5	132,7	110,2	104,2	19,7	18	53,5	52,2
mummy	202,5	201,5	253,1	233,1	46,3	42,4	121,8	120,8
oilpump	203,1	202,1	149	139	29,1	27,4	78,9	78,3
poly	19,6	19,3	24,2	23,3	4,5	4,2	11,8	11,7
teapot	112,4	111,8	114,2	108,4	26,9	25,1	64,4	63,5

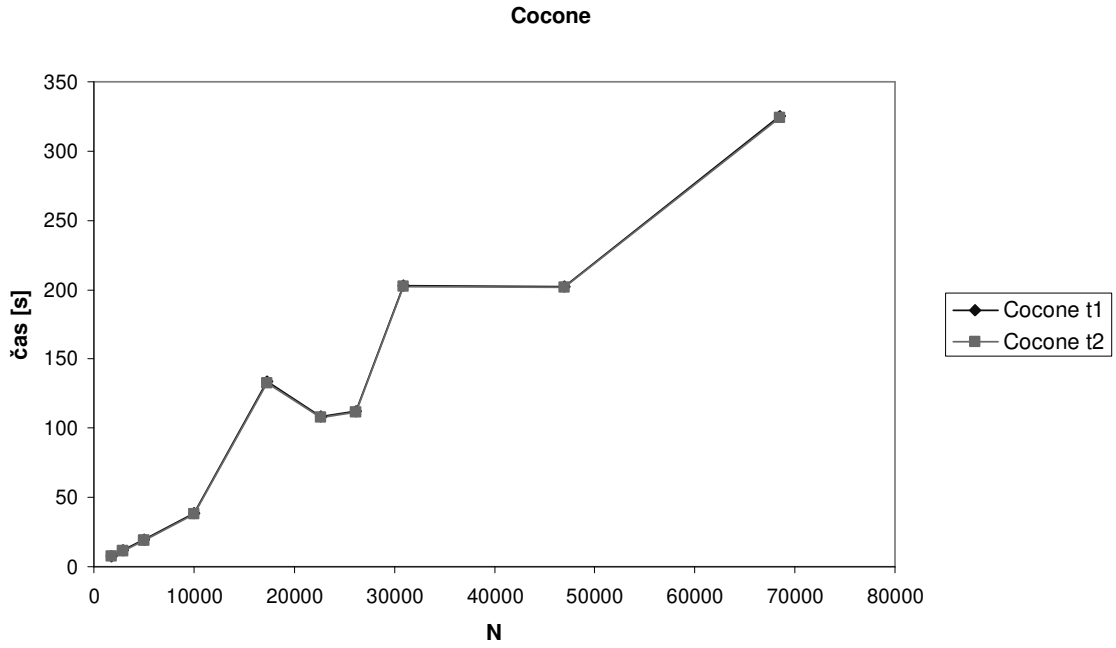
Tab. 5-2-1: Naměřené časy

Graf 5-2-1 ukazuje tyto hodnoty seřazené a přehlednější.

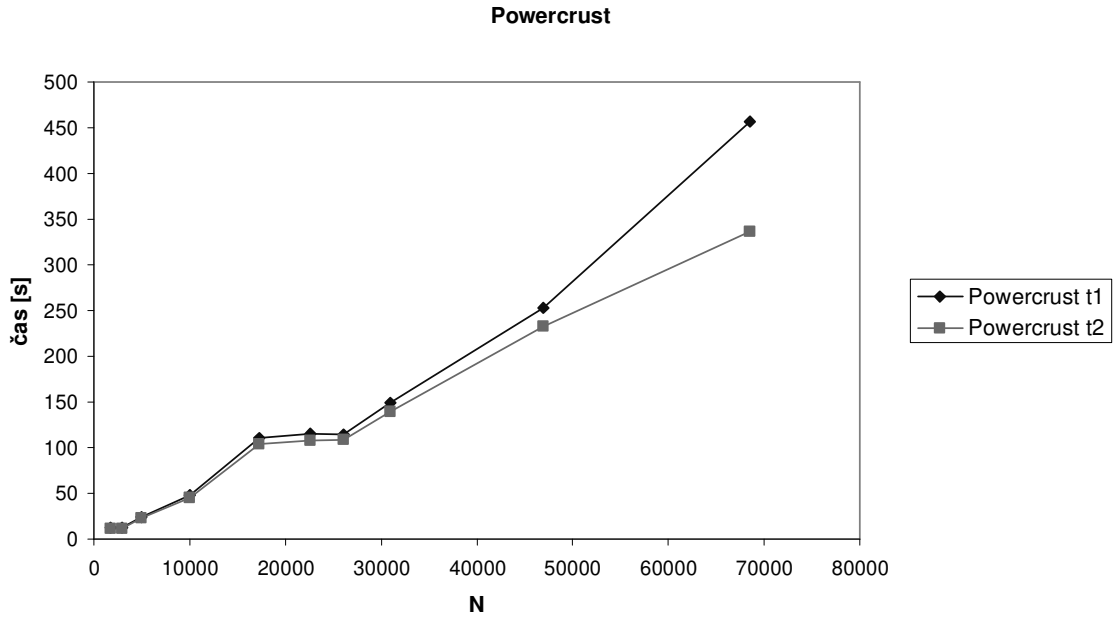


Graf 5-2-1: Jednoduše porovnatelné časy pro každý model

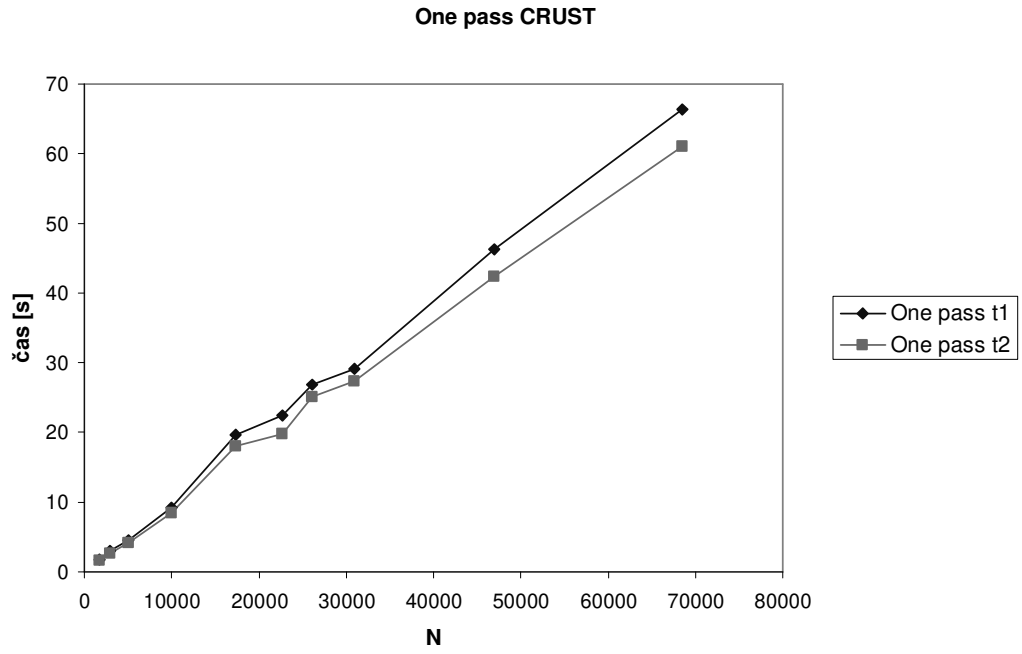
V grafech 5-2-2, 5-2-3, 5-2-4 a 5-2-5 vidíme závislost času výpočtu na velikosti vstupních dat (N udává počet bodů na vstupu).



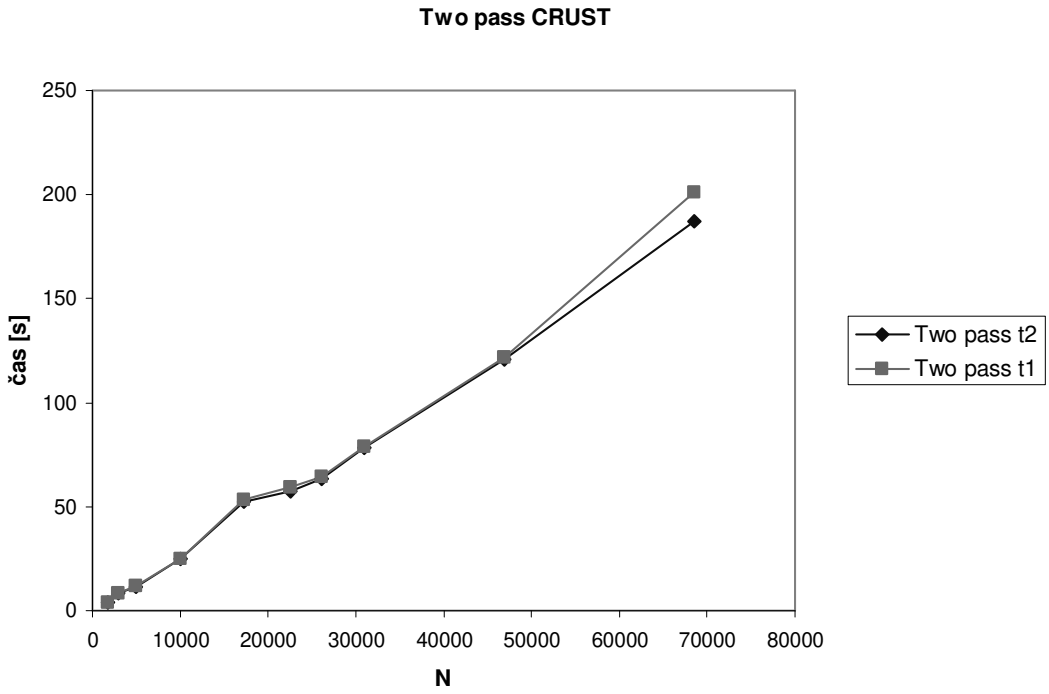
Graf 5-2-2: Závislost doby výpočtu na počtu vstupních bodů u Cocone



Graf 5-2-3: Závislost doby výpočtu na počtu vstupních bodů u Powercrustu



Graf 5-2-4: Závislost doby výpočtu na počtu vstupních bodů u CRUSTu (jednoprůchodová varianta)



Graf 5-2-5: Závislost doby výpočtu na počtu vstupních bodů u CRUSTu (dvouprůchodová varianta)

Z výsledků je patrné, že program CRUST vytvořený na Západočeské univerzitě je ze všech tří nejrychlejší. Ať už se použije jednoprůchodová (one pass) nebo dvouprůchodová (two pass) verze algoritmu. Nejhůře dopadl program Powercrust, ten vykazuje i značnou spotřebu paměti spolu se zdlouhavým generováním výstupu.

