VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

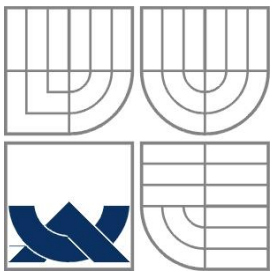
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUER GRAPHICS AND MULTIMEDIA

KLASIFIKACE ZUBŮ NA 3D POLYGONÁLNÍM MODELU ČELISTI

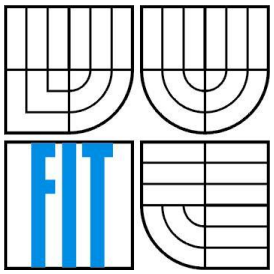
DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE Bc. Rostislav Hulík
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUER GRAPHICS AND MULTIMEDIA

KLASIFIKACE ZUBŮ NA 3D POLYGONÁLNÍM MODELU ČELISTI

TOOTH CLASSIFICATION ON JAW 3D POLYGONAL MODEL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Rostislav Hulík

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. Přemysl Kršek, Ph.D.

BRNO 2010

Abstrakt

Tato práce se zabývá návrhem a implementací řešení klasifikace zubů na polygonálním modelu otisku čelisti. Postupně v ní popisuji možnosti procházení a reprezentace polygonálního modelu v paměti počítače, detekci roviny prokládající křivku zubního oblouku a proložení plochy reprezentující přibližnou zubní rovinu. Dále se zabývám možností generování výškové mapy usnadňující zařazení jednotlivých zubů do širšího kontextu (zubního oblouku) a detekcí těchto zubů na úrovni výškové mapy. Současně nabízím možnosti segmentace trojrozměrného modelu tak, abych z něho byl schopen extrahovat hranice vymezující zuby. V závěru navrženého algoritmu spojuji tyto dva postupy ve výsledný způsob detekce a klasifikace jednotlivých zubů tak, že aplikace bude schopna automaticky rozpoznávat a zařazovat zuby ke korespondujícím jménům s minimálním zásahem uživatele. V druhé části tohoto dokumentu se nachází popis implementovaného řešení. Podle zadání navrhuji výše zmíněné postupy s důrazem na multiplatformní chování a maximální pohodlí uživatele.

Klíčová slova

klasifikace, detekce, zuby, čelist, polygonální, model, 3D, otisk, výšková mapa, křivost, RANSAC, rozvodí, watershed, fuzzy, MDSTk, OSG, wxWidgets

Abstract

This document discusses a solution for tooth classification on 3D jaw polygonal model. Sequentially, I describe techniques for representation and browsing of polygonal model saved in computer memory, techniques for dental curve detection and finally, creation of surface representing approximated tooth plane. After it, I analyze possibilities of height map creation from jaw model which helps in tooth classification in the scope of entire dental curve context and, as a last step, final detection of these teeth in two dimensions. In the same time, I discuss 3D polygonal model segmentation for border extraction, which separates teeth from the rest of the model. In the end of proposed algorithm, I join these two runs into one final detection and classification process of separate teeth, so presented application can automatically indentify and classify teeth to corresponding names and positions with a minimum user interaction. In a second half of this document, I describe implemented solution. According to primary goal, I propose these techniques forcefully to multiplatform approach and maximal user comfort.

Keywords

classification, detection, jaw, tooth, teeth, polygonal, model, 3D, impression, height map, curvature, RANSAC, watersheds, fuzzy, MDSTk, OSG, wxWidgets

Citace

Hulík, Rostislav, Bc.: Klasifikace zubů na 3D polygonálním modelu čelisti. Brno, 2010, diplomová práce, FIT VUT v Brně.

Klasifikace zubů na 3D polygonálním modelu čelisti

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Ing. Přemysla Krška Ph.D. a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Rostislav Hulík
10.5.2010

Poděkování

Chtěl bych poděkovat mému vedoucímu Doc. Ing. Přemyslu Kršskovi, Ph.D. za jeho výborné vedení, připomínky, ochotu řešit se mnou problémy, jež vyvstaly v souvislosti s mým časovým rozvrhem během mého magisterského studia. V důsledku toho jsem mohl tuto diplomovou práci řádně splnit. Děkuji též společnosti 3Dim Laboratory [18] za možnost řešit tento problém a za jimi zapůjčená testovací data a knihovny. V neposlední řadě bych chtěl poděkovat své přítelkyni a rodině za trpělivost a podporu po dobu celého mého studia i během tvorby této diplomové práce.

Obsah

Obsah 1
1 Úvod 3
2 Teoretický rozbor 4
2.1 Reprezentace 3D modelu 4
2.1.1 CSG – Constructive Solid Geometry 4
2.1.2 Dekompoziční modelování 5
2.1.3 Implicitní plochy 5
2.1.4 Hraniční reprezentace 6
2.2 Extrakce příznaků z polygonálního modelu 7
2.2.1 Křivost v daném směru 7
2.3 Tvorba výškové mapy 8
2.3.1 Rasterizace trojúhelníku 9
2.4 Segmentace obrazu/tělesa 10
2.4.1 Segmentace metodou rozvodí 11
2.5 Fuzzy logika 12
2.6 Rozpoznávání, aproximace tvarů 13
2.6.1 RANSAC 14
2.7 Proložení roviny mračnem bodů 14
3 Návrh řešení 17
3.1 Celkový návrh aplikace 17
3.1.1 Navrhované externí knihovny 17
3.2 Návrh algoritmu detekce 18
3.2.1 Aproximace plochy 20
3.2.2 Tvorba výškové mapy 20
3.2.3 Detekce zubní křivky 21
3.2.4 Detekce a klasifikace zubů na výškové mapě 22
3.2.5 Označení regionů na výsledném trojrozměrném modelu 23
3.2.6 Manuální korekce 24
3.2.7 Oddělení jednotlivých modelů 24
3.2.8 Úpravy pozic zubů 24
4 Implementace 25
4.1 Celková koncepce aplikace 25
4.2 Modul Application 26
4.3 Modul OSG 26

4.4 Modul VCTL 27
4.5 Modul Other 28
4.6 Modul Analysis2D 28
4.6.1 Hledání zubní křivky 29
4.6.2 Detekce a klasifikace zubů 30
4.7 Modul Analysis3D 34
4.7.1 Hlavní obsluha aplikace 34
4.7.2 Proložení roviny modelem 36
4.7.3 Křivosti a segmentace 36
4.7.4 Spojení analýzy výškové mapy a modelu 36
4.8 Shrnutí konceptu aplikace 37
5 Shrnutí výsledků 38
5.1 Aproximace plochy a průmět výškové mapy 38
5.1.1 Možné budoucí úpravy 39
5.2 Detekce zubní křivky 39
5.2.1 Možné budoucí úpravy 42
5.3 Křivosti a segmentace 42
5.3.1 Možné budoucí úpravy 43
5.4 Klasifikace zubů 44
5.4.1 Možné budoucí úpravy 46
5.5 Celkový koncept klasifikace zubů 46
5.5.1 Možné budoucí úpravy 48
6 Závěr 49
Literatura 50
Seznam příloh 52

1 Úvod

V této práci se zabývám problematikou klasifikace zubů na 3D počítačovém modelu čelisti. Vstupem celého algoritmu je model otisku zubů v hraniční reprezentaci (trojúhelníkový model) naskenovaný a oříznutý ihned po sejmutí otisku zubní čelisti, případně naskenovaný sádrový odlitek tohoto otisku. Ideálním výstupem by měl být model čelisti s oddělenými a klasifikovanými zuby. To vše by mělo být přístupné ve vhodně navržené aplikaci, jejíž využití se předpokládá ve stomatologii, kde by měla výrazně urychlit zpracování modelů čelistí, usnadnit celkovou orientaci a možnost plánování korekcí zubních oblouků.

V první kapitole seznámím čtenáře s předpokládaným postupem řešení. Poté rozeberu podrobněji problematiku reprezentace a práce s 3D modely, styl jejich uložení v paměti apod. Následně prozkoumám možnosti extrakce příznaků tvarů z polygonálního modelu, konkrétně pomocí výpočtu křivosti. Tento způsob bych chtěl ve výsledné aplikaci kombinovat s analýzou výškové mapy vhodně vytvořené z modelu, popíši tedy i možnosti segmentace dvourozměrného obrázku a případnou extrakci příznaků.

Druhá kapitola dále pojednává o mém návrhu řešení s využitím konkrétních aplikací metod zmíněných výše. Popisuji zde vhodné proložení roviny modelem, následné generování výškové mapy a extrakci jejích příznaků. Současně blíže specifikuji segmentaci samotného polygonálního modelu, kterou provádím využitím výpočtů křivosti a aplikací vhodného segmentačního algoritmu. Nakonec zmíním možnosti propojení výše zmíněných postupů ve výsledný algoritmus, kombinující právě segmentaci ve 3D s analýzou dvourozměrného průmětu.

Následující kapitola hovoří o výsledné implementaci aplikace. Popisuje konkrétní dekompozice problému, umístění ve třídách a modulech. Též se zajímá o výsledný koncept aplikace pro snadnější orientaci v kódu. Nejedná se však o programovou dokumentaci, nýbrž o vysvětlení implementace problému. Programová dokumentace se nachází v příloze na CD v elektronické podobě.

Kapitola Shrnutí výsledků poté srovnává dosažené výsledky segmentace pomocí této metody. Jsou zde doplněny výstupní obrázky i vizualizace z průběhu samotných výpočtů během klasifikace. Popisuji též klady a zápory navrženého algoritmu. Jako další bod této části zmíním možné trendy v budoucím výzkumu tohoto tématu, který bude jistě nutný, jelikož se jedná o problematiku značně rozsáhlou a hlavně časově náročnou na testování a návrh jednotlivých postupů.

Na závěr zhodnotím práci na diplomovém projektu, podrobněji rozeberu, na kolik bylo splněno zadání a uvážím možná budoucí vylepšení a případné změny, které by byly vhodné provést pro další zpřesnění výpočtů.

2 Teoretický rozbor

Tato kapitola popisuje jednotlivé techniky využité v mém návrhu postupu řešení po teoretické stránce. V tom je potřeba vyřešit nejprve samotné uložení a reprezentaci 3D modelu, jeho segmentaci, v mém postupu prováděnou za pomoci výpočtů křivosti, a aplikaci vhodného segmentačního nástroje. Současně zmíním možnosti segmentace dvourozměrného obrázku, to vše za předpokladu, že bude možné model vhodně promítnout do roviny, kde se pokusím zuby vyhledat. Pro samotnou klasifikaci zde též představím metodu vhodnou pro registraci známého tvaru na obrazu (výškové mapě) a možnosti klasifikace zubů.

2.1 Reprezentace 3D modelu

Vstupní model je možné získat vícero způsoby. Tím nejčastějším (a předpokládaným) by mělo být v mém případě naskenování reálného modelu a přímá reprezentace polygonální sítí. Samozřejmě existují i jiné metody, např. získání hraniční reprezentace převodem z volumetrických dat apod.

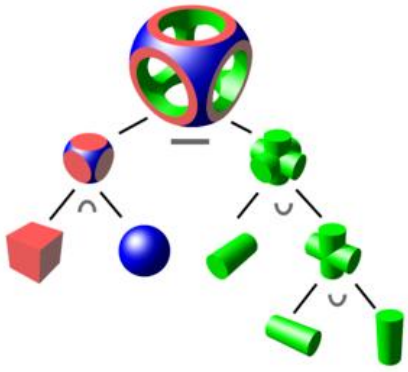
Valná většina dnešních grafických výstupních zařízení je však dvourozměrná - tedy obsahuje dvourozměrnou matici pixelů reprezentující zobrazenou scénu. Modely je tedy nutné pro zobrazení převézt do dvourozměrného rastru, který je následně vykreslen. Zde však ztrácíme příliš mnoho informací, je tedy třeba pracovat se samotnou trojrozměrnou reprezentací. V první podkapitole nejprve rozeberu jednotlivé možnosti interního uložení 3D objektu. Těch je několik, proto zde uvedu stručný popis alespoň těch nejpoužívanějších[1]:

2.1.1 CSG – Constructive Solid Geometry

Konstruktivní geometrie reprezentuje těleso jako soubor grafických primitiv (kvádr, jehlan, čtyřstěn aj.) s definovanými booleovskými operacemi mezi nimi. Celý model je uložen ve stromu (nejčastěji binárním) s jednotlivými primitivy v listech a operacemi v uzlech.

Hlavními výhodami tohoto způsobu je zajisté velice intuitivní přístup ke konstrukci modelů a možnosti uložení jisté informace o materiálu primitiv (jednotlivá primitiva jsou samozřejmě homogenní, výsledný model se však může skládat z více materiálů). Hlavní nevýhodou je složitější režie zobrazení výsledného tělesa, kdy je nutné pro každý list stromu počítat výsledný tvar apod. Další nevýhodou je pak samozřejmě nemožnost lokálních změn, tedy změní-li se jakýkoli objekt ve stromu, výsledný model se celý přepočítává. To se v současnosti řeší dalším dělením CSG stromu pro reprezentaci částí, které se vzájemně neovlivňují.

Tento přístup se využívá v CAD/CAM aplikacích. Mezi nevýhody patří jistě obtížnější převod do trojúhelníkové reprezentace výsledného tělesa, i tyto algoritmy jsou však dnes běžné.



Obr. 2.1.1.1 CSG binární strom[2]

2.1.2 Dekompoziční modelování

Dalším typem reprezentace trojrozměrného tělesa je dekompoziční model. Těleso je navzorkováno v trojrozměné matici, kde jedna hodnota udává určitou vlastnost místa v objektu (barva, hustota...). Tyto základní stavební jednotky se nazývají voxely (z angl. volume element). Je tedy zřejmé, že výsledek neukládá pouze hranice objektu, ale celý jeho objem. Zde však zůstává problém zobrazení neprůhledných částí, ale to není předmětem této práce (tzv. problematika volume renderingu).

S tímto typem dat se setkáváme např. v medicíně, kdy výstup z CT/MR skenů převádíme právě do volumetrických dat, kde hodnota voxelu udává Roentgenovou hustotu vyjádřenou v tzv. Hounsfieldově jednotce[3]. Toto zmiňuji z důvodu, že v praxi může nastat situace, kdy polygonální model reprezentující lidskou čelist získáme právě převodem z volumetrických CT/MR dat.

Zmiňovaný převod do polygonálního modelu je popsán metodou Marching Cubes, která se využívá pro převod izoplochy z objemových dat do polygonální sítě. Izoplochou v tomto kontextu myslím plochu se společnými hodnotami voxelů. Tímto způsobem lze např. z volumetrických dat obsahujících CT scan obličeje extrahovat právě zubní oblouk (izoplochu stejné hustoty – zuby).

2.1.3 Implicitní plochy

Popis modelu pomocí implicitních ploch se skládá z kostry a potenciálního pole okolo každého jejího bodu definovaného určitou funkcí. To následně odděluje prostor uvnitř a vně tělesa. Tento postup je hojně využíván v grafice, např. při implementaci ray-tracingu, případně při řešení kolizí těles, kdy je tento postup velice efektivní. Pro každý model totiž existuje funkce, podle níž lze jednoduše zjistit, je-li bod uvnitř/vně tělesa.

Zobrazit tyto modely je však možné buď převedením do hraniční reprezentace, případně pomocí ray-tracingu scény. Převod do B-rep je tedy možný.

2.1.4 Hraniční reprezentace

Hraniční reprezentace (boundary representation, B-rep) je pravděpodobně nejčastějším způsobem zobrazení modelu. Dále tento způsob můžeme dělit na reprezentaci pomocí mnohoúhelníků nebo surfelů.

2.1.4.1 Surfel model

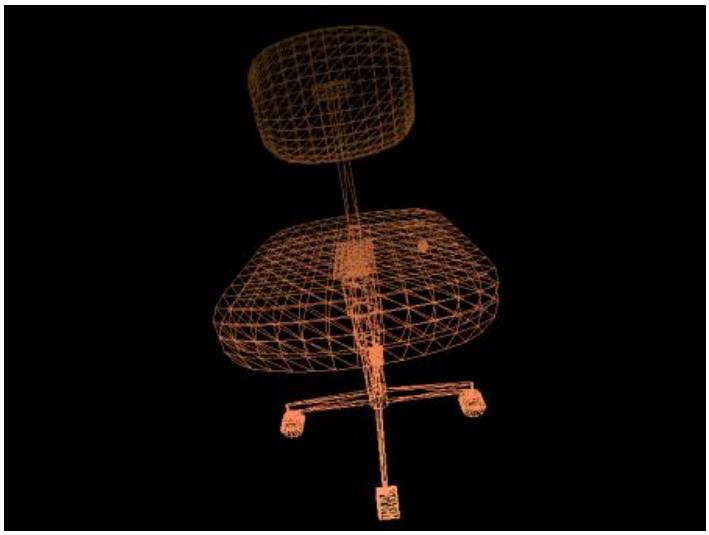
Hraniční reprezentace modelu pomocí surfelů je speciálním případem B-rep. Surfel (z anglického surface element) je bod s informací o jeho normále a barvě. Takto lze bezpečně počítat stínování a jiné operace závislé na normálovém vektoru. Nevýhodou je samozřejmě nespojitý povrch, který může způsobovat problémy jak při zobrazení, tak hlavně při operacích s modelem, neboť body nemají žádnou informaci o vzájemné poloze stejně jako nevyplňují celý povrch spojitě. Operace jako průsečík s tělesem mohou být tedy velice nejasné.

2.1.4.2 Polygonální model

Polygonálními modely se zabývám ve své práci, proto následující kapitoly budou pojednávat o operacích pouze s tímto typem zobrazení objektů. Ty jsou složeny z mračna bodů (vrcholů) náležících mnohoúhelníkům, kterými je reprezentován povrch tělesa. To přináší mnoho výhod, jednou z nich je snadný výpočet průsečíků s tělesem, stínování apod. Navíc uložená struktura (zmíněná dále) obsahuje informace o sousedech, a to jak mnohoúhelníků samotných, tak hran i vrcholů. To usnadňuje orientaci na modelu. Výhodu skýtá i způsob reakce na změny modelu, kde oproti všem třem výše zmíněným je nutné obnovit pouze komponenty, kterých se změna přímo týká (lokální změny).

Nejčastěji využívaným polygonálním modelem je trojúhelníkový model, složený pouze s nejjednodušších mnohoúhelníků Eukleidovského prostoru – trojúhelníků. Pomocí trojúhelníků lze velice snadno skládat libovolné polygony, navíc trojúhelník vždy leží v jedné rovině. Je tedy možné mít pouze jeden normálový vektor pro jeden trojúhelník, a tedy výpočty průsečíků a jiných operací se dále zjednodušují.

Obr. 2.1.4.2.1 – Hraniční trojúhelníková reprezentace



2.2 Extrakce příznaků z polygonálního modelu

Nyní představím podrobněji možnosti zjišťování informací o vlastnostech hraniční reprezentace polygonálního modelu. Jak jsem již zmínil v předchozí kapitole, struktura je vzájemně velice dobře propojená, neobsahuje však žádné informace vyjadřující určitou část modelu ve větším kontextu. Každý vrchol/hrana/stěna tedy zná své sousedy, mým úkolem je ale rozpoznávání tvarů na modelu. To je jednoduše řečeno závislé na různých nerovnostech povrchu, respektive na tvaru povrchu samotného.

Jednou z možností, jak vyjádřit informace o tvaru povrchu je výpočet křivosti [6]:

2.2.1 Křivost v daném směru

Základní myšlenka normálové křivosti je jednoduchá[7]. Mějme plochu **S** a bod **p** náležící S. Poloměr normálové křivosti je poté definován následovně:

Mějme vektor **v** náležící do množiny tečných vektorů plochy **S** v bodě **p.** Pak průnik plochy S a roviny dané vektory **v** a **N** se nazývá **normálová sekce** plochy S v bodě p kolem v. Pokud do normálové sekce vepíšeme kružnici tak, aby Směla s plochou S společný pouze jeden bod a zároveň měla maximální možnou velikost, její poloměr bude reprezentovat právě poloměr křivosti **r** v daném směru.

Křivost v daném směru lze v daném bodě vyjádřit též pomocí vztahu:

$$r = \frac{Edu^2+2Fdudv+Gdv^2}{|Ldu^2+2Mdudv+Ndv^2|}\cos(\Theta) \quad (1),$$

kde θ je úhel, který svírá rovina křivky s normálou v bodě *P* plochy, *E*, *F*, *G* jsou základní veličiny plochy prvního řádu a *L*,*M*,*N* jsou základní veličiny plochy druhého řádu (viz např. [8]). Pro normálový řez bude tedy cos(θ) roven jedné.

