VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANALÝZA POLYGONÁLNÍCH MODELŮ POMOCÍ NEURONOVÝCH SÍTÍ

ANALYSIS OF POLYGONAL MODELS USING NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **MICHAELA DRONZEKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. OLDŘICH KODYM,**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Studentka: **Dronzeková Michaela, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Analýza polygonálních modelů pomocí neuronových sítí**
Analysis of Polygonal Models Using Neural Networks
Kategorie: Zpracování obrazu

Zadání:

1. Identifikujte a seznamte se s architekturami neuronových sítí vhodnými pro práci s polygonálními modely.
2. Obstarejte si vhodná data a definujte úlohu, na které budete provádět experimenty.
3. Úlohu nejprve řešte standardním způsobem s využitím rasterizace modelu a konvoluční neuronové sítě.
4. Vyberte jednu nebo více metod pro přímé zpracování polygonálního modelu.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Qi, Charles Ruizhongtai, Hao Su, Kaichun Mo and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016).
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. "Attention is All you Need." NIPS (2017).
- Kipf, Thomas N. and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." ICLR (2016).

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Kodym Oldřich, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 20. listopadu 2019

Abstrakt

Práca sa zaoberá odhadom rotácie na 3D modeloch čeľuste. Popisuje metódy na priame spracovanie 3D modelov aj spracovanie modelu pomocou rasterizácie a porovnáva tieto metódy. Na vyhodnotenie sietí bola zavedená metrika, ktorá počíta počet prípadov, v ktorých sieť urobila v odhade rotácie chybu menšiu ako 30° od skutočnej rotácie. Navrhnutá metóda, ktorá pracovala s rasterizáciou, spracovávala röntgenové snímky z troch pohľadov pomocou konvolučných sietí. Dosiahla úspešnosť 99% vo vyššie uvedenej metrike. Metóda priameho spracovania polygoniálneho modelu ako sekvencie využíva attention mechanizmus a je inšpirovaná transformer architektúrou. Pre túto metódu bola navrhnutá aj špeciálna operácia poolingu, ktorá umožnila znížiť hlavne pamäťovú náročnosť siete. Dosiahla horšie výsledky 88%, ale nevyžadovala rasterizáciu a priamo spracovávala polygoniálny model. Okrem toho, dosiahla lepšie výsledky ako metóda s rasterizáciou, kde model nebol zobrazený ako röntge, ale len ako render. Posledná metóda spracováva model v grafovej reprezentácii. Grafová sieť mala veľký problém s pretrénovaním, preto nedosiahla dobré výsledky a nepovažujem ju za dobrú metódu priameho spracovania polygoniálneho modelu.

Abstract

This thesis deals with rotation estimation of 3D model of human jaw. It describes and compares methods for direct analysis od 3D models as well as method to analyze model using rasterization. To evaluate perfomance of proposed method, a metric that computes number of cases when prediction was less than 30° from ground truth is used. Proposed method that uses rasterization, takes three x-ray views of model as an input and processes it with convolutional network. It achieves best preformance, 99% with described metric. Method to directly analyze polygonal model as a sequence uses attention mechanism to do so and was inspired by transformer architecture. A special pooling function was proposed for this network that decreases memory requirements of the network. This method achieves 88%, but does not use rasterization and can process polygonal model directly. It is not as good as rasterization method with x-ray display, byt it is better than rasterization method with model not rendered as x-ray. The last method uses graph representation of mesh. Graph network had problems with overfitting, that is why it did not get good results and I think this method is not very suitable for analyzing plygonal model.

Kľúčové slová

polygoniálne modely, neurónové siete, odhad rotácie, attention mechanizmus, transformer, grafové konvolučné siete

Keywords

polygonal models, neural networks, rotation estimation, attention mechanism, transformer, graph convolutional network

Citácia

DRONZEKOVÁ, Michaela. *Analýza polygonálních modelů pomocí neuronových sítí.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Oldřich Kodym,

Analýza polygonálních modelů pomocí neuronových sítí

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracovala samostatne pod vedením Ing. Oldřicha Kodyma. Uviedla som všetky literárne pramene, publikácie a zdroje, z ktorých som čerpala.

.....
Michaela Dronzeková
26. mája 2020

Poďakovanie

Chcela by som poďakovať vedúcemu mojej práce za veľmi dobré rady a postrehy, ktoré pomohli zlepšovať výsledky navrhnutých metód. Ďakujem taktiež firme Tescan 3Dim za poskytnutie datasetu k práci.

Obsah

Kapitola 1

Úvod

S tým, ako sa v poslednej dobe začalo rozširovať používanie neurónových sietí hlavne v oblasti spracovania obrazu, je veľmi žiadané, aby sa dali používať aj na spracovanie 3D modelov. Spracovanie 3D modelov je výpočetne náročná úloha hlavne z toho dôvodu, že časová zložitosť veľmi často používanej konvolúcie vzrastie pri 3D dátach až na $O(n^6)$ oproti $O(n^4)$ pre 2D konvolúciu. Je preto lepšie hľadať iné spôsoby a reprezentácie objemových dát, aby sa mohli siete dobre a rýchlo trénovať.

V tejto práci sa zaoberám spôsobmi, ako spracovávať 3D model reprezentovaný pomocou polygoniálnej siete. Cieľom je porovnanie metód pracujúcich so zobrazením modelu do 2D obrázka a metód pracujúcich priamo nad modelmi. Pri projekcii modelu do 2D roviny sa stratí veľa informácií, ktoré by mohli neurónovým sieťam uľahčiť učenie sa nad 3D modelom. Ako úlohu, ktorú budem riešiť som si zvolila odhad rotácie modelu.

Túto úlohu najskôr riešim tak, že zobrazím projekciu modelu v 2D priestore a učím siete na obrázkoch. Potom ju riešim priamym spracovaním 3D modelu a nakoniec tieto dve metódy porovnávam. Vzhľadom na to, že sieť, ktorej vstupom je polygoniálny model by mala mať informácie o všetkých pohľadoch na tento model, predpokladám, že by priame spracovanie modelu mohlo dosiahnuť lepšie výsledky ako projekcia do 2D.

V práci najskôr predstavujem niekoľko metód, ako sa dá pristúpiť k odhadu rotácie pomocou rasterizácie (kapitola 3). Neskôr, v kapitole 3.2 ukazujem, aké metódy boli navrhnuté na spracovanie 3D modelu pomocou iných reprezentácií. Popisujem tu metódy pre model reprezentovaný pomocou , point cloudu, polygoniálnej siete, octree štruktúry a grafových konvolučných sietí. V kapitole 3.3.1 popisujem attention mechanizmus, ktorý som použila na spracovanie modelu reprezentovaného ako sekvencia.

Kapitola 4 popisuje navrhnuté metódy na analýzu modelu. Pracovala som s tromi metódami. Prvá bola založená na rasterizácii. Model som si zobrazila ako röntgenový 2D obrázok, ktorý som spracovala konvolučnými sieťami. Druhá metóda pracovala priamo nad polygoniálnym modelom, ktorý reprezentovala ako sekvenciu. Na jej spracovanie som využila architektúru založenú na attention mechanizme. Posledná metóda opäť pracovala priamo nad polygoníálnym modelom, ktorý reprezentovala ako graf. Na analýzu tejto reprezentácie som následne použila grafové konvolučné siete.

V kapitole 5 sú popísané experimenty, ktoré som s navrhnutými metódami vykonala a ich vzájomné porovnanie. Experimenty ukázali, že najlepšie výsledky dosahuje metóda rasterizujúca model a zobrazujúca ho ako röntgen. Veľmi dobré výsledky dosahuje aj metóda s attention mechanizmom. Naopak grafové siete sa ukázali ako nevhodné pre mnou riešenú úlohu odhadu rotácie.

Kapitola 2

Využitie odhadu rotácie 3D modelov

Využitie neurónových sietí v oblasti medicíny je veľmi známe. Dajú sa použiť na odhad choroby z anamnézy pacienta, pomôcť lekárovi určiť, či sa na röntgenovej snímke nachádza choroba napríklad [7]. Na spracovanie röntgenových snímok sú vhodné konvolučné neurónové siete. Plne prepojené neurónové siete, prípadne rekurentné siete sa dajú zasa použiť na predikciu choroby pacienta z informácií o pacientovi a jeho zdravotnom stave napríklad [11]. Vďaka svojej robustnosti sú neurónové siete schopné sa naučiť predikovať choroby aj z dát, ktoré obsahujú šum.. Okrem toho sú schopné dosiahnuť lepšie výsledky ako iné metódy umelej inteligencie a to často na úplne odlišných typoch úloh. Nevýhodou pri použití neurónových sietí oproti klasickým metódam umelej inteligencie, ako je napríklad rozhodovací strom, je to, že nevieme určiť, na základe čoho sa sieť rozhodla pre daný výsledok. Toto môže byť niekedy pre doktora dôležitá informácia. Ďalšou nevýhodou je, že neurónové siete vyžadujú veľké množstvo dát na to, aby sa naučili dobre niečo predikovať. Pri citlivých medicínskych dátach môže byť niekedy problém získanie dostatočného počtu vzoriek.

2.1 Motivácia a vymedzenie úlohy

Okrem obrazových a štruktúrovaných dát sa v medicíne používajú aj 3D modely. Sú dobre využiteľné napríklad pri práci stomatológov. Existujúci software poskytuje možnosti načítania čeľuste pacienta a zobrazenie 3D modelu chrupu. Dá sa použiť na navrhovanie implantátu zubov a asistenciu pri vykonávaní operácie. Pri tomto zákroku sa najskôr vytvorí šablóna (obrázok 2.1), ktorá pomáha doktorovi pri operácii. Dentálny software umožňuje návrh modelu tejto šablóny a jej vytlačenie na 3D tlačiarni.

Práce z oblasti spracovania 3D modelov (popísané v 3.1.2) ukazujú, že analýza 3D modelov je výpočetne a pamäťovo náročná, kvôli použitiu 3D konvolúcie. Kvôli tomu nie sú neurónové siete schopné spracovať modely väčšie ako 32x32x32 voxelov. V praxi by ich lekár s nie veľmi výkonným počítačom ani nemohol použiť. Preto je dobré hľadať iné reprezentácie alebo spôsoby spracovania modelov, napríklad tie popísané v 3.2.

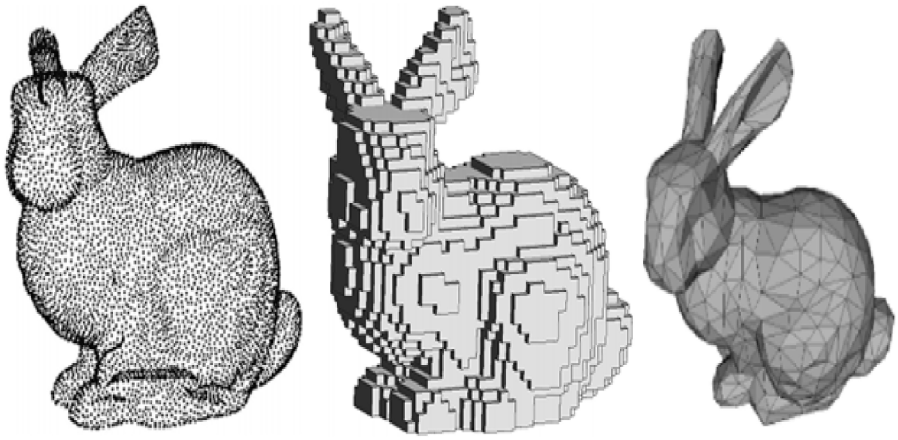
[1] Prevzaté z https://formlabs.ib-caddy.com/dental.php

Obr. 2.1: Šablóna na zuby používaná ako pomôcka pri zavádzaní implantátu zubu[1].

2.1.1 Návrh implantátu a tvorba šablóny

Proces celého navrhovania implantátu môže byť pre doktora náročný, musí dávať pozor, aby implantát správne sadol, aby nezasahoval do nervu a podobne. Niektoré časti návrhu by bolo možné automatizovať. Proces celého návrhu implantátu a šablóny na zuby, za pomoci softwaru, sa dá zhrnúť v nasledujúcich krokoch [3, 4]:

1. Vytvorenie CT skenu pacienta a jeho načítanie do softwaru. Zo skenu sa vytvorí 3D model čeľuste.
2. Detekcia mandibulárneho nervu. Na plánovanie implantátu je potrebné vedieť, kde sa nachádza nerv, aby ho implantát nepoškodil. Tento krok by sa dal taktiež automatizovať pomocou neurónových sietí.
3. Výber implantátu a jeho zasadenie do čeľuste v modeli. Toto vyžaduje nastavenie správnej veľkosti korunky implantátu, jeho sklonu a dĺžky tak, aby nezasahoval do nervu.
4. Vytvorenie modelu chrupu. To bude viesť k vytvoreniu šablóny na zuby.
5. Zarovanie modelu chrupu s CT skenom. Toto sa robí zatiaľ manuálne tak, že sa označí niekoľko zubov na modelu a CT skene a na základe nich sa modely zarovnajú do rovnakej pozície. Toto je bod, v ktorom sa dá využiť odhad rotácie modelu.
6. Na modeli chrupu sa vyznačí len časť okolo implantátu, ktorá sa bude používať ako výsledná šablóna pri vkladaní implantátu.
7. Exportovanie šablóny a vytlačenie na 3D tlačiarni.



Obr. 2.2: Rôzne reprezentácie modelu. Vľavo sa nachádza point cloud, v strede je 3D mriežka a vpravo polygoniálny model. Prevzaté z [17].

2.2 Reprezentácie 3D modelov

Pri spracovaní 3D modelov pomocou neurónových sietí sa využívajú hlavne reprezentácie modelu pomocou mriežky voxelov alebo point cloudu. Veľmi častá reprezentácia modelu, ktorá sa používa na prácu s nimi a modelovanie je polygoniálna.

Voxelová mriežka

Model reprezentovaný pomocou 3D mriežky je zložený z jednotkových kociek, ktoré sa nazývajú voxely, podobne ako je obraz v počítači reprezentovaný pomocou pixelu. Každý voxel má priradené numerické hodnoty, ktoré reprezentujú jeho vlastnosti, napríklad farbu, materiál, hustotu a podobne [19]. Na rozdiel od povrchových modelov, pri tejto reprezentácii je uložená informácia aj o tom, čo sa nachádza vo vnútri modelu, nie len na povrchu.

Point cloud

Autori článku [29] prvýkrát predstavili túto reprezentáciu povrchu modelu. Reprezentovali ho pomocou veľkého počtu častíc, na ktoré pôsobili vonkajšie sily ako napríklad gravitácia. Častica bola reprezentovaná svojou polohou, rýchlosťou a smerom pohybu, zrýchlením, hmotnosťou a ďalšími vlastnosťami. Boli použité na modelovanie napríklad toku vody alebo šírenia sa ohňa. Statické modely môžu byť tiež reprezentované množinou bodov, ktoré sa dajú získať meraním bodov na povrchu modelu. Každý bod je potom reprezentovaný svojou pozíciou, normálou a ďalšími vlastnosťami.

Polygoniálny model

Podobne ako point cloud, aj polygoniálna reprezentácia sa používa na zobrazenie povrchu modelu. Model je reprezentovaný pomocou vrcholov, hrán a plôch. Plochy sú najčastejšie tvorené trojuholníkmi.

Príklad modelu v rôznych reprezentáciách je na obrázku 2.2.

2.3 Odhad rotácie 3D modelu čeľuste

Pri odhade rotácie čeľuste chcem dosiahnuť to, že model má definovanú určitú počiatočnú pozíciu, do ktorej by sa mal dostať, aby bol zarovnaný. Je však rotovaný o neznámy uhol v priestore. Cieľom je odhadnúť rotáciu od počiatočnej pozície tak, aby sa model podľa tejto rotácie potom mohol vrátiť do počiatočnej polohy. Očakávaný výstup z navrhnutých sietí je uhol otočenia od počiatočnej pozície zapísaný vo zvolenej reprezentácii. V nasledujúcej podkapitole popíšem možnosti reprezentácie uhlu v priestore.

2.3.1 Rotačná matica

Rotačná matica je matica veľkosti 3x3. Pre maticu platí, že:

1. je ortogonálna, teda $RR^T = I$.
2. jej determinant je rovný 1 ($det(R) = 1$).

Základné rotácie podľa osí x, y a z o uhol θ sa dajú vyjadriť nasledovne [1]:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}$$

$$R_y = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

$$R_z = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ďalšie rotácie sa dajú skladať násobením rotačných matíc.

2.3.2 Axis-angle

Táto reprezentácia vyjadruje rotáciu ako uhol otočenia okolo nejakej jednotkovej osi. Na obrázku 2.3 je zobrazený význam tejto rotácie. Axis-angle sa potom vyjadruje ako dvojica (a, θ), kde a je vektor reprezentujúci os a θ je uhol otočenia. Táto reprezentácia sa dá ešte zjednodušiť na 3 hodnoty a to tak, že vynásobíme vektor a hodnotou θ. Aby sme z tejto reprezentácie získali uhol a vektor, stačí vypočítať normu vektoru. Tým zistíme uhol. Po vydelení jednotlivých zložiek vektoru jeho normou dostaneme jednotkovú os a. S týmto je však problém, ak $\theta = 0$. Vtedy po vynásobení dostaneme samé nuly a už z toho nevieme späť získať os a.

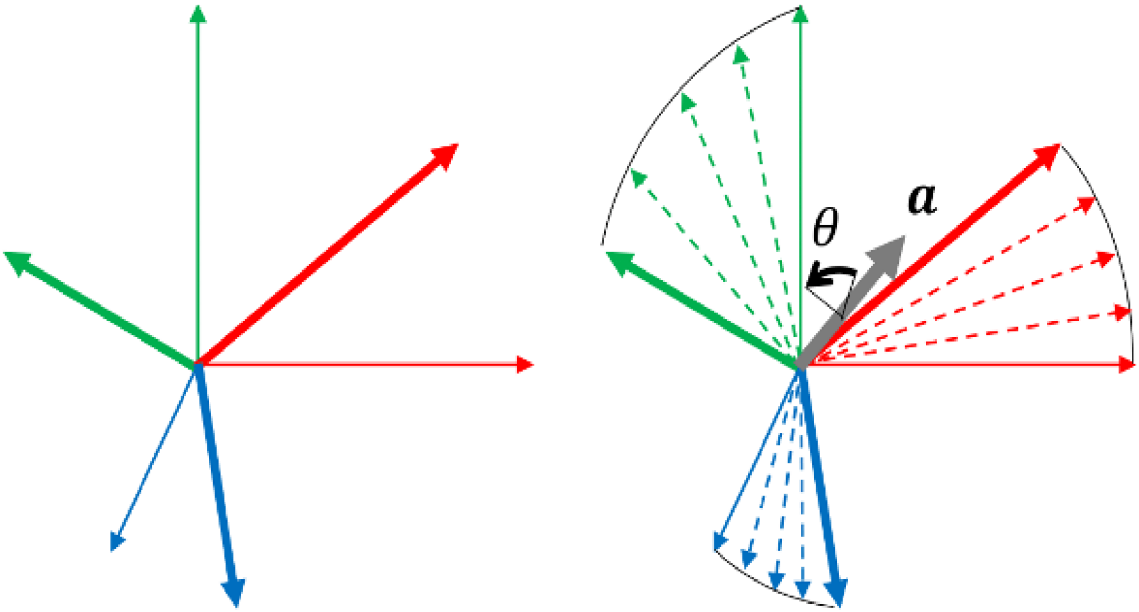
V prípade, že $\theta = \pi$, každá os a jej negácia predstavujú tú istú rotáciu.

Prevod na rotačnú maticu a naopak

Rotačná matica sa dá z axis-angle reprezentácie získať výpočtom podľa 2.1 [1].

$$R_{aa}(a, \theta) = I + sin(\theta)\hat{a} + (1 - cos(\theta))\hat{a}^2 \tag{2.1}$$

Matica I predstavuje jednotkovú maticu, $\hat{a}$ je matica vektorového súčinu a pre vektor $a = (a_x, a_y, a_z)$ sa dá napísať ako:



Obr. 2.3: Rotácia reprezentovaná pomocou axis-angle. Tenké čiary predstavujú pôvodné osi, hrubé čiary predstavujú rotáciu, ako by bola vyjadrená pomocou rotačnej matice. Sivá čiara predstavuje os v axis-angle reprezentácii a uhol θ vyjadruje uhol otočenia okolo tejto osi. Prevzaté z [1].

$$\hat{a} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

Na prevod z rotačnje matice na axis-angle reprezentáciu sa použijú nasledujúce rovnice. Uhol otočenia získame výpočtom 2.2 a os a získame pomocou 2.3, kde r_{ab} reprezentuje prvok rotačnej matice [1].

$$\theta = cos^{-1}\left(\frac{tr(R) - 1}{2}\right) \tag{2.2}$$

$$\begin{aligned} a_x &= (r_{32} - r_{23})2sin(\theta) \\ a_y &= (r_{13} - r_{31})2sin(\theta) \\ a_x &= (r_{21} - r_{12})2sin(\theta) \end{aligned} \tag{2.3}$$

2.3.3 Kvaternióny

Kvaternióny predstavujú rozšírenie komplexných čísel. Komplexné číslo $p = a + ib$ môžeme zapísať aj v tvare $p = \cos(\theta) + i\sin(\theta)$. Násobenie komplexných čísel predstavuje rotáciu v 2D priestore [5]:

$$p = a + ib$$

$$q = \cos(\theta) + i\sin(\theta)$$

$$pq = (a + ib)(\cos(\theta) + i\sin(\theta)) = a\cos(\theta) - b\sin(\theta) + i(\sin(\theta) + \cos(\theta))$$

Toto sa dá rozšíriť na 3D priestor tým, že pridáme ďalšie dve imaginárne čísla:

$$q = a + bi + cj + dk$$

Bod v priestore $p = (0, x, y, z)$ môžeme potom rotovať vykonaním operácie, ktorá sa nazýva **conjugation**:

$$p' = qpq^{-1}$$

Prevod na iné reprezentácie

Vzťah medzi kvaterniónmi a axis-angle reprezentáciou je nasledovný [1]:

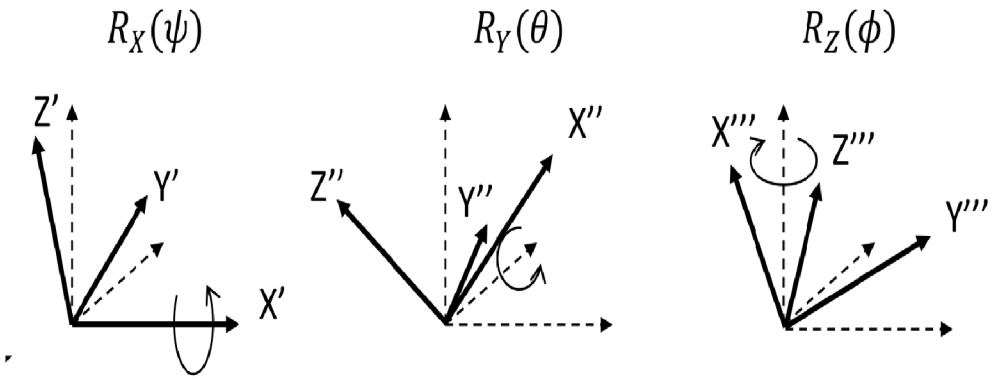
$$(q_0, q_1, q_2, q_3) = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})a_x, \sin(\frac{\theta}{2})a_y, \sin(\frac{\theta}{2})a_z)$$

Prvky rotačnej matice sa z kvaterniónovej reprezentácie vypočítajú nasledovne:

$$R = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix}$$

2.3.4 Eulerove uhly

Táto reprezentácia vychádza z definície rotačných matíc R_x, R_y a R_z. Vyjadruje sa ako 3 uhly (α, β, γ), ktoré predstavujú rotáciu okolo jednej zo zvolených osí. Poradie zvolených osí závisí na konvencii.



Obr. 2.4: Eulerove uhly aplikované podľa konvencie roll-pitch-yaw. Prvý obrázok zľava predstavuje rotáciu okolo osi X, prostredný okolo osi Y a posledný rotáciu okolo osi Z.

Najčastejšie sa používa **roll-pitch-yaw**, kde **roll** predstavuje rotovanie okolo osi X, **pitch** okolo osi Y a **yaw** okolo osi Z. Toto sa označuje aj ako ZYX konvencia. Poradie, v akom sa objekt rotuje je X, Y, Z [1], ako je znázornené na obrázku 2.4.

Kapitola 3

Metódy analýzy 3D modelu

V tejto kapitole popisujem rôzne spôsoby spracovania 3D modelu pomocou neurónových sietí. Najskôr sa zameriavam na techniky, ktoré využívajú rasterizáciu, neskôr popisujem techniky, ktoré pracujú s rôznymi reprezentáciami modelu.

3.1 Metódy analýzy 3D modelu pomocou rasterizácie

V tejto sekcii sú popísané rôzne spôsoby analýzy 3D modelu pomocou jeho rasterizácie. Rozdeľujem ich na dve kategórie. Prvá používa projekciu 3D modelu na 2D obrázok, ktorý sa následne spracováva klasickými konvolučnými vrstvami. Druhá kategória využíva voxelizované modely a tiež sa spracováva konvolučnými sieťami. Na rozdiel od obrázkov tieto metódy využívajú 3D konvolúciu.

3.1.1 Existujúce riešenia odhadu 3D rotácie z 2D obrázka

Na odhad rotácie v 3D priestore sa využíva prevažne prístup, v ktorom sa vypočíta projekcia 3D modelu alebo scény do 2D obrázka. Veľa riešení odhadu rotácie skúša priamo odhadnúť 3D rotáciu, líšia sa len použitou reprezentáciou rotácie, prípadne použitou loss funkciou. Popisujem tu hlavne metódy, z ktorých som vychádzala pri návrhu mojich neurónových sietí a ich loss funkcií. Väčšina nižšie spomenutých metód využíva na odhad rotácie jeden pohľad na model, existujú však aj metódy, ktoré využívajú viac pohľadov na model, napríklad [27].

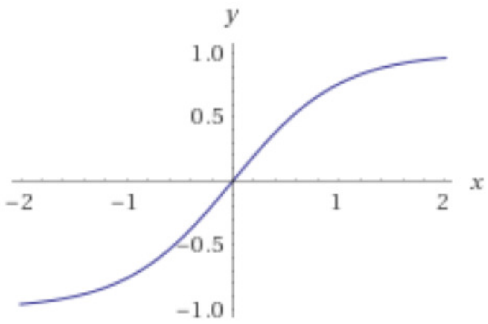
Tento spôsob odhadu 3D rotácie je preferovaný pred priamym spracovaním 3D modelu, kvôli jeho vysokým výpočetným nárokom popísaným v 3.1.2.

Priamy odhad rotácie

Metódy, ktoré sa snažili priamo odhadnúť rotáciu fungujú tak, že na vstup dostanú jeden pohľad na model a z neho sa snažia najskôr extrahovať rysy pomocou konvolučných vrstiev. Rysy sa potom spracujú pomocou plne prepojenej vrstvy a na výstupe siete je odhad rotácie väčšinou ako kvaternión [23, 20].

Architektúra siete, ktorú tieto metódy využívali bola podobná VGG sieti. V plne prepojenej vrstve používali väčší počet neurónov (512 - 4096). V článku [20] to odôvodňovali tým, že vzťahy medzi rysmi extrahovanými z obrázka a výsledným kvaterniónom sú komplikované.

Ako aktivačné funkcie sa vo všetkých vrstvách okrem poslednej využívajú ReLU. V poslednej vrstve je to **tanh** (zobrazená na obrázku 3.1) kvôli tomu, že výstup nadobúda



Obr. 3.1: Priebeh funkcie hyperbolický tangens

hodnoty z intervalu $(-1, 1)$. V prípade, že sa ako výstup zvolila iná reprezentácia rotácie napríklad axis-angle, tak sa použila aktivácia π **tanh**, lebo výstupné hodnoty mohli byť z intervalu $(-\pi, \pi)$.

Loss funkcie. Typicky používané loss funkcie na priamy odhad rotácie sú **mean squared error** (rovnica 3.1) alebo **geodetická vzdialenosť** dvoch rotačných matíc (rovnica 3.2), prípadne geodetická vzdialenosť kvaterniónov (rovnica 3.3).

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2 \tag{3.1}$$

$$d_A(R_{true}, R_{pred}) = cos^{-1}\left[\frac{tr(R_{true}^T R_{pred}) - 1}{2}\right] \tag{3.2}$$

$$d_Q(q_{true}, q_{pred}) = 2cos^{-1}(|\langle q_{true}, q_{pred}\rangle|) \tag{3.3}$$

Funkcia mean squared error sa typicky používa pre úlohy regresie, avšak v tomto prípade by sme nevedeli z hodnôt funkcie určiť, ako dobre model odhaduje rotáciu. Preto je lepšie použiť geodetické funkcie. Avšak problém s týmito funkciami je, že majú veľa lokálnych miním a preto je treba siete predtrénovať napríklad práve pomocou funkcie mean squared error [23].

Metóda, ktorú využili autori siete PoseNet [35] odhadovala rotáciu pomocou kvaterniónov. Navrhnutá loss funkcia počítala v 3D priestore modelu. Počítala priemernú vzdialenosť medzi bodmi na korektne natočenom modeli a korešpondujúcimi bodmi na modeli otočenom podľa odhadnutej rotácie. Keďže sa transformácie bodov lepšie počítajú pomocou rotačnej matice, kvaternión q najskôr previedli na rotačnú maticu $R(q)$ Celý vzorec pre túto loss funkciu udáva rovnica 3.4.

$$PLoss(q_{pred}, q_{true}) = \frac{1}{2m}\sum_{x \in M} \parallel R(q_{pred})x - R(q_{true})x \parallel^2 \tag{3.4}$$

Pri symetrických objektoch však existuje viac správnych rotácií a táto loss penalizovala aj takéto rotácie. Preto zaviedli modifikovanú loss funkciu (rovnica 3.5), ktorá tieto rotácie nepenalizovala. Funkcia meria vzdialenosť medzi každým bodom na modeli rotovaných podľa odhadnutej rotácie a medzi najbližším bodom z modelu rotovaného podľa skutočnej

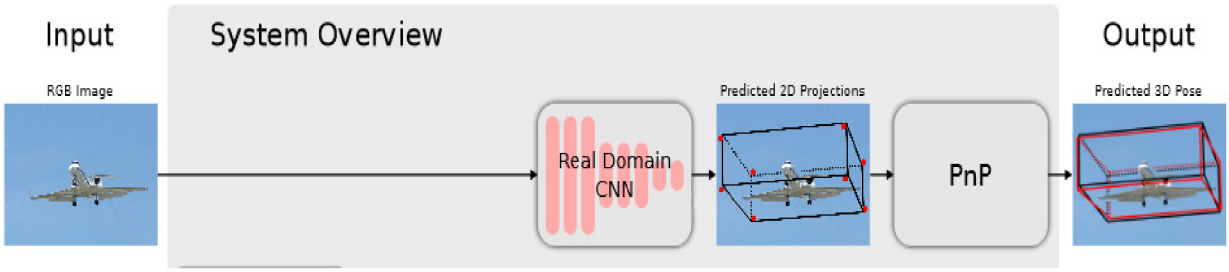
rotácie. Táto funkcia je minimálna, keď sa dva modely úplne prekrývajú.

$$SLoss(q_{pred}, q_{true}) = \frac{1}{2m} \sum_{x_1 \in M} min_{x_2 \in M} \parallel R(q_{pred})x_1 - R(q_{true})x_2 \parallel^2 \tag{3.5}$$

Detekcia 3D bounding boxu a odhad rotácie

Ďalšia metóda, ktorá sa trochu odlišuje od metód využívajúcich čisto len konvolučné siete, funguje tak, že odhadne 2D projekciu bodov pre 3D bounding box okolo modelu a potom odhadne 3D pozície týchto bodov a rotáciu pomocou PnP algoritmu [15].

Detekciu bounding boxu môže vykonávať neurónová sieť, buď VGG alebo ResNet, ktorá je nasledovaná jednou plne prepojenou vrstvou. Autorom v tomto prípade stačilo 19 neurónov v jednej vrstve. Architektúra je znázornená na obrázku 3.2. PnP problém spočíva



Obr. 3.2: Architektúra systému využívajúca PnP algoritmus. Prevzaté z [15].

v tom nájsť pozíciu a orientáciu kamery, keď máme k dispozícii korešpondenciu medzi 3D bodmi a ich 2D projekciou [21].

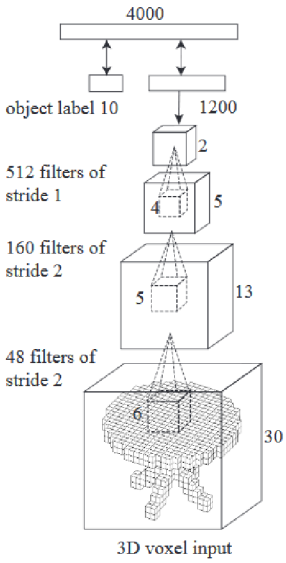
3.1.2 Metódy využívajúce objemovú reprezentáciu

V tejto skecii popisujem metódy priamej analýzy 3D modelu pomocou jeho voxelizácie (reprezentácia pomocou voxelovej mriežky je popísaná v 2.2). Keďže sa nad touto reprezentáciou riešia hlavne úlohy klasifikácie a segmentácie, nie sú tu popísané architektúry, ktoré by priamo riešili úlohu odhadu rotácie. Avšak dajú sa veľmi jednoducho upraviť tak, aby odhadovali rotáciu a to zmenou poslednej aktivačnej funkcie a pouitím inej loss funkcie.

3D ShapeNet

Táto architektúra pracuje priamo s voxelmi modelu. Bola navrhnutá na to, aby dokázala zaradiť objekt do kategórie a doplnenie chýbajúcej časti modelu. Modely reprezentovali ako binárny tensor, kde 1 znamenalo, že sa voxel nachádza vo vnútri modelu a 0, že sa nachádza mimo. Sieť sa snaží naučiť reprezentáciu 3D modelu tak, že odhaduje pravdepodobnosť, že voxel sa nachádza vo vnútri modelu alebo mimo modelu [33] . Na tento účel použili autori tzv. konvolučnú deep belief sieť (architektúra znázornená na obrázku 3.3), ktorá je schopná sa dobre učiť pravdepodobnostné rozloženie. Problém však bol v náročnosti spracovávania 3D modelu. Pre zvolenú veľkosť 30x30x30 by sa obyčajná deep belief sieť musela naučiť veľké množstvo parametrov. Preto tu použili konvolučné vrstvy. V sieti nepoužívali žiadne pooling vrstvy, lebo by to zvýšilo neistotu pri rekonštrukcii tvaru modelu. Táto architektúra dosahovala pri odhade kategórie modelu na datasete ModelNet [1] úspešnosť 77%.

[1] https://modelnet.cs.princeton.edu/



Obr. 3.3: Architektúra ShapeNet siete. Prevzaté z [33].

VoxNet

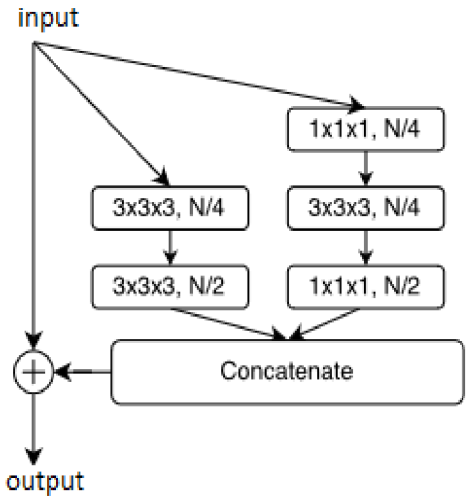
VoxNet je podobná architektúra siete, ktorá spracováva voxelizované 3D modely. Modely reprezentovali pomocou tzv. 3D occupancy grid. Táto reprezentácia udržiava stav prostredia ako 3D mriežku náhodných hodnôt a zachováva pravdepodobnostný odhad ich obsadenia. Táto reprezentácia bola zvolená preto, aby sa dalo ľahko zistiť, či je voxel voľný, obsadený alebo je táto informácia neznáma [25]. Cieľom návrhu tejto siete bolo predikovať kategóriu objektu. Pracovali s modelmi veľkosti 32x32x32. Architektúra siete bolo jednoduchá, použili dve 3D konvolučné vrstvy, pooling a dve plne prepojené vrstvy.

Okrem toho, že používali priamo 3D voxelizované modely, vykonávali aj experimenty s point cloud reprezentáciou, ktorú si previedli na occupancy grid. Pri vytváraní vlastnej occupancy grid si mohli zvoliť rôzne rozlíšenie voxelov. Na týchto modeloch sledovali aj to, ako rôzne rozlíšenie voxelov ovplyvní dosiahnuté výsledky. Zvolili 2 rôzne rozlíšenia, kedy si veľkosť voxelu zvolili 0.1 a 0.2 metru. Už pri rozlíšení 0.2m sa ukázalo, že to postačuje na obsiahnutie všetkých informácií. Pri trénovaní sietí samostatne najskôr s rozlíšením 0.1 a potom s 0.2 sa ukázalo, že výsledky sú rovnaké. Pri skombinovaní oboch rozlíšení do jednej siete však dosiahli o trochu lepšie výsledky. Na datasete ModelNet, kde boli rovno k dispozícii voxelizované modely, teda sa na nich nemohlo nastavovať rozlíšenie, dosahovali pri klasifikácii úspešnosť 83%.

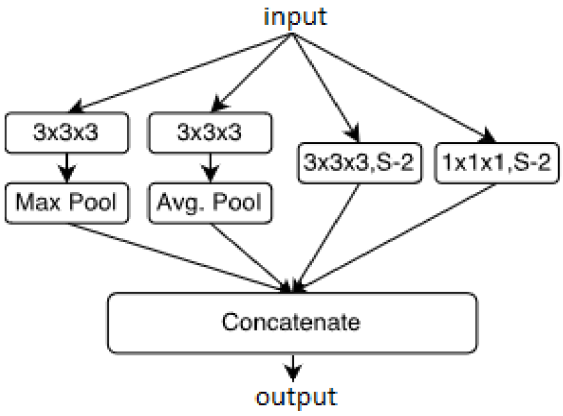
Voxception ResNet

Architektúra navrhnutá [9] pracovala taktiež nad voxelizovanými modelmi veľkosti 32x32x32. Boli navrhnuté dve metódy, jedna z nich využívala autoenkóder na rekonštrukciu modelov. Druhá, ktorá je tu viac popísaná sa opäť snažila zaradiť modely do kategórií. Tentokrát však autori zvolili prístup, kde skombinovali Inception a ResNet architektúry. V ResNet bloku sa nachádzajú 2 rôzne vetvy, ktoré predstavujú inception princíp - v jednej vetve sa dvakrát za sebou vykoná konvolúcia s jadrom veľkosti 3x3x3, v druhej vetve sa vykoná konvolúcia s jadrom 1x1x1 nasledovaná konvolúciou 3x3x3 a ďalšou konvolúciou 1x1x1. Vý-

stupy z týchto dvoch vetví sa zlúčia. Tento výstup sa ešte sčíta so vstupom bloku, tak ako je to pri ResNet architektúre, prípadne sa namiesto identity použije projekcia (obázok 3.4). Aj podvzorkovanie je robené pomocou špeciálnych blokov tzv. voxception-downsample, kde využívajú princíp inception bloku (obrázok 3.5). Na datasete ModelNet tým dosiahli veľmi dobré výsledky (91%).



Obr. 3.4: ResNet blok skombinovaný s inception princípom. Prevzaté z [9].



Obr. 3.5: Povzorkovanie vykonávané inception blokom. Prevzaté z [9].

3.2 Metódy využívajúce iné reprezentácie 3D modelov

V tejto sekcii popisujem metódy, ktoré analyzujú 3D modely reprezentované point cloudom a polygónmi. Niektoré z uvedených metód si z týchto základných reprezentácií vytvárajú vlastné, ktoré sú menej náročné na spracovanie napríklad octree v sekcii 3.2.3.

3.2.1 Metódy pracujúce s point cloud reprezentáciou

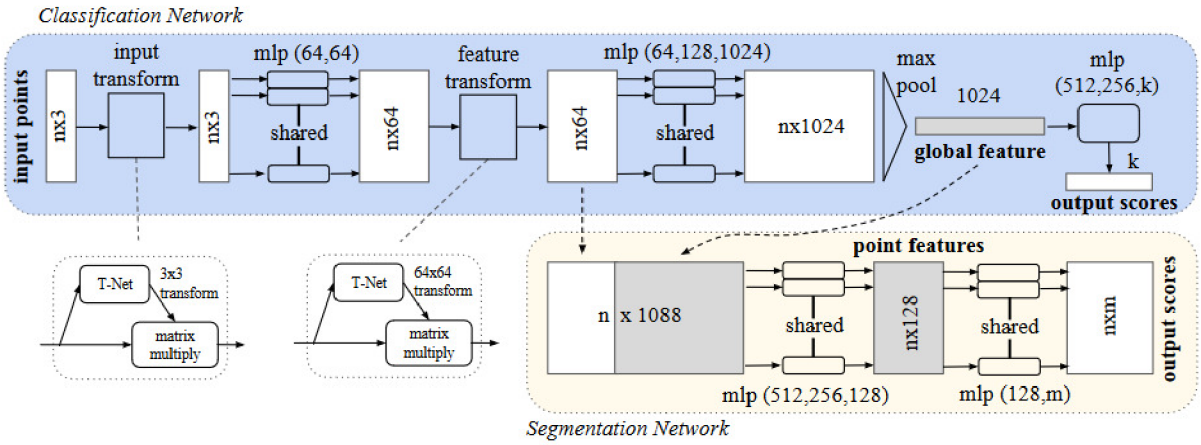
Vyššie zmienené metódy dokázali pracovať aj s modelmi reprezentovanými pomocou point cloudu (popísané v 2.2), ale museli si ich najskôr voxelizovať do 3D mriežky. Existujú však aj metódy, ktoré pracujú priamo nad point clound reprezentáciou.

PointNet

Táto metóda pracuje nad point cloud reprezentáciou tak, že sa na jednotlivé body pozerá ako na množinu, teda body v nej sú nezoradené [10].

Sieť bola navrhnutá tak, aby bola schopná odhadnúť kategóriu objeku a taktiež segmentovať model. Na odhadnutie kategórie by stačilo extrahovať globálne rysy. Segmentácia však vyžaduje aj extrakciu lokálnych rysov. To je dosiahnuté tak, že k rysom extrahovaným pre každý bod sa pripoja globálne rysy a z týchto vektorov sa sieť učí ďašie rysy pre každý bod tak, aby v nich bola zahrnutá ako lokálna, tak globálna informácia.

Celá architektúra je znázornená na obrázku 3.6.



Obr. 3.6: Architektúra PointNet. V modrom obdĺžniku je klasifikačná časť siete, ktorá sa pomocou multi-layer perceptron (mlp) sietí učí predikovať globálne rysy. V spodnej časti je sieť používaná na segmentáciu, ktorá sa učí lokálne rysy pre každý bod. Prevzaté z [10]

Point2Sequence

Architektúra navrhnutá v článku [22] sa snažila využiť attention mechanizmus (bližšie popísaný v 3.3.1) na zachytenie korelácií medzi rôznymi oblasťami lokálnych regiónov v modeli.

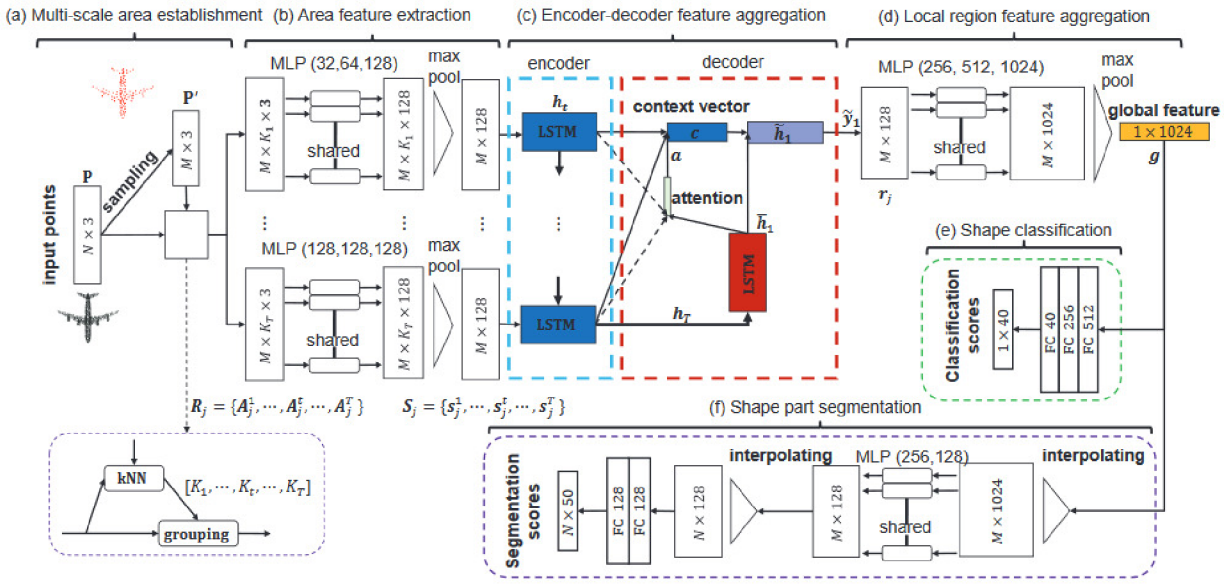
Predtým, ako použijú samotný attention mechanizmus v encoder-decoder architektúre, najskôr musia model predspracovať a rozdeliť body do lokálnych regiónov. Celá sieť sa skladá z niekoľkých častí (viz obrázok 3.7). Autori túto architektúru používali na klasifikáciu modelu a po úprave aj na jeho segmentáciu. Podobne by sa dal použiť aj na odhad rotácie.

Prvá časť nazvaná *multi-scale area establishment* rozdelí model na lokálne regióny. Najskôr nájde centroidy týchto regiónov a potom ku každému centroidu nájde jeho T najbližších susedov. Títo susedia budú predstavovať oblasti pre jeden región. Ako posledné sa každej oblasti priradia body, ktoré k nej patria.

Druhá časť, *multi-scale area feature extraction* sa skladá z multi-layer perceptron sietí, ktoré extrahujú rysy pre každú oblasť v lokálnom regióne. Extrahované rysy z oblastí pre jeden región sa následne agregujú do jedného vektoru rysov.

Nasleduje encoder-decoder architektúra s attention mechanizmom. Pre enkóder aj dekóder sú použité RNN siete. Attention je použitý v dekódovacej časti. Cieľom je vygenerovať kontextový vektor $c = \sum \alpha(t)h_t$. Vektor $\alpha(t)$ sa spočíta tak, že sa porovná aktuálny vnútorný stav h_1 s každým vstupným stavom h_t.

Rysy transformované encoder-decoder architektúrou s attention mechanizmom sa ďalej spracujú a agregujú sa do vektoru, ktorý reprezentuje globálne rysy extrahované z celého modelu.



Obr. 3.7: Architektúra Point2Sequence siete. Prevzaté z [22]

.

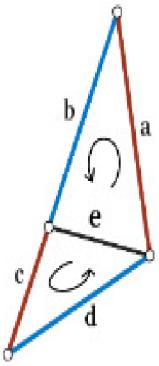
3.2.2 Metódy pracujúce nad polygoniálnymi modelmi

Metódy, ktoré pracujú nad polygoniálnymi dátami (popísané v 2.2) môžu pracovať dvomi spôsobmi. Buď si sieť polygónov rozložia do 2D obrázka [26] [14], alebo sú schopné spracovávať tento model priamo. Metódy na priame spracovanie polygoniálneho modelu definujú operáciu konvolúcie tak, aby sedela na túto reprezentáciu. Využíva sa napríklad geodetická konvolúcia [24]. Ďalej v tejto sekcii popisujem spôsob, ako sa dajú definovať klasické operácie v neurónových sieťach nad hranami modelu.

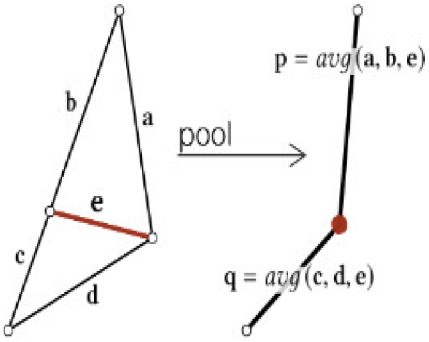
MeshCNN

MeshCNN [16] je sieť, ktorá dokáže priamo spracovať polygoniálny model a to tak, že upravuje operácie konvolúcie a poolingu tak, aby vedeli pracovať nad hranami modelu.

Konvolúcia nad hranou sa vykonáva tak, že sa zoberú 4 okolité hrany a nad nimi sa spočíta konvolúcia. Hrany sa môžu brať v rôznom poradí (viz obrázok 3.8). Aby na tomto



Obr. 3.8: Poradie hrán susedných k hrane e môže byť (a, b, c, d) alebo (c, d, a, b). Prevzaté z [16]



Obr. 3.9: Pooling operácia na štyroch susedných hranách. Výsledok sú dve nové hrany, ktoré sa spočítajú ako priemer vektoru rysov. Prevzaté z [16]

poradí nezáležalo, aplikujú sa na hrany symetrické operácie a nad nimi sa potom spočíta konvolúcia. Hrany sa modifikujú nasledovne:

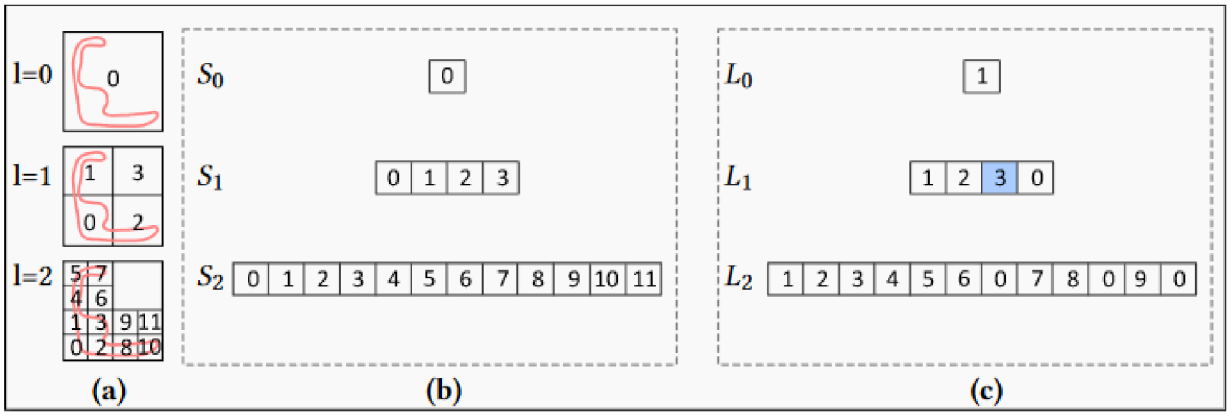
$$(e^1, e^2, e^3, e^4) = (\mid a - c \mid, a + c, \mid b - d \mid, b + d)$$

Pooling je realizovaný tak, že sa spája viac susedných hrán do jednej. Keby sme chceli vykonať pooling na hrane e z obrázku 3.8, tak by sa hrany c a d spojili do jednej a hrany b a a tiež. Z hrany e by zostal len jeden bod. Aby sa určili výsledné dve hrany, vypočíta sa priemer ich vektoru rysov (obrázok 3.9).

S takto definovanými operáciami sa dajú vytvárať konvolučné siete s rôznymi architektúrami. Fungujú dobre hlavne pri segmentačných úlohách.

3.2.3 Octree reprezentáca modelu

Priestorový model sa dá reprezentovať aj pomocou grafu, lepšie povedané stromu, kde každý uzol predstavuje určitú časť modelu. Octree je príkladom takéhoto stromu. Konštruuje sa tak, že sa okolo modelu vytvorí 3D kocka, ktorá sa rozdelí na 8 častí. Každá časť sa potom skontroluje a ak je neprázdna, tak sa delí na ďalších 8 častí [32]. To sa dá nakresliť ako strom, kde má každý uzol (ďalej nazývaný aj oktant) okrem listových 8 potomkov. Obrázok Najľavejšia časť obrázka 3.10 znázorňuje túto štruktúru, ale pre jednoduchosť a prehľadnosť je to ukázané len na štvorci, ktorý sa rozdeľuje na 4, nie 8 častí. Tento strom



Obr. 3.10: Ukážka stromu. Obrázok a) ukazuje ako sa štvorec delí na menšie a menšie časti. Čísla reprezentujú shuffle key. Na obrázku b) sú už znázornené shuffle key v stromovej štruktúre. Na obrácku c) sa nachádzajú pridelené anotácie jednotlivým uzlom. Prevzaté z [32]

.

sa dá zostrojiť z orientovaného polygoniálneho modelu alebo orientovaného point cloudu. Aby vedeli neurónové siete extrahovať nejakú informáciu z takejto reprezentácie, treba ešte každému oktantu priradiť určité vlastnosti.

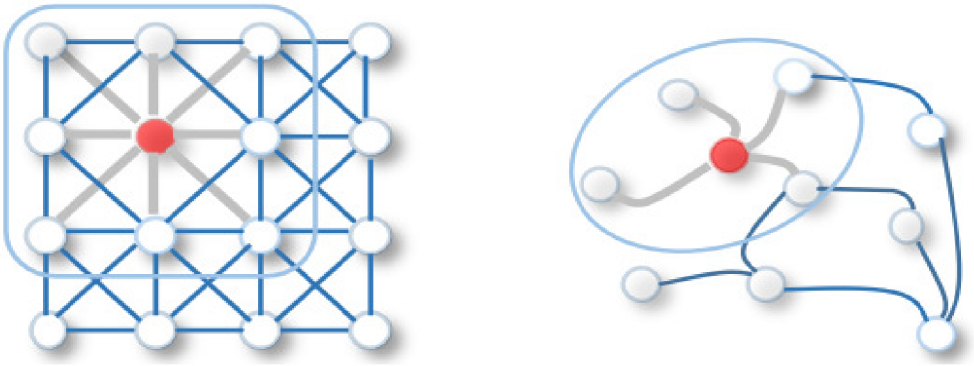
Nad takto definovanou reprezentáciou modelu autori definovali špeciálne operácie, ktoré sa správajú ako konvolúcia a pooling. Neurónové siete potom môžu mať podobu známych architektúr ako VGG a ResNet.

Operácie nad octree štruktúrou

Konvolúcia a pooling sú zmenené nasledovným spôsobom:

- **3D konvolúcia** pre daný oktant sa počítá zo susedných oktantov v tej istej hĺbke podľa vzorca 3.6. O_{ijk} reprezentuje susedný oktant, $T(O_{ijk})$ predstavuje vektor rysov pridelený oktantu na pozícii (i, j, k) a $T^{(n)}(O_{ijk})$ predstavuje n-tý kanál vektoru rysov. $W_{ijk}^{(n)}$ predstavuje váhu pre n-tý kanál vektoru rysov oktantu na pozícii (i, j, k). V prípade, že oktant na pozícii (i, j, k) neexistuje, lebo sa v ňom nenachádza model, je vektor rysov nastavený na 0.

$$\Phi(O) = \sum_n \sum_i \sum_j \sum_k W_{ijk}^{(n)} \cdot T^{(n)}(O_{ijk}) \tag{3.6}$$



Obr. 3.11: Vľavo je zobrazené počítanie konvolúcie nad mriežkou, napríklad 2D obrázkom. Červenou je zobrazený spracovávaný pixel. Jeho popis vznikne agregáciou hodnôt z jeho okolia. V pravo je konvolúcia na všeobecnom grafe. Popis spracovávaného červeného uzlu vznikne ako agregácia hodnôt z jeho susedných uzlov. Prevzaté z [34]

- **Pooling** vrstva slúži na zmenšenie dimenzie vstupu. Najpoužívanejší typ poolingu je max-pooling. Ten se na octree reprezentácii implementuje veľmi jednoducho a to tak, že se vyberie maximálny prvok z osmice.

3.2.4 Grafové konvolučné neurónové siete

Z polygoniálneho modelu alebo point-cloud reprezentácie 3D modelu sa dá vytvoriť grafová štruktúra predstavujúca tento model. Tá sa dá spracovať pomocou grafových konvolučných neurónových sietí.

Grafové konvolučné siete používajú operácie, ktoré sa používajú v bežných konvolučných sieťach a to konvolúciu a pooling. Tieto operácie sú však zmenené tak, aby sa dali počítať nad grafmi.

Konvolúcia

Myšlienka konvolúcie nad grafmi je podobná ako pri klasických konvolúciách. Konvolúcia nad 2D obrázkom sa pozerá na určité okolie spracovávaného pixelu a vlastnosti tohto okolia agreguje do jednej hodnoty, ktorá popisuje daný pixel. Podobne aj pri grafoch, sieť sa pozerá na určitý počet susedných uzlov práve spracovávaného uzlu a agreguje ich vlastnosti, aby popísala daný uzol. V prípade grafov však treba počítať s tým, že nezáleží na poradí uzlov a agregačný operátor by mal byť invariantný voči permutácii (napríklad sčítanie). Obrázok 3.11 znázorňuje rozdiel medzi konvolúciou nad mriežkou a nad všeobecným grafom.

Konvolučné filtre môžu byť definované dvomi spôsobmi, buď z priestorového alebo spektrálneho pohľadu [12]. Neurónová sieť na spracovanie 3D modelov [36] využíva práve spektrálny pohľad spolu s aproximáciou Chebychevovým polynómom, ktorý bližšie popíšem.

Graf, nad ktorým je definovaný signál je daný ako $G = (V, E, W)$, kde V je množina vrcholov, $|V| = n$, E je množina hrán a W je váhovaná matica susedov. Signál x je definovaný nad uzlami grafu. Je to vektor o veľkosti $|V|$, kde x_i prdstavuje položku signálu pre i-ty uzol grafu.

Dôležitým operátorom je tzv. Laplacian, ktorý je definovaný ako $L = D - W$, kde D je diagonálna matica stupňov uzlu. Táto matica má n vlastných vektorov $\{u_l\}_{l=0}^{n-1}$ a s nimi spojené vlastné hodnoty $\{\lambda_l\}_{l=0}^{n-1}$, ktoré predstavujú frekvencie grafu. Nech $U = [u_0, ..., u_n - 1] \in \mathbb{R}^{n \times n}$. Fourierova transformácia signálu x je potom definovaná ako $\hat{x} = U^T x$

Signál x je filtrovaný filtrom g_θ ako

$$y = g_\theta(L)x = U g_\theta(\Lambda) U^T x,$$

kde $\Lambda = diag([\lambda_0, ..., \lambda_{n-1}])$. Koeficienty, ktoré sa sieť učí sú $g_\theta(\Lambda) = diag(\theta)$ a $\theta \in (R)^n$ je vektor Fourierových koeficientov.

Aby sa zmenšila náročnosť učenia týchto parametrov, tak sa filter g_θ nahradí polynomiálnym filtrom veľkosti K a to takýmto spôsobom:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$$

Parameter θ je vektor koeficientov polynómu.

Polynóm, ktorý je vhodné na toto použiť sa nazýva Chebyshevov polynóm. Polynóm rádu k sa dá vypočítať pomocou rekurzívneho vzťahu $T_k(x) = 2xT_{k-1} - T_{k-2}(x)$. Filter je potom parametrizovaný polynomiálny rozvoj rádu $K - 1$:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda})$$

Parameter θ predstavuje vektor Chebyshevových koeficientov (trénovacie parametre) a $T_k(\tilde{\Lambda})$ je Chebyshevov polynóm rádu k vyhodnocovaný nad diagonálnou maticou škálovaných vlastných hodnôt v intervale $(-1, 1)$ [12].

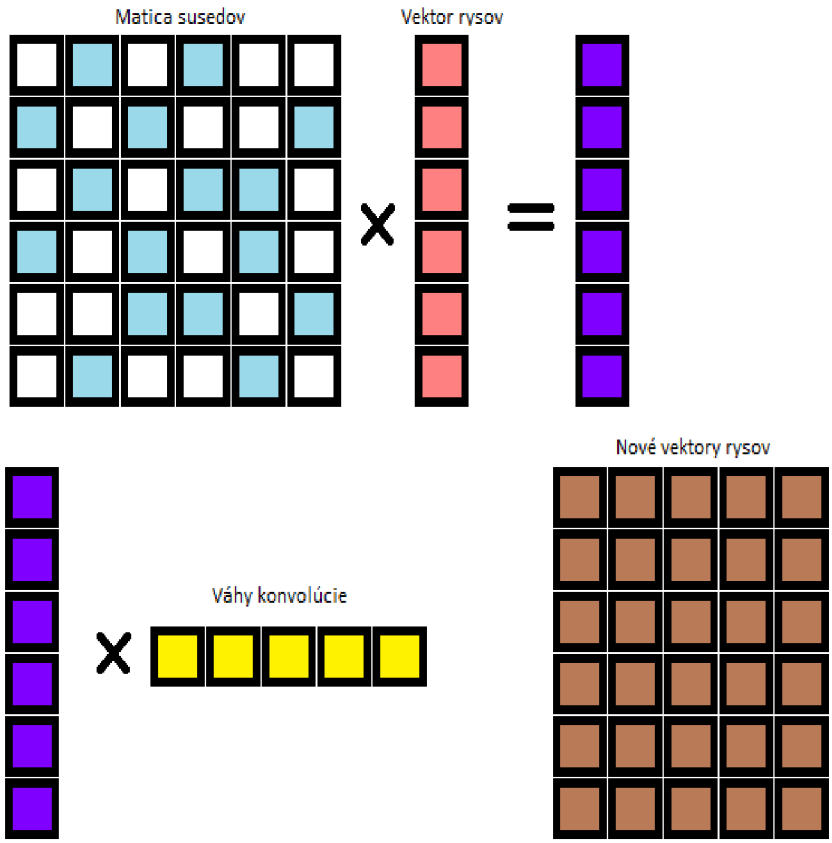
Pre lepšiu predstavu, čo sa deje v sieti pri grafovej konvolúcii si predstavme, že máme graf o 6 uzloch. Do neurónovej siete vstupuje maticová reprezentácia tohto grafu. K tomu do siete vstupuje signál x, čo je vektor dĺžky 6. Pri počítaní Chebychevovho polynómu je $T(0)$ rovné vstupnému signálu a $T(1)$ je rovné súčinu matice grafu a vstupného signálu. Pri násobení sa neberú do úvahy miesta, v ktorých má matica nulovú hodnotu. Toto teda predstavuje počítanie konvolúcie nad grafom tak, ako je to znázornené na obrázku 3.11 vpravo. Výstupom je opäť vektor o veľkosti 6x1, v ktorom je pre každý uzol nová hodnota spočítaná z rysov jeho susedov. Pre vyššie rády Chebyshevovho polynómu by to platilo podobne.

Pridanie váh ku konvolúcii prebieha podobne. Vygeneruje sa vektor o veľkosti požadovaných výstupných rysov, napríklad ich chceme 5. Vytvorí sa teda vektor váh o veľkosti 1x5 a vynásobí sa s každým spočítaným Chebyshevovým polynómom (tie majú veľkosť 6x1). Výstupom bude niekoľko matíc o veľkosti 6x5. Tieto matice sa sčítajú po prvkoch a výstupom bude nová matica 6x5, v ktorej každý riadok predstavuje vektor rysov popisujúci príslušný uzol v grafe.

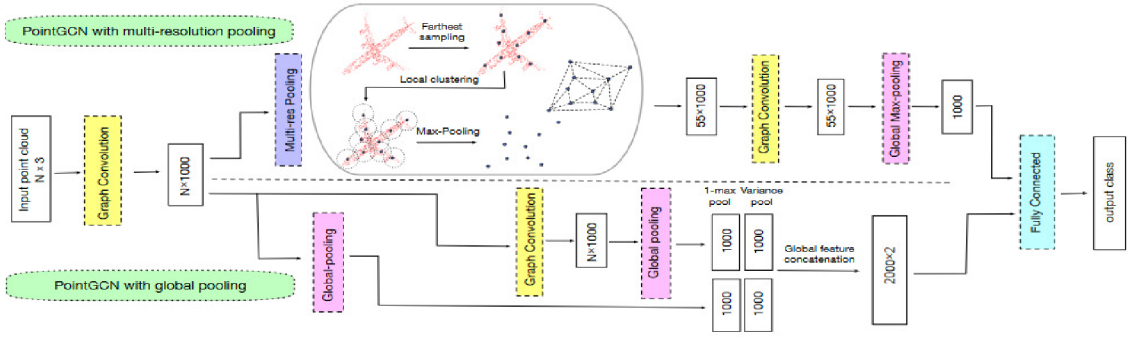
Celý tento proces je znázornený na obrázku 3.12.

Operácia poolingu

Autori siete [36], ktorá klasifikuje 3D modely do tried, navrhli dva spôsoby poolingu. Prvý, globálny pooling, pracuje nad celou grafovou reprezentáciou a výstupnými signálmi z konvolučnej vrstvy. Používajú 1-max-pooling, teda zoberú maximum z výstupov všetkých filtrov



Obr. 3.12: Jednoduchý príklad ilustrujúci spôsob fungovania konvolučnej vrstvy v grafových sieťach.



Obr. 3.13: Príklad architektúry grafovej neurónovej siete. Vrchná časť nad čiarkovanou čiarou zobrazuje sieť s multi-resolution poolingom, spodná časť zobrazuje sieť s globálnym poolingom. Prevzaté z [36].

a variance pooling, teda spočítajú rozptyl výstupu každého filtru. Pri použití viacerých konvolučných vrstiev sa tieto štatistiky počítajú pre každú vrstvu a na konci sa konkatenujú do jedného vektoru rysov, ktorý sa klasifikuje.

Druhý typ je multi-resolution pooling. Používa sa ako lokálny pooling pre grafové signály. Počíta sa tak, že sa vytvorí množina bodov definovanej veľkosti, ktoré sú navzájom čo najďalej od seba. K nim sa potom pridelia vrcholy, ktoré sú im najbližšie. Z takto vytvorených skupín bodov sa spočíta max-pooling.

Architektúra siete

Pomocou vyššie definovaných operácií konvolúcie a poolingu je možné vytvárať architektúry podobné klasickým konvolučným sieťam. Obrázok 3.13 ukazuje príklad dvoch týchto sietí s variantami globálneho aj multi-resolution poolingu.

3.3 Attention mechanizmus

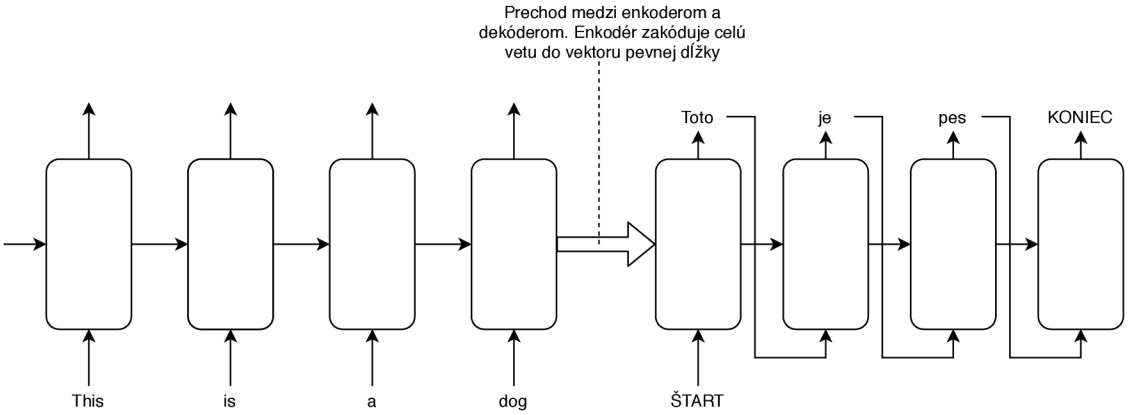
V tejto sekcii popisujem základný princíp fungovania attention mechanizmu a transformer architektúru, ktorá je na ňom postavená. Táto sieť sa zatiaľ nepoužíva na spracovanie 3D modelu, ale v experimentoch na odhad rotácie priamo z polygoniálneho modelu (viz 4.3.3) bola použitá.

3.3.1 Princípy attention mechanizmu

Táto technika sa používa hlavne v sieťach s encoder-decoder architektúrou, ktoré sa používajú na analýzu reči, napríklad preklad z jedného jazyka do druhého. Enkóder slúži na zakódovanie sekvencie na vektor rysov. Dekóder zasa slúži na dekódovanie vektoru rysov na inú (alebo aj rovnakú) sekvenciu.

Zavedenie attention mechanizmu do sietí, ktoré spracovávajú reč bolo z toho dôvodu, že enkóder produkoval ako svoj výstup tzv. **kontextový vektor** [28]. Tento vektor mal pevnú dĺžku. Nezáležalo teda na tom, ako dlhá je sekvencia, ktorá vstupovala do enkóderu, vždy sa komprimovala na vektor pevnej veľkosti. Takáto sieť je znázornená na obrázku 3.14.

Attention mechanizmus zaviedol zmenu, že sa negeneroval jeden kontextový vektor pre celú sekvenciu, ale pre jej jednotlivé položky (napríklad pre každé slovo vo vete). Tentokrát



Obr. 3.14: Príklad sequence-to-sequence architektúry, v ktorej je problé, že enkodér musí rôzne dlhé vety zakódovať do rovnako dlhého vektoru.

sa architektúra zmení len v tom zmysle, že výstup enkóderu bude m vnútorných stavov a vstup dekóderu bude m kontextových vektorov. Tento kontextový vektor sa môže počítať viacerými spôsobmi. najčastejšie sú aditívny a multiplikatívny spôsob. Na vysvetlenie samotného mechanizmu v tejto sekcii predstavím aditívny spôsob. Transformer architektúra popísaná v 3.3.2 používa multiplikatívny spôsob.

Attention mechanizmus definuje, ako sa vypočítajú kontextové vektory (rovnica 3.7). Je to suma súčinov attention skóre (α_{ij}) a všetkých vnútorných stavov enkóderu h_j.

$$c_i = \sum_{j=1}^{m} \alpha_{ij} h_j \tag{3.7}$$

Skóre α_{ij} sa vypočíta podľa rovnice 3.8 [8].

$$\alpha_{ij} = \frac{e^{e_{ij}}}{\sum_{k=1}^{m} e^{e_{ik}}} \tag{3.8}$$

Hodnota e_{ij} je daná v rovnici 3.9. Predstavuje model (neurónová sieť), ktorého výstup je skóre. Toto skóre udáva ako veľmi súhlasia vstup na pozícii j s výstupom na pozícii i. Vstup tohto modelu je j-ty vnútorný stav enkóderu a $i-1$ vnútorný stav dekóderu, ktorý sa nachádza tesne predtým ako sa dá na výstup y_i.

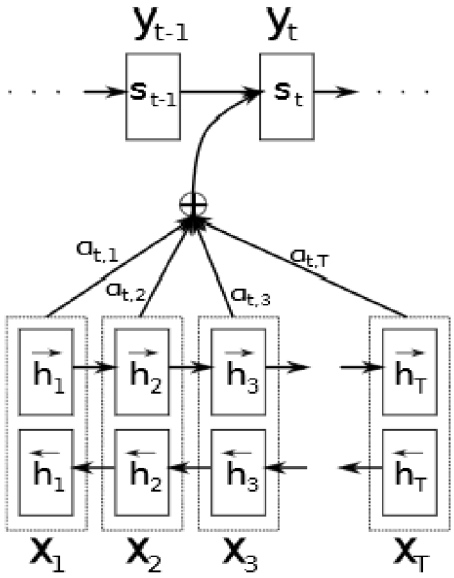
$$e_{ij} = a(s_{i-1}, h_j) \tag{3.9}$$

Zavedenie attention do encoder-decoder siete s použitím rekurentných sietí je na obrázku 3.15.

Attention vrstva teda pre aktuálne spracovávanú položku sekvencie zistí, ktoré iné položky s ňou najviac súvisia. Položku potom zakóduje tak, aby sa vo výslednom vektore rysov prejavili aj ďalšie položky, ktoré na ňu majú vplyv. Zavedenie attention mechanizmu do dekóderu bude znamenať, že dekóder bude "venovať viac pozornosti"niektorým položkám vstupu.

3.3.2 Transformer architektúra

Transformer [31] je architektúra, ktorá bola navrhnutá na využitie hlavne v oblasti spracovania prirodzeného jazyka. Je založená na attention mechanizme. Tu ale použili tzv.



Obr. 3.15: Attention mechanizmus v encoder-dekocer sieti. Enkóder tu predstavujú spodné vrstvy h_1 až h_T. Dekóder zase vrstvy $s_t - 1$ a s_t. Kontextový vektor pre jeden výstup y_t sa počíta zo všetkých vrstiev enkóderu h_1 až h_T. Prevzaté z [8].

multiplikatívny attention. Celá architektúra sa skladá z niekoľkých blokov enkóderu nasledovaných blokmi dekóderu.. Ako vidno z obrázka 3.16, v architektúre sa nenachádzajú žiadne rekurentné vrstvy, je poskladaná len z attention blokov a dopredných sietí.

V enkóderi aj dekóderi je použitý self-attention, ktorý sa vypočíta podľa 3.10.

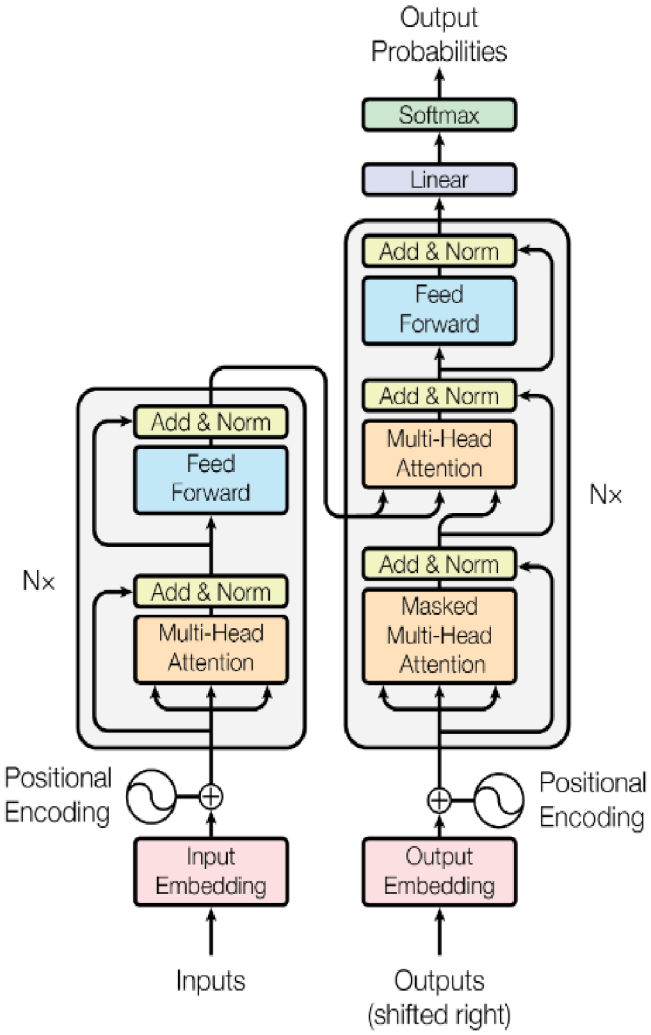
$$Attention(Q, K, V) = softmax(QK^T)V \tag{3.10}$$

Vektor Q a matice K a V sa označujú aj ako **query**, **key** a **value**. Na query sa dá pozerať ako na dotaz do databáze, K predstavuje identifikátory jednotlivých položiek databázy a V predstavuje atribúty. Po vypočítaní $softmax(QK^T)$ získame pravdepodobnostné rozloženie, ktoré bude mať najvyššie hodnoty tam, kde je Q podobné K. Teda sme akoby v databázi našli tie položky, ktoré sa najviac podobajú hľadanej hodnote Q. Výsledok sa násobí s maticou V, aby sme tým zmenili jej hodnoty podľa podobnosti jednotlivých položiek. Ak si budú málo podobné, číslo bude blízko nule a teda aj výsledok bude malé číslo. Ak si budú veľmi podobné, výsledok bude vyššie číslo. Attention vrstva je znázornená na obrázku 3.18.

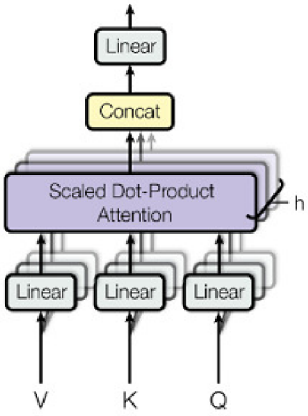
Multi-head attention

je rozšírenie attention vrstvy popísanej vyššie. V transformer architektúre sa používa preto, že umožňuje modelu, aby sa zameriaval na rôzne reprezentácie na rôznych pozíciách. Namiesto jednej matice Q, K a V ich teraz je h. Každá matica Q_i, K_i, V_i sa spočíta tak, že sa Q, K a V vynásobia váhovými maticami W_{Q_i}, W_{K_i} a W_{V_i}, ktoré predstavujú určitú transformáciu. Potom sa osobitne spočíta h samostatných attention výstupov. Všetky výstupy sa konkatenujú a vynásobia váhovou maticou W^O:

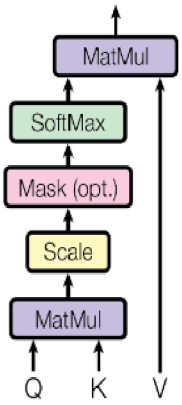
$$MultiHead = Concat(head_1, head_2, ..., head_h)W^O$$



Obr. 3.16: Architektúra transformer siete. Enkóder dostane na vstup zakódovanú sekvenciu. Dekóder dostane na vstup už vygenerovanú časť výslednej sekvencie, ktorú najskôr spracuje pomocou multi-head attention vrstvy. Výstup tejto vrstvy sa použije ako Q vstup pre ďalšiu vrstvu. K a V pre túto vrstvu sa zoberú z výstupu enkóderu. Prevzaté z [31].



Obr. 3.17: Multi-head attention vrstva [31].



Obr. 3.18: Operácie v attention vrstve [31].

$$head_i = Attention(QW_{Q_i}, KW_{K_i}, VW_{V_i})$$

Tým dostaneme jeden výstup, v ktorom sú obsiahnuté informácie zo všetkých h attention vrstiev. Multi-head attention vrstva je znázornená na obrázku 3.17.

Prvá multi-head attention vrstva v dekóderi sa trochu líši od ďalších dvoch, ktoré sú naznačené na obrázku 3.16. Je iná v tom, že tu chceme spočítať attention nie pre všetky položky sekvencie, ale len pre tie, ktoré sú spracované do aktuálneho indexu. Ak sme v sekvencii dekódovali len prvých 5 položiek, dekóder by sa nemal pozerať na tie, ktoré sú za nimi, lebo by na výstup dával tieto neskoršie položky a nič by sa nenaučil. Preto sa tá časť sekvencie, ktorú nechceme spracovávať vymaskuje. Výstup dekóderu nám dá pravdepodobnosti toho, aká ďalšia položka by mala byť na výstupe.

Kapitola 4

Navrhnuté metódy analýzy 3D modelov

V tejto kapitole popisujem navrhnuté metódy analýzy modelov. Vyskúšala som 3 metódy, prvá používala rasterizáciu a zobrazenie modelu do 2D obrázka. Druhá metóda využíva prevedenie modelu do sekvencie a attention mechanizmus. Tretia metóda skúša analyzovať model reprezentovaný pomocou grafu.

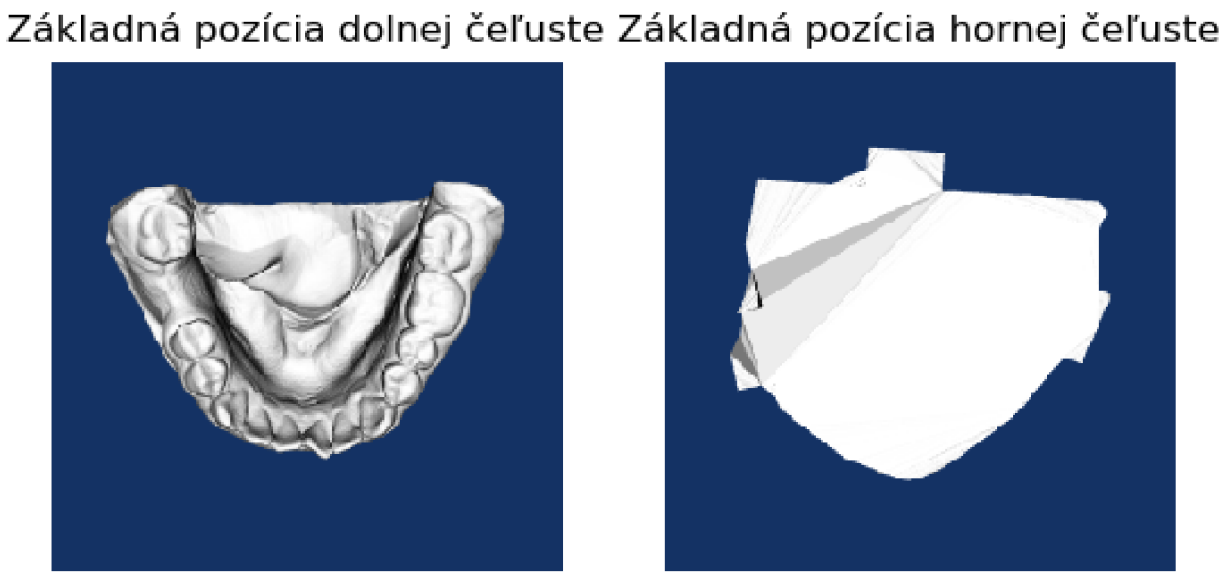
4.1 Definícia úlohy a metriky vyhodnocovania

Ako už bolo spomínané v kapitole 2, modely budem analyzovať tak, že budem odhadovať rotáciu modelu. Táto úloha je formulovaná ako regresia, výsledok siete by teda mal byť čo najpresnejší uhol, o ktorý bol model rotovaný od základnej pozície. Experimenty som vykonávala s modelmi čeľuste, ktoré mi poskytla firma Tescan3Dim. Základné pozície hornej a dolnej čeľuste sú zobrazené na obrázku 4.1. Rôzne spôsoby reprezentácie rotácie sú popísané v podkapitole 2.3. Reprezentácia, ktorú som zvolila pre experimenty je popísaná v 5.1.

Mojím cieľom bolo naučiť siete odhadovať všeobecnú rotáciu. Pri vytváraní trénovacieho a testovacieho datasetu som náhodnú všeobecnú rotáciu získala tak, že som si náhodne vygenerovala vektor o veľkosti 1 a dopočítala som ďalšie dva vektory, ktoré sú na neho kolmé. To mi udávalo rotáciu od súradnicových osí x, y a z. K tomuto spôsobu generovania rotácie som došla experimentálne. Skúšala som aj iný spôsob, kedy som si vygenerovala náhodný vektor a uhol, o ktorý sa otočí a z toho spočítala rotáciu. Tento spôsob sa však ukázal nevhodný. Po natrénovaní siete sa ukázalo, že vie lepšie odhadovať malé rotácie. To bolo spôsobené tým, že som rovnomerne generovala náhodný vektor a náhodnú rotáciu. V datasete bolo rovnako veľa malých rotácií ako veľkých. Problém však je, že pri malej rotácii okolo náhodného vektora sa model zmení veľmi málo, zatiaľ čo pri veľkej rotácii veľa. V datasete teda bolo veľa príkladov modelov, ktoré vyzerali veľmi podobe a menej tých, ktoré boli odlišné. To je ukázané na obrázku 4.2.

4.1.1 Odhad pozície bodov

Experimenty ukázali, že priamy odhad rotácie nefunguje dobre. Namiesto toho som úlohu pre sieť formulovala tak, že má odhadnúť pozíciu troch bodov. Niečo podobné robili autori siete popísanej v 3.1.1. Zvolila som si body (1,0,0), (0,1,0) a (0,0,1). Nemuseli to byť žiadne landmarky na modeli čeľuste, lebo som sa nesnažila odhadnúť rotáciu pomocou nich. Na



Obr. 4.1: Základné pozície dolnej čeľuste (vľavo) a hornej čeľuste (vpravo).

týchto modeloch by bolo aj dosť ťažké určiť nejaké body, ktoré by boli spoločné každý model, lebo boli rôznorodé, na niektorých sa dokonca nenachádzali žiadne zuby, pomocou ktorých by sa dalo orientovať. Rotácia sa potom vypočíta zo súradníc týchto bodov pomocou **singular value decomposition**. Tomuto som prispôsobila aj loss funckiu používanú pri všetkých modeloch okrem tých, ktoré odhadovali priamo rotáciu. Počítala som ju ako priemer vzdialeností medzi skutočnými a odhadnutými bodmi. Pri experimentoch s priamym odhadom rotácie som použila funkciu mean squared error.

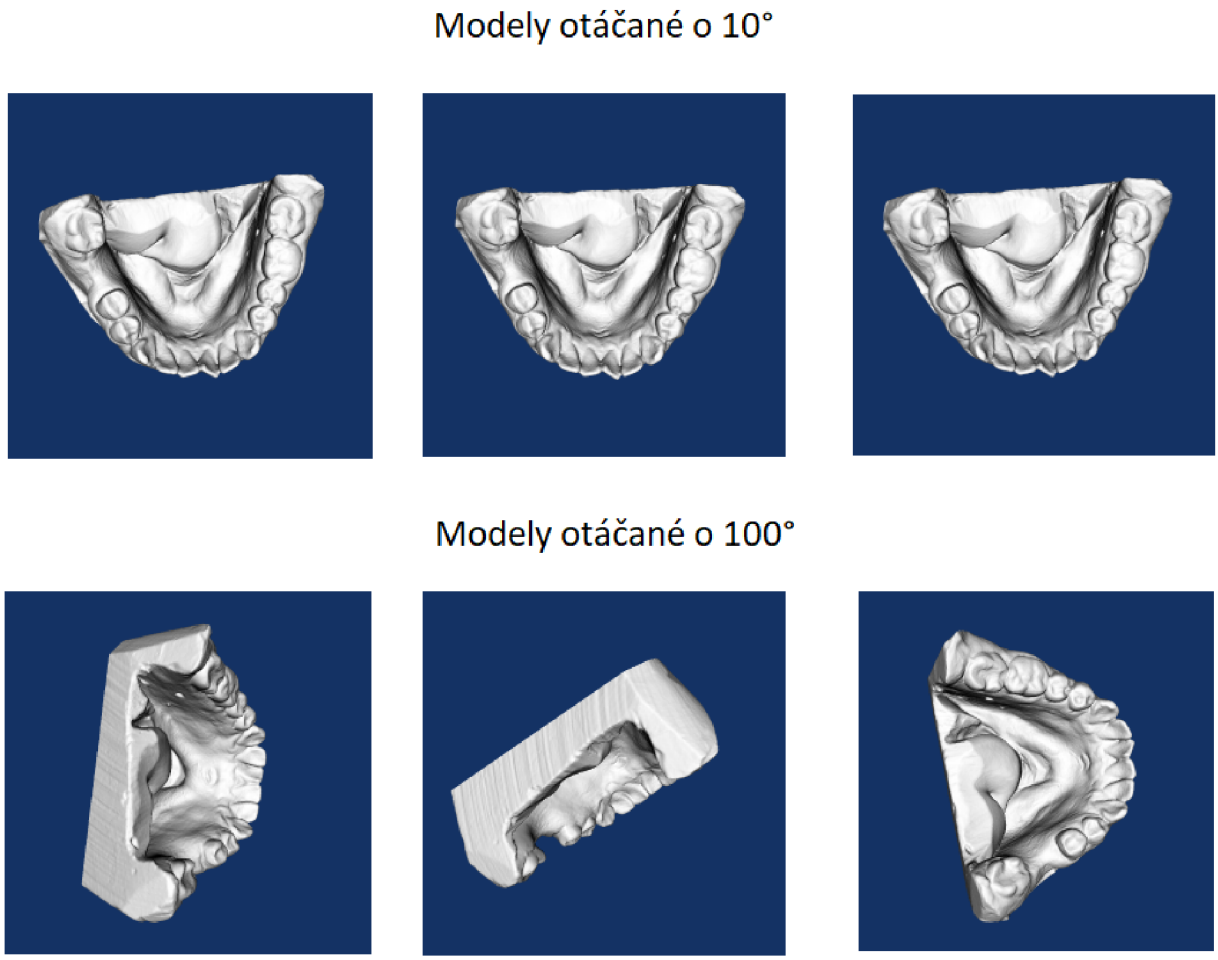
Singular value decomposition je jedna z metód, ktoré sa dajú využiť pri odhade rotácie [18, 30]. Rozkladá množinu vektorov na ortogonálne osi a dĺžku projekcií. Pri rozložení vektoru získame [6]:

1. **smer projekcie**, ktorý predstavujú jednotkové vektory, na obrázku 4.3 sú to vektory v_1 a v_2.
2. **dĺžka projekcií** zobrazená ako dĺžky s_{a1} a s_{a2}
3. **projekčné vektory** p_{a1} a p_{a2}, ktoré slúžia na rekonštrukciu pôvodného vektoru tak, že sa sčítajú.

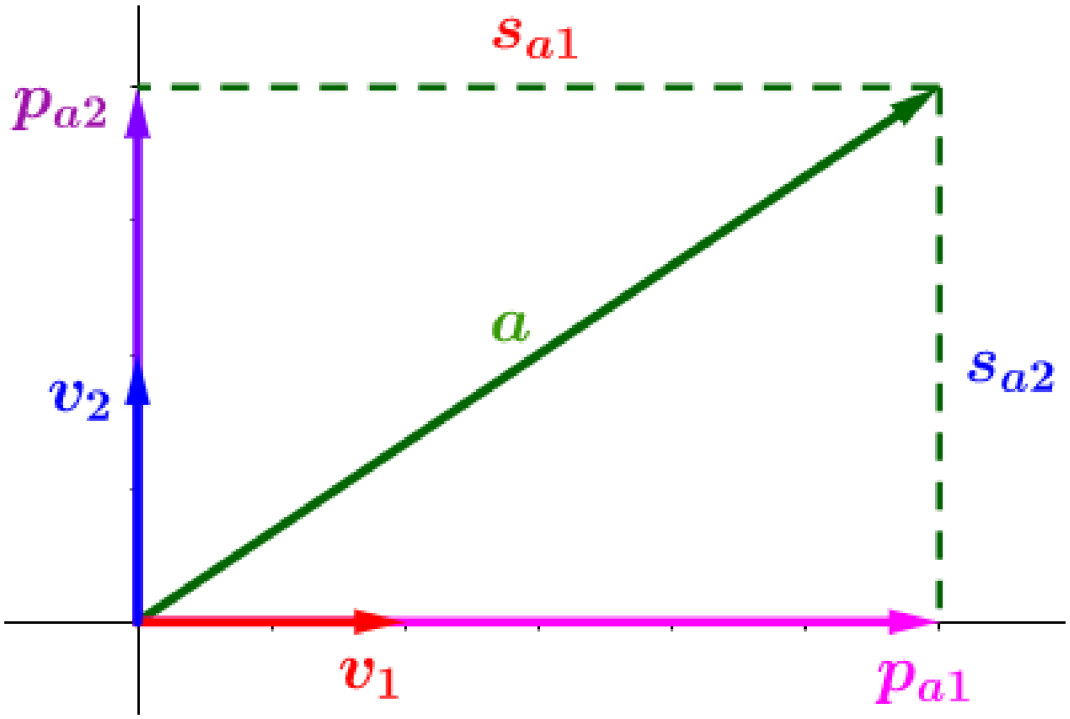
Pre viac vektorov sa používa maticový zápis jednotlivých komponent. Pre jednotkové vektory to je matica V a pre dĺžku vektorov matica S.

Rozklad vektorov sa dá napísať rovnicou 4.1. Matica U obsahuje normalizované dĺžky vektorov a Σ je matica, ktorá má len na diagonále nenulové prvky a tie majú také hodnoty, aby keď vynásobíme matice U a Σ, dostaneme maticu dĺžok vektorov S. Matica A obsahuje pôvodné vektory, ktoré chceme rozložiť.

$$A = U\Sigma V^T \tag{4.1}$$



Obr. 4.2: Príklad rotácie okolo náhodných osí. Vrchné tri obrázky predstavujú model rotovaný o 10°, spodné tri o 100°.



Obr. 4.3: Rozloženie jedného vektoru na komponenty.

Na matice U Σ a V^T sa dá pozerať aj tak, že vykonávajú transformácie [2]. Matice U a V^T vykonávajú rotáciu a matica Σ zväčšenie alebo zmenšenie. Preto, aby som získala rotáciu z tohto rozkladu, stačí vynásobiť matice U a V^T.

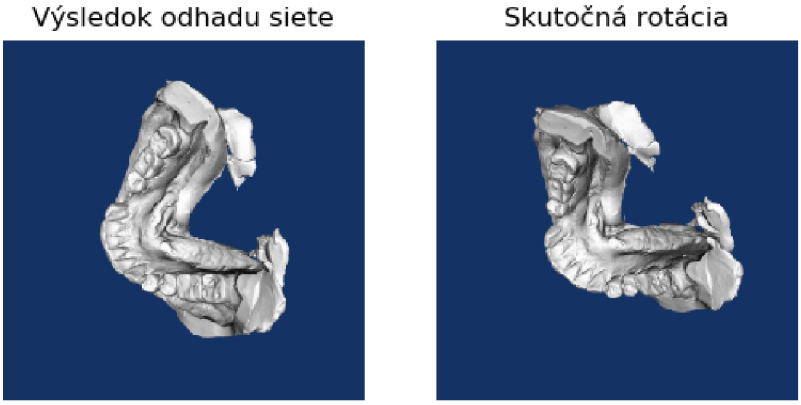
4.1.2 Metriky hodnotenia experimentov

Na vyhodnotenie úspešnosti sietí pri odhadovaní rotácie modelu som sa rozhodla použiť metriku, ktorá počíta percento prípadov z testovacích dát, v ktorých sa rozdiel odhadnutej rotácie od skutočnosti líšil o menej ako 30°, čo využíva viacero publikácií, napríklad [13]. Príklad rozdielu rotácie od skutočnosti o približne 30° sa nachádza na obrázku 4.4. Okrem toho som sledovala hodnoty kvartilov, hlavne medián z rozdielu odhadu od skutočnosti. Tieto hodnoty potom zobrazujem na boxplote a ukazujú, v akom rozsahu uhlov sieť najčastejšie chybovala.

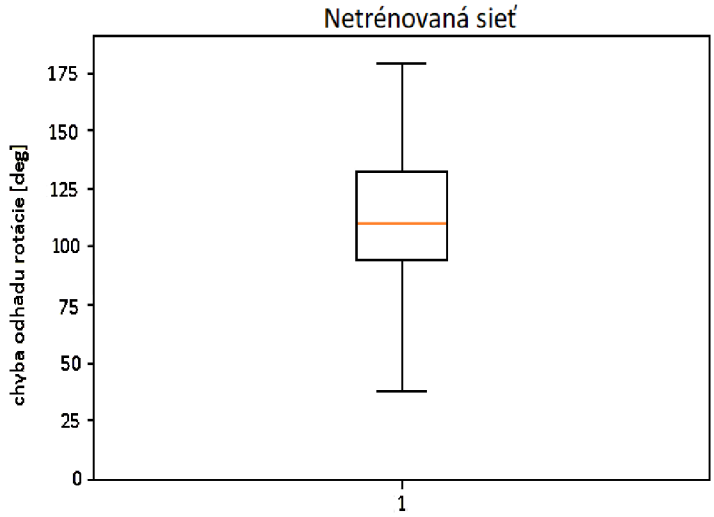
Aby som vedela určiť, či sa sieť dokáže naučiť niečo rozumné a ako dobre, pozrela som sa na mnou zavedenú metriku pre sieť, ktorá nebola vôbec natrénovaná, teda na výstup dávala náhodné výsledky. Na testovacom datasete dosiahla úspešnosť 1.47%, medián bol 109°. Boxplot pre náhodné hádanie rotácie je zobrazený na obrázku 4.5.

4.2 Datasety

Ako už bolo spomínané, môj dataset obsahoval modely hornej a dolnej čeľuste. Celkovo som mala k dispozícii približne 500 rôznych modelov. Keďže modely boli exportované zo softwaru, ktorý používajú stomatológovia, neboli zarovnané v rovnakej pozícii. Mala som k dispozícii XML súbor, v ktorom bola zaznamenaná transformácia modelu. Z týchto 500 modelov bolo treba vybrať tie, ktoré boli vhodné na použitie na trénovanie. V niektorých



Obr. 4.4: Príklad odhadu, ktorý sa líši od skutočnosti zhruba o 30°. Vľavo je odhad siete, vpravo je skutočná rotácia.

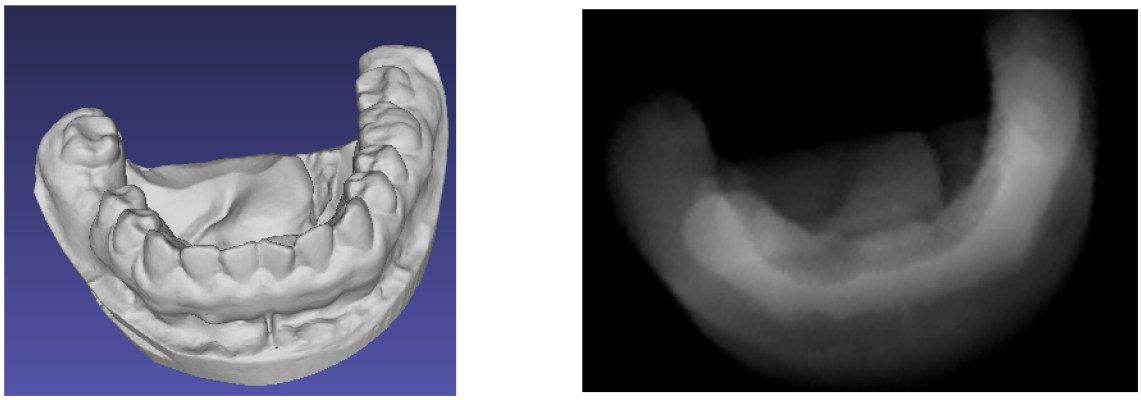


Obr. 4.5: Boxplot pre netrénovanú sieť.

prípadoch, keď som model zobrazovala v pôvodnej polohe pomocou inverznej transformácie, tak nie vždy bola poloha modelov rovnaká. Nakoniec som vo všetkých datasetoch použila približne 40 rôznych modelov a pre každý som vygenerovala 1000 náhodných rotácií.

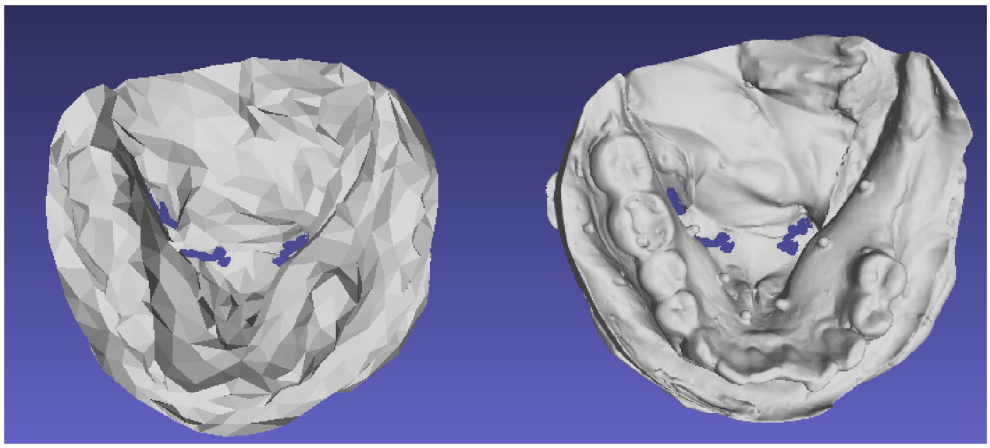
Pre metódu pracujúcu s rasterizáciou a zobrazením modelu na 2D röntgenový obrázok som generovala dataset tak, že som si model najskôr voxelizovala a následne zobrazila a rotovala pomocou knižnice VTK. Obrázky mali veľkosť 256x256 pixelov. Ako je vidieť na obrázku 4.6, pre človeka je viac viditeľná rotácia na obyčajnom renderovanom modeli, pre neurónové siete to však nemusí platiť. Z röntgenového snímku môžu vyčítať intenzity a tým mať lepšiu predstavu o polohe modelu v priestore.

Dataset pre metódu, ktorá reprezentuje model ako sekvenciu som generovala tak, že som zmenšila veľkosť modelu, čo sa týka počtu polygónov na 3500. Pôvodné modely mali aj niekoľko stotisíc polygónov. Robila som to pomocou metódy quadratic edge colapse decimation. Na jej spočítanie som využila nástroj MeshLab. Tento počet polygónov bol zvolený tak, aby výpočetná náročnosť siete nebola veľmi veľká a zároveň sa príliš nedeformoval tvar



Obr. 4.6: Porovnanie renderovaného modelu (vľavo) a modelu zobrazeného ako röntgenový snímok (vpravo).

modelu. Rozdiel medzi originálnym modelom a tým so zmenšeným počtom polygónov je na obrázku 4.7. Model som previedla na sekvenciu tak, že jedna položka predstavovala jeden polygón. Ten bol popísaný normálou a súradnicami troch bodov, ktoré tvorili popisovaný trojuholník. Okrem tejto reprezentácie som skúšala aj takú, kde jedna položka sekvencie predstavovala jeden bod modelu. Tento bod bol popísaný svojimi súradnicami a normálou, ktorá bola spočítaná ako priemer normál trojuholníkov, do ktorých tento bod patril. Problém s touto reprezentáciou bol v tom, že hoci mali všetky modely rovnaký počet trojuholníkov, neznamenalo to, že mali rovnaký počet bodov. Preto aby som mala sekvencie rovnakej dĺžky, musela som ich dopĺňať nulovými položkami. Experimenty ukázali, že natrénovaná sieť dáva horšie výsledky na datasete so sekvenciami, ktoré boli doplnené o nuly.



Obr. 4.7: Porovnanie pôvodného modelu (vpravo) a zjednodušeného modelu na menší počet polygónov (vľavo).

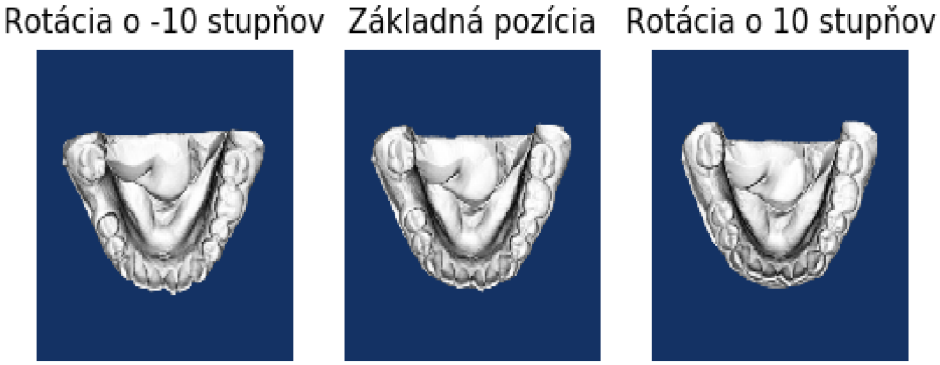
Dataset pre metódu pracujúcu s grafovou reprezentáciou som tvorila tak, že som pre každý trojuholník zaznamenala pozíciu bodu, ktorý sa vypočítal ako priemer súradníc vrcholov tohto trojuholníka.

4.3 Návrh riešenia

V tejto sekcii popisujem architektúry sietí pre odhad rotácie modelu pomocou rôznych jeho reprezentácií. Prvá metóda to rieši rasterizáciou na 2D obrázok a s použitím troch pohľadov na model. Druhá reprezentuje model pomocou sekvencie a tretia pomocou grafu.

4.3.1 Metóda pracujúca s rasterizáciou

Keďže existujúce riešenia odhadu 3D rotácie používali projekciu modelu do 2D, použila som tento spôsob ako baseline metódu. Najčastejšie používané architektúry pre túto úlohu boli VGG a ResNet. Skúsila som oba typy architektúr a experimenty ukázali, že je lepšie použiť ResNet. Taktiež som zistila, že dosahujem lepšie výsledky, keď použijem 3 pohľady na model namiesto jedného. Z jedného pohľadu je niekedy ťažké určiť presnú rotáciu, lebo pri malých rotáciách modely vyzerajú veľmi podobne (viz obrázok 4.8) Zvolila som pohľad spredu, zboku a zhora, čo je zobrazené na obrázku 4.9.

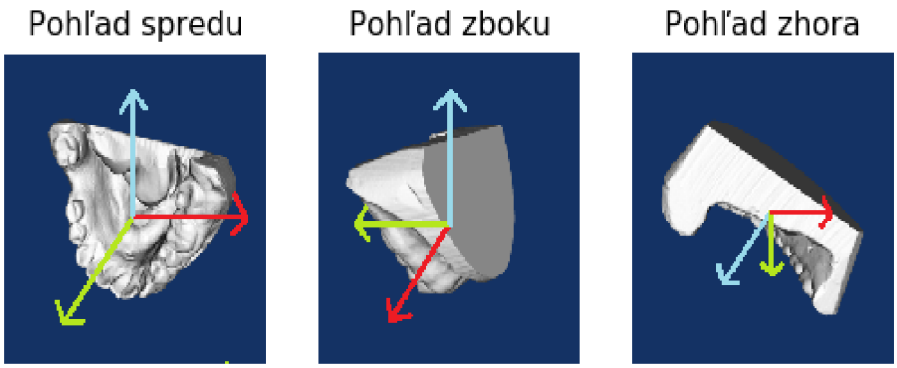


Obr. 4.8: Ukážka rotácie podľa jednej osi. Vľavo je rotácia o -10°, v strede sa nachádza základná pozícia modelu a vpravo je rotácia o +10°. Rozdiely v obrázkoch sú veľmi malé a ani človek by nebol schopný presne povedať ako veľmi je model natočený.

Lepšie sa taktiež ukázalo použitie röntgenového zobrazenia než obyčajného. Navrhnutá sieť je zobrazená na obrázku 4.10. Keďže vygenerované röntgenové snímky sú pre človeka dosť neprehľadné, ako vstup siete je na obrázku model zobrazený pomocou obyčajného renderovania, aby bolo lepšie vidieť 3 pohľady.

4.3.2 Metóda spracovávajúca model reprezentovaný grafom

Pri tejto metóde som použila architektúru siet popísanú v 3.2.4. Jeden uzol grafu predstavoval jeden polygón modelu. V pôvodnej architektúre autori ako vstupný vektor rysov používali len súradnice bodu. Ja som pre každý polygón vytvorila jeden bod ako priemer súradníc jeho vrcholov a pridala k súradniciam normálu polygónu. Tento vektor potom predstavoval signál pre jeden uzol grafu. Sieť som zmenila tak, aby počítala regresiu a nie klasifikáciu tým, že som zmenila výstupnú aktivačnú funkciu a loss. Použila som globálny pooling.

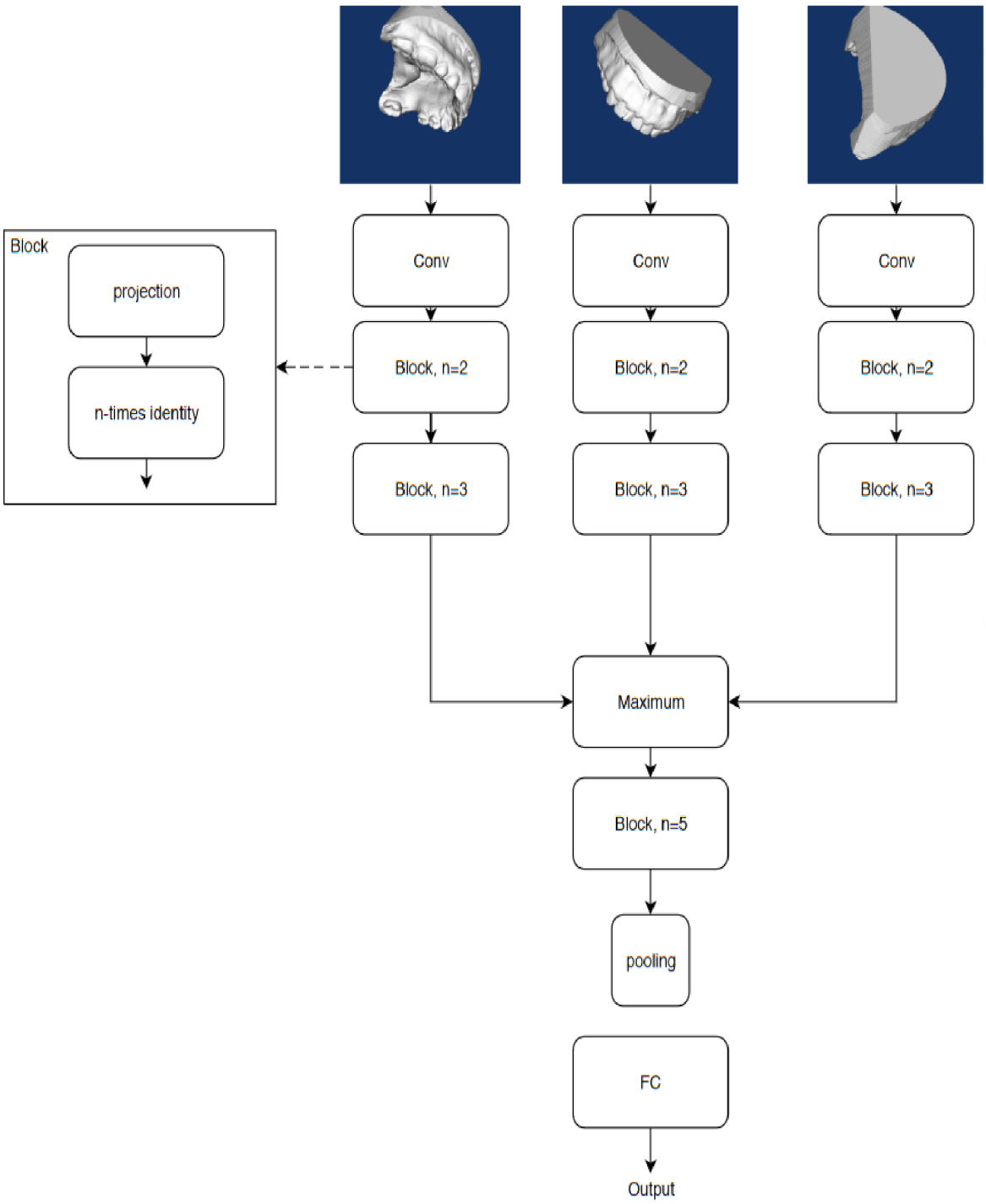


Obr. 4.9: Pohľady na jeden model. Vľavo je pohľad spredu, v strede pohľad zboku a vpravo pohľad zhora. Smer šípok naznačuje, kde sa nachádzajú zvyšné dva pohľady. Zelená ukazuje pohľad spredu, červená zboku a modrá zhora.

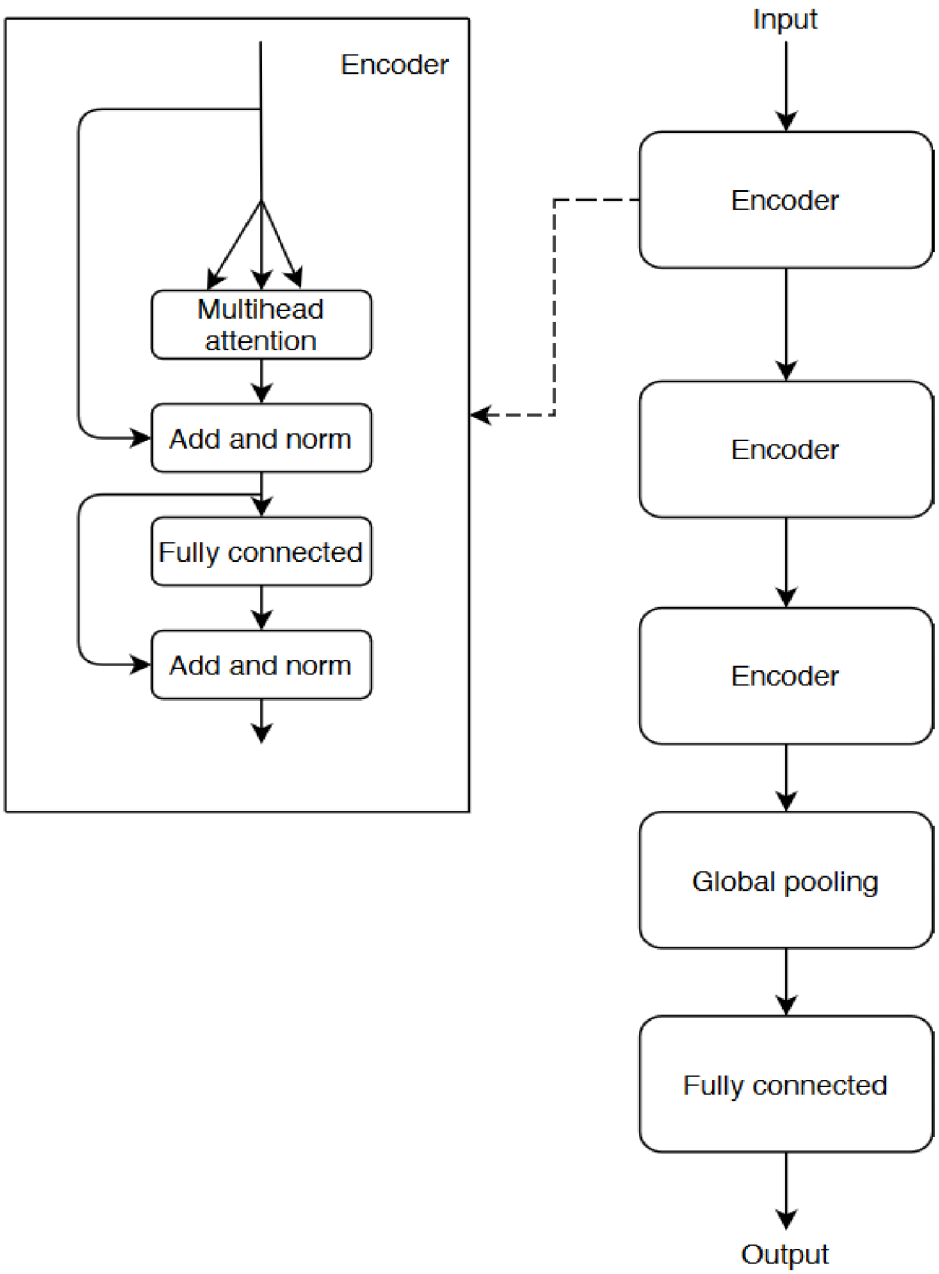
4.3.3 Metóda spracovávajúca model ako sekvenciu

Pri návrhu tejto metódy som sa nechala inšpirovať transformer architektúrou popísanou v 3.3.2. Pre moje účely stačilo, aby sa sieť naučila zo sekvencie extrahovať rysy a na základe nich určiť rotáciu modelu. Preto som v architektúre využila len enkóder časť z celého transformeru. Architektúra mojej siete je na obrázku 4.11. Experimenty s ňou ukázali, že pri výpočte attention sa násobili matice veľkosti 3500x3500. Trénovanie teda zaberalo veľa pamäte a nemohla som využívať batche väčšie ako 4. Tým pádom aj jedna trénovacia epocha trvala dlho, približne 1.5 hodiny.

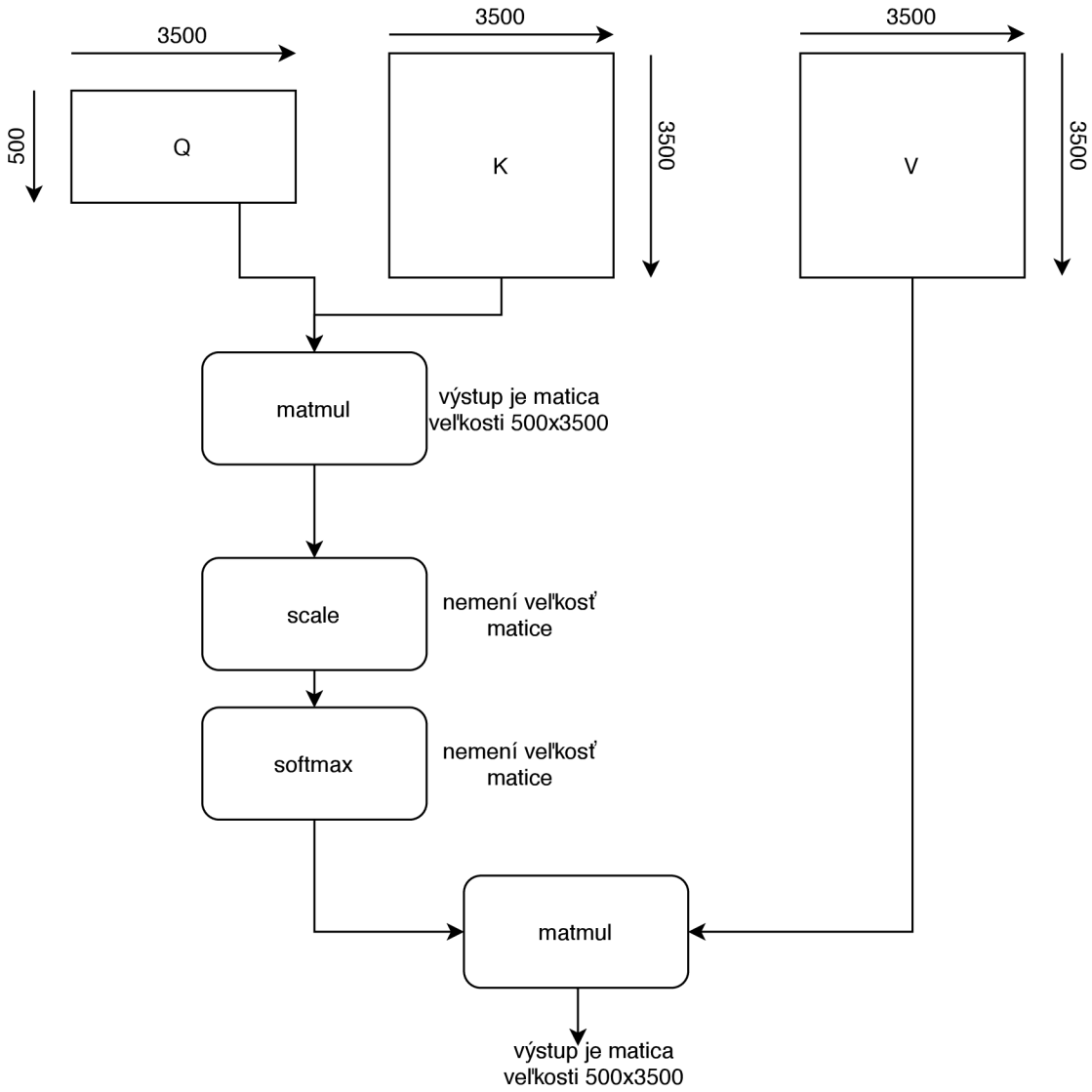
Skúsila som do architektúry zaviesť zmenu v podobe pooling operácie, ktorá zmenšovala veľkosť sekvencie. To mi umožnilo používať väčšie batche a vytvárať väčšiu sieť. Princíp tejto operácie spočíva v tom, že v niektorých vrstvách enkóderu nahradím self-attention obyčajným attention. V tejto vrstve si pri inicializácii siete náhodne vytvorím query vektor, ktorý má takú dĺžku, ako chcem zmenšiť sekvenciu. Po vynásobení matíc sa pôvodná matica 3500x3500 zmenší práve na definovanú veľkosť query vektora. Princíp operácie je zobrazený na obrázku 4.12. Toto mi umožnilo znížiť pamäťové nároky modelu, zvýšiť počet prvkov v jednom batchi až na 8 a znížiť čas potrebný na jednu epochu. Architektúra je znázornená na obrázku 4.13.



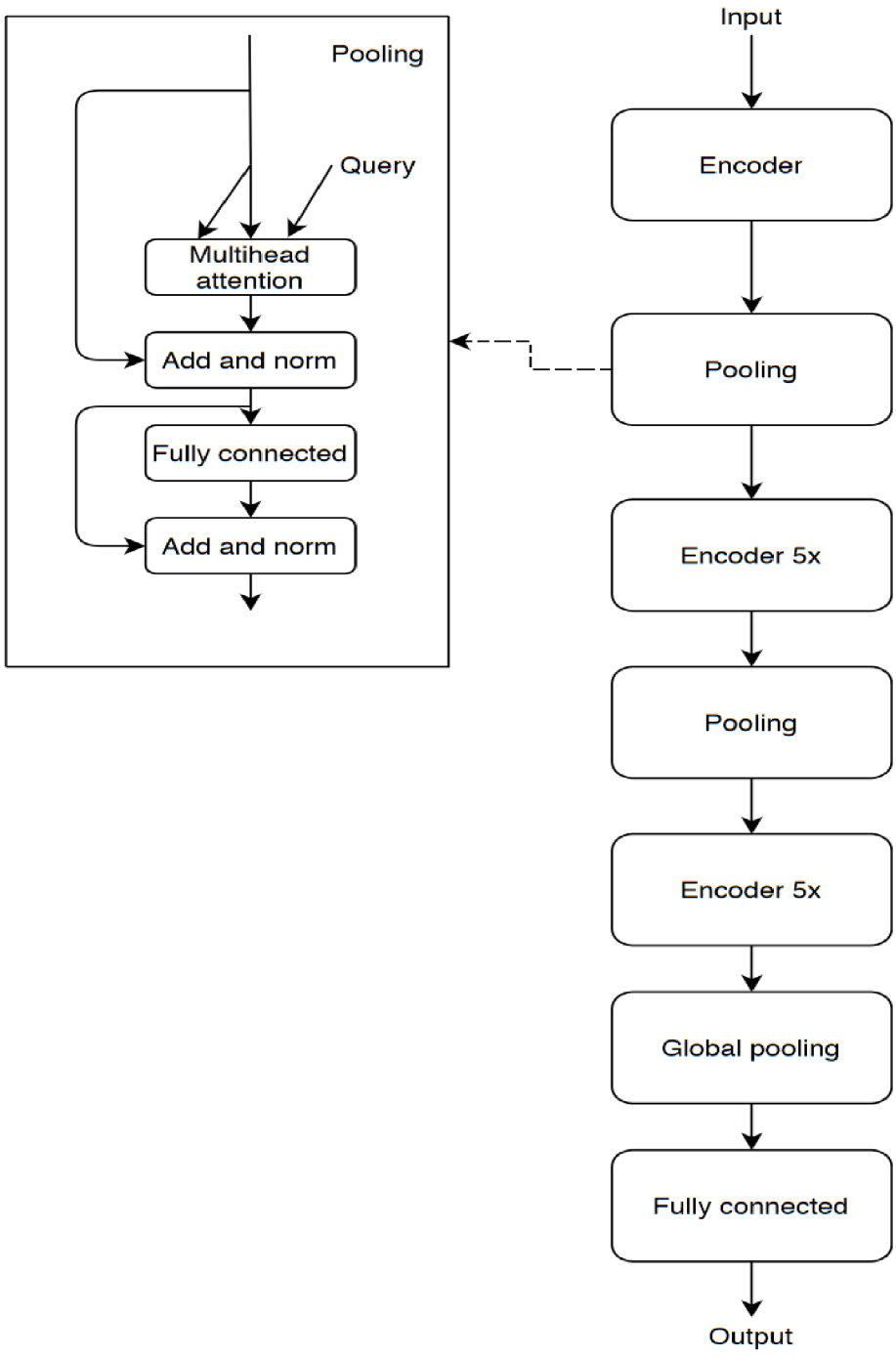
Obr. 4.10: ResNet sieť pracujúca s tromi pohľadmi na model.



Obr. 4.11: Architektúra s attention mechanizmom bez poolingu.



Obr. 4.12: Príklad fungovania pooling operácie v attention architektúre na zmenšení sekvencie veľkosti 3500x3500 na 500x3500.



Obr. 4.13: Architektúra siete s attention mechanizmom po zavedení operácie poolingu.

Kapitola 5

Experimenty a výsledky

V tejto kapitole popisujem výsledky s navrhnutými architektúrami. Vo výsledkoch udávam aj výsledky niektorých experimentov, na základe ktorých som sa rozhodovala, čo bude najlepšie fungovať pri odhade rotácie.

5.1 Reprezentácia rotácie použitá v experimentoch

Na určenie toho, aká reprezentácia rotácie je vhodná, som vykonala niekoľko experimentov, kde som generovala rotácie tak, že som vygenerovala náhodný uhol do 180° pre každú z osí x, y a z a podľa toho som rotovala model a zobrazila ho ako 2D obrázok. Skúšala som 2 architektúry, VGG a ResNet. ResNet architektúra dávala lepšie výsledky, preto som ju použila aj v ďalších experimentoch. Testovala som reprezentácie axis-angle, rotačnú maticu a kvaternióny popísané v 2.3. Ako loss funkciu som používala mean squared error. V tabuľke 5.1 sa nachádzajú výsledky experimentov. Z nich vyšlo, že najlepšie bude používať reprezentáciu pomocou rotačnej matice.

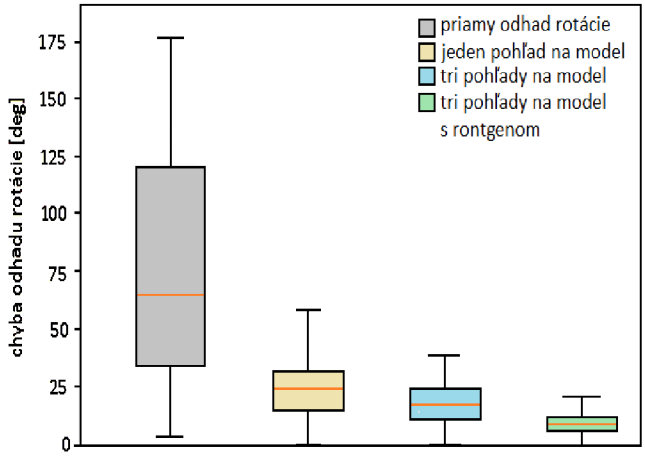
5.2 Metóda pracujúca s rasterizáciou

V sekcii 4.3.1 je popísané, že ako hlavnú baseline metódu som zvolila ResNet sieť, ktorá dostane na vstup 3 rôzne pohľady na model zobrazené ako röntgen. Táto metóda dosiahla presnosť až 99% a medián 12°. Okrem toho som vykonala ďalšie experimenty, aby som zistila, čo funguje dobre na odhad rotácie a čo nie. Výsledky všetkých experimentov sú zhrnuté v tabuľke 5.2. Výsledky hlavnej metódy sú ukázané aj na grafe 5.7 v porovnaní s ostatnými metódami. Graf zobrazuje rozloženie hodnôt predstavujúcich rozdiel odhadu siete od skutočnej rotácie. Vidno na ňom, že väčšina odchýlok sa pohybuje medzi 10° až 20°. Oranžová čiara v grafe predstavuje medián.

Čo sa týka ďalších experimentov, ukázalo sa, že priamy odhad rotácie je nevhodný. Ako bolo popísané v 5.1, tento spôsob nefungoval dobre ani na jednoduchšie formulovanej

Metrika	Rotačná matica	Axis-angle	Kvaternióny
% prípadov s rozdielom menším ako 30°	24%	19%	21%
Medián	69°	77°	73°

Tabuľka 5.1: Výsledky testovania rôznych reprezentácií rotácie.



Obr. 5.1: Porovnanie rôznych experimentov. Prvý boxplot zľava je získaný z experimentu s priamym odhadom rotácie. Zvyšné boxploty ukazujú metódy, ktoré pracovali s odhadom pozície bodu. Druhý je z experimentu s jedným pohľadom na model. Tretí je z experimentu s tromi pohľadmi na model a obyčajným renderingom. Posledný je z experimentu s tromi pohľadmi a röntgenovým zobrazením.

Metóda	% odhadov s rozdielom menej ako 30°	Medián
Priamy odhad, rotačná matica	24%	69°
Odhad troch bodov, jeden pohľad	84%	19°
Odhad troch bodov, tri pohľady	93%	13°
Odhad troch bodov, tri pohľady, röntgen	**99%**	**12°**

Tabuľka 5.2: Zhrnutie výsledkov metódy, ktorá pracovala nad 2D obrázkami.

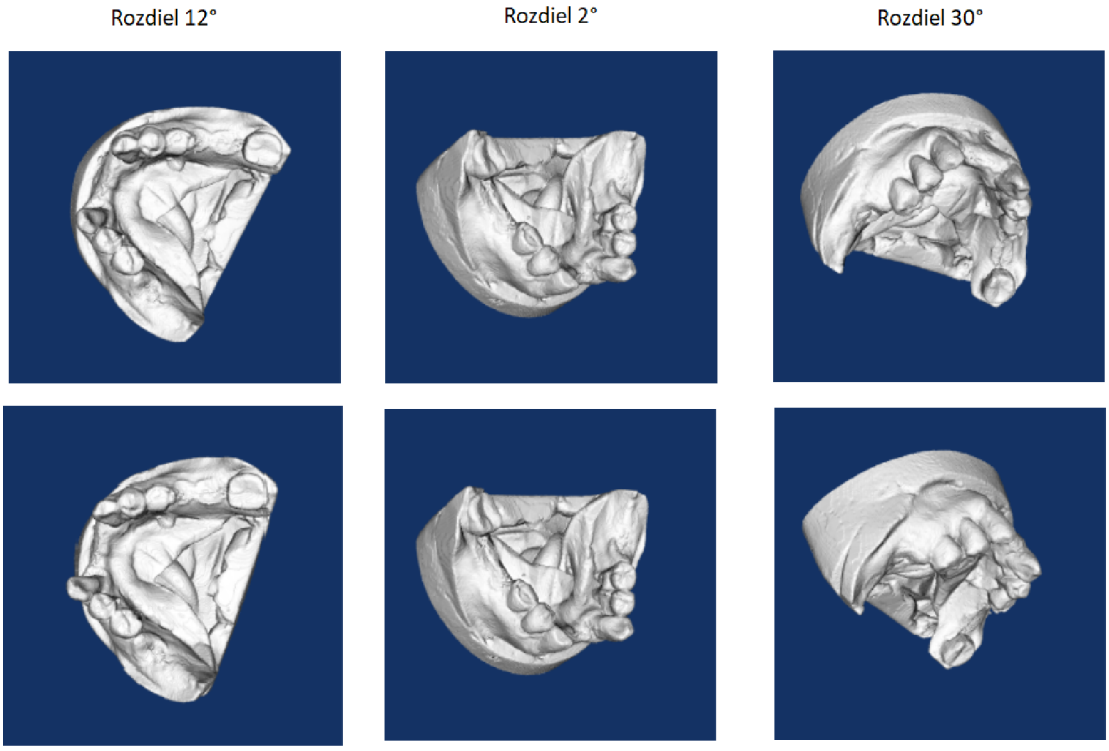
úlohe. Siete dosahovali úspