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

BAKALÁŘSKÁ PRÁCE

B-Spline Remeshing

Plzeň, 2014 Petr Šroub

Zadání

Sem se má vložit zadání z katedry – s razítkama

1) Seznamte se s problematikou remeshing techniky používané pro automatické nebo semi-automatické nahrazení povrchové trojúhelníkové sítě reprezentující reálný objekt jinou povrchovou trojúhelníkovou sítí s výhodnějšími vlastnostmi (např. tvar a počet trojúhelníků, hladkost povrchu, vlastnost manifoldu).

2) Seznamte se s B-Spline plochami, zejména pak s Coonsovými B-Spline plochami, a dále se seznamte s principy lineární a nelineární regrese a se způsobem využití nelineární regrese pro aproximaci bodů v E3 B-Spline plochou.

3) Prozkoumejte existující přístupy pro remeshing trojúhelníkové sítě prostřednictvím B-Spline ploch, zejména pak prozkoumejte techniky používané pro automatickou segmentaci vstupní sítě na dílčí části vhodné pro nelineární regresi z předchozího bodu.

4) Po dohodě s vedoucím práce si vyberte / navrhněte metodu vhodnou pro B-Spline remeshing trojúhelníkových sítí, které obsahují šum a různé artefakty (zejmnéna překrývající se trojúhelníky, nemanifoldní hrany a vrcholy).

5) Proveďte implementaci metody jako modul VTK.

6) Výsledné řešení důkladně otestujte na datech uměle vytvořených (anuloid, válec, krychle) i reálných (sítě automaticky extrahované z volumetrických dat) a zhodnoťte z pohledu chyby remeshingu, kvality sítě, časové a paměťové náročnosti implementace, aj.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni, dne 6. května 2014 *vlastnoruční podpis*

Poděkování

Děkuji tímto Doc. Ing. Josefovi Kohoutovi, Ph.D., za odborný dohled, konzultace a pomoc při zpracovávání této bakalářské práce.

Dále děkuji mé rodině za podporu během celého studia.

Abstact

ŠROUB, Petr. *B-Spline Remeshing*. Bachelors. Pilsen: Faculty of Applied Science WBU. 45p.

Keywords: Remeshing, Triangle mesh, Parametric surfaces

The subject of this work is „B-Spline Remeshing“. Herein are described different categories of approaches to solving the remeshing problem for triangle meshes depending upon given constraints and requirements. More thoroughly is described one specific established approach to this problem and its adjustment to the given problem – remeshing focused on smoothing and repair of a triangle mesh affected by noise and containing small defects. Furthermore, the actual implementation of this algorithm is described along with why B-Spline surfaces were abandoned in favor of the more simple Bézier surfaces, the results of testing this implementation on both artificially generated objects and meshes representing real objects and the problems that have arisen during this testing. Finally, it is explained how to approach fixing these issues and which direction any further improvement of the program might take.

Abstrakt

ŠROUB, Petr. B-Spline Remeshing. Bakalářská práce. Plzeň: Fakulta aplikovaných věd ZČU. 45 s.

Klíčová slova: Remeshing, Trojúhelníková síť, Parametrické plochy

Tématem práce je „B-Spline Remeshing“. Jsou zde popsány různé kategorie přístupu k problému remeshingu trojúhelníkových sítí podle daných podmínek a požadavku, podrobně popsán jeden z již zavedených přístupů k tomuto problému, a jeho úprava pro daný problém – remeshing soustředěný na vyhlazení a opravu sítě zatížené šumem a obsahující drobné závady. Dále je zde popsána konkrétní implementace tohoto algoritmu, proč bylo od B-Spline ploch upuštěno ve prospěch Beziérovo ploch, výsledky otestování této implementace na uměle vytvořených i reálných datech a problémy, ke kterým došlo během testování. Na závěr je uvedeno, jak by se postupovalo pro opravu těchto problémů a kterým směrem lze postupovat pro možné budoucí vylepšení práce.

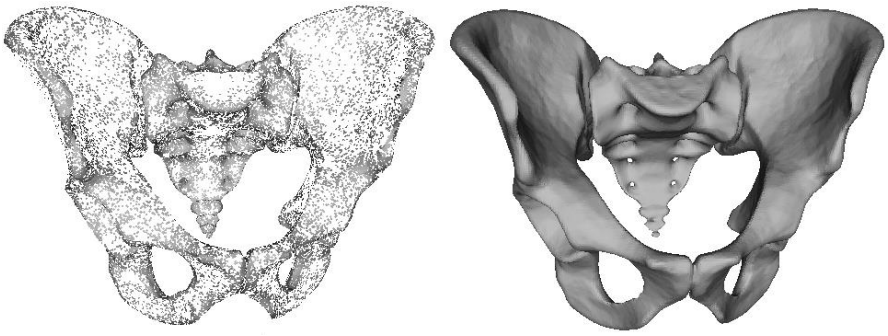
Obsah

1 Úvod

Definice problému

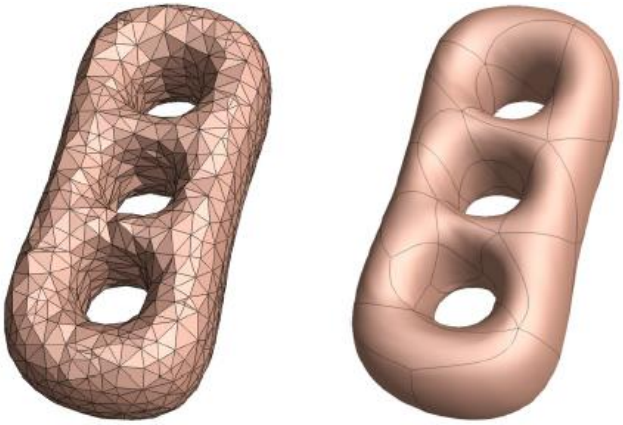
Při digitální reprezentaci reálných objektů získané například 3D scannerem nelze počítat s absolutně přesnými, bezchybnými výsledky. Výsledná data jsou zaprvé zatížena chybou skenovací metody (šumem), a dále dochází ke ztrátě topologických vlastností (obvyklým výsledkem je hustý oblak bodů bez jejich vzájemných vztahů).

Pro rekonstrukci povrchové sítě z těchto bodů existuje mnoho různých metod (například [1]), ale výsledné sítě často obsahují drobné povrchové závady a topologické nepřesnosti, které mohou v algoritmech předpokládající bezchybný povrch způsobovat problémy, nebo v horších případech mohou být i viditelné lidským okem.



Obr. 1: Rekonstrukce povrchové trojúhelníkové sítě z neorganizovaných bodů [1]

Účelem této práce je převedení reprezentace povrchové trojúhelníkové sítě, ve které existují drobné defekty a artefakty na parametrické plochy tak, aby trojúhelníková síť daná propojením bodů na těchto plochách byla kvalitnějšího charakteru. Taková síť má pak už lepší vlastnosti (kompatibilní s algoritmy předpokládající uzavřený objekt bez defektů, hladkost povrchu, samotná kvalita).



Obr. 2: Trojúhelníková síť a na ní založená kolekce parametrických B-Spline ploch [2]

2 Předchozí řešení

Na téma remeshingu trojrozměrných dat již existuje mnoho předchozích prací, ze kterých lze čerpat. Přístup k problému se u každého řešení různí jak podle předpokladů dané práce, tak požadavků na vstupní síť i na různé vlastnosti sítě výstupní.

[3] dělí remeshing přístupy do pěti hlavních kategorií:

1) Structured: Nahrazuje vstupní síť takovou sítí, ve které má (skoro) každý bod stejný počet sousedů. Díky této vlastnosti lze pak použít různé optimalizované algoritmy, které pro obvyklou neregulární síť nelze jednoduše použít.
2) Compatible: Vytváří ze vstupních sítí takové jim odpovídající sítě, které mají stejnou strukturu a respektují původní tvar sítí. Tyto sítě lze pak použít pro hladký převod jedné vstupní sítě do druhé.
3) High quality: Tyto přístupy se soustředí na vytvoření co nejkvalitnější výstupní sítě vzhledem k rozmístění bodů a poměru stran trojúhelníků.
4) Feature: Jako vstupní předpoklad je dáno, že trojúhelníková síť je diskrétní reprezentací nějakého parametrického povrchu nebo kolekce více parametrických povrchů. Výsledkem těchto postupů jsou tedy aproximace těchto parametrických povrchů.
5) Error driven: Tyto přístupy se soustředí na minimalizaci chyby polohy mezi původní sítí a její aproximací pomocí parametrických ploch.

Další důležitý rozdíl mezi přístupy je množství automatizace algoritmu. Existují tři hlavní kategorie:

1) Plně manuální: Ponechává rozhodnutí o tom, kde a jak trojúhelníkovou síť rozdělit, plně na uživateli programu. Algoritmus pouze použije zadaná rozdělení bez jakékoli snahy o jejich úpravu.
2) Částečně automatizované: Uživatelem zadané počáteční body/dělící hrany jsou použity jako výchozí bod programu, ale v případě potřeby je program může přesunout, odstranit, nebo rozšířit o další.
3) Plně automatizované: Program svou funkci plní plně automaticky bez jakéhokoli vstupu uživatele, maximálně s výjimkou výstupních požadavků (např. maximální chyby nebo hustoty výsledných parametrických ploch).

Další způsob, jak rozlišit odlišné přístupy, je podle jejich požadavků. Některé z požadavků, na které se daný přístup může soustředit:

- Časová náročnost, ať už z důvodu časného zpracování i pro sítě s velkým počtem bodů, nebo naopak zpracování relativně malých sítí v reálném čase.
- Paměťová náročnost, kritická hlavně u starších algoritmů, ale důležitá i v moderní době
- Paměťová kapacita potřebná k uložení výsledku algoritmu
- G^0 kontinuita
- G^1 kontinuita
- Chyba výsledku vzhledem k objemu objektu

- Chyba výsledku vzhledem k tvaru objektu
- Chyba výsledku vzhledem k zachování ostrých hran
- Hladkost výsledného objektu
- Eliminace šumu na vstupu
- Eliminace nemanifoldních vrcholů/hran

Většina přístupů se bude soustředit hned na více než jeden z těchto požadavků – nelze ovšem plně splnit všechny požadavky (například: čím přesněji aproximuje výsledek počáteční síť, tím více místa je třeba k jeho uložení a tím více se na něm projeví vstupní šum).

Z požadavků práce tedy plyne, že hledaný přístup spadá do kategorie error-driven a musí být plně automatizovaný. Hlavními cíli je eliminace šumu a chyb, a dále vyhlazení při co nejlepším zachování tvaru a objemu. K tomu se nejvíce blíží postup popsaný v [2]. Ten popisuje automatizovaný algoritmus pro nahrazení trojúhelníkové sítě B-Spline plochami, který se s určitými rozdíly (místo počáteční trojúhelníkové sítě předpokládá oblak bodů, je také zaměřen na minimalizaci velikosti výstupních dat) podobá potřebnému. Jeho konstrukce je také odolná vůči chybovým hranám a vrcholům. Nejdříve je tedy třeba popsat tento výchozí algoritmus, a poté na něm provedené úpravy.

3 Algoritmus podle Eck et al.

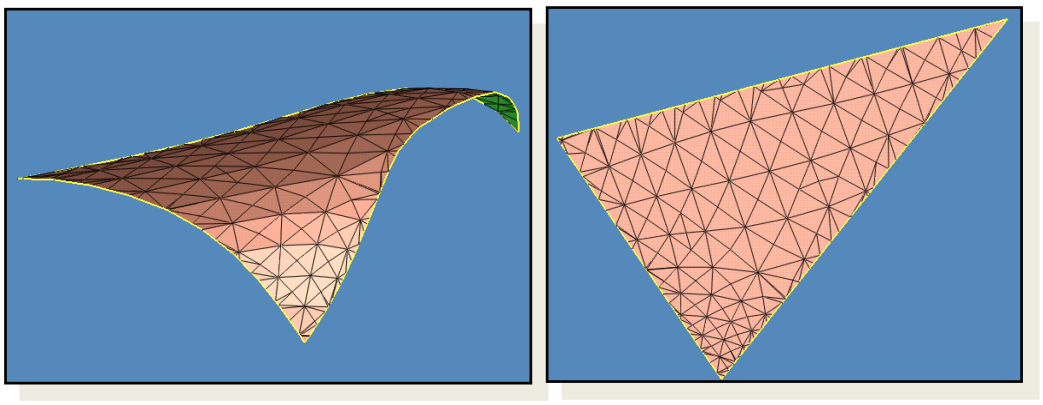
Algoritmus postupuje tímto způsobem:

1. Získání počáteční trojúhelníkové sítě.
2. Nalezení pseudo-Voroného buněk a jejich centrálních bodů.
3. Použití těchto buněk a bodů k získání trojúhelníkových oblastí.
4. Spojení těchto oblastí po dvojicích pro získání čtyřhranných oblastí, které lze nahradit B-spline plochami.
5. Nahrazení oblastí B-spline plochami s minimální možnou chybou vůči původní síti při zachování podmínek spojitosti G^0 a G^1.
6. Zvyšování úrovně detailů pro ty plochy, kde je chyba od původních dat příliš velká.
7. Uložení výstupních dat.

Nejprve je ale třeba zmínit, jak algoritmus využívá harmonických map.

3.1 Harmonické mapy

Algoritmus Eck et al [2] využívá takzvané harmonické mapy (ve skutečnosti jejich definici přímo neodpovídají, ale v textu jsou tak pojmenovány pro ušetření místa – použitá parametrizace je unikátní minimalizace po částech definované lineární funkce, která danou harmonickou mapu pouze dobře aproximuje). Jejich použití je v tomto případě analogické například UV mapám používaných v 3D grafických aplikacích – každému bodu, spadajícímu do dané mapy (obr. 3-a), přiřadí unikátní 2D souřadnice (obr. 3-b) tak, aby okrajové body ležely na hranách polygonu tvořeného rohovými body oblasti rozmístěnými na jednotkové kružnici, a také tak, aby došlo k co nejmenšímu zkreslení tvarů trojúhelníků na síti.



Obr. 3: a) 3D oblast b) Odpovídající harmonická mapa [1]

Toto přiřazení není triviální. Neexistuje žádná projekce, která by nevytvářela žádnou deformační chybu. Tudíž je nutné najít takovou parametrizaci, kde je tato chyba minimalizována. K tomu je použita metoda nejmenších čtverců, kde se jako počáteční

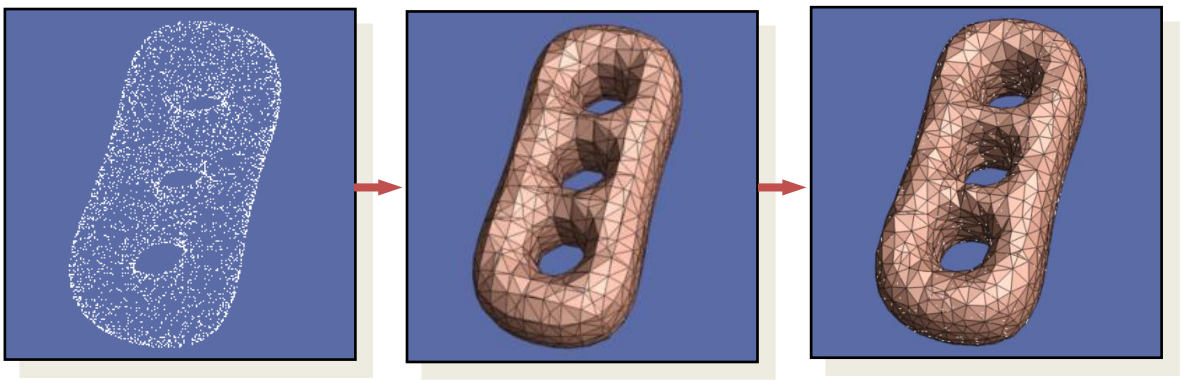
aproximace dosadí pouze okrajové body (u kterých je jejich UV pozice známa), a jako podmínky se nastaví zachování pozice okrajových bodů a minimalizace změny délky hran a velikosti úhlů mezi hranami.