6. Vizuální srovnání

Nyní se budu zabývat porovnáním výstupů po stránce kvality

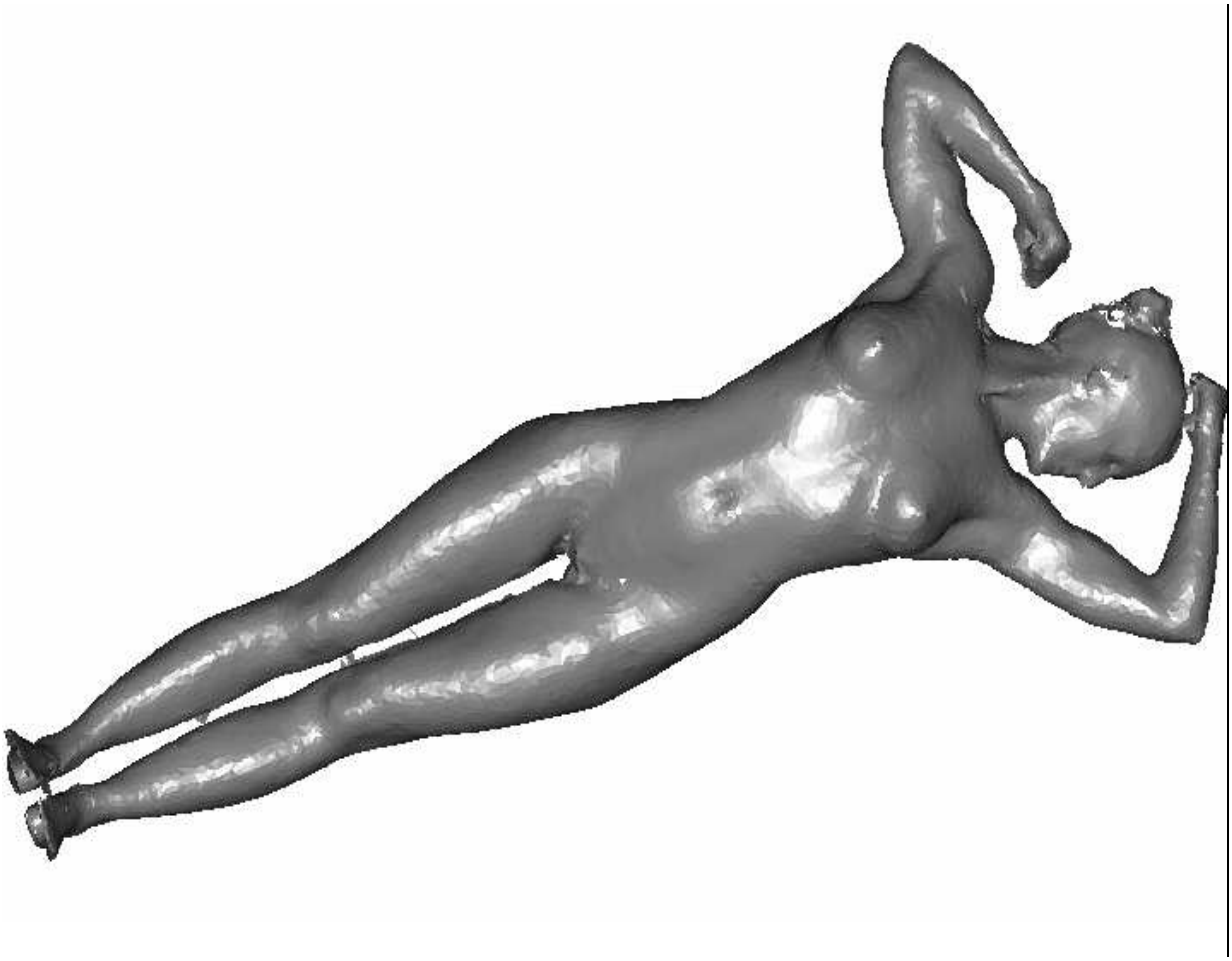
6.1 Popis

Vizuální srovnání je záležitost subjektivní. Lidské oko může lehce přehlédnout některé chyby ve výsledných datech. Já jsem se především zaměřil na hledání děr ve vygenerovaném povrchu, na nežádoucí polygony a na kvalitu hran nebo oblastí s nedostatečným počtem vstupních bodů. Svoji pozornost jsem věnoval ještě dalším problémovým oblastem a to překrývajícím se trojúhelníkům a jejich správné orientaci. U výsledného povrchu z Cocone totiž nebyly normály shodně orientované, tedy normály směřovaly dovnitř i ven z tělesa. U zbylých dvou programů byla orientace v pořádku. To již u každého modelu nebudu znovu uvádět.

6.2 Výsledky

Následuje série ilustrativních obrázků s popisky ke kvalitě výstupu testovaných programů. Vždy platí, že první uvedu celkový pohled na model (zvolil jsem výstup z CRUSTu, číslováno např. 6-2-1), pak celkově zhodnotím výstupy jednotlivých programů a nakonec následuje série obrázků dokumentující problémové oblasti modelu (za číslem obrázku následuje ještě písmeno, např. 6-2-2a). Pro série platí pořadí zleva: Cocone, Powercrust, CRUST.

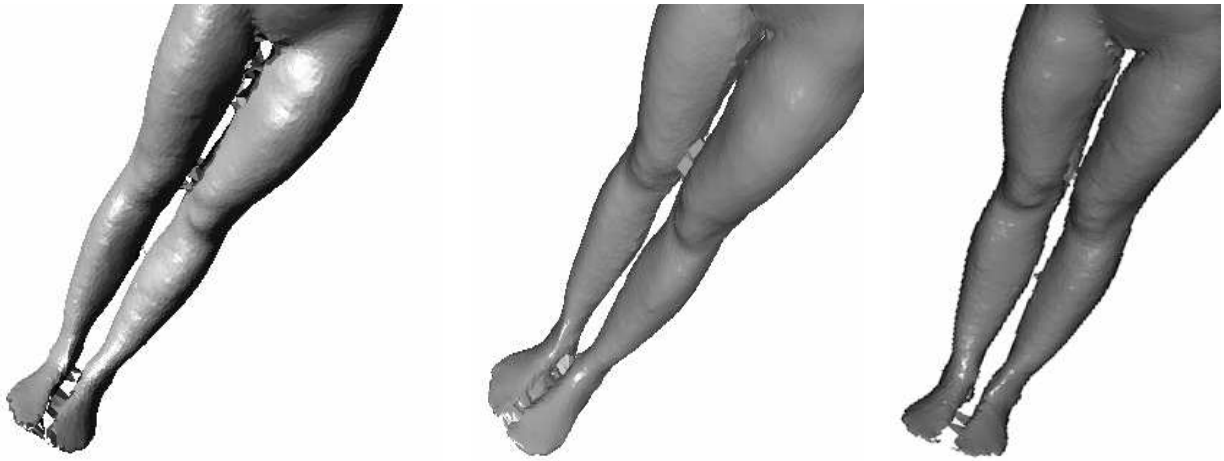
- *Baba*



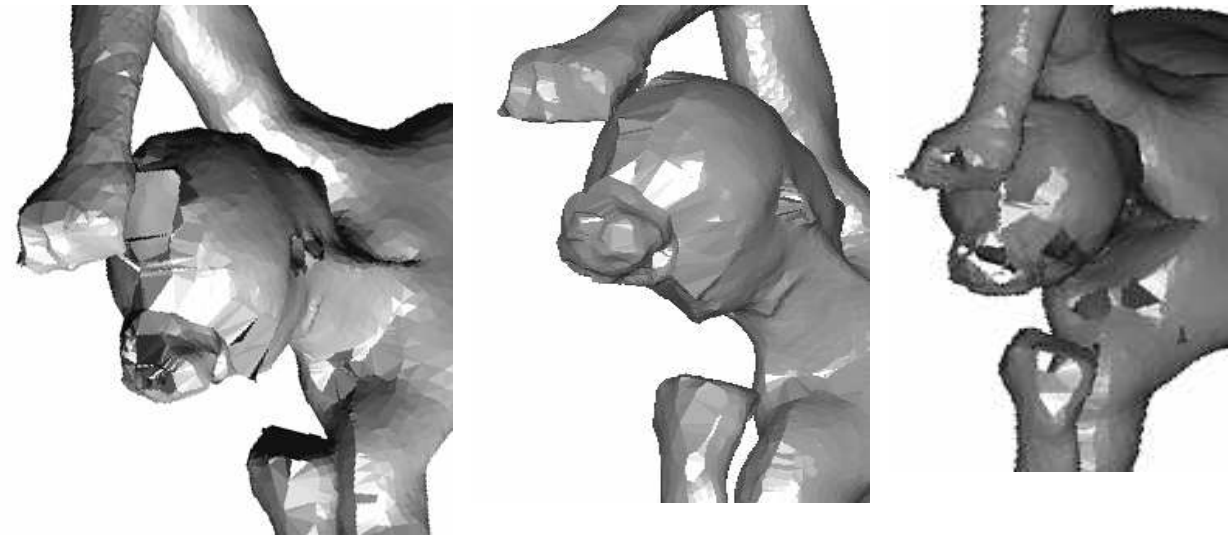
Obr. 6-2-1 ***baba***

Tento model obsahoval místa s těsnou sousedností ploch, jedná se o nohy. Testované programy měly tendenci vytvářet trojúhelníky i mezi těmito plochami (Obr. 6-2-1a). Nejvíce se efekt projevoval na chodidlech. Na hlavě kolem drdolu nebyl dostatečný počet vzorků.

