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

Diplomová práce

Rozpoznávání význačných rysů triangularizovaných modelů

Plzeň, 2014 Lukáš Karlíček

Originální zadání

Prohlášení

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

V Plzni dne ....

Lukáš Karlíček

Poděkování

Především bych rád poděkoval Prof. Dr. Ing. Ivaně Kolingerové za odbornou pomoc, cenné připomínky a čas strávený konzultacemi v průběhu vytváření této diplomové práce. Dále bych chtěl poděkovat Ing. Tomáši Bayerovi, Ph.D. a Ing. Jakubovi Šilhavému za ochotu při konzultaci navržené metody.

Abstract

Feature Recognition on Triangulated Models

Feature detection is used in various fields such as geoinformatics or computer graphics. A lot of methods for detection of feature lines on regular object representations (e.g., 2D images or regular grids) have been developed in the past years but much less methods have been created for triangle meshes. The goal of this master thesis is to propose, implement and test a method for recognition of feature lines from unstructured triangle models. This method is based on the use of discrete curvature analysis and morphological operators.

Keywords

feature line, skeleton, curvature, thresholding, morphological operators, triangle model

Anotace

Detekce význačných rysů je využívána v různých odvětvích, např. v geoinformatice nebo počítačové grafice. V průběhu let vzniklo mnoho metod pro detekci hran založených na pravidelné reprezentaci objektu (tj. 2D obrázek nebo pravidelná mřížka), ale pro trojúhelníkové sítě existuje méně vhodných metod. Cílem této diplomové práce je navržení, implementace a otestování metody pro rozpoznávání význačných hran z nestrukturovaných trojúhelníkových modelů. Metoda je založena na výpočtu diskrétní křivosti a morfologických operátorech.

Klíčová slova

význačná hrana, kostra, křivost, prahování, morfologické operátory, trojúhelníkový model

Obsah

1 Úvod

Hledání význačných rysů spočívá v nalezení charakteristických částí objektu, kterých si lidské oko při pohledu na model všimne. Pro člověka je tento proces samozřejmostí a dělá ho automaticky, avšak počítači je nutné předložit konkrétní postup, kterým se má řídit. Vytvoření korektního algoritmu je obtížnou záležitostí z důvodu tvarové rozmanitosti modelů. Objekty vyrobené člověkem často obsahují mnoho míst, kde dochází k velkému zakřivení. Příkladem může být nábytek, technické součástky nebo budovy. Naproti tomu v reálném terénu se taková místa vyskytují pouze sporadicky (např. útesy). Častěji to bývá tak, že je terén pozvolný a hrana vzniká u přilehlých svahů. Typickou ukázkou jsou hřebeny hor nebo údolí. Velmi snadno se také může stát, že model obsahuje místa s velkým i malým zakřivením. Příkladem nechť je model terénu obsahující i budovy. Pokud navíc uvažujeme objekt, který je v některé jeho charakteristické části poškozen, dostáváme se do problémů. Lidský mozek je schopný ve většině případů snadno rozpoznat původní tvar. Pro detekční algoritmy je to ovšem velká překážka, protože neumožňují celkový pohled na objekt, ale používají pouze lokální okolí.

S detekcí hran se lze setkat ve více různorodých oborech. V počítačové grafice se používá pro analyzování tvaru objektu, jeho rozpoznávání nebo ohodnocení kvality. Význačné hrany lze dále využít při vyhlazování nebo zjednodušování modelu. Do samostatné kategorie patří nerealistické zobrazování, které je na význačných rysech založeno. Druhým příkladem oblasti může být geoinformatika nebo kartografie. V těchto oborech se většinou pracuje pouze s terénními daty. Jeden z jejich hlavních úkolů je vyhledávání terénních linií (např. údolnic nebo hřbetnic), které lze následně zakreslit do mapy.

V průběhu let vzniklo mnoho metod pro hledání význačných rysů. Většina z nich je založena na pravidelné reprezentaci objektu (tj. výšková mapa ve formě šedotónového obrázku nebo pravidelná mřížka), která je velmi využívána v geoinformatice nebo kartografii. V počítačové grafice se ovšem častěji setkáváme s reprezentací modelu pomocí trojúhelníkové sítě. Použitelných algoritmů založených na této reprezentaci pro detekci význačných hran na komplikovaných a rozsáhlých modelech neexistuje mnoho.

Cílem diplomové práce je tedy navržení, implementování a otestování metody pro detekci význačných hran na trojúhelníkových modelech. Tato metoda by měla být schopná zpracovávat jak data terénní, tak i trojrozměrná. Navržená metoda nejprve ohodnotí jednotlivé vrcholy trojúhelníkové sítě pomocí diskrétní křivosti. Na získané hodnoty je aplikováno prahování, čímž dojde k vytvoření význačné oblasti. Následně je možné pomocí morfologických operátorů redukovat šum nebo odstranit nežádoucí artefakty (např. díry uvnitř oblasti). Dále lze použít operaci skeletonizace, a tím získat význačné hrany.

2 Teoretický úvod

V této kapitole budou popsány základní teoretické definice a poznatky z vědních oblastí, jež se diplomová práce dotýká. Jedná se zejména o geoinformatiku nebo kartografii, kde hledání význačných terénních hran je jednou z jejich hlavních aplikací. Dále pak oblast diferenciální geometrie z důvodu, že mnoho metod pro detekci hran je založeno na výpočtu křivostí ploch. Poslední popsanou oblastí je matematická morfologie, která je používána v námi vybrané metodě.

2.1 Základní pojmy geoinformatiky a kartografie

Jestliže se budeme snažit vymezit pojem význačná hrana, tak lze říci, že je to místo na modelu, kde se náhle a výrazně mění tvar povrchu. Tato definice je obecně platná jak pro modely terénní, tak i pro trojrozměrné. Oblast geoinformatiky se ovšem zabývá pouze terény, proto jsou následující definice v této kapitole platné pouze pro terénní data.

Obecně lze říci, že zemský povrch je matematicky nevyjádřitelná plocha, neboť je nepravidelný a má velmi komplikovaný průběh. Jeho větší část je po zjednodušení hladká, avšak obsahuje i ostrá místa, kterými mohou být zlomy, zářezy, hrany nebo umělé terénní tvary. Právě tato místa znemožňují jednoduše popsat terén matematicky, a proto je nutné jej zjednodušit [1]. Pro popis terénu v digitální podobě a poskytnutí dalších operací nad terénem se používají následující prostorové modely zemského povrchu [2]:

- ***Digitální model terénu (DMT)*** – model povrchu Země v digitální podobě bez staveb, vegetace a dalších objektů nacházejících se na jeho povrchu, který lze zpracovat prostředky informačních a komunikačních technologií. Je zřejmé, že tento model je velmi zjednodušený oproti reálnému povrchu. Proto se zobrazuje pouze v definované přesnosti a podrobnosti. V některých literaturách se také uvádí pojem *digitální model reliéfu* (*DMR*), což je synonymum pro *DMT*.
- ***Digitální výškový model*** – je jednou z variant *DMT*. Uvádí se ve formě rastru se zadanou hustotou bodů, kde každý bod představuje nadmořskou výšku. V angličtině se často uvádí termín *Digital Elevation Model* (*DEM*).
- ***Digitální model povrchu*** – model povrchu Země, ovšem se všemi objekty, které se na něm nacházejí. Vzniká použitím automatizovaného sběru bodů pomocí obrazové korelace ve fotogrametrii[1], laserového skenování nebo radarového měření v *dálkovém průzkumu Země* (*DPZ*). Následným zpracováním digitálního modelu povrchu lze získat *DMT*. Jeho hlavním využitím je vizualizace a modelování měst nebo krajiny včetně vegetace.

[1] Fotogrametrie se zabývá rekonstrukcí tvarů, měřením rozměrů a určováním polohy předmětů, které jsou zobrazeny na fotografických snímcích.

Výše uvedené digitální modely mohou obsahovat mnoho kartografických technik, pomocí kterých lze vyjádřit tvar terénu nebo jeho průběh (např. vrstevnicový model, barevná hypsometrie[2] nebo terénní šrafy) [1]. S použitím *DMT* se lze setkat v mnoha různorodých oborech a jeho význam stále stoupá. Příkladem mohou být následující odvětví [2]:

- ***Vědy o Zemi*** **–** do této kategorie lze zařadit studie vlivu klimatu, geologické a hydrologické modelování, geomorfologickou[3] analýzu, hydrologické analýzy odtoku vody, analýzy říčních koryt a mnoho dalších.
- ***DPZ a mapování*** – v této kategorii je *DMT* použit společně s *GIS* (*geografický informační systém*) nástroji k úpravě snímků a získání tematické informace s ohledem na geometrii reliéfu.
- ***Stavebnictví*** – *DMT* je využíváno při projektování silnic, železnic, přehrad, nádrží, pozemních úprav a těžby. Dále pak k trojrozměrnému modelování území, prezentaci projektů nebo vizualizacím a objemovým výpočtům.
- ***Plánování a management zdrojů*** – skupina obsahující obory jako zemědělství, klimatologie, meteorologie nebo lesnictví. Jejich hlavním cílem je spravování přírodních zdrojů. Příkladem aplikace může být výběr vhodného místa pro osev obilovin.
- ***Vojenské aplikace*** – vojenské složky jsou jedny z hlavních uživatelů *DMT* a jejich kvalita je pro ně více než důležitá. Obecně lze říci, že jsou největší producenti digitálních modelů povrchu a *DMT*. Digitální modely používají pro analýzu viditelnosti na bojišti, simulaci letu nebo 3D zobrazení navádění zbraní.

Pro reprezentaci terénu se v geoinformatice a *GIS* systémech nejčastěji používá funkce dvou proměnných $z = f(x, y)$, kde souřadnice z určuje výšku v bodě $[x, y]$. Velmi často se lze také setkat s označením 2.5D. Nevýhodou je, že neumožňuje vyjádřit speciální terénní tvary, jako jsou jeskyně nebo převisy. Zřejmým rozšířením je použití funkce tří proměnných, kde souřadnice z již není pouhým atributem a jednotlivé části plochy jsou vyjádřeny parametrickými rovnicemi. Takto definovaný terén už umožňuje vyjádřit jakýkoliv tvar. Poslední používanou reprezentací je 2D, kde jsou uvažovány pouze souřadnice x, y [1].

Při tvorbě *DMT* je na počátku neuspořádaná množina dat, která je získána např. pomocí laserového skenování zemského povrchu. Tuto množinu je nutné uspořádat, vytvořit mezi daty topologické vztahy a určit vhodnou interpolační funkci, která bude umožňovat odvození výšky v neznámých bodech. Získaná data je nutné uložit do vhodných datových struktur s použitím různých datových modelů. Mezi nejvíce používané datové modely patří [3]:

[2] Barevná hypsometrie se zabývá vyjadřováním výškopisu pomocí barevných odstínů, což je obvyklé u fyzických map.

[3] Geomorfologie se zabývá studiem tvarů, vzniku a stáří zemského povrchu.

- ***Rastrový model terénu*** – tvořen maticí bodů, kde jeden konkrétní bod představuje čtvercovou plošku s konstantní nadmořskou výškou. Pro ukládání dat se velmi často používají klasické rastrové obrazové formáty (např. *png*). Přesnost reprezentace terénu silně závisí na vzdálenosti bodů v rastru, přičemž při velké vzdálenosti nemusejí být správně vystiženy některé důležité prvky terénu, jako je údolí nebo hřbety. Tento nepříjemný fakt je možné napravit zmenšením vzdálenosti bodů. Tím ovšem dochází ke zvýšení objemu dat a zbytečné redundanci v místech, kde požadovaná přesnost nemá smysl. Z toho je zřejmé, jaké jsou nevýhody tohoto datového modelu. Mezi jeho hlavní výhody patří jednoduchost algoritmů. Obecně se dá říci, že je vhodný pro modelování plochých terénů, které neobsahují náhlé změny.
- ***TIN (Triangulated Irregular Network)*** – terén je tvořen pomocí nepravidelné trojúhelníkové sítě. V geoinformatické terminologii se občas tento datový model také nazývá jako vektorový *DMT*. Jeho velkou výhodou je možnost nerovnoměrného rozmístění vstupních dat tak, aby co nejlépe vystihovaly tvar terénu. To znamená, že v rozmanitých místech lze použít trojúhelníkovou síť hustší, naopak v rovinných částech použijeme síť řidší. Tím dochází ke značné redukci objemu dat oproti rastrovému modelu terénu. Výsledná kvalita trojúhelníkové sítě (tj. zda dobře aproximuje skutečný terén) také závisí na způsobu vytvoření trojúhelníků. Nejčastěji se používá *Delaunayova* triangulace, která se snaží vytvářet co nejvíce rovnostranné trojúhelníky. Mezi hlavní výhody *TIN* (kromě výše zmíněných) patří možnost reprezentace terénu obsahující díry, nepravidelné hranice nebo svislé srázy. Dále se zde neprovádějí žádné interpolace, ale pracuje se přímo se vstupními daty. *TIN* má menší paměťové nároky a některé analýzy jsou přesnější. Oproti rastrovému přístupu má nevýhodu v komplikovanějších algoritmech.
- ***Vrstevnicový model terénu*** – velmi známý způsob reprezentace terénního reliéfu. Jeho přesnost je závislá na způsobu získání vrstevnic. Při použití fotogrametrie je kvalita obvykle vysoká, přičemž při výpočtu vrstevnic z bodových dat je nižší. Terén mezi jednotlivými vrstevnicemi není definován, proto je nutné jej interpolovat. Při analýze terénu často dochází k převodu z vrstevnicového modelu do rastru nebo *TIN*, protože obvykle nejsou dostupné žádné nástroje, které umožňují provádět analýzu v tomto datovém modelu.
- ***Plátový model terénu*** – tato reprezentace nahrazuje reálný terén pomocí většího množství plátů, které jsou na sebe napojeny. Nejčastěji se používají trojúhelníkové nebo čtyřúhelníkové kubické pláty. Jejich napojení může být hladké nebo ostré v závislosti na způsobu provázání řídících bodů. Plátový model se využívá spíše kvůli lepší vizualizaci terénu než pro analýzu.

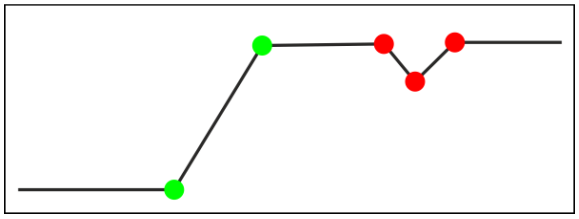
Pojem význačná hrana lze bez problémů použít v oblasti 3D modelů, kde neexistuje další terminologie zabývající se rozdělením hran do různých kategorií, ovšem v oboru geoinformatiky se tento pojem obecně nepoužívá. V průběhu let si geoinformatika vytvořila nepřeberné množství definic různých typů terénních křivek, přičemž zde budou uvedeny pouze základní [4]:

- ***Terénní kostra*** – soubor terénních čar, na kterých se stýkají jednotlivé terénní plochy a body terénní kostry.
- ***Spádnice*** – čára vybíhající na obě strany ze hřbetnice ve směru největšího spádu na úbočích. V každém svém bodě je kolmá k vrstevnicím procházející těmito body. S její pomocí je možné určovat směr tekoucí vody nebo měřit rozestup vrstevnic.
- ***Hřbetnice*** – spojnice průmětů relativně nejvyšších bodů vypuklé terénní plochy. Ze všech spádnic má v oblasti hřbetu nejmenší sklon a ostatní spádnice se od ní rozbíhají.
- ***Údolnice*** – čára spojující průměty nejníže položených bodů terénních útvarů. Pomocí ní se určuje směr vodního toku. Oproti hřbetnici se k ní ostatní spádnice sbíhají.
- ***Terénní hrana*** – spojuje místa, kde se výrazně mění sklonové poměry. Někdy je označována jako „čára nespojitosti dvou terénních ploch“.
- ***Body terénní kostry*** – nejnižší místa prohlubní nebo nejvyšší místa vypuklých ploch. Příkladem může být střed sedla, styk údolnic či vrchol kupy.

Při rozpoznávání terénních křivek je velkým problémem definování míry podrobnosti, se kterou má být vyhledávání provedeno. Ta závisí zejména na vstupních datech a účelu, ke kterému mají vyhledané křivky sloužit. Pokud uvažujeme jako datovou reprezentaci terénu *TIN*, lze v nejhorším případě říci, že každá hrana v trojúhelníkové síti je význačná. Tato informace pro nás ovšem není moc cenná. Z tohoto důvodu je důležité nejprve určit prahovou hodnotu, která bude vybírat pouze takové hrany, které jsou pro daný účel zajímavé. Nastavený práh by měl zohledňovat dva základní parametry:

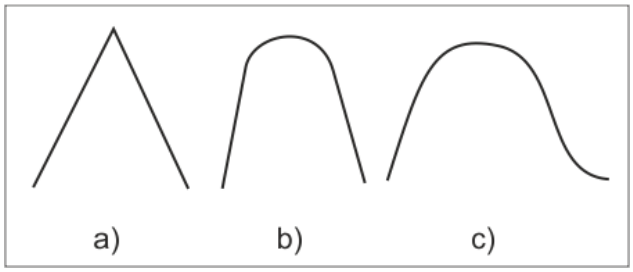
1. ***Úhel mezi svahy tvořících hranu*** – hrana vzniká v místě, kde spolu sousedí dva přilehlé svahy. Pro vymezování terénních křivek je důležité stanovit maximální (popř. minimální) úhel mezi svahy, při kterém bude hrana ještě považována za význačnou.
2. ***Délka svahů tvořících hranu*** – podle účelu vymezování terénních křivek mohou být důležité pouze takové hrany, které mají určitou minimální (popř. maximální) délku.

Je však nutné podotknout, že ani tyto dva parametry nejsou dostačující při vymezování význačných hran. Možný problém je zobrazen na Obr. 2.1.



Obr. 2.1: Problém při vymezování terénních křivek.

Výše uvedený obrázek znázorňuje situaci, kdy zelené hrany splňují základní kritéria (tj. úhel a délka hrany) a červené nikoliv. Z globálního hlediska jsou tyto hrany ovšem také význačné. Tento problém se nejčastěji vyskytuje u lokálně orientovaných algoritmů, ve kterých se zohledňuje pouze jejich nejbližší okolí. Jiným problémem, který velmi často nastává, je rozpad terénní hrany na více kratších nenavazujících částí, jež by měla být reprezentována jednou čarou. Tento defekt je způsoben různou proměnlivostí terénu ve svých odlišných částech, přičemž vybíráme hrany, které jen těsně splňují výše uvedené podmínky, zatímco hrany, které tuto podmínku jen těsně nesplňují, nevybereme. Další nepříjemnou vlastností reálných terénů je, že téměř vůbec neobsahují terénní křivky tvořené ostrým zlomem. Mnohem častěji je hrana pozvolná, což vytváří oblouk s různě velkým poloměrem. Příklad takové situace je nastíněn na Obr. 2.2.



Obr. 2.2: Ukázka hřebenů zobrazených v příčném profilu.

Z obrázku je zřejmé, že varianta *a)* je v reálném terénu raritou, zatímco varianta *c)* je velmi častá. Při vymezování terénních křivek je tedy nutné počítat i s poloměrem oblouku. Výše popsané problémy lze samozřejmě zobecnit i pro 3D modely, ve kterých se podobné situace také vyskytují.

2.2 Základy diferenciální geometrie

Tato kapitola je vytvořena na základě literatury [5-10].

2.2.1 Diferenciální geometrie křivek

V diferenciální geometrii se pro popis křivek používají vektorové funkce jedné reálné proměnné, které je možné definovat následovně:

Nechť $I \subset \mathbb{R}$. *Zobrazení* $\boldsymbol{P}: I \to \mathbb{R}^n$ *se nazývá vektorová funkce jedné reálné proměnné s definičním oborem* I.

Derivaci vektorové funkce $\boldsymbol{P}(t)$ v bodě t_0 lze vyjádřit ve tvaru:

$$\dot{\boldsymbol{P}}(t_0) = \lim_{t \to t_0} \frac{\boldsymbol{P}(t) - \boldsymbol{P}(t_0)}{t - t_0}. \tag{2.1}$$

Dále definujeme třídu C_n vektorové funkce $\boldsymbol{P}(t)$:

Řekneme, že vektorová funkce $\boldsymbol{P}(t)$ je třídy C_n $(n \in \mathbb{N})$, jestliže je na dané množině spojitá i se svými derivacemi až do řádu n.

Nyní již můžeme formulovat pojem regulární křivky:

Regulární křivkou třídy C_n v $\boldsymbol{E}^3$ rozumíme množinu $k \subset \boldsymbol{E}^3$, pro kterou existuje vektorová funkce $\boldsymbol{P}(t),\ t \in I$ tak, že

1. $\boldsymbol{P}: I \to k, I$ *je otevřený interval,*
2. $\boldsymbol{P}$ *je třídy* C_n,
3. $\left|\dot{\boldsymbol{P}}(t_0)\right| \neq 0$ *pro všechna* $t_0 \in I$,
4. $t_1 \neq t_2 \Rightarrow \boldsymbol{P}(t_1) \neq \boldsymbol{P}(t_2)$.

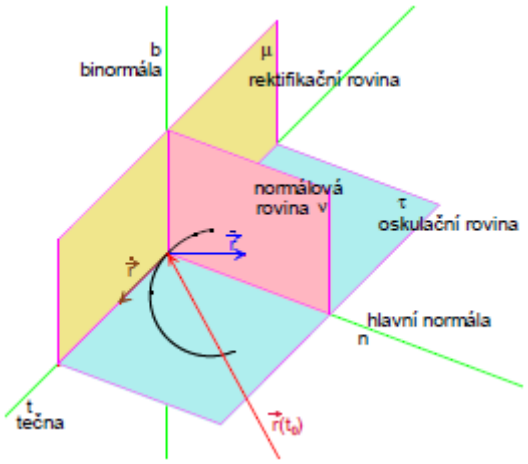
Rozepsáním křivky $\boldsymbol{P}(t)$ do jednotlivých složek dostáváme její parametrické vyjádření, se kterým se v diferenciální geometrii pracuje nejčastěji. Body na obecné křivce (tj. křivka nemusí být regulární), pro které platí $\left|\dot{\boldsymbol{P}}(t_0)\right| = 0$, se nazývají singulární. Vektor $\dot{\boldsymbol{P}}(t_0)$ je označován tečným vektorem křivky $\boldsymbol{P}(t), t \in I$ v bodě t_0. Tečnou křivky v tomto bodě poté rozumíme přímku $\boldsymbol{R}(m) = \boldsymbol{P}(t_0) + m\dot{\boldsymbol{P}}(t_0)$. Jinak řečeno, tečna křivky je přímka spojující její dva nekonečně blízké body okolo bodu t_0. Platí, že regulární křivka má v konkrétním bodě t_0 pouze jedinou tečnu. Nyní definujeme pojem oskulační roviny:

Nechť $\boldsymbol{P}(t),\ t \in I$ je regulární křivka a nechť vektory $\dot{\boldsymbol{P}}(t_0)$ a $\ddot{\boldsymbol{P}}(t_0)$ nejsou kolineární. Potom rovinu $\boldsymbol{R}(u, v) = \boldsymbol{P}(t_0) + u\dot{\boldsymbol{P}}(t_0) + v\ddot{\boldsymbol{P}}(t_0)$ nazýváme oskulační rovinou křivky v daném bodě.

Oskulační rovinu lze chápat jako rovinu, která se v daném bodě ke křivce nejvíce přimyká (tj. rovina proložená třemi nekonečně blízkými body okolo bodu t_0). Pro rovinnou křivku tedy platí, že celá leží v oskulační rovině. Jestliže na křivce existuje bod, pro který jsou $\dot{\boldsymbol{P}}(t_0)$ a $\ddot{\boldsymbol{P}}(t_0)$ kolineární (vektory jsou lineárně závislé), pak se označuje jako inflexní. Kromě tečny a oskulační roviny existují ještě další významné vektory a roviny:

- ***Normála křivky*** – v konkrétním bodě každá přímka $\boldsymbol{R}(m) = \boldsymbol{P}(t_0) + m \cdot \boldsymbol{n}$, kde $\dot{\boldsymbol{P}}(t_0) \cdot \boldsymbol{n} = 0$.
- ***Hlavní normála*** – normála ležící v oskulační rovině.
- ***Binormála*** – normála, která je kolmá k oskulační rovině.
- ***Rektifikační rovina*** – rovina určená bodem na křivce, tečným a binormálovým vektorem.
- ***Normálová rovina*** – rovina určená bodem na křivce, binormálovým a normálovým vektorem.

Výše uvedené vektory a roviny jsou přehledně vyobrazeny na Obr. 2.3.



Obr. 2.3: Významné vektory a roviny používané v diferenciální geometrii (převzato z [5]).

Velmi důležitým prvkem, který charakterizuje křivku, je její křivost. Pro naše účely vymezování význačných hran je křivost zásadní, neboť udává míru zakřivení křivky v konkrétním bodě. V oblasti diferenciální geometrie křivek se používají dvě různé křivosti – první a druhá. První křivost neboli flexi si lze představit následovně: při pohybu bodu po křivce se mění také směr tečny. Rychlost změny směru tečny udává velikost zakřivení křivky. Čím více se v okolí bodu křivka odklání od tečny, tím větší má v tomto bodě první křivost. Pro výpočet lze využít vztah

$$k_1^2 = \frac{\left|\dot{\boldsymbol{P}} \times \ddot{\boldsymbol{P}}\right|^2}{\left(\dot{\boldsymbol{P}} \cdot \dot{\boldsymbol{P}}\right)^3}. \tag{2.2}$$

Druhá křivost neboli torze představuje vychýlení křivky z oskulační roviny v konkrétním bodě. Lze ji vypočítat jako

$$k_2 = \frac{\left(\dot{\boldsymbol{P}}, \ddot{\boldsymbol{P}}, \dddot{\boldsymbol{P}}\right)}{\left|\dot{\boldsymbol{P}} \times \ddot{\mathbf{P}}\right|^2}. \tag{2.3}$$

Pomocí výše uvedených křivostí lze jednoduše určit zajímavé vlastnosti křivek:

- Je-li $k_1(t) = 0, \forall t \in I$, potom $\boldsymbol{P}(t)$ je úsečka (popř. přímka).
- Je-li $k_2(t) = 0$ a $k_1(t) \neq 0, \forall t \in I$, potom $\boldsymbol{P}(t)$ je rovinná křivka.
- Je-li $k_2(t) = 0, \forall t \in I$ a k_1 je nenulová konstanta, potom $\boldsymbol{P}(t)$ je oblouk kružnice o poloměru $1/k_1$.

2.2.2 Diferenciální geometrie ploch

Nejprve opět uvedeme definici vektorové funkce, která je používána v diferenciální geometrii ploch (vztahuje se pouze na prostor $\boldsymbol{E}^3$):

Vektorová funkce $\boldsymbol{P}(u^1, u^2)$ *proměnných* $u^1, u^2 \in \Omega$ *je zobrazení množiny* $\Omega \subset \mathbb{R}x\mathbb{R}$ *do vektorového prostoru* $\boldsymbol{E}^3$.

Parciální derivace vektorové funkce $\boldsymbol{P}(u^1, u^2)$ je definována následovně:

$$\frac{\partial \boldsymbol{P}}{\partial u^1} = \lim_{h \to 0} \frac{\boldsymbol{P}(u^1 + h, u^2) - \boldsymbol{P}(u^1, u^2)}{h} = \left(\frac{\partial x}{\partial u^1}, \frac{\partial y}{\partial u^1}, \frac{\partial z}{\partial u^1} \right),$$
$$\frac{\partial \boldsymbol{P}}{\partial u^2} = \lim_{h \to 0} \frac{\boldsymbol{P}(u^1, u^2 + h) - \boldsymbol{P}(u^1, u^2)}{h} = \left(\frac{\partial x}{\partial u^2}, \frac{\partial y}{\partial u^2}, \frac{\partial z}{\partial u^2} \right). \quad (2.4)$$

Dále uvedeme formulaci regulární plochy:

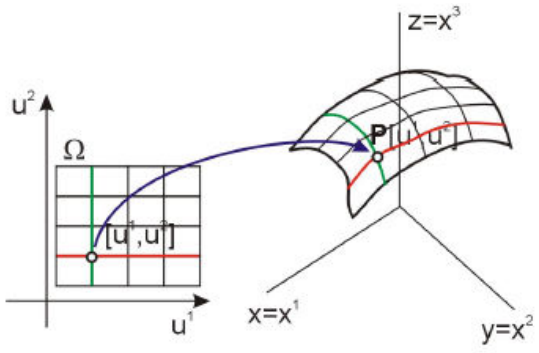
Regulární plochou třídy C_n v $\boldsymbol{E}^3$ rozumíme množinu $\mathcal{P} \subset \boldsymbol{E}^3$, pro niž existuje vektorová funkce $\boldsymbol{P}(u^1, u^2), (u^1, u^2) \subset \Omega$, kde Ω je oblast (otevřená kompaktní množina), taková že

1. $\boldsymbol{P}: \Omega \to \mathcal{P}$ *je zobrazení na množinu,*
2. $\boldsymbol{P}$ *je třídy* C_n $(n \geq 3)$,
3. $\frac{\partial \boldsymbol{P}}{\partial u^1}$ *a* $\frac{\partial \boldsymbol{P}}{\partial u^2}$ *jsou lineárně nezávislé ve všech bodech oblasti* Ω,
4. $(u_0^1, u_0^2) \in \Omega, (u_1^1, u_1^2) \in \Omega$ *a* $(u_0^1, u_0^2) \neq (u_1^1, u_1^2) \Longrightarrow \boldsymbol{P}(u_0^1, u_0^2) \neq \boldsymbol{P}(u_1^1, u_1^2)$.