3.2 Odolnost vůči chybám

Protože algoritmus je už přímo určen pro vylepšení šumem zatížené sítě, je proti jeho vlivu odolný. Plochy, které nakonec vytvoří, nejsou přímo závislé na pozicích jednotlivých bodů. Překrývající trojúhelníky se buď ztratí jako součást Voronových oblastí, nebo na jejich místě maximálně vznikne jejich okraj. V obou případech tato situace není problematická. Nemanifoldní hrany, pokud na ně nenarazí v průběhu generace čar, algoritmus nijak neovlivní. I v případě, že na ně generátor čar narazí, bude důsledkem pouze čára horší kvality. Nemanifoldní vrcholy se buď jednoduše stanou součástí regionu na jedné straně sítě, nebo v nejhorším případě povedou k vzniku otvoru ve výsledném objektu.

3.3 Trojúhelníková síť

Na rozdíl od popisovaného algoritmu tato práce na vstupu již předpokládá existující síť. Z tohoto důvodu není tato část algoritmu z pohledu této práce zajímavá, proto jen stručně. Eck et al [2] používá pro generaci počáteční sítě algoritmus podle H. Hoppe [4], jehož výsledek je pak vylepšen pomocí dalšího algoritmu stejných autorů [5]. Výsledná síť pak tvoří dobrou aproximaci povrchu objektu reprezentovaného oblakem bodů.



*Obr. 4: Převod oblaku bodů na počáteční síť podle [*1]

3.4 Pseudo-Voroného buňky

Protože jedna B-Spline plocha je logicky schopna nahradit pouze plošné objekty, je síť nutné rozdělit na nahraditelné oblasti. Rozhodnout se jaký způsob rozdělení použít není

triviální, a protože jedním z požadavků je plně automatické zpracování, nelze využít uživatelského vstupu. Algoritmus tedy musí o tom, jak oblasti rozmístit, rozhodnout sám. Eck et al [2] využívá postupu, který na síti (obr. 5-a) rozmístí seedy, které jsou pak ohraničeny buňkami, ve kterých každý bod má po této síti nejkratší vzdálenost k danému seedu (obr. 5-b). Protože se myšlenka těchto buněk blíží prostorovým Voronovým oblastím, Eck et al [2] je označuje jako pseudo-Voroného buňky.

Algoritmus pro rozdělení trojúhelníkové sítě do pseudo-Voroného oblastí popsaný v [2] má ale závadu v tom, že pro některé sítě může selhat. (V případě, že třetí z níže uvedených podmínek je pro nějaký bod nesplněna, ale všechny jeho okolní body už jsou seedy. Algoritmus pak její nesplnění nemůže opravit, takže uvázne.) Protože je algoritmus popsaný A. Kleinem et al v [9] přímo učen pro jeho náhradu, je v této části textu vhodné ho rovnou popsat místo původního. Rozdílem mezi nimi je využití bodů (vertex) místo stěn (face). Kromě tohoto rozdílu je princip algoritmu identický a postup velmi podobný. Oba algoritmy jsou variantou Dijkstrovo algoritmu pro nejkratší cestu s více cílovými body.

Síť je nutné bod od bodu projít a všechny body přiřadit nějaké buňce. Tyto oblasti je ale nejprve třeba vytvořit, což nastávající algoritmus řeší iteračním přístupem: začít s oblastí jednou, a k té body přidávat od nejbližšího tak dlouho, dokud není porušeno jedno z níže uvedených pravidel. Jakmile dojde k porušení, bod, který ho způsobil, je přidán jako střed nové oblasti, a začne se znovu.

Na začátku se buď náhodně vybere jeden z bodů jako seed (v případě sítě bez okrajů), nebo tři body rovnoměrně rozložené po každém okraji jako seedy, a vloží se do prioritní fronty F, ve které jsou body seřazené podle vzdálenosti od nejbližšího seedu. Pak:

1) Z fronty F se vybere horní bod, značený *v*.
2) Pokud by přidání bodu *v* do oblasti T, které seed je *v* nejbližší, porušilo podmínku a) nebo b), označíme *v* jako seed, vyprázdníme frontu F, a vrátíme se na začátek.
3) Jinak *v* přidáme do T a všechny jeho sousedy, které ještě nespadají do žádné oblasti, přidáme do F, přičemž jejich vzdálenost nastavíme jako vzdálenost *v* + Eukleidovskou vzdálenost *v* a přidávaného bodu. Pokud již je ve frontě, pouze aktualizujeme jeho prioritu, pokud je nová vyšší než jeho původní. Potom přejdeme zpět do kroku 1)
4) Jakmile je F vyprázdněná, zkontrolujeme podmínky d) a e). Pokud je nějaká podmínka nesplněna, nalezneme bod, který má být použit jako seed, vložíme ho a všechny existující body do nové prioritní fronty, a vrátíme se do kroku 1). Jinak algoritmus skončí.

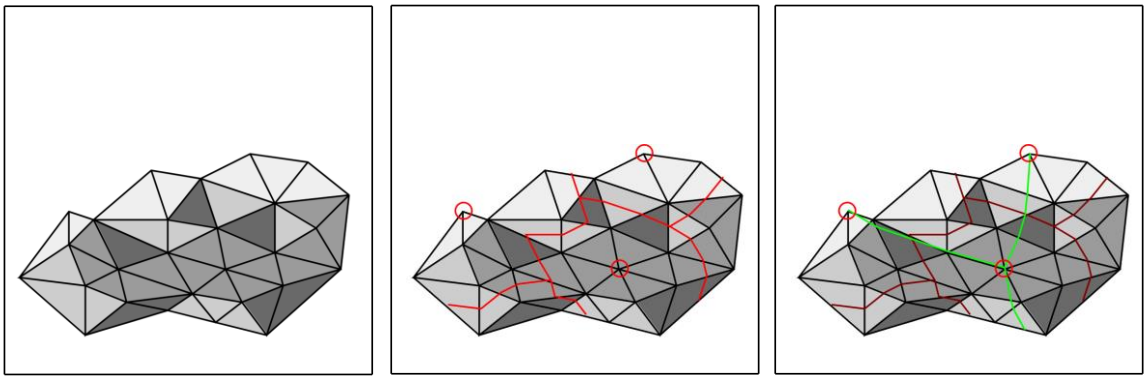
Podmínky jsou definované takto:

a) Protože cílem je vytvoření oblastí zobrazitelných na rovinu, každá oblast musí být homomorfická na disk.

b) Všechny body na okrajích (pokud existují) musí spadat do oblasti, jejíž seed je také na okraji.
c) Ne víc jak 3 oblasti se mohou setkat v jednom bodě uvnitř sítě, a ne víc jak 2 oblasti se mohou setkat v bodě na okraji. Tato podmínka je algoritmem z [9] splněna automaticky.
d) Protože středy oblastí jsou dále využity k tvorbě sítě tvořené trojúhelníky, žádné dvě oblasti nesmějí sdílet více jak jeden řez.
e) Délka dvou následujících řezů v jedné oblasti musí být více než 10% celkové délky okraje.

Ověření podmínek probíhá následovně:

a) Všechny body z okraje oblasti T, které leží vedle bodu *v*, musí být kontiguitní, jinak není podmínka splněna.
b) Prostá kontrola, jestli body v okolí *v* nebo *v* samotné leží na okraji a nesplňují podmínku.
c) Automaticky splněno.
d) Pokud existuje více rohů (body oblasti, které leží vedle bodů obsažených ve dvou oblastech) než přilehlých oblastí, tak je podmínka nesplněna.
e) Lze zkontrolovat jednoduchým součtem délky všech hran na okraji oblasti a porovnáním jejich délek vůči délkám řezů.



Obr. 5: a) Počáteční trojúhelníková síť
b) Na ní vygenerované seedy a pseudo-Voroného oblasti
c) Trojúhelníkové oblasti tvořené propojením seedů (viz dále)

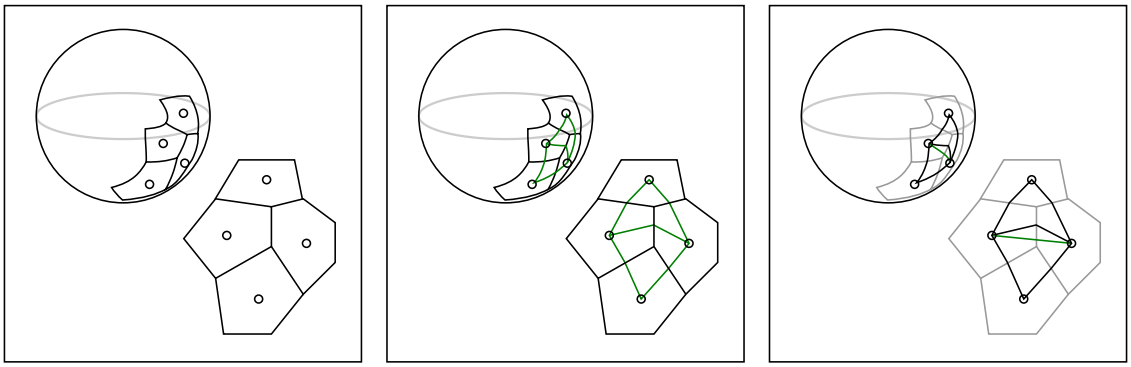
3.5 Trojúhelníkové oblasti

Výsledkem předchozího kroku je rozdělení sítě na buňky, jejichž středy po propojení do trojúhelníkových oblastí (obr 5-c) tvoří velmi zjednodušenou verzi původní sítě s tou vlastností, že každá jednotlivá oblast je homomorfní na disk (a tedy zobrazitelná na rovinu). Zjevný postup propojení těchto bodů nejkratší cestou ale nevede k dobrým výsledkům, protože tyto cesty se mohou křížit. Navíc samotné nalezení této cesty není

triviální. Je tedy třeba získat propojení takové, které se těmto problémům vyhne. K tomu lze využít harmonické mapy.

Každou buňku z předchozího kroku je třeba zobrazit na harmonickou mapu, ve které jsou rohy tvořeny body, ve kterých se setkávají více než dvě oblasti (obr. 6-a). První aproximace trojúhelníkových oblastí pak vznikne propojením středu hrany mezi dvěma oblastmi a středovým bodem rovnou čarou pro každou dvojici hrana-střed na odpovídající mapě (obr. 6-b). V případě oblastí na okraji sítě je místo středu použit středový bod hrany na okraji. Protože tyto čáry obecně nespadají na hrany trojúhelníkové sítě, je třeba o ně síť rozšířit. To počet trojúhelníků zvýší pouze o druhou odmocninu jejich počátečního počtu.

Na takto získaných cestách ale vzniká nekvalitní přechod v místě přechodu z jedné oblasti do druhé (vzniká viditelný ostrý úhel), takže je třeba je vyrovnat. Proto je pro každou hranu mezi body vytvořena další harmonická mapa (tvořena dvěma oblastmi hraně náležící), a na té je vytvořena nová, tentokrát už rovná cesta (obr. 6-c).



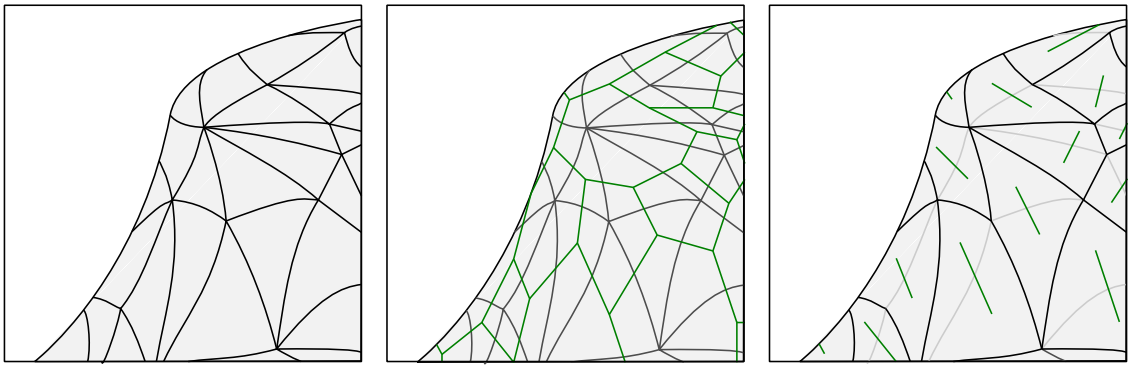
Obr. 6: a) pseudo-Voroného oblasti na síti a jim odpovídající harmonické mapy
b) První propojení středů oblastí v mapách a výsledek zobrazený na síť
c) Zarovnání jedné z těchto cest pomocí mapy tvořené cestami z prvního propojení

3.6 Propojení oblastí

Trojúhelníková síť je nyní rozdělena do trojúhelníkových oblastí (obr. 7-a) s okraji rovnými vzhledem k harmonické mapě, na kterou je daná oblast zobrazitelná. Protože ale B-spline plochy pracují se čtyřúhelníkovými oblastmi, je třeba ještě tyto oblasti vhodně upravit. Jednoduchým řešením tohoto problému je každou oblast rozdělit na čtyři čtyřúhelníky vložením nových bodů do středů okrajů + středového bodů stěny. K tomuto postupu se lze vrátit, pokud lepší řešení selže.

Dalším přístupem, který vytvoří šestkrát méně čtyřúhelníků, je oblasti spojit do dvojic. Postup je jednoduchý – lze definovat graf, ve kterém jsou vrcholy oblasti, hrany jsou ty okraje, které oblasti sdílí, přičemž jejich cena je chyba vzniklá nahrazením dané dvojice oblastí harmonickou mapou (obr. 7-b). Potom je problém jednoduchý – nalézt takové seskupení, ve kterém je každá oblast v právě jedné dvojici tvořené sousedy. Takových řešení je mnoho, proto jako další podmínka může být zaveden požadavek na co

nejmenší chybu aproximace těmito dvojicemi vzniklou. Heuristickým přístupem je pro každý možný pár přes harmonické mapy spočítat odchylku, což problém převede na MAX-MIN MATCHING problém. Řešením je takový shluk kombinací, pro který je maximální globální chyba minimální (obr. 7-c).