Powercrust nejlépe zrekonstruoval oblast drdolu a ostré hrany. Nejméně nežádoucích trojúhelníků mezi blízkými plochami vykazoval CRUST, ale v oblasti hlavy se nacházelo hodně děr (Obr 6-2-1b).

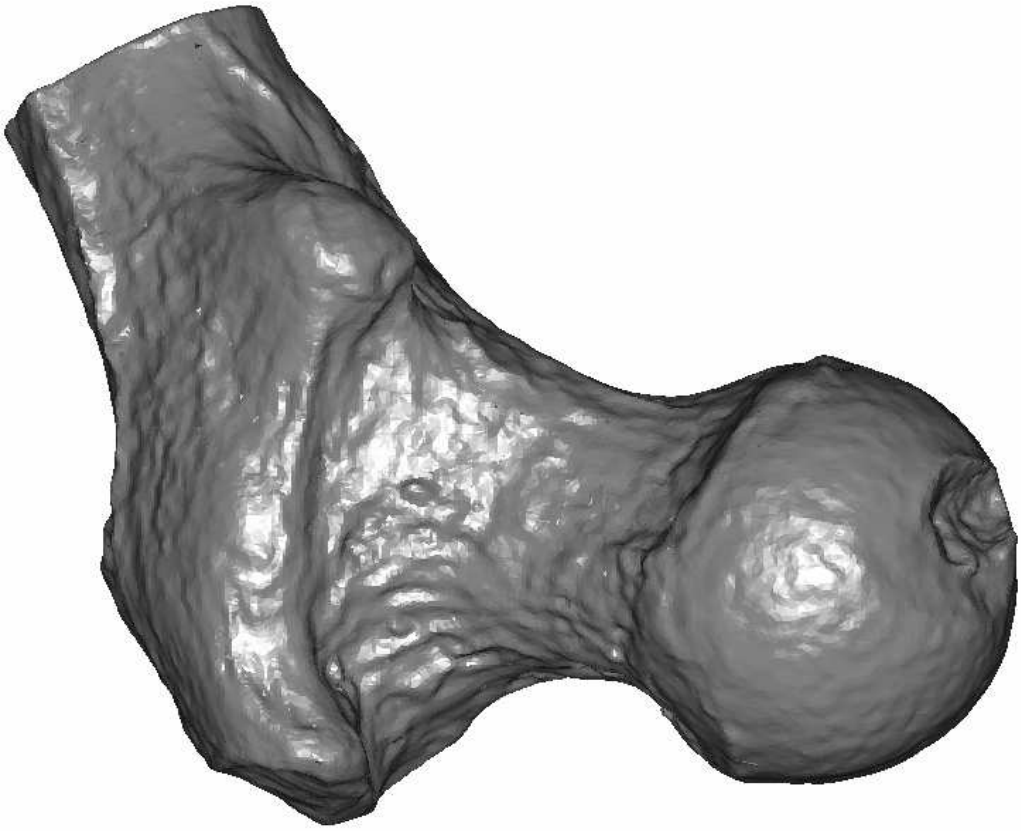


Obr 6-2-1a: *baba*, oblast nohou



Obr 6-2-1b: *baba*, oblast copu

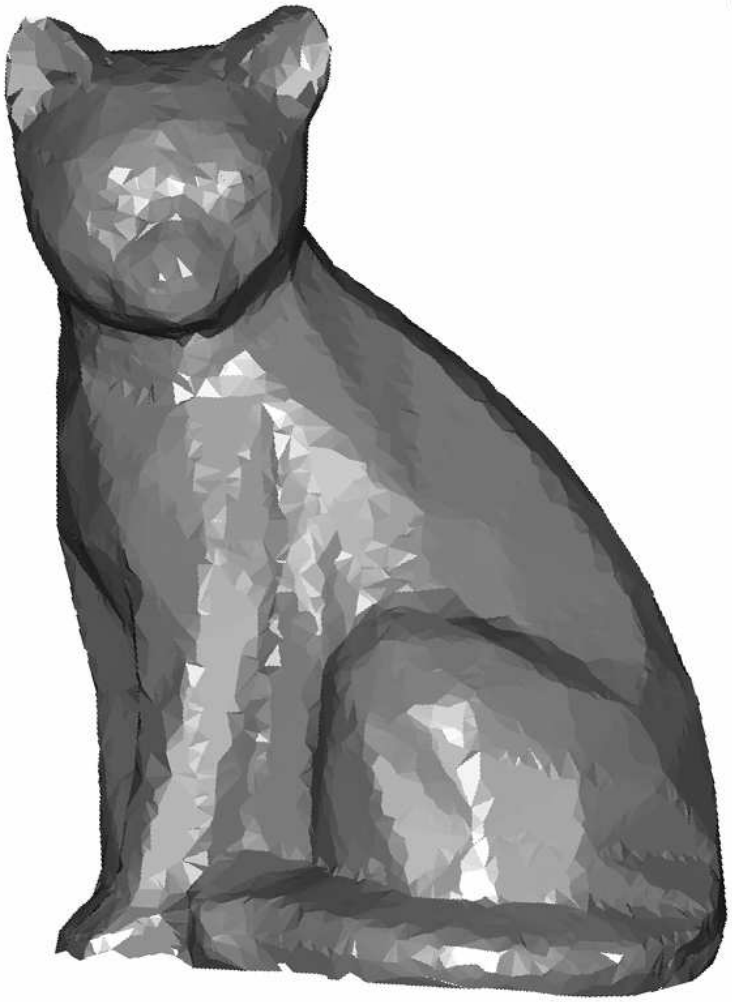
- *Bone*



Obr. 6-2-2: ***bone***

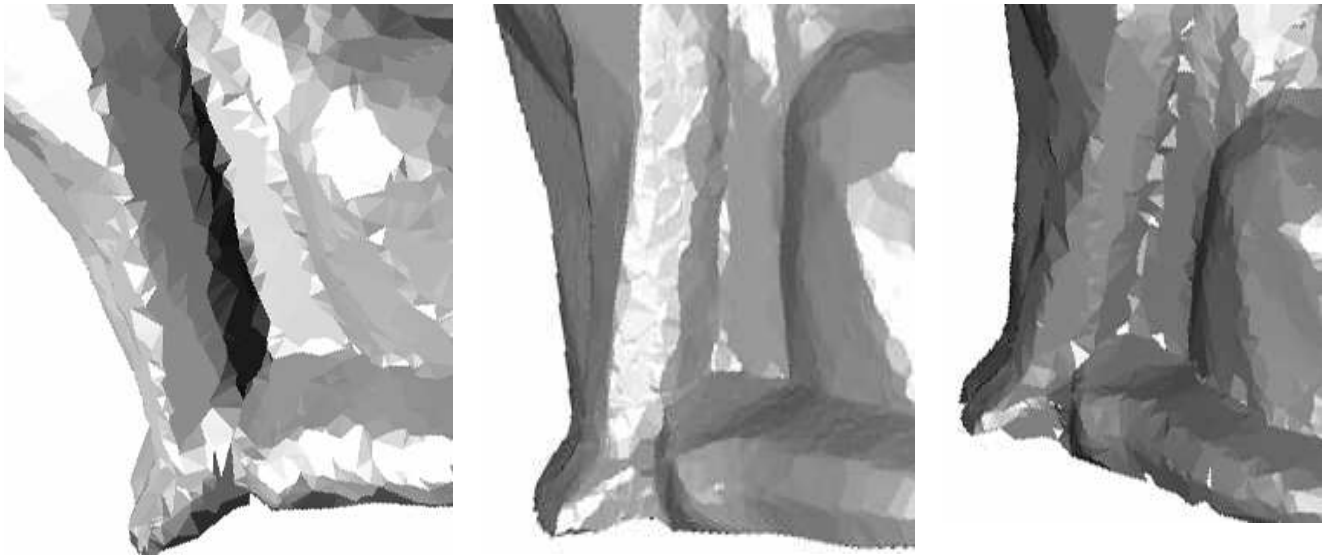
Zde se ukázalo, že model obsahuje dostatečné množství bodů a programům nedělalo problémy jej dobře zrekonstruovat. Zajímavý byl spíše při testování časové náročnosti. Rozdíly mezi kvalitou výstupu jednotlivých programů jsou zde sotva postřehnutelné.

- *Cat*

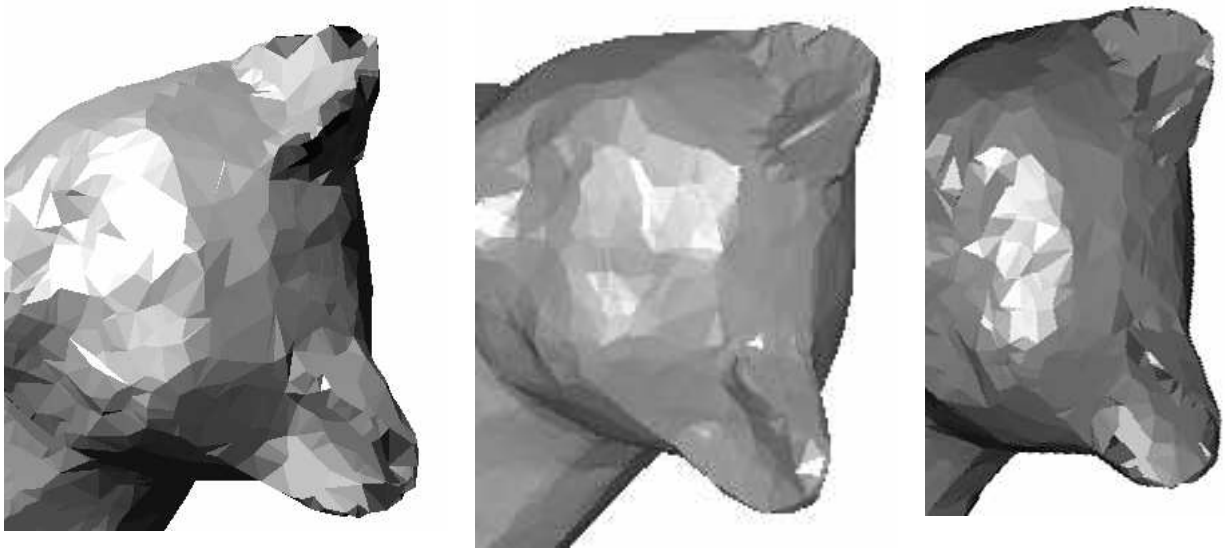


Obr. 6-2-3: ***cat***

V těchto datech se vyskytovalo hodně ostrých přechodů, jako mezi tělem a nohou (Obr. 6-2-3a) a také na uších. V těchto místech byl nejúspěšnější Powercrust, ostatní dva programy produkovaly drsné přechody, výstup z CRUSTu v nich ještě navíc obsahoval díry. Cocone vypadal subjektivně nejhůře i v oblasti uší (Obr. 6-2-3b).



Obr. 6-2-3a: *cat*, oblast nohy



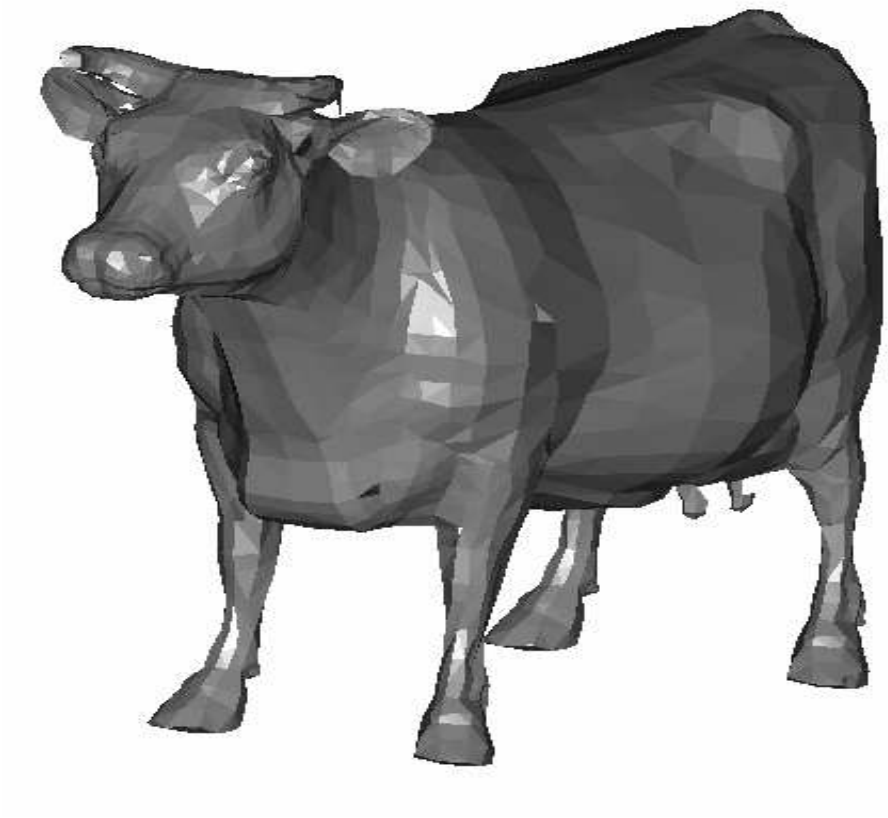
Obr. 6-2-3b: *cat*, oblast uší

- *Chair*

Obr. 6-2-4: ***chair***

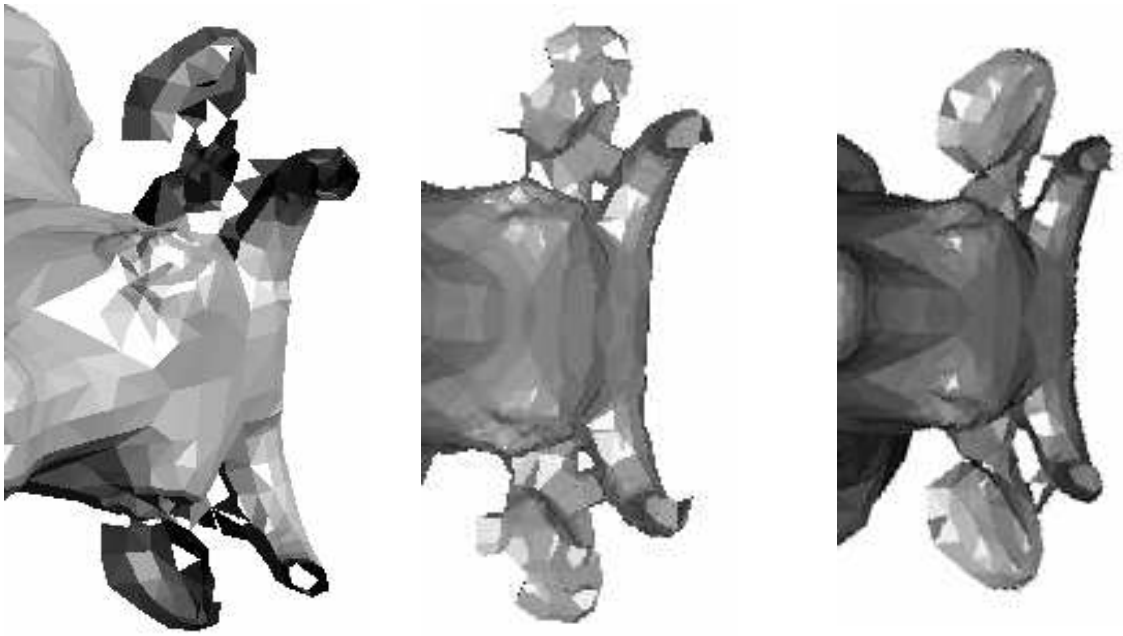
Uvedený model byl nejsložitější, obsahoval mnoho hran a oblastí podvzorkování. Programy si s ním také poradily ze všech dat nejhůře. Špatně vypadal výstup z Cocone, obsahoval mnoho děr. A Powercrust byl přeci jen úspěšnější než CRUST. Uvádím pouze ilustrativní obrázky z každého programu vedle sebe, tak vidíme nejlépe rozdíly (Obr. 6-2-4).

- *Cow*

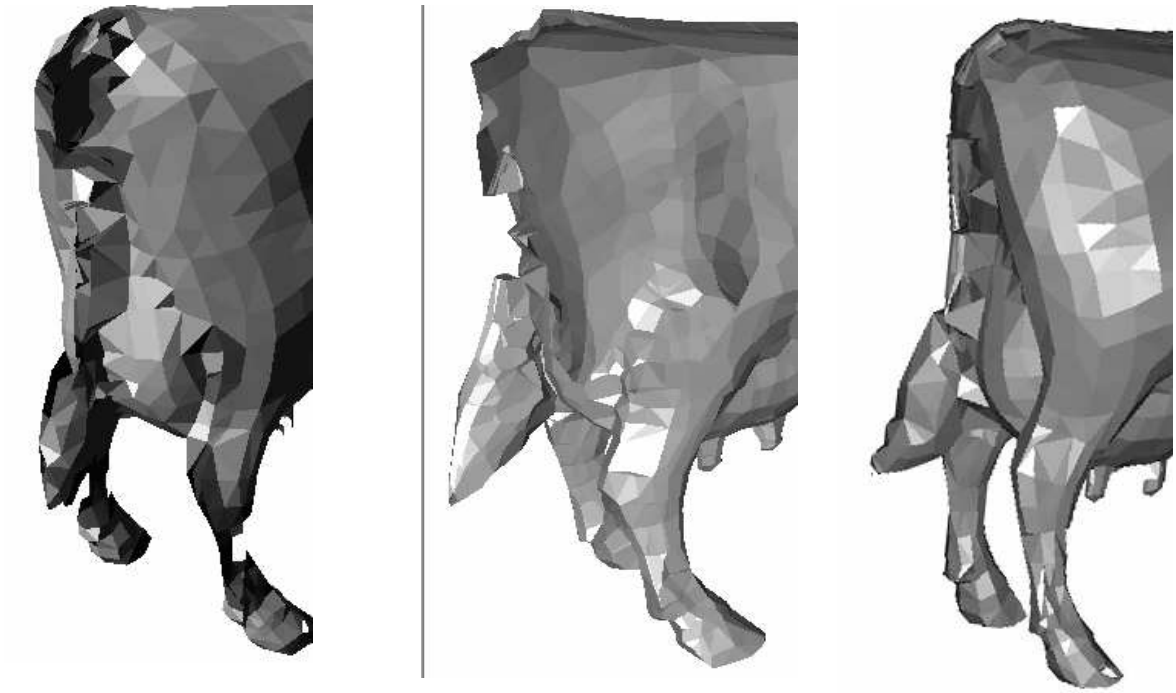


Obr. 6-2-5: ***cow***

Známý model *cannonical cow* byl záludný v oblasti ostrých hran, tedy na rozích a v detailech kolem uší. CRUST zde dopadl nejlépe před Powercrustem, který byl slabší na uších (Obr. 6-2-5a). Cocone produkoval příliš mnoho děr. Také oblast kolem ocasu obsahovala mnoho děr (Obr 6-2-5b).