Z výše uvedené definice vyplývá, že pro regulární plochu platí $\left| \frac{\partial \boldsymbol{P}}{\partial u^1} \times \frac{\partial \boldsymbol{P}}{\partial u^2} \right| \neq 0$ ve všech jejich bodech. Pokud plocha není regulární, pak body, pro které neplatí tento výraz, se označují singulární. Nyní formulujeme pojem křivky na ploše:

Nechť je dána plocha určená vektorovou funkcí $\boldsymbol{P}(u^1, u^2)$ na oblasti Ω a nechť jsou dány funkce $\alpha^1(t)$ a $\alpha^2(t)$, $t \in I$ určující křivku v Ω, pak $\boldsymbol{P}(\alpha^1(t), \alpha^2(t))$ nazýváme křivkou na ploše.

Pro lepší představu si lze prohlédnout Obr. 2.4.

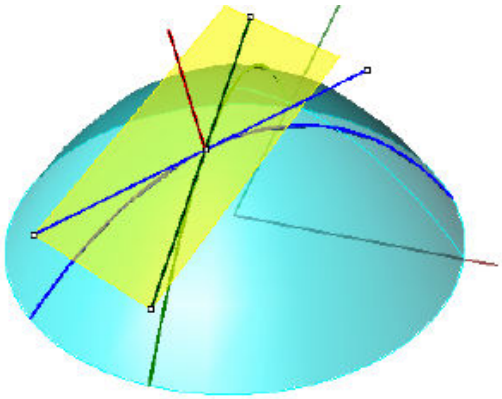


Obr. 2.4: Zobrazení křivky na ploše (převzato z [8]).

Tečnou plochy označujeme přímku, která se dotýká některé křivky ležící na ploše. Dále platí, že všechny tečny regulárních křivek na regulární ploše v daném bodě leží na jedné rovině, kterou označujeme tečnou. Tuto rovinu lze vyjádřit jako

$$\boldsymbol{R}(\alpha^1, \alpha^2) = \boldsymbol{P} + \alpha^1 \boldsymbol{P}_1 + \alpha^2 \boldsymbol{P}_2, \quad (2.5)$$

kde $\boldsymbol{P}_1 = \frac{\partial \boldsymbol{P}}{\partial u^1}$ a $\boldsymbol{P}_2 = \frac{\partial \boldsymbol{P}}{\partial u^2}$. Přímku $\boldsymbol{R}(t) = \boldsymbol{P} + t(\boldsymbol{P}_1 \times \boldsymbol{P}_2)$ nazýváme normálou plochy a vektor $\lambda(\boldsymbol{P}_1 \times \boldsymbol{P}_2)$, $\lambda \neq 0$, normálovým vektorem v konkrétním bodě. Vše je zobrazeno na Obr. 2.5.



Obr. 2.5: Ukázka tečné roviny, normály, tečen a křivek na ploše (převzato z [8]).

Nyní již můžeme popsat výpočet jednotlivých křivostí ploch. První uvedenou křivostí je tzv. ***normálová křivost***, která hledá souvislost mezi plochou a křivostí křivky na ploše. Pomocí ní je možné jednoduše určit křivost v konkrétním směru. Uvažujme tedy křivku na ploše $\boldsymbol{P}(s) = \boldsymbol{P}\big(u^1(s), u^2(s)\big)$, která je parametrizována obloukem. Pro její první křivost platí $\ddot{\boldsymbol{P}} = k_1 \cdot \boldsymbol{v}$, kde $\boldsymbol{v}$ je jednotkový vektor hlavní normály křivky. Je nutné zdůraznit, že tento vztah platí pouze pro křivku parametrizovanou obloukem. Normálovou křivostí křivky v bodě poté rozumíme velikost k_n kolmého průmětu (se znaménkem) vektoru $\ddot{\boldsymbol{P}}$ do jednotkového vektoru $\boldsymbol{n}$ normály plochy:

$$k_n = \ddot{\boldsymbol{P}} \cdot \boldsymbol{n}. \tag{2.6}$$

Pokud označíme $\gamma = \angle(\boldsymbol{n}, \boldsymbol{v})$, pak lze také psát

$$k_n = k_1 \cdot \cos\gamma. \tag{2.7}$$

Z výše uvedené definice je patrné, že znaménko normálové křivosti závisí na orientaci normálového vektoru plochy. Dále lze dokázat, že normálová křivost všech křivek plochy se společnou tečnou v daném bodě je stejná. To určuje význam této křivosti jako vlastnosti tečny křivek na ploše - směru plochy. Rozlišujeme dva základní směry v konkrétním bodě:

- ***Hlavní*** – je-li normálová křivost extremální (tj. minimální nebo maximální).
- ***Asymptotický*** – je-li normálová křivost nulová.

Hlavní křivkou poté označujeme křivku, jejíž tečna leží v hlavním směru v každém jejím bodě. Na základě již definované normálové křivosti je možné jednoduše formulovat ***hlavní křivosti*** - extrémní hodnoty normálové křivosti v konkrétním bodě označené k_n^{min} a k_n^{max} (normálové křivosti v hlavních směrech). Lze dokázat, že v konkrétním bodě jsou všechny směry hlavní nebo existují právě dva, které jsou na

sebe kolmé. Pokud jsou všechny směry hlavní, pak platí $k_n^{min} = k_n^{max}$. Uvedeme příklad speciálních ploch a jejich hlavních křivostí:

- ***Rovina*** – všechny směry hlavní s normálovou křivostí rovné nule, tedy $k_n^{min} = k_n^{max} = 0$.
- ***Kulová plocha*** – všechny směry hlavní s normálovou křivostí $k_n = {}^{1}/_{r}$, kde r je poloměr kulové plochy $\left(k_n^{min} = k_n^{max} = {}^{1}/_{r}\right)$.
- ***Rotační válcová plocha*** – v každém jejím bodě platí $k_n^{min} = 0$ a $k_n^{max} = {}^{1}/_{r}$.

Hlavní křivosti vymezují rozsah zakřivení plochy v okolí konkrétního bodu. Lze z nich v omezené míře vyvozovat závěry o tvaru plochy, ovšem hlavně se využívají pro definici dalších typů křivosti. ***Gaussovou křivostí*** plochy v daném bodě rozumíme číslo

$$G = k_n^{min} \cdot k_n^{max}. \tag{2.8}$$

Z definice Gaussovy křivosti vyplývá, že je invariantní vůči změně orientace plochy (opačný normálový vektor). Při změně orientace se změní znaménko normálové křivosti, avšak hlavní křivosti stále zůstanou extrémy. Jejich násobek poté zajistí neměnnost Gaussovy křivosti. Podle znaménka v daném bodě plochy je možné rozlišovat tři typy bodů na ploše:

- ***Parabolické*** – je-li $G = 0$.
- ***Eliptické*** – je-li $G > 0$.
- ***Hyperbolické*** – je-li $G < 0$.

Velký význam Gaussovy křivosti spočívá v definici rozvinutelné plochy:

Plocha je rozvinutelná právě tehdy, když má Gaussovu křivost nulovou ve všech svých bodech.

Tato věta má praktické uplatnění např. při konstrukci vzduchotechnického potrubí nebo přechodových dílů. Poslední námi popsanou křivostí je ***střední křivost*** plochy, kterou lze vypočítat jako

$$H = \frac{k_n^{min} + k_n^{max}}{2}. \tag{2.9}$$

Z definice střední křivosti je patrné, že jsou opět využívány hlavní křivosti. Na rozdíl od Gaussovy křivosti, střední křivost mění znaménko při změně orientace plochy. Pokud platí, že $H = 0$ v každém bodě plochy, pak ji nazýváme minimální. Z názvu je zřejmé, že minimální plochy jsou takové, které mají nejmenší možný obsah na daných okrajích. Příkladem takové plochy může být např. katenoid nebo helikoid (viz [10]).

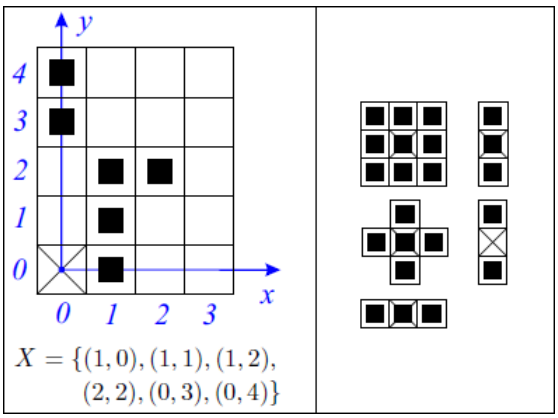
2.3 Matematická morfologie

Pojem matematické morfologie se poprvé objevil v šedesátých letech dvacátého stolení, kdy se používala k analýze binárních obrazů. Obecně lze říci, že je to technika pro zpracování geometrických struktur, která je založená na teorii množin. Je vhodná např. pro analyzování tvaru objektů. Hlavní myšlenkou je získání znalostí z relace obrazu a malého strukturního elementu, který má předdefinovaný tvar. V každém místě obrazu se porovnává, jak strukturní element odpovídá lokálním tvarům v obraze [11], [12].

Matematická morfologie se později dočkala zobecnění. Obraz je možné popsat pomocí bodových množin libovolné dimenze, např. $\boldsymbol{E}^2$ prostor a jeho podmnožiny jsou spojitým definičním oborem pro popis rovinných tvarů. V digitálních obrazech se spojitý $\boldsymbol{E}^2$ prostor reprezentuje pomocí celých čísel $\mathbb{Z}$. Nejznámější varianty matematické morfologie jsou následující [12]:

- ***Binární matematická morfologie ve 2D*** – bodová množina je vyjádřená pomocí dvojic celých čísel $(x, y) \in \mathbb{Z}_2$. Výskyt dvojice říká, že příslušný pixel je obsazený.
- ***Binární matematická morfologie ve 3D*** – bodová množina je vyjádřená pomocí trojic celých čísel $(x, y, z) \in \mathbb{Z}_3$, kde daná trojice informuje o obsazenosti příslušného voxelu.
- ***Šedotónová matematická morfologie ve 2D*** – bodová množina vyjádřená pomocí trojic celých čísel $(x, y, z) \in \mathbb{Z}_3$, kde x, y jsou souřadnice v obraze a g je hodnota šedi příslušného pixelu.

V našem případě se omezíme pouze na binární matematickou morfologii, jejíž definiční obor je $\mathbb{Z}_2$ a obor hodnot $\{0,1\}$. Příklad bodové množiny a strukturních elementů je zobrazen na Obr. 2.6.



Obr. 2.6: Levý obrázek: bodová množina v prostoru $\mathbb{Z}_2$. Pravý obrázek: příklady strukturních elementů v prostoru $\mathbb{Z}_2$. Křížek značí lokální počátek (převzato z [12]).

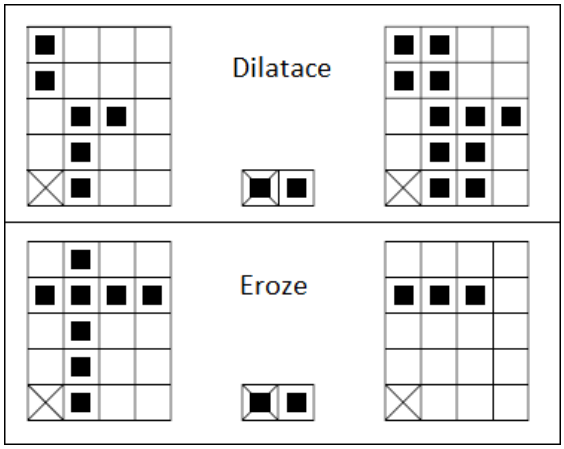
Základními operacemi matematické morfologie jsou ***dilatace*** a ***eroze***. Dilataci je možné vyjádřit jako

$$X \oplus B = \{p \in \boldsymbol{E}^2 : p = x + b, x \in X \wedge b \in B\}. \tag{2.10}$$

Lze si ji představit tak, že se na každý bod původního obrazu X obtiskne strukturní element B podle jeho lokálního počátku. Vzniklá množina je poté výsledkem dilatace, která se běžně používá k zaplnění malých děr nebo úzkých zálivů. Je zřejmé, že po použití tohoto operátoru se objekt zvětší. Chceme-li, aby byla zachována jeho původní velikost, je nutné použít další operátor zvaný eroze [12]. Ten je definován jako

$$X \ominus B = \{p \in \boldsymbol{E}^2 : p = x + b \in X \; pro \; \forall b \in B\}. \tag{2.11}$$

Opět si ho můžeme představit tak, že se na každý bod původního obrazu X obtiskne strukturní element B podle jeho lokálního počátku. Je-li celý strukturní element obsažen v původní množině X, zapíšeme do výsledného obrazu hodnotu 1, jinak 0. Eroze se velmi často používá ke zjednodušení struktury objektu, tj. rozložení na jednodušší části. Pokud jsou objekty menší než strukturní element, pak budou odstraněny (např. úzké čáry) [12]. Příklad dilatace a eroze je vidět na Obr. 2.7.



Obr. 2.7: Ukázkový příklad dilatace a eroze. Vlevo: původní obraz. Uprostřed: strukturní element. Vpravo: výsledný obraz (převzato z [12]).

Za pomoci eroze v kombinaci s množinovým rozdílem lze velmi jednoduše získat obrys objektu. Víme, že po použití eroze se objekt zmenší o body ležící na hranici. Pokud tedy odečteme výsledek operace eroze od původního obrazu, získáme obrys objektu. Matematicky to lze zapsat jako $\partial X = X \setminus (X \ominus B)$, kde ∂X značí výsledný obraz hranice. Dilatace a eroze jsou duální morfologické operace, nikoliv však inverzní. To znamená, že pokud na obraz použijeme erozi následovanou dilatací, nezískáme zpět původní množinu X. Proto jsou vytvořeny další operátory, které kombinují použití dilatace a eroze. Morfologický operátor eroze následované dilatací se nazývá ***otevření***. Lze jej napsat jako

$$X \circ B = (X \ominus B) \oplus B. \tag{2.12}$$

Prohozením dilatace a eroze ve vztahu (2.12) vzniká morfologický operátor zvaný ***uzavření***, který je definován jako

$$X \cdot B = (X \oplus B) \ominus B. \tag{2.13}$$

Uzavření se nejvíce používá pro spojení blízkých objektů a zaplňování děr, přičemž velikost původních objektů je zachována. Míra jejich spojení a zaplnění je dána velikostí a tvarem strukturního elementu. Morfologický operátor otevření se běžně používá k redukci šumu a rozpojení objektů, které jsou propojeny tenkou čarou. Oba tyto operátory se využívají ke snížení množství detailů v obraze při zachování základních tvarů objektů [11]. Ukázkový příklad je vidět na Obr. 2.8.

Obr. 2.8: Příklad použití morfologického operátoru otevření a uzavření (převzato z [13]).

Další morfologickou operací je operátor ***tref či miň***, který pracuje na principu vyhledávání shody mezi vstupním obrazem X a definovaným složeným strukturním elementem $B = \{B_1, B_2\}$. Sub-element B_1 obsahuje požadované body náležející objektu, B_2 požadované body náležející pozadí a na zbylých pozicích strukturního elementu nezáleží. Princip tohoto operátoru si můžeme představit tak, že zachovává ty body z množiny X, které odpovídají sub-elementu B_1 a současně neodpovídají B_2 [11]. Matematicky to lze vyjádřit jako

$$X \otimes B = \{p \in X : B_1 \subset X \wedge B_2 \subset X^c\}, \tag{2.14}$$

kde X^c značí doplněk množiny X. Složený strukturní element si můžeme vytvořit podle potřeby, ke které má operátor tref či miň sloužit (např. nalezení rohů objektu). Hlavní použití tohoto operátoru je ovšem pro operaci ***ztenčování*** nebo ***zesilování***. Ztenčování bodové množiny X složeným strukturním elementem B je definováno jako

$$X \oslash B = X \setminus (X \otimes B). \tag{2.15}$$

Pro morfologickou operaci zesilování platí

$$X \odot B = X \cup (X^c \otimes B). \tag{2.16}$$

Sekvenční ztenčování nebo zesilování je pak opakovaná aplikace prostého operátoru ztenčování nebo zesilování s posloupností $\{B_i\} = \{B_1, B_2, \dots\}$ složených strukturních elementů. Opakování sekvenčního ztenčování vede na čáry o šířce jednoho bodu, což se s výhodou používá jako aproximace skeletu (tento postup je jeden ze způsobů ***skeletonizace***) [11].

3 Přehled známých metod

V této kapitole bude uvedena rešerše vybraných metod pro hledání význačných rysů na terénech nebo trojrozměrných modelech.

3.1 Rastrově založené algoritmy

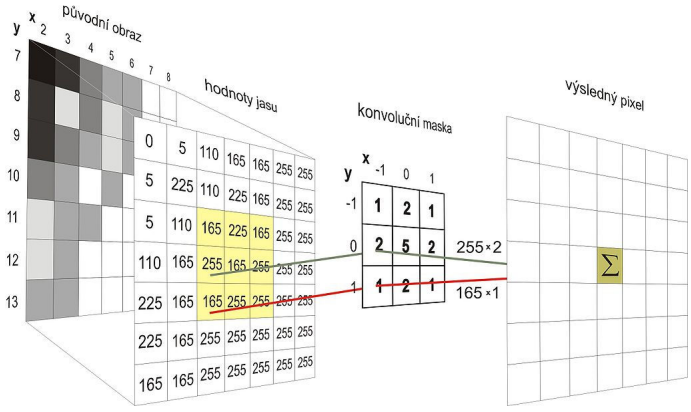
Rastrově založené algoritmy zaměřené na hledání význačných hran předpokládají jako vstup dvourozměrnou matici obsahující údaje o výšce. Nejčastěji je tato matice reprezentována ve formě šedotónového obrázku mající rozsah hodnot 0 až 255, který se také označuje jako výšková mapa. Z toho je zřejmé, že tyto metody pracují pouze s terény. Obecně lze říci, že mnoho rastrových algoritmů je postaveno na tzv. konvoluci, což je matematický operátor zpracovávající dvě funkce, značící se * a je definován jako

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\propto) g(x - \propto) d \propto. \tag{3.1}$$

Ve vztahu (3.1) se funkce $g(x-\propto)$ nazývá konvoluční jádro, které lze chápat jako okénko posouvající se po obraze. Obrazem je v terminologii konvoluce funkce $f(\propto)$. Jelikož je digitální obraz dvourozměrný a není spojitý, je nutné definovat dvourozměrnou diskrétní konvoluci jako

$$(f * g)(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x - i, y - j) \cdot h(i, j), \tag{3.2}$$

kde funkce $h(i,j)$ je konvoluční jádro a funkce $f(x-i, y-j)$ reprezentuje vstupní obrázek. Výpočet probíhá tak, že na každý pixel obrazu aplikujeme konvoluční jádro. Pro konkrétní pixel na pozici $[i,j]$ to znamená, že vynásobíme hodnoty konvoluční masky s odpovídajícími pixely v obrázku a vše sečteme. Výsledek poté udává novou hodnotu na pozici $[i,j]$ ve výsledném obrázku [14]. Vše je znázorněno na Obr. 3.1.



Obr. 3.1: Princip výpočtu konvoluce (převzato z [14]).

Hodnoty konvoluční masky definují typ operace, která bude na vstupní obrázek prováděna. Modifikovat je možné jak samotné hodnoty, tak i velikost nebo tvar masky. Základní varianta, která ve většině případů postačuje, je maska čtvercového tvaru velikosti 3x3.

3.1.1 Metody vycházející z první derivace

Význačná hrana v obrázku je vysokofrekvenční informace, kterou lze vyjádřit jako prudkou změnu jasu. Z toho je zřejmé, že právě tyto změny potřebujeme vyhledávat. Velká skupina metod je založena na první derivaci, protože ta nám dává maximální hodnoty u velkých změn, naopak minimální u malých. Hrany v obraze tedy odpovídají vysokým hodnotám derivace jasové funkce [14]. Z této úvahy je možné vyvodit, že u dvourozměrného obrázku budeme hledat gradient funkce dvou proměnných, který je definován jako

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right). \tag{3.3}$$

Z vypočteného gradientu lze jednoduše získat jeho velikost (což odpovídá velikosti hrany) a směr (gradient je kolmý na hranu):

$$\|\nabla f(x,y)\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}, \varphi = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right). \tag{3.4}$$

Parciální derivace dobře fungují na spojité funkce, ovšem obraz je reprezentován funkcí diskrétní. Je tedy potřeba tyto derivace aproximovat konečnými diferencemi. Nejjednodušší aproximace jednorozměrné diskrétní funkce v celočíselném bodě i je rovna $f'(i) \approx f(i) - f(i-1)$, kterou lze nazvat nesymetrickou derivací (ve skutečnosti odpovídá bodu $i - \frac{1}{2}$). V digitálním obraze takový bod neexistuje, proto byl vytvořen symetrický tvar $f'(i) \approx f(i+1) - f(i-1)$, který ovšem zanedbává vliv bodu na pozici i [14]. Oba vztahy je možné přepsat do formy konvoluce:

$$f' \approx [-1, +1] * f, f' \approx [-1, 0, +1] * f. \tag{3.5}$$

Ze vztahu (3.5) je názorně vidět, jak jednoduše lze za pomoci konvoluce vypočítat konečné diference pro celý obraz f. Nejjednodušší detekční metoda je pouhým rozšířením vztahu (3.5), který počítá konečné diference ve směru 0° (tj. hrany ve směru 90°). Za pomoci konvoluční masky velikosti 3x3 lze jednoduše napsat výpočet parciální derivace (resp. diference) podle x a y.

$$0^\circ = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, 90^\circ = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.6}$$

Po aplikování těchto masek je možné pomocí (3.4) dopočítat velikost gradientu v konkrétním bodě, která udává významnost dané hrany. Na stejném principu pracují i ostatní metody založené na první derivaci. ***Robertsův*** operátor používá pro detekci hran konvoluční masky velikosti 2x2 a jejich tvar je následující [14]:

$$R_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, R_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{3.7}$$

Z masek uvedených v (3.7) je patrné, že konečné diference jsou počítány ve směru 45° a 135°. To je z důvodu, že na matici 2x2 lze počítat pouze nesymetrické derivace. Takto vytvořené masky počítají derivace pro stejný bod ležící „uprostřed“ těchto masek [14]. Další metoda se nazývá ***Prewittové*** operátor, která využívá masky velikosti 3x3 ve formě:

$$P_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, P_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}. \tag{3.8}$$

U této metody je zřetelně vidět použití symetrické derivace. Dále si lze všimnout, že diference jsou počítány na větším okolí, což snižuje reakci na šum. Tyto vlastnosti jsou společné pro většinu z následujících metod.

Sobelův operátor:

$$S_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, S_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{3.9}$$

Robinsonův operátor:

$$Ro_1 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}, Ro_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{3.10}$$

Kirschův operátor:

$$K_1 = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}, K_2 = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}. \tag{3.11}$$

Výše uvedené postupy (počínaje Prewittové operátorem) se také často uvádějí ve formě rotujících masek, což je aproximace výpočtu velikosti gradientu. To znamená, že je definováno celkem osm masek, které počítají derivace ve všech možných směrech (v digitálním obrázku je jich právě osm). Poté jsou jednotlivé masky aplikovány na konkrétní bod a nejvyšší hodnota je prohlášena za velikost gradientu. Skupinu osmi masek lze vytvořit jednoduchou rotací hodnot okolo středu.

Po výpočtu derivací v každém bodě vstupního obrázku je možné použít prahovou hodnotu říkající, které hrany jsou význačné. Typicky ji zadává uživatel aplikace a je

závislá na konkrétním rozsahu hodnot. Jednoznačnou výhodou metod založených na první derivaci je jejich jednoduchost a snadná implementace. Nevýhodou je fakt, že jsou velmi citlivé na šum, který se ve vstupních datech velmi často vyskytuje. K jejich výpočtu se používají pouze nejbližší sousedé, což může mít za následek zpřetrhání hran, které mají být souvislé.

3.1.2 Metody vycházející z druhé derivace

Tato skupina metod vyhledává místa v obraze, ve kterých je druhá derivace rovna nule. Důvodem je to, že druhá derivace prochází nulou v místě, kde má první derivace extrém, přičemž první derivace má extrém v místě, kde dochází k prudké změně jasu [14]. Pro nalezení takového místa nám poslouží ***Laplaceův*** operátor, který je pro funkci dvou proměnných definován jako

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}. \tag{3.12}$$

Laplaceův operátor v místě (x, y) dává skalární hodnotu, což znamená, že jsme ztratili informaci o směru hrany, která je ovšem při detekci hran nepotřebná. Opět je nutné nahradit druhé parciální derivace druhými konečnými diferencemi. Platí tedy $f''(i) \approx f'(i+1) - f'(i) \approx f(i-1) - 2f(i) + f(i+1)$. Přepsáním tohoto vztahu do formy konvoluce dostaneme

$$f'' \approx [+1, -2, +1] * f. \tag{3.13}$$

Vztah (3.13) reprezentuje druhé konečné diference ve směru 0°. Pro směr 90° je výpočet analogický. Jejich dosazením do vztahu (3.12) získáme konvoluční masku, která odpovídá Laplaceovu operátoru:

$$L_4 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{3.14}$$

Vzhledem k tomu, že Laplaceův operátor nabývá nulové hodnoty prakticky při jakékoliv změně intenzity v obraze, je velmi citlivý na šum (zejména pak na samostatný bod). Vychází to z toho, že se snažíme aproximovat druhou derivaci primitivními prostředky. Další nevýhodou může být dvojitá odezva na hrany. Mezi výhody patří fakt, že hrany bývají užší a detailnější než pomocí gradientních metod.

Nevýhody Laplaceova operátoru lze minimalizovat robustnějším řešením zvaným ***LoG*** (***Laplacian of Gaussian***). Tato metoda je založena na principu rozmazání obrazové funkce před vlastním výpočtem druhé derivace, čímž dojde ke zmenšení odstupu signálu od šumu v obraze. Jako vyhlazující filtr se používá konvoluční maska, jejíž hodnoty odpovídají dvourozměrnému Gaussovu rozložení

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.15}$$

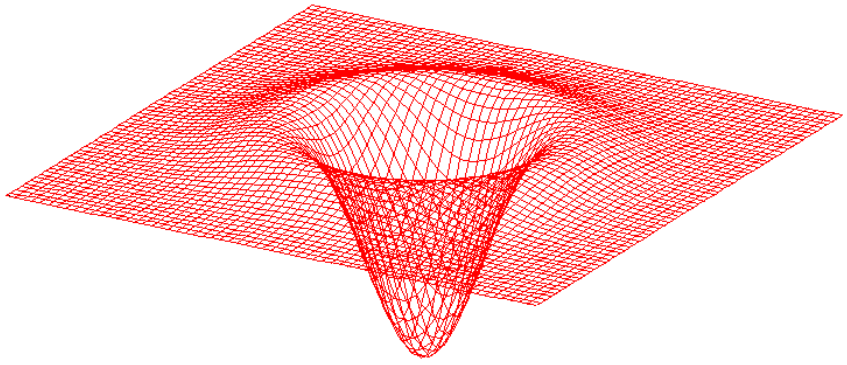
kde x, y představují souřadnice v obraze a σ je směrodatná odchylka. Parametr σ lze měnit podle potřeby, přičemž reprezentuje velikost okolí, které je při filtraci uvažováno. Z Gaussovy funkce vyplývá, že čím blíže jsou pixely středu, tím je jejich váha vyšší. Naopak vliv pixelů vzdálenějších než 3σ je zanedbatelný. Výpočet *LoG* operátoru tedy může probíhat například tak, že si nejprve předpočítáme hodnoty Gaussova filtru pro zadané σ do konvoluční masky. Poté provedeme konvoluci obrazové funkce s touto maskou a následně vypočítáme Laplaceův operátor podle (3.14). Tento postup lze matematicky zapsat jako $\nabla^2[G(x,y,\sigma) * f(x,y)]$. Vzhledem k linearitě operací je možné zaměnit pořadí konvoluce a derivace, tedy $\left(\nabla^2 G(x,y,\sigma)\right) * f(x,y)$. Hodnoty derivace Gaussovy funkce můžeme analyticky předpočítat, neboť nezávisí na konkrétním obraze. Pro zjednodušení derivování provedeme substituci $x^2 + y^2 = r^2$, kde r udává vzdálenost od středu Gaussovy funkce (substituce je možná, protože funkce je symetrická) [16]. Pro první a druhou derivaci poté platí

$$\begin{aligned} G'(r) &= -\frac{1}{\sigma^2} r e^{-\frac{r^2}{2\sigma^2}}, \\ G''(r) &= \frac{1}{\sigma^2}\left(\frac{r^2}{\sigma^2} - 1\right) e^{-\frac{r^2}{2\sigma^2}}. \end{aligned} \tag{3.16}$$

Zpětnou substitucí a zavedením normalizačního koeficientu c zajišťujícího, že součet všech hodnot v konvoluční masce je rovný 0, získáme vztah (3.17), s jehož pomocí můžeme vypočítat hodnoty konvoluční masky reprezentující *LoG* operátor [16].

$$h(x,y) = c\left(\frac{x^2 + y^2 - \sigma^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{3.17}$$

Průběh této funkce je názorně vidět na Obr. 3.2.