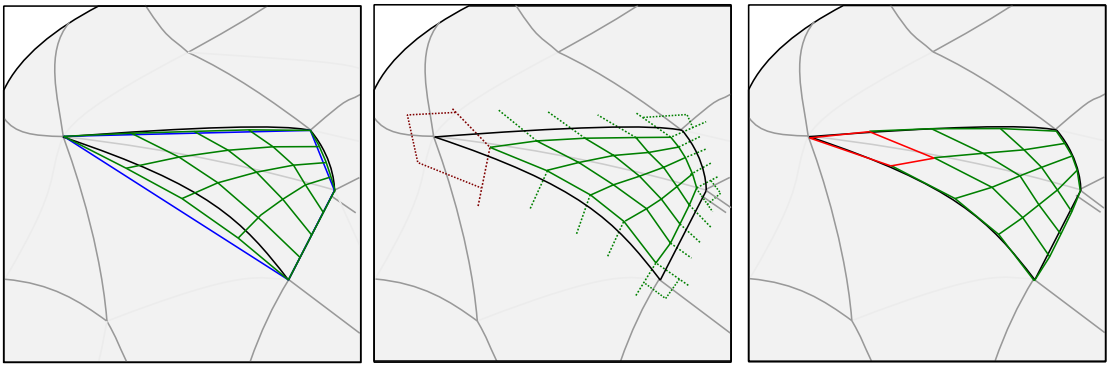


Obr. 7: a) Trojúhelníkové oblasti na trojúhelníkové síti
b) Převedení na grafový problém
c) Propojení trojúhelníkových oblastí do čtyřúhelníkových

3.7 B-Spline plochy

Problém nahrazení jedné oblasti je celkem jednoduchý – okraje B-spline plochy se umístí na okraj oblasti a vnitřní body se promítnou na jí odpovídající harmonickou mapu, z té zpět na původní trojúhelníkovou síť, a rozdíl mezi touto pozicí a pozicí na ploše určuje chyby, které jsou pak minimalizovány metodou nejmenších čtverců. Problém se stává složitějším po zavedení více jak jedné oblasti, a tudíž požadavku návaznosti v G^0 (zajišťuje spojitý přechod z jedné oblasti do druhé) a G^1 (zajišťuje plynulý přechod). Eck et al [2] tento problém řeší úpravou metody P. Jorga [10].

Čtyřúhelníková síť oblastí je nejdříve dvakrát subdivizována metodou Doo-Sabin [4]. Tím se každá oblast rozdělí na 4 x 4 podoblasti (obr. 8-a), ve kterých pouze rohové mohou obsahovat body jiného stupně než čtyři (tedy takové, kde se setkává jiný počet oblastí než právě čtyři), každý roh navíc obsahuje právě jeden, a tyto případy jsou navzájem izolované (obr. 8-b). Pak je pro okolí každého bodu vytvořena Beziérova plocha – pro rohové body bikubická, pro ostatní bikvadratická (obr. 8-c). P. Allez et al [3] dokazuje, že vzniklé plochy jsou navzájem G^1 spojité.



Obr. 8: a) Nahrazovaná čtyřúhelníková oblast (modře), subdivize (zeleně)
b) Síť tvořená kontrolními body ploch (červeně zvýrazněn speciální případ rohu)
c) Beziérovy plochy (červeně zvýrazněna bikvadratická) / B-Spline plocha

Beziérovy plochy jsou pak převedeny na B-Spline plochy. Každou vzniklou 4 x 4 mřížku oblastí lze zkombinovat do jediné B-Spline plochy, protože jejich kontrolní body odpovídají kontrolním bodům odpovídající B-Spline plochy.

Výsledné B-Spline plochy je pak ještě třeba přiblížit vstupním datům. Toho je dosaženo promítnutím bodů na ploše na odpovídající harmonickou mapu, a z té zpět na vstupní trojúhelníkovou síť. Vzdálenost takto nalezeného bodu a odpovídajícího bodu na ploše je pak použita jako chyba k minimalizování v metodě nejmenších čtverců.

3.8 Úrovně detailů

Plochy, které vychází z předchozího kroku, dosahují minimální chyby pro úroveň volnosti, kterou dovoluje jejich velikost. V případě, že je přesnost aproximace nedostatečná, lze zvýšit úroveň detailů jejich rozdělením a přepočítáním. K tomu existují dva přístupy:

- Globálně rozdělit všechny plochy na čtyři podplochy (subdivize). Takto nevznikne žádný problém a krok 5 může znovu začít ihned. Výhodou tohoto přístupu je jednoduchost implementace a zachování uniformního rozložení detailů.
- Zvýšit úroveň přesnosti lokálně, rozdělením pouze těch oblastí, ve kterých vznikl problém nedostatečné přesnosti. Tím ale vznikne problém na okrajích, kde se setkávají dvě oblasti různého stupně přesnosti. Proto je subdivizi třeba šířit dále do okolí. Tím lze zvýšení detailů držet na okolí oblasti, kde je třeba. V nejhorším případě tento přístup dopadne jako předchozí. Výhodou tohoto přístupu je použití jen takové přesnosti, jaké je lokálně třeba. Algoritmus podle Eck et al [2] používá tento přístup.

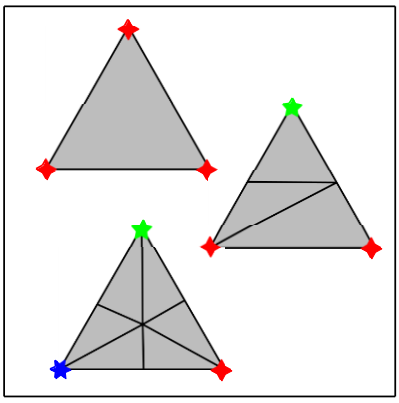
3.9 Uložení dat

Po skončení algoritmu je možné B-Spline plochy převést na novou trojúhelníkovou síť. To je celkem triviální problém dosazení rovnoměrně rozložených bodů do ploch a jejich propojení hranami. Výsledek je pak uložen do souboru v nějakém ze standardizovaných formátů pro 3D data. Další možností umožňující vytvoření sítě podle potřebné hustoty je uložení přímo kontrolních bodů B-spline ploch.

4 Upravený algoritmus

Na rozdíl od výše popsaného algoritmu má tato práce částečně odlišné předpoklady. Protože cílem není získání co nejmenší (z pohledu potřebné paměti) hladké aproximace vstupních bodů, ale naopak pouze vyhlazení a vylepšení sítě již existující, je třeba v algoritmu provést částečné úpravy. Nejprve pro zopakování shrnutí původního algoritmu:

1. Získání počáteční trojúhelníkové sítě.
2. Nalezení pseudo-Voroného buněk a jejich centrálních bodů.
3. Použití těchto buněk a bodů k získání trojúhelníkových oblastí.
4. Spojení těchto oblastí po dvojicích pro získání čtyřhranných oblastí, které lze nahradit B-spline plochami.
5. Nahrazení oblastí B-spline plochami s minimální možnou chybou vůči původní síti při zachování podmínek spojitosti G^0 a G^1.
6. Zvyšování úrovně detailů pro ty plochy, kde je chyba od původních dat příliš velká.
7. Uložení výstupních dat.



Obr. 9: Možné stavy trojúhelníka a jejich výsledné zpracování

4.1 Získání počáteční trojúhelníkové sítě

Tento krok je vynechán úplně, protože počáteční trojúhelníková síť je přítomná od začátku.

4.2 Nalezení pseudo-Voroného buněk a jejich centrálních bodů

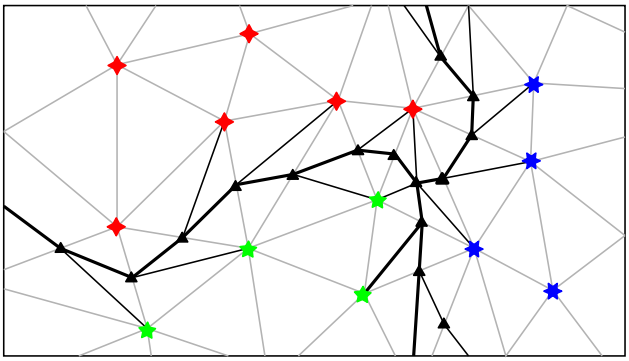
Zde je kvůli odlišným požadavkům vhodné místo snahy o co nejmenší množství výstupních oblastí snažit se o jejich rovnoměrné rozmístění po síti. Navíc není požadována podpora sítí s okraji, tedy sítí obsahující otvory. Z toho plyne následující:

- Podmínka 2 (všechny body na okraji musí spadat do oblasti, jejíž seed je také na okraji) je vynechána.
- Místo inicializace fronty pouze jedním náhodným seedem je fronta na začátku naplněna body rovnoměrně rozmístěnými po síti (a to tak, že je spočtena celková velikost sítě, a pak je do fronty postupně přidáván každý bod, který je od všech ostatních vzdálen více jak daný zlomek této velikosti).

Z [2] dále plyne nutnost úpravy okraje oblasti. Protože každý trojúhelník sítě může logicky spadat do maximálně tří oblastí (podle toho, do kterých oblastí spadají jeho vrcholy, obr. 9), toto je celkem triviální problém, řešený následovně:

Pro každý trojúhelník sítě zkontroluj oblasti, do kterých spadají jeho vrcholy.

- Pokud jsou všechny ve stejné oblasti, ponech trojúhelník ve stavu, ve kterém je.
- Pokud je jeden vrchol v jedné oblasti a dva v jiné, vytvoř dva nové vrcholy na stranách mezi těmito dvěma vrcholy a vrcholem odlišným, přiřaď je do obou oblastí, a rozděl trojúhelník na tři nové tak, aby nový trojúhelník obsahující oba nové vrcholy ležel u odlišného vrcholu.
- Pokud jsou všechny vrcholy v jiné oblasti, vytvoř tři nové vrcholy ležící ve středech hran, přiřaď je do jim náležících oblastí, a přidej další vrchol umístěný do středu trojúhelníka. Ten přiřaď do všech tří oblastí. Pak vytvoř šest nových trojúhelníků, z nichž každý propojuje jeden z původních vrcholů, jeden z hranových vrcholů a středový vrchol.



Obr. 10: Nově vzniklé hrany a vrcholy na okrajích oblastí

Na konci je pak nutno sloučit vzniklé body. Ideálním, ale složitějším řešením, by bylo pro každý další trojúhelník použít odpovídající body vzniklé úpravou předchozích, což je ale implementačně složitější. Vzhledem k relativně malému počtu bodů v jednotlivých oblastech je naivní řešení postačující.

Další úpravou tohoto kroku je automatické přepočítání středu oblasti do toho do ní patřícího bodu, který má nejmenší maximální vzdálenost od všech ostatních bodů patřícím do dané oblasti. Tato úprava není nutná, ale výrazně vylepšuje výsledky kroku 3.

4.3 Použití buněk a jejich centrálních bodů k získání trojúhelníkových oblastí

Upravený postup je následující:

- Propojení středů oblastí se středy jejich okrajů stejně jako v původním algoritmu za pomocí níže uvedeného algoritmu generování čar (obr. 6). Sem by patřilo i vyrovnání těchto čar, ale protože výsledky bez jejich vyrovnání jsou dostatečně kvalitní, je tento mezikrok vynechán.
- Spojení dvojic čar odpovídající všem okrajům mezi oblastmi. Toto je nutné provést pro následující mezikrok.
- Označení všech bodů na čarách identifikačním číslem čáry, označením všech koncových bodů jako rohových (protože každý konec a začátek čáry je logicky tvořen počátečním bodem čáry původně vygenerované), a označení všech hran na čarách ve všech trojúhelnících obsahujících danou hranu.
- Vytvoření trojúhelníkových oblastí za pomocí metody založené na principu bucket fill (viz dále).

4.3.1 Generování geodetických čar

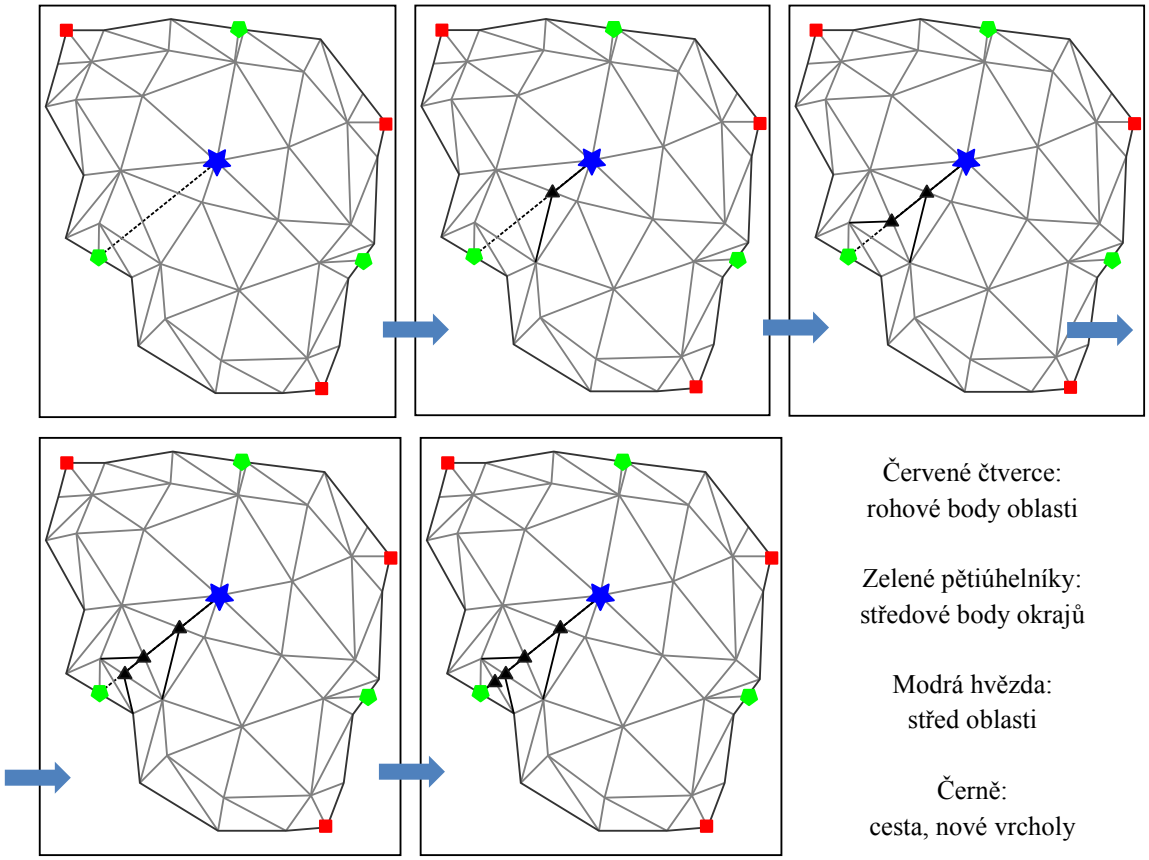
I po vytvoření oblasti odpovídající harmonické mapy není nalezení geodetické cesty mezi středem a okrajem oblasti triviálním problémem. Samotné propojení okraje a středu rovnou čarou je samozřejmě jednoduché, ale trojúhelníková síť ve většině případů neobsahuje hrany, které by této úsečce odpovídaly. Proto je takovou cestu trojúhelníky nutno „protunelovat“. Algoritmus pracuje následujícím postupem:

- Na začátku naplň frontu F všemi trojúhelníky obsahujícími počáteční bod jako jeden z vrcholů. Přidej počáteční bod do listu vrcholů čáry Č.
- Začátek vnější smyčky (pokračuje, dokud koncový bod Č není cílovým bodem).
- Pokud některý z trojúhelníků ve frontě F obsahuje koncový bod, přidej koncový bod do listu Č a ukonči smyčku.
- Začátek vnitřní smyčky (pokračuje, dokud fronta F obsahuje trojúhelníky).
- Vyber trojúhelník P z fronty F. Pokud jsou oba vrcholy P protiležící poslednímu bodu čáry více vzdálené od cílového bodu než poslední bod čáry, přeskoč na další iteraci vnitřní smyčky.
- Nalezni průsečík úsečky z počátečního bodu do koncového s hranou protiležící poslednímu bodu čáry. Pokud tento průsečík leží na této hraně, rozděl tento trojúhelník na dva podle průsečíku, dále rozděl trojúhelník sdílející tuto hranu

stejným způsobem, vyprázdni frontu F, přidej do ní trojúhelníky vzniklé rozdělením souležícího trojúhelníku, a ukonči vnitřní smyčku.