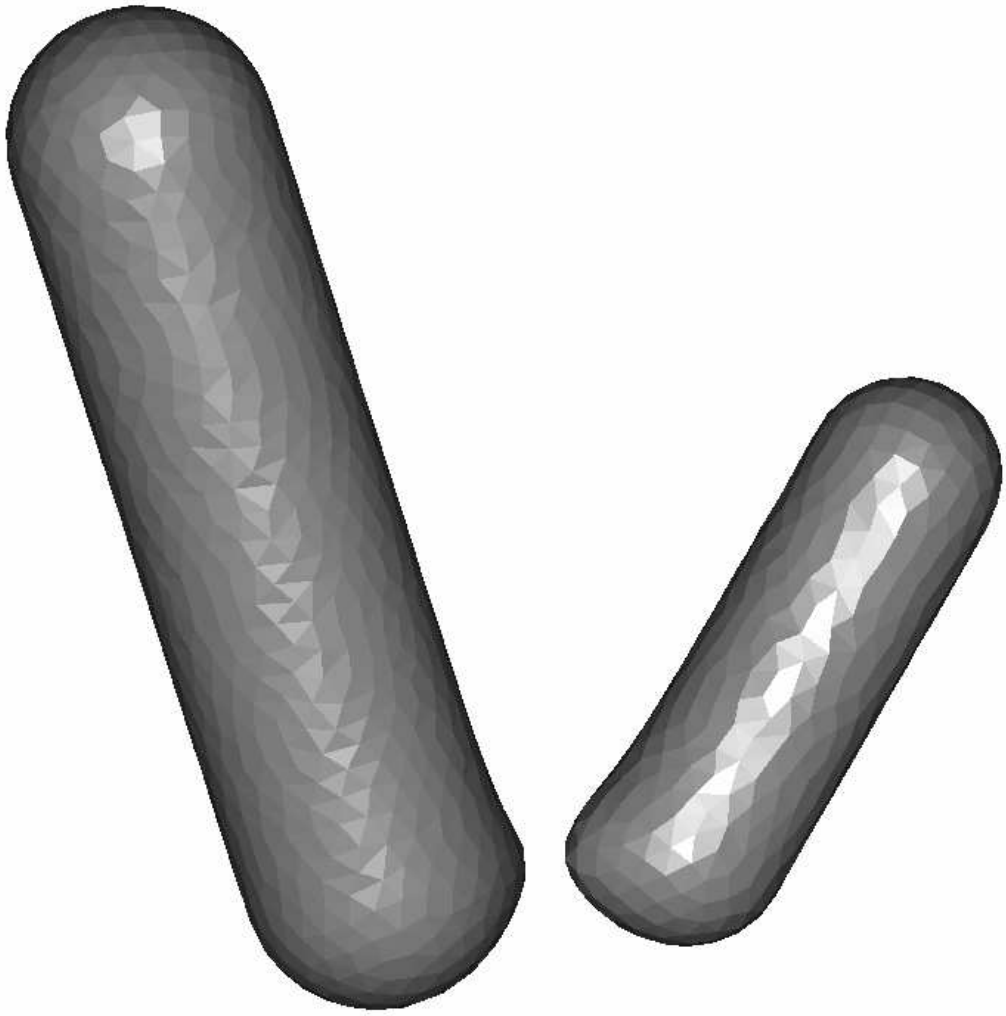


Obr. 6-2-5a: *cow*, oblast kolem rohů



Obr. 6-2-5b: *cow*, oblast kolem ocasu

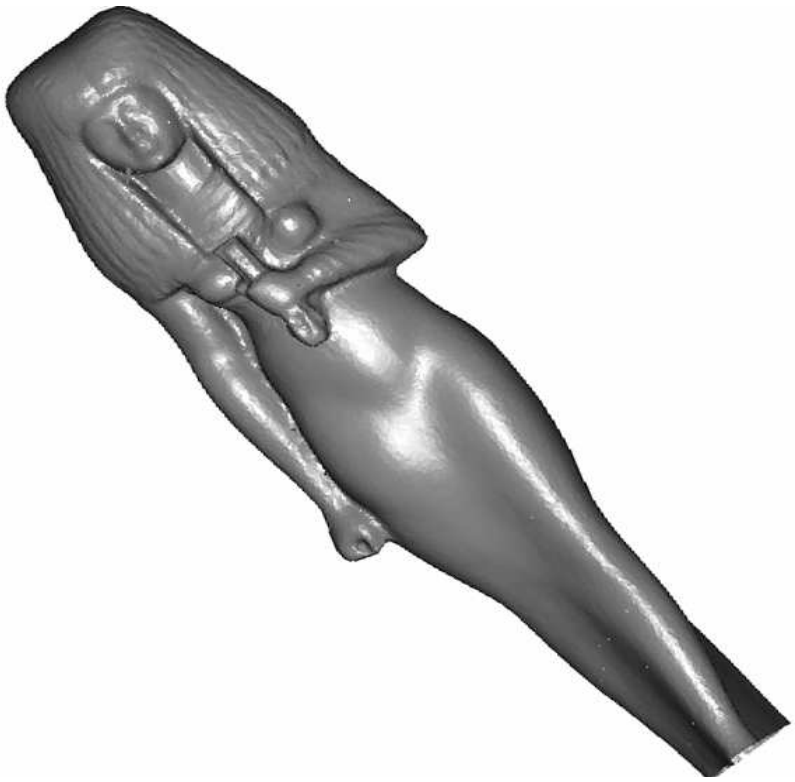
- *Hotdogs*



Obr. 6-2-6: ***hotdogs***

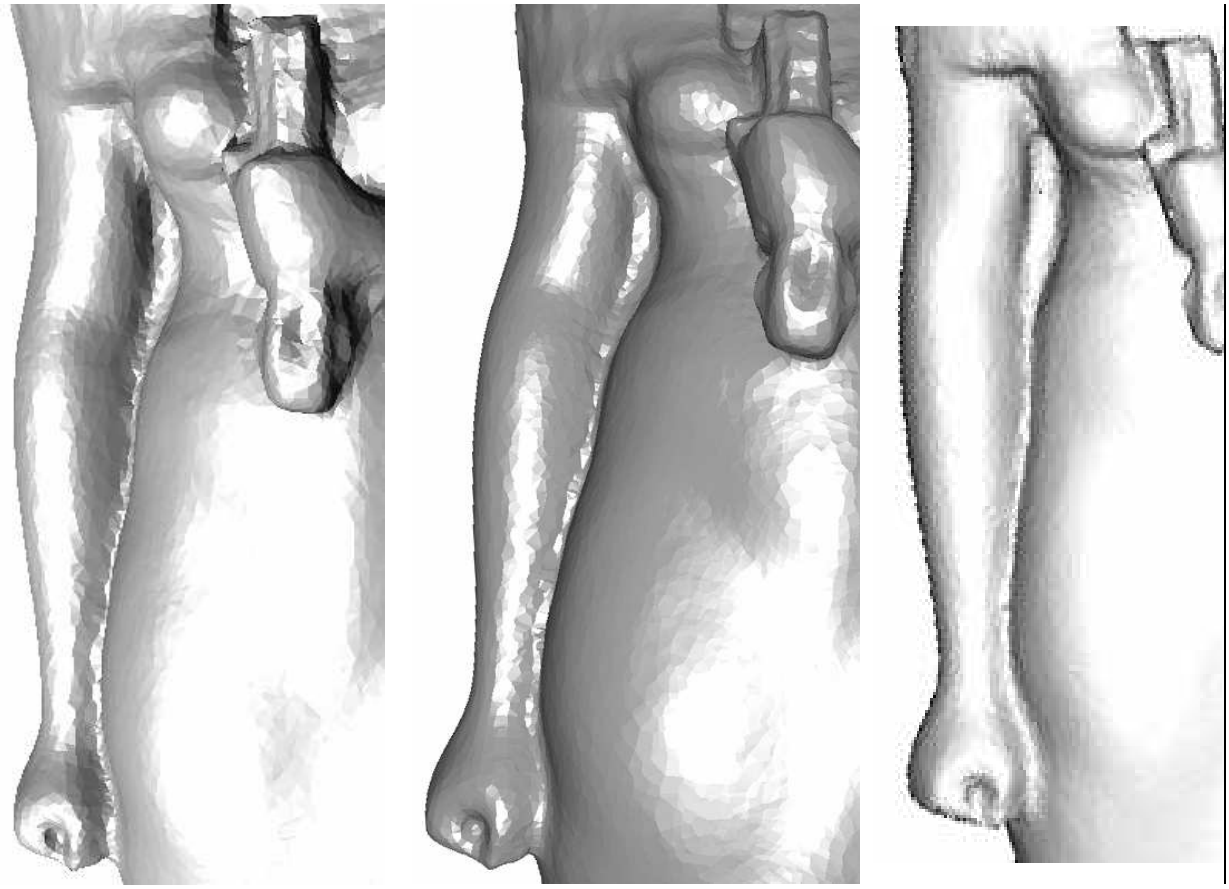
Testovací data Hotdogs z Powercrustu zvládly všechny tři programy výborně. Nedaly se mezi nimi rozeznat rozdíly.

- *Mummy*



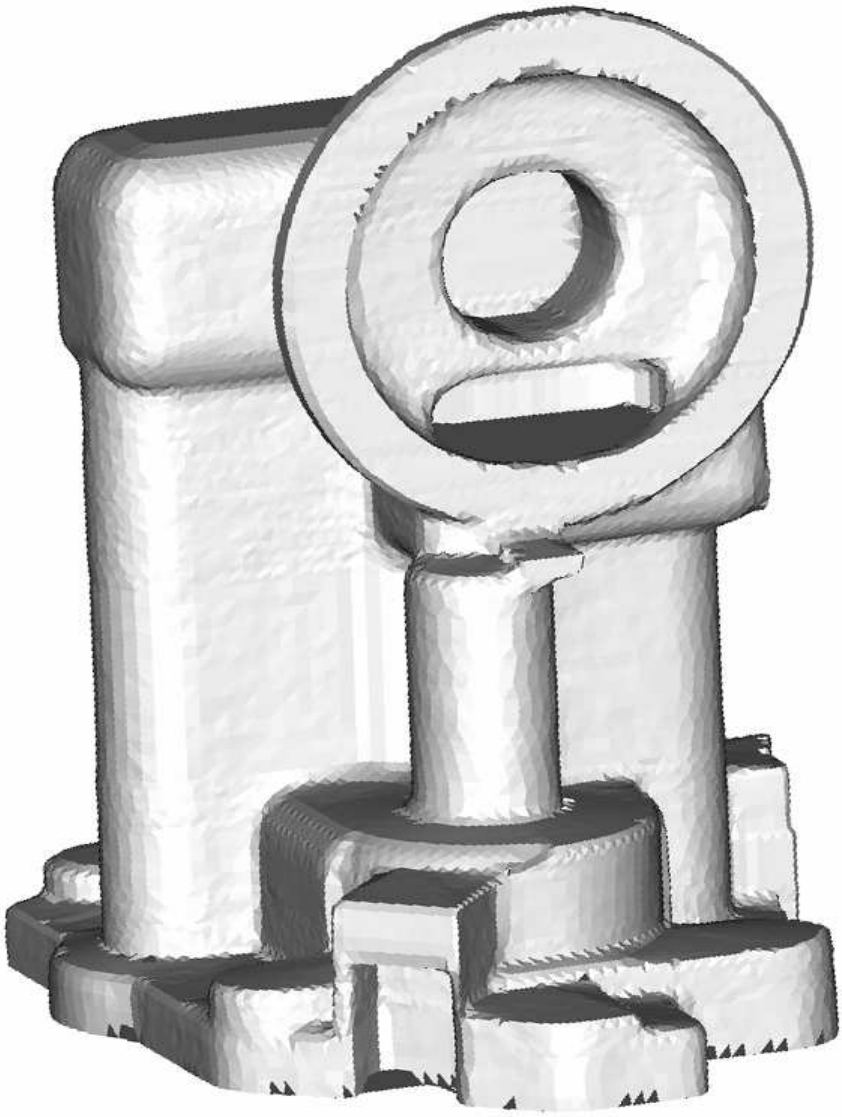
Obr. 6-2-7: ***mummy***

Tento model obsahoval docela vysoký počet bodů. Také ostré hrany mezi rukou a tělem (Obr. 6-2-7a) a v obličeji. Zde měl problémy Cocone, který produkoval příliš „zubaté“ přechody. Powercrust vypadal nejlépe, i když jen těsně před CRUSTem, kterému se na výstupu zobrazilo několik nechtěných trojúhelníků.



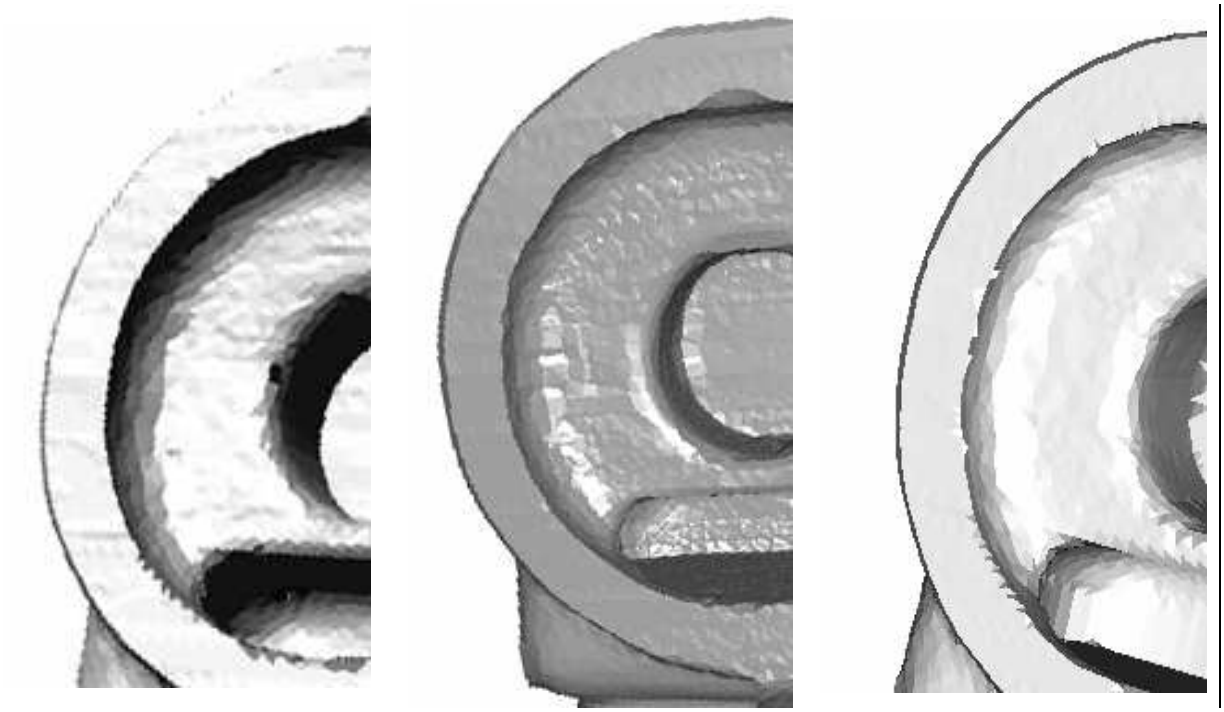
Obr. 6-2-7a: *mummy*, oblast mezi rukou a tělem

- *Oilpump*



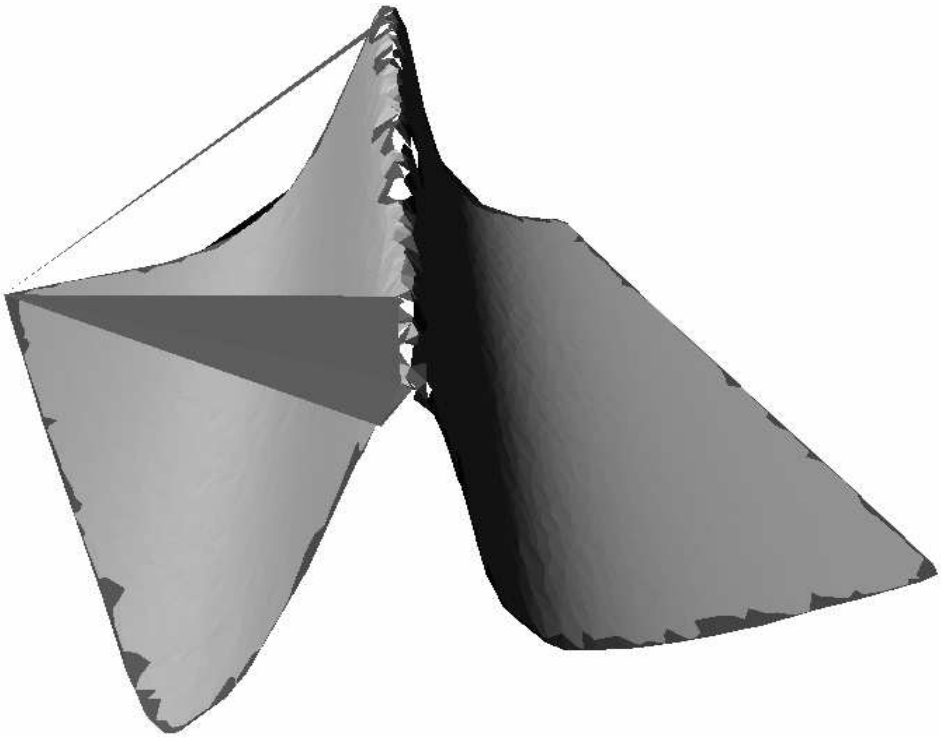
Obr. 6-2-8: ***oilpump***

Na tomto modelu bylo velké množství ostrých hran a pravých úhlů. Powercrust měl nejhezčí hrany a negeneroval žádné díry. Cocone i CRUST měli hrany drsnější (Obr. 6-2-8a), občas se u nich objevily nechtěné trojúhelníky. CRUST měl na tomto modelu nejvíce problémů. Hladké plochy podstav byly v pořádku u všech programů. Powercrust se jevil nejlépe navzdory několika trojúhelníkům s obrácenými normálami. To se projevilo pouze u vystínování modelu podle Gorauda.

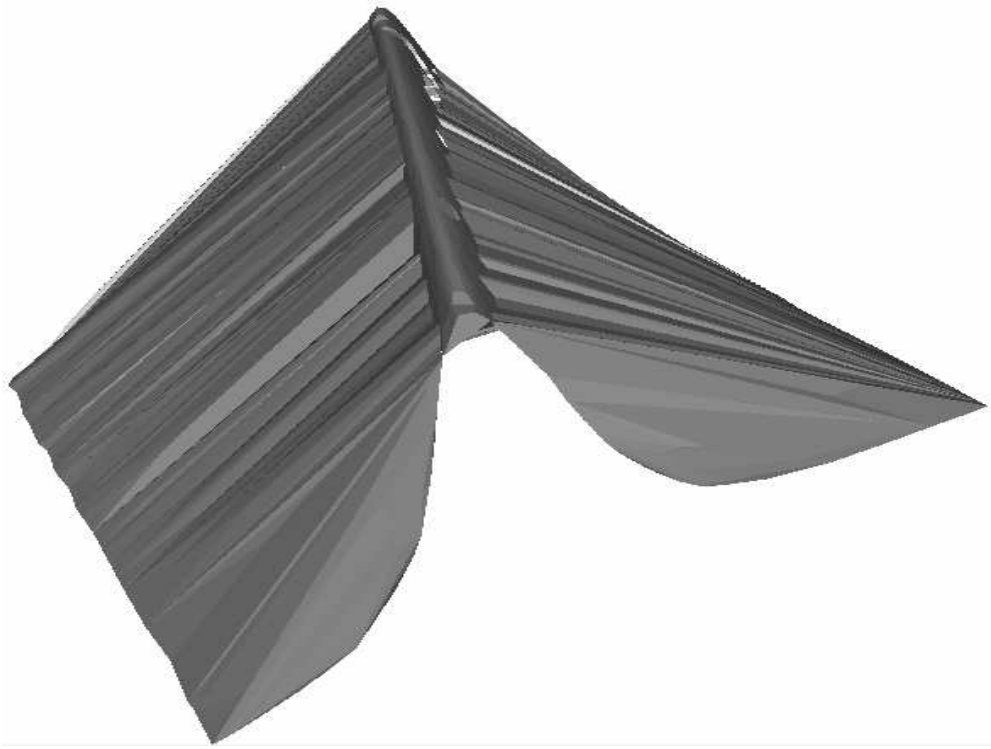


Obr. 6-2-8a: *oilpump*, hrany

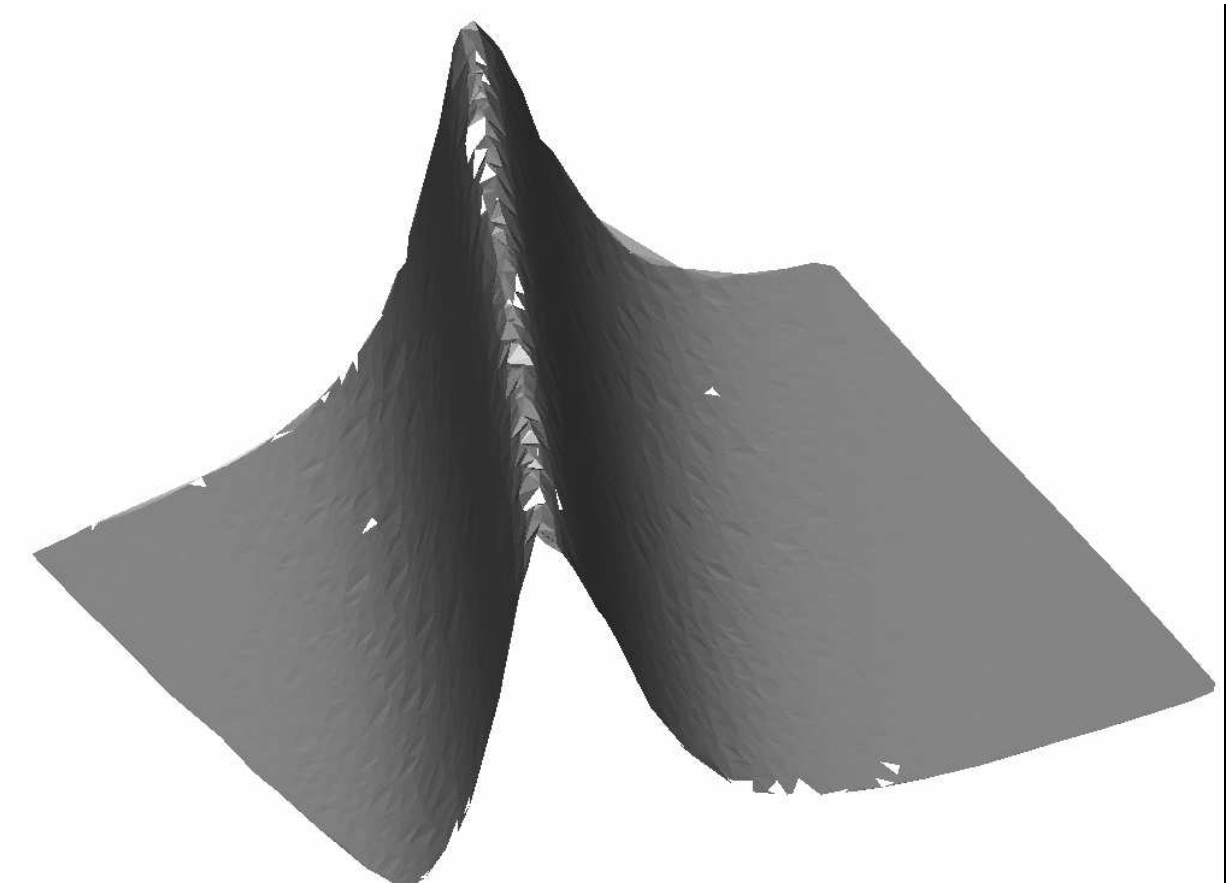
- *Poly*



Obr. 6-2-9a: *poly* z Cocone



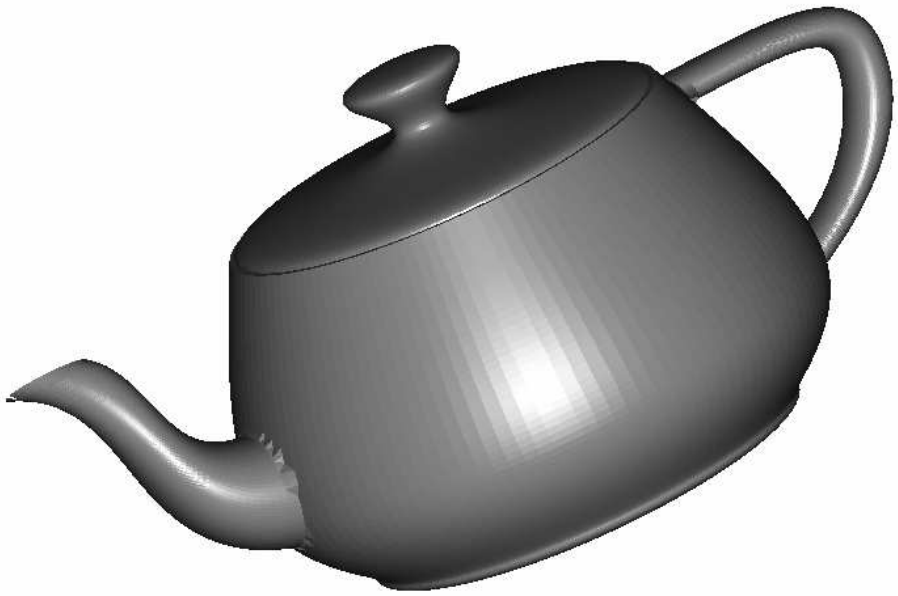
Obr. 6-2-9a: *poly* v podání Powercrustu



Obr. 6-2-9a: *poly* z CRUSTu

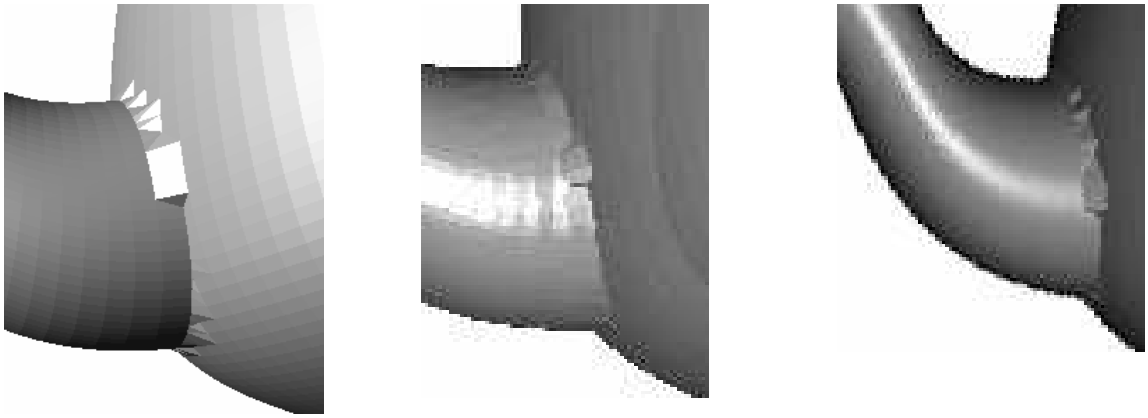
Na těchto datech naprosto ztroskotal Powercrust (6-2-9b), skoro není poznat, jak měla plocha vypadat. Přestože se Cocone (Obr. 6-2-9a) do výsledku přimíchaly nějaké nechtěné trojúhelníky, měl méně děr ve zlomové části i na ploše a tedy rekonstrukce z CRUSTU (Obr. 6-2-9c) vypadá hůře.

- *Teapot*



Obr. 6-2-10: ***teapot***

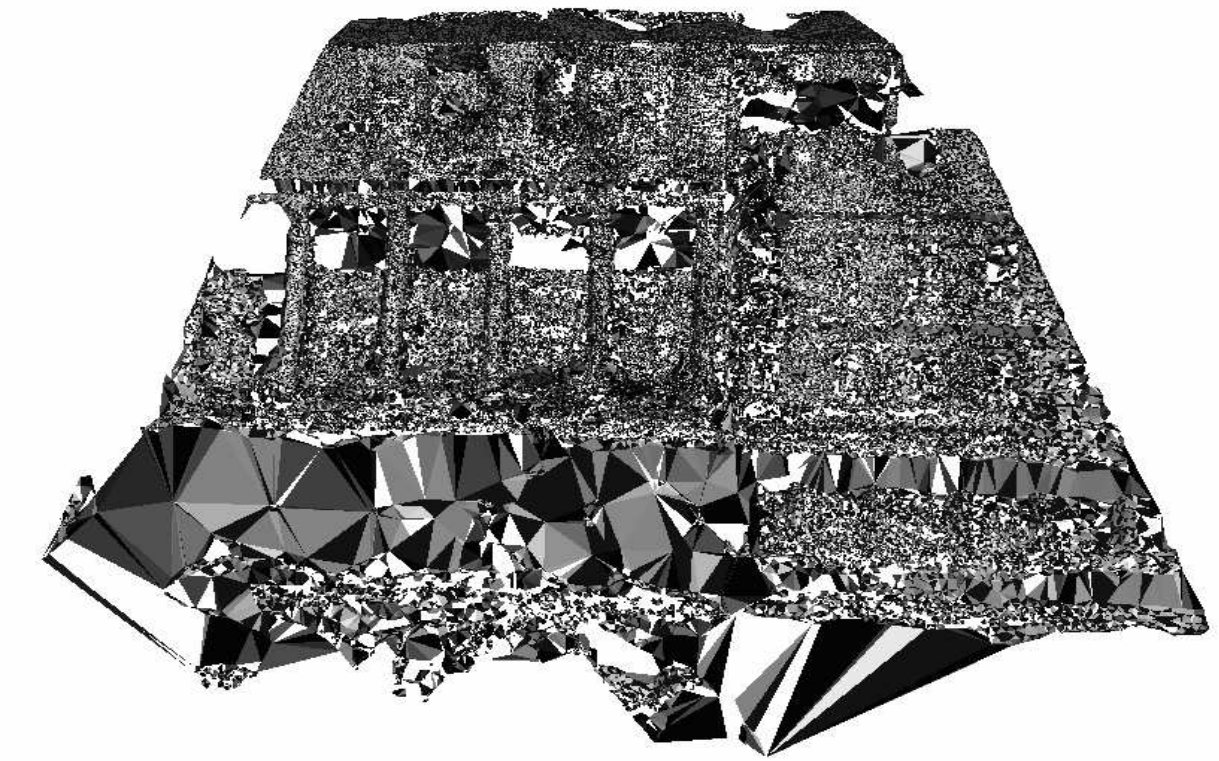
Model konvice dopadl vcelku dobře, na rozhraní mezi tělem a hrdlem (Obr 6-2-10a) nebo držadlem vznikaly nechtěné artefakty více či méně u každého programu. Dvě plochy tam totiž svírají pravý úhel. Powercrust měl nejdrsnější horní hranu u pokličky. Zde se CRUST celkově ukázal jako vítěz.



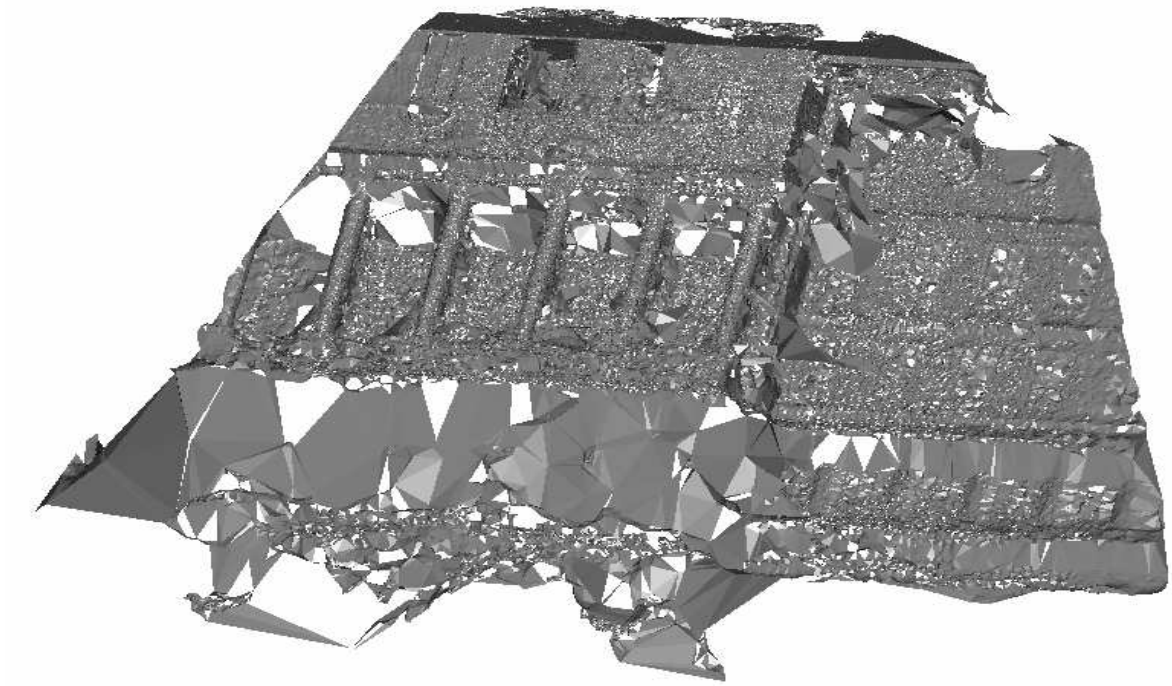
Obr. 6-2-5a: *teapot* kolem hrdla

- *Wien*

Na obrázcích 6-2-11 a 6-2-12 můžeme vidět rekonstruovaná data „Vídeň“. Uvádím je zde jen jako ilustrativní, jedná se o extrémně zašuměná data, se kterými si žádný z testovaných programů neporadí.



Obr 6-2-11: *wien* z Cocone



Obr 6-2-12: *wien* z CRUSTu

7. Závěr

Pokud srovnáváme z hlediska rychlosti rekonstrukce výsledné plochy, jasně vítězí program CRUST. Jednoprůchodová varianta algoritmu se ukázala jako vůbec nejrychlejší. I když použijeme dvouprůchodovou verzi, konkurenční programy jsou pomalejší. Pak ale roste použité množství paměti.

Vizuální srovnávání dopadlo pro CRUST rovněž dobře. Nedá se říci, že kvalita jeho výstupu je horší než u konkurence. Pro každý model je vhodná jiná metoda přístupu a nelze určit jednoznačně nejlepšího.

Hlavní výhodou CRUSTu je hned po rychlosti intuitivní ovládání a široké možnosti nastavení parametrů algoritmů rekonstrukce. Při dostatečné píli s nastavením, pro každý model specifickým, lze dosáhnout kvalitních výsledků. Ze všech programů vůbec prokazoval nejmenší rozptyl v kvalitě výstupu. Program obsahuje také velmi zdařilý interní vizualizátor. Nevýhodou zůstává nemožnost dávkově spouštět úlohy.

Cocone má hlavní výhodu v jednoduchém řádkovém spouštění a ve výstupním formátu – pokud vlastníme VRML prohlížeč, ihned si prohlédneme výsledek. Avšak uživatel nemá žádnou kontrolu nad parametry algoritmu rekonstrukce, musí se spoléhat na hodnoty dodané autorem. V průběhu testování se také ukázalo, že výstupní data nemají shodnou orientaci normál povrchu. Ty mohou mířit dovnitř i ven z tělesa, není dána jejich jednoznačná orientace. Při zobrazování výsledků může uvedená skutečnost působit problémy. Program dosahoval obdobné kvality u všech modelů, neměl vyložené tendence na některém ztroskotat.

Powercrust se ve většině testů ukázal být nejpomalejší. Také proto, že potřebuje externí programy na generování výstupního souboru. Paměťové nároky má větší ve srovnání s Cocone nebo jednoprůchodovou verzí CRUST. Kvalitativně občas selhal – některá data mu prostě „nesedí“. Výhoda spočívá hlavně v přenositelnosti programu na různé platformy, protože autorka poskytuje zdrojový kód psaný v ANSI C.

8. Zdroje

[1] M. Varnuška**,** Rekonstrukce povrchů geometrických objektů z roztroušených bodů. Diplomová práce. Západočeská univerzita. Plzeň 2002.

[2] T. K. Dey and J. Giesen. Detecting undersampling in surface reconstruction. Proc. 17th Ann.Sympos.Comput. Geom., 2001, s. 257-263.

[3] N. Amenta, S. Choi, T. K. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. Proc. 16th Sympos. Comput. Geom., 2000, s. 213-222.

[4] N. Amenta, S. Choi and R. Kolluri. The power crust, unions of balls, and the medial axis transform, Computational Geometry: Theory and Applications, 2001, 19:(2-3), s. 127-153

[URL1] http://www.cis.ohio-state.edu/~tamaldey/

[URL2] http://www.cs.utexas.edu/users/amenta/

[URL3] http://sources.redhat.com/cygwin/

[URL4] http://www.sim.no/

[URL5] http://herakles.zcu.cz/research.php/

[URL6] http://www.public.asu.edu/~rgholmes/graphics/roffview/