Obr. 3.2: Průběh LoG funkce (převzato z [15]).

Nejznámější konvoluční maskou aproximující *LoG* operátor je maska velikosti 5x5 nazývaná „mexický klobouk“:

$$LoG = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}. \quad (3.18)$$

Dalším hranovým detektorem, pracujícím na podobném principu jako *LoG*, je ***DoG*** (***Difference of Gaussians***). Jak už z názvu vyplývá, jedná se o rozdíl dvou obrazů, které vznikly konvolucí s dvourozměrnou Gaussovou funkcí o různém σ (viz vztah (3.15)). Tento operátor je velmi často využíván jako aproximace *LoG* operátoru. Výhoda operátorů *LoG* a *DoG*, oproti dříve uvedeným, je zřejmá – menší citlivost na šum. Mezi jejich nevýhody patří přílišné vyhlazování ostrých tvarů nebo snaha spojovat hrany do uzavřených křivek [16].

3.1.3 Pokročilé metody

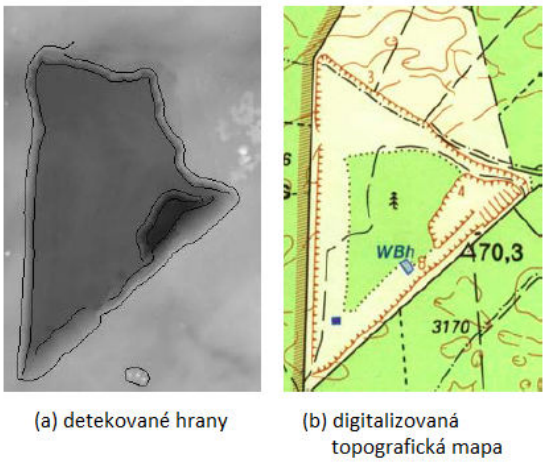
Mezi pokročilé metody lze zařadit ***Cannyho*** hranový detektor, který je v dnešní době asi nejvíce využíván pro detekci hran v oblasti zpracování digitálního obrazu. Algoritmus je rozdělen na více částí. Jelikož se předpokládá, že detektor bude aplikován na neupravená data obsahující šum, je v první části použit vyhlazující filtr. Ten je založen na dvourozměrné Gaussově funkci (viz vztah (3.15)). Výsledkem konvoluce původního obrazu s Gaussovým filtrem získáme mírně rozostřený obraz, který má větší odstup signálu od šumu. Dalším krokem je výpočet velikosti a směru gradientu v každém bodě rozostřeného obrazu, tedy standardní detekce hran. K tomu dobře poslouží klasické hranové detektory (např. Robertsův, Prewittové, Sobelův, atd.). Jelikož hrana v digitálním obraze může mít pouze čtyři různé směry (horizontální, vertikální a dva diagonální), je nutné vypočtený úhel gradientu zaokrouhlit na některý z těchto směrů. V třetí fázi algoritmu dochází ke ztenčení, jehož úkolem je vybrat pouze takové hodnoty gradientů, které jsou lokálním maximem (resp. odebrat všechny body, které nejsou maximem). To znamená, že se pro každý bod obrazu podíváme na pixely ve směru a proti směru gradientu (skutečná hrana je kolmá na směr gradientu) [17]. Jestliže má jeden z nich vyšší hodnotu než právě vyšetřovaný bod, nastavíme aktuálnímu pixelu hodnotu 0 (není hranový). V opačném případě (tj. oba dva pixely mají nižší hodnotu) neděláme nic. Uvedeme příklad:

- Pokud je zaokrouhlený úhel gradientu rovný 0^0 (skutečná hrana je svislá), konkrétní pixel bude brán jako hranový, pokud levý i pravý soused má nižší velikost gradientu.
- V případě, že úhel je rovný 90^0 (hrana je vodorovná), konkrétní bod bude považován na hranový, jestliže horní i dolní soused má nižší velikost gradientu.

Pro zbývající dva diagonální směry provádíme analogický postup. Posledním krokem algoritmu je prahování s hysterezí, které používá dvě prahové hodnoty – minimální T_1 a maximální T_2. Nejprve vyhledáme všechny pixely mající vyšší hodnotu než T_2. Ty jsou rovnou prohlášeny za hranové. Jestliže posuzujeme bod, jehož hodnota leží mezi prahy T_1 a T_2, pak je uznán jedině tehdy, pokud sousedí s bodem, který už byl

jako hrana označen dříve [17]. Programově je možné tento postup implementovat např. pomocí algoritmu prohledávání do šířky, přičemž startovní body jsou ty, které mají vyšší hodnotu než práh T_2.

Podobný postup, jako je Cannyho hranový detektor, je uveden v [18]. Autoři se zabývají automatickým vymezením význačných hran z dat, která byla pořízena při leteckém snímání laserovým skenerem. Nejprve jsou získaná data podrobena předzpracování, ve kterém dochází k odstranění nežádoucího šumu vznikajícího díky objektům v terénu (např. stromy nebo keře). Výsledkem této operace vznikne *DMT*. Následně jsou nepravidelně distribuované body interpolovány do rastru, čímž získáme *DEM*. Rozsah původních výškových hodnot je poté převeden na 0 až 255, aby bylo možné data uložit ve formě šedotónového obrazu. Ten je dále podroben filtraci, ve které dochází k vyhlazení a následnému zesílení hran (např. použitím *SUSAN* filtru). Poté je na data aplikován Cannyho hranový detektor, kterým získáme výsledné hrany. Autoři článku ještě následně prokládají sousední pixely parabolou pro získání hladší terénní křivky. Výstup algoritmu si lze prohlédnout na Obr. 3.3.



Obr. 3.3: Výstup algoritmu detekce terénních hran podle [18] (převzato z [18]).

Jiný přístup pro hledání význačných hran z výškové mapy je uveden v [19]. Hlavní myšlenka algoritmu je postavena na testování statistických hypotéz. Nejprve je pro každý pixel obrazu vypočtena druhá derivace společně s použitím vyhlazujícího Gaussova filtru (viz *LoG* operátor). Následně je z plochého místa v obraze odhadnuta míra šumu, která slouží při výpočtu hodnot testovací statistiky pomocí definovaného vztahu. Získané hodnoty jsou poté porovnány s tabulkovou hodnotou χ^2 testu, ze kterého obdržíme význačné oblasti. Ty jsou dále ztenčeny a výsledné hrany jsou převedeny do vektorové reprezentace pomocí spline křivek. Podrobnější popis algoritmu je uveden v [19].

Je nutné podotknout, že existují i vektorově založené algoritmy pracující s *DEM*. Pro jejich popis zde ovšem není dostatek prostoru. Více se lze dočíst v [20-25].

3.2 Algoritmy založené na trojúhelníkových sítích

Tato skupina metod předpokládá jako vstup obecnou trojúhelníkovou síť. Z toho vyplývá, že je možné zpracovávat jak terénní data (*TIN*), tak i trojrozměrné modely. Vymezování význačných hran, stejně jako u digitálního obrazu, se většinou rozděluje minimálně na dvě části:

1. ***Ohodnocení*** – fáze, ve které dochází k přiřazení váhy každému vrcholu nebo hraně. Čím vyšší je vypočtená hodnota, tím je pro nás daný vrchol (nebo hrana) důležitější.
2. ***Prahování*** – na základě předchozího ohodnocení se vyberou pouze takové hrany, které splňují konkrétní kritérium.

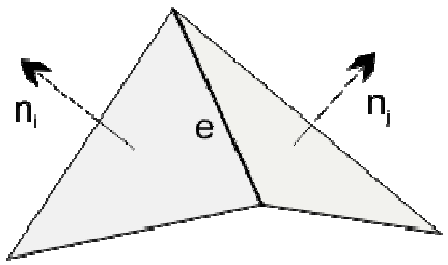
Hodnota prahu bývá závislá na konkrétních datech (zejména na velikosti a geometrii modelu), proto se zpravidla nechává zadat uživatelem aplikace. Nejčastěji se používají dva typy prahování [26]:

- ***Standardní*** – nejjednodušší metoda, ve které je každá hrana samostatně porovnána vůči definovanému prahu. Je-li váha hrany vyšší než stanovená mez, je považována na význačnou. Jinou variantou standardního prahování je tzv. procentní, ve kterém vybíráme určité procento nejvyšších hodnot podle nastaveného prahu.
- ***Hysterezní*** – princip algoritmu je stejný jako u Cannyho hranového detektoru (viz kap. 3.1.3).

V následujících pěti podkapitolách budou uvedeny klasické metody používané k výpočtu ohodnocení jednotlivých vrcholů nebo hran. Ty je možné spojit s výše popsaným prahováním, a tím získat význačné hrany. V šesté podkapitola poté budou popsány pokročilé algoritmy.

3.2.1 Metoda SOD (Second Order Difference)

Jedna z nejznámějších a nejjednodušších metod, která přiřazuje váhu jednotlivým hranám na základě úhlu mezi jednotkovými normálami dvou sousedních trojúhelníků spojených danou hranou. Situace je zobrazena na Obr. 3.4.



Obr. 3.4: Princip SOD metody (úhel mezi normálami).

Při výpočtu ohodnocení hrany je nutné nejprve určit normály sousedních trojúhelníků. Ty lze stanovit pomocí vztahu

$$\boldsymbol{n} = (B - A) \times (C - A), \tag{3.19}$$

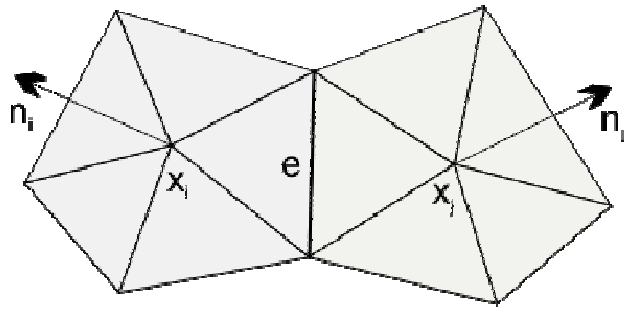
kde A, B a C představují body konkrétního trojúhelníku. Poté již můžeme vypočítat úhel mezi normálami $\boldsymbol{n_i}$ a $\boldsymbol{n_j}$ jako

$$w(e) = \cos^{-1}\left(\frac{\boldsymbol{n_i} \cdot \boldsymbol{n_j}}{\|\boldsymbol{n_i}\| \cdot \|\boldsymbol{n_j}\|}\right). \tag{3.20}$$

Hodnota získaná ze vztahu (3.20) udává významnost dané hrany a je dále použita při prahování. Mezi velké výhody této metody patří její jednoduchost. Hodí se na hrubé modely obsahující ostré hrany (např. různé *CAD* součástky). Při použití na hladších nebo zašuměných objektech ovšem dává velmi špatné výsledky, protože pro výpočet používá velmi malé okolí [26].

3.2.2 Metoda ESOD (Extended Second Order Difference)

Tento postup je prostým rozšířením předchozí metody. Namísto ohodnocení hrany na základě úhlu mezi normálami sousedních trojúhelníků je úhel počítán z normál vrcholů ležících naproti uvažované hraně (viz Obr. 3.5).



Obr. 3.5: Princip ESOD metody (převzato z [26]).

Pro určení úhlu (resp. ohodnocení hrany) můžeme opět využít vztah (3.20). Pokud normály ve vrcholech nejsou známy, lze je vypočítat jako aritmetický průměr normál z okolních trojúhelníků obsahující daný vrchol. Matematicky to lze zapsat jako

$$\boldsymbol{n} = \frac{\sum_{i=1}^{k} \boldsymbol{n_i}}{k}, \tag{3.21}$$

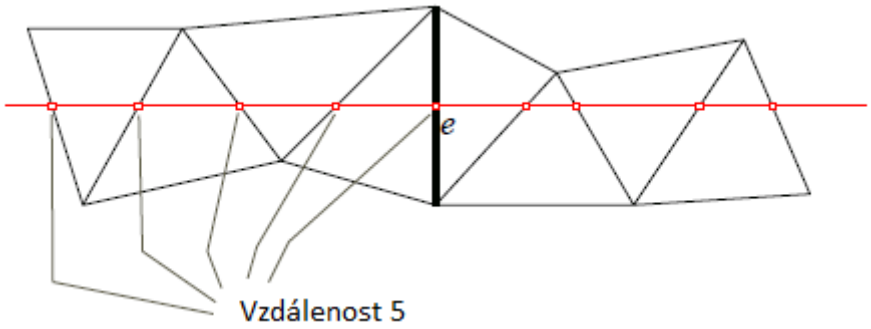
kde k je celkový počet sousedních trojúhelníků obsahující konkrétní vrchol a $\boldsymbol{n_i}$ jsou odpovídající normály trojúhelníků. Výhodou této metody oproti *SOD* je vyšší odolnost proti šumu. Nevýhodnou je fakt, že u hrubých modelů dává horší výsledky než předchozí metoda [26].

3.2.3 Metoda BFP (Best Fit Polynomial)

Myšlenka této metody je založena na aproximaci okolí hrany polynomem $p(u)$ stupně n, který lze interpretovat jako rovinnou křivku v trojrozměrném prostoru (tj. nejedná se o aproximaci plochou). Pomocí vytvořeného polynomu můžeme snadno vypočítat druhou derivaci v místě hrany, která udává její ohodnocení představující lokální zakřivení. Pro její váhu tedy platí

$$w(e) = p''(e). \tag{3.22}$$

Hlavním problémem tohoto postupu je, jakým způsobem získat body, které poté slouží při tvorbě polynomu. Rozumným řešením je vytvořit rovinu kolmou na uvažovanou hranu a umístit ji do jejího středu. Poté můžeme vypočítat průsečíky roviny s hranami sousedních trojúhelníků, které nám budou dávat požadované body při konstrukci polynomu [26]. Tato situace je zobrazena na Obr. 3.6.



Obr. 3.6: Princip konstrukce polynomu v BFP metodě. Červená čára představuje uvažovanou rovinu při pohledu shora (převzato z [26]).

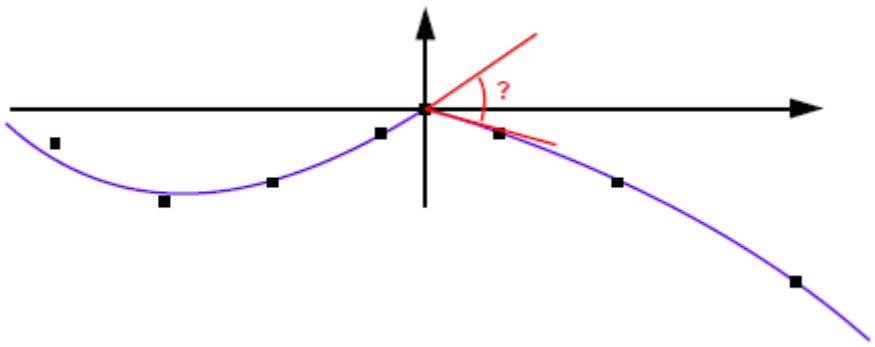
Výhodou tohoto přístup je skutečnost, že počet nejbližších průsečíků okolo uvažované hrany lze nastavit podle potřeby. Ve výše uvedeném obrázku je uvažováno pět průsečíků na každou stranu od hrany (celkem tedy deset). Navíc lze libovolně definovat také stupeň vytvářeného polynomu [26].

3.2.4 Metoda ABBFP (Angle Between Best Fit Polynomials)

Tato metoda je jednoduchým rozšířením předchozího postupu. Namísto vytvoření jednoho polynomu aproximujícího všechny průsečíky s danou rovinou se zde definují polynomy dva. První aproximuje body ležící na jedné straně od uvažované hrany a druhý body na straně opačné. Ohodnocení hrany je poté vypočteno na základě úhlu mezi tečnami křivek v bodě hrany, tedy

$$w(e) = \cos^{-1}\left(\frac{\left(1, p_l'(e)\right)}{\left\|\left(1, p_l'(e)\right)\right\|} \cdot \frac{\left(1, p_p'(e)\right)}{\left\|\left(1, p_p'(e)\right)\right\|}\right), \tag{3.23}$$

kde $p_l'(e)$ a $p_p'(e)$ představují derivaci prvního a druhého polynomu v místě hrany. Princip metody *ABBFP* je zobrazen na Obr. 3.7.



Obr. 3.7: Princip metody ABBFP (převzato z [26]).

Výhody tohoto postupu jsou podobné jako u *BFP*. Opět je možné nastavit podle potřeby počet průsečíků nebo stupeň polynomů.

3.2.5 Metoda založená na Laplace-Beltramiho operátoru

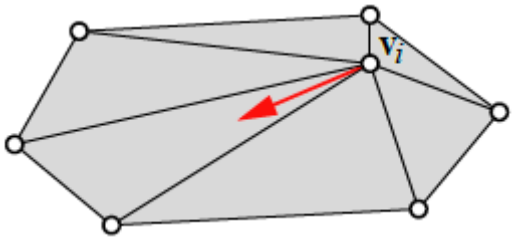
V oblasti diferenciální geometrie je Laplace-Beltramiho operátor definován jako divergence gradientu (součet druhých parciálních derivací). Matematicky to lze zapsat jako

$$\Delta f = div(grad\, f). \tag{3.24}$$

Takto definovaný operátor dobře funguje pro hladké funkce, ovšem trojúhelníkový model je funkce po částech lineární. Pro správnou formulaci tohoto operátoru v diskrétním prostředí je nutné zavést definici triangularizovaných modelů. Obecný trojúhelníkový model je možné reprezentovat jako graf $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, kde množina $\boldsymbol{V} = [\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_n}]$ představuje jednotlivé vrcholy a $\boldsymbol{E} = [\boldsymbol{e_1}, \boldsymbol{e_2}, \dots, \boldsymbol{e_m}]$ je množina hran [27]. Nyní můžeme definovat diskrétní Laplace-Beltramiho operátor pro konkrétní vrchol $\boldsymbol{v_i}$ jako

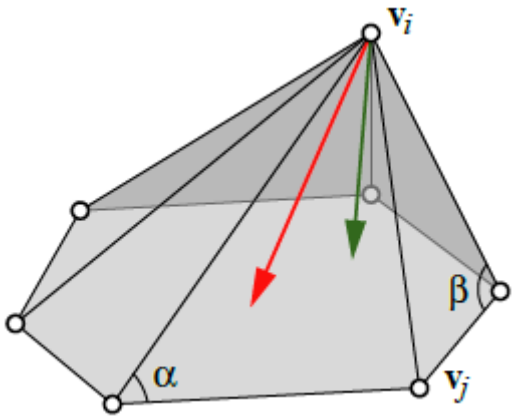
$$\boldsymbol{\delta_i} = \frac{1}{d_i} \sum_{\{i,j\} \in \boldsymbol{E}} w_{ij} (\boldsymbol{v_i} - \boldsymbol{v_j}), \tag{3.25}$$

kde w_{ij} je váha hrany (i, j) a d_i představuje normalizační koeficient asociovaný s vrcholem $\boldsymbol{v_i}$. Je třeba podotknout, že výsledek této operace je vektor, nikoliv skalární hodnota. Pro ohodnocení konkrétního vrcholu $\boldsymbol{v_i}$ použijeme velikost vektoru, tj. $w(\boldsymbol{v_i}) = \|\boldsymbol{\delta_i}\|$. Jednoduchou volbou parametrů $d_i = 1$ a $w_{ij} = 1$ pro všechny hrany okolo vrcholu $\boldsymbol{v_i}$ získáme tzv. jednotkový Laplacián. Ten je možné použít pro ohodnocení kvality sítě ve smyslu rovnoměrnosti trojúhelníků. Jeho základní vlastností je, že výsledný vektor směřuje do těžiště vějíře obklopujícího daný vrchol, neboť se pro výpočet uvažuje pouze topologie sítě (bez geometrie). Z toho vyplývá, že i pro rovnou plochu může být jednotkový Laplacián nenulový (viz Obr. 3.8). Takto zvolené parametry jsou tedy nevhodné pro hledání význačných hran.



Obr. 3.8: Ukázka jednotkového Laplaciánu na rovné ploše (převzato z [27]).

Při zvolení parametrů $d_i = 1$ a $w_{ij} = \frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right)$ získáme takzvaný kotangentový Laplacián, kde α_{ij} a β_{ij} představují úhly naproti hraně (i, j). Jelikož váha konkrétní hrany závisí na úhlech trojúhelníků, je možné odvodit, že pro rovinnou plochu bude výsledný vektor nulový. Navíc platí, že čím větší je lokální zakřivení okolo vrcholu $\boldsymbol{v_i}$, tím větší je velikost kotangentového Laplaciánu. Porovnání výše zmíněných přístupů je zobrazeno na Obr. 3.9.

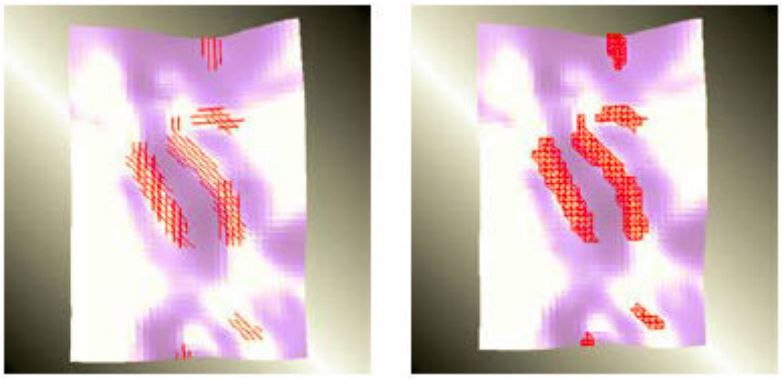


Obr. 3.9: Porovnání jednotkového Laplaciánu (červený vektor) s kotangentovým (převzato z [27]).

Kotangentový Laplacián je také známý jako dobrá aproximace normálového vektoru. Pro výpočet ohodnocení vrcholu je však stále nevhodný, protože jeho velikost závisí na rozměrech trojúhelníků. Proto se velmi často normalizační koeficient volí jako $d_i = a(i)/3$, kde $a(i)$ je obsah všech trojúhelníků okolo vrcholu $\boldsymbol{v_i}$ [28].

3.2.6 Pokročilé metody

Metoda popsaná v [26] je jednoduchým rozšířením výše uvedených algoritmů. V první fázi je jednotlivým hranám přiřazena váha na základě metody *SOD*, *ESOD*, *BFP* nebo *ABBFP*. Dále je pomocí prahování (standardního nebo hysterezního) vybrána pouze taková množina hran, které mohou být považovány za význačné. Následně jsou vytvářeny oblasti. To znamená, že do aktuální množiny označených hran přidáme každou hranu, která má oba vrcholy ve významné oblasti a není označena. Cílem tohoto procesu je vyplnění děr uvnitř oblastí (viz Obr. 3.10).



Obr. 3.10: Vlevo: po prahování. Vpravo:po vytvoření význačných oblastí (převzato z [26]).

V poslední fázi algoritmu je aplikována skeletonizace, s jejíž pomocí dochází k převodu oblastí na množinu význačných hran. Zde je možné použít dva různé přístupy:

- ***Skeletonizace založená na trojúhelnících*** – redukuje význačné oblasti vyhazováním trojúhelníků jeden po druhém. Za účelem získání správného výsledku je nutné zavést určitá omezení, která však v [26] nejsou uvedena. Nevýhoda tohoto přístupu spočívá v možnosti zpřetrhání delších linií, které mají být spojeny.
- ***Skeletonizace založená na vrcholech*** – tento postup nejprve ohodnotí jednotlivé body nacházející se ve význačné oblasti na základě vzdálenosti od okraje. Následně jsou vyhozeny všechny vrcholy mající alespoň jednoho souseda s vyšším ohodnocením. Výsledkem je nespojitá množina bodů, ze kterých je nutné zpětně vytvořit význačné hrany.

Tato metoda funguje uspokojivě na hrubých modelech, ale při použití na hladších objektech selhává. Její kvalita záleží zejména na skeletonizaci a typu metody pro ohodnocení vrcholů.

Další postup, který staví na algoritmu *SOD*, je popsán v [29]. Po určení ohodnocení jednotlivých hran je z modelu automaticky vypočtena mez pro prahování. Její vymezení probíhá nad oblastí v datech, ve které dochází k minimálním změnám. Ze všech úhlů mezi normálami sousedních trojúhelníků v dané oblasti je vypočten aritmetický průměr a směrodatná odchylka. Hodnota odchylky bývá zpravidla větší než průměr, proto je nastavena jako práh. V další fázi algoritmu jsou vybrané hrany převedeny na delší úsečky pomocí aproximace přímkou nebo obecně polynomem vyššího stupně.

Jiným známým přístupem jsou metody založené na rekonstrukci hladké plochy. Metoda může být použita v lokálním nebo globálním pojetí. Nejčastěji se používá lokální aproximace na základě vrcholu a jeho sousedů. Z dané plochy poté můžeme určit například střední křivost, která bude udávat váhu konkrétního vrcholu. Na vypočtené hodnoty je možné následně aplikovat prahování, a tím získat význačné hrany. Více o těchto metodách se lze dočíst v [30-32].

4 Navržená metoda detekce hran

Základní požadavek kladený na navrhovanou metodu se týkal funkčnosti na trojúhelníkových sítích a zároveň její použitelnosti na terénních i trojrozměrných modelech. Hlavní myšlenka je založena na postupu popsaném v [33], avšak některé části algoritmu byly upraveny z důvodu nalezených chyb nebo plně nahrazeny jinou metodou. Navrhovaný postup lze stručně rozdělit do tří částí:

1. ***Ohodnocení vrcholů*** – pro každý vrchol modelu se vypočítá pomocí křivostí váha definující jeho významnost.
2. ***Získání význačných oblastí*** – z vah vypočtených v bodu 1 vybereme pouze takové, které nás nejvíce zajímají. Typicky se jedná o hodnoty větší než prahová mez nebo ležící v určitém intervalu.
3. ***Použití morfologických operátorů*** – na vzniklé význačné oblasti můžeme aplikovat morfologické operátory a následně využít skeletonizaci pro získání význačných hran ve formě úzké čáry.

Jedním z klíčových důvodů výběru této metody bylo netradiční uplatnění morfologických operátorů, které jsou nejčastěji využívány při zpracování digitálního obrazu (viz kap. 2.3), avšak zde se používají na trojúhelníkové sítě. Pomocí nich je možné z význačných oblastí odstranit nežádoucí artefakty (např. vnitřní díry) nebo spojit oddělené oblasti, které mají být reprezentovány jako jedna (viz problémy při vymezování terénních linií v kap. 2.1). Dalším argumentem při výběru metody byl výpočet ohodnocení vrcholů na základě křivostí, které jsou matematickým aparátem pro určování lokálního zakřivení plochy. Proto jsou vhodnou volbou při výpočtu vah. Navíc o tom svědčí i fakt, že všechny pokročilé algoritmy založené na rekonstrukci hladké plochy používají pro výpočet křivosti.

Před vlastní implementací byl návrh metody prodiskutován s Ing. Tomášem Bayerem, Ph.D., který je odborníkem v oblasti aplikované geoinformatiky a kartografie. Po prostudování zvoleného postupu nebyly z jeho strany vzneseny žádné připomínky. Dále byl algoritmus probrán s Ing. Jakubem Šilhavým, který se v rámci své disertační práce zabývá automatickým vymezováním morfolineamentů (tj. hřbetnic a údolnic). Ani z jeho strany nebyly navrženy žádné změny postupu. Navíc byla jeho specializovaná metoda porovnána s námi navrženou na stejných vstupních datech (viz kap. 6.2.1).

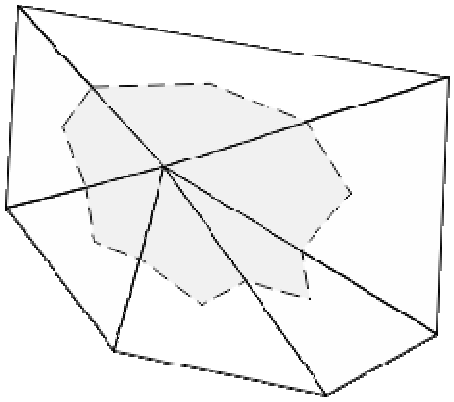
4.1 Ohodnocení vrcholů

V této části navrhované metody je každému vrcholu trojúhelníkového modelu přiřazena hodnota udávající, jak moc se plocha v jeho okolí mění. Jak už bylo uvedeno výše, změnu tvaru plochy lze popsat pomocí křivostí. Pro jejich výpočet je nutné umět spočítat derivaci v konkrétním bodě. Problém je, že derivace je definována pro hladkou

funkci, ale trojúhelníkový model je funkce po částech lineární. Z tohoto důvodu bylo vytvořeno několik přístupů k určení požadované hodnoty. Je nutné podotknout, že všechny metody jsou pouze aproximační, neboť není možné získat korektní výsledek, pokud neznáme původní funkci.