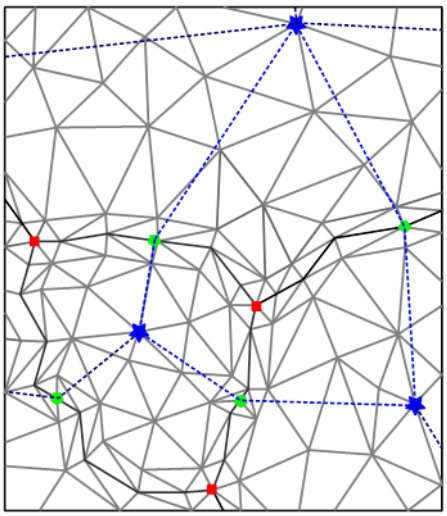
- Konec vnitřní smyčky.
- Pokud nebyla ve vnitřní smyčce provedena žádná změna, najdi vrchol sousedící s posledním vrcholem na čáře Č nejblíže cílovému neležící na Č, přidej ho do Č, a dále přidej všechny trojúhelníky obsahující tento bod do fronty F.
- Konec vnější smyčky.

Tento algoritmus zaručeně vygeneruje cestu mezi jakýmikoli dvěma body v oblasti. I v případě špatně tvarovaných oblastí nebo oblastí obsahujících defekty bude cesta nalezena, maximálně bude horší kvality (což se na konečném výsledku moc důrazně neprojeví).



Obr. 11: Algoritmus generování cesty krok po kroku

4.3.2 Vytvoření trojúhelníkových oblastí



Obr. 12: Typická trojúhelníková

Z obrázku 12 je vidět, že na okolí každého rohového bodu jsou právě tři středové body – jeho okolí tudíž tvoří trojúhelníkovou oblast jimi danou. Algoritmus tedy začne přidáním všech trojúhelníků, do kterých rohový bod spadá, a pak postupně přidává sousedící trojúhelníky tak, aby nebyly překročeny žádné čáry. Takto pro každý rohový bod vznikne jemu odpovídající trojúhelníková oblast.

V případě, že po jejich vyčerpání zůstanou nepoužité trojúhelníky, lze tyto přebytky použít jako další začátky – výsledkem jsou pak mikrooblasti tvořené trojúhelníky, které uvázly mezi čarami horší kvality. Tato eventualita ale ve většině případů nenastane.

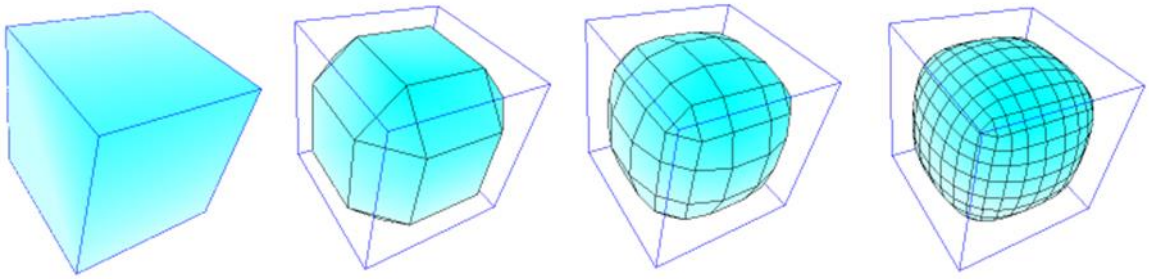
4.4 Spojení trojúhelníkových oblastí do čtyřúhelníkových

Díky poměrně vysokému počtu oblastí oproti řešení podle Eck et al [2] není ideální propojení trojúhelníkových oblastí kritické pro správný výsledek. Proto je implementace tohoto kroku zjednodušena na jednoduchý algoritmus, který oblasti prostě propojuje podle pořadí. Pro zkvalitnění výběru jsou oblasti propojovány prioritně podle toho, která zatím nepropojená oblast s danou oblastí sdílí její největší část okraje.

V případě, že v nějaké části algoritmu prozatímní sdružení oblastí způsobí, že některá z nepropojených oblastí už nemá žádné volné sousedy k propojení, algoritmus se vrátí o krok nebo více zpět a prozkoumá jiný způsob propojení. V nejlepším případě tedy pouze po dvojicích propojí oblasti tak, jak přijdou, a i v nejhorším případě po prohledání všech ostatních možností najde řešení.

4.5 Nahrazení oblastí parametrickými plochami

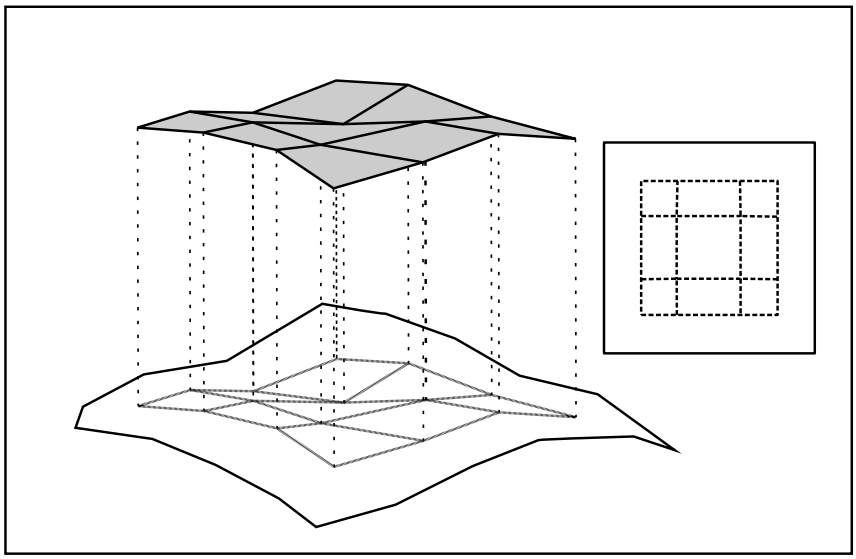
Stejně jako v metodě podle Eck et al [2] je nyní potřeba zajistit G^0 a G^1 kontinuitu. Subdivizní metoda Doo-Sabin [6] je jednoduše implementovatelná, ale po jejím dvojnásobném použití dochází k nepoměrně velkému nežádoucímu zaoblení objektu, což už logicky vyplívá z jejího postupu (viz obrázek).



Obr. 13: Doo-Sabin subdivize krychle [7]

[2] tento problém řešili použitím metody nejmenších čtverců pro přiblížení kontrolních bodů ploch z těchto Doo-Sabin bodů vypočítaných bodům na počáteční síti, ale jako jednoduší řešení se jeví opětovné využití harmonických map(obr. 14). Za předpokladu, že tyto mapy byly vytvořeny na čtverci o rozměru 1,1, jsou UV souřadnice bodů triviální. Tímto způsobem lze Doo-Sabin bodům přiřadit 3D koordináty přímo bez potřeby využití iteračních metod.

Pro potřeby další části tohoto kroku nejsou body uloženy do matice 4 x 4, ale jsou přiřazeny rohovým bodům oblasti do pole velikosti 4 tak, aby první bod pole byl vždy nejbližší rohovému, a poslední vždy nejbližší středu oblasti.



Obr. 14: použití harmonické mapy k nalezení 3D souřadnic Doo-Sabin bodů jedné plochy

Dalším krokem je vypočítání kontrolních bodů ploch. Každá oblast je nahrazena 16 Beziérovými podplochami. Na rozdíl od postupu [2] jsou všechny tyto plochy nahrazeny bikvadratickými plochami bez převádění na B-Spline plochu (to bylo částí původního algoritmu pouze pro ušetření paměti potřebné k uložení výsledku, ale vzhledem k tomu, že výstupem tohoto algoritmu nejsou parametrické plochy samotné, ale nová trojúhelníková síť, je tato optimalizace zbytečná). Kontrolní body ploch jsou definovány takto:

$$b_{00} = (C_{00} + C_{10} + C_{01} + C_{11}) / 4$$

$$b_{10} = (C_{11} + C_{10}) / 2$$

$$b_{01} = (C_{11} + C_{01}) / 2$$

$$b_{11} = C_{11}$$

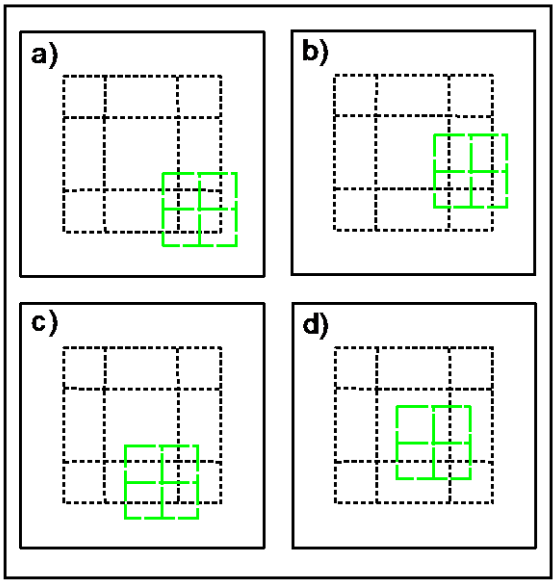
$$b_{02} = (C_{02} + C_{12} + C_{01} + C_{11}) / 4$$

$$b_{12} = (C_{11} + C_{12}) / 2$$

$$b_{22} = (C_{22} + C_{12} + C_{21} + C_{11}) / 4$$

$$b_{21} = (C_{11} + C_{21}) / 2$$

$$b_{20} = (C_{22} + C_{10} + C_{21} + C_{11}) / 4,$$



Obr. 15: Čtyři parametrické plochy vzniklé u jednoho rohového bodu

kde body b_{uv} jsou kontrolní body dané bikvadratické Beziérovy plochy a body C_{xy} jsou dříve vypočítané okolní body Doo-Sabin subdivize. Plochy pro každý rohový bod oblasti jsou definovány stejným způsobem.

Z obrázku 15 je vidět, že pouze v případě d) vnitřní plochy lze body spočítat za použití Doo-Sabin bodů spadajících do oblasti. Pro případ b) a c) ploch ležících na kraji je třeba nalézt oblast, která se zpracovávanou oblastí sdílí okraj (která ale sdílený okraj nemusí nutně sdílet ve stejném pořadí rohových vrcholů) a pro daný rohový vrchol (který je vždy sdílen) najít hledané odpovídající Doo-Sabin body. V případě rohové plochy a) je navíc třeba vypočítat rohový bod, protože oblastí sdílejících tento rohový vrchol nemusí být nutně 4, z průměru odpovídajících Doo-Sabin bodů sdílejících daný rohový vrchol.

Protože všechny přiléhající oblasti sdílejí své okrajové kontrolní body, je dosaženo G_0 kontinuity, a protože okrajové body použité pro jejich výpočet jsou také sdílené (i mezi přilehlými oblastmi), a protože výpočet rohového bodu pro každý rohový vrchol podá pro každou oblast stejný výsledek, je dosaženo i G_1 kontinuity. Díky tomu, že Doo-Sabin body odpovídají vrcholům na aproximované trojúhelníkové síti, navíc plochy dosahují dobré aproximace tvaru a objemu objektu daného trojúhelníkovou sítí.

4.6 Zvyšování úrovně detailů

Vzhledem k implementační náročnosti je tento krok vynechán. Vyšší úrovně detailů lze místo toho dosáhnout zmenšením minimálního zlomku vzdálenosti požadované v kroku 2 (nalezení pseudo-Voroného buněk).

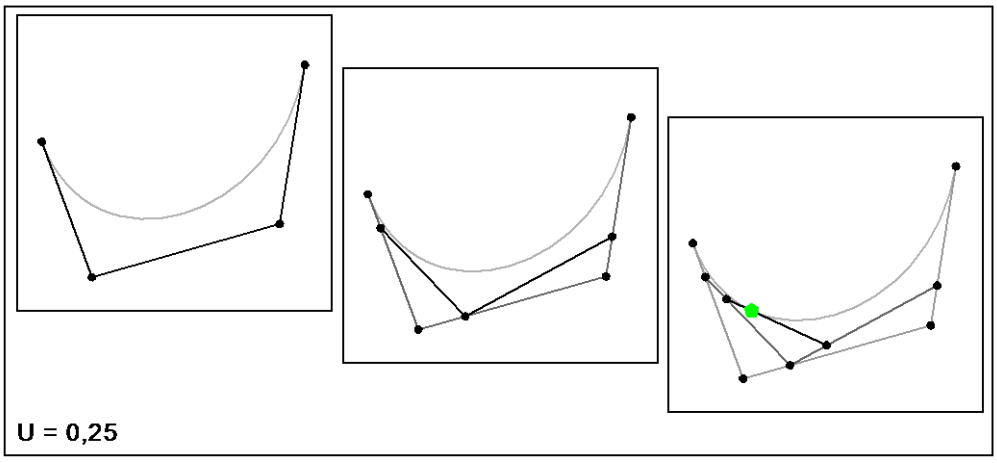
4.7 Uložení výstupních dat

Z parametrických ploch je nyní nutné vygenerovat novou trojúhelníkovou síť. Jednoduchým postupem je vytvoření rovnoměrně rozložené mřížky bodů (jejíž hustota určuje jak počet výsledných vrcholů, tak i hladkost výsledné sítě, a tudíž její určení je ponecháno na uživateli) a jejich následné postupné propojení trojúhelníky. Pro hustotu *n* tedy pro každou jednotlivou parametrickou plochu vznikne $(n+1)^2$ vrcholů a $2n^2$ trojúhelníků.

Pro získání bodů ležících na ploše je použit algoritmus De Casteljau [8]. Ten pro kontrolní body vstupní Beziérovy křivky vypočítá hodnotu odpovídající vstupní souřadnici *u* spadající mezi 0-1 pomocí rekurzivního postupu, kde je postupně snižován stupeň křivky, dokud není redukována na jednoduchý případ úsečky. V každém rekurzivním kroku jsou nové kontrolní body Beziérovy křivky nižšího stupně spočteny pomocí této rovnice:

$$Bn = An * (1 - u) + An + 1 * u,$$

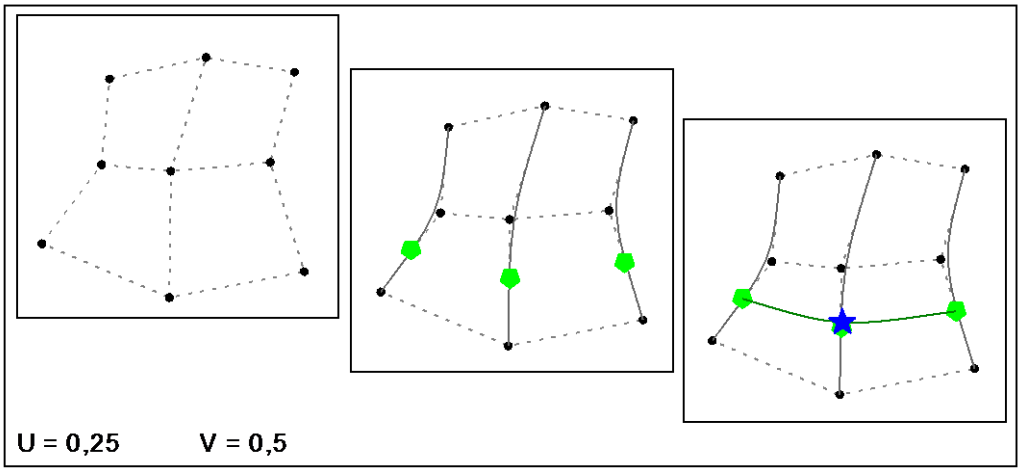
kde B jsou kontrolní body Beziérovy křivky nižšího stupně počtu n, A jsou body vstupní Beziérovy křivky počtu n+1 a u je požadovaná souřadnice. Pokud vstupní křivka má pouze jeden kontrolní bod, pak je tento bod výsledkem a rekurze skončí jeho návratem (obr. 16).



Obr. 16: De Castejau rekurzivní algoritmus pro výpočet bodu na Beziérově křivce

Na rozdíl od standartního De Casteljau algoritmu je ale potřeba získat třírozměrné souřadnice *X, Y, Z* dané dvourozměrnými souřadnicemi U, V. Algoritmus je proto použit dvakrát:

- V prvním kroku je použit pro získání kontrolních bodů nové křivky použitím na všechny křivky danými kontrolními body sdílejícími souřadnici *V* za použití vstupní souřadnice *U*
- Tyto body jsou pak použity jako definice nové Beziérovy křivky, na které je De Castejau algoritmus spuštěn s použitím vstupní souřadnice *V*.
- Výsledek je spočítán třikrát, zvlášť pro každou ze souřadnic *X, Y, Z*.



Obr. 17: Příklad výpočtu bodu na souřadnicích (0.25, 0.5)

Takto získané body parametrických ploch jsou už pak jednoduše propojeny do pravidelné mřížky. Pro jejich uložení do souboru (nebo zpracování jako jediné trojúhelníkové sítě) je ještě vhodné propojit okrajové body souležících Beziérových ploch.

5 Implementace

5.1 Software

Práce je zpracována v programovacím jazyku C#. K zobrazení dat je použit VTK (Visualisation Toolkit), specificky wrapper ActiViz .NET, který VTK (napsané v C++) portuje do C#. Jako matematická knihovna je použit balíček Accord .NET. K práci na projektu je třeba:

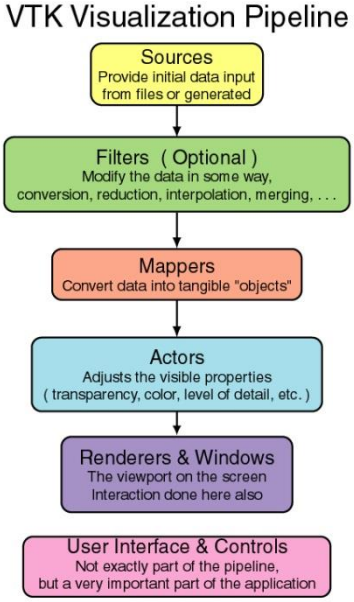
- Kitware ActiViz .NET Library
- Accord .NET Library
- Microsoft Visual Studio 2010 nebo vyšší

Visual Studio je dále třeba nastavit k použití ActiViz .NET a Accord .NET knihoven. K tomu lze přistupovat různě, avšak nejjednodušším způsobem je použít NuGet

Package Manager. Pro použití instalované knihovny je nutné jen pro daný projekt otevřít manager instalovaných NuGet balíčků a potřebnou knihovnu zaškrtnout pro použití.

5.1.1 VTK

Vizualization Toolkit je zaměřen pro zobrazení a práci s 3D daty. K tomuto účelu je už přímo v toolkitu připravena sbírka užitečných jak algoritmů, například pro řezání, vyhlazení nebo redukci trojúhelníkové sítě, tak i možnost jednoduchého vykreslení různých typů dat - běžné trojúhelníkové sítě, oblaky bodů, bitmap nebo i volumetrická data. Jedním ze základních principů je rozdělení programu do oddělených modulů, z nichž každý má právě jednu funkci, ale může mít více vstupů i výstupů (příkladem modulu by mohl být třeba modul, který ze vstupního obrázku vytvoří oddělené červené, modré a zelené kanály a ty podá na výstup). Další důležitou věcí je oddělení grafické reprezentace dat od dat samotných – data se automaticky převádí do zobrazitelné podoby pomocí Mapperů, jejichž výsledkem jsou objekty typu Actor, které už jsou přímo zobrazeny Rendererem do daného RenderWindow okna.



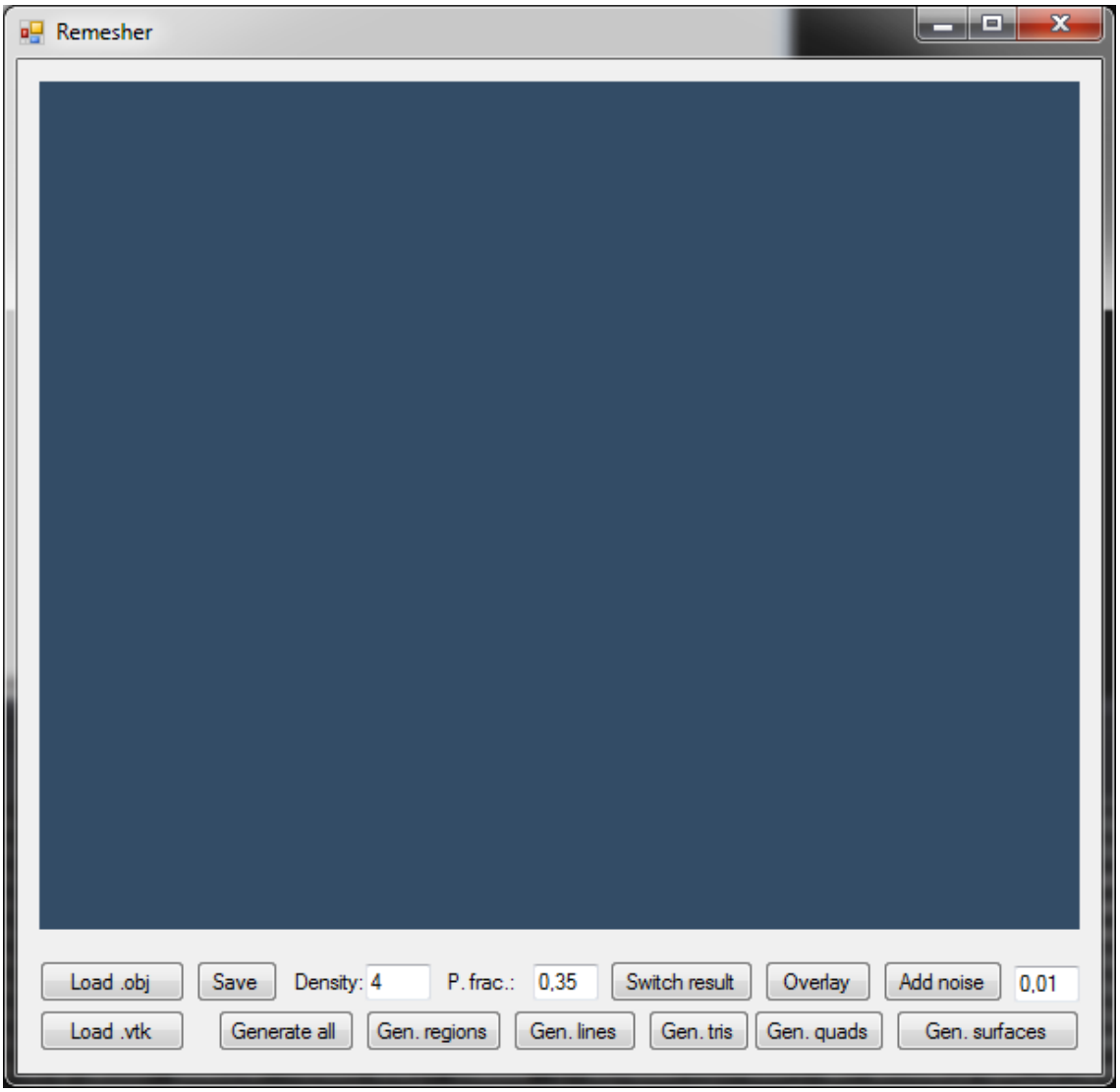
Obr. 18: VTK Pipeline

5.1.2 Accord .NET

Accord .NET je framework pro vědecké výpočty v .NET. Vychází z AForge .NET, frameworku určeného pro zpracování obrazových dat. Obsahuje široké spektrum vědeckých výpočetních funkcí, funkcí pro zpracování statistických dat, rozpoznávání vzorů a mnoho jiných. Dále umí mimo jiné generovat různá rozdělení pravděpodobnosti, testování hypotéz a měření výkonnostních metrik. Pro tuto práci je použit hlavně pro svoji matematickou část.

5.2 Uživatelské rozhraní

Vzhledem k míře automatizace je uživatelské rozhraní práce velmi jednoduché. Lze přes něj nastavit požadované množství bodů na Beziérovu plochu, načíst/uložit data a zobrazit výsledky všech kroků. Dále je zde samozřejmě tlačítko pro spuštění algoritmu, nebo pro ladění přidaná možnost spuštění jednotlivých kroků algoritmu. Tlačíto Overlay umožňuje zobrazení původní sítě a vygenerovaných čar přes síť vygenerovanou. Switch Result pak přepíná mezi zobrazením mezivýsledků.

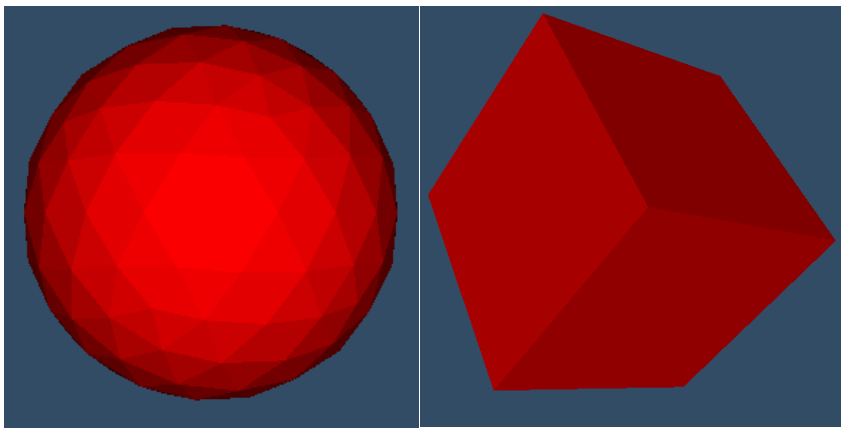


Obr. 19: Uživatelské rozhraní

5.3 Výsledek práce

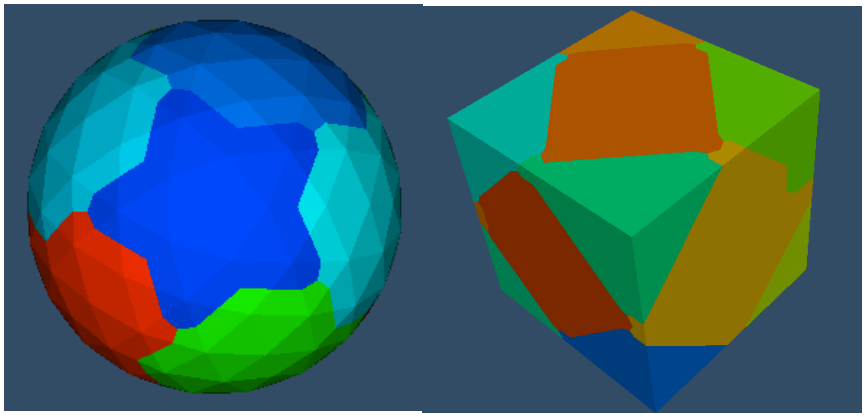
Kvalitu výstupu lze nejlépe ilustrovat graficky.