Hodnota křivosti je poté definována jako převrácená hodnota poloměru křivosti, tedy:

$$k = \frac{1}{r} \quad (2).$$

2.2.1.1 Hlavní křivosti

Pro každý bod $p \in S$ existuje ortonormální báze $\{e_1, e_2\}$ z množiny všech tečných vektorů taková, že $dN_p(e_1) = -k_1 e_1$ a $dN_p(e_2) = -k_2 e_2$. Pro moji práci je důležité, že Shodnoty k_1 a k_2 reprezentují v daných směrech extrémy ze všech možných křivostí pro bod p.

Tyto směry v tzv. hlavních normálových řezech plochy v bodě p označujeme jako hlavní směry plochy a jejich příslušné křivosti jako **hlavní křivosti**, konkrétně maximální a minimální křivost k_1 a k_2 pro bod p.

Ve vztahu s (1) lze hlavní poloměry křivosti R řešit rovnicí:

$$(EG - F^2)\frac{1}{R} - (EN - 2FM + GL)\frac{1}{R} + (LN - M^2) = 0 \quad (3).$$

2.2.1.2 Gaussova a střední křivost

Pro vyjádření vlastnosti povrchu je zajisté výhodou vyjádřit daný bod pouze jedním číslem. Existuje větší množství popisů křivostí, zde zmíním pouze dva z nich – Gaussovu a střední křivost.

Střední křivost v získáme v podstatě průměrováním hlavních křivostí, tedy:

$$H = \frac{1}{2}(k_1 + k_2) \quad (5).$$

Plocha, která má střední křivost nulovou ve všech bodech, se nazývá minimální.

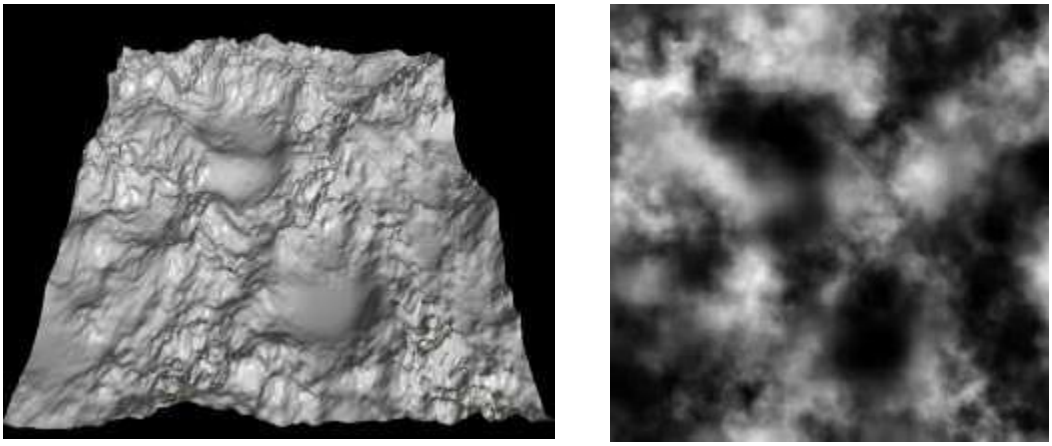
Gaussova křivost se vyjadřuje z diferenční Gaussovy mapy pro danou plochu v daném bodě, kde Gaussova křivost je definována jako determinant této matice. Pokud má plocha ve všech bodech Gaussovu křivost nulovou, nazývá se plochou rozvinutou. V kontextu minimální a maximální křivosti lze tu Gaussovu vyjádřit jako součin dvou výše zmíněných.

$$G = k_1 k_2 \quad (6)$$

2.3 Tvorba výškové mapy

Pro mou práci bude též důležité mít možnost vykreslit si výškovou mapu pro danou plochu. Výšková mapa znamená v podstatě průmět daného tělesa na plochu se zachováním informace o vzdálenosti konkrétního bodu ve formě výškové informace. Jednoduše řečeno výšková mapa je stejně jako mapa geografická pohled na objekt z dané roviny s různými hodnotami pixelů náležícím výšce průniku s objektem.

Na následujícím obrázku je možné vidět vpravo výškovou mapu, nalevo poté korespondující model[9]. Mapa byla vytvořena průmětem na plochu z = 0;



Obr. 2.4.1 – Model a výšková mapa

Pro samotné vytvoření výškové mapy z již stávajícího modelu je nejsnadnější promítnout daný model do souřadnic plochy a poté ho rasterizovat. Podrobněji zde popíši právě rasterizaci trojúhelníku.

2.3.1 Rasterizace trojúhelníku

Nejprve každý trojúhelník modelu transformuji podle průmětné roviny. Úprava je nutná pro urychlení výpočtu tak, aby osa **z** udávala vzdálenost každého bodu objektu od průmětny. Změna způsobí, že jednoduchým zanedbáním souřadnice **z** původního transformovaného objektu vyjádřím souřadnice trojúhelníku promítnutého do dvourozměrné roviny.

Samotná rasterizace probíhá podle následujícího schématu[10]:

- Mějme dvourozměrný trojúhelník definovaný body $p_1, p_2, p_3 \in R^2$ a nechť $x \in R^2$ je libovolný bod v prostoru. Je nutné najít barycentrické souřadnice takové, že:

$$x = \alpha p_1 + \beta p_2 + \gamma p_3 \quad (7) \qquad \text{kde} \qquad \alpha + \beta + \gamma = 1.$$

  Barycentrické souřadnice jsou v podstatě vyjádřením neortogonálního bazického systému pro rovinu. Pro každý bod x náležící trojúhelníku platí $\alpha, \beta, \gamma \in [0,1]$.

- Samotná teorie rasterizace vychází právě z předpokladu parametů barycentrických souřadnic. Každý příchozí trojúhelník nejprve normalizujeme tak, aby jeho minimální obvodový

obdélník byl v rozmezí [-1, 1] x [-1, 1]. Následně pro každý pixel tohoto obvodového obdélníku spočítáme jeho korespondující barycentrické souřadnice:

Nechť x, y jsou souřadnice testovaného bodu, pak položíme:

$$f_{ab}(x,y) = (y_a - y_b)x + (x_b - x_b)y + x_a y_b - x_b y_a. \quad (8)$$

Pro konkrétní body trojúhelníku jsou barycentrické souřadnice bodu vyjádřeny jako

$$\alpha = \frac{f_{12}(x,y)}{f_{12}(x_0,y_0)}, \quad (9)$$

$$\beta = \frac{f_{20}(x,y)}{f_{20}(x_1,y_1)}, \quad (10)$$

$$\gamma = \frac{f_{01}(x,y)}{f_{01}(x_2,y_2)}, \quad (11)$$

kde indexy {0, 1, 2} značí vrcholy trojúhelníku. Pokud hodnoty leží v intervalu $\alpha, \beta, \gamma \in [0,1]$, bod je součástí trojúhelníku a stačí pro něj vypočítat hodnotu z vzdálenosti od průmětné plochy.

- Pro spočítání konkrétní hodnoty vzdálenosti bodu na trojúhelníku od průmětné plochy stačí vhodně interpolovat hodnoty jeho vrcholů. K tomu lze též jednoduše použít barycentrické souřadnice

$$z = \alpha z_0 + \beta z_1 + \gamma z_2, \quad (12)$$

kde indexy {0,1,2} značí indexy vrcholů a jejich souřadnici z. Vztah má ale i více použití, pokud do něho dosadíme místo skalární hodnoty vektory (například normály ve vrcholech), interpoluje též tyto vektory do výsledné hodnoty v bodě x. Tato informace může být užitečná například při interpolaci normál pro jeden ze složitějších modelů stínování.

2.4 Segmentace obrazu/tělesa

Pokud již existuje objekt s extrahovanými určitými příznaky, např.:

- polygonální model se spočítanými křivostmi pro každý vrchol,
- gradientní obraz výškové mapy

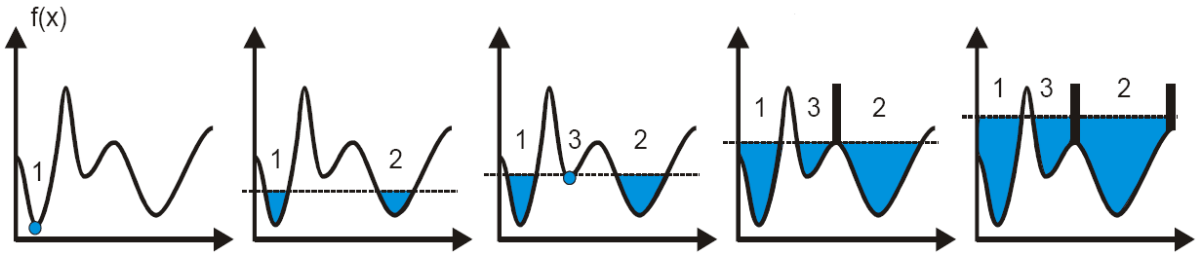
je nutné elementy těchto těles zařadit do vyššího kontextu. V obou případech se jedná o segmentaci obrazu, a jak jsem si již ověřil v mé bakalářské práci [6], na tento typ dat má velice příznivou odezvu metoda Watershed. V následujícím popisu budu metodu popisovat na dvourozměrném obrázku, v kontextu trojrozměrného modelu je její použití shodné. Pro každý

vrchol modelu lze též vyjádřit jeho sousedy, jako hodnota se bere spočítaná křivost. Algoritmus je tedy shodný:

2.4.1 Segmentace metodou rozvodí

Watersheds je morfologická metoda segmentace, kdy je na obraz pohlíženo jako na topografický terén, který postupně zaplavujeme vodou. Pro vstup do algoritmu se nepoužívá originální obrázek, nýbrž gradientní obraz vypočtený z originálu. V mém případě mapa křivosti však již ze své podstaty reprezentuje gradientní obraz, naopak, výškovou mapu bude třeba nejprve do gradientního obrazu převézt.

Samotný algoritmus poté funguje na principu zaplavování, kde postupně zvedáme hladinu vody a místa, kde se slévají dva zdroje („nádrže“), označujeme jako hráze. Zdroje vody v původní implementaci algoritmu vznikají všude, kam dosahuje aktuální hladina vody. Po zařazení každého elementu ze segmentovaného obrazu do jednoho z povodí algoritmus končí. Postup je lépe čitelný na následujícím obrázku[11]:



Obr. 2.5.1.1 – Watersheds algoritmus

Zde se ale dostávám k hlavní nevýhodě segmentace pomocí rozvodí. Pokud je totiž vstupní obraz příliš zašuměný, dochází k velkému přesegmentování. To lze řešit vícero způsoby[12]:

- Určitým způsobem oddálit tvoření nových segmentů.
- Zaplavovat území pouze z explicitně definovaných zdrojů, čili automaticky nezakládat nové zdroje.
- Využít jeden ze způsobů region-merging, čili spojování již segmentovaného objektu.

2.4.1.1 Oddálení tvorby nových segmentů

Tento způsob jsem použil u své bakalářské práce a úspěšně snížil počet nově vzniklých segmentů. Jedná se o počáteční naplnění nejnižších hodnot, které následně algoritmus rozšiřuje. Není však možné, aby dané rozvodí „přeteklo“ do sousedního údolí, pouze zvyšuje do maxima stávající segmenty. Jakmile se v obraze přestanou provádět změny, vyberu z obrázku několik nových pramenů a pokračuji, dokud nezaplním celý obraz.

2.4.1.2 Zaplavování z explicitně definovaných zdrojů

Tento způsob řeší otázku přesegmentování úplně. Nová rozvodí nevznikají vůbec, pouze se plní ty stávající s tím, že pokud se narazí na nezaplněné údolí, je vyplněno nejbližším zaplněným rozvodím. Tento přístup je poté velice přesný na detekci nejsilnějších hranic, bohužel je však nutné nějak inicializovat počáteční rozvodí. To lze provézt v některých aplikacích manuálně, v mém případě však bude třeba zajistit způsob zjištění automatického.

2.4.1.3 Spojování oblastí

Posledním způsobem, který lze aplikovat, je přesegmentované oblasti spojit podle určitého kritéria. V praxi se používají různé metody počínající podobností průměrných hodnot regionů, podobností středních odchylek až po geometrické podobnosti apod.

2.5 Fuzzy logika

Předpokládám, že pro korektní vyřešení klasifikace jednotlivých zubů, které nemusí být umístěny v přesných intervalech na zubní křivce (chybějící, posunuté zuby) bude nutné využít logiku umožňující vyjádřit pouze částečnou příslušnost prvku k množině. Nejřesněji tento přístup popisují fuzzy množiny a fuzzy logika[16].

Jak jsem již zmínil, jedná se o zobecnění standartní teorie množin, které je schopno vyjádřit i částečnou příslušnost prvku k množině. Příslušnost je vyjádřena číslem [0, 1] udávající pro každý prvek hodnotu jeho příslušnosti k dané množině. Pro jednoduchost výpočtů se ve spojitém prostoru vyjadřuje funkcí příslušnosti, která je z důvodu snadnosti výpočtu nejčastěji gaussovská nebo trojúhelníková. Samozřejmě funkce mohou být definovány i diskrétně. Navíc jednotlivé množiny se mohou překrývat, tedy prvek může patřit současně do více množin najednou.

Pro fuzzy množiny platí stejné predikátové symboly jako pro ostré množiny, čili můžeme definovat:

- rovnost – $M_1 = M_2 <=> \forall x\big(m_1(x) = m_2(x)\big)$,
- inkluzi – $M_1 \subseteq M_2 <=> \forall x\big(m_1(x) \leq m_2(x)\big)$,
- support – $support(x) = \{x | m(x) > 0\}$.

Definovat můžeme i funkce známé z ostrých množin:

- sjednocení – $M_{m_1 \cup m_2} = \forall x\big(\max\big(m_1(x), m_2(x)\big)\big)$,
- průnik – $M_{m_1 \cap m_2} = \forall x\big(\min\big(m_1(x), m_2(x)\big)\big)$,
- doplněk – $M' = \forall x\big(1 - m(x)\big)$.

Následně je možné definovat vlastnosti množin, které se od klasických ostrých množin v detailech liší. Fuzzy množiny tedy splňují tyto vlastnosti:

- komutativitu,
- asociativitu,
- distributivitu,
- deMorganovy zákony,
- dvojí doplněk,
- idempotenci,
- identitu.

Na rozdíl od ostrých množin však nesplňují následující pravidla:

- zákon protikladu ($A \cup A' \neq 1$),
- zákon vyloučení třetího ($A \cap A' \neq \phi$).

Pro samotné rozhodování bude třeba v mém případě pouze vyřešit, do které množiny daný prvek patří. Proto z původního návrhu zahrnujícího fuzzifikaci, fuzzy inferenci a defuzzifikaci bude pro mé rozhodování nejdůležitější fuzzifikace.

V této části se vstupní hodnoty (pravděpodobně informace o pozici) převedou do fuzzy hodnot, jinak řečeno, pro každou hodnotu se spočítají příslušnosti do všech množin podle daných funkcí. Z nich se vyberou nenulové a ty postupují do dalšího řízení. Pokud najdeme množinu, která jasně vítězí nad ostatními (podle vhodně zvoleného prahu), rozhodnutí je u konce. Pokud je však příslušnost dané hodnoty na pomezí mezi dvěma hodnotami, bude třeba pro rozhodnutí využít i jiných znalostí (počet zubů, pořadí zubů apod.).

2.6 Rozpoznávání, aproximace tvarů

Dalším problémem nutným vyřešit je proložení předem známého tvaru do objektu. V mém případě se situace zjednodušuje na dva rozměry (výšková mapa) a jeden tvar – zubní křivku.

Tento problém je nutné dělit na dvě části, z nichž jedna je přesně popsaná v teorii, druhou budu muset řešit spíše v praxi. V prvním případě se jedná o samotné nalezení vhodné transformace šablony křivky zubů, kterou je nutné přesně napasovat na zubní oblouk, druhým problémem je nutnost řešit doladění křivky na místě s ohledem na různou stavbu pacientovy čelisti.

Problém registrace dat s křivkou by se dal velice elegantně řešit algoritmem RANSAC.

2.6.1 RANSAC

Tento algoritmus je speciálně navržen pro práci s velkým množstvím dat, jako jsou např. právě obrázky. Základní myšlenkou je neprocházet iteračně všechna data stále stejně, ale v každém kroku najít náhodné vzorky dat a ty poté použít k nalezení nejvhodnější transformace. Samotný chod algoritmu je velmi robustní a i pro velká množství dat (např. napasování dvou fotografií do sebe) dosahuje velice dobrých výsledků. Koncept algoritmu je následující:

Mějme model o celkovém počtu M vzorků, na kterém definujeme fitness funkci, která udává hodnotu správného usazení objektu. Pro můj případ postačí jednoduchá suma čtverců vzdáleností ke křivce, čili:

$$fitness(curve) = \sum_{i=1}^{M} lenght(sample_i, curve)^2 \quad (13).$$

Při řešení této problematiky však může dojít k případům, kdy suma čtverců vzdáleností nepostačí. Fitness funkci lze kombinovat z více dílčích řešení, složených například z již zmiňované vzdálenosti od křivky, délky křivky, její křivosti apod.

Je samozřejmě možné hledat i převrácenou hodnotu, podle toho hledáme-li ve výpočtu minimum či maximum. Dále si definujeme transformační funkci T, která se použije na šablonu vzorové zubní křivky tak, aby ve výsledku tato křivka pasovala co nejlépe na zkoumanou množinu dat. Transformaci T má za úkol najít právě algoritmus RANSAC. Celý průběh je popsán následovně:

- Vyber N vzorků dat z celkového počtu M.
- Vypočti transformaci T tak, aby křivka procházela všemi zvolenými body.
- Aplikuj transformaci T na původní křivku.
- Vypočti hodnotu funkce **fitness** ze všech vzorků dat.
- Pokud hodnota funkce splňuje podmínky (např. minimální hodnotu), nebo bylo dosaženo nejvyššího možného počtu opakování, skonči.
- Jinak opakuj od začátku.

V mém případě, pokud budou data reprezentována maskou detekovaných zubů, postačí vybírat přímo z hodnot těch polí, která jsou označena právě jako region zubů. Jednotlivé body mohou poté reprezentovat přímo řídící body přímky, mohou však sloužit pouze k výpočtu transformace.

2.7 Proložení roviny mračnem bodů

V mém navrhovaném algoritmu bude nutné též vyřešit problém proložení roviny mračnem bodů. Pro to využiji jednu z nejběžnějších metod – metodu nejmenších čtverců. Celý algoritmus je založen na minimalizaci sumy čtverce vzdáleností jednotlivých bodů od proložené roviny:

$$Q = \sum_{i=0}^{i<N} Dist_i^2 \quad (14),$$

kde $Dist_i$ je vzdálenost bodu i od roviny. Je tedy nutné najít rovinu, která má Q minimální. Předpokládáme-li, že hledaná rovina je normalizovaná (Ax + By + Cz = 1), lze tento vztah přepsat jako:

$$Q = \sum_{i=0}^{i<N} (Ax_i + By_i + Cz_i - D)^2 \quad (15).$$

Ze vztahu (15) nyní vyjádříme parciální derivace pro vyhledání minima Q (je-li derivace rovna 0, v bodě se nachází lokální extrém), tedy:

$$\frac{\partial Q}{\partial A} = \sum_{i=0}^{i<N} (2x_i(Ax_i + By_i + Cz_i - D)) = 0 \quad (16),$$

$$\frac{\partial Q}{\partial B} = \sum_{i=0}^{i<N} (2y_i(Ax_i + By_i + Cz_i - D)) = 0 \quad (17),$$

$$\frac{\partial Q}{\partial C} = \sum_{i=0}^{i<N} (2z_i(Ax_i + By_i + Cz_i - D)) = 0 \quad (18),$$

$$\frac{\partial Q}{\partial D} = \sum_{i=0}^{i<N} (-2(Ax_i + By_i + Cz_i - D)) = 0 \quad (19).$$

Uvědomíme-li si, že $\sum_{i=0}^{i<N} D = ND$, můžeme vztah (19) přepsat jako:

$$D = Ax_0 + By_0 + Cz_0 \quad (20),$$

kde $x_0 = \sum_{i=0}^{i<N} \frac{x_i}{N}, y_0 = \sum_{i=0}^{i<N} \frac{y_i}{N}, z_0 = \sum_{i=0}^{i<N} \frac{z_i}{N}$.

Z tohoto zápisu je zřejmé, že x_0, y_0, z_0 značí těžiště mračna bodů. Jinými slovy hledaná rovina bude jistě procházet těžištěm tohoto mraku. Nyní postačí pouze vyjádřit normálový vektor roviny. Pokud odečteme těžiště od každého bodu a substituujeme do vztahů 16 – 18, dostaneme soustavu rovnic:

$$WP = 0 \quad (20),$$

kde:

$$W = \begin{vmatrix} \sum_{i=0}^{i<N} (x_i - x_0)^2 & \sum_{i=0}^{i<N} (x_i - x_0)(y_i - y_0) & \sum_{i=0}^{i<N} (x_i - x_0)(z_i - z_0) \\ \sum_{i=0}^{i<N} (y_i - y_0)(x_i - x_0) & \sum_{i=0}^{i<N} (y_i - y_0)^2 & \sum_{i=0}^{i<N} (y_i - y_0)(z_i - z_0) \\ \sum_{i=0}^{i<N} (z_i - z_0)(x_i - x_0) & \sum_{i=0}^{i<N} (z_i - z_0)(y_i - y_0) & \sum_{i=0}^{i<N} (z_i - z_0)^2 \end{vmatrix},$$

$$P = \begin{vmatrix} A \\ B \\ C \end{vmatrix}.$$

Řešení soustavy těchto rovnic pro nás znamená řešení metody nejmenších čtverců. Protože však pro tuto soustavu existuje triviální řešení A=B=C=0, je třeba stanovit počáteční podmínku pro naši rovinu tak, abychom toto triviální řešení při výpočtu eliminovali. Předpokládali jsme, že rovina je normovaná, můžeme tedy psát, že:

$$A^2 + B^2 + C^2 = 1 \quad (21).$$

Díky této podmínce můžeme přepsat výše zmíněnou matici (20) jako problém vlastního hodnot a čísel, tedy:

$$WE = VE \quad (20),$$

kde V značí matici vlastních čísel a E matici vlastních vektorů. Po spočítání vlastních hodnot a vektorů obdržíme matici tří vlastních hodnot, vyjadřujících součty vzdáleností, a matici 3x3 vlastních vektorů, vyjadřující vektory hlavní variance dat. To pro nás znamená, že ze třech nalezených vektorů postačí vybrat ten, který reprezentuje největší varianci dat.

Po obdržení vektoru již snadno zjistíme normálu hledané plochy a dopočítáme hodnotu D díky jedinému bodu na ploše, který známe, tedy těžiště mračna bodů (zdroj [20]).

3 Návrh řešení

Tato kapitola pojednává o mnou navrhovaném řešení, které bude třeba implementovat. Postupně popíši celkový koncept navrhované aplikace, způsob načítání vstupních dat, jejich předzpracování apod. Poté se pokusím přiblížit ideu vlastní detekce zubů, vycházející z detekce zubního oblouku ve dvourozměrném prostoru, dále ideu následného segmentování samotného modelu a vyhledání regionů náležících zubům. Jako poslední se pokusím vyřešit otázku klasifikace zubů, čili přesného přiřazení zubů jednotlivým třídám.

Jako první tedy popíši koncept aplikace jako celku:

3.1 Celkový návrh aplikace

Aplikace má být podle zadání použitelná v praxi, nebo musí mít alespoň možnost jednoduchého ovládání. Podle mého názoru by bylo ideální grafické uživatelské rozhraní s pohledem na zpracovávaný model s jednoduchou lištou pro změny v nastavení. Dále by mohla obsahovat panel s tlačítky pro pohyb mezi jednotlivými detekovanými zuby.

Lze předpokládat, že detekce nebude fungovat na 100% z různých důvodů. To může být způsobeno nepřesným sejmutím otisku, abnormálním tvarováním pacientovy čelisti (což se dá předpokládat, bereme-li v úvahu, že aplikace bude používána právě ve stomatologii ke korekcím abnormalit), případně úplnou absencí některých zubů. Tyto možnosti by měl navrhovaný algoritmus ve většině případů filtrovat, bohužel lze předpokládat, že nebude vždy úspěšný. Proto by bylo vhodné připravit pro uživatele určité korekční dialogy, kterými bude možné modifikovat nebo úplně změnit výsledek klasifikace.

V poslední řadě by bylo vhodné implementovat mechanizmus na vizualizaci korekcí zubů, ve formě možnosti posunovat jednotlivé zuby do jiných pozic a měnit detekované regiony. Aplikace by též měla mít možnost dále oddělené zuby posouvat a tím plánovat korekce zubů, ukládat detekované zuby do samostatných modelů, případně jiné operace.

3.1.1 Navrhované externí knihovny

Podle návrhu popsaného v hlavičce této sekce jsem se rozhodl skombinovat tři technologie při implementaci výsledné aplikace:

- Pro samotný průchod modelem a operace nad ním jsem zvolil knihovnu Medical Data Segmentation Toolkit (MDSTk)[13], která byla původně určena pro segmentaci medicínských 2D/3D dat. Obsahuje řadu rutin pro zpracování volumetrických dat (filtrace, detekce hran apod.), stejně tak jako modul pro zpracování vektorové grafiky pro modelování objektů uložených v hraniční reprezentaci. Použití tohoto modulu jsem zvolil právě z tohoto

důvodu – bude třeba mnohokrát rychle a přehledně procházet modelem a **VectorEntity** poskytuje řadu usnadňujících funkcí. Navíc se zde nachází i modul pro segmentaci dvourozměrných obrázků, který bych rád využil pro extrakci informací z výškové mapy, a matematický modul usnadňující práci s geometrickými transformacemi.

- Pro zobrazení scény a práci s ní využívám knihovnu Open Scene Graph (OSG) [14]. OSG je opensource projekt pro výkonné zpracování trojrozměrné grafiky, hojně využívaný výrobci programových řešení na poli vizuální simulace, her, virtuální reality apod. Knihovna je postavená na komunikaci s OpenGL a je vytvořena s důrazem na multiplatformní využití. Z knihovny předpokládám využití funkcí pro zobrazení a přímé interakce uživatele s modelem (posuny myší, pohledy do scény apod.).
- Poslední předpokládanou externí knihovnou je wxWidgets[15]. Celá aplikace by měla mít příjemné a intuitivní uživatelské rozhraní, což poskytuje právě tato knihovna. Obsahuje multiplatformní programátorské rozhraní pro vytváření grafických UI nezávislých na daném operačním systému.

Při samotné implementaci se v programu též vyskytují pomocné soubory zapůjčené společností 3Dim-Laboratory [18], která mi je laskavě poskytla. Jedná se o rutiny pro počítání křivky, výškové mapy a indexování regionů. Podrobněji jsou tyto zdroje popsány v kapitole 4 v popisu implementace problému a v samotném kódu aplikace jsou tyto části řádně označeny.

S popsanými technologiemi by neměl být problém dosáhnout předpokládané funkčnosti a vizuální stránky. Okno OSG pro zobrazení lze snadno vnořit do standardního okna wxWidgets, takže bude pro uživatele připraveno okno, na které je zvyklý, rozšířené o pohled do 3D scény. Navíc všechny tři externí knihovny jsou psány multiplatformně, nebude tedy v budoucnosti problém s přenositelností na jiné platformy (vývoj předpokládám na platformě MS Windows 7).

3.2 Návrh algoritmu detekce

V mé bakalářské práci jsem se přesvědčil, že není možné zuby vyhledávat pouze pomocí samotného tvaru, protože lidská čelist je plná nerovností, které detekci komplikují, a ve výsledku se musí model příliš filtrovat. To způsobuje nepříjemné vedlejší účinky, jako například nerozpoznání zubů s nevýrazným oddělením od dásně apod. Proto bude nutné umístit detekované regiony do širšího kontextu. Tím by mohlo být umístění na zubní křivce.

Protože však může detekce a napasování křivky ve třech rozměrech dosahovat příliš velké výpočetní náročnosti (a pravděpodobně nedostatečné přesnosti), rozhodl jsem se křivku najít v dvourozměrné výškové mapě.

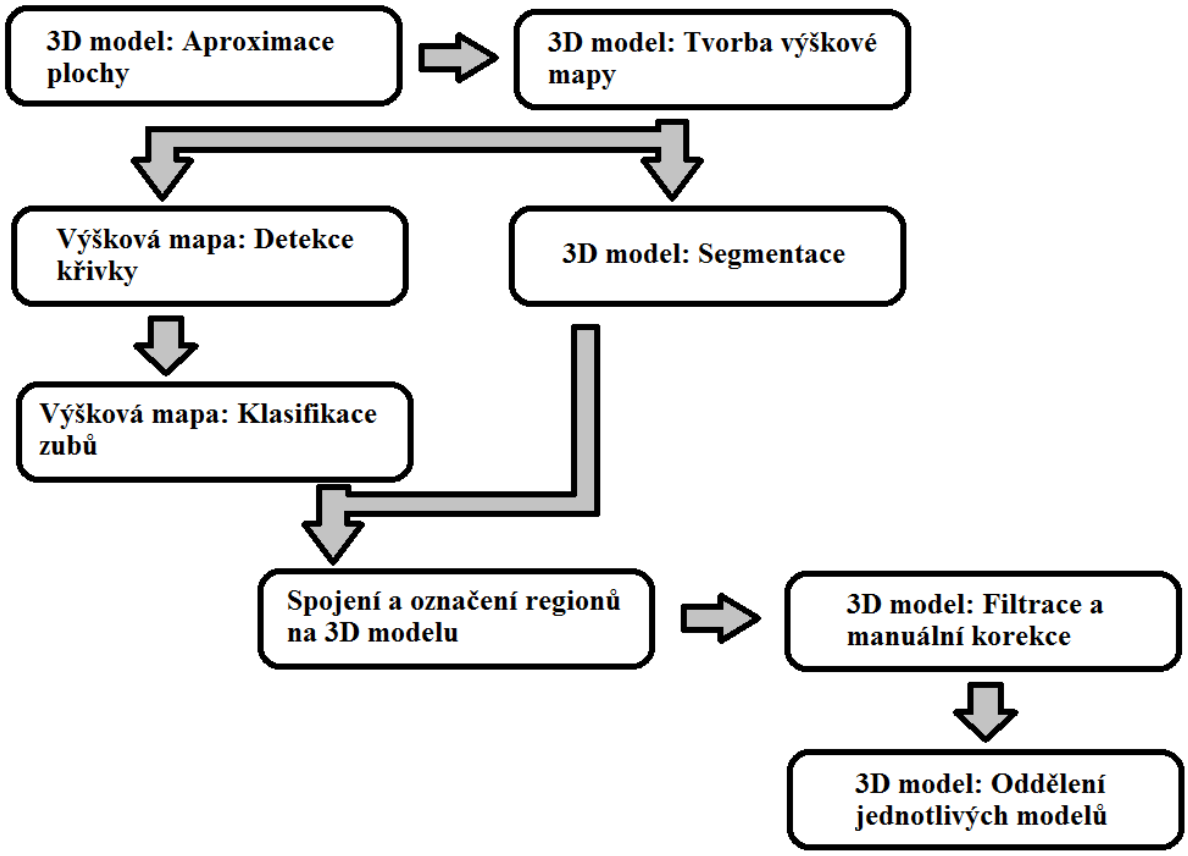
Prvním krokem algoritmu tedy bude vyhledání vhodné aproximace roviny, pomocí které nejprve transformuji lopatku s otiskem na povrch reprezentující pouze zuby. Ten následně promítnu

jako výškovou mapu přímo do detekované roviny. Tímto bych měl získat obrázek s dosti výrazným zubním obloukem (v podstatě pohled shora na zubní oblouk), na kterém nebude problém vyhledat zubní křivku.

Následně je nutné na křivce vyhledat a klasifikovat zuby. Zde se dostáváme ke kritické části navrhovaného klasifikačního algoritmu. Zde bude nutné nejvíce experimentovat, aby byla klasifikace co nejpřesnější.

Zde prozatím detekce ve dvou rozměrech končí a algoritmus přesune svůj zájem na trojrozměrný model, který nejprve segmentuje pomocí výpočtu křivosti a následné aplikace metody rozvodí.

V tomto bodě by se měly oba přístupy spojit, protože nebude možné detekovat zuby podle tvaru (např. stoličky jsou všechny příliš podobné). Podle detekované křivky, regionů na výškové mapě a fuzzy množin příslušnosti jednotlivých zubů na určité místo na zubním oblouku, jež se kombinují s jednoduchou klasifikací podle tvaru zubu, bych měl dosáhnout podle všech předpokladů relativně vysoké úspěšnosti detekce a klasifikace jednotlivých zubů. Na obrázku 3.2.1.1 můžete vidět navržený postup schematicky. Nyní popíši podrobně jednotlivé kroky tohoto algoritmu.



Obr. 3.2.1.1 – Schéma postupu detekce

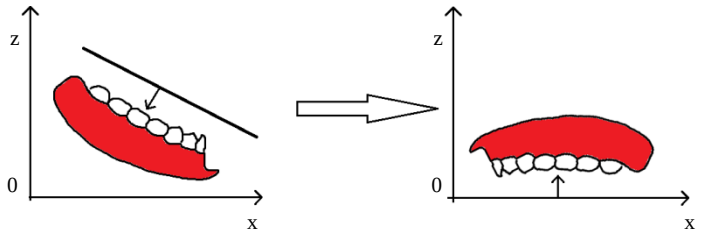
3.2.1 Aproximace plochy

K tomuto kroku by mohla postačit jednoduchá aproximační metoda nejmenších čtverců. Protože samotná čelist má rozměry dosti specifické, měla by tato metoda bez problémů stačit (i hodně zdeformovaná čelist by měla být více široká než vysoká). Pokud však bude tento postup nedostatečný, navrhuji použití složitějšího algoritmu, např. RANSAC (viz modifikace algoritmu z kapitoly 2.7.1) apod., který by přímo na místo dosadil i zubní křivku. Podle všeho ale tento krok nebude nutný. Ke konci by bylo vhodné detekovanou rovinu posunout tak, aby model neprotínala, ale aby se nacházela na jednom jeho konci. Takto bude možné nakonec promítnout na rovinu výškovou mapu reprezentující „pohled shora".

Bude samozřejmě nutné též vyřešit problém s orientací modelu, ale zde se dá předpokládat, že při součtu normál všech trojúhelníků modelu bude výsledná normála ukazovat směrem na zubní oblouk od výškové mapy. To z důvodu, že každý zubní model má buď mnohonásobně vyšší počet trojúhelníků na straně zubů, nebo pokud se jedná o upravený převedený CT model, obsahuje trojúhelníky tvořící pouze zuby, orientované právě tímto směrem.

3.2.2 Tvorba výškové mapy

Zde použiji transformace k vyrovnání modelu a proložené roviny. Nejprve vypočítám transformační matici pro detekovanou rovinu tak, abych ji srovnal s rovinou XY a posunul do souřadnice z=0, poté aplikuji tutéž transformaci na model. Tím docílím zjednodušení tvorby výškové mapy tak, že po aplikaci transformace stačí model pravoúhle promítnout do roviny. Jeho hodnoty osy Z pak budou reprezentovat přímo vzdálenost od průmětné roviny.



Obr. 3.1.5.1 – Transformace do roviny XY

Jakmile bude model vyrovnán s rovinou XY, převedu jej do výškové mapy metodou rasterizace všech trojúhelníků. Postupně projdu celý model, z každého trojúhelníku extrahuji hodnoty jeho vrcholů a ty následně promítnu do roviny XY (pravoúhlá projekce), kde jej jednoduše rasterizuji pomocí algoritmu popsaného v kapitole 2.4.1. Při samotné rasterizaci však bude nejvýše vhodné algoritmus upravit tak, aby nejlépe využíval pouze celočíselný datový typ (integer) a výpočtů probíhalo co nejméně, vše z důvodu maximálního urychlení výpočtu algoritmu. Pokud dojde ke kolizi výšek (stále se může stát, že se budou dva trojúhelníky překrývat), bude nutné přidat

pravidlo pro ponechání/přepsání stávajícího pixelu. Podle mého názoru, pokud bude rovina v pozici shodné s obr. 3.1.5.1, postačí ponechat nejnižší Z hodnotu (případné artefakty „pod zuby“ nás nezajímají).

V tomto kroku využívám již zmíněné algoritmy poskytnuté společností 3Dim-Laboratory [18], umožňující tvorbu výškové mapy. Přidávám však ještě normalizace výškové mapy na rozmezí (0..255) ve float hodnotách. Rozmezí (0..255) jsem zvolil proto, že v mnoha případech bude nutné tuto mapu zjednodušit do 8 bitových hodnot pro urychlení výpočtu. Samotnou normalizaci provádím z důvodu, že rozdílných souřadnic každého modelu a následné unifikace budoucích algoritmů.

3.2.3 Detekce zubní křivky

Zubní křivku jsem se rozhodl detekovat na průmětu do výškové mapy, protože pouze zde jsem schopen přesněji umístit jednotlivé zuby do kontextu celé zubní křivky. Křivku reprezentuji jako spline, zatím blíže nespecifikovaný (tento problém budu řešit až v implementační fázi), ale s největší pravděpodobností lze předpokládat využití poskytnutého software [18].

Nejprve bude nutné na výškové mapě segmentovat regiony reprezentující zuby. V mém semestrálním projektu jsem předpokládal využití watershed algoritmu na vyhledání zubních regionů, tento postup nebyl ale ideální. Zubní oblouk totiž není ve většině případů dokonalý pro metodu rozvodí, proto segmentace nedopadla vždy podle plánu. Druhou, neméně výraznou nevýhodou byla přítomnost jiných regionů neobsahujících zuby vzniklých různými artefakty na modelu. Ty bylo nutné filtrovat.

Proto jsem navrhl jinou metodu pro detekci zubního oblouku, a to nejprve segmentovat průmět pomocí regionů stejných výšek (výšková mapa převedená do 8bitového celočíselného obrázku). Ty následně spojuji ve větší segmenty s podobnou výškou. Z těchto regionů se vybere 20% nejnižších, reprezentujících zubní oblouk. Toto číslo bylo ověřeno experimentálně, na všech testovaných modelech úspěšně filtrovalo zbytek modelu.

Takto filtrovaným modelem provedu první odhad křivky, a to pevným natočením, posunutím a zoomem pevné šablony zubního oblouku. Tím získám přibližnou polohu zubního oblouku. Na tom vyhledám nejbližší nejnižší regiony a vyznačím přibližný začátek a konec křivky (podle šablony). To mi umožní následně v ponechaných regionech najít body, kterými bude křivka procházet (upravený A* algoritmus hledání cesty tak, aby přibližně sledoval odhad křivky a hledal cestu od začátku až do konce).

Z takto ponechaných regionů se pak RANSAC algoritmem najdou výsledné 6 – 10 bodů, které tvoří řídící body výsledné spline křivky.

Tento postup je složitý, ale v mém případě nutný, protože detekce křivky je základním pilířem celého klasifikačního algoritmu a při pokusech s modely se ukázalo, že obsahují příliš artefaktů na použití jednoduchého RANSAC algoritmu.

3.2.4 Detekce a klasifikace zubů na výškové mapě

Nyní je třeba klasifikovat zuby okolo detekované zubní křivky. Z předchozího kroku mám již uloženu segmentovanou výškovou mapu (využívám nespojovaný, původní segmentovaný obraz).

Nejprve však provedu projekci výšek okolo zubní křivky. Protože výškovou mapu vždy normalizuji jak výškově, tak velikostně, lze použít pevný práh na prohledání okolí křivky a uložení do projekce. Tím lze z modelu vyčíst poměrně přesný obraz zubní křivky s výraznými oddělenými zuby.

Tuto projekci je však nutné nejprve normalizovat tak, aby měla vždy pevnou velikost. Jednotlivé zuby se pak dají segmentovat pomocí prohledávání lokálních minim. Je nutné používat určitou toleranci při prohledávání minim, protože např. stoličky mají velice členitý povrch a obsahují velké množství lokálních minim. Nikdy však nedosahují takového rozdílu, jako minima oddělující dva zuby.

Následně je možné uložit segmenty jako kandidáty zubů. Samozřejmě můžou nastat dvě chybové situace:

- Zub je rozdělen na více segmentů: Bude nutné projít tedy všechny segmenty, a pokud nedosahují určité velikosti (která se na pozici křivky mění, např. stoličky ležící v první třetině zubního oblouku jsou větší než řezáky), je nutné je spojit. Pro toto je nutné implementovat algoritmus.
- Více zubů je označeno jako jeden: Zde je nutné provést obdobný, leč opačný postup. Příliš velké regiony (v závislosti na pozici na křivce) je nutné pravidelně rozdělit (lze předpokládat, že na okrajích jsou rozděleny správně).

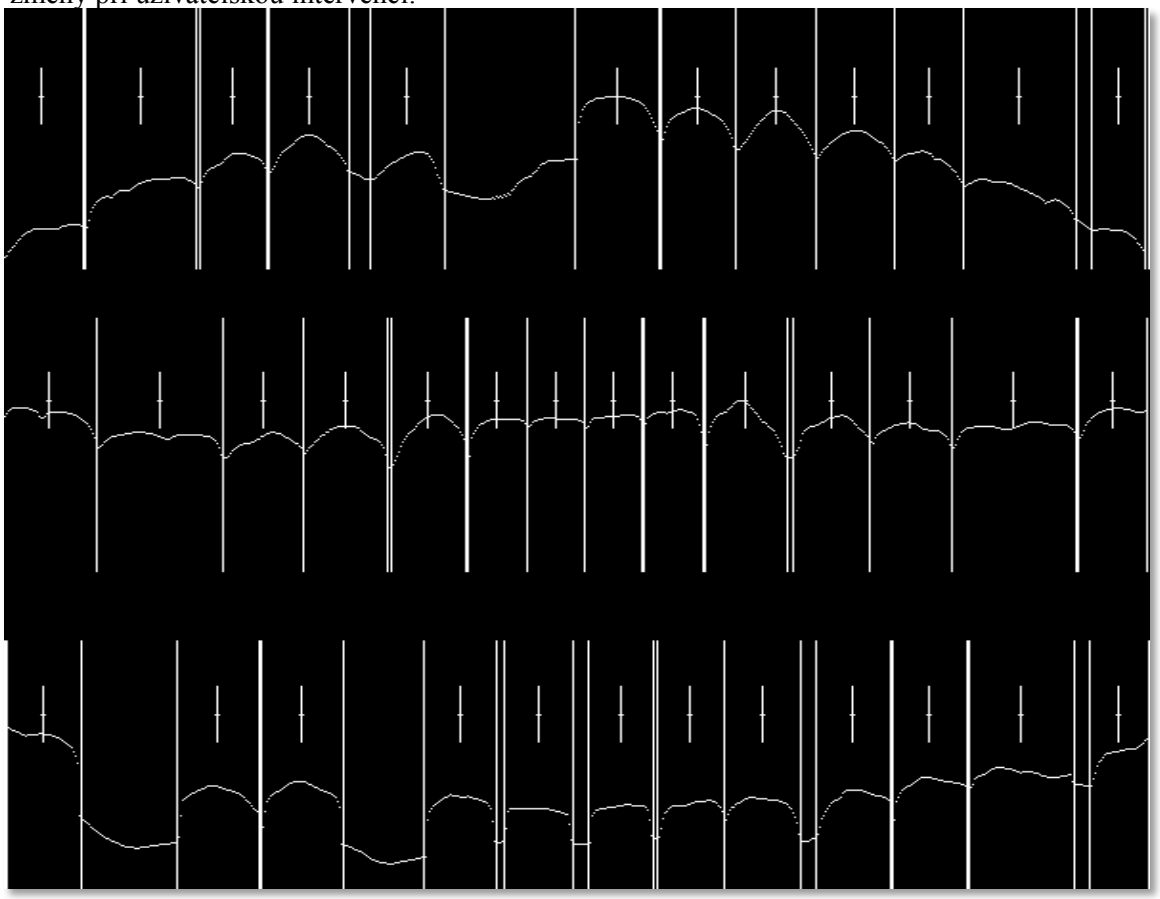
Pro oba tyto postupy jsem využil velikosti zubů z testovacích modelů, které byly ve všech případech velice podobné (normovaná křivka), proto jsem usoudil, že jsou použitelné.

Následně je ještě nutné zpřesnit regiony, které obsahují jak zub, tak jistou část díry chybějícího zubu, v jejíž středu bylo nalezeno minimum. Pro toto existuje kontrola gradientu od maxima zubu. Ten budu postupně procházet a gradient se bude zvyšovat. Maximální hodnotu gradientu průběžně ukládám, a pokud se bude při průchodu dalšího bodu příliš zmenšovat, vyznačím konec zubu. Tím se mi zubní regiony zmenší pouze na regiony zubů.

Posledním filtrovacím zařízením bude vyloučení kandidátů reprezentujících pouze díry. To se provede jednoduchým výpočtem počtu bodů ležících nad a pod přímkou spojujících první a poslední bod kandidáta. Pokud je více bodů pod křivkou, zub je vyřazen. Jak je možno vidět na následujících obrázcích, díry jsou většinou konvexní, kdežto zuby konkávní. Zbývá již pouze označit názvy detekovaných zubů.

Tento problém by ale měla řešit fuzzy logika klasifikace jednotlivých zubů, jejichž množiny příslušnosti by měly zasahovat i do sousedních pozic. Samozřejmě pokud bude zub příliš odchýlen,

nebude ho možné klasifikovat. V programu to bude řešeno chybným klasifikováním a možností změny při uživatelskou intervencí.



Obr. 3.2.4.1 – Projekce na křivku s detekovanými zuby a vyznačenými středy zubů

3.2.5 Označení regionů na výsledném trojrozměrném modelu

V tomto kroku spojím veškeré znalosti z kroků předchozích, tzn. detekované a klasifikované zuby na dvourozměrném průmětu a segmentovaný 3D model. Tak bych měl označit většinu regionů na segmentovaném polygonálním objektu, které patří právě zubům. Je zřejmé, že tyto oblasti nebudou celé zuby, pouze jejich vrchní části, ty jsou však pro klasifikaci nejsložitější z důvodu vysoké křivosti, např. u stoliček.

Nejprve bude nutné označit regiony reprezentující zuby na výškové mapě. Zde postačí projít průmět na křivce a z každého bodu označit určité rozmezí kolmo na křivku do určitého prahu. Jak již jsem zmínil několikrát, je možno zvolit pevný práh z důvodu výškové podobnosti zubů a normalizované výškové mapy. Tím získáme označené regiony, které patří jednotlivým zubům ve dvou rozměrech.

S takto označenými zuby lze poté pomocí transformační matice model => výšková mapa lehce označit regiony na reálném modelu. Na každém segmentovaném modelu jsou boční regiony velké, takže s označením bočních částí by neměl být problém (zasahují též na vrcholky). Co se týká regionů

okolo vrchu zubu, ty jsou více segmentované, ale díky označené masce na výškové mapě by mělo být též možné je v pořádku označit. Samozřejmě, pokud bude nutné, je možné tento postup doplnit také algoritmem na korekci chybějících regionů v zubu. Zda tento postup použít či ne, to rozhodnu až po otestování navrženého algoritmu.

3.2.6 Manuální korekce

V části filtrace již zbývá pouze projít jednotlivé zuby a kontrolovat počet segmentů tak, aby výsledek klasifikace co nejvěrněji kopíroval realitu. Probíhá zde interakce s uživatelem, a to v případě, že nedojde k plně automatické detekci všech zubů. Důvodů může být několik, od anatomické anomálie pacientovy čelisti až po špatně naskenované, nevýrazné hrany na modelu. Po automatické klasifikaci bude mít uživatel možnost jednotlivé zuby označit a manuálně jim přiřazené regiony přidat, případně odebrat. Totéž může udělat s přiřazeným názvem zubu – je možné ho přímo v programu změnit. To například v případě posunu zubů po extrakci jednoho z nich a následné špatné klasifikace.

3.2.7 Oddělení jednotlivých modelů

Zde se dostávám k poslední části navrhovaného algoritmu. Vstupní data do této části obsahují korektně klasifikovaná data, ať již automaticky, nebo s pomocí uživatele. Proto se již nemusí zabývat žádnými korekcemi apod., jejím hlavním úkolem bude oddělit segmenty zubů a dásně.

To by však neměl být velký problém, jednotlivé třídy dat (zuby) jsou přesně popsané, bude třeba pouze rozpoznat, který trojúhelník patří do kterého modelu, a ten posléze vytvořit v modelu novém.

Možným vylepšením této části by mohlo být automatické doplnění/vyhlazení ořezaných modelů, které budou mít po této operaci zajisté výrazně ostré hrany. Toto je však pouze kosmetická úprava pro uživatele.

Následně bude muset program obsahovat možnosti tyto modely ukládat.

3.2.8 Úpravy pozic zubů

Možnou úpravou po vlastní segmentaci modelů by mohlo být ještě vlastní plánování korekcí zubních oblouků. To bych mohl řešit implementací vlastního modulu pro interaktivní transformace modelů přímo v trojrozměrném pohledu. Jedná se o přidání jednoduchého „draggeru“ obsahujícího ovládání pro posun a rotaci ve třech směrech.

Uživatel si tak bude moci libovolně upravovat a promítat segmentovaný model podle jeho přání a plánů, ověřovat, zda jsou korekce fyzicky možné apod.

4 Implementace

Tato kapitola podrobně popisuje vlastní implementaci navržené metody. Popisuji zde koncepci navrženého programu, užité třídy a jejich funkce tak, jak byly použity v této práci.

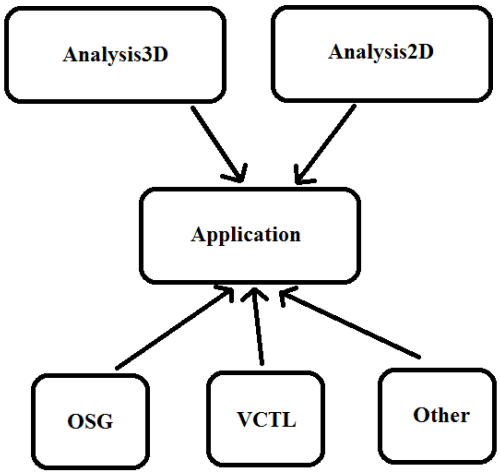
Kapitola je rozdělena podle modulů v návrhu, reflektujících jejich funkci přímo v programu. Pokud jsou užity externí zdroje, jsou zde řádně označeny.

4.1 Celková koncepce aplikace

Implementaci jsem rozdělil do několika modulů, každý obsluhuje specifickou funkci v aplikaci. Jedná se o tyto moduly, které budu později více popisovat v jednotlivých kapitolách:

- **Application**
  - Modul obsahuje GUI aplikace, vytvořené pomocí nástroje DialogBlocks [19].
- **OSG**
  - Modul obsahuje nástroje užité při práci s modelem a současně využívající OpenSceneGraph.
- **Analysis2D**
  - První část navrhovaného algoritmu – veškeré práce s dvourozměrnou výškovou mapou.
- **Analysis3D**
  - Druhá část navrhovaného algoritmu – práce s trojrozměrným modelem, jeho segmentace a výpočty křivosti.
- **Vctl**
  - Modul rozšiřující určité třídy z Medical Segmentation Toolkit.
- **Other**
  - Jedná se o zdrojový kód poskytnutý společností 3Dim Laboratory [18]. Nacházejí se zde rutiny pro práci s výškovou mapou, křivkou apod.

Koncepce a umístění jednotlivých modulů je patrné na grafice:



Obr. 4.1.1 – Koncepce modulů

V následujících kapitolách podrobněji rozeberu některé moduly. Zvláštní pozornost však věnuji hlavně části reprezentující čistý navržený algoritmus. Pomocným třídám věnuji pouze okrajovou pozornost.

4.2 Modul Application

Jedná se o implementaci grafického uživatelského rozhraní. Většina tříd implementovaných v tomto modulu obsahuje pouze definice vzhledů oken pro interakci s uživatelem. Zvláštní pozornost si však zasluhuje třída `ToothClassifier`, ve které se nachází rutiny volané při různých vstupech. Ty obsluhuje speciální proměnná `logic`, obsahující funkce pro obsluhu jednotlivých uživatelských akcí.

4.3 Modul OSG

Zde se nacházejí třídy zajišťující spojení aplikace – MDSTk – OSG. Jedná se o tyto třídy:

- `OSGCanvas`
  - Tato třída reprezentuje hlavní OSG rozhraní ve wxWidgets. Byla vytvořena podle tutoriálu dostupného na internetových stránkách projektu OSG [15], proto je zde převážná většina kódu shodná s tímto ukázkovým příkladem. Jedná se o funkce reagující na různé uživatelské vstupy, pohyby myši apod. Tyto zprávy třída deleguje dále OpenSceneGraph prohlížeči.
- `OSGModelHandler`

- V této části jsem implementoval hlavní rozhraní mezi MDSTk modelem a OSG modelem pro zobrazení. Třída si uchovává svoji vlastní strukturu ve stromu OSG (může obsahovat více nezávislých modelů), které tvoří z původních Vector Entity trojúhelníkových modelů. Obsahuje funkci na smazání veškerého interního obsahu a pro přidání jednoho modelu do stromu. Tak lze v průběhu programu libovolně měnit zobrazovaná data. Též umožňuje měnit barvu jednotlivým vrcholům, případně skrývat/zobrazovat dané modely. To umožňuje uživateli libovolně měnit pohledy na model zubní čelisti.
- Další činností této třídy je údržba a obsluha draggerů pro interaktivní změnu pozic zubů pro plánování korekcí pacientovy čelisti. Tato funkce představuje náznak možného budoucího využití této aplikace v praxi.

- `OSGEventHandler`
  - Poslední třída v tomto modulu má na starosti ošetření uživatelských vstupů přímo na modelu. Pro maximální zjednodušení jsem zde implementoval pouze reakci na kliknutí myší na model, přičemž se deleguje dále pouze index vrcholu a ID modelu, ke kterému patří. Další funkcí této třídy je obsluha ovládacích prvků jednotlivých částí modelu a v poslední řadě jejich zobrazování/vypínání. Tak vznikla přenositelná třída, kterou lze použít jak na model oddělený, tak na ten původní. Pokud se uživatel nedotkne klikem modelu, zpráva se předává řízení OSG Vieweru, který ji následně zpracuje jako vstup pro rotaci/posun/zoom modelu.

4.4 Modul VCTL

Tento modul tvoří pouze jedna třída – `MCTriSExt`. Tu jsem pojmenoval ve formátu všech tříd ve Vector Entity a též ji umístil do shodného **namespace**. Jedná se o rozšíření MDSTk trojúhelníkového modelu o určité funkce, usnadňující procházení v mých algoritmech. Jedná se o tyto rozšiřující funkce:

- Snadnější načítání modelu ze souboru – nyní se jedná o jednoduchou funkci s jedním parametrem – název souboru.
- Existenci mapovací funkce (vrchol → barva), kde je možno každému bodu přiřadit určitou barvu. To se využívá pro OSG zobrazení.
- Přidání určitých funkcí při načítání modelu, hlavně pak přiřazení indexu ke každému vrcholu. K tomu jsou využity původní funkce MDSTk. Novou funkcí je počítání indexů vrcholů tak, aby korespondovaly s indexy OSG. Tento přístup velice zjednodušuje propojení OSG a MDSTk.

- Nejdůležitější rozšíření – funkce pro rozdělení modelu na více částí. Této funkci je na vstupu předána mapa (vrchol → model), přičemž je poté model rozdělen do vektoru nových modelů podle této mapy. Vrcholy, které nejsou obsaženy v mapovací funkci, jsou ponechány v původním modelu, ostatní jsou přesunuty (tedy vloženy do nového a z původního smazány).

4.5 Modul Other

Zde se nachází zdrojové soubory zapůjčené společností 3Dim Laboratory [18]. Jedná se o pomocné rutiny při hlavních výpočtech:

- `CCardinalSpline` a `CCurve2D`
  - Tyto třídy zapouzdřují křivku mnou využívanou pro reprezentaci zubního oblouku.
- `CHeightMap`
  - Třída obsahuje funkce pro práci s výškovou mapou. Po předání modelu a roviny dané normálou a posunem promítne daný model do roviny. Zároveň spočítá i transformační matici mezi reálnými souřadnicemi modelu a jejich průmětem na výškovou mapu.
- `CRegionIndex`
  - Poslední zapůjčenou třídou je CRegionIndex. V té je implementována logika indexace segmentovaného obrazu – na vstupu je předán MDSTk obraz regionů, který je následně indexován, jsou spočítány jisté statistické charakteristiky a tato třída obsahuje též pár funkcí pro úpravu těchto regionů.

Modul **Other** též obsahuje třídu `CTriIterator`, která je využívána při projekci výškové mapy. Jedná se o rasterizátor trojúhelníku, tedy iterační pomůcku pro procházení všech pixelů 2D trojúhelníku.

Zde bych chtěl poděkovat společnosti 3Dim Laboratory za možnost využívat tyto soubory v mojí práci.

4.6 Modul Analysis2D

Nyní se dostávám ke dvěma hlavním modulům celého mého navrženého algoritmu. Prvním z nich je **Analysis2D**, který jak již název napovídá, obsahuje veškerou logiku detekce a klasifikace ve dvou rozměrech, tedy na výškové mapě. Jedná se o obsahově nejrozsáhlejší modul, protože se zde nachází nejdůležitější funkce celého klasifikačního algoritmu.

Tuto kapitolu pro snadnější orientaci dále rozdělím do více částí podle jednotlivých funkcí.

4.6.1 Hledání zubní křivky

Jak již jsem zmínil v předchozích kapitolách, hledání zubní křivky je zásadní pro můj postup, protože zuby nelze klasifikovat jen podle jejich vzhledu, ale je nutné je zasadit do vyššího kontextu. Právě touto funkcí bych se měl zabývat zde v této kapitole.

Hlavní volaná funkce `ImgToothClassifier::findCurve()` obsahuje ve svém těle hlavní postup hledání zubní křivky. K tomu využívá větší množství pomocných rutin, ze kterých ty nejdůležitější popíši též. Nejprve ale představím hlavní postup při hledání zubní křivky:

1. Prvním krokem algoritmu je segmentace výškové mapy. Jedná se v podstatě o jednoduchý segmentační nástroj spojující pixely shodné výšky. K tomu je zapotřebí upravit vstupní obraz z čísel s plovoucí řádovou čárkou na celočíselný datový typ. Reálným vstupem je tedy 8bitový obrázek reprezentující výškovou mapu a výstupem, poté obrázek reprezentující regiony.
2. Následně tento obraz indexuji a odstraním malé segmenty, případně segment největší (vzduch okolo otisku).
3. Po tomto filtrování volám svůj vlastní indexovací algoritmus (`ImgRegions`), který obsahuje funkce pro spojování a mazání regionů podle různých kritérií:
    a. Spojování podle podobné výšky – této funkci lze předat experimentálně zjištěné maximální rozdíly (pevné hodnoty), protože stále pracujeme s normalizovanou výškovou mapou.
    b. Mazání regionů, které představují vrcholky a hřebeny na výškové mapě (zuby jsou vždy reprezentovány jako údolí).
    c. Filtrování malých regionů – tento postup lze použít ve větší míře, protože každý zub lze po předchozím spojování regionů považovat za velký region. Pokud zde zmizí i určité regiony reprezentující zuby, na následující postup by tento fakt neměl mít zásadní vliv.
    d. Filtrování regionů, které nevyplňují určité procento příslušného orientovaného obvodového obdélníku. Tím docílím, že z modelu odstraním regiony, které reprezentují například dáseň okolo zubu, tedy ty, které jsou sice početně velké, ale jejich obvodový obdélník zabírá mnohonásobně větší plochu.
    e. Poslední z využitých funkcí je konverze regionálního obrazu na graf. V následujícím postupu budu hledat nejkratší cestu mezi regiony, proto jako vstup potřebuji graf s uzly reprezentujícími středy regionů a hranami ohodnocenými jejich vzdálenostmi vůči sobě.
4. Ještě však dříve, než přejdu na finální odhad křivky, pokusím se odhadnout křivku pouze pomocí středů regionů. Ty by měly v současné době reprezentovat zuby s případnými artefakty okolo. Jako odhad jsem zvolil algoritmus, který vezme šablonu

zubního oblouku, tu se pokusí zvětšit/zmenšit podle detekovaných regionů, posunout ji do jejich středu a následně procházením všech možností určit vhodný úhel natočení. Křivka nebude přesně na zubním oblouku, ale jako původní odhad postačí.

5. Podle křivky jsem poté schopen označit přibližný počáteční a koncový bod zubního oblouku. To provádím odhadnutím nejbližších uzlů grafu ke koncovým bodům odhadnuté křivky.
6. Se znalostmi o koncových bodech, odhadu křivky a grafu regionů dále mohu volat algoritmus uložený ve třídě `ImgAStar`, který pomocí mnou upraveného algoritmu A* hledajícího nejkratší cestu regiony od počátečního bodu ke koncovému označí středy regionů, jež budou pokračovat k dalšímu zpracování. Tento postup má dva důvody:
    a. Docílím tím uspořádání regionů podle předpokládané zubní křivky.
    b. Filtruji regiony, které se příliš vzdalují původnímu odhadu křivky.

    Původní A* algoritmus jsem pro tento případ upravil tak, aby při odhadu celkové vzdálenosti a počítání reálné vzdálenosti bral v úvahu přílišné odklonění od odhadu křivky a hodnotu výsledné cesty patřičně znevýhodnil.
7. Posledním krokem hledání křivky je algoritmus RANSAC, který z vybraných kandidátů stochasticky vybere určitý počet nejlépe vystihující danou křivku.

Výstupem celého tohoto postupu je tedy křivka ve výše zmíněné struktuře, reprezentující zubní oblouk. Protože se však používá stochastický algoritmus, který není vždy stoprocentně účinný, uživatel je následně dotázán na upravení křivky tak, aby co nejlépe odpovídala realitě.

Největším problémem tohoto postupu, jak se ukázalo při testování, je příliš velké množství artefaktů ve výškové mapě, ať již se jedná o chybně filtrované regiony, nebo regiony tvarem zub připomínající, ale ve skutečnosti reprezentující např. jiný předmět přítomný v modelu (častý znak u modelů vzniklých konverzí z CT dat).

4.6.2 Detekce a klasifikace zubů

V této části již máme spočítanou a řádně usazenou zubní křivku (předpokládejme, že díky intervenci uživatele je možné vždy uvažovat křivku za správně usazenou). Nyní je nutné pomocí zubní křivky a výškové mapy detekovat jednotlivé zuby. Postup a volaná funkce se nachází v `ImgToothClassifier::findTeeth()`.

K tomu využiji původní výškovou mapu, ze které vytvořím vertikální projekci na křivku. V praxi to znamená, že v každém bodě křivky prohlédnu její nejbližší okolí (v reálné podobě se jedná o rozmezí kolem 1/20 šířky celé výškové mapy, pro každý bod postupuji po normále) a do výsledných hodnot vynáším nejnižší nalezené hodnoty. To má za následek, že celý detekční algoritmus není závislý na přesné pozici křivky, pokud bude zub v prohledávaném rozmezí, na křivkovém průmětu bude vždy shodně zobrazen.

Tuto projekci je třeba též provést normovaně, tedy tak, aby měla vždy shodnou velikost a aby respektovala rozdílné délky segmentů křivky. Výsledkem by tedy měl být seznam **n** nejnižších hodnot okolo zubní křivky.

4.6.2.1 Zkoumané a nepoužité postupy

Zde bych si dovolil informovat o jiných alternativách detekce zubů, které jsem při řešení této práce zkoumal, leč nakonec nepoužil, z důvodu malé úspěšnosti na testovacích modelech.

Segmentace pomocí detekovaných regionů

Myšlenkou tohoto postupu bylo využít již segmentovaného obrazu z předchozí kapitoly (4.6.1) a vhodným spojením regionů (zcela jistě ne takového rozsahu jako při detekci křivky) filtrovaných vzdáleností ke křivce označit zubní regiony. Tento postup sice při detekci vykazoval určité kladné výsledky, zcela jistě označil všechny zuby po celém jejich obsahu. Objevily se zde i dvě hlavní nevýhody:

- Zuby nejsou vždy kolmo k promítané rovině, tedy regiony, které vznikly spojováním pixelů se stejnou výškou, často zasahovaly do více zubů. Tyto se poté velice těžko detekovaly a následně klasifikovaly.
- Druhou nevýhodou se ukázala nemožnost spolehlivého oddělení chybějících zubů od zubních regionů. Protože se segmenty označovaly podle nejnižší hodnoty na křivce, dosáhl jsem případů, kdy část zubního oblouku byla natolik velká, že již nezasáhla do prohledávaného rozsahu a dáseň byla označena jako zub.

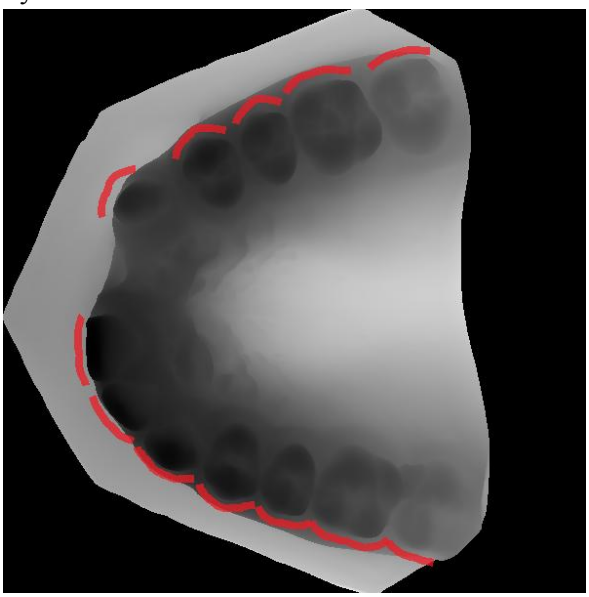
Tuto myšlenku jsem tedy opustil, zajisté by se ale dala použít v budoucnu v kombinaci s více nezávislými metodami pro zpřesnění výsledků.

Segmentace pomocí průmětu hranice s dásní

Tato myšlenka vycházela z faktu, že přední hranice zubního oblouku na výškové mapě je velice dobře čitelná (dobře čitelná na místech se zuby, nevýrazná nebo příliš blízká křivce v oblastech, kde zuby chybí). Na tomto základu jsem postavil detekční algoritmus podobný výslednému, který však nebyl plně úspěšný.

Hlavním problémem byl fakt obtížné detekce této hranice. Jak již jsem zmínil v minulé kapitole, zuby nejsou plně proložené rovinou. Neexistuje tedy přesný způsob, v jaké výšce tuto hranici detekovat. Bylo by nutné použít nějaký adaptivní algoritmus, ale ten neřeší druhý problém vycházející z tohoto přístupu – nechtěnou detekci šumu na stoličkách. Jak je známo, stoličky jsou nejhorší pro klasifikaci, z důvodu jejich členitosti. Často se stávalo, že detekční algoritmus nezachytil

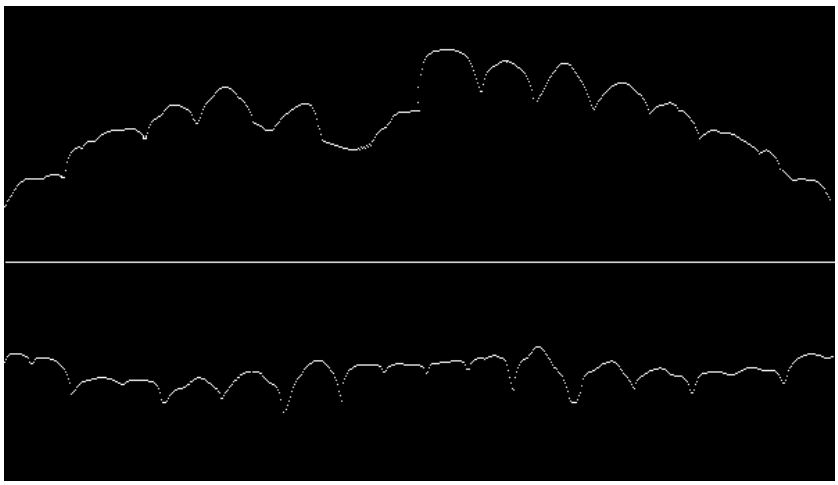
hranici zubu a dásně, ale jeden z výstupků stoličky. Následná detekce takto zašuměného zubu byla téměř nemožná. Dalším důvodem, proč jsem tento přístup vyřadil, byla náchylnost na velice přesné umístění křivky na středy zubů.



Obr. 4.6.2.1.1 – Ukázka výškové mapy s vyznačenými vnějšími hranami

4.6.2.2 Použitý algoritmus při detekci

Jak jsem tedy již zmínil, klasifikaci jsem založil na analýze vertikálního průmětu vrcholů zubů na křivku. Na následujícím obrázku (4.6.2.2.1) je možno vidět dva z testovaných zubních oblouků promítnutých na křivku. Jsou zde zcela zřetelné hranice mezi zuby reprezentované lokálními minimy.



Obr. 4.6.2.2 – Průmět minim na křivku

Na těchto datech poté provádím sadu filtrací tak, abych postupně označil pouze zubní segmenty:

1. Jako první se jedná o segmentaci samotných dat, pomocí detekce minim. Zde je však nutné nehledat pouze lokální minima, ale též hledat jejich rozdíl oproti okolí. U předních zubů to nečiní problém, u stoliček však, jak je patrné z grafiky, existuje i více minim v rámci jednoho zubu. Docházelo by tedy k přesegmentování obrazu.
2. Následujícím filtrováním je nutné sloučit příliš malé regiony. K tomu využívám procentuální umístění zubu na křivce spolu s informací o jeho šířce. Pokud je zub o polovinu menší, než je průměrná šířka zubu v dané oblasti, hledám pro něj vhodný region ke spojení. Pokud po spojení nedojde k přílišnému zvětšení zubu (více jak 1.5 násobku průměrné šířky), segmenty nebudou spojeny.
3. Analogií k předchozímu kroku je rozdělování příliš velkých regionů. To je nutné zejména u nekvalitních a málo detailních modelů, kdy minimum mezi zuby není příliš výrazné. Pokud tedy zubní kandidát dosáhne určitého násobku průměrné šířky zubu, rozděluji ho. Tento postup lze využít, protože zubní křivka se předpokládá minimálně ke druhým stoličkám. Samozřejmě, pokud by uživatel zadal křivku pouze přes několik zubů a ne přes celou šířku čelisti, algoritmus selže. Nicméně rozdíly v šířkách zubů pro spojování/rozdělování jsou i při vynechání několika zubů na koncích křivky minimální.
4. Další filtrací je úprava hranic detekovaných kandidátů. Zde se kontroluje gradient od maxima zubu. Jakmile začne gradient prudce klesat, zubu se nastaví nová hranice. To ošetřuje případy, kdy jsou dva zuby rozděleny mezerou po zubu chybějícím.
5. Posledním krokem je kontrola konvexnosti/konkávnosti kandidáta. Jak je patrné na obrázku 4.6.2.2, části obsahující zub jsou silně konkávní, kdežto mezery jsou buď konvexní, případně lineární. Tento případ ošetřuji výpočtem průměrné orientované vzdálenosti od přímky spojující koncové body zubu.

Tímto výpočtem končí filtrace zubních segmentů a předpokládá se, že v detekovaném vektoru zubních kandidátů se nacházejí již pouze celé zuby. Ty je nutné pojmenovat a klasifikovat.

Důležité pro další užití regionů, po výpočtu se ukládá mapa regionů přímo do proměnné ve třídě `ImgRegions`. To proto, že bude následně využita při spojování dvourozměrné a trojrozměrné analýzy.

4.6.2.3 Klasifikace zubů

Samotná klasifikace se provádí pomocí předpočítaných hodnot fuzzy množin reprezentujících umístění zubů na křivce. Ty jsem experimentálně odvodil z osmi vzorků zubních oblouků a vyjádřil je procentuálně jako umístění středu zubu na křivce. Pro větší přesnost tohoto postupu by samozřejmě bylo vhodné vyšetřit mnohem větší datovou sadu.

Z pozorování jsem rozdělil sady fuzzy množin na dva případy – na zubní oblouk končící druhými stoličkami (molaris secundus) a na zubní oblouk obsahující i zuby moudrosti (molaris tertius). Rozdíly pozic se u testovací sady měnily též pouze minimálně. Do budoucna bych zcela jistě doporučil rozšířit tuto sadu o více možností, případně program doplnil o adaptivní rozhodování.

Posledním nutným krokem je vybrat užití správných fuzzy množin. To jsem vyřešil jednoduchým rozhodovacím algoritmem počítající s reálnou délkou křivky na normalizované výškové mapě a průměrnou šířkou zubů. Rozdíl mezi šířkou zubů na oblouku obsahujícím 14 zubů a 16 zubů se liší dostatečně pro tento typ rozhodnutí (průmět je vždy stejně dlouhý). Zajisté i tu by bylo vhodné implementovat přesnější algoritmus.

Je zcela jasné, že pro přesnější klasifikaci zubů by bylo vhodné klasifikovat je ne pouze podle pozice, ale i podle tvaru. Tuto myšlenku jsem analyzoval též, nicméně pro její realizaci jsem neměl dostatek času a k podrobnějšímu vysvětlení se tedy dostanu v kapitole o možném budoucím vývoji. V potaz by např. připadala klasifikace podle příčného řezu zubem, kde by bylo minimálně možné oddělit řezáky a špičáky od ostatních vzorků, případně zuby zároveň klasifikovat podle předem daných šablon např. ve 3D. Tyto metody však příliš zvyšují časové náklady na vývoj, a tak je zde probírám pouze jako možnost do budoucna a podrobně je popíši v kapitole pro to určené.

4.6.2.4 Výstup detekce a klasifikace

Výstupem celého klasifikačního algoritmu je poté vektor s detekovanými a klasifikovanými zuby. Ty obsahují různé informace, např.:

- typ,
- X a Y souřadnice středu na výškové mapě,
- šířku v rámci křivky apod.

Ve skutečnosti je statistických informací pro každý zub mnohem více, ať již z důvodu minulého využití při experimentech, tak z možného využití v budoucnu. Samozřejmě je možné kdykoli tyto informace přidávat/odebírat.

4.7 Modul Analysis3D

V tomto modulu se provádějí výpočty v trojrozměrném prostředí. Jedná se o proložení roviny modelem, výpočty křivostí, následné segmentace modelu a v poslední řadě spojení dvou pohledů do jednoho modelu. Samostatnou kapitolou je třída `ModelMainLogic` obsahující hlavní funkce aplikace.

4.7.1 Hlavní obsluha aplikace

Zde jsou sjednoceny výsledné postupy při automatické klasifikaci zubů (jedná se o třídu `ModelMainLogic`). V této třídě se nachází metody volané grafickým uživatelským rozhraním

reagující na jednotlivé uživatelské vstupy. Hlavní metoda této třídy `DoTeethClassification()` obsahuje základní postup při detekci zubů:

1. Nejprve dojde k otevření vstupního souboru ve formátu .stl, reprezentujícího model otisku, případně oříznutý model získaný z CT dat.
2. Následuje tvorba instance třídy `ModelTeethCreator`, který má na starosti vhodné úpravy modelu. Tuto třídu je potřeba změnit, pokud by se v budoucnu měnil vstupní formát dat. Na jejím výstupu musí být model reprezentující pouze zubní oblouk (se správně orientovanými normálami, v případě konverze z modelu lopatky s otiskem) spolu se spočítanou plochou prokládající přibližnou zubní rovinu.
3. Dalším krokem je spočítání křivosti na modelu. V mém případě využívám pro všechny operace křivost střední, nicméně je možné z implementovaného modulu extrahovat i křivosti Gaussovy, minimální a maximální.
4. Po výpočtu křivosti je nutné model segmentovat. O to se stará třída `ModelRegions`, jež implementuje rozšířenou watershed segmentaci popsanou v následujících kapitolách.
5. V této části se režie přesouvá na výpočty ve dvou rozměrech. Nejprve se pomocí `CHeightMap` vykreslí výšková mapa podle detekované roviny. Kromě samotného obrázku výškové mapy tato třída též uchovává transformační matici ze souřadnic původního modelu do souřadnic výškové mapy. Obraz následně normalizuji na rozmezí 0..255.
6. Poté přichází na řadu detektor zubní křivky, který se snaží do výškové mapy napasovat křivku, reprezentující zubní oblouk. Samotnou detekci následuje zobrazení dialogu, který uživateli nabídne možnost upravení detekované křivky (křivka je pro můj klasifikační algoritmus stěžejní, proto musí být bez problému umístěna).
7. S nalezenou křivkou je možné najít a klasifikovat zuby na výškové mapě. O to, stejně jako o krok předchozí, se stará třída `ImgToothClassifier`. Ve výsledku obsahuje tato třída informace o nalezeném seznamu zubů.
8. S těmito informacemi je možné dva nalezené přístupy spojit. Třída `ModelComposer` analyzuje výsledky klasifikace ve 2D a na trojrozměrném modelu vyznačí regiony zubů, které korespondují s regiony ve dvou rozměrech.
9. Následuje pouze vykreslení a barevné označení zubů pro přesnější orientaci uživatele.

Mimo hlavní rutinu klasifikace zubů se v této hlavní třídě aplikace též nacházejí funkce pro obsluhu změn modelů, jako např. změna barvy podle označení, případně podle typu zubu. Jednou z hlavních funkcí je též dělení modelu podle detekovaných regionů a jeho následné ukládání a nahrávání.

4.7.2 Proložení roviny modelem

Na začátku celé analýzy modelu je nutné jím vhodně proložit rovinu tak, aby korespondovala se zubním obloukem. Pro samotnou aproximaci křivky jsem využil algoritmu popsaného Suhailem A Islamem [20]. Jedná se o jednoduché proložení roviny mrakem bodů. Po několika experimentech jsem usoudil, že tento algoritmus opravdu postačí, jelikož rozměry zubních oblouků a modelů jsou dosti specifické a neměnné. Algoritmus i s výslednou rovinou popsanou vektorem a posunem je zapouzdřen ve třídě `ModelPlaneApproximation`.

V úvodu je samozřejmě ještě nutné ověřit správnou orientaci normály, podle které se bude model promítat do výškové mapy. Zde jsem též pokračoval přímo podle návrhu z kapitoly 3. Pro samotnou detekci správnosti orientace postačí analyzovat rozdíly normálových vektorů každého z trojúhelníků modelu a porovnat je normálovým vektorem roviny. Protože počty trojúhelníků orientovaných jedním směrem se liší až v řádech s těmi opačnými, lze podle této indicie bezpečně identifikovat správnou orientaci výsledné normály.

4.7.3 Křivosti a segmentace

První důležitější třídou v modulu trojrozměrné analýzy je `ModelCurvature`. V této třídě jsem využil znalostí z mé bakalářské práce a implementoval rozhraní pro výpočty a správu křivostí pro model. Hlavní novinkou je kompletní zapouzdření všech počítání křivostí. Pomocí identifikátorů v konstruktoru uživatel zajistí výpočet křivosti, která se následně uloží do každé ze čtyř map uchovávajících informace pro pozdější využití.

Pokud chce uživatel přistupovat k jednotlivým křivostem, může využít funkce, která do modelu (hodnota Value u každého vrcholu) uloží hodnotu dané křivosti. Tímto způsobem se výrazně urychlí případné přepínání mezi křivostmi bez nutnosti jejich přepínání.

Se spočítanou křivostí je dále možné provést segmentaci modelu. Ta se na rozdíl od mé předchozí práce zjednodušila. Jedná se hlavně o celé rozhraní a uchovávání regionů, které již není součástí modelu, ale je uloženo a zapouzdřeno ve třídě `ModelRegions`.

Samotný watershed algoritmus probíhá stejně, jako v originále [6], výsledek je však ukládán podobně jako u dvourozměrného segmentačního nástroje do paralelní struktury. Ta přiřazuje každému z vrcholů určitý region. Třída obsahuje i funkce pro spojování regionů s určitými parametry, v mojí finální práci jsem však tyto metody nevyužil. Nicméně nevylučuji jejich budoucí použití, proto je ve třídě ponechávám.

4.7.4 Spojení analýzy výškové mapy a modelu

Posledním případem popisovaným v rámci tohoto modulu je spojení informací získaných ve všech předchozích krocích do jednoho. Pro tento účel zde slouží třída `ModelComposer`, jež

řízení předává všechny dostupné informace, které byly spočítány z obou pohledů, a tato třída označí vhodné regiony na samotném modelu a přiřadí je k jednotlivým zubům.

Nejprve je nutné výškovou mapu předzpracovat a vyznačit na ní regiony korespondující se zuby. To se provádí průchodem výškové mapy a označením oblastí s nejmenší výškou. Prozatím nezáleží na tom, zda na modelu najde zub nebo mezeru. O to se postará následující kontrolní systém, který tyto regiony důkladně porovná s vektorem označených a klasifikovaných zubů a jednotlivým regionům buď přiřadí konkrétní zub, nebo je smaže z dalšího zpracování.

Samotný algoritmus sloučení probíhá dále ve dvou fázích. V první prochází řízení každý vrchol modelu, přičemž ho transformuje do souřadnic výškové mapy a porovná s označenými regiony na ní. K ošetření výškového rozdílu se neuvažuje přímá náležitost daného bodu, ale určitá minimální vzdálenost. Pokud je tento vrchol označen jako člen zubu, ukládá se hodnota trojrozměrného regionu, ke kterému patří, do seznamu pro tento zub.

Druhým a posledním krokem je označení všech vrcholů nalezených regionů korespondujících se zuby. To se provede druhým průchodem modelu. Během toho se vytváří dva seznamy nutné pro budoucí snadnější orientaci na modelu:

- `ModelToTooth`
  - Tento seznam mapuje indexy regionů na trojrozměrném modelu na indexy zubu. Indexem zubu zde myslíme umístění ve vektoru výsledných zubů, nemusí se tedy jednat přímo o pozici. Zuby mohou být ve vektoru uloženy libovolně.
- `ToothToModel`
  - Seznam přiřazuje indexy naopak, ale ne zcela analogicky. Díky tomuto seznamu můžeme při specifikaci indexu zubu extrahovat informace o indexech všech vrcholů k němu patřícím. Tato informace zásadně urychluje hlavně vykreslování, označování a práci se zuby.

4.8 Shrnutí konceptu aplikace

Z celkového konceptu aplikace je patrné, že jsem se snažil o maximální zaměnitelnost jednotlivých komponent. Do budoucna je tedy možné kterýkoli z modulů, případně i jejich součástí vyměnit tak, aby nebyla porušena funkčnost aplikace. Je samozřejmě nutné zachovat formát vstupních a výstupních dat, případně implementovat modul adaptéru, který konverzi zajistí.

Hlavní výhodou tohoto přístupu je i to, že veškeré informace o regionech vzniklé segmentací, ať již dvourozměrnou, nebo trojrozměrnou, se ukládají v oddělených zapouzdřených třídách. Tím není nutné jakkoli zasahovat do stávajících datových struktur a lze tak bez problému využívat již existujících.

5 Shrnutí výsledků

V této kapitole se budu věnovat výsledkům své práce. Popisuji zde konkrétní výsledky výstupu programu na různých datových vstupech, chování navrženého algoritmu apod. Též se budu věnovat analýze funkčnosti jednotlivých částí programu, jejich výstupům, možným vylepšením a zdokonalení do budoucnosti. Celé testování výsledků jsem prováděl na datech zapůjčených společností 3Dim Laboratory.

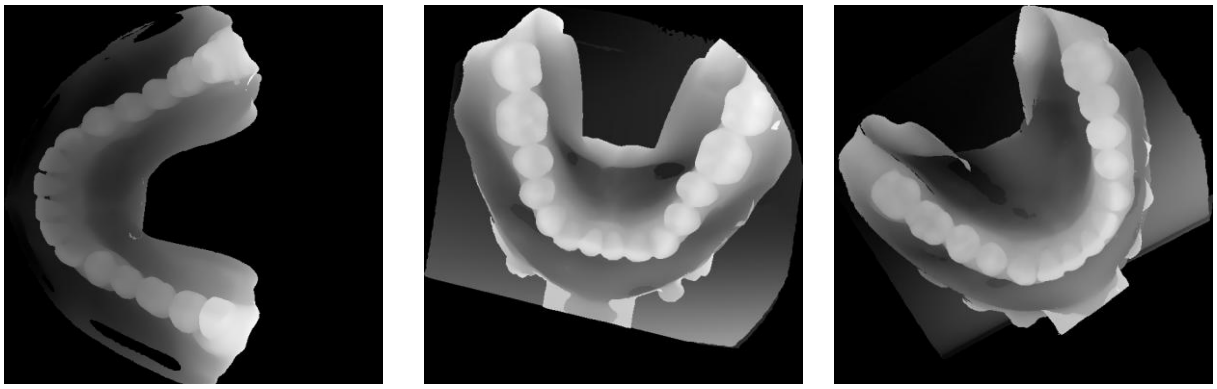
5.1 Aproximace plochy a průmět výškové mapy

V této části programu jsem dosáhl velice uspokojivého výsledku. Plocha aproximovaná metodou nejmenších čtverců ve třech rozměrech na všech zkoumaných modelech plně vyhovovala požadavkům, které jsem na ni v návrhu kladl. To znamená, že ve všech případech aproximovala rovinu zubního oblouku přesně tak, jak jsem očekával. V některých případech samozřejmě docházelo k menším odchylkám, ale rozhodně ne v měřítku takovém, které by mohlo ovlivnit následující zpracování.

Co se týká korekce normály této roviny, tak tento algoritmus bez problému fungoval jak na ořezaných modelech přepočítaných z CT dat, tak na modelech z naskenovaných sádrových otisků. Samozřejmě že se předpokládá model s decimovanými málo zakřivenými plochami. Pokud by tento požadavek vstupní data nesplňovala, mohlo by dojít k případu, kdy bude normála upravena směrem opačným. Celý algoritmus totiž vychází z úvahy, že počet trojúhelníků na straně se zuby je mnohonásobně vyšší než počet trojúhelníků strany opačné. K problémům by tedy došlo, pokud by model vznikl např. pravidelným rozdělením trojúhelníků. Toto však je pouhá úvaha, tento postup, jak již jsem zmínil, pracoval na celé datové sadě bez problémů.

S řádně spočítanou rovinou není problém model promítnout na výškovou mapu. To, díky zapůjčené knihovně, proběhlo též přesně podle plánu a výšková mapa zcela koresponduje s modelem. Díky spočítané transformační matici mezi modelem a hodnotami na výškové mapě je možné libovolně procházet její hodnoty a přiřazovat je zpět ke konkrétním vrcholům.

Na následujících obrázcích jsou patrné detekované mapy několika modelů z testovací sady. Všechny byly pořízeny přímo v aplikaci ihned po automatické detekci roviny. Jedná se tedy o mezivýsledek finálního algoritmu:



Obr. 5.1.1 – Ukázka pořízených výškových map z testovací datové sady

Na těchto obrázcích si dovoluji upozornit na vznikající artefakty stejných výškových hodnot vzniklých nepřesným převedením CT modelu otisku na model čelisti. Jedná se o druhou a třetí ukázkovou výškovou mapu, kde jsou před zuby patrné výstupky dosahující podobných hodnot. To tvořilo v dalším zpracování nemalé problémy.

5.1.1 Možné budoucí úpravy

V této části nepředpokládám nějaké zásadní změny ve funkčnosti, protože chování algoritmu koresponduje s předpokládanými výsledky. Jediný rozdíl by mohl tvořit postup při korekci normály plochy, ale to je možné pouze s existencí daleko rozsáhlejší vstupní datové sady, která ověří nebo vyvrátí funkčnost této metody.

5.2 Detekce zubní křivky

V této části jsem dosáhl víceméně kladných výsledků, stále však přetrvává prostor pro silné vylepšení tohoto postupu. V následujících odstavcích popíši výsledky v dané problematice.

Detekce zubní křivky se ukázala být nejdůležitějším, ale také nejtěžším problémem v celé mé práci, protože se podle ní odvíjí úspěšnost samotného klasifikačního algoritmu. Samotnou detekci ovlivňují požadavky na křivku, která musí mít následující atributy:

- Křivka musí protínat všechny zuby, nebo se k nim alespoň blížit.
- Kvůli zvolenému klasifikačnímu algoritmu musí být křivka symetrická. To znamená, že musí začínat i končit ve stejné části čelisti.

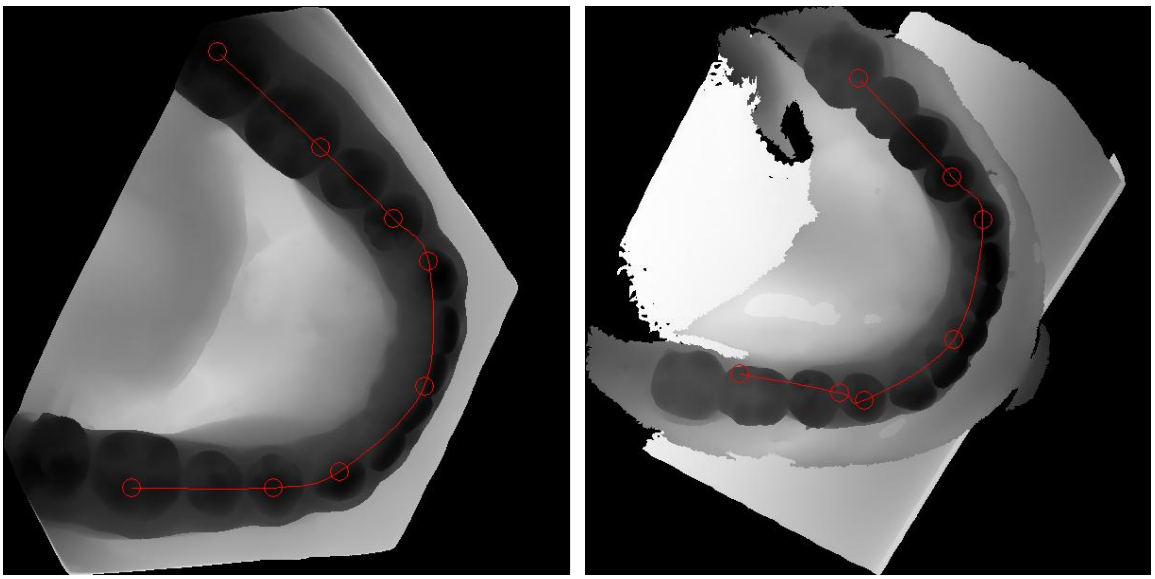
Tyto požadavky tvořily hlavní problémy celé detekce a zatím se je nepodařilo úspěšně vyřešit, ať již z důvodu existence velkého množství artefaktů na některých modelech, tak přílišnou odlišností jednotlivých zubů a požadavku na symetričnost.

V testovaných algoritmech jsem zkoumal vícero přístupů pro vyřešení této problematiky, finálně implementovaná metoda však vykazovala nejlepší výsledky. Mezi testy byly postupy využívající informace o výšce, jako například watershed segmentace aj., jež měly filtrovat regiony,

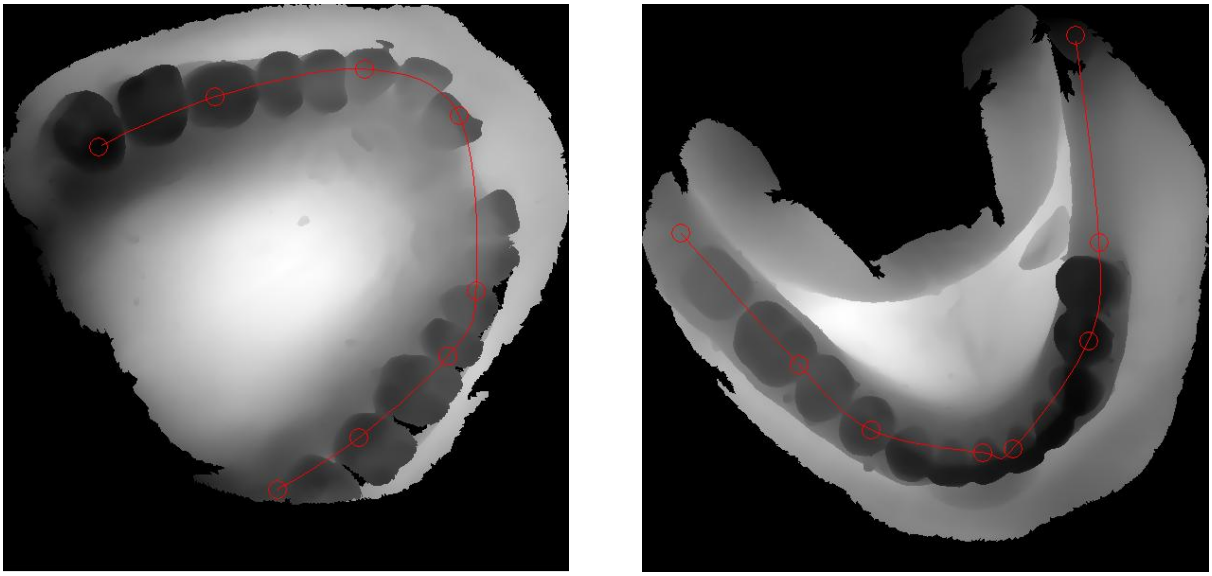
kterými nejsou zuby. Tento postup však nebyl ideální, protože dostatečně neomezoval počet špatně detekovaných regionů. Stejně tak čistá metoda RANSAC nedosáhla kýžených výsledků, protože kvůli vysokému číslu předzpracovaných regionů nebylo možné v přijatelném čase dostát správně orientované křivce. Tato metoda navíc neřešila symetričnost.

Finálně zvolená hybridní metoda se snaží eliminovat oba dva hlavní omezující faktory. První část algoritmu najde přibližnou pozici pevně dané šablony zubního oblouku, čímž se zajistí symetričnost výsledné křivky. Druhá část poté pomocí A* algoritmu regiony seřadí a RANSAC vybere nejvhodnější kandidáty pro zubní regiony. I tak ale metoda vykazuje nedostatky, pramenící ve většině případů právě ze špatného předzpracování modelu, kde i nadále zůstávají artefakty. A* hledání cesty započtení artefaktů značně snižuje (artefakty jsou ve většině případů vně zubní křivky), nicméně je neeliminuje zcela.

I po automatickém nalezení křivky je tedy nutné požádat vždy uživatele o manuální korekci. Podle mého názoru tento krok nijak nesnižuje komfort uživatele, protože se jedná o jednoduchou akci trvající uživateli seznámenému s aplikací několik sekund. Za výsledek pak lze považovat zcela správně aproximovanou křivku zubním obloukem. Na následujících několika obrázcích je zachycen stav automatické detekce křivky na testovacích modelech (pohled z okna manuální korekce před zásahem uživatele). Kružnice na křivce reprezentují řídící body umožňující uživateli korekci.



Obr. 5.2.1 – Ukázka automatické detekce zubní křivky – 1. sada příkladů



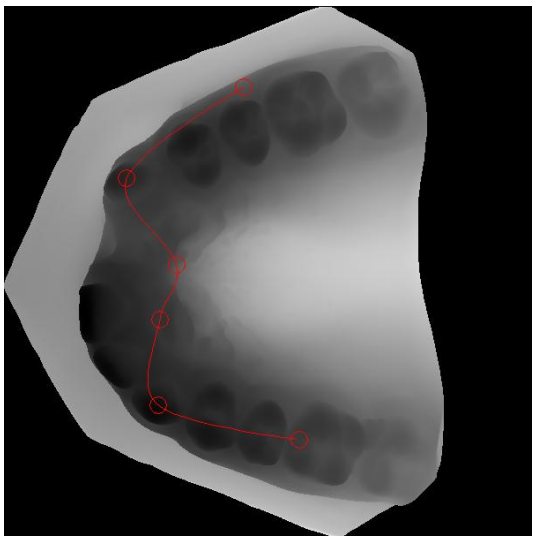
Obr. 5.2.2 – Ukázka automatické detekce zubní křivky – 2. sada příkladů

Jak je patrné z těchto příkladů, algoritmus má zajisté nevýhody. Zcela jistě není dodržena požadovaná symetrie křivky, což je dáno nepřesným umístěním prvního odhadu.

Dalším faktem, který lze vyčíst z ukázek je, že křivka nekopíruje úplně přesně zubní oblouk, což ale v mém případě detekce není zásadní problém. Detekce se totiž děje v dost širokém okolí křivky, takže stačí, aby se aproximovaná křivka k zubům alespoň blížila.

Třetím problémem, který zde není patrný, nicméně existuje a projevuje se v nezanedbatelné míře, je vliv stochastického algoritmu na opakovatelnost detekce. Výsledky jsou samozřejmě velice podobné, z důvodu stochastického algoritmu však ne stejné.

Často též dochází k případům, kdy algoritmus selže úplně. Nebo tedy nedokáže zubní oblouk zcela dobře vystihnout. Zásadní problém v takovém případě vzniká kombinací špatného původního odhadu křivky spolu s neúčinnou filtrací zubních regionů. Takovéto selhání je patrné na následující výškové mapě:



Obr. 5.2.3 – Chyba v aproximaci křivky

5.2.1 Možné budoucí úpravy

Celý algoritmus hledání křivky na modelu by mohl být zajisté předmětem dalšího šetření. Zcela jistě nedosahuje výsledků takových, jaké by se čekaly od správně fungujícího postupu. Důvodem je velká rozmanitost vstupních dat.

Jako první krok bych rozhodně doporučil přesněji filtrovat regiony zubů. Zde se nachází klíč k úspěšné aproximaci křivky, protože v současné době přetrvává stále velké množství regionů špatně označených. Avšak je nutné k tomuto problému přistoupit lokálně, tzn. určitě nepoužívat pravidla pro filtraci celého obrazu. Zuby jsou jen zřídka v jedné rovině (nezapomeňme, jedná se o pacienty stomatologie), a proto např. globální filtrace podle výšky regionů nepřipadá v úvahu.

S přesněji filtrovanými regiony by se podle mého názoru značně zvýšila přesnost celé detekce. Samozřejmě drobné úpravy bych doporučil i do podmínek tvorby RANSAC křivky, kterou by bylo vhodné omezit například počtem řídících bodů, větší symetrií apod.

5.3 Křivosti a segmentace

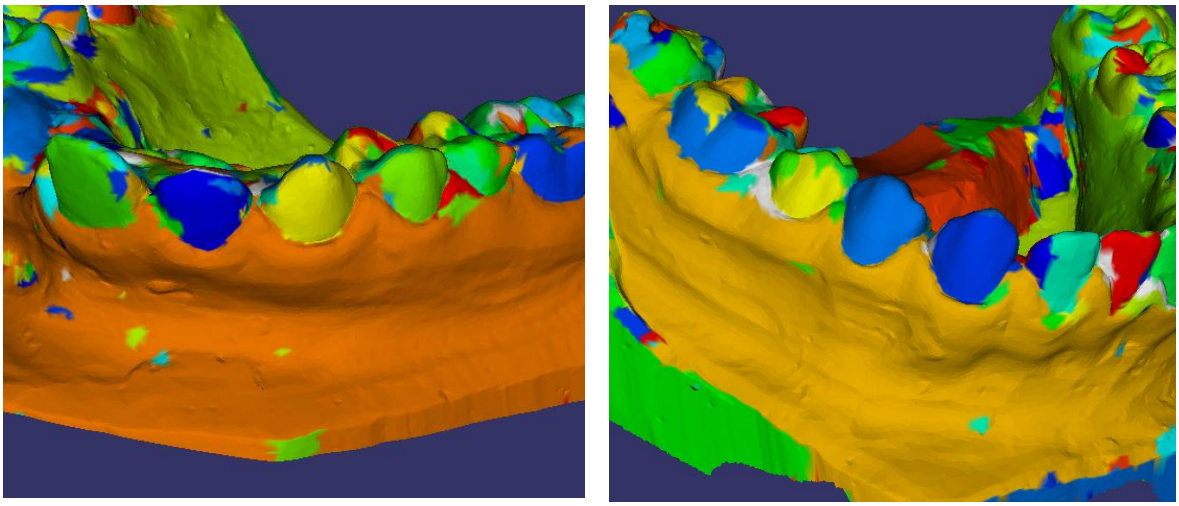
K tomuto tématu se vztahuje moje bývalá bakalářská práce [6], takže výsledky zde zmíním pouze okrajově. Samotné počítání křivosti jsem převzal zcela z této práce, protože jeho funkčnost byla bezproblémová.

Co se týká samotné upravené Watershed segmentace, zde jsem pár změn provedl, jedná se hlavně o zpožděné vytváření nových regionů tak, aby nedocházelo k přílišné segmentaci modelu. Tento přístup se osvědčil a na testovací sadě jsem ověřil jeho funkčnost.

Zde se však objevil jeden zásadní problém s novými modely. Trojúhelníkové reprezentace sádrových odlitků, které vznikly naskenováním reálních modelů, reagovaly na tento přístup bez sebemenšího problému. Zuby jsou zde dosti přesně oddělené mezerou, která je zachycena na modelu. Z tohoto důvodu není problém pro watershed algoritmus zaplňovat regiony a tvořit hranice právě v těchto místech.

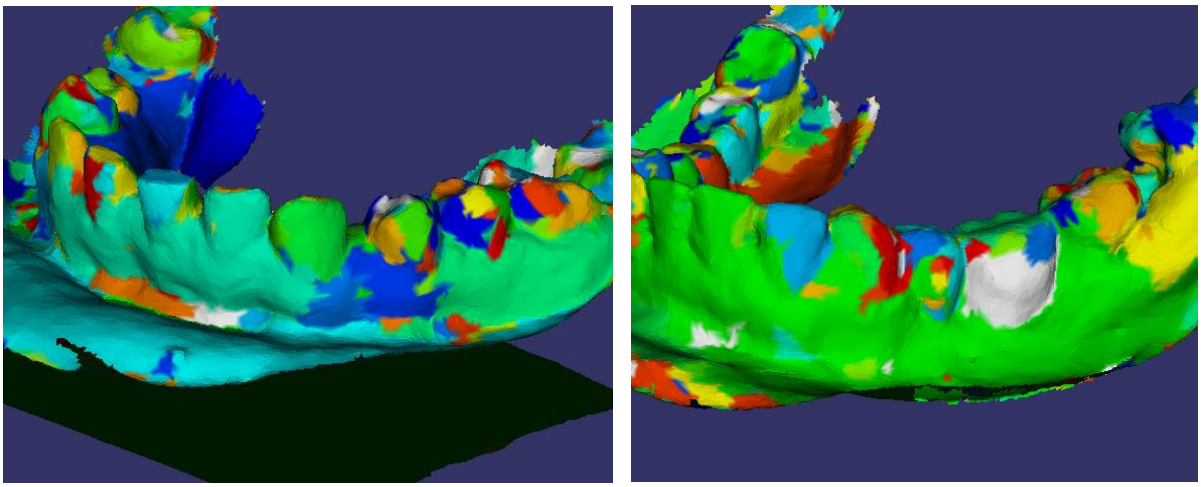
Na nové datové sadě však vznikl problém. Modely jsou přesností řádově nižší než modely sádrových odlitků. To lze dát za vinu několikanásobné konverzi modelu, nejprve z převodu CT řezů na volumetrická data, následně pak z objemového modelu na model trojúhelníkový. Poslední zmíněnou verzi jsem vyloučil, protože jsem sám testoval převod z objemové reprezentace na hraniční a ani při vypnutém vyhlazování zde nedošlo ke znatelnému zvýraznění oddělení zubů a dásní. Tento problém lze přisoudit přesnosti dentální počítačové tomografie, jež se pohybuje okolo 0,2 mm. S přesností lze tedy po konverzi rozlišovat oblasti větší než 0,4 mm, což by teoreticky mohlo ohrozit detekci hranice zub-dáseň. Přesnější řešení jsem v mé práci nezkoumal, protože se jedná o rozsáhlejší problematiku. Na výsledném modelu totiž není žádný topologický znak, který by dokázal z lokálního hlediska odlišit zub od dásně.

Na následujících obrázcích demonstruji rozdíly ve dvou různých datových sadách. Na obrázku 5.3.1 jsou zobrazeny segmentované modely naskenované ze sádrových otisků. Je patrné přesné oddělení zubů jak vůči sobě, tak vůči zbytku modelu. Různé barvy oddělují různé regiony (více regionů může mít stejnou barvu, jedná se pouze o vizualizaci).



Obr. 5.3.1 – Správné segmentování detailního modelu

Na druhé datové sadě (5.3.2) vzniklé segmentací dat z CT volumetrického modelu jsou patrné regiony přecházející ze zubů do dásně, případně nepřesně označené hranice mezi zuby. To je dáno absencí detekovatelného oddělení těchto regionů na modelu.



Obr. 5.3.2 – Absence oddělení zubních regionů od zbytku modelu – malé rozlišení modelu

5.3.1 Možné budoucí úpravy

Zde je patrné, že pro vstupní data pořízená skenem s vysokým rozlišením (alespoň takovým, aby byly zachyceny objekty pro oddělení zubů) tato metoda pracuje bez sebemenších problémů.

Pro modely s nižším rozlišením je však nutné implementovat vhodný korekční prvek, který bude brát tuto skutečnost v úvahu.

Jako návrh budoucího vylepšení bych tedy zmínil adaptivní segmentaci, která by upravovala počet segmentů na modelu. Bohužel, ani zde si nejsem jistý, zda by tento postup řešil tento problém, jelikož v určitých místech je křivost na modelu téměř nulová – nelze tedy toto ohraničení označit jako hranici. V úvahu by proto připadal i jiný postup, například doplňování hranic z okolí, ale testováním takových postupů jsem se během řešení této práce z časových důvodů příliš nezabýval. Zajisté by ale bylo vhodné tento problém dále zkoumat.

5.4 Klasifikace zubů

V samotném klasifikačním postupu jsem též zkoušel větší množství postupů, tak jak jsem je popisoval v kapitole 4.6.2.1. Každé z řešení vykazovalo určité výsledky. Zde však nepovažuji za nutné podrobně rozebírat každý z postupů, který jsem testoval. Podrobně tedy popíši výsledky zvoleného klasifikačního algoritmu, který jsem zvolil jako finální. Na všech datových sadách tato metoda představila uspokojivé výsledky, které zde podrobněji popíši.

Na přesnějších modelech tento algoritmus funguje bez závad. Rozdíly mezi zuby jsou patrné a dobře rozpoznatelné, proto není problém mezi nimi detekovat hranice zubů, které oddělují. Stejně tak korekční algoritmy tuto detekci ještě zpřesní. Výsledkem tohoto postupu jsou kvalitně ohraničené zuby na aproximované křivce.

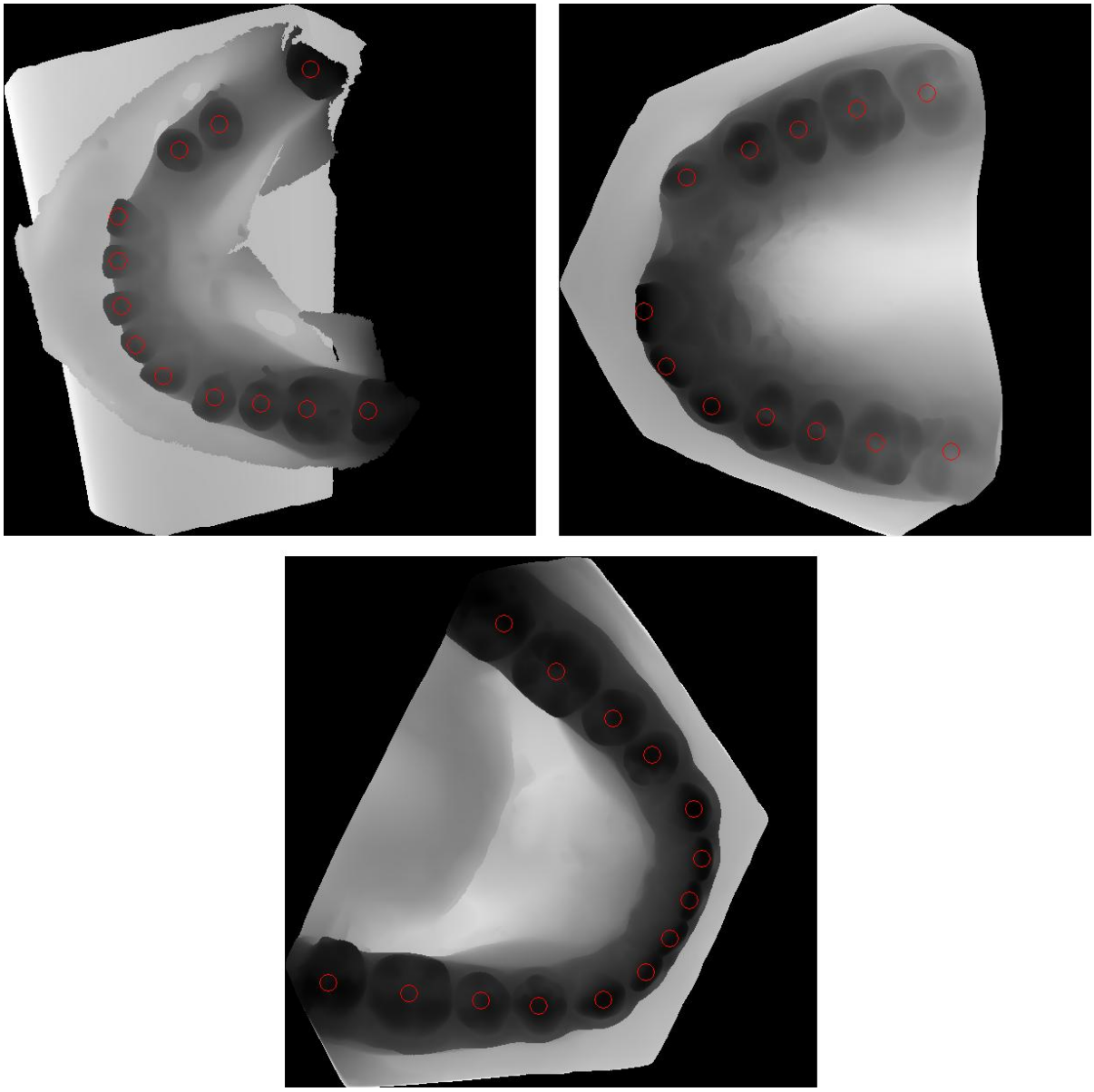
Co se týká modelů s nižším rozlišením, zde občas dochází ke splynutí dvou a více zubů, případně nepřesného oddělení takto detekovaných kandidátů. Tento problém je patrný například na řezácích, které často splývají do jednoho segmentu. Tyto chyby většinou filtruje navržená segmentace, nicméně zde bych viděl možná vylepšení do budoucnosti.

Stejný výsledek lze přiřadit i metodě klasifikace. Ta probíhá bez problému za předpokladu přesného usazení zubní křivky. Je patrné, že pokud křivka není symetrická, tak pozice zubů souhlasit nebudou. Tento fakt je samozřejmě uživatel schopný ovlivnit, k nesprávné klasifikaci může též dojít při reálném posunu zubu do mezery po extrakci. Hranice a segmentace zubů je samozřejmě díky segmentačnímu nástroji průmětu na křivku neporušena.

Během řešení mé práce jsem se mnohokrát zabýval též klasifikací podle tvaru zubu, nicméně zde jsem uspokojivých výsledků nedosáhl. Zuby je sice možné rozlišit podle základního tvaru, ale přesnější určení je vždy nutné podle pozice (např. v průmětu nelze rozlišit řezáky a špičáky, malý rozdíl je též mezi třenovými zuby a stoličkami). I když jsem dokázal podle určitých charakteristik odlišit přední skupinu zubů od zadních, vždy tu vyvstane problém s chybějícími zuby. Nelze přesně určit, zda první stoličky detekované na křivce hned vedle zubů třenových jsou opravdu první stoličky (molaris primus), nebo jestli zde určitý zub chybí. Tato znalost bohužel klade požadavky

na inteligentní rozhodování, jehož důkladné prozkoumání nebylo v časovém rozmezí pro řešení této práce možné.

I přes tyto nedostatky je podle mého názoru klasifikační algoritmus funkční a zcela jistě pro modely s vyšším rozlišením dostatečný. Ve většině případů dokonce více než dostatečný, protože úspěšně filtruje chybějící zuby a mezery, při kterých měly ostatní zkoumané postupy nemalé problémy. Na následujících obrázcích jsou patrné rozpoznané a klasifikované zuby na zubním oblouku (kružnice označují detekované a klasifikované středy zubů). Obrázky nezachycují výstup klasifikace, tedy přiřazené názvy zubů, ale výstup je shodný s výstupem v kapitole následující, kde jsou tyto segmenty řádně popsány.



Obr. 5.4.1 – Detekované a klasifikované středy zubů po automatické klasifikaci s manuální korekcí aproximované křivky

Středy zubů se počítají jako průměr souřadnic detekovaných zubů spolu s doplněním informací o regionech k nim náležícím. Spojí se tedy tak informace s průmětu křivky a výškové mapy.

5.4.1 Možné budoucí úpravy

Jako hlavní vylepšení klasifikace předpokládám přidání klasifikace podle vzhledu zubů. To by mohlo značně usnadnit a zpřesnit stávající metodu. V zásadě by stačilo doplnit informace na průmětu po křivce o trojrozměrné informace.

Druhou možností by bylo model transformovat celý do průmětu na křivku ve 3D a klasifikovat ho podle podobnosti ve třech rozměrech. Tato možnost tu je též a podle mého názoru by mohla dostatečně ovlivnit přesnost celého detekčního mechanizmu. Bohužel však ani tato metoda nedokáže rozlišit všechny zuby, protože stále existují například řezáky a stoličky, které jsou si navzájem příliš podobné. Proto určitý klasifikační mechanizmus zahrnující do kontextu pozici na zubní křivce bude muset být vždy brán v úvahu.

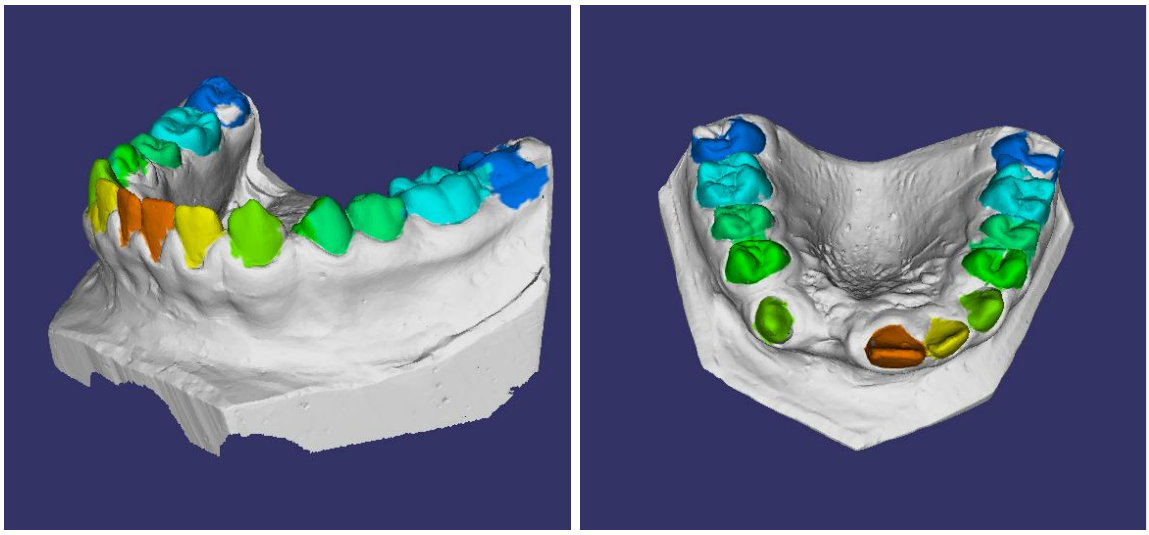
5.5 Celkový koncept klasifikace zubů

V této kapitole zhodnotím výsledky práce jako celku. Po klasifikaci jednotlivých zubů a jejich segmentaci je nutné všechny znalosti spojit. Zde se, díky přesné segmentaci na trojrozměrném modelu, podařilo dosáhnout též uspokojivých výsledků. Je však nutné znovu rozdělit tuto kapitolu na dvě části, na modely s vysokým rozlišením a na modely s nízkým rozlišením.

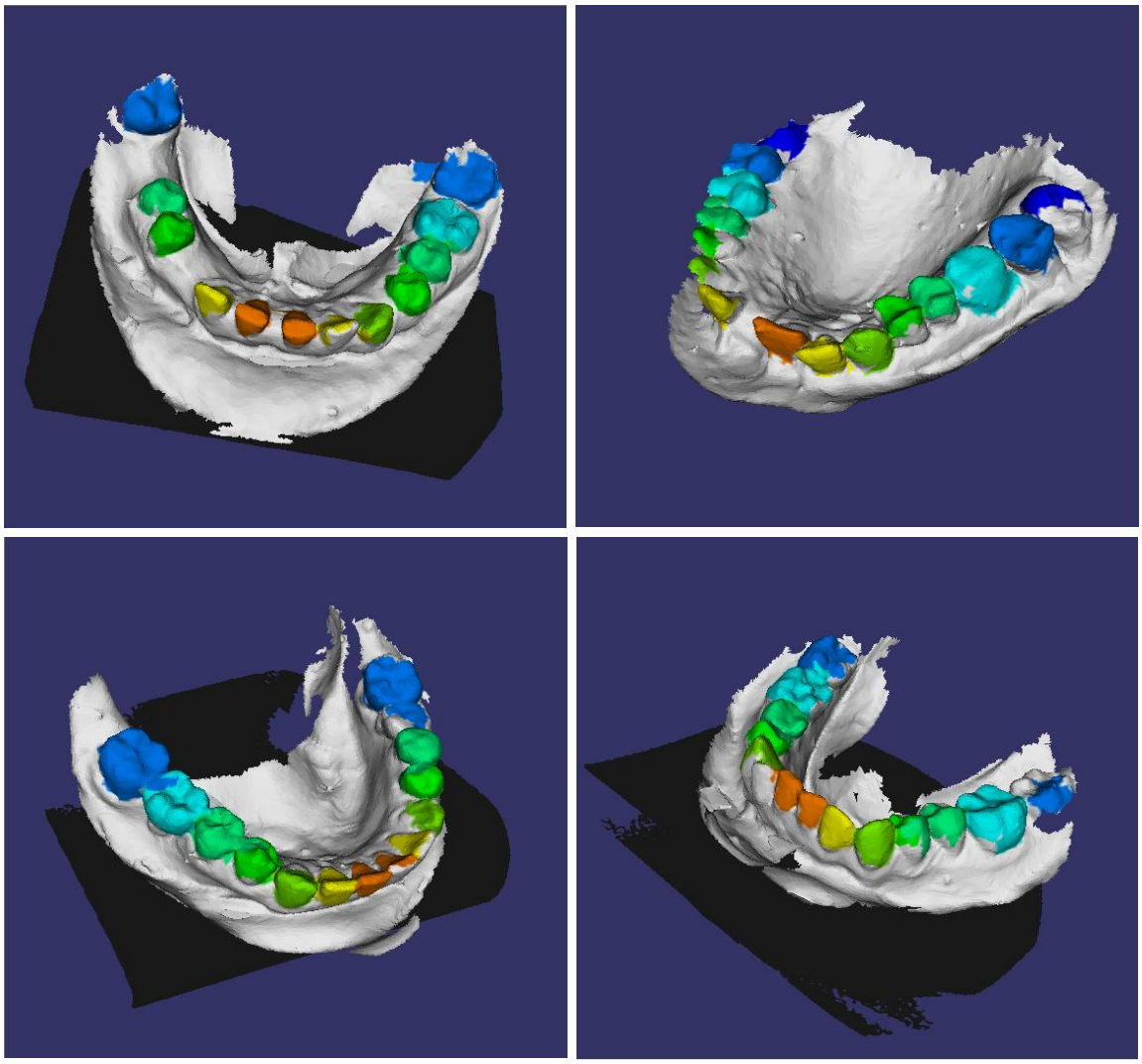
Pokud se jedná o přesné modely s vysokým rozlišením, samotné označení klasifikovaných regionů probíhá úspěšně. Až na určité nepřesnosti vyplývající s příliš natěsno sousedících zubů se daří regiony náležící určitým třídám zubů přiřazovat. Po aplikaci algoritmu, který doplní chybějící regiony uvnitř zubu, lze označit tento algoritmus za úspěšně fungující, a tedy celý koncept klasifikace zubů za úspěšně fungující.

V případě modelů s nižším rozlišením se bez problému daří označovat zadní zuby, kdy je hranice přesněji definována. Výsledek je poté shodný s přesnějšími modely. Problém nastává hlavně u předních řezáků, kdy nejsou patrné hranice mezi zuby. Zde dochází často k neoznačení, nebo naopak ke špatné detekci zubního segmentu. Bohužel se stávajícím postupem nelze tento fenomén nijak obejít, vše závisí na správné segmentaci trojrozměrného modelu.

V zásadě však mohu celý navržený postup označit za funkční, a to i z důvodu, že s malými úpravami se ho podařilo naplnit včetně mnoha malých vylepšení pozitivně působících na funkčnost. Toto téma není rozhodně uzavřenou záležitostí, stále je možnost upravovat mnoho algoritmů představených zde tak, aby dosáhly mnohem kvalitnějších výsledků. Na konci této kapitoly prezentuji několik automaticky segmentovaných modelů.



Obr. 5.5.1 – Klasifikované zuby modelů s vysokým rozlišením (legenda viz níže)



Obr. 5.5.2 – Klasifikované zuby modelů s nízkým rozlišením (legenda viz níže)

Legenda ke klasifikovaným zubům

Jak obrázky dokládají, na modelech s vysokým rozlišením lze považovat klasifikaci a označení zubů za dostačující. U ostatních modelů je patrné, že na některých zubech označení chybí, na některých přebývá apod., a to z důvodu nepřesností v segmentaci modelu. Zde by bylo nutné implementovat algoritmus na např. manuální segmentaci, případně korekci segmentace.

5.5.1 Možné budoucí úpravy

Samotné označování regionů lze v budoucnu zajisté vylepšovat, například porovnáváním zubu s určitou šablonou nebo kontrolou křivosti hranice. U druhého zmiňovaného si však nejsem jistý, nakolik by tato metoda byla použitelná například u stoliček, které mají příliš vysokou křivost po celém povrchu. Zde by bylo možné docílit mnohem spojitější označené struktury.

Jako hlavní vylepšení by však bylo vhodné implementovat algoritmus pro manuální korekce segmentace, která by umožnila uživateli model vhodně rozdělit i v místech, kde jsou regiony špatně značené. To by zajisté umožnilo přesnou detekci zubů i na méně přesných modelech, kde je automatický detekční algoritmus příliš slabý.

6 Závěr

V této práci jsem postupně popsal návrh a implementaci algoritmu pro klasifikaci zubů na trojrozměrném modelu čelisti. Během implementace jsem musel řešit několik změn návrhu, způsobených převážně nepřesnostmi na modelech a větší nutností filtrace artefaktů. Hlavní problém se následně ukázal v odlišném rozlišení jednotlivých modelů, kdy u těch méně přesných bylo velice obtížné přesněji ohraničit zuby tak, aby je mohl následně klasifikační algoritmus detekovat. I přes tyto obtíže umožňuje výsledný postup automatickou klasifikaci zubů na čelisti s minimálním zásahem uživatele, který spočívá v manuální korekci detekované zubní křivky.

Uživatel má též možnost upravit automaticky detekované regiony, pokud dojde k nežádoucímu označení zubů. Aplikace dále umožňuje rozdělení modelů a postupné uložení jednotlivých modelů zubů do samostatných trojúhelníkových modelů, případně jejich posunutí a plánování korekcí přímo v aplikaci. Vzniklé modely lze poté libovolně ukládat a nahrávat.

Celý navržený algoritmus je však v prvé řadě studií přístupu k této problematice. Před samotným zavedením do praxe je nutné analyzovat řadu změn a upřesnění jednotlivých postupů, například tak, jak navrhuji v kapitole 5. Tato práce sloužila tedy hlavně k identifikování těchto problémů, jejich analýze a studii možných budoucích řešení.

V této analýze vidím největší přínos práce, protože samotný zde implementovaný postup je nutné nejprve řádně validovat a upravit, než by byl případně uveden do praxe.

Co se týká implementačních detailů, celou aplikaci jsem navrhl tak, aby takovéto změny dovolovala, tedy přístup k ní je zcela modulární a umožňuje záměnu kterékoli funkční třídy za jinou, případně upravenou. Tímto lze ve vývoji tohoto softwaru bez problému pokračovat.

K vývoji jsem využíval MS Visual Studio 2008 a jazyk C++, díky kterému jsem vytvářel aplikaci, která je teoreticky přenositelná na jiné platformy. Samozřejmě vývoj probíhal na platformě Windows, pokud tedy bude požadováno spuštění tohoto softwaru na jiných platformách, bude nutné provést drobné změny v kódu. Kompatibilita nebyla testována.

Literatura

[1] Kršek, Přemysl, Doc., Ing., Ph.D.: přednášky předmětu IZG: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IZG-IT/lectures, 18.12.2009

[2] Wikipedia, Constructive solid geometry: http://en.wikipedia.org/wiki/Constructive_solid_geometry, 18.12.2009

[3] Medcyclopaedia, Hounsfield unit: http://www.medcyclopaedia.com/library/topics/volume_i/h/hounsfield_unit.aspx, 18.12.2009

[4] Zobrazení 3D modelů v DFX pomocí OpenGL – zdroj obrázku: http://www.root.cz/clanky/zobrazeni-3d-modelu-v-dxf-pomoci-opengl/, 19.12.2009

[5] The winged edge data structure, Michigan Tech. University: http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/winged-e.html, 19.12.2009

[6] Hulík, Rostislav: Detekce zubů na 3D počítačovém polygonálním modelu čelisti, Bakalářská práce, VUT FIT 2008

[7] Image Processing and Analysis Group, Meusnier theorem seminar: http://noodle.med.yale.edu/seminar/shi/lecture4.pdf, 19.12.2009

[8] Wikipedia, Základní forma plochy: http://cs.wikipedia.org/wiki/Z%C3%A1kladn%C3%AD_veli%C4%8Dina_plochy, 19.12.2009

[9] Herout, Adam, Ing, Ph.D.: přednášky předmětu PGR: https://www.fit.vutbr.cz/study/courses/PGR/private/, 19.12.2009

[10] McAllister, Keegan: Triangle rasterizarion, October 23, 2007 http://www.cs.caltech.edu/courses/cs171/barycentric.pdf, 19.12.2009

[11] Španěl, Michal, Ing., přednášky předmětu POV, segmentace obrazu: https://www.fit.vutbr.cz/study/courses/index.php?id=6922, 19.12.2009

[12] Image segmentation and mathematical morphology, Centre de Morphologie Mathématique, Paris: http://cmm.ensmp.fr/~beucher/wtshed.html, 19.12.2009

[13] Medical Data Segmentation Toolkit, http://mdstk.sourceforge.net/, 19.12.2009

[14] Open Scene Graph, http://www.openscenegraph.org/, 19.12.2009

[15] wxWidgets, http://www.wxwidgets.org/, 19.12.2009

[16] Janoušek, Vladimír, Doc., Ing., Ph.D.: Přednášky předmětu SIN – Fuzzy logika a fuzzy řízení: http://perchta.fit.vutbr.cz:8000/vyuka-sin, 20.12.2009

[17] Sun-Jeong, Kim – Chang-Hun, Kim – Levin, David: Surface simplification using a discrete curvature norm: http://kucg.korea.ac.kr/~sjkim/paper/cag2002.pdf, 5.dubna 2008

[18] 3Dim Laboratory s.r.o.: http://www.3dim-laboratory.cz/, 10. května.2010

[19] DialogBlocks, Cross platform GUI programming assistant, http://www.dialogblocks.com/, 11. Května 2010

[20] A Islam, Suhail: Math notes http://www.bmm.icnet.uk/people/suhail/plane.html, 12. Května 2010

Seznam příloh

Příloha 1. Manuál

Příloha 2. CD obsahující:

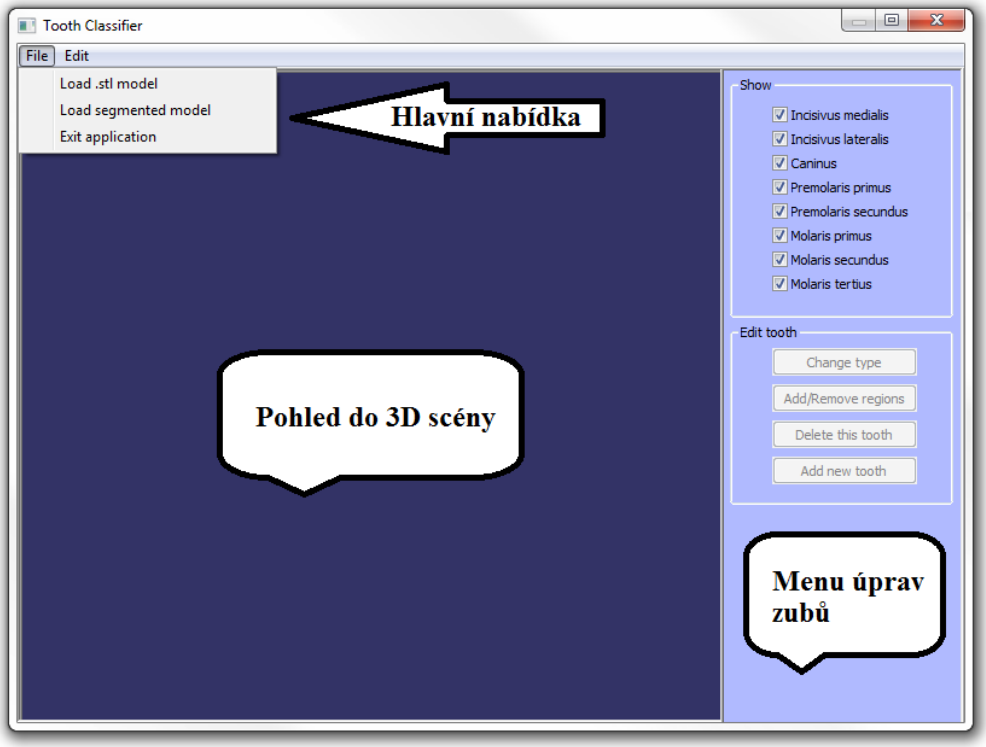
- Tuto dokumentaci: **dokumentace.pdf**
- Programovou dokumentace: **programova_dokumentace.chm** (doxygen)
- zdrojové soubory spolu s projektem pro MS VS 2008
- přeložený spustitelný program (včetně ukázkových otisků).
- soubor README s instrukcemi pro překlad a instalaci

Příloha 1: Návod k obsluze aplikace

Samotná aplikace obsahuje několik funkcí, jejichž ovládání zde popisuji.

Spuštění aplikace a klasifikace zubů

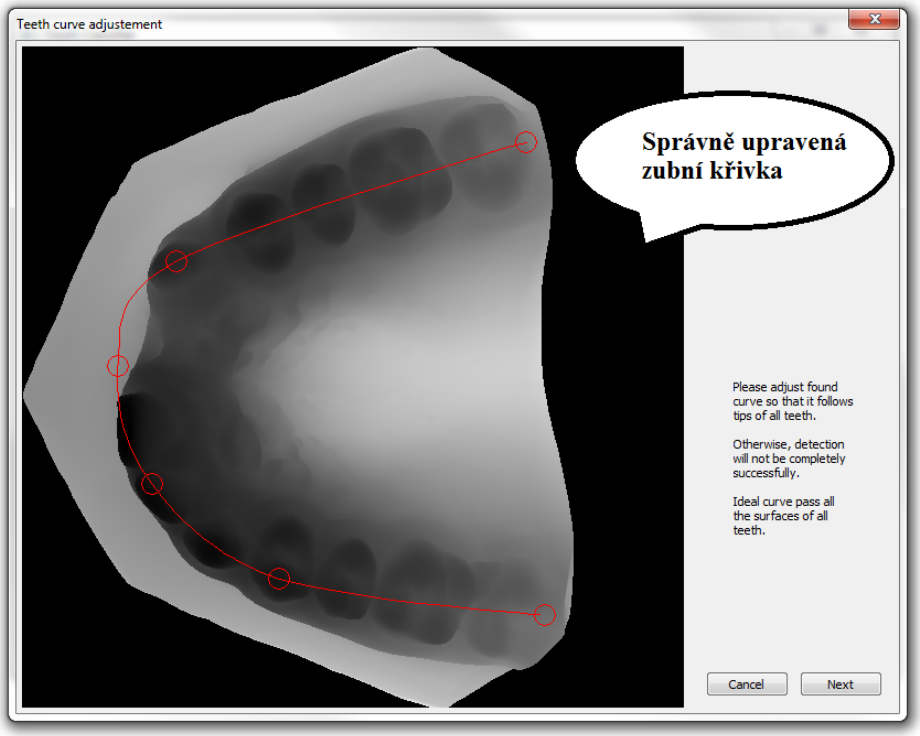
Po samotném spuštění aplikace je uživateli nabídnuto hlavní okno aplikace obsahující ve své levé části pohled do trojrozměrné scény, v pravé poté ovládací prvky zubů (v této chvíli nepřístupné).



V hlavním menu se nachází jak možnost aplikaci ukončit (Upozornění! Aplikace nezobrazuje dialog pro ujištění, zda nedošlo k náhodnému kliknutí na tuto možnost.), tak nahrát .stl model obsahující zubní oblouk.

Možnost **Load segmented model** umožňuje uživateli načíst již zpracovaný model zubů spolu s naplánovanou korekcí. Model se musí nacházet v adresáři spolu s jednotlivými zuby tak, jak ho aplikace uložila.

Po zvolení možnosti **Load .stl model** se automaticky spustí klasifikační algoritmus, je tedy možné, že aplikace nebude po nějaký čas odpovídat (řádově jednotky sekund). Následně je uživatel dotázán na úpravu zubní křivky:



Zde je nutné křivku usadit tak, aby kopírovala alespoň přibližně tvar pacientovy čelisti. Ve většině případů proběhne detekce v pořádku, nicméně v detailech bude nutné křivku upravit. Ve výsledku by měla být křivka podobná té na obrázku.

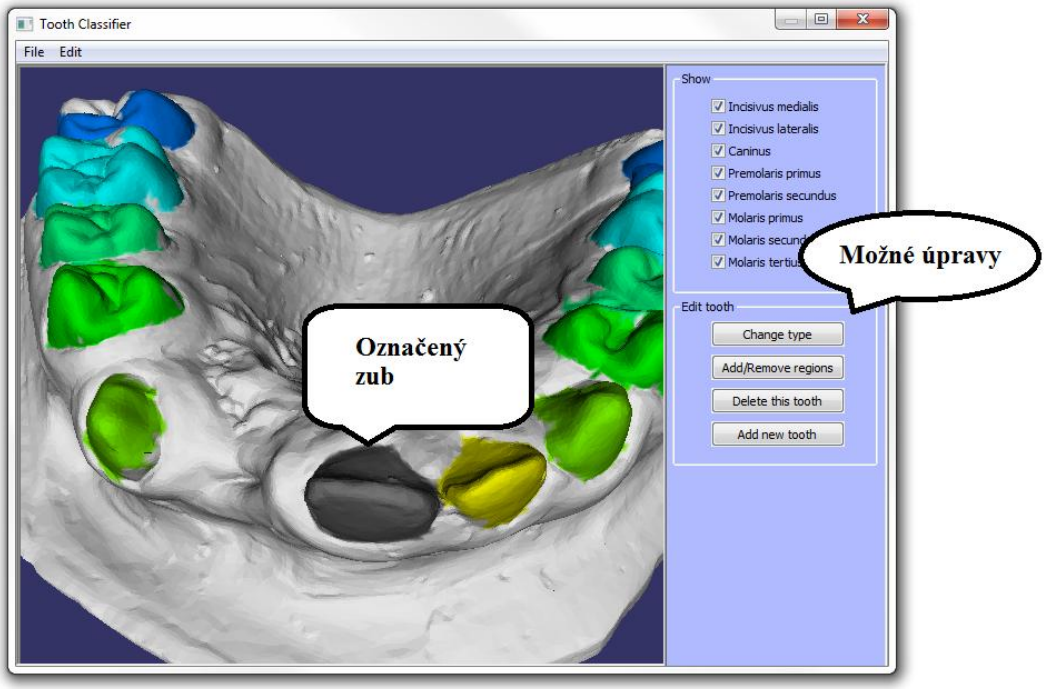
Pozn.: Pokud pacientovi chybí zuby na okrajích zubního oblouku, je nutné tuto křivku vyznačit i v těchto místech, tzn. křivka musí procházet po celém tvaru čelisti a MUSÍ být symetrická (musí končit ve stejných místech).

Po úspěšné úpravě křivky se stiskem tlačítka **Next** dokončí klasifikace.

Úprava klasifikovaných zubů

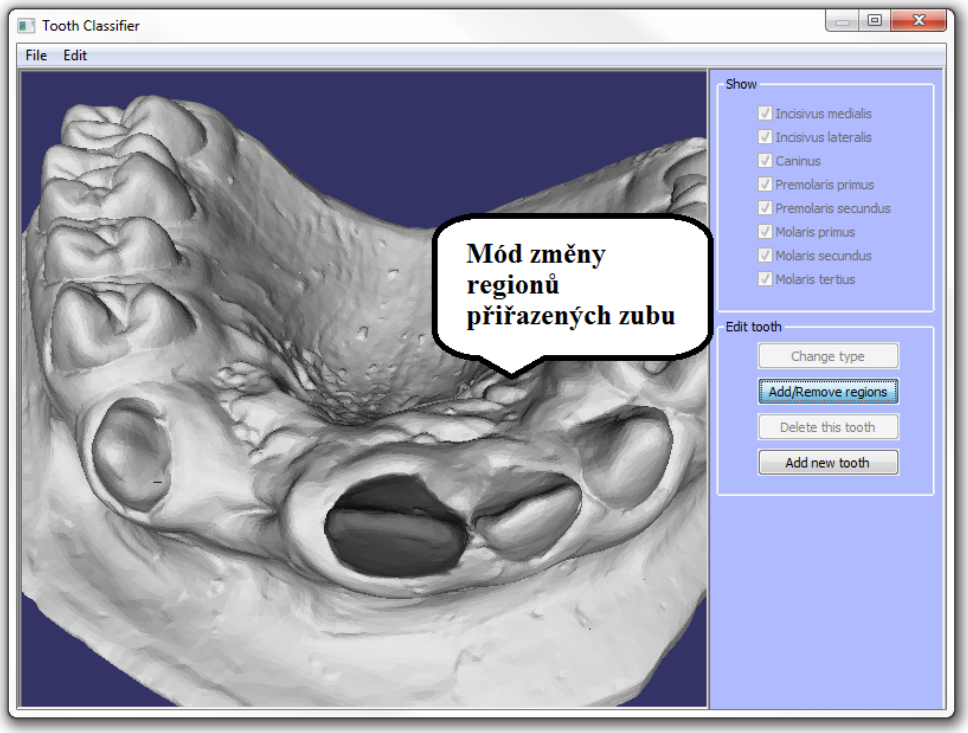
Po dokončení klasifikace se řízení aplikace vrátí zpět na úvodní obrazovku, nyní však již se zobrazeným modelem s vyznačenými zuby. V pravém sloupci menu lze jednotlivé zuby vypínat (nyní pouze označení), nebo stiskem **pravého tlačítka myši** označovat, čímž se zpřístupní nabídka úprav:

1. U označeného zubu lze změnit jemu přiřazený typ – k tomu slouží první možnost v nabídce. Uživatel je dotázán jednoduchým dialogem na změnu typu zubu.
2. Je možné též měnit regiony segmentovaného modelu, jež jsou zubu přiřazeny. To vyvolá speciální zobrazení popsané níže.
3. Samotný zub lze samozřejmě též zcela smazat.
4. Lze též založit zub nový, přičemž se řízení aplikace automaticky přepne do režimu úpravy nově vytvořeného zubu (bod 2).



Ad 2.:

Při stisku tlačítka **Add/Remove regions** nebo po přidání zubu nového se aplikace přepne do módu pro editaci regionů zubu. Ta se vyznačuje **odznačením** ostatních zubů při zachování označení zubu zvoleného. Nyní je možné **pravým tlačítkem myši** přidávat, případně odebírat regiony, které byly předem automaticky segmentovány na modelu. POZOR! Přidávat lze pouze regiony, které **nejsou** součástí jiného zubu. Pokud tento region chceme mít v editovaném zubu, je nutné nejprve upravit zub, ke kterému region přísluší.



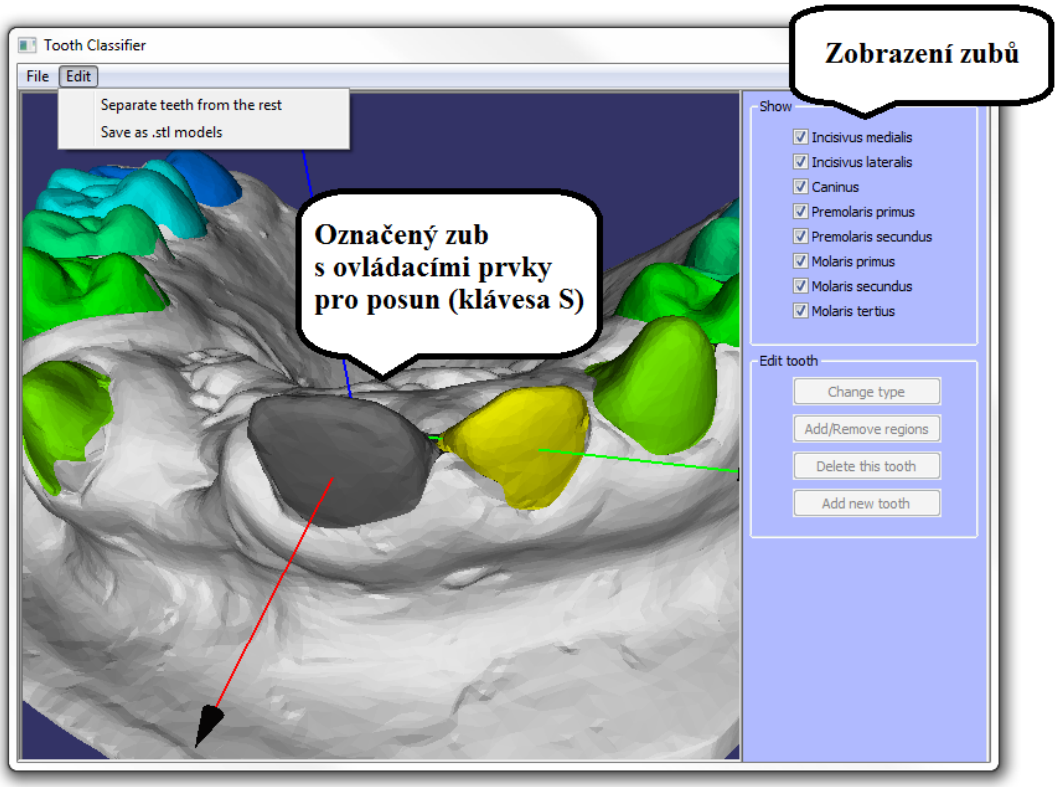
Editační mód lze opustit opětovným stiskem tlačítka **Add/Remove regions.**

Oddělení jednotlivých zubů a jejich úpravy

Pokud již na modelu není co změnit, všechny úpravy proběhly úspěšně, je možné oddělit jednotlivé zuby do samostatných modelů. Pro tento účel existuje v horní liště menu nabídka **Edit=>Separate teeth from the rest**.

Po této akci již není možné zubní segmenty nijak upravovat, tento mód však umožňuje množství nových funkcí. Prvky zapínání/vypínání jednotlivých zubů v pravé nabídce již plně vypínají a zapínají zobrazení jednotlivých zubů. Navíc je možné označený zub posunovat a otáčet podle přání uživatele pro plánování korekcí zubního oblouku.

Jednotlivé korekce lze zpřístupnit stiskem klávesy **S** pro posun a **R** pro rotaci. Tato možnost však funguje pouze u označeného zubu.



Zuby je možné též jednotlivě uložit pomocí příkazu v hlavní nabídce. Pro uložení je nutné vybrat adresář, kam program automaticky uloží jednotlivé zuby jako oddělené .stl modely (zuby se ukládají s aktuální transformací, čili tak, jak je uživatel upravil).

Analogicky lze zuby též načíst z takto uloženého adresáře, a to přes hlavní nabídku, přesněji **File=>Load Segmented Model**. U načteného modelu lze pouze měnit pozice zubů a plánovat korekce.