Prvním přístupem je snaha zrekonstruovat hladkou funkci, ze které lze dopočítat křivosti. Toho lze docílit dvěma různými postupy. Jedním z nich je vytvořit původní funkci popisující celý model pomocí vrcholů a normál trojúhelníkové sítě. Tento postup je ovšem dosti obtížný a pro objemné modely výpočetně náročný. Jednodušší a více používaná metoda se snaží rekonstruovat hladkou funkci pouze v lokálním okolí vyšetřovaného bodu. K tomu se nejčastěji využívá daný vrchol, jeho sousedé a normály v těchto vrcholech. Po vytvoření plochy již není složité dopočítat křivosti v konkrétním bodě. Na podobném principu je také založeno ohodnocení vrcholů v původní verzi námi zvoleného postupu. Metoda se snaží lokálně odhadnout první a druhou základní formu plochy v každém vrcholu trojúhelníkové sítě, ze kterých lze následně přímočaře dopočítat křivosti. Nejprve je vytvořena téměř izometrická parametrizace pro vyšetřovaný vrchol a jeho sousedy. Na základě této parametrizace je poté možné plochu lokálně aproximovat použitím Taylorova polynomu druhého stupně pro funkci dvou proměnných. Řešením soustavy lineárních rovnic získáme koeficienty představující parciální derivace, s jejichž pomocí můžeme dopočítat např. normálový vektor, střední nebo maximální křivost v konkrétním vrcholu.

Uvedený postup je však pro modely obsahující milióny trojúhelníků velmi neefektivní z důvodu časové náročnosti. Proto byla tato část zvolené metody plně nahrazena jiným přístupem, který odhaduje křivosti přímo z trojúhelníkového modelu. Nový algoritmus vychází z článku [35], jenž se zabývá diferenciální geometrií pro trojúhelníkové sítě. Stejně jako v předchozích metodách, i zde se k výpočtu používá pouze konkrétní vrchol a jeho sousedé. Dále je nutné uvažovat určitou oblast plochy, ke které se křivost vztahuje. Jednou z možností je spojení oblastí vymezených těžišti trojúhelníků a středy hran obsahujících daný vrchol. Tato varianta je přehledně zobrazena na Obr. 4.1.

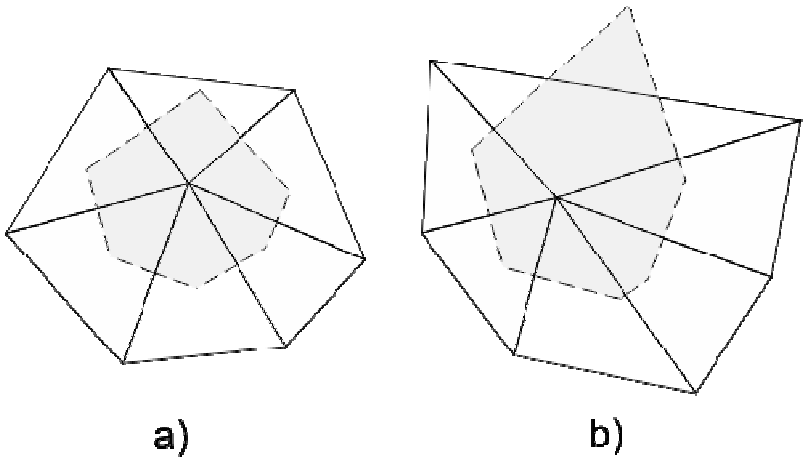


Obr. 4.1: Oblast plochy vytvořená na základě těžišť sousedních trojúhelníků a středů hran obsahujících daný vrchol.

Obsah této plochy (označeno $A_{Těžiště}$) lze jednoduše vypočítat jako jednu třetinu součtu obsahů okolních trojúhelníků. Pro úplnost uvedeme vztah pro určení obsahu obecného ΔABC:

$$A_{\Delta ABC} = \frac{1}{2}\|(B - A) \times (C - A)\|. \tag{4.1}$$

Lepší variantou při výpočtu křivostí na trojúhelníkových sítích je použití Voroného oblasti, protože poskytuje prokazatelně menší chybu než oblast předchozí. Důkaz tohoto tvrzení je možné nalézt v [35]. Jeho základní myšlenka vychází z toho, že tato oblast obsahuje pouze takové body, které jsou ke konkrétnímu vrcholu nejblíže, a proto minimalizuje chybu ve výpočtu křivosti. Voroného oblast pro konkrétní bod trojúhelníku vznikne tak, že vezmeme středy hran obsahujících zvolený bod a propojíme je se středem kružnice opsané tohoto trojúhelníku. Sjednocením těchto podoblastí přes všechny trojúhelníky okolo konkrétního vrcholu získáme celkovou Voroného oblast. Příklady si lze prohlédnout na Obr. 4.2.



Obr. 4.2: Voroného oblast pro konkrétní vrchol trojúhelníkové sítě. a) Oblast má všechny trojúhelníky ostroúhlé. b) Oblast obsahuje i tupoúhlý trojúhelník.

Obsah této oblasti pro bod P ***ostroúhlého*** ΔPQR lze vypočítat jako

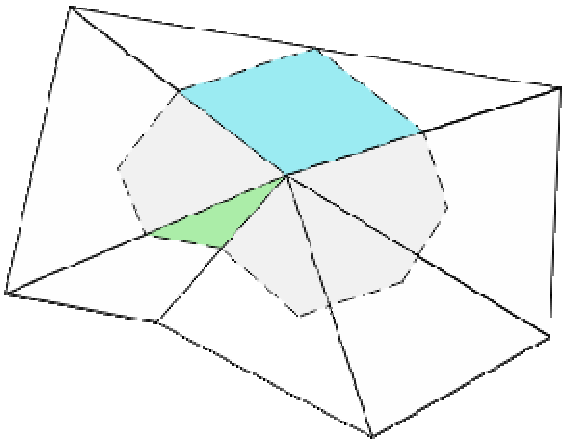
$$A_{Voronoi\,\Delta PQR} = \frac{1}{8}(|PR|^2 \cot\angle Q + |PQ|^2 \cot\angle R). \tag{4.2}$$

Součtem jednotlivých obsahů přes všechny trojúhelníky okolo zvoleného vrcholu dostaneme celkový obsah Voroného oblasti (označeno $A_{Voronoi}$). Je však třeba zdůraznit, že všechny trojúhelníky musejí být ostroúhlé. Kotangens úhlu uvedený v rovnici (4.2) lze efektivně vypočítat pomocí vztahu

$$\cot(\boldsymbol{a}, \boldsymbol{b}) = \frac{\cos(\boldsymbol{a}, \boldsymbol{b})}{\sin(\boldsymbol{a}, \boldsymbol{b})} = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\sqrt{\|\boldsymbol{a}\|^2 \cdot \|\boldsymbol{b}\|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2}}, \tag{4.3}$$

kde $\boldsymbol{a}$ a $\boldsymbol{b}$ jsou vektory, mezi kterými je úhel počítán.

Jak již bylo uvedeno výše, obsah Voroného oblasti pro konkrétní vrchol je možné vypočítat pomocí rovnice (4.2) pouze tehdy, pokud jsou všechny sousední trojúhelníky ostroúhlé. V praxi se ovšem velmi často stává, že tomu tak není (viz Obr. 4.2b). V takovém případě nejenže vztah (4.2) neplatí, ale navíc může docházet k překrývání sousedních oblastí nebo mezi nimi vznikne díra. Proto byla navržena tzv. kombinovaná oblast, která zajišťuje perfektní navazování a navíc se snaží co nejvíce využívat oblast Voroného. Pro konkrétní vrchol trojúhelníkové sítě ji lze získat následovně: je-li sousední trojúhelník ostroúhlý, použijeme střed kružnice opsané (tj. Voroného oblast). V opačném případě používáme střed té hrany, která leží naproti tupému úhlu. Získaný bod poté propojíme se středy přilehlých hran stejně, jako tomu bylo v předchozích variantách. Uvedené možnosti jsou zobrazeny na Obr. 4.3.



Obr. 4.3: Příklad kombinované oblasti. Šedá barva značí ostroúhlé trojúhelníky (tj. Voroného oblast). Modrá barva určuje trojúhelníky, které mají tupý úhel u vyšetřovaného vrcholu. Zelená barva představuje tupoúhlé trojúhelníky s tupým úhlem u jednoho ze zbývajících dvou vrcholů.

Obsah této oblasti (označeno $A_{Kombinovaná}$) lze získat velmi jednoduše. Popis algoritmu v pseudokódu je zobrazen na Obr. 4.4.

```
Akombinovaná = 0
For každý sousední trojúhelník T vrcholu x
    If T je ostroúhlý
        Akombinovaná += obsah Voronoiovy oblasti vrcholu x v T
    Else
        If úhel u vrcholu x v T je tupý
                Akombinovaná += obsah(T) / 2
        Else
                Akombinovaná += obsah(T) / 4
End
```

Obr. 4.4: Pseudokód algoritmu pro výpočet obsahu kombinované oblasti (převzato z [35]).

Jednotlivé dílčí obsahy uvedené v algoritmu lze vypočítat na základě rovnic (4.1) nebo (4.2). Nyní již máme vše potřebné, abychom mohli formulovat vztahy pro výpočet křivostí.

4.1.1 Střední křivost

Střední křivost pro konkrétní vrchol $\boldsymbol{v_i}$ trojúhelníkové sítě lze vypočítat pomocí vztahu

$$H = \left\| \frac{1}{4 \cdot A_{Kombinovaná}} \sum_{\{i,j\} \in \boldsymbol{E}} \left(\cot \alpha_{ij} + \cot \beta_{ij} \right) \left(\boldsymbol{v_i} - \boldsymbol{v_j} \right) \right\|, \tag{4.4}$$

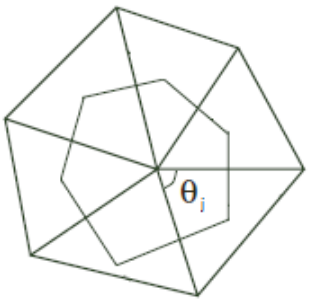
kde $\cot \alpha_{ij}$ a $\cot \beta_{ij}$ představují úhly naproti hraně (i, j) a $A_{Kombinovaná}$ značí obsah kombinované oblasti. Z této rovnice si lze všimnout, že je velmi podobná výpočtu kotangentového Laplaciánu (viz kap. 3.2.5). Ve skutečnosti se liší pouze tím, že zde používáme kombinovanou oblast a výsledný vektor navíc dělíme dvěma.

4.1.2 Gaussova křivost

Gaussovu křivost pro konkrétní vrchol $\boldsymbol{v_i}$ trojúhelníkové sítě je možné určit pomocí vztahu

$$G = \left(2\pi - \sum_{j=1}^{\#f} \theta_j \right) \Big/ A_{Kombinovaná}, \tag{4.5}$$

kde θ_j je úhel j-tého trojúhelníku u vrcholu $\boldsymbol{v_i}$ (viz Obr. 4.5) a $\#f$ představuje počet sousedních trojúhelníků.



Obr. 4.5: Ukázka vnitřního úhlu používaného při výpočtu Gaussovy křivosti (převzato z [35]).

Z (4.5) je zřejmé, že Gaussova křivost je nulová pro jakoukoliv rovnou plochu nebo takovou, kde dochází k zakřivení pouze v jednom směru (např. střecha domu).

4.1.3 Hlavní křivosti

Z diferenciální geometrie víme, že Gaussovu a střední křivost lze velmi jednoduše určit na základě hlavních křivostí. Jelikož hodnoty G a H pro trojúhelníkovou síť již umíme vypočítat, pro maximální a minimální křivost ve vrcholu $\boldsymbol{v_i}$ můžeme psát

$$k_n^{max} = H + \sqrt{H^2 - G},$$
$$k_n^{min} = H - \sqrt{H^2 - G}. \tag{4.6}$$

Na rozdíl od hladké plochy, kde $H^2 - G$ je vždy kladné číslo, na trojúhelníkových sítích musíme kontrolovat, že H^2 je větší než G, abychom se vyhnuli numerickým problémům. Jestliže je menší (vzácný případ), je nutné nastavit $\sqrt{H^2 - G}$ na nulu.

4.2 Získání význačných oblastí

Tato fáze algoritmu přímo navazuje na předchozí a má za cíl vybrat podmnožinu ohodnocených vrcholů, které jsou pro naši aplikaci nejdůležitější. To znamená, že pokud v předchozí části ohodnocujeme vrcholy pomocí křivostí, budeme se snažit vybírat pouze takové vrcholy, které mají vysokou hodnotu. Naopak, jestliže hledáme rovnější části modelu, vybíráme vrcholy s nízkou křivostí.

Obecně lze říci, že pokud je trojúhelníkový model vytvořen na základě dat, která byla nasnímána z reálného objektu, obsahují šum. Aplikování dalších částí algoritmu přímo na vypočtené hodnoty by mohlo způsobit špatné výsledky. Proto je vhodné získané hodnoty z první fáze metody nějakým způsobem filtrovat. Pro tyto účely lze využít např. jednoduchý mediánový filtr nebo průměr z nejbližšího okolí.

Jak již víme, morfologické operátory pracují nejčastěji s binární hodnotou (tj. 0 nebo 1). Z tohoto důvodu je nutné ohodnocení vrcholů klasifikovat do těchto dvou skupin. K tomu je možné použít následující formulaci:

$$1 \leq i \leq N: \quad F_i = \begin{cases} 1, & X_i \in [a, b], \\ 0, & v\ ostatních\ případech, \end{cases} \tag{4.7}$$

kde N udává počet vrcholů trojúhelníkového modelu, F_i představuje zařazení vrcholu do význačné (pro $F_i = 1$) nebo nevýznačné (pro $F_i = 0$) oblasti, X_i určuje váhu vypočtenou v předchozí fázi algoritmu a hodnoty $a, b \in \mathbb{R}$ udávají práh. Určení prahových hodnot většinou závisí na konkrétní aplikaci. Lze je vypočítat automaticky nebo nechat uživatele, aby je zadal.

4.3 Použití morfologických operátorů

Jak již bylo uvedeno v teoretickém úvodu, pojem matematická morfologie je delší dobu známý především ze zpracování digitálních obrazů. Základní výhoda morfologických operátorů spočívá v jednoduchosti a snadné implementaci. Definice těchto operátorů ovšem neumožňuje přímočaré použití na obecných trojúhelníkových modelech kvůli nestrukturované topologii. Proto je potřeba jejich předefinování. Nejdříve vytvoříme množinu (označenou jako význačná), která obsahuje všechny vrcholy, které byly zařazeny do skupiny 1:

$$F := \{j \in \{1, \dots, N\} \mid F_j = 1\}. \tag{4.8}$$

Dále formulujeme sousedství $\boldsymbol{nhd}$ každého vrcholu V_i jako

$$\boldsymbol{nhd}\{i\} = \{i\} \cup \{j \mid \exists hrana(V_i, V_j)\}. \tag{4.9}$$

Ze zápisu (4.9) je zřejmé, že sousedství konkrétního vrcholu i je množina skládající se z vrcholu i a jeho sousedů přes hranu. Lze si také všimnout, že v (4.8) a (4.9) pracujeme pouze s indexy vrcholů. Poloměr okolí je samozřejmě možné rekurzivně zvětšovat. *N*-sousedství $\boldsymbol{nhd}^n$ lze definovat jako

$$\begin{aligned} \boldsymbol{nhd}\{i_1, \dots, i_k\} &:= \bigcup_{1 \le \mu \le k} \boldsymbol{nhd}\{i_\mu\}, \\ \boldsymbol{nhd}^1\{i\} &:= \boldsymbol{nhd}\{i\}, \\ \boldsymbol{nhd}^{n+1}\{i\} &:= \boldsymbol{nhd}(\boldsymbol{nhd}^n\{i\}) \ \ (n > 1). \end{aligned} \tag{4.10}$$

Nyní již můžeme formulovat jednotlivé morfologické operátory. Nechť $F \subseteq \{1, \dots, N\}$. ***Dilatace*** množiny F s použitím $\boldsymbol{nhd}^n$ je definována jako

$$\boldsymbol{dilatace}^n(F) := \{j \mid \exists i \in F : \ j \in \boldsymbol{nhd}^n\{i\}\}. \tag{4.11}$$

N-sousedství $\boldsymbol{nhd}^n$ je zde použito jako strukturní element. Tím dochází k přizpůsobení topologie trojúhelníkového modelu pro použití morfologických operací. Z definice dilatace je zřejmé, že tato operace přidává nové vrcholy do význačné oblasti. Proto může být použita pro zaplňování děr (např. uvnitř význačné oblasti jsou dva neoznačené vrcholy, které by ovšem do této oblasti patřit měly).

Operátor eroze dělá opak než dilatace, snaží se některé vrcholy z význačné oblasti vymazat. Proto je možné jej použít k odstranění nepotřebných „větví“. ***Eroze*** množiny F s použitím $\boldsymbol{nhd}^n$ je formulována jako

$$\boldsymbol{eroze}^n(F) := \{j \mid \boldsymbol{nhd}^n\{j\} \subseteq F\}. \tag{4.12}$$

U obou výše uvedených operátorů je možné použít *N*-sousedství. Pro implementaci je dobré vědět, že stejného výsledku je možné dosáhnout opakovaným použitím konkrétní operace, tedy

$$\begin{aligned} \boldsymbol{eroze}^n &= \boldsymbol{eroze}^1 \circ \dots\dots \circ \boldsymbol{eroze}^1 \quad (n\ krát), \\ \boldsymbol{dilatace}^n &= \boldsymbol{dilatace}^1 \circ \dots\dots \circ \boldsymbol{dilatace}^1 \quad (n\ krát). \end{aligned} \tag{4.13}$$

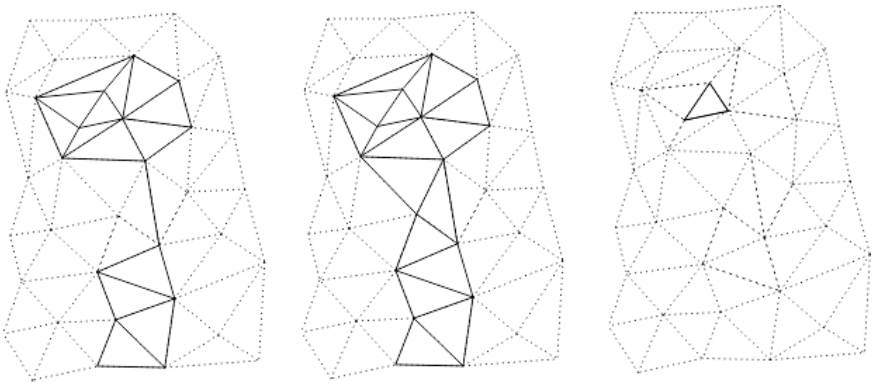
Pomocí operátorů dilatace a eroze lze snadno odstranit nepotřebné artefakty, které zůstaly ve význačné oblasti i přes použití filtrace. Pokud využijeme erozi k odstranění nepotřebných „větví“, dojde také k celkovému zmenšení význačné oblasti. Proto byly vytvořeny následující dva operátory. ***Otevření*** se definuje jako

$$\boldsymbol{otevření}^n = \boldsymbol{dilatace}^n \circ \boldsymbol{eroze}^n. \tag{4.14}$$

Operace otevření je tedy formulována jako eroze, po které následuje dilatace. Tím je možné vymazat nepotřebné „větve“ význačné oblasti při zachování její velikosti. Dále se tento operátor používá k odstranění malých „ostrovů“. Prohozením pořadí operací dostaneme operátor ***uzavření***, který je definován jako

$$\boldsymbol{uzavření}^n = \boldsymbol{eroze}^n \circ \boldsymbol{dilatace}^n. \tag{4.15}$$

Tato operace se používá k zaplnění děr uvnitř význačných oblastí nebo k vyplnění výklenků podél hranice. Příklad operátoru uzavření a otevření je vidět na Obr. 4.6.



Obr. 4.6: Vlevo: původní význačná oblast. Uprostřed: po aplikování uzavření. Vpravo: po aplikování otevření.

Naším hlavním cílem je ovšem získání význačných hran trojúhelníkového modelu, které jsou většinou reprezentovány pomocí úzké linie. Význačná oblast je však zatím příliš široká na to, abychom ji mohli prohlásit za hranu. Z tohoto důvodu definujeme operátor nazývající se skeletonizace. Cílem této operace je získat kostru významné oblasti, která není tenčí než jeden vrchol a popisuje její topologii. To lze udělat tak, že opakovaně odstraňujeme vrcholy nalézající se na hranici oblasti. Na rozdíl od eroze u skeletonizace existují části význačné oblasti, které musejí být zachovány. Je tedy potřeba definovat kritéria, zda může nebo nemůže být daný vrchol odstraněn.

Nechť $F \in \{1,0\}^N$ je vektor popisující rozdělení vrcholů trojúhelníkového modelu do skupiny s hodnotou 0 nebo 1. Nechť $(u_\mu^i)_{\mu=0}^{\mu=n_i-1}$ označuje sekvenci indexů n_i okolních vrcholů vrcholu V_i seřazenou podle hodinových ručiček (i je index daného vrcholu a n_i je počet jeho sousedů). Dále definujeme komplexitu c_i vrcholu V_i jako

$$c_i := \sum_{\mu=0}^{\mu=n_i-1} \left| F_{u_\mu^i} - F_{u_{\mu+1 \bmod n_i}^i} \right|, \tag{4.16}$$

kde $F_{u_\mu^i}$ představuje hodnotu skupiny sousedního vrcholu μ k vrcholu i. Řekneme, že vrchol V_i je ***komplexní***, pokud $F_i = 1$ a $c_i \geq 4$. Pro určení komplexity je tedy nutné projít všechny sousedy a sčítat počet přechodů z označených vrcholů na neoznačené a naopak. Komplexní vrcholy jsou buď částí význačné hrany se šířkou jedna (pro $c_i = 4$) nebo patří uzlu, kde se schází více význačných hran (pro $c_i > 4$). Dále definujeme ***střed*** a ***disk*** význačné oblasti. Nechť $F \subseteq \{1, \dots, N\}$ označuje význačnou oblast a $i \in F$ jeden

z jejích vrcholů V_i. Poté i je definované jako střed, pokud $\boldsymbol{nhd}\{i\} \subset F$. Množina $O_i := \{j \mid i\ je\ střed \wedge j \in \boldsymbol{nhd}\{i\} \setminus \{i\}\}$ určuje disk okolo středu i. Samozřejmě je možné, že vrchol označený jako disk může být také středem.

Nyní již můžeme formulovat operátor skeletonizace, který je podobný erozi, ovšem respektuje výše uvedené omezení. Nechť $C \subseteq F$ je množina všech komplexních vrcholů význačné oblasti. $O \subset F$ určuje sjednocení všech disků a $\odot \subset F$ je sjednocení všech středů. Operátor ***skeletonizace*** je definován jako

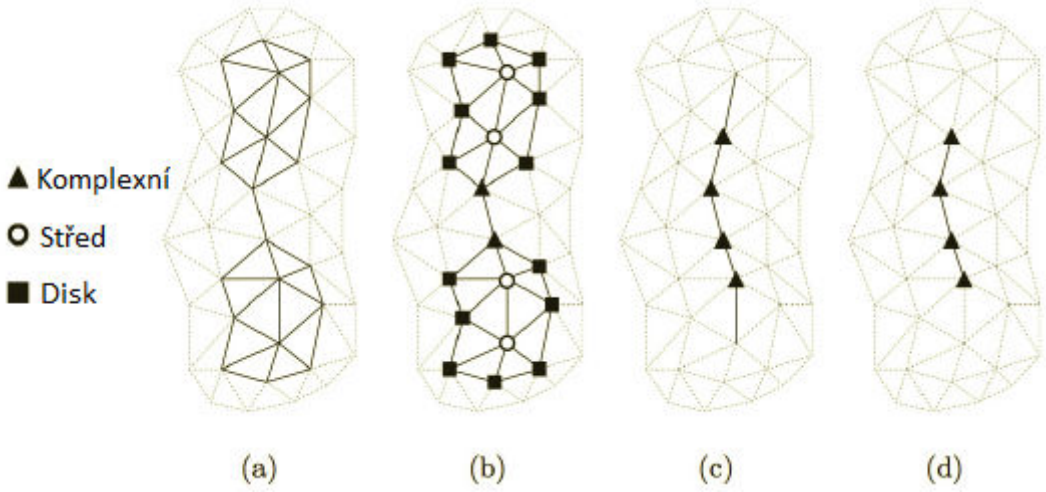
$$\boldsymbol{skeletonizace}(F) := F \setminus (O \cap \overline{C \cup \odot}). \tag{4.17}$$

Při každém použití tohoto operátoru jsou odstraněny vrcholy obsažené v množině disků a zároveň nacházející se na okraji význačné oblasti. Jestliže je tato oblast úzká, jsou jednotlivé vrcholy označeny jako komplexní, a proto nejsou odstraněny. Operátor skeletonizace se aplikuje iterativně, dokud dochází k nějakým změnám v oblasti. Změna nenastane, pokud $\odot = \emptyset$ nebo $\odot$ obsahuje pouze takové středy, jejichž disk má pouze komplexní vrcholy. Výsledkem této operace je množina F obsahující pouze komplexní vrcholy, až na konce „větví“, které nejsou připojené k žádnému dalšímu komplexnímu uzlu.

Malé „větve“ výsledné množiny jsou často nežádoucí, a proto bývají odstraňovány. K tomuto účelu je vytvořen operátor nazývající se prořezávání. Nechť $S \subseteq F$ je množina význačných vrcholů a $C \subseteq F$ je množina komplexních vrcholů. Poté operátor ***prořezávání*** je definován jako

$$\boldsymbol{prořezávání}(S) := S \setminus \bar{C}. \tag{4.18}$$

Ukázka těchto operací je vidět na Obr. 4.7.

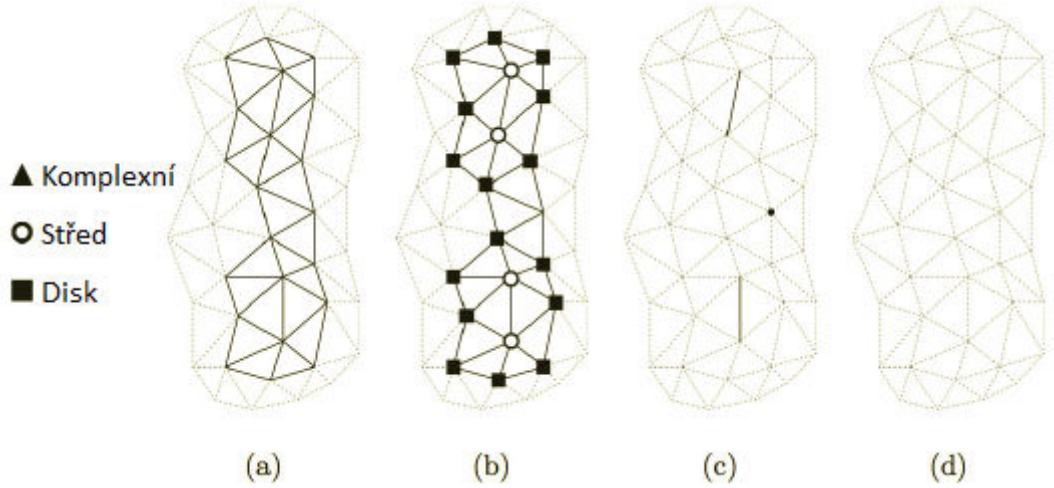


Obr. 4.7: Příklad skeletonizace a prořezávání. a) Význačná oblast. b) Rozdělení vrcholů do jednotlivých tříd. c) Výsledek po aplikování skeletonizace. d) Po použití prořezávání (převzato z [34]).

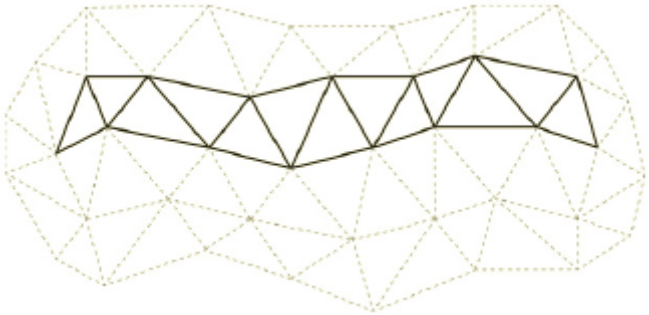
Posledním krokem algoritmu je pospojování vrcholů tak, aby tvořily výslednou linii hran. To je možné udělat pomocí trojúhelníků nacházejících se v modelu nebo interpolací hladkou křivkou.

4.3.1 Oprava chybné skeletonizace

Testováním výše uvedených morfologických operátorů byla u operace skeletonizace nalezena chyba, která se vyskytuje pouze v určitých konfiguracích trojúhelníkové sítě. Velmi často totiž může dojít k oddělení oblastí, které byly původně spojeny. Další chybou je možnost, že výsledný skeleton obsahuje i čáry tloušťky větší než jedna. Vzniklé problémy jsou zobrazeny na Obr. 4.8 a Obr. 4.9. Při hledání řešení byl objeven článek [34], který se tímto zabývá.



Obr. 4.8: Rozpojení význačné oblasti při použití skeletonizace. a) Význačná oblast. b) Rozdělení vrcholů do jednotlivých tříd. c) Po aplikování skeletonizace. d) Po prořezávání (převzato z [34]).



Obr. 4.9: Ukázkový příklad, kdy použitím skeletonizace se význačná oblast vůbec nezmění a po aplikování prořezávání celá zmizí. Dále je možné si všimnout, že vrcholy nejsou zařazeny do žádné třídy (převzato z [34]).

Pro správný algoritmus skeletonizace je nutné formulovat další definice. Jak ukazují výše uvedené obrázky, některé vrcholy nemusí být vůbec označeny (tj. přiřazeny do nějaké třídy). Proto je vytvořena další množina. Řekneme, že vrchol V_i je ***vnější***, pokud $F_i = 1$ a $i \notin (\mathrm{O} \cup \mathrm{C} \cup \odot)$. Množina všech vnějších vrcholů $\circledcirc$ je definována jako $\circledcirc := F \setminus (\mathrm{O} \cup \mathrm{C} \cup \odot)$. Jelikož vrchol během skeletonizace může změnit třídu (což

může mít za následek rozdělení spojených oblastí), zavedeme mezi nimi priority. Platí, že třída disků má menší prioritu než třídy ostatní. Pokud je vrchol klasifikován jako disk, může se během skeletonizace změnit např. na komplexní.

Zabránit rozdělení spojených oblastí můžeme tak, že nebudeme vyhazovat všechny vrcholy označené jako disky, ale pouze takové, které nezmění prioritu během skeletonizace. Toto vyžaduje přidání dalšího kroku do aktuálního algoritmu. Před každým vyhozením vrcholu je tedy přepočítána jeho třída. Jestliže např. vrchol zařazený mezi disky změní třídu na komplexní, nesmí být smazán. Je možné, že výsledná kostra bude příliš úzká pro použití této techniky (např. pokud obsahuje pouze vnější vrcholy, viz Obr. 4.9). Proto je nutné do algoritmu zařadit další část. Po aplikování předchozí metody obsahuje kostra pouze komplexní nebo vnější vrcholy. K získání výsledné úzké linie je potřeba použít další dva postupy:

1. Vnější vrcholy, které mají více než dva sousedy ve význačné oblasti F, jsou vyhozeny.
2. Vnější vrcholy s nejvýše jedním sousedem v F jsou ponechány.

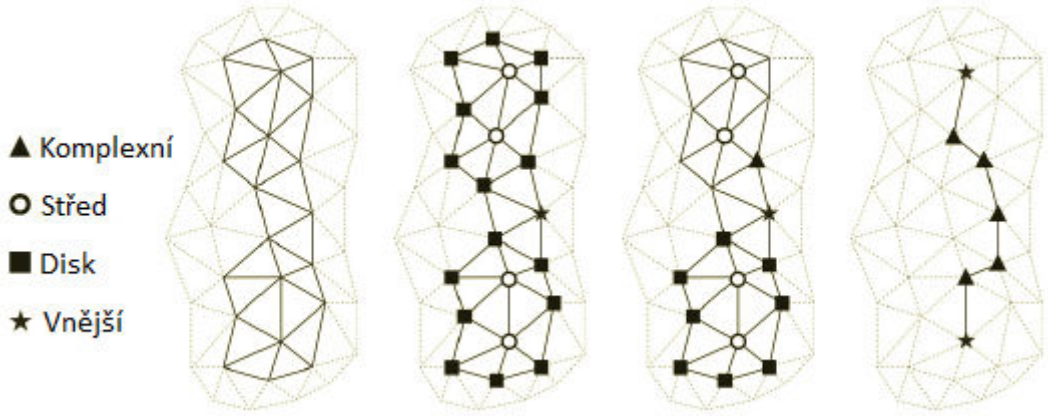
Celkový algoritmus skeletonizace je shrnut v pseudokódu na Obr. 4.10. Jeho výstup je zobrazen na Obr. 4.11 (lze porovnat s chybnou skeletonizací na Obr. 4.8).

```
Repeat
    For všechny vrcholy V_i ∈ F do
        If V_i je disk na okraji oblasti then
            Vypočítej komplexitu vrcholu V_i
            If priorita V_i se nezměnila then
                Vymaž V_i z množiny F
Until množina F zůstává stejná

Repeat
    For všechny vrcholy V_i ∈ F do
        If V_i je vnější then
            Vypočítej komplexitu vrcholu V_i
            If priorita V_i se nezměnila and počet sousedů v F > 2 then
                Vymaž V_i z množiny F
Until množina F zůstává stejná

Repeat
    For všechny vrcholy V_i ∈ F do
        If V_i je vnější then
            Vypočítej komplexitu vrcholu V_i
            If priorita V_i se nezměnila and počet sousedů v F > 1 then
                Vymaž V_i z množiny F
Until množina F zůstává stejná
```

Obr. 4.10: Pseudokód upraveného algoritmu skeletonizace (převzato z [34]).



Obr. 4.11: Ukázka upravené skeletonizace použité na stejná data (převzato z [34]).

5 Implementace navržené metody

Navržená metoda pro hledání význačných rysů na trojúhelníkových modelech byla implementována ve vývojovém prostředí *Microsoft Visual Studio 2010* s využitím programovacího jazyka *C#* a *.NET* frameworku verze 2.0. Vizualizace byla provedena s využitím knihovny *SlimDX*, která zapouzdřuje jednotlivé funkce grafické knihovny *DirectX* pro použití v prostředí *.NET*.

5.1 Vstupní soubory

V průběhu vývoje aplikace vznikaly požadavky na otestování navržené metody na různých trojúhelníkových modelech, které však byly uloženy v různých formátech. Proto výsledný program umožňuje načítání více druhů vstupních souborů, které zde budou stručně vysvětleny.

Prvním podporovaným formátem je výšková mapa ve formě šedotónového obrázku (tj. při použití *RGB* modelu mají všechny tři složky stejnou hodnotu). To si lze představit tak, že barva (resp. intenzita) jednotlivých pixelů udává nadmořskou výšku v konkrétním bodě. Nejčastěji černá barva reprezentuje nejnižší hodnotu a barva bílá nejvyšší. Po načtení všech pixelů představujících vrcholy o konkrétní výšce je nutné vytvořit trojúhelníkový model. Jelikož víme, že rozmístění jednotlivých vrcholů v rovině *xy* (*z* je výška) je pravidelné, můžeme jednoduše udělat pravidelnou mřížku. Vytvořená aplikace podporuje mnoho obrazových formátů - ***.jpg***, ***.png***, ***.gif***, ***.bmp*** a ***.tif***. Jediným omezením je rozlišení vstupního obrázku. Je vyzkoušeno, že program zvládá zobrazovat 4,5 miliónů vrcholů (tj. okolo 10 miliónů trojúhelníků při této reprezentaci), což splňuje rozlišení např. *3350x1300* pixelů. Při větším rozlišení může aplikace oznámit chybu, pokud nebude možné alokovat dostatek zdrojů na grafické kartě.

Druhým podporovaným formátem je ***.obj*** soubor, který se využívá i v komerční sféře. Jeho struktura je velmi jednoduchá. Lze uchovávat nejen klasické atributy modelu (pozice vrcholu, normála, apod.), ale také např. materiál nebo jeden objekt rozdělit na více skupin. Ve vytvořené aplikaci je implementována podpora pouze pro načítání trojúhelníků a pozice vrcholu, což ovšem pro naše účely postačuje. Triviálním příkladem tohoto formátu může být následující ukázka:

```
# 3ds Max Wavefront OBJ Exporter v0.94b - (c)2007 guruware

v  1.4870 0.3736 2.2576
v  1.5803 0.3451 2.1859
v  1.6275 0.3111 2.2261
v  -2.4608 1.5227 1.1936
# 4 vertices

f 1 2 3
f 1 3 4
# 0 polygons - 2 triangles
```

První slovo každé řádky udává její význam. V našem případě stačí načítat řádky označené slovem ***v*** (vrchol) nebo ***f*** (trojúhelník). Ostatní identifikátory lze jednoduše ignorovat. Při zpracovávání tohoto formátu je nutné dávat pozor na indexování trojúhelníků (obecně polygonů) od jedné.

Dalším všeobecně známým formátem je ***.ply*** soubor. Jeho struktura je opět jednoduchá, ovšem existují dvě verze – textová a binární. Z hlavičky souboru lze poznat, o kterou variantu se jedná. Vytvořený program předpokládá pouze textovou verzi. Na následující jednoduché ukázce si lze prohlédnout strukturu formátu:

```
ply
format ascii 1.0
comment generated by ply_writer
element vertex 4
property float x
property float y
property float z
element face 2
property list uchar int vertex_indices
end_header
0.032027 0.056575 -0.0503644
0.0317978 0.056575 -0.0503375
0.032377 0.0565645 -0.0503375
0.032377 0.056575 -0.0503416
3 0 1 2
3 0 2 3
```

Na začátku souboru se nachází hlavička označená identifikátorem ***ply*** a ukončená slovem ***end_header***. Z ní je možné vyčíst mnoho zajímavých informací, např. počet vrcholů modelu (*element vertex*), počet polygonů (*element face*), formát souboru (*format*) nebo datové typy jednotlivých atributů vrcholu (*property*). Hned za hlavičkou je na každé řádce uložen jeden vrchol, přičemž jeho atributy jsou oddělené mezerou. Výše uvedený příklad obsahuje pouze souřadnice bodu v prostoru, ale obecně toho může obsahovat více. Po všech vrcholech následuje na každé řádce definice jednoho polygonu, kde první číslo udává počet jeho bodů (pro trojúhelníkové modely vždy trojka).

Dále uvedené formáty souborů jsou pouze proprietárním řešením osob, které se zúčastnily konzultace této práce. Jejich přípona je vytvořena na základě iniciálů dané osoby. Struktura souboru ***.ik*** je názorně vysvětlena na následující ukázce (orientace trojúhelníků se předpokládá proti směru hodinových ručiček):

```
4                   # počet vrcholů
0.0   0.0   0.0     # na každé řádce jeden vrchol (souřadnice [x y z])
1.0   0.0   0.0
0.0   1.0   0.0
1.0   1.0   0.0
2                   # počet trojúhelníků
0   3   2           # na každé řádce jeden trojúhelník (indexováno od 0)
0   1   3
```

Jinou variantou jsou soubory s příponou ***.tbp*** a ***.tbt***. Pro jeden trojúhelníkový model musí být vytvořeny oba, a navíc umístěny ve stejné složce. Soubor ***.tbp*** obsahuje na každé řádce jeden vrchol se souřadnicemi $[x, y, z]$ oddělené mezerou. Druhý soubor má na každé řádce jeden trojúhelník indexován od jedné s orientací proti směru hodinových ručiček.

Soubor s příponou ***.lk*** má formát vysvětlený na následující ukázce:

```
4            # počet vrcholů
Z            # výšková osa (předpokládá se použití na terény)
743187.40    1044836.32    195.29       # jednotlivé vrcholy [x y z]
743188.25    1044832.39    195.27
743185.29    1044831.58    195.43
743189.29    1044833.58    195.50
2            # počet trojúhelníků
CCW          # orientace, CCW – proti směru hodinových ručiček, CW - opak
1            # od jakého čísla jsou trojúhelníky indexovány
1     2      3      # jednotlivé trojúhelníky
1     3      4
```

Posledním proprietárním formátem je soubor s příponou ***.js***. Na rozdíl od předchozích, tento soubor představuje výškovou mapu. Jeho strukturu si lze prohlédnout na následujícím příkladu:

```
ncols         6                          # počet sloupců
nrows         4                          # počet řádek
xllcorner     -815115,071                # pozice levého dolního rohu
yllcorner     -1083594,876
cellsize      30                         # vzdálenost mezi sousedy
NODATA_value  -9999                      # číslo určující nezadanou výšku
-9999 -9999 -9999 -9999 -9999 -9999      # jednotlivé výšky
-9999 15,68 235,2156 -9999 -9999 -9999
-9999 189,54 157,8 15,2164 -9999 -9999
-9999 155,234 155,2 251,56 -9999 -9999
```

5.2 Datové struktury

Vytvořená aplikace je založena na jednoduché datové struktuře uchovávající trojúhelníkovou síť. Jelikož lze načíst pouze jediný model, není nutné vytvářet komplikované struktury pro rychlé vyhledávání objektů ve scéně. V našem případě ovšem potřebujeme rychlý přístup k sousedním trojúhelníkům konkrétního vrcholu. Proto byla vytvořena datová struktura skládající se z následujících částí:

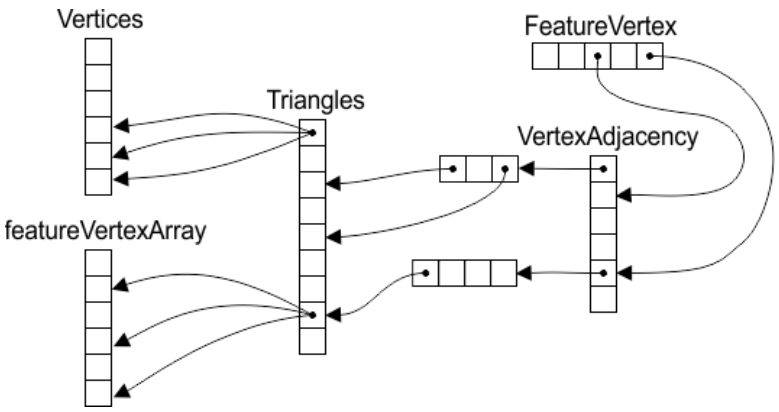
- ***Vertices*** - pole struktur datového typu *PositionNormalColored* obsahující jednotlivé vrcholy s jejich atributy.
- ***Triangles*** - pole struktur datového typu *Triangle* obsahující všechny trojúhelníky.
- ***VertexAdjacency*** - pole spojových seznamů, které obsahují indexy sousedních trojúhelníků (tj. indexy do pole *Triangles*) ke konkrétnímu vrcholu. Jeho velikost musí být totožná s velikostí pole *Vertices*.

Definice výše uvedených struktur vypadá následovně:

```
public struct PositionNormalColored
{
    public Vector3 Position;
    public Vector3 Normal;
    public int Color;
}
```

```
public struct Triangle
{
    public int v1;
    public int v2;
    public int v3;
}
```

Datový typ *Vector3* je struktura, která se skládá ze tří reálných čísel představující souřadnice *x*, *y* a *z*. Při používání programu se po aplikaci prahování vytvoří význačná oblast, která je dále využívána morfologickými operátory. K jejímu uchování byl vytvořen spojový seznam zvaný ***FeatureVertex***, který obsahuje indexy význačných vrcholů (tj. indexy do pole *Vertices* nebo *VertexAdjacency*). Pro snížení časové náročnosti morfologických operátorů bylo navíc vytvořeno pole ***featureVertexArray*** datového typu *bool*, které říká, jestli je daný vrchol součástí význačné oblasti. Tím dochází k redundanci dat, neboť tato informace je také uložena v seznamu *FeatureVertex*, ovšem za cenu toho, že v čase $O(1)$ lze zjistit „význačnost“ daného vrcholu (jinak bychom museli projít celý spojový seznam). Uvedené datové struktury a jejich propojení si je možné prohlédnout na Obr. 5.1.



Obr. 5.1: Datové struktury ve vytvořeném programu.

5.3 Ukázky implementovaných algoritmů

Z důvodu rozsáhlosti zde není možné popsat všechny algoritmy uvedené v kap. 4. Proto budou předvedeny pouze dvě kratší ukázky. Ostatní postupy je možné nalézt v dokumentovaných zdrojových kódech, které jsou dostupné na přiloženém *DVD*.

První ukázkou je morfologický operátor dilatace, která přidává do význačné oblasti nové vrcholy sousedící s jejím okrajem. To znamená, že pro konkrétní vrchol význačné oblasti zkontrolujeme, zdali v ní má všechny sousedy. Jestliže ne, tak takový vrchol do této oblasti přidáme. V opačném případě neděláme nic. Celý algoritmus si lze prohlédnout na následující ukázce:

```
public void Dilation(int neighbourhoodSize)
{
   // ziskej vrcholy
   Vertex.PositionNormalColored[] vertices = model.Vertices;
   // ziskej trojuhelniky
   Triangle[] triangles = model.Triangles;
   // ziskej sousedni trojuhelniky ke kazdemu vrcholu
   List<int>[] vertexAdjacency = model.VertexAdjacency;

   // velikost sousedstvi
   for (int k = 0; k < neighbourhoodSize; k++)
   {
      // uloz aktualni velikost (oblast se zvetsuje)
      int featureVertexCount = FeatureVertex.Count;
      // projdi vsechny vrcholy vyznacne oblasti
      for (int i = 0; i < featureVertexCount; i++)
      {
         // ziskej sousedni trojuhelniky k aktualnimu vrcholu
         List<int> adjacencyTriangles = vertexAdjacency[FeatureVertex[i]];
         // projdi vsechny sousedni trojuhelniky
         for (int j = 0; j < adjacencyTriangles.Count; j++)
         {
            Triangle currentTriangle = triangles[adjacencyTriangles[j]];
            // pokud vrchol v1 neni v oblasti, tak ho pridej
            if (!featureVertexArray[currentTriangle.v1])
            {
               FeatureVertex.Add(currentTriangle.v1);
               featureVertexArray[currentTriangle.v1] = true;
               vertices[currentTriangle.v1].Color = Color.Red.ToArgb();
            }
            // pokud vrchol v2 neni v oblasti, tak ho pridej
            if (!featureVertexArray[currentTriangle.v2])
            {
               FeatureVertex.Add(currentTriangle.v2);
               featureVertexArray[currentTriangle.v2] = true;
               vertices[currentTriangle.v2].Color = Color.Red.ToArgb();
            }
            // pokud vrchol v3 neni v oblasti, tak ho pridej
            if (!featureVertexArray[currentTriangle.v3])
            {
               FeatureVertex.Add(currentTriangle.v3);
               featureVertexArray[currentTriangle.v3] = true;
               vertices[currentTriangle.v3].Color = Color.Red.ToArgb();
            }
         }
      }
   }
   // uloz nove obarveni modelu
   model.SetNewVertices(vertices);
}
```

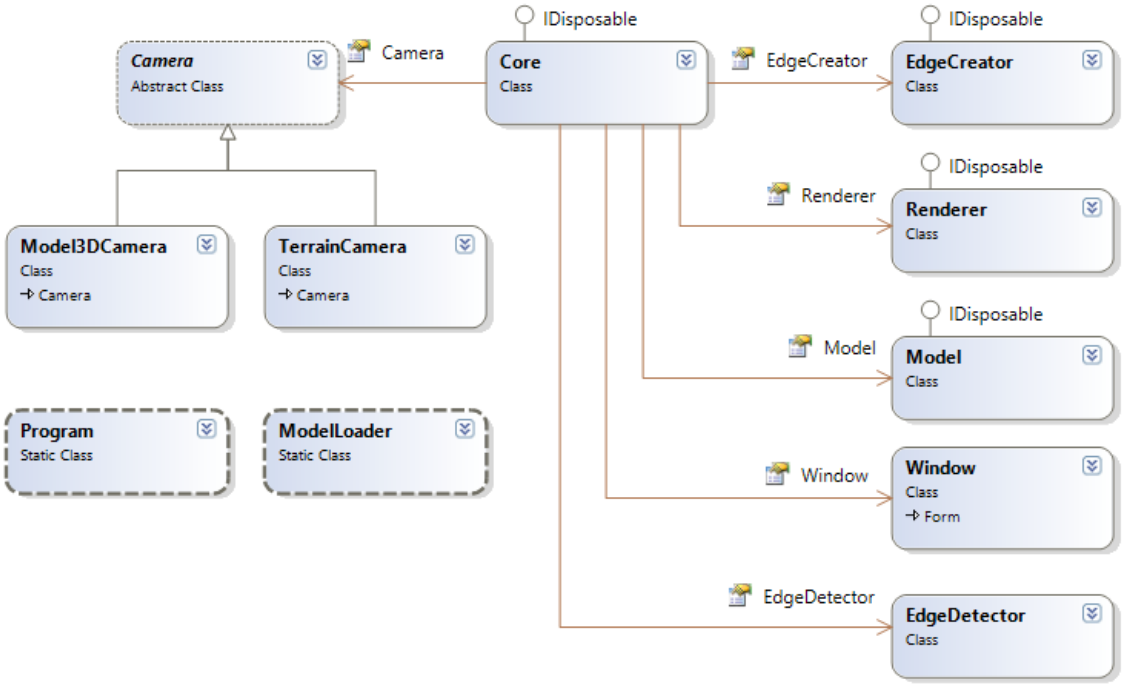
Druhou ukázkou je operace prořezávání, které je definováno vztahem (4.18). Z něho je zřejmé, že vyhazujeme pouze takové vrcholy význačné oblasti, které nejsou komplexní. Komplexitu vrcholu lze určit na základě vztahu (4.16). Jestliže je získané číslo menší než čtyři, vrchol z význačné oblasti odstraníme. V opačném případě neděláme nic. Celá metoda je shrnutá v následujícím kódu:

```
public void Pruning(int neighbourhoodSize)
{
   // ziskej vrcholy
   Vertex.PositionNormalColored[] vertices = model.Vertices;
   // vytvor pomocny seznam
   List<int> deletedVertices = new List<int>();
   // velikost sousedstvi
   for (int k = 0; k < neighbourhoodSize; k++)
   {
      deletedVertices.Clear();
      // projdi vsechny vrcholy vyznacne oblasti
      for (int i = 0; i < FeatureVertex.Count; i++)
      {
          // vypocti komplexitu, pokud je mensi nez 4 => vyhod vrchol
          int vertexComplexity = ComputeVertexComplexity(FeatureVertex[i]);
          if (vertexComplexity < 4)
          {
              deletedVertices.Add(FeatureVertex[i]);
              vertices[FeatureVertex[i]].Color = Color.White.ToArgb();
              FeatureVertex.RemoveAt(i);
              i--;
          }
      }
      // nastav odstranenym vrcholum false ve featureVertexArray
      for (int i = 0; i < deletedVertices.Count; i++)
      {
          featureVertexArray[deletedVertices[i]] = false;
      }
   }
   // uloz nove obarveni modelu
   model.SetNewVertices(vertices);
}
```

5.4 Objektový návrh a popis tříd

Vytvořený program se skládá celkem ze 14-ti tříd a 5-ti struktur. Po spuštění aplikace předá operační systém řízení metodě *Main()*, která se nachází ve statické třídě pojmenované *Program*. Ta má za úkol vytvořit instanci třídy *Core*, která představuje jádro celého programu. Hlavní metoda také volá v nekonečné smyčce metodu pro reakci na vstupy uživatele a překreslení zobrazovaného modelu (pokud je načten). Nejdůležitějším prvkem aplikace je ovšem třída *Core*. Mezi její hlavní úkoly patří vytvoření a zprostředkování přístupu k jednotlivým komponentám programu, kterými mohou být např. okno, kamera, hranový detektor nebo samotný trojúhelníkový model. Další významnou třídou je *Window*, která vytváří grafické uživatelské rozhraní (tzv. *GUI*) a stará se o reakci na uživatelské vstupy. To znamená, že pokud uživatel klikne na nějaké konkrétní tlačítko, tak v této třídě se nachází obslužná rutina zpracovávající danou událost.

Objektový model vytvořené aplikace si lze prohlédnout na *UML* diagramu zobrazeném na Obr. 5.2, který obsahuje nejdůležitější třídy a jejich vztahy. V následující části této kapitoly budou stručně popsány jednotlivé třídy a některé z jejich metod.



Obr. 5.2: Jednoduchý UML diagram vytvořené aplikace.

Program

Jak již bylo řečeno, tato třída spouští celou aplikaci a obsahuje pouze jedinou metodu – *void Main()*. Její význam byl vysvětlen výše.

Core

Hlavní třída zastřešující a uchovávající nejdůležitější objekty aplikace. Mezi její metody patří:

- *void Render()* – pomocí instance třídy *Renderer* volá metodu pro vykreslení aktuální scény.
- *void UpdateKeyboard()* – kontroluje, jestli byla stisknuta klávesa pro pohyb. Pokud ano, zavolá odpovídající metodu kamery.
- *void SetNewModel(Model model)* – nově načtený model uloží do vnitřní proměnné (případně starý uvolní z paměti).

Camera

Abstraktní třída definující společné vlastnosti všech kamer, mezi které patří její pozice, směr, „up“ vektor, „right“ vektor nebo transformační matice. Každá kamera také umožňuje uložit počáteční stav, do kterého se lze v případě potřeby vrátit pomocí metody *Reset()*. Mezi hlavní metody patří:

- *abstract void SetRotationX(float deltaX)* – zajišťuje rotaci kamery na základě pohybu myši ve směru *x*-ové osy.

- *abstract void SetRotationY(float deltaY)* – zajišťuje rotaci kamery na základě pohybu myši ve směru *y*-ové osy.
- *abstract void Move(MovementDirection direction)* – pohyb kamery podle stisknuté klávesy.

Model3DCamera

Tato třída je potomkem předchozí třídy. Je vytvořena pro použití na trojrozměrné modely. Kromě implementace výše uvedených abstraktních metod obsahuje také metodu *void SetHeightAxis(Axis axis)*, pomocí které lze změnit výškovou osu trojrozměrného objektu. Tím je možné jednoduše umístit zobrazovaný model do správné pozice.

TerrainCamera

Kamera, která je vytvořena speciálně pro použití na terénní data. Od její trojrozměrné verze se liší hlavně v ovládání. Dále neobsahuje metodu pro změnu výškové osy, neboť ta je jasně určena. Je opět potomkem třídy *Camera*.

EdgeCreator

Hlavním cílem této třídy je umožnění ručního vytváření hran v trojúhelníkovém modelu. Ty lze následně vyexportovat a dále s nimi pracovat. Mezi nejdůležitější metody patří:

- *void AddVertex(int vertexIndex)* – uloží nově vybraný vrchol do vnitřní struktury objektu. Dvojice vybraných vrcholů reprezentuje jednu hranu.
- *void RemoveVertex(int vertexIndex)* – vymaže všechny hrany obsahující zvolený vrchol.
- *void Render()* – vykreslí všechny vybrané hrany.
- *void Export(string fileName, int startIndex)* – umožňuje vyexportovat hrany do souboru s názvem *fileName* indexované od *startIndex*.

EdgeDetector

Jedna z velmi důležitých tříd, neboť obsahuje veškeré algoritmy pro hledání význačných hran. Navíc jsou v ní kromě křivostí uloženy vrcholy, které aktuálně patří do význačné oblasti. Mimo metod pro exportování význačných vrcholů nebo hran obsahuje třída následující metody (pouze výběr):

- *void ComputeVertexRank(VertexRankingMethod method, VertexRankingFilter filter, int percentForClamp)* – vypočítá ohodnocení vrcholů na základě vybrané metody (typ křivosti), zvoleného filtru (medián nebo průměr) a procenta oříznutí nejvyšších hodnot.
- *void VertexThresholding(float downThreshold, float upThreshold)* – aplikuje prahování na ohodnocené vrcholy pomocí dvou zvolených prahových hodnot.

- *void Dilation(int neighbourhoodSize)* – provede dilataci nad význačnou oblastí pomocí *N*-sousedství.
- *void Erosion(int neighbourhoodSize)* – vykoná erozi nad význačnou oblastí pomocí *N*-sousedství.
- *void Opening(int neighbourhoodSize)* – realizuje operaci otevření nad význačnou oblastí s použitím *N*-sousedství.
- *void Closing(int neighbourhoodSize)* - vypočte uzavření nad význačnou oblastí pomocí *N*-sousedství.
- *void Skeletonization()* – provede operaci skeletonizace nad význačnou oblastí.
- *void Pruning(int neighbourhoodSize)* – aplikuje operaci prořezávání na výslednou kostru.

Model

Třída reprezentující načtený trojúhelníkový model. Obsahuje také vrcholy, trojúhelníky, odkazy na sousední trojúhelníky ke každému vrcholu nebo transformační matici. Mezi hlavní metody patří:

- *void Render(bool solidRendering)* – umožňuje vykreslit trojúhelníkový model.
- *int PickVertex(Ray ray)* – vypočítá, zda došlo k průsečíku paprsku s modelem. Pokud ano, vrátí index vrcholu, který se nachází nejblíže.

ModelLoader

Statická třída obsahující metody pro načítání všech podporovaných formátů vstupních dat. Důležité metody jsou:

- *static Model Load3DModelFromFile(String fileName, Device graphicsDevice)* – umožňuje načítání trojrozměrných objektů z formátu ***.obj*** nebo ***.ply***.
- *static Model LoadTerrainFromFile(String fileName, Device graphicsDevice)* – načítání terénu ze souboru s příponou ***.ik***, ***.lk***, ***.js*** nebo ***.tbb*** (***.tbt***).
- *static Model LoadTerrainFromBitmap(Bitmap bitmap, Device graphicsDevice)* – načítání terénu z šedotónového obrázku.

Renderer

Třída zajišťující inicializaci a nastavení grafického adaptéru. Umožňuje vypnout osvětlení, nastavit nový směr světla, změnit barvu pozadí nebo přepnout režim vykreslování. Mezi hlavní metody patří:

- *void Render(Model model, EdgeCreator edgeCreator, Matrix matView)* – vykresluje veškeré objekty nacházející se ve scéně podle aktuálně nastavených parametrů zobrazování.
- *void Reset(Panel renderPanel)* – nastaví grafický adaptér na nové parametry při změně velikosti okna.

Utils

Statická třída obsahující obecně použitelné algoritmy. Příkladem může být metoda pro nalezení vrcholů definující osově zarovnaný obalový kvádr, výpočet normál ve vrcholech nebo vytvoření datové struktury pro získání sousedních trojúhelníků ke konkrétnímu vrcholu.

Window

Jedna z velmi důležitých tříd. Vytváří grafické uživatelské rozhraní celé aplikace a obsahuje všechny metody reagující na uživatelské vstupy, např. kliknutí na tlačítko nebo pohyb myši po zobrazovacím plátně.

WindowsInput

Třída má pouze jedinou metodu: *static bool IsKeyDown(Keys Key)*. Ta umožňuje zjistit, jestli byla stisknuta konkrétní klávesa. Oproti klasické metodě v *C#* je asynchronní, a proto podporuje stisk více kláves zároveň.

WindowsTimer

Statická třída sloužící k přesnému měření času. Používá se pro plynulý pohyb kamery na různě výkonných počítačích. Má dvě metody:

- *static void Refresh()* – uloží do vnitřní proměnné rozdíl časů předchozího zavolání této metody a aktuálního času.
- *static float GetDifference()* – vrátí rozdíl časů.

HSLColor

Struktura představující barvu v *HSL* barevném modelu. Navíc umožňuje pomocí metody *Color ToColor()* převod do *RGB* modelu. Důvodem vytvoření této struktury bylo, že *C#* nepodporuje *HSL* model.

Triangle

Struktura sloužící pro definici trojúhelníků. Obsahuje tři celočíselné indexy do pole *Vertices* (viz kap. 5.2).

Vertex

Třída zastřešující různé definice vrcholů. Obsahuje následující struktury:

- **PositionOnly** – představuje vrchol s jediným atributem – pozice.
- **PositionColored** – vrchol se dvěma atributy – pozice a barva.
- **PositionNormalColored** – vrchol obsahující tři atributy – pozice, normála a barva.

6 Testování a zhodnocení implementované metody

Testování navržené metody pro detekci hran probíhalo na počítači s následující konfigurací:

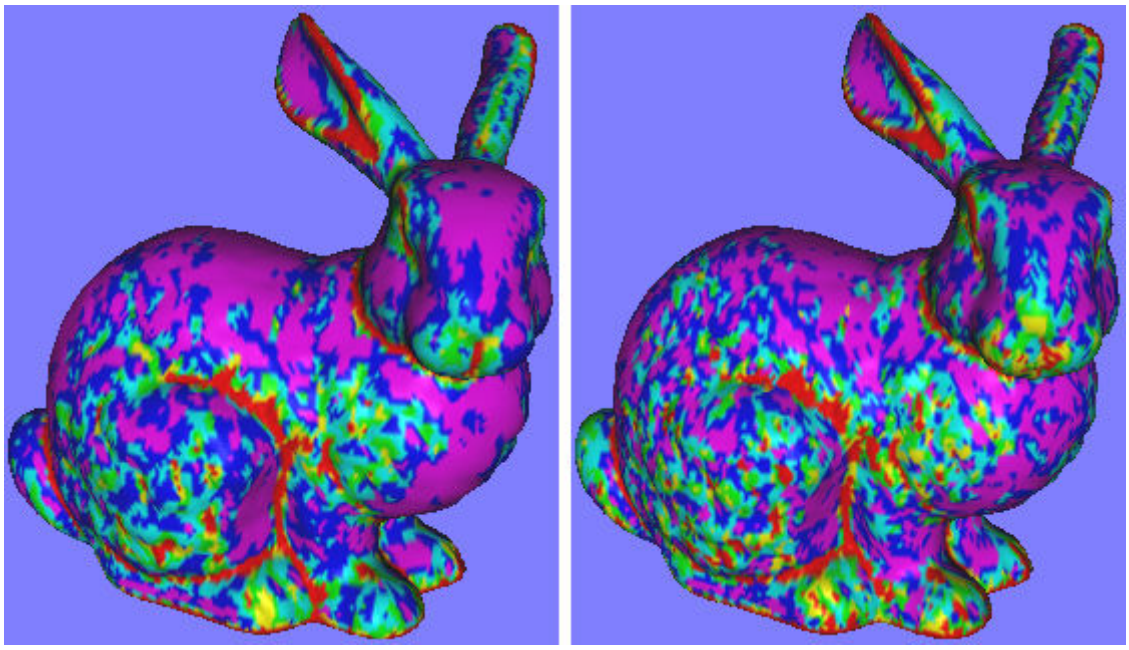
- *Procesor* - Intel Pentium B980 (2M cache, 2.4 GHz).
- *Systémová paměť* - 4 GB, DDR3 1333MHz.
- *Grafická karta* - integrovaná Intel HD Graphics.
- *Operační systém* - Microsoft Windows 8 64 bit.

6.1 Otestování jednotlivých částí metody

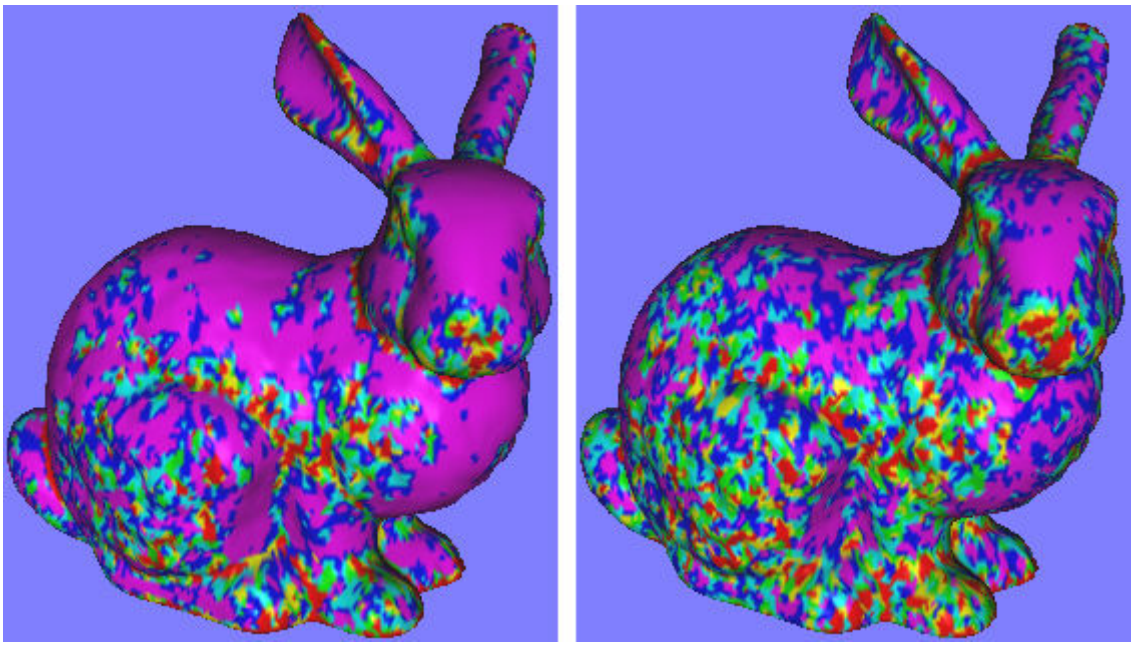
Tato kapitola se zabývá otestováním všech částí navržené metody, čímž bude ověřena správnost algoritmu.

6.1.1 Výpočet křivostí

V první fázi navržené metody dochází k ohodnocení všech vrcholů na základě diskrétní křivosti. V kap. 4.1 byly uvedeny čtyři možné varianty výpočtu (maximální, minimální, střední a Gaussova křivost). Jednotlivé možnosti byly porovnány na stejném trojúhelníkovém modelu (viz Obr. 6.1 a Obr. 6.2). Při výpočtu křivostí je vhodné použít filtr a oříznutí nejvyšších hodnot. Jako vhodná volba pro většinu modelů se jeví mediánový filtr a oříznutí 5% nejvyšších hodnot (význam těchto operací je vysvětlen v uživatelské dokumentaci, viz příloha B).

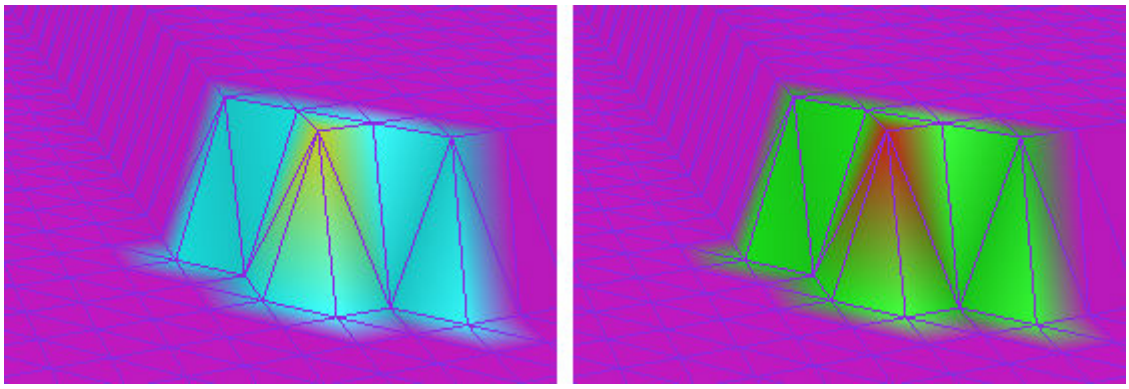


Obr. 6.1: Ohodnocení vrcholů pomocí diskrétní křivosti (filtr: medián, oříznutí hodnot: 5%). Červená barva značí místa s nevyšší hodnotou a fialová s nejnižší. Vlevo: maximální křivost. Vpravo: střední křivost. Model převzat z [39].



Obr. 6.2: Ohodnocení vrcholů pomocí diskrétní křivosti (filtr: medián, oříznutí hodnot: 5%). Vlevo: Gaussova křivost. Vpravo: minimální křivost. Model převzat z [39].

Z výše uvedených obrázků je patrné, že nejhůře dopadla minimální a Gaussova křivost. To je ovšem předpokládaný stav vzhledem k jejich definici. Obě křivosti jsou nulové v místech plochy, kde dochází k zakřivení pouze v jednom směru (viz Obr. 6.3). Proto jsou nevhodné pro hledání význačných hran. Pokud by bylo cílem detekce nalézt ostré „špičky“ nebo „důlky“, pak je správnou volbou Gaussova křivost. Nejvhodnější variantou je podle Obr. 6.1 maximální křivost, protože neobsahuje tolik „falešných“ míst, které do význačné oblasti nepatří a zakřivené úseky jsou více kompaktní.

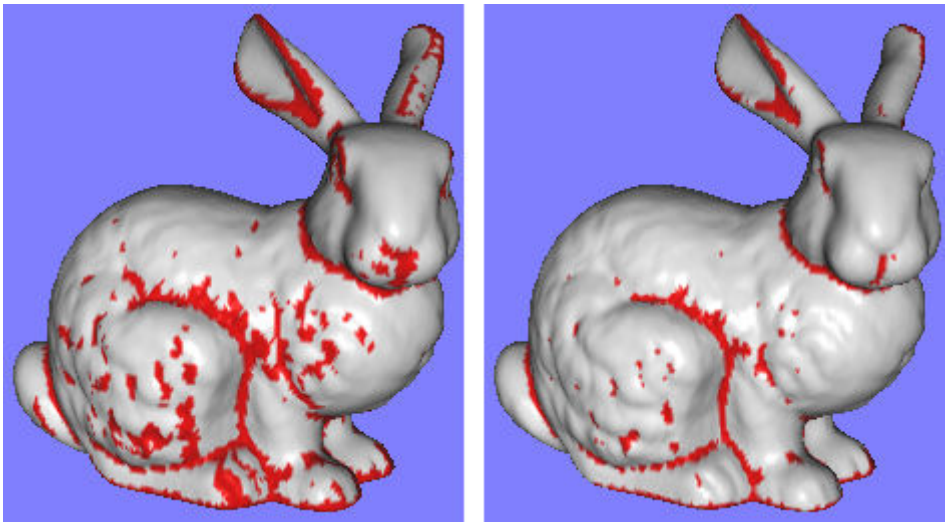


Obr. 6.3: Ukázka zakřivení části plochy pouze v jednom směru (filtr: žádný, oříznutí hodnot: 0%). Vlevo: Gaussova křivost. Vpravo: minimální křivost.

6.1.2 Prahování

Po výpočtu ohodnocení je nutné vybrat pouze takové vrcholy, které jsou pro nás zajímavé. Jelikož vyhledáváme místa plochy, kde dochází k jejímu zakřivení, zajímají nás vysoké hodnoty. Nastavení prahových mezí automaticky je velmi obtížnou záležitostí, protože jednotlivé modely mohou být tvarově rozmanité a různě veliké.

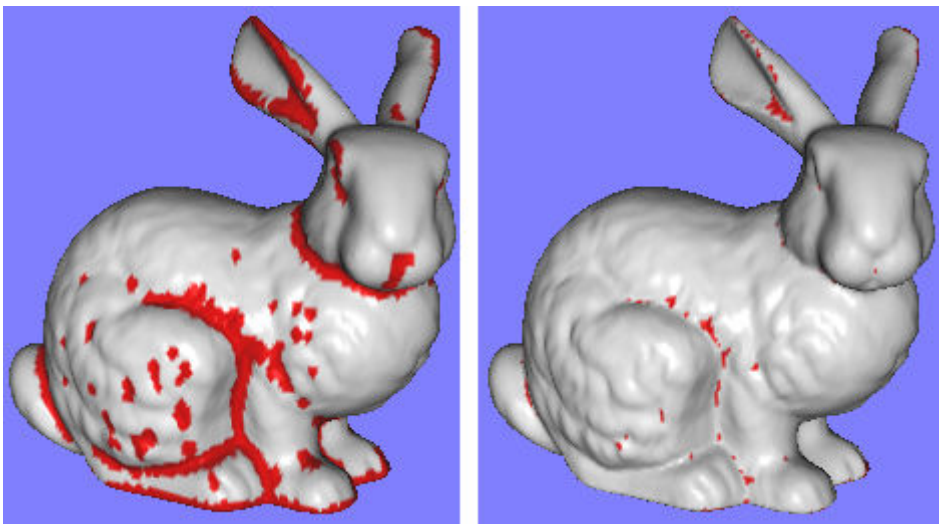
Jednou z možností je vybírat zvolený počet procent nejvyšších hodnot. Pokud ovšem vstupní data reprezentují rovinu, dostáváme se do problémů. Proto jsou většinou prahové hodnoty zadávány uživatelem programu. Po aplikaci prahování je vytvořena význačná oblast. Příklad takové oblasti pro různě zvolené meze lze vidět na Obr. 6.4.



Obr. 6.4: Význačná oblast po aplikaci prahování. Zvolené parametry: křivost – maximální, filtr – medián, ořízntutí hodnot – 5%, rozsah ohodnocení - <0,84; 4,97>. Levý obrázek: nastavené prahy [3; 4,97]. Pravý obrázek: nastavené prahy [4,2; 4,97]. Model převzat z [39].

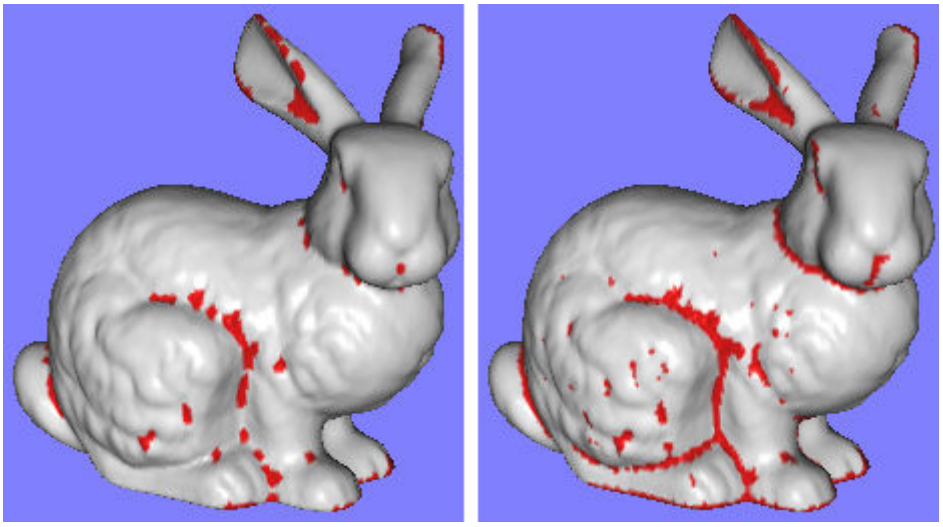
6.1.3 Morfologické operátory

Jestliže nám význačná oblast po aplikaci prahování nevyhovuje, můžeme využít morfologické operátory k její úpravě. Dilataci lze využít pro přidání neoznačených vrcholů uvnitř význačné oblasti nebo na jejím okraji (dojde také k celkovému zvětšení oblasti). Erozi lze použít k odstranění nepotřebných „větví“, ale je potřeba dávat pozor na šířku význačné oblasti. Jestliže je v některé části široká pouze dva vrcholy (nebo méně), tak dojde k jejímu zpřetrhání. Příklad dilatace a eroze je vidět na Obr. 6.5 (původní význačná oblast je stejná jako pravá část Obr. 6.4).



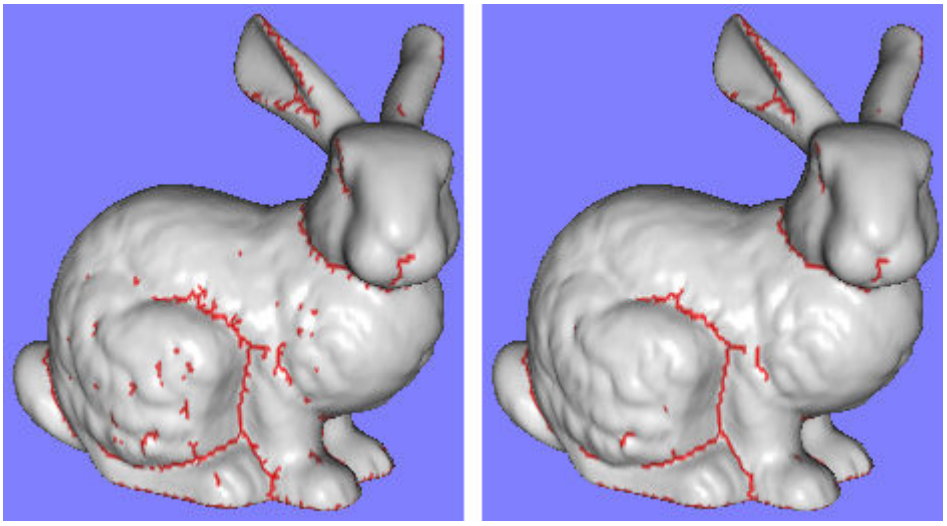
Obr. 6.5: Vlevo: dilatace. Vpravo: eroze (oblast je úzká, došlo k poškození). Model převzat z [39].

Pro naše účely je spíše vhodnější operátor uzavření nebo otevření. Pomocí uzavření lze vyplnit díry nebo spojit sousední části význačné oblasti při zachování původní velikosti. Operátor otevření se používá k odstranění malých „ostrovů“ nebo „větví“. Je-li oblast příliš úzká, může dojít k jejímu zpřetrhání. Příklady těchto operací jsou názorně vidět na Obr. 6.6 (původní význačná oblast je stejná jako pravá část Obr. 6.4).



Obr. 6.6: Vlevo: operátor otevření (došlo k odstranění malých „ostrovů“, ale zároveň také k porušení význačné linie z důvodu úzké oblasti). Vpravo: operátor uzavření (v určitých místech došlo k napojení sousedních částí význačné oblasti). Model převzat z [39].

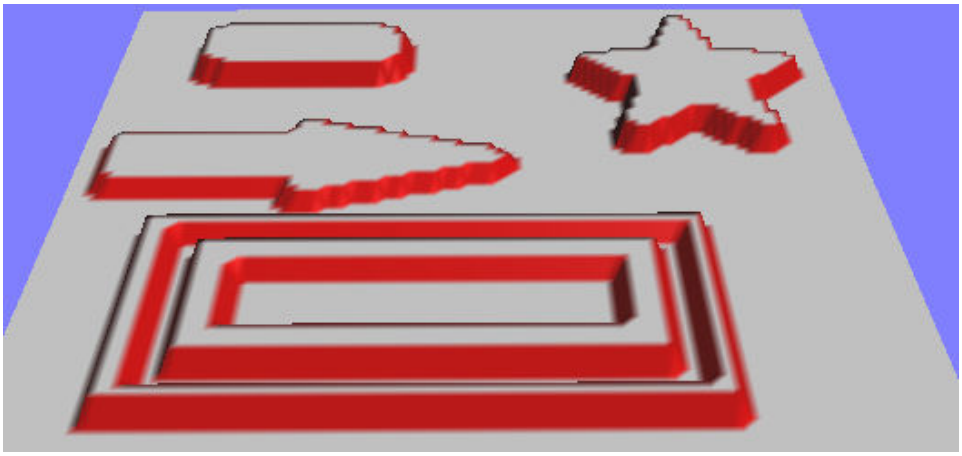
Pokud chceme získat význačné hrany ve formě úzké linie, je nutné aplikovat operátor skeletonizace. Tím dojde vytvoření kostry význačné oblasti. Na získané hrany lze následně použít operaci prořezávání k odstranění kratších úseků, které jsou často nežádoucí. Ukázka zmíněných operací je zobrazena na Obr. 6.7 (původní oblast je stejná jako pravá část Obr. 6.6).



Obr. 6.7: Vlevo: použití operátoru skeletonizace. Vpravo: na získanou kostru bylo aplikováno 2x prořezávání, čímž došlo k odstranění nežádoucích úseků. Model převzat z [39].

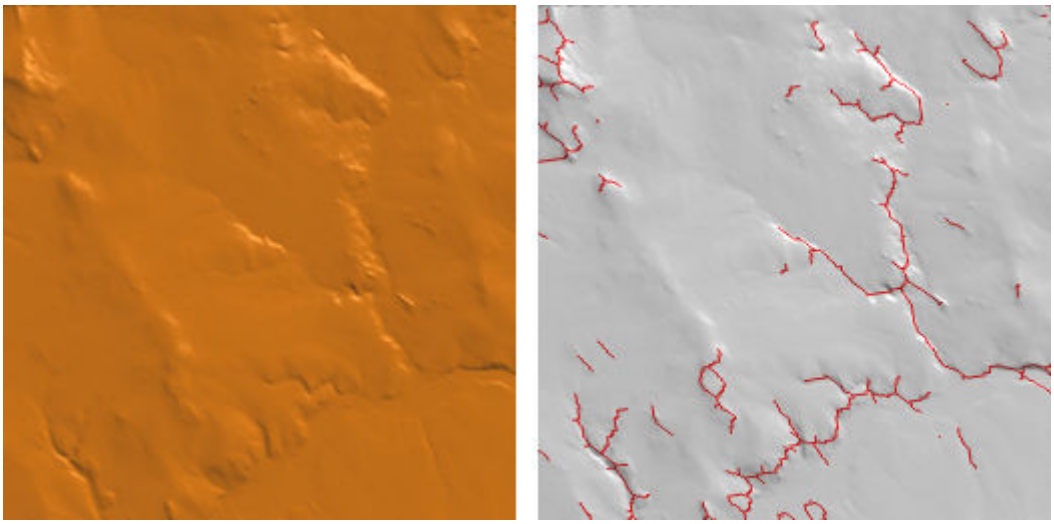
6.2 Testování na terénních datech

Navržená metoda byla podrobena testům nejen na uměle vytvořených datech, ale také na datech reálných. V této kapitole budou uvedeny terény (kromě prvního), které byly získány na základě bodů nasnímaných při *DPZ*. Prvním příkladem je jednoduchý terén vytvořený podle výškové mapy ve formě šedotónového obrázku. Z Obr. 6.8 je patrné, že metoda nalezla všechny význačné hrany, které se v terénu nacházejí. Navíc u takto jednoduchých dat ani není potřeba používat morfologické operátory.



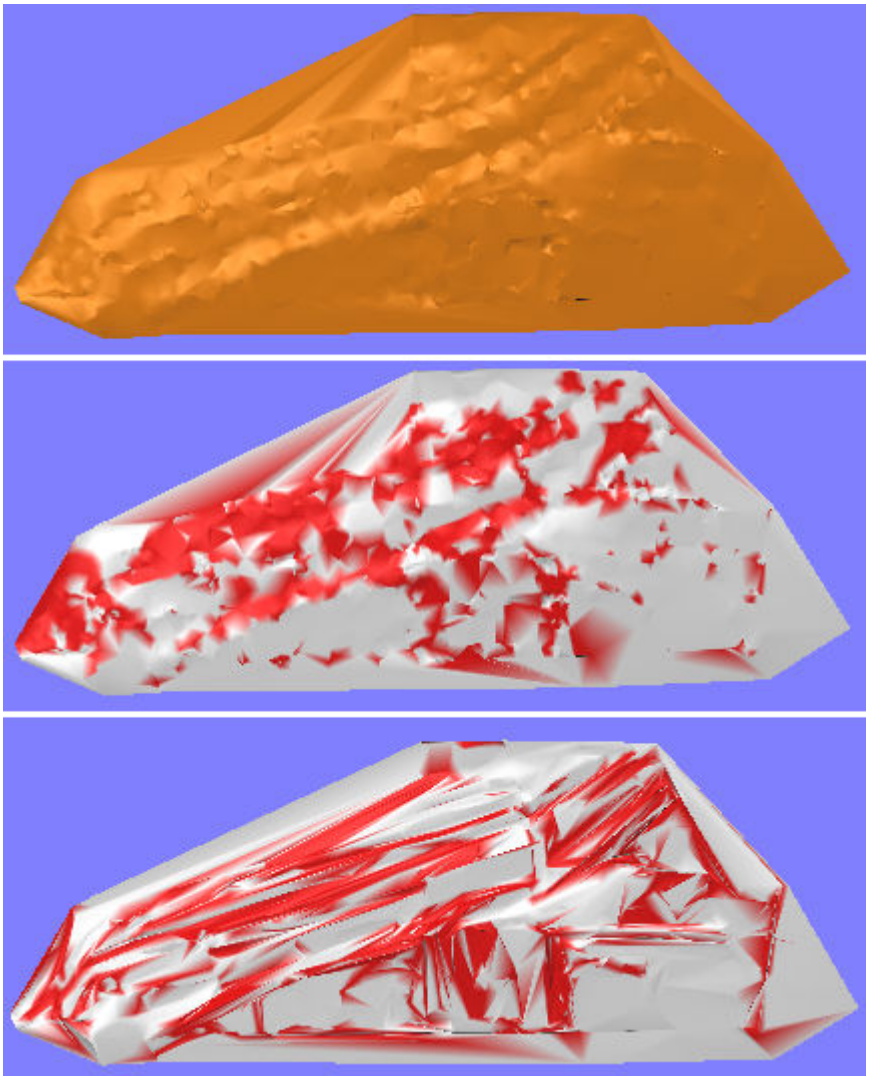
Obr. 6.8: Ukázka detekce význačných hran na uměle vytvořeném terénu.

Druhým příkladem je nasnímaný terénem okolí zámku Kozel ve formě výškové mapy. Levá část Obr. 6.9 ukazuje, že tento terén je daleko komplikovanější, neboť neobsahuje žádné ostré zlomy. Pravá část zobrazuje nalezené hrany. Je vidět, že navržená metoda dokáže při vhodně zvolených parametrech nalézt hrany i na velmi komplikovaných terénech.



Obr. 6.9: Nasnímaný terén v okolí zámku Kozel (převzat z [38]). Vlevo: model terénu. Vpravo: nalezené hrany.

Další ukázkou jsou data nasnímaná v oblasti botanické zahrady. Z obdržených bodů byly vytvořeny dvě triangulace – Delaunayova a datově závislá s využitím simulovaného žíhání. Výstup navržené metody pro obě triangulace je vidět na Obr. 6.10.

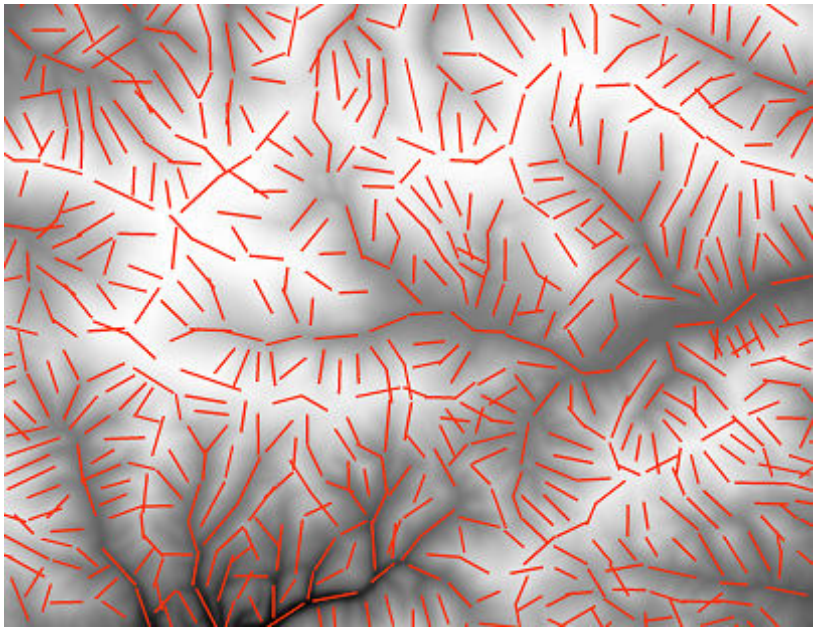


Obr. 6.10: Nahoře: původní model botanické zahrady. Uprostřed: nalezené hrany při použití Delaunayovy triangulace. Dole: nalezené hrany při použití datově závislé triangulace. Terén převzat z [36].

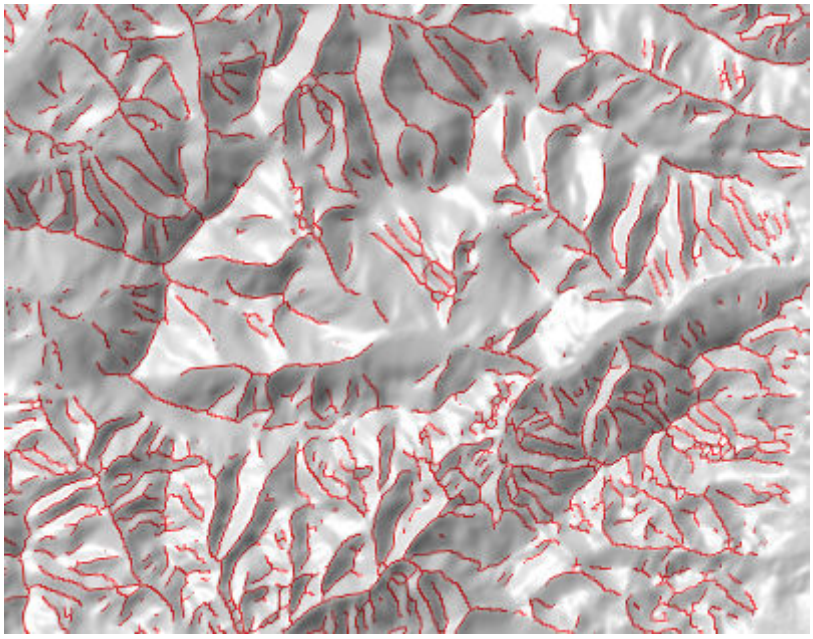
Z uvedeného obrázku je patrné, že metoda na těchto vstupních datech selhala. Model botanické zahrady obsahuje „terasy“, které byly detekovány jen stěží. Příčinou je fakt, že obě triangulace obsahují mnoho lokálních míst, ve kterých dochází k velkému zakřivení, ale z globálního hlediska do význačné oblasti nepatří. Navíc vizualizace nalezených hran nepůsobí hezky, protože terén obsahuje malé množství trojúhelníků. Tím dojde při interpolaci barvy k rozmazání. Jedním ze záměrů na těchto datech bylo prokázat, že pomocí navržené metody (resp. nalezených hran) lze vylepšit triangulaci použitím Delaunayovy triangulace s omezením. Výstup metody na těchto datech je ovšem natolik špatný, že experiment nemá smysl.

6.2.1 Porovnání navržené metody

Implementovaný algoritmus pro detekci hran na trojúhelníkových modelech byl porovnán s metodou Ing. Jakuba Šilhavého, který v rámci své disertační práce vytvořil postup pro automatické vymezování morfolineamentů (tj. údolnic a hřbetnic). Tato metoda je rastrově založený algoritmus, tudíž funguje pouze pro terénní data. Porovnávání probíhalo na oblasti Turčanské kotliny nacházející se na Slovenku (vstupní data jsou ve formě výškové mapy). Výstup algoritmu Ing. Jakuba Šilhavého je zobrazen na Obr. 6.11. Výsledné hrany námi navržené metody jsou vidět Obr. 6.12.



Obr. 6.11: Výstup metody Ing. Jakuba Šilhavého pro oblast Turčanské kotliny. Převzato z [38].



Obr. 6.12: Výstup námi navržené metody pro oblast Turčanské kotliny. Terén převzat z [38].

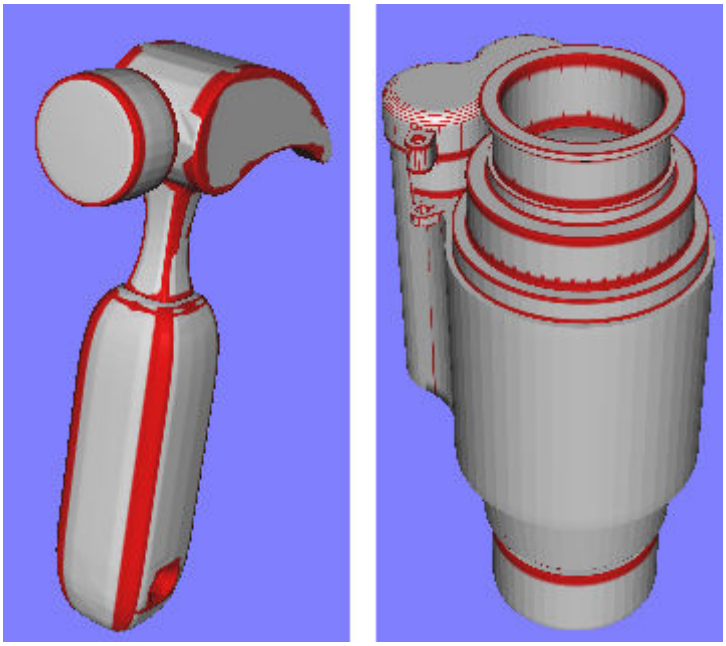
Uvedené metody nelze snadno porovnat, neboť algoritmus Ing. Jakuba Šilhavého je specializován na vyhledávání hran ve formě úseček, které mezi sebou nemusejí být propojeny. Můžeme však jednoduše vizuálně porovnat navržený postup s jinou metodou. Z výše uvedených obrázků lze vypozorovat, že námi navržený algoritmus nedopadl vůbec špatně. Byla nalezena většina hran a jednotlivé terénní linie nejsou zpřetrhány. Menším problémem je široké údolí nacházející se uprostřed obrázku. Dále nebyly oproti porovnávané metodě detektovány oblasti, ve kterých dochází k malé změně tvaru.

6.3 Testování na trojrozměrných modelech

Jelikož je navržená metoda založena na trojúhelníkových sítích, můžeme ji jednoduše použít i na trojrozměrné objekty. V této kapitole budou uvedeny tři specializované skupiny modelů.

6.3.1 Technické modely

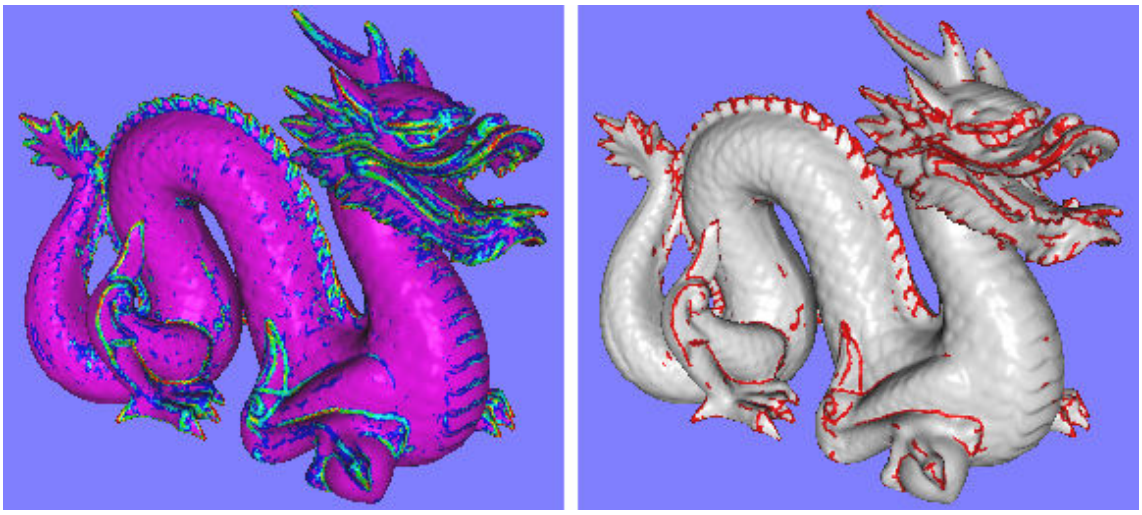
Tato skupina obsahuje technické součástky, které často mívají pouze ostré hrany. Příklad nalezených hran na technických modelech je uveden na Obr. 6.13. Lze si všimnout, že většina hran byla nalezena správně. Při testování však bylo zjištěno, že problémy mohou dělat ostré hrany, u kterých jsou trojúhelníky z jedné strany malé a z druhé velké. To ovšem platí pouze v případě, když model obsahuje také ostré hrany, u kterých jsou z obou stran malé trojúhelníky. Důvodem je skutečnost, že pokud jsou malé trojúhelníky z obou stran, dochází ke stejnému zakřivení, ale na daleko menší ploše. Zde se tudíž nabízí možnost navrženou metodu dále vylepšit.



Obr. 6.13: Nalezené hrany na technických součástkách. Modely převzaty z [40].

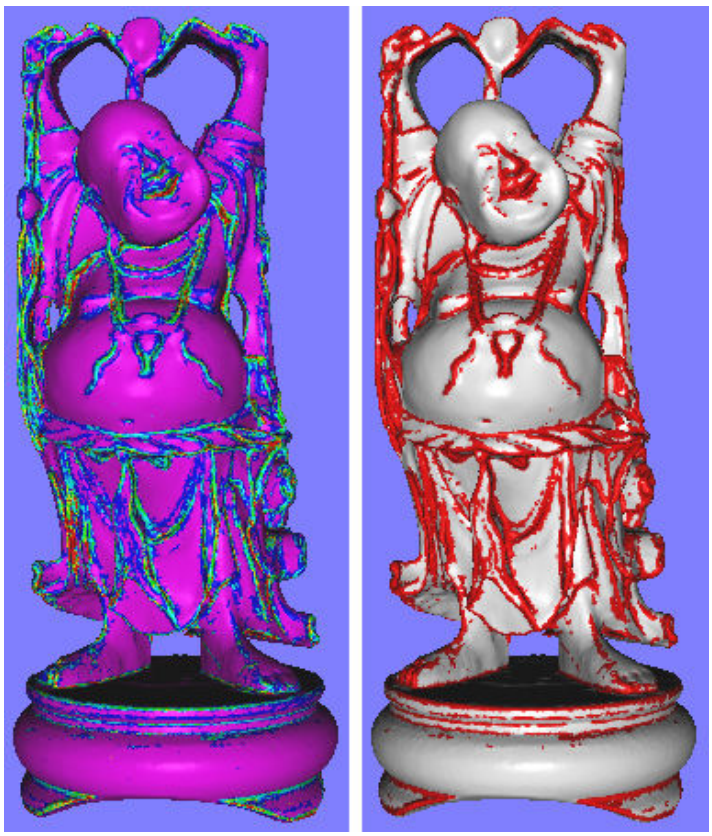
6.3.2 Obecné modely

V této kapitole bude navržená metoda otestována na obecně známých modelech, které většinou neobsahují ostré hrany. Prvním objektem je model draka. Výstup algoritmu je zobrazen na Obr. 6.14.



Obr. 6.14: Vlevo: ukázka po výpočtu ohodnocení vrcholů (maximální křivost). Vpravo: nalezené hrany. Model převzat z [39].

Druhou ukázkou je slavný model *Happy Buddha* (viz Obr. 6.15).



Obr. 6.15: Vlevo: ukázka po výpočtu ohodnocení vrcholů (maximální křivost). Vpravo: nalezené hrany. Model převzat z [39].

Z uvedených obrázků je vidět, že metoda je schopná zpracovávat i velmi složité objekty obsahující přes milión trojúhelníků. Vhodným pořadím morfologických operátorů je možné výsledek ještě vylepšit (např. odstranit malé „ostrovy“).

6.3.3 Modely hlav

Poslední skupinou trojrozměrných objektů, na kterých byla metoda otestována, jsou objekty představující model lidské hlavy. Ukázka výstupu je zobrazena na Obr. 6.16. Na těchto modelech navržený algoritmus bez problémů najde všechny význačné části lidské hlavy – uši, oči, nos a ústa. Rozpoznané hrany je možné dále využití v jiných aplikacích.

Obr. 6.16: Vlevo: ukázka po výpočtu ohodnocení vrcholů (maximální křivost). Vpravo: nalezené hrany. Model převzat z [37].

6.4 Zhodnocení navržené metody

Z výše uvedeného testování vyplývá, že navržená metoda dává uspokojivé výsledky pro většinu trojúhelníkových sítí. Pokud však model obsahuje mnoho lokálních míst, ve kterých dochází k velkému zakřivení a tato místa z globálního hlediska nepatří do význačné oblasti, metoda selhává. To je ovšem problém všech lokálně zaměřených algoritmů. Velkou výhodnou navržené metody je možnost aplikace morfologických operátorů, pomocí kterých můžeme odstranit nežádoucí artefakty. Z testování vyplývá, že pro naše účely se nejvíce hodí operátor skeletonizace, uzavření a otevření (popř. prořezávání). Dále se jako vhodná volba parametrů pro většinu modelů jeví následující konfigurace: křivost – maximální, filtr – medián, oříznutí hodnot – 5%, prahy – závisí na konkrétním modelu, skeletonizace (popř. před skeletonizací použít uzavření). Ukázky testování na dalších modelech je možné nalézt v příloze A.

7 Závěr

Cílem této diplomové práce bylo navrhnout, implementovat a otestovat metodu pro rozpoznávání význačných rysů na trojúhelníkových modelech. Metoda měla být schopná zpracovávat jak data terénní, tak i trojrozměrná. Samotnému návrhu algoritmu předcházela fáze analýzy současných metod. Ty je možné rozdělit do dvou skupin – rastrové algoritmy a metody založené na trojúhelníkových sítích. Nevýhodou rastrového přístupu je možnost fungování pouze pro terénní data. S těmito algoritmy se setkáváme zejména v oblasti geoinformatiky a kartografie. Naproti tomu metody založené na trojúhelníkových sítích mohou spolehlivě zpracovávat i trojrozměrné objekty. Tato reprezentace je velmi častá v počítačové grafice. Při analýze současného stavu bylo zjištěno, že většina metod pro detekci význačných hran je založena na rastrovém přístupu. Zbývající metody (tj. algoritmy založené na trojúhelníkových sítích) jsou buď velmi jednoduché, nebo příliš složité. Mezi jednoduché postupy patří např. nalezení hran na základě úhlu mezi přilehlými trojúhelníky. Nevýhoda těchto metod je, že jsou téměř nepoužitelné na hladších modelech. Druhým přístupem jsou algoritmy, které se snaží rekonstruovat původní plochu (lokálně nebo globálně). Jejich nevýhoda spočívá ve velké časové náročnosti, tudíž jsou téměř nepoužitelné na rozsáhlé trojúhelníkové sítě.

Navržená metoda v této práci se snaží sjednotit výhody obou přístupů, tj. rychlost výpočtu a zároveň použitelnost na různé trojúhelníkové modely. Její hlavní myšlenka pochází z článku [33], avšak některé části algoritmu byly upraveny z důvodu nalezených chyb nebo plně nahrazeny jinou metodou. Navržený postup lze rozdělit do tří částí: ohodnocení vrcholů, získání význačné oblasti a použití morfologických operátorů. Původní algoritmus ohodnocení vrcholů byl založen na principu lokální rekonstrukce plochy. To je ovšem časově náročný postup, proto byla tato část metody nahrazena výpočtem diskrétní křivosti, která se určuje přímo z trojúhelníkového modelu. Na základě získaných hodnot poté dochází k prahování, čímž dostaneme význačnou oblast. Prahové hodnoty jsou silně závislé na konkrétním modelu, proto je jejich nastavení ponecháno na uživateli aplikace. Poslední částí metody je použití morfologických operátorů, pomocí kterých je možné odstranit nežádoucí artefakty nebo získat kostru význačné oblasti.

Implementovaný algoritmus byl otestován na terénních i trojrozměrných datech. Dále došlo k porovnání navržené metody s algoritmem Ing. Jakuba Šilhavého, který v rámci disertační práce vytvořil postup pro automatické vymezování hřbetnic a údolnic. Navržená metoda dává dobré výsledky téměř na každém trojúhelníkovém modelu. Problém se vyskytuje pouze u modelů obsahujících mnoho lokálních míst, ve kterých dochází k velkému zakřivení, ale z globálního hlediska jsou nevýznačná. To je ovšem záležitost všech lokálně orientovaných algoritmů. Další problém nastává v případě, pokud jsou u význačné hrany z jedné strany malé trojúhelníky a ze strany druhé velké. Zde je tedy možnost metodu dále vylepšit.

Přehled zkratek

DMT – *Digitální model terénu*

DMR – *Digitální model reliéfu*

DEM – *Digital Elevation Model*

DPZ – *Dálkový průzkum Země*

GIS – *Geografický informační systém*

TIN – *Triangulated Irregular Network*

LoG – *Laplacian of Gaussian*

DoG – *Difference of Gaussians*

SOD – *Second Order Difference*

CAD – *Computer-aided design*

ESOD – *Extended Second Order Difference*

BFP – *Best Fit Polynomial*

ABBFP – *Angle Between Best Fit Polynomials*

GUI – *Graphical User Interface*

UML – *Unified Modeling Language*

Literatura

[1] **Bayer, Tomáš.** *Digitální modely terénu*. Praha: Přírodovědecká fakulta UK.

[2] **Oršulák, Tomáš a Pacina, Jan.** *3D modelování a virtuální realita*. Ústí nad Labem: Centrum pro virtuální realitu a modelování krajiny, 2010.

[3] **Rapant, Petr.** *Digitální modely reliéfu IV*. Ostrava: Institut geoinformatiky VŠB-TU, 2009.

[4] **Vichrová, Martina.** *Topografické mapování*. Plzeň: Fakulta aplikovaných věd ZČU, 2011.

[5] *Základy geometrie křivek a ploch.* České Budějovice: Fakulta pedagogická JČU.

[6] **Bastl, Bohumír.** *Základy diferenciální geometrie křivek a ploch*. Plzeň: Fakulta aplikovaných věd ZČU.

[7] **Ježek, František.** *Geometrické a počítačové modelování*. Plzeň: Fakulta aplikovaných věd ZČU, leden 2010.

[8] **Tomiczková, Světlana a Ježek, František.** *Diferenciální geometrie-Plochy*. Plzeň: Fakulta aplikovaných věd ZČU, duben 2014.

[9] **Tomiczková, Světlana a Ježek, František.** *Diferenciální geometrie-Plochy 2. část*. Plzeň: Fakulta aplikovaných věd ZČU, duben 2014.

[10] **Hellus, Jiří.** *Bakalářská práce – Křivosti ploch a jejich aplikace*. Plzeň: Fakulta aplikovaných věd ZČU, 2012.

[11] **Horák, Karel.** *Matematická morfologie*. Brno: Fakulta elektrotechniky a komunikačních technologií VUT.

[12] **Hlaváč, Václav.** *Matematická morfologie*. Praha: Fakulta elektrotechnická ČVUT.

[13] **The Scientist and Engineer's Guide to Digital Signal Processing.** *Morphological Image Processing, http://www.dspguide.com/ch25/4.htm* [online] [citace: 29.4.2014].

[14] **Karlíček, Lukáš.** *Semestrální práce ze ZVI – Detekce hran ve snímku*. Plzeň: Fakulta aplikovaných věd ZČU, 2013.

[15] **Academic dictionaries and encyclopedias.** *Laplacian of Gaussian, http://de.academic.ru/dic.nsf/dewiki/828780* [online] [citace: 29.4.2014].

[16] **E-learning.** *Detekce hran na základě průchodu druhé derivace obrazové funkce nulou, http://e-learning.tul.cz/cgi-bin/elearning/elearning.fcgi?ID_tema=67& ID_obsah=1146&stranka=publ_tema&akce=polozka_vstup* [online] [citace: 29.4.2014]

[17] **Wikipedia.** *Canny edge detector, http://en.wikipedia.org/wiki/Canny_edge_ detector* [online] [citace: 29.4.2014].

[18] **Sui, Lichun.** *Processing of Laser Scanner Data and Automatic Extraction of Structure Lines*. International Archives of Photogrammetry and Remote Sensing, Volume 34, Part2, Commission II, p. 429-436, August 2002.

[19] **Brügelmann, Regine.** *Automatic Break Line Detection from Airborne Laser Range Data*. Survey Department, Section of Remote Sensing and Photogrammetry, Working Group III/5, Netherlands, 2002.

[20] **Berkhahn, Volker and Mai, Shephan.** *Detection of Terrain Features Embedded in a Pre-Processor for Topographic Data*. 7^{th} International Conference on Hydroinformatics, France, 2006.

[21] **Berkhahn, V., Kaapke, K., Rath, S. and Pasche, E.** *A Hybrid Meshing Scheme Based on Terrain Feature Identification*. 14^{th} International Meshing Roundtable, p. 129-145, 2005, ISBN: 978-3-540-25137-8.

[22] **Briese, Ch., Mandlburger, G., Ressl, C., Brockmann, H.** *Automatic Break Line Determination the Generation of a DTM along the River Main*. Laserscanning'09, Volume 38, p. 236-241, France, 2009.

[23] **Brzank, A., Lohmann, P., Heipke, C.** *Automated Extraction of Pair Wise Structure Lines Using Airborne Laserscanner Data in Coastal Areas*. International Society for Photogrammetry and Remote Sensing, p. 36-41, Netherlands, September 2005.

[24] **Kraus, K. and Pfeifer, N.** *Advanced DTM Generation from LIDAR Data.* International Archives of Photogrammetry and Remote Sensing, Volume 34-3, p. 23-30, October 2001.

[25] **Pedrini, Hélio and Schwartz, William Robson.** *Topographic Feature Identification Based on Triangular Meshes*. 9^{th} International Conference, CAIP 2001, p. 621-629, Poland, 2001, ISBN: 978-3-540-42513-7.

[26] **Hubeli, A., Meyer, K. and Gross, M.** *Mesh Edge Detection*. CS Technical Report, Institute of Scientific Computing, Switzerland, December 2000.

[27] **Nealen, A., Igarishi, T., Sorkine, O., Alexa, M.** *Laplacian Mesh Optimization*. GRAPHITE '06, p. 381-389, ISBN: 1-59593-564-9.

[28] **Reuter, M., Biasotti, S., Giorgi, D., Patane, G., Spagnuolo, M.** *Discrete Laplace-Beltrami Operators for Shape Analysis and Segmentation*. Computer & Graphics, Volume 33, Issue 3, p. 381-390, June 2009.

[29] **Wehr, A., Petrescu, E., Duzelovic, H., Punz, Ch.** *Automatic Break Line Detection Out of High Resolution Airborne Laser Scanner Data*.

[30] **Ohtake, Y., Belyaev, A., Seidel, H.-P.** *Ridge-Valley Lines on Meshes via Implicit Surface Fitting*. Journal ACM Transactions on Graphics, Volume 23, Issue 3, p. 609-612, August 2004.

[31] **Yoshizawa, S., Belyaev, A., Seidel, H.-P.** *Fast and Robust Detection of Crest Lines on Meshes*. ACM symposium on Solid and Physical Modeling, p. 227-232, ISBN: 1-59593-015-9.

[32] **Watanabe, K. and Belyaev, A.** *Detection of Salient Curvature Features on Polygonal Surfaces*. Computer Graphics Forum, Volume 20, Issue 3, p. 385-392, September 2001.

[33] **Rössl, Ch., Kobbelt, L., Seidel, H.-P.** *Extraction of Feature Lines on Triangulated Surface Using Morphological Operators*. Smart Graphics, AAAI Symposium, p. 71-75, 2000.

[34] **Kudelski, D., Viseur, S., Mari, J.-L.** *Skeleton Extraction of Vertex Sets Lying on Arbitrary Triangulated 3D Meshes*. 17^{th} IAPR International Conference, DGCI, p. 203-214, 2013, ISBN: 978-3-642-37066-3.

[35] **Meyer, M., Desbrun, M., Schröder, P., Barr, A. H.** *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*. Visualization and Mathematics III, p. 35-57, 2003, ISBN: 978-3-662-05105-4.

[36] **Bayer, Tomáš.** *Katedra aplikované geoinformatiky a kartografie*. Praha: Přírodovědecká fakulta UK.

[37] **Martínek, Petr**. *Katedra informatiky a výpočetní techniky.* Plzeň: Fakulta aplikovaných věd ZČU.

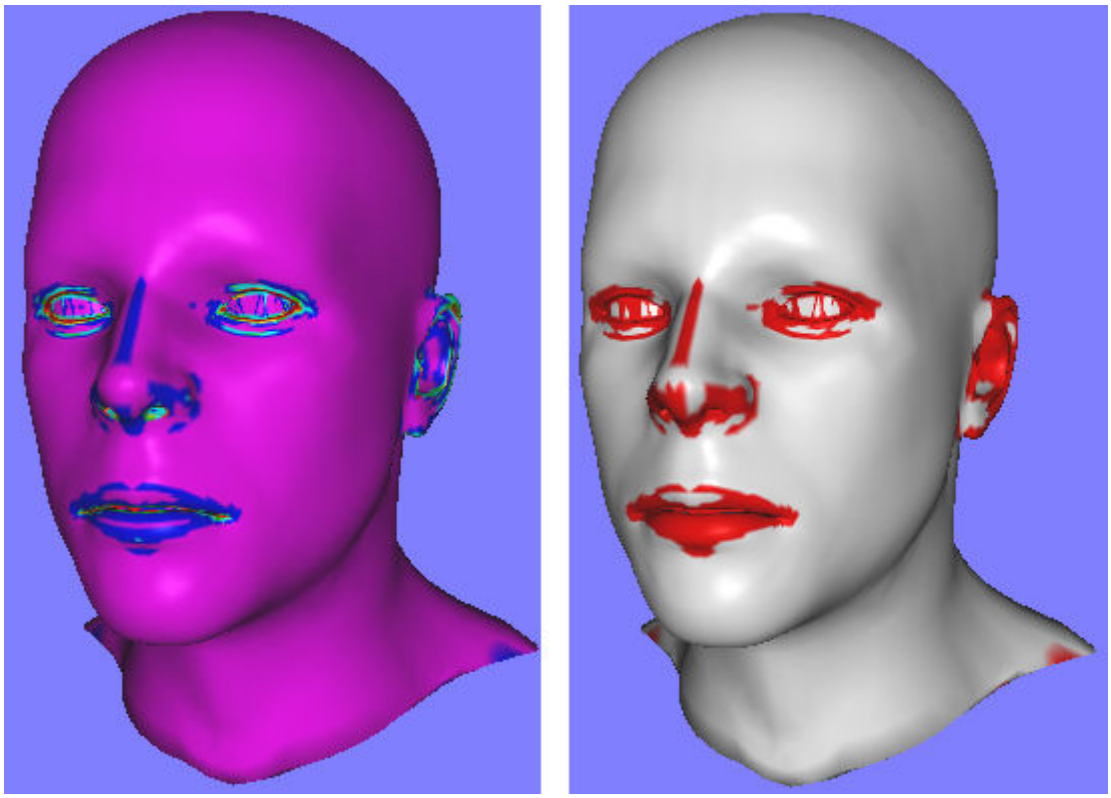
[38] **Šilhavý, Jakub**. *Katedra matematiky*. Plzeň: Fakulta aplikovaných věd ZČU.

[39] **CMLab Graphics.** *3D Models, http://graphics.csie.ntu.edu.tw/~robin/courses/cg04/model/index.html* [online] [citace: 14.5.2014].

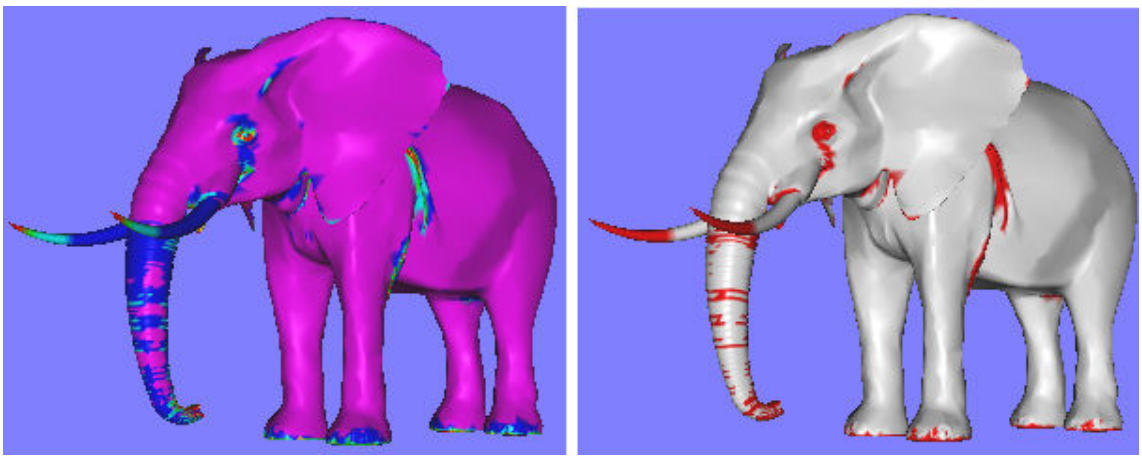
[40] **3D CAD Browser.** *3D Models, http://www.3dcadbrowser.com* [online] [citace: 14.5.2014].

Přílohy

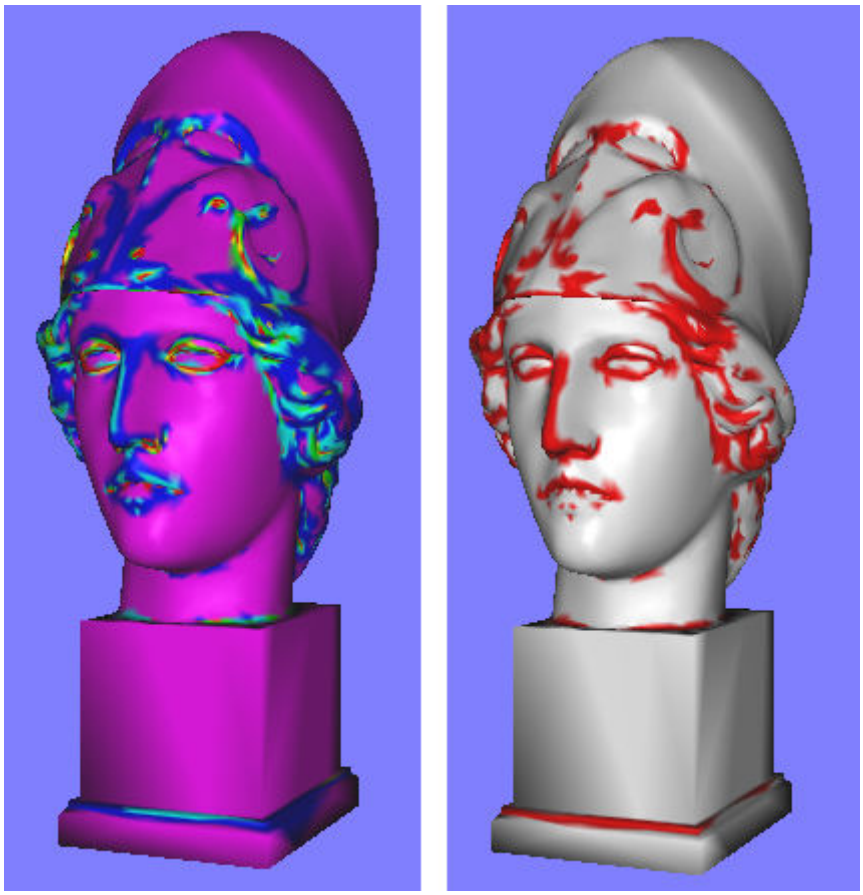
A Testování na dalších modelech



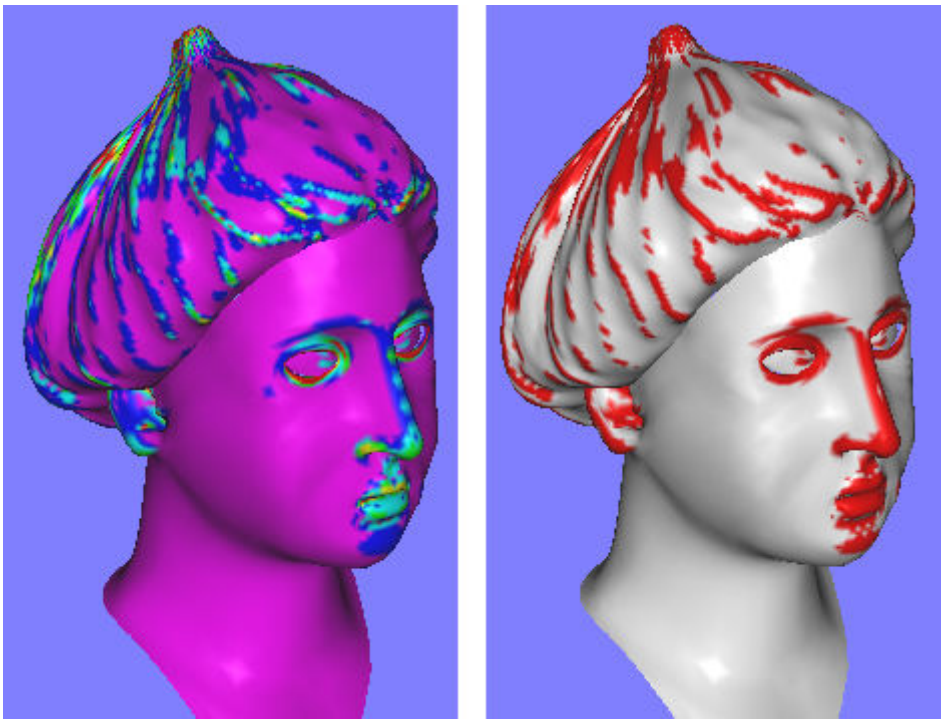
Obr. A.1: Model lidské hlavy (převzato z [37]). Vlevo: ohodnocení vrcholů. Vpravo: nalezené hrany.



Obr. A.2: Model slona (převzato z [39]). Vlevo: ohodnocení vrcholů. Vpravo: nalezené hrany.



Obr. A.3: Model sochy (převzato z [39]). Vlevo: ohodnocení vrcholů. Vpravo: nalezené hrany.

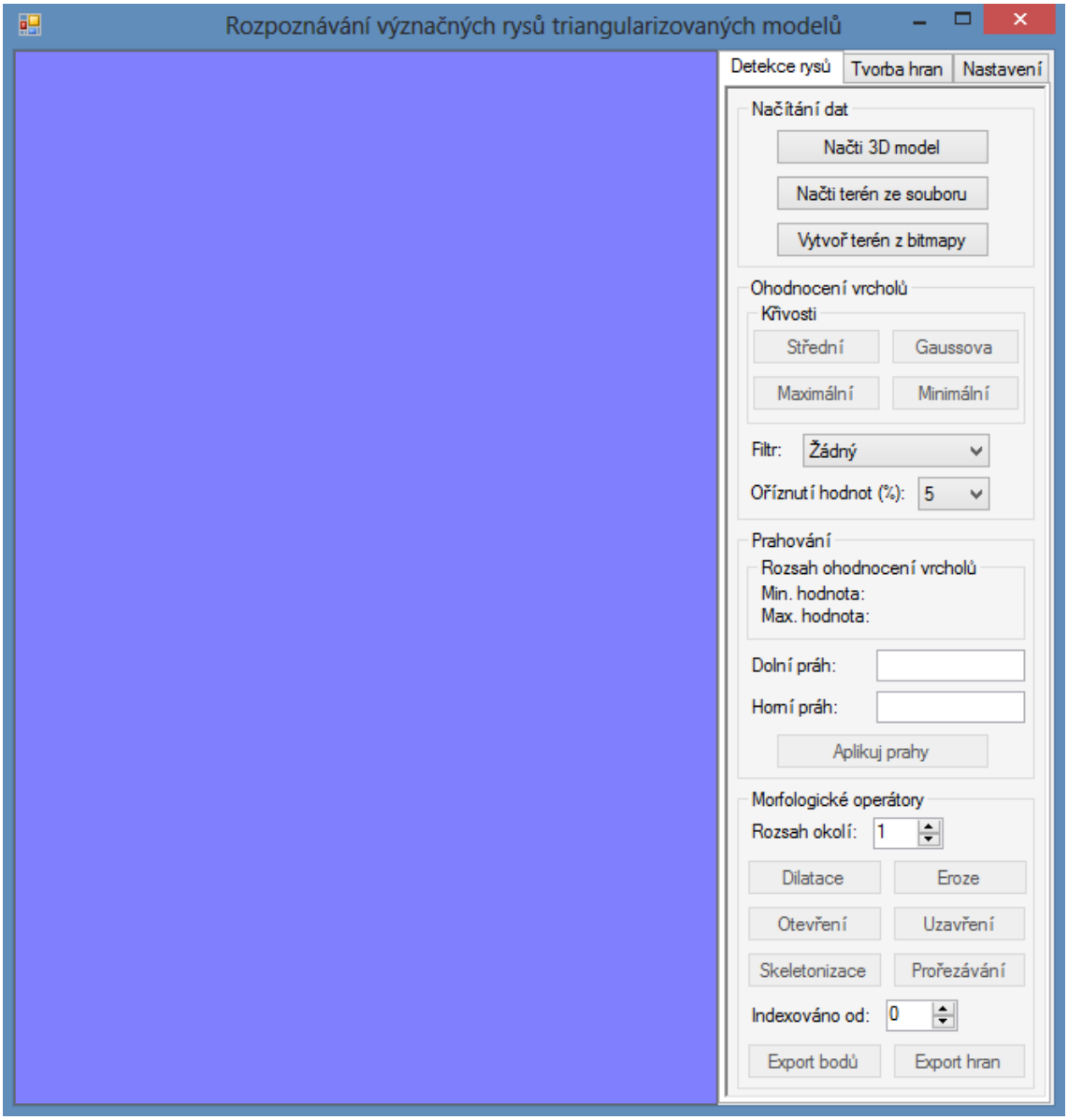


Obr. A.4: Model hlavy s čepicí (převzato z [40]). Vlevo: ohodnocení vrcholů. Vpravo: nalezené hrany.

Obr. A.5: Model zvířete (převzato z [40]). Vlevo: ohodnocení vrcholů. Vpravo: nalezené hrany.

B Uživatelská dokumentace

Vytvořenou aplikaci lze spustit pomocí souboru *Diplomova_prace.exe*. Je nutné, aby na daném počítači byl nainstalován *.NET* framework verze minimálně 2.0 (mají všechny *Windows* od verze *XP*) a běhové prostředí knihovny *SlimDX*, která se nachází na přiloženém *DVD*. Po spuštění se zobrazí formulář, který je vidět na Obr. B.1.



Obr. B.1: Grafické uživatelské rozhraní vytvořené aplikace.

Z výše uvedeného obrázku je patrné, že okno je rozděleno na dvě části. V levé části se nachází panel sloužící pro zobrazování načteného modelu. Pravá část obsahuje veškeré nastavení a možnosti aplikace. Navíc je rozdělena na tři záložky:

- ***Detekce rysů*** – obsahuje vše potřebné pro detekci význačných hran na základě navržené metody.

- ***Tvorba hran*** – ruční vytváření hran na trojúhelníkové síti. Hrany lze následně vyexportovat do souboru.
- ***Nastavení*** – umožňuje změnit nastavení zobrazování nebo kamery.

Načítání dat

Při používání aplikace je potřeba nejprve načíst model. To lze udělat pomocí tlačítek nacházejících se v záložce *Detekce rysů*. Možné varianty jsou:

- ***Načti 3D model*** – načítání dat ze souboru s příponou ***.obj*** nebo ***.ply***.
- ***Načti terén ze souboru*** – umožňuje načíst terén ze souboru s příponou ***.ik***, ***.lk***, ***.js*** nebo ***.tbb*** (***.tbt***).
- ***Vytvoř terén z bitmapy*** – podporuje načítání terénu z obrazového formátu.

Manipulace s objektem

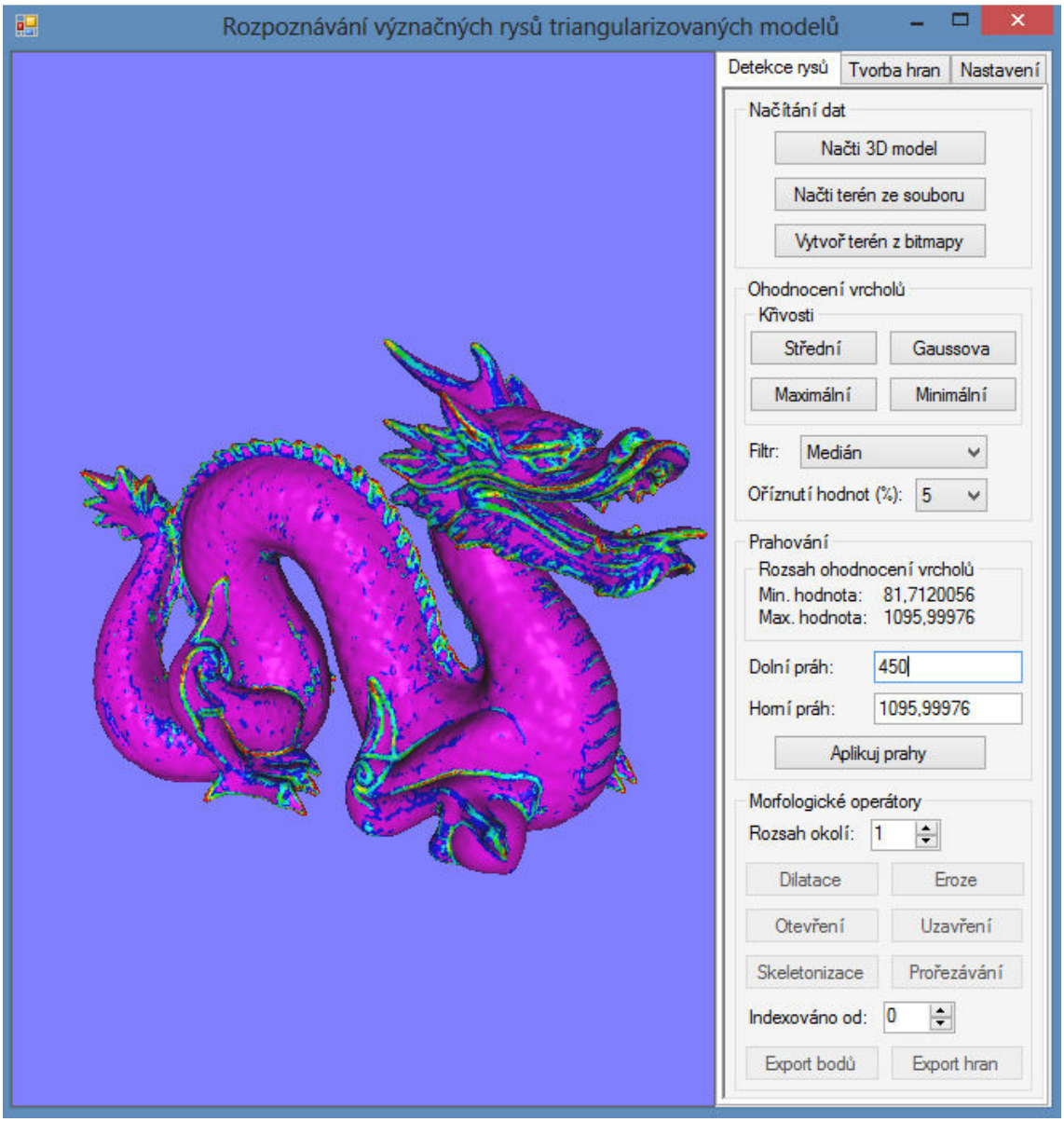
S načteným modelem je možné jednoduše manipulovat. Způsob ovládání je ovšem závislý na typu dat:

- ***Terénní model***:
  - Pro pohyb je možné využít klávesy *W*, *A*, *S*, *D*.
  - Otáčet se lze stisknutím levého tlačítka myši nad zobrazovacím plátnem a následným táhnutím.
- ***Trojrozměrný model***:
  - Otáčet s objektem je možné stejným postupem jako u terénu.
  - Přibližování (resp. oddalování) modelu lze provést pomocí klávesy *W* (resp. *S*).

Detekce hran

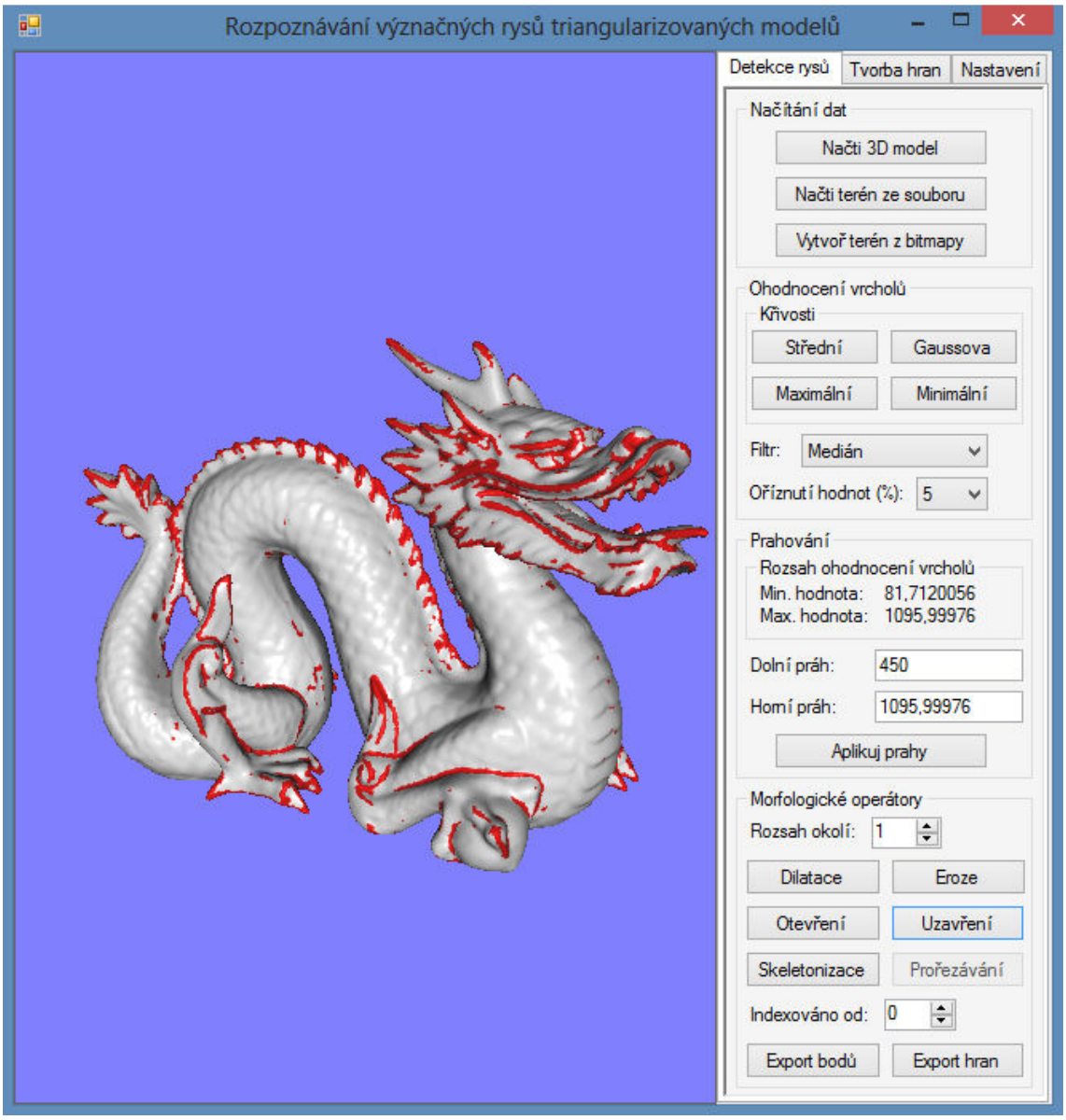
Po načtení modelu lze jednoduše detekovat význačné hrany pomocí navržené metody. Všechny potřebné prvky se nachází v záložce *Detekce rysů*. Nejprve je nutné vypočítat ohodnocení všech vrcholů trojúhelníkové sítě. To lze udělat kliknutím na některé z tlačítek nacházejících se ve skupině *Křivosti* (tj. tlačítko s názvem *Střední*, *Maximální*, *Gaussova* nebo *Minimální*). Navíc je možné nastavit filtr nebo oříznutí nejvyšších hodnot (nutné zadat před výpočtem ohodnocení). Filtr podle vybrané metody vypočítá novou váhu vrcholu na základě vah jeho sousedů. Oříznutí se používá k odstranění (resp. ke snížení) odlehlých hodnot. Podle zvoleného čísla se vybere daný počet procent vrcholů s nejvyšší hodnotou a nastaví se jim nová váha odpovídající ohodnocení vrcholu, který má nejvyšší hodnotu bez již vybraných vrcholů.

Po výpočtu ohodnocení se do textového pole *Dolní práh* a *Horní práh* doplní maximální a minimální váha (zároveň se také doplní k popisu *Min. hodnota* a *Max. hodnota*, neboť zde je nelze přepsat). Změnou těchto hodnot nastavujeme meze, které jsou využívány při prahování. Příklad uživatelského rozhraní po výpočtu ohodnocení vrcholů a nastavení prahových hodnot je zobrazen na Obr. B.2.



Obr. B.2: Uživatelské rozhraní aplikace po výpočtu ohodnocení vrcholů. Model převzat z [39].

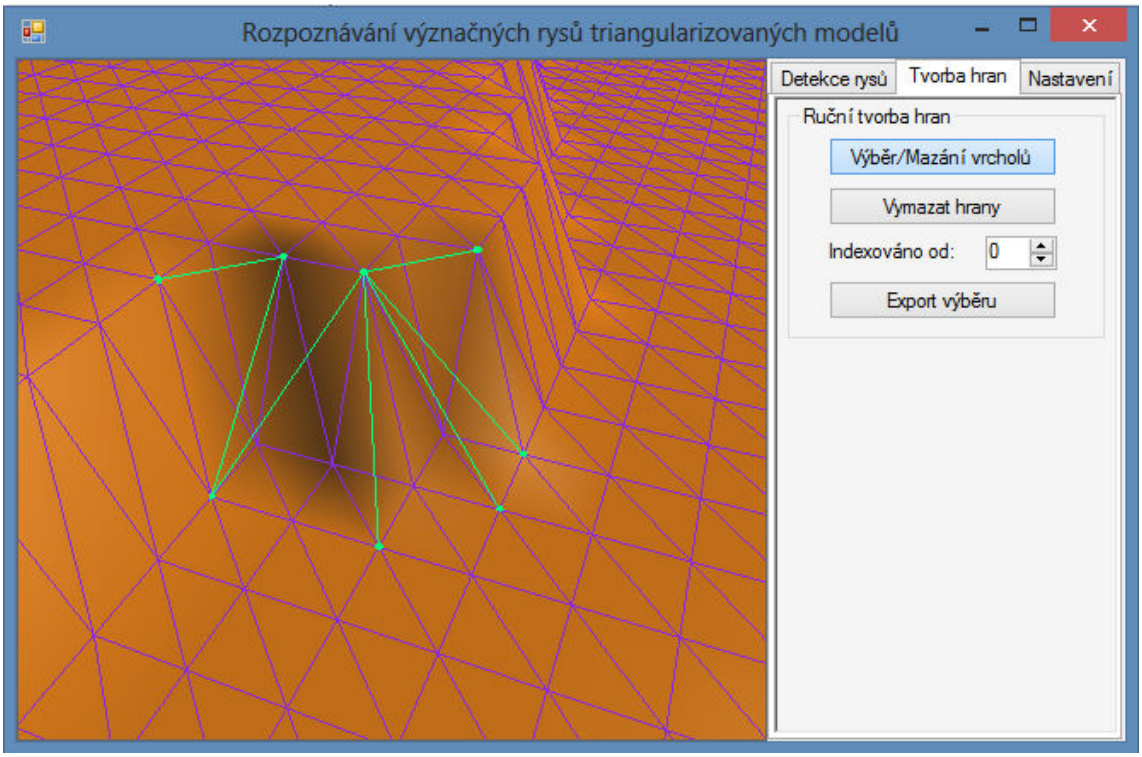
Prahování podle aktuálně nastavených mezí lze provést stisknutím tlačítka *Aplikuj prahy*. Tím dojde k vytvoření význačné oblasti a odblokování tlačítek s morfologickými operacemi (kromě operace prořezávání), které lze aplikovat v různém pořadí. Pro operátor dilatace, eroze, otevření, uzavření a prořezávání je možné nastavit velikost *N*-sousedství pomocí číselníku *Rozsah okolí*. Operace prořezávání je povolena až po aplikování skeletonizace. Jestliže nám aktuální stav význačné oblasti vyhovuje, můžeme použít tlačítko *Export bodů* nebo *Export hran*. Tím dojde k vyexportování význačné oblasti do ***.txt*** souboru (ukládají se indexy vrcholů). Pokud potřebujeme, aby indexy začínaly od jiné hodnoty než 0, můžeme nastavit hodnotu v číselníku *Indexováno od*. Při exportování bodů je na každé řádce uložen jeden index reprezentující význačný vrchol (na první řádce je celkový počet vrcholů). V případě hran jsou na každé řádce indexy dva. Ukázka uživatelského rozhraní po aplikování prahování a použití morfologického operátoru uzavření je vidět na Obr. B.3.



Obr. B.3: Uživatelské rozhraní aplikace po použití prahování a operátoru uzavření. Model převzat z [39].

Ruční tvorba hran

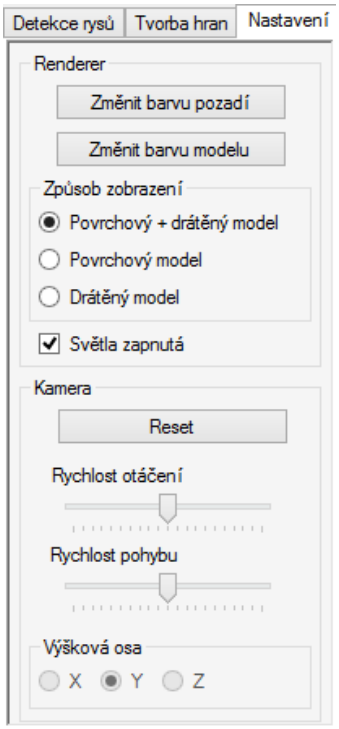
Jak již bylo řečeno, aplikace také umožňuje vytváření význačných hran ručně. Pro tento účel slouží záložka *Tvorba hran*. Nejprve je nutné zakliknout tlačítko *Výběr/Mazání vrcholů*. Tím se aktivuje ruční výběr. Vrchol je možné označit kliknutím levého tlačítka myši se stisknutým *Ctrl*. Hrana se vytvoří vždy po výběru dvou vrcholů. Vymazat vybraný vrchol lze kliknutím pravého tlačítka myši se stisknutým *Ctrl*. Tím dojde k odstranění všech hran, které tento vrchol využívaly. Pomocí tlačítka *Vymazat hrany* lze odstranit všechny hrany najednou. Jestliže jsme s tvorbou hotovi, můžeme vytvořené hrany vyexportovat kliknutím na tlačítko *Export výběru*. Formát výstupního souboru je stejný jako u detekce hran (opět je možné nastavit jiné indexování pomocí číselníku *Indexováno od*). Příklad uživatelského rozhraní lze vidět na Obr. B.4.



Obr. B.4: Ukázka ruční tvorby hran (zelené čáry).

Změna nastavení

Nastavení aplikace lze měnit v záložce *Nastavení*. Význam jednotlivých ovládacích prvků je vcelku zřejmý (viz Obr. B.5).



Obr. B.5: Uživatelské rozhraní nastavení aplikace.

C Obsah přiloženého DVD

Přiložené *DVD* má následující strukturu:

- ***Readme.txt*** – popisuje obsah jednotlivých složek.
- ***Dokument*** – složka obsahující text diplomové práce v *.pdf* a *.docx* formátu.
- ***Aplikace*** – obsahuje vytvořený program.
    - ***bin*** – spustitelná verze programu (*.exe*). Před spuštěním aplikace je potřeba nainstalovat běhové prostředí knihovny *SlimDX* (viz složka *Knihovny/user runtime*).
    - ***src*** – zdrojové kódy vytvořené aplikace.
    - ***VS2010*** – *Visual Studio 2010* projekt (*.sln*). Slouží pro jednoduché ověření funkčnosti zdrojových kódů, pokud je nainstalováno *Visual Studio* a *SlimDX Developer SDK*.
- ***Data*** – příklady vstupních dat.
    - ***3D modely*** – složka obsahující trojrozměrné modely.
    - ***terény*** – obsahuje terénní data.
- ***Knihovny*** – zahrnuje instalační soubory pomocné knihovny, kterou vytvořená aplikace používá (*SlimDX*).
    - ***user runtime*** – obsahuje instalační soubor pro uživatele aplikace.
    - ***developer SDK*** – instalační soubor pro vývojáře. Je nutné nainstalovat před kompilací zdrojových kódů.
- ***Ukázky výstupu*** – obsahuje obrázky reprezentující výstup navržené metody.