Koule Krychle



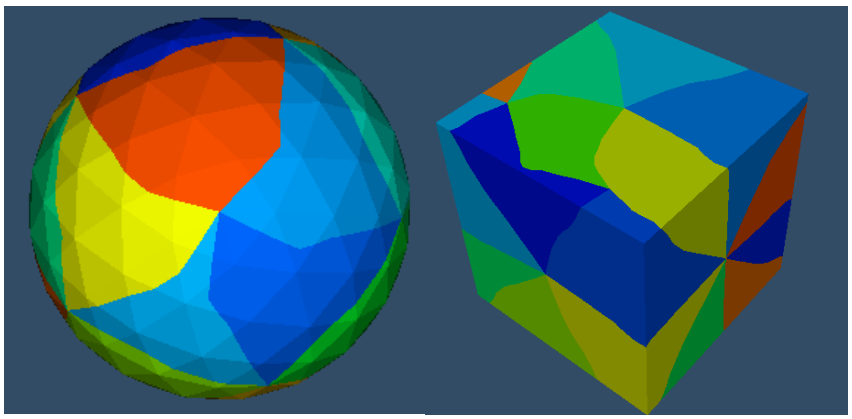
Obr. 20

Vonorové buňky



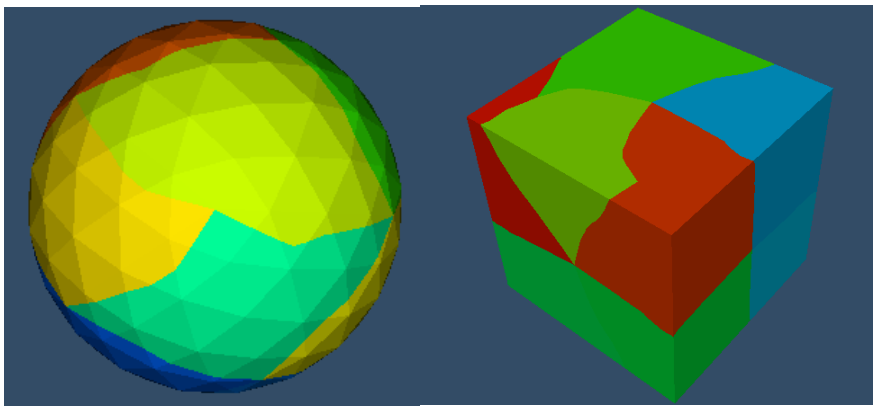
Obr. 21

Trojúhelníkové oblasti



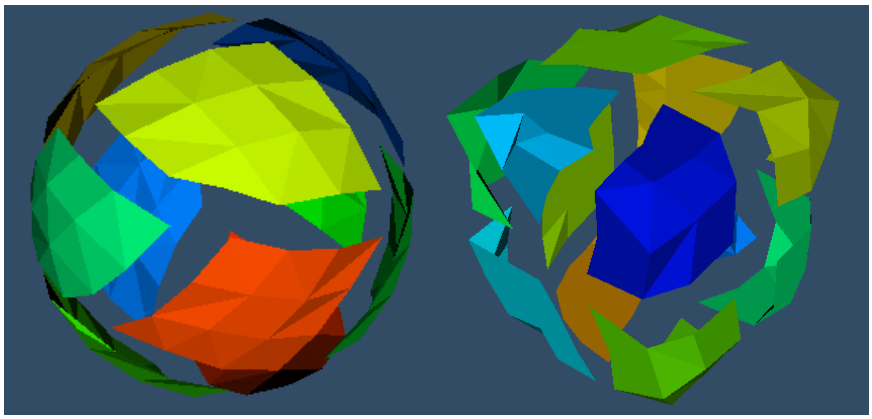
Obr. 22

Čtyřúhelníkové oblasti



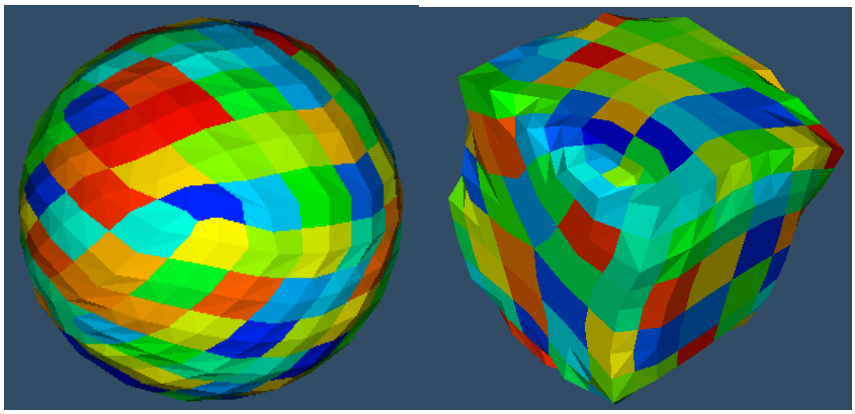
Obr. 23

Doo-Sabin body



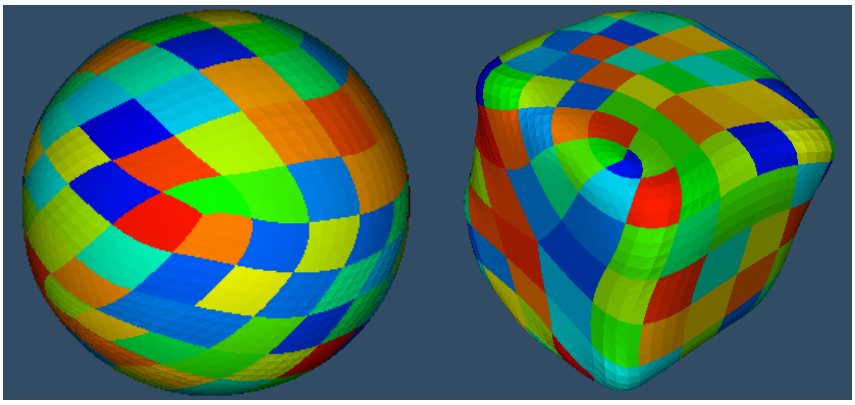
Obr. 24

Kontrolní body Beziérových ploch



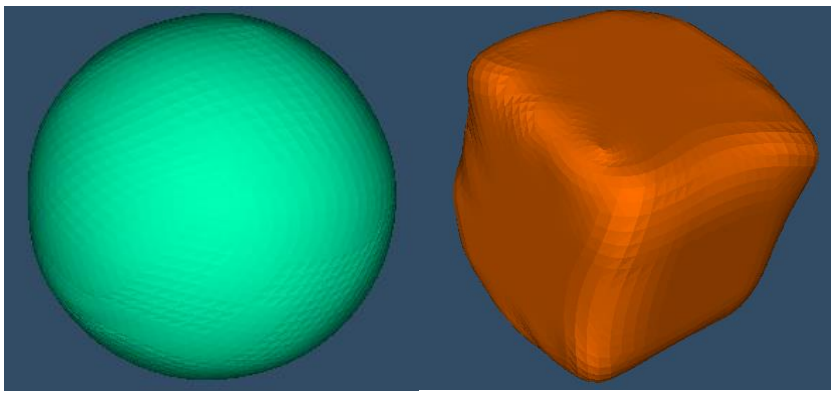
Obr. 25

Beziérovy plochy



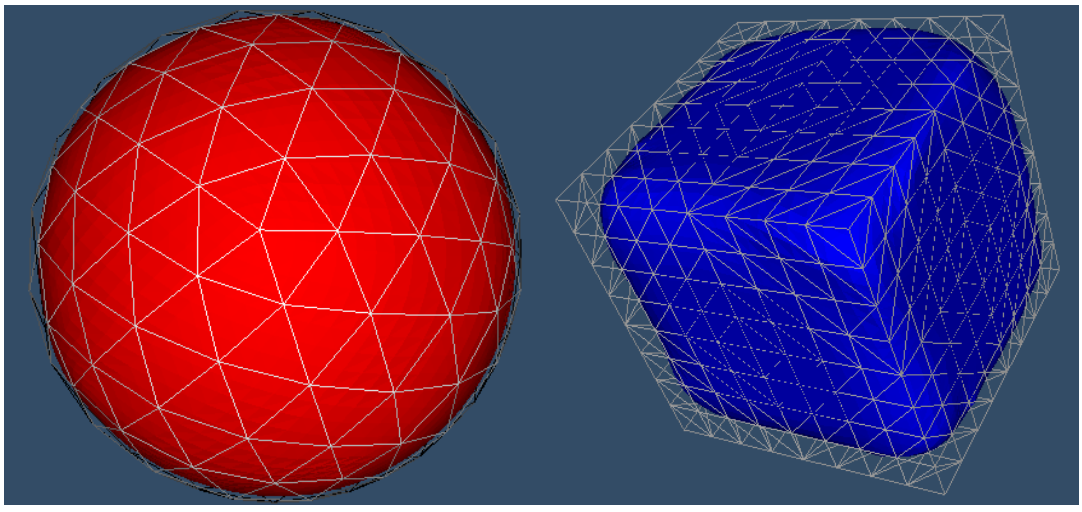
Obr. 26

Výstupní trojúhelníková síť



Obr. 27

Porovnání výstupu se vstupem



Obr. 28

Z obrázku 28 je jasně vidět, že zatímco pro relativně hladké objekty je výsledek programu dobré kvality, pro objekty dosahující ostré hrany není ideální. To je samozřejmě způsobeno tím, že algoritmus si už od začátku za cíl klade vyhlazení a zaoblení objektu.

5.4 Testování

Cílem testování je ověřit kvalitu výstupu programu. Protože je třeba ověřit, jaké výsledky program podá pro data ovlivněná šumem nebo obsahující různé defekty, je ideální použití uměle vygenerovaných topologicky jednoduchých objektů, u kterých jsou jejich vlastnosti (objem, povrch, topologie) předem známé. Pro umělé objekty jsou tedy použita tyto testovací situace:

- Zvyšování počtu vrcholů a trojúhelníků pro stejný objekt
- Zvyšování požadovaného množství trojúhelníků výstupních pro konstantní vstup
- Zvyšování šumu (náhodné posunutí bodů pomocí Gaussovo pravděpodobnostního rozdělení) aplikovaného na konstantní vstup

Reálná testovací data jsou pak využita přímo bez úprav.

Testovaná kritéria:

- Časová náročnost algoritmu
- Paměťová náročnost algoritmu
- Změna objemu objektu
- Změna povrchu objektu
- Vizuální porovnání

Výsledky jsou buď získány pomocí VTK knihovní funkce vtkMassProperties, nebo programově. Pro všechny testy bylo nastaveno 32 trojúhelníků (přesnost 4 => (4*4*2)) na parametrickou plochu.

5.4.1 Umělá data

Koule (Ikosféra)

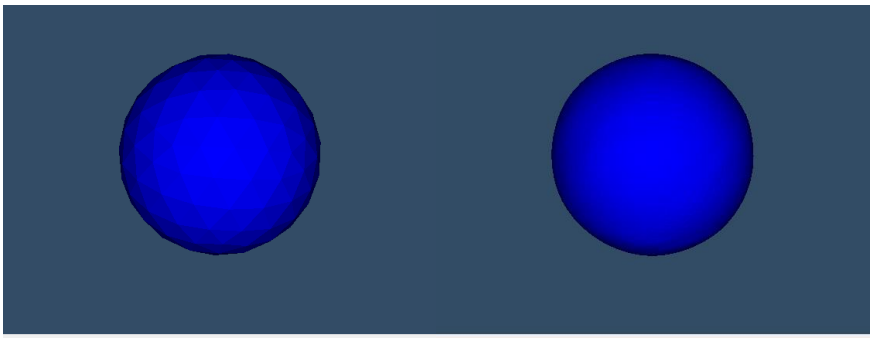
Koule je jako testovací objekt zvolena z důvodu své topologické jednoduchosti. Velká deformace na výsledku remeshingu by byla evidentní. Koule byla také použita pro většinu vývoje programu.

Matematická definice:

Objem	Povrch	Poloměr
4,189	12,556	1

Bez úprav:

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	4,189	12,556	162	320		
Výstup	4,047	11,740	514	1024	1,6s	8MB

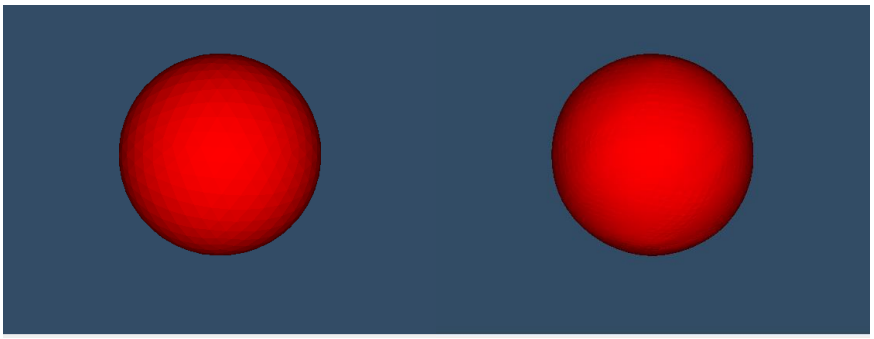


Obr. 29

Je vidět, že kouli program aproximuje velmi přesně. Výsledná síť je oproti vstupní viditelně zaoblená. Pro tento relativně malý počet trojúhelníku jsou časové a paměťové nároky zanedbatelné.

Po zvýšení počtu trojúhelníků:

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	4,152	12,506	642	1280		
Výstup	3,873	11,938	1412	2820	11,5s	13MB

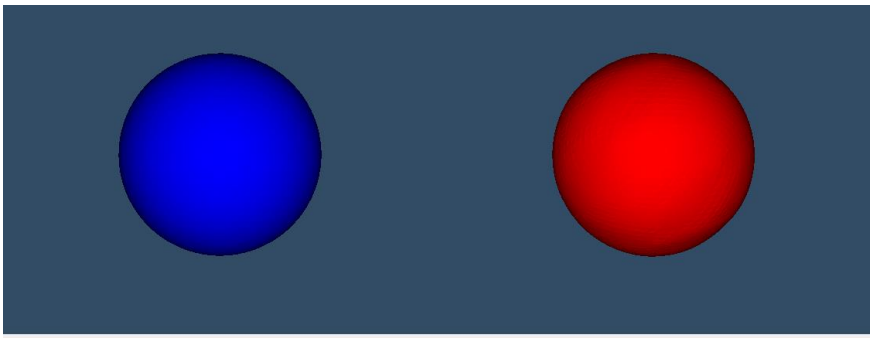


Obr. 30

Pro lepší vstupní kouli už jsou výhody remeshingu zanedbatelné. Výsledná koule je navíc drobně zmenšená oproti vstupní, což je důsledkem dalšího vyhlazování už velmi hladkého objektu.

Po dalším zvýšení počtu trojúhelníků:

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	4,152	12,506	2562	5120		
Výstup	3,873	11,938	4110	8216	2m 17,8s	19MB

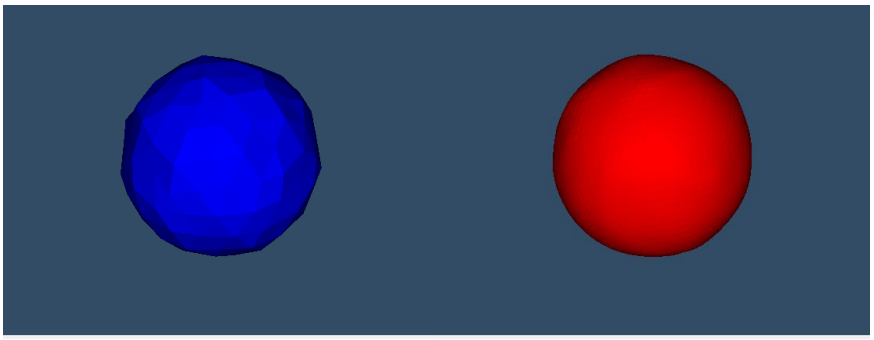


Obr. 31

V tomto testu už není rozdíl mezi vstupem a výstupem lidským okem bez přiblížení viditelný. Stejně jako v předchozím testu dochází ke zmenšení koule.

Po zatížení šumem o distribuci 0,02

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	4,041	12,427	162	320		
Výstup	3,767	11,736	508	1012	1,6s	8MB

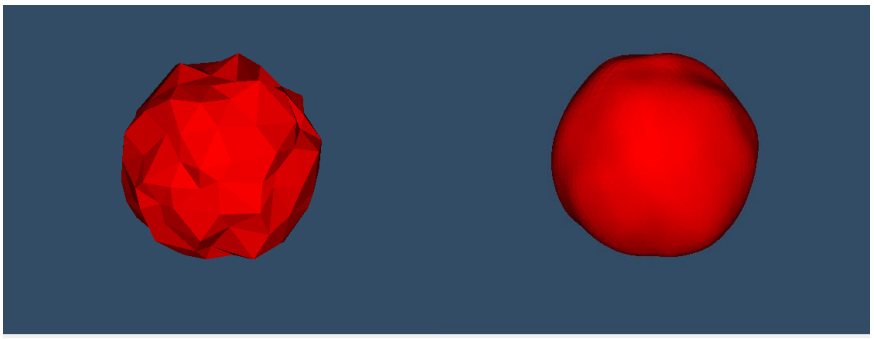


Obr. 32

Pro menší množství šumu, který je přesto na vstupní kouli viditelný, dochází k dobrému zaoblení výstupu. Jsou zde sice viditelné hrboly, ale ty jsou oproti velikosti koule relativně malé výšky.

Po zatížení šumem o distribuci 0,05

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	4,109	13,294	162	320		
Výstup	3,801	11,878	**525**	**1046**	1,6s	8MB



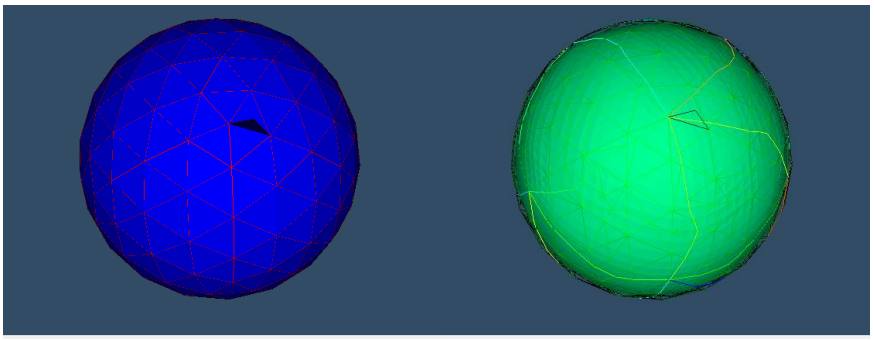
Obr. 33

Pro větší množství šumu už se na výstupu rozdíly výšek projeví výrazněji. I přesto, že je trochu hrbolatá, lze však o výsledku prohlásit, že je to koule, zatímco vstup už se jí moc nepodobá.

Koule (Ikosféra) obsahující nemanifoldní hranu

Zahrnuta pro ověření, že program je schopný zvládnout lokalizované chyby sítě.

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	4,047	12,391	163	322	-	-
Výstup	2,689	11,737	2562	5120	1,7s	8MB



Obr. 34

Program neselhal a výsledná koule je identická s tou vzniklou remeshingem koule bez chyby. Z výsledku je tedy patrné že osamocené chyby výstup neovlivní.

Krychle

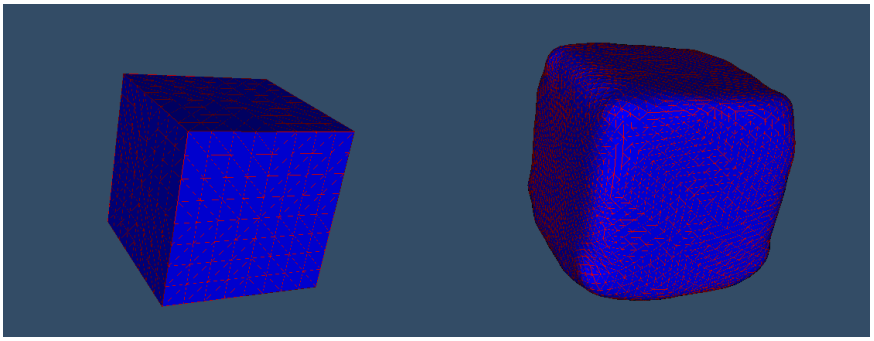
Krychle je jako testovací objekt zvolena jak z důvodu jednoduchosti výpočtu jejího přesného objemu a povrchu (čímž lze snadno ověřit správnost funkce použité metody spočtu objemu/povrchu), tak i pro vyzkoušení funkce programu na objektu obsahujícím ostré hrany.

Matematická definice:

Objem	Povrch	Délka hrany
8	24	2

Bez úprav:

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	8	24	386	768	-	-
Výstup	7,573	20,893	1122	2240	7,7s	8MB

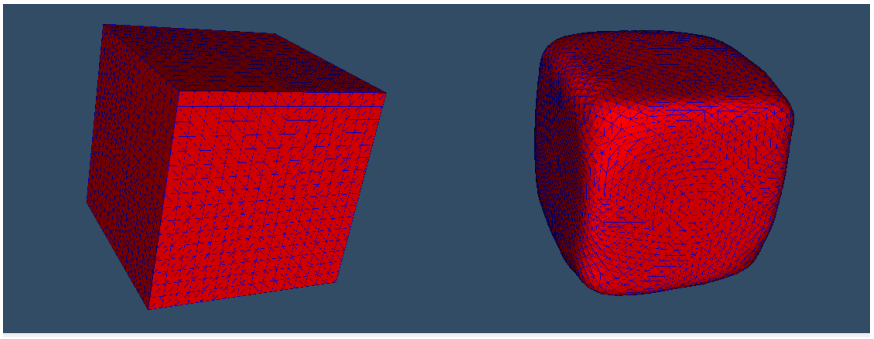


Obr. 35

S krychlí má program logicky problémy. Detekce ostrých hran není součástí algoritmu, takže je v tomto smyslu správným výsledkem, že dojde k jejich zaoblení. Objem a povrch odpovídá očekávání pro krychli se zaoblenými hranami.

Po zvýšení počtu trojúhelníků:

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	8	24	1538	3072	-	-
Výstup	7,573	20,893	3019	6034	1m 6,5s	12MB

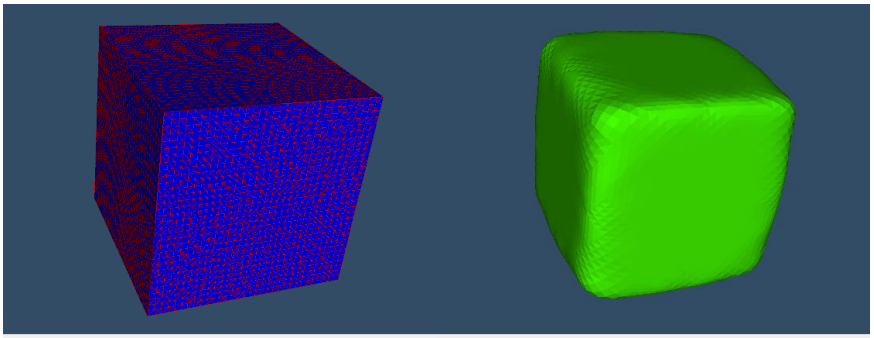


Obr. 36

I pro větší počet trojúhelníků je výsledek prakticky identický, což vychází z principu algoritmu detekce regionů i z toho, že přestože má více trojúhelníků, je vstupní krychle tvarově identická.

Po dalším zvýšení počtu trojúhelníků:

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	8	24	6146	12288	-	-
Výstup	7,572	20,876	9111	18218	16m 2,0s	19MB

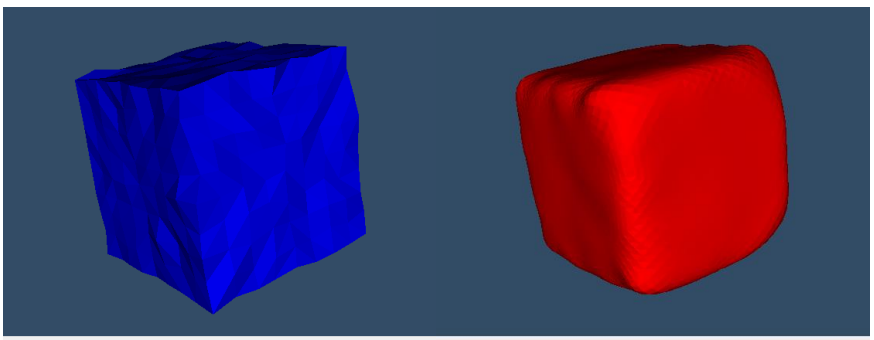


Obr. 37

Po dalším zvětšení počtu trojúhelníků už došlo k výraznému zpomalení programu. Výsledek je ale opět prakticky stejný.

Po zatížení šumem o distribuci 0,02

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	7,978	24,202	386	768		
Výstup	7,564	20,937	1114	2224	7,7s	8MB

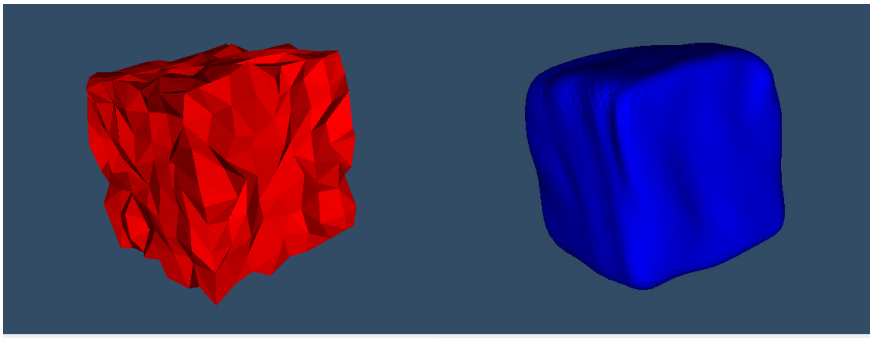


Obr. 38

Deformovaná krychle má své deformace skoro vyhlazeny. Na hranách se však deformace projevily, což je důsledkem závislosti výsledku tvaru krychle na bodech na nich ležících.

Po zatížení šumem o distribuci 0,05

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	7,982	26,21	386	768		
Výstup	7,399	20,539	1008	2012	5,8s	8MB



Obr. 39

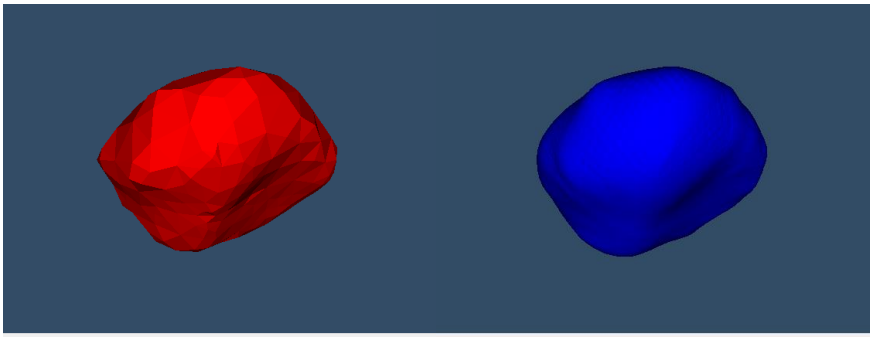
Pro více deformovanou krychli se deformace na výstupu projeví jako malé prohlubeniny a vyvýšeniny. Pro tento test lze i tak prohlásit, že výsledný objekt se krychli podobá víc jak vstupní.

5.4.2 Reálná data

Malý orgán

Vybrán jako reprezentace relativně jednoduchých reálných dat. Jeho malý počet vrcholů a trojúhelníků umožňoval rychlejší kontrolu a opravu vzniklých chyb.

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	17348	3693	252	500	-	-
Výstup	15696	3332	706	1408	2,9s	8MB



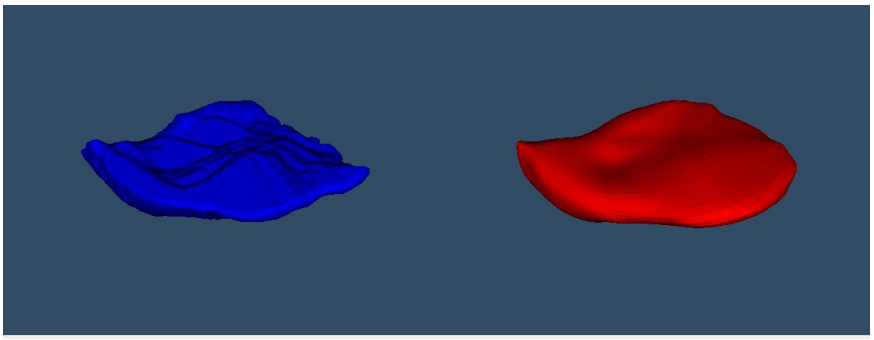
Obr. 40

Na tomto relativně jednoduchém objektu došlo k velkému žádoucímu zaoblení. To je výsledkem nízkého počtu trojúhelníku reprezentujících hladký objekt, jež byly nahrazeny mnohem hustější sítí. Změna objemu a povrchu byla také minimální a časové a paměťové nároky jsou zanedbatelné.

Velký orgán

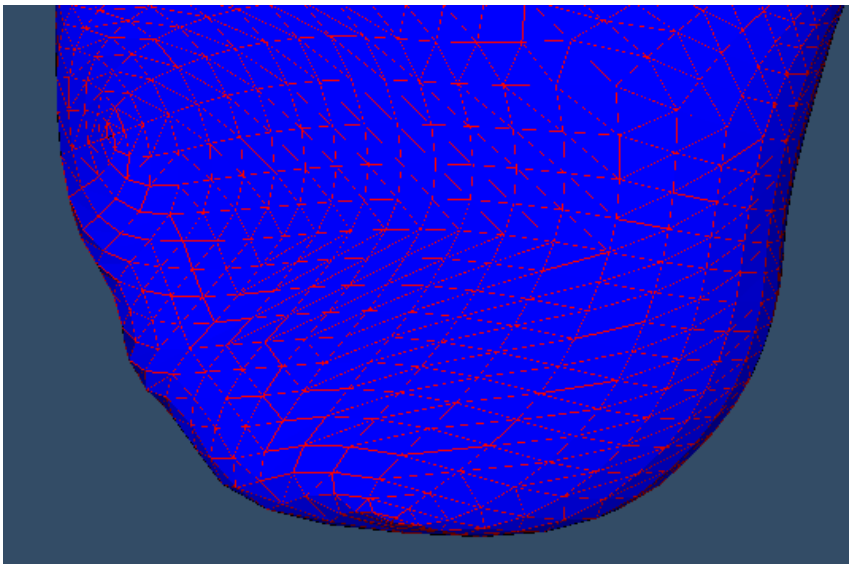
Vybrán jako reprezentant složitějších dat. Obsahuje mnoho hran, které by potřebovaly vyhladit. Jeho tvar je relativně jednoduchý, ale ne triviální.

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	619406	61106	7502	15000	-	-
Výstup	496410	47610	10222	20440	48m 50s	264MB



Obr. 41

Výsledek pro tento model dosahuje překvapivě dobré kvality. Všechny nežádoucí hrany na původním objektu byly vyhlazeny beze ztráty celkového tvaru objektu. Vzniklá trojúhelníková síť je i z pohledu tvaru trojúhelníků celkem kvalitní (viz obr. 42).

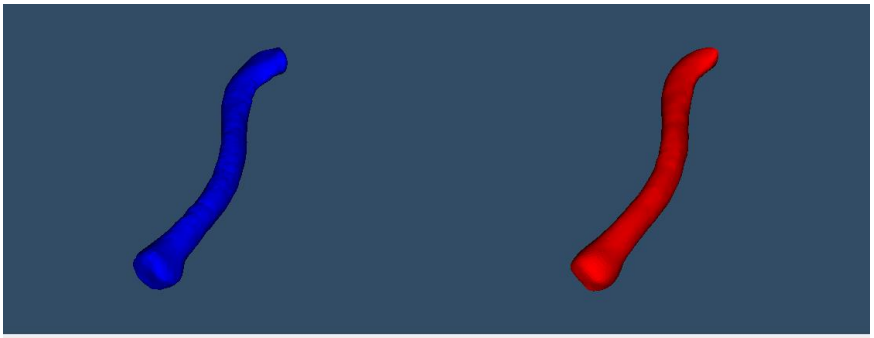


Obr. 42: Trojúhelníková síť výsledku testu na velkém orgánu

Kost

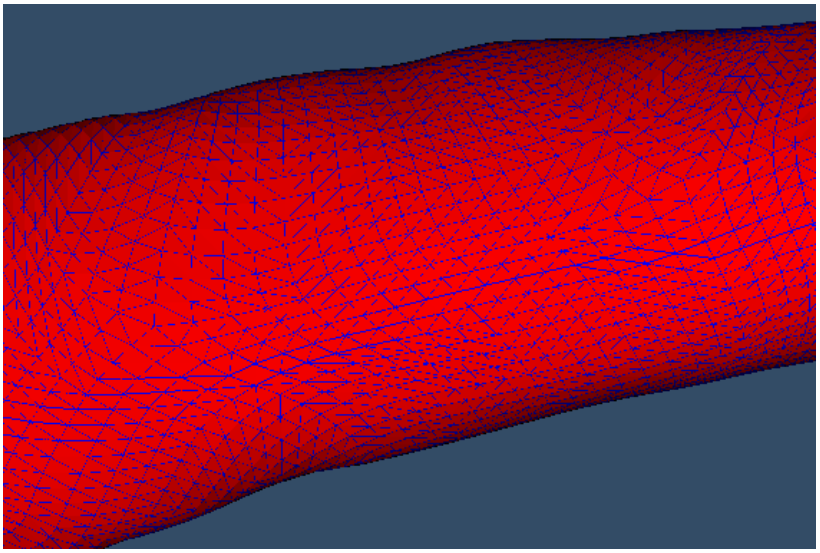
Model kosti byl vybrán pro jeho ověření funkčnosti programu na síti složitějšího tvaru.

	Objem	Povrch	Vrcholů	Trojúhelníků	Čas	Paměť
Vstup	619406	61106	1502	3000	-	-
Výstup	496410	47610	3785	7590	3m 27s	28MB



Obr. 43

Výsledek programu pro tento test je také značné kvality. Jediným vzniklým problémem je ztráta tvaru jednoho konce kosti. Jinak je její tvar dobře dodržen. Z pohledu kvality trojúhelníků je výsledek také podstatným vylepšením (obr. 44). I pro plochy, na kterých má většina dvojic trojúhelníků spojnici na delší diagonále, je rozložení mřížky rovnoměrné, a tento výsledek lze jednoduše vylepšit použitím filtru z externího nástroje (např. Blender).

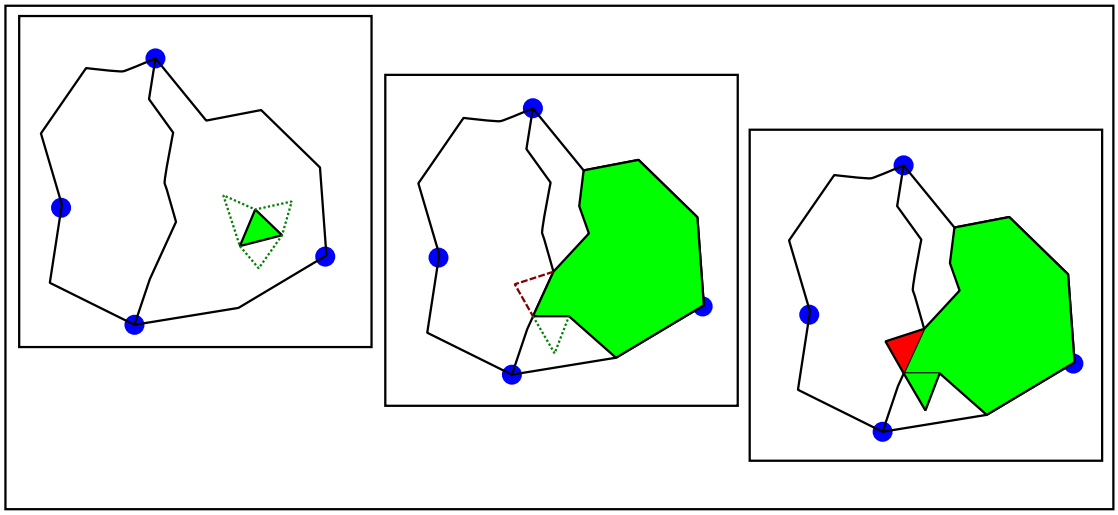


Obr. 44: Trojúhelníková síť výsledku testu na kosti

5.5 Analýza výsledků

Pro většinu použitých reálných dat algoritmus na začátku testování selhal v kroku generace trojúhelníkových oblastí vznikem nesprávné oblasti (počet rohů jiný než 3), ať už z důvodu chybné čáry oddělující dvě vznikající trojúhelníkové oblasti, nebo kvůli „proklouznutí“ nějakého trojúhelníka přes validní čáru (obr. 45). Po mnoha opravených chybách v algoritmech generování čar a algoritmu generování trojúhelníkových oblastí byly sice výskyty tohoto problému sníženy, ale bohužel pro velké množství objektů přetrvaly. Vzhledem k jednoduchosti algoritmu tvorby trojúhelníkových oblastí je velmi pravděpodobné, že k chybě dochází kvůli numerickým nepřesnostem, jejichž odhalení je bohužel složité a časově náročné. Tato situace ve většině případů navíc nastane jen pro velmi malý zlomek zpracovávaných trojúhelníků, což dále zesložiťuje jeho odhalení.

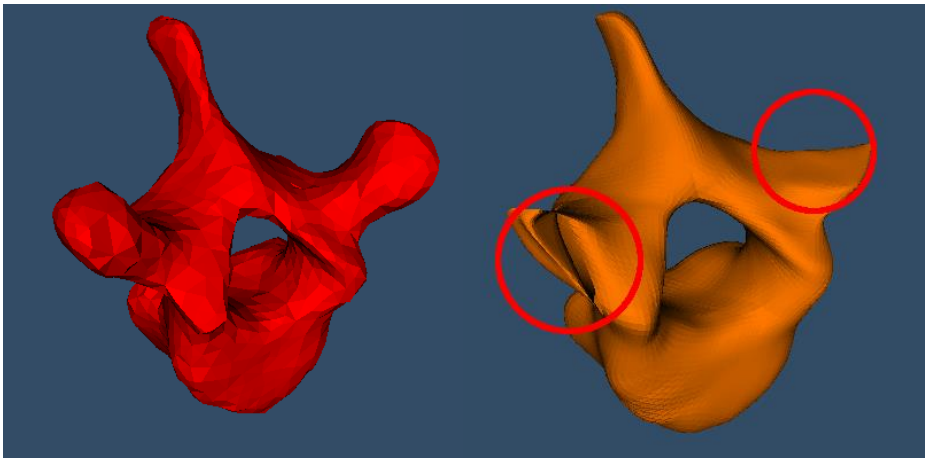
Čáry jsou pak sice vygenerovány (a z pohledu lidského oka jsou často bezchybné), ale algoritmus generování trojúhelníkových oblastí přes ně přesto pronikne, i přes robustní kontrolu hran ležících na čarách (jak hrany, tak i vrcholy jsou označené jako ležící na čáře). Navíc i přes mnoho pokusů, jak tento problém obejít opravou problematických oblastí, nebylo nalezeno uspokojivé řešení. Například pokus o vytvoření nové čáry spojující body oblasti ve většině případů selhal, ať už z důvodu numerických chyb nebo i z důvodu překážky v podobě již existující cesty. Další řešení použitím více naivních čar nalezených za použití Dijkstrovo algoritmu nejkratších cest se také prokázalo jako nepříliš spolehlivé. Pokusy o jiný způsob rozhodnutí, které trojúhelníky spadají do které oblasti, selhaly díky nesplnění podmínky o sdílení rohů.



Obr. 45: Problematická situace.

a) Začátek algoritmu růstu oblasti
b) Trojúhelník s hranou na čáře
c) Přidání trojúhelníku do oblasti nespadajícího

Jako záplatové řešení tedy algoritmus generování oblastí z chybných trojúhelníkových oblastí se čtyřmi rohy vytvoří rovnou čtyřúhelníkové a ostatní (počet rohů <3 značí pominutelné mikrooblasti) vynechá. V už mnohem vzácnějším případě oblastí s počtem rohů >4 tedy v objektu vznikne otvor nebo agresivní zaoblení (příklad viz. obr. 46-b), ale alespoň se algoritmus dokončí pro zbytek objektu (pro oblasti které by s touto oblastí sousedily je bohužel nutné použít rezervní a velmi nepřesný způsob vypočtení na ní závisejících kontrolních bodů). Z časových důvodů už bohužel nebylo možno všechny zbývající chyby nalézt a opravit. Kromě těchto situací ale lze prohlásit, že princip algoritmu funguje přesně podle požadavků práce.



Obr. 46: a) model obratle b) na něm vzniklé chyby

Pro všechny objekty dochází celkem konstantně ke zmenšení objemu způsobeného jeho zaoblením. Pro povrch záleží hlavně na tvaru objektu a na šumu na objektu existujícím.

Zajímavým důsledkem remeshingu je, že výsledný povrch a objem zůstává skoro konstantní i pro větší množství šumu. Pro objekt se šumem dojde k distinktivnímu vyhlazení i pro značné hodnoty šumu. I pro objekty s relativně složitým tvarem je obecný tvar objektu zachován.

Algoritmus je odolný vůči osamostatněným nemanifoldním hranám a vrcholům.

Časová náročnost roste exponenciálně s počtem vstupních vrcholů / trojúhelníků. Paměťová náročnost roste také, ale pouze lineárně.

V případě, že by byly nalezeny a opraveny chyby v generaci čar a oblastí, by algoritmus podával kvalitní výsledky i pro objekty relativně složitých tvarů. I bez těchto oprav pro mnoho objektů podá dobrý výsledek, avšak pro některé díky nenalezeným chybám bude obsahovat chybné části.

6 Závěr

Při zpětném pohledu na práci je evidentní, že použité řešení bylo vzhledem k požadavkům práce velmi komplikované, což mělo za důsledek složitou implementaci a obtížné opravování chyb. Přestože algoritmus generování čar byl na implementaci velice komplikovaný, jeho náhrada více naivním, ale jednodušším algoritmem se projevila jako nepoužitelná. Jeho komplikovaná implementace tudíž způsobila spotřebu neúměrného množství času (občas neúspěšnými) pokusy o jeho opravu. Algoritmus je kvůli tomu při uzavření práce pro testované objekty funkční, ale lze nalézt objekty jiné, pro které na výstupu vznikne lokalizovaná chyba.

Časová náročnost zpracování je neúměrně velká vzhledem k počtu zpracovávaných vrcholů a to tak, že je pro objekty o více jak zhruba 1000 vrcholech vyžadováno nadměrné množství času. V programu existuje mnoho míst, kde je zbytečně prováděna kontrola každého trojúhelníka s každým nebo kontrola každého vrcholu s každým. Zde by byla vhodná další optimalizace.

Paměťové nároky aplikace jsou naopak skoro lineární. Nebylo by tudíž na škodu pro optimalizaci programu využít algoritmy, které snižují časové nároky na úkor paměťové náročnosti – například využitím Octree stromové struktury pro drastickou redukci kontrolovaných trojúhelníků při hledání sousedních a podobně.

Vzhledem ke složitosti programu a jeho nepříliš robustní implementaci by nejlepším navázáním na tuto práci byl důkladný refactoring. Ze zpětného pohledu je totiž evidentně mnoho míst, kde bylo použito nerozumné řešení, avšak jeho předělání by vyžadovalo přepis celého programu nebo alespoň jeho velké části.

V případě pokračování v této práci by jedním z možných směrů rozšíření byla implementace adaptivního zvyšování detailů podle [2]. Výsledná síť by pak dosahovala menšího počtu trojúhelníků, nebo naopak lepšího zachování drobných detailů objektu. V případě, že by nastala potřeba, existuje i možnost implementace detekce a zachování ostrých hran objektu.

Pro objekty, pro které program neselže, jsou ale výsledky kvalitní. Výsledné objekty jsou vždy hladké a vliv šumu je podstatně snížen. Výsledná síť je také vždy bezchybná vzhledem k nemanifoldním hranám, vrcholům a jiných lokálním chybám trojúhelníkové sítě. Kvalita trojúhelníků by mohla být celkem jednoduše vylepšena zajištěním použití kratší diagonály při generaci trojúhelníků z parametrických ploch, a i bez tohoto vylepšení je v průměru lepší než na původní síti.

Citace

[1] Reconstruction of Solid Models from Oriented Point Sets. Michael Kazhdan. *www.cs.jhu.edu.* [Online] [Citace: 25. 3. 2014.] Dostupné z http://www.cs.jhu.edu/~misha/MyPapers/SGP05.pdf

[2] Automatic reconstruction of B-spline surfaces of arbitrary topological type. Matthias Eck and Hugues Hoppe. *www.msr-waypoint.com*. [Online] [Citace: 25. 3. 2014.] Dostupné z http://www.msr-waypoint.com/en-us/um/people/hoppe/bspline.pdf

[3] Recent advances in remeshing of surfaces. Alliez, P., Ucelli, G., Gotsman, C., and Atiene, M. www.cs.technion.ac.il. [Online] [Citace: 28. 3. 2014.] Dostupné z http://www.cs.technion.ac.il/~gotsman/AmendedPubl/Pierre/remeshing_survey.pdf

[4] Surface reconstruction from unorganized points. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W. *http://research.microsoft.com.* [Online] [Citace: 28. 3. 2014.] Dostupné z http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf

[5] Mesh optimization. Hoppe, H. DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W. *http://research.microsoft.com.* [Online] [Citace: 25. 3. 2014.] Dostupné z http://research.microsoft.com/en-us/um/people/hoppe/meshopt.pdf

[6] A subdivision algorithm for smoothing down irregularly shaped polyhedrons, Proceedings on Interactive Techniques. *http://trac2.assembla.com* [Online] [Citace: 25. 3. 2014.] http://trac2.assembla.com/DooSabinSurfaces/export/12/trunk/docs/Doo%201978%20Subdivision%20algorithm.pdf

Behavior of recursive division surfaces near extraordinary points. D. Doo, M. Sabin *http://users.cms.caltech.edu.* [Online] [Citace: 25. 3. 2014.] Dostupné z http://users.cms.caltech.edu/~cs175/cs175-02/resources/DS.pdf

[7] Subdivision Surfaces. *www.student.cs.uwaterloo.ca.* [Online] [Citace: 25. 3. 2014.] Dostupné z https://www.student.cs.uwaterloo.ca/~cs779/Gallery/Winter2003/ijrutherford/

[8] Piecewise Linear Approximation of Bézier Curves. *http://citeseerx.ist.psu.edu.* [Online] [Citace: 25. 3. 2014.] Dostupné z http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.162&rep=rep1&type=pdf

[9] Vertex-based Delaunay triangulation of meshes of arbitrary topological type. *www.math.washington.edu.* [Online] [Citace: 25. 3. 2014.] Dostupné z https://www.math.washington.edu/~duchamp/preprints/tiling.pdf

[10] Constructing C^1 Surfaces of Arbitrary Topology using Biquadratic and Bicubic Splines. *http://docs.lib.purdue.edu*. [Online] [Citace: 25. 3. 2014.] Dostupné z http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1990&context=cstech

Příloha A

Obsah DVD

- Bin

  Obsahuje zkompilovaný program

- Src

  Obsahuje zdrojové soubory

- Input

  Obsahuje použité testovací soubory

- Output

  Obsahuje výsledky testování

- Práce.pdf

  Text této práce v digitální podobě

- Readme.txt

  Podrobnější popis obsahu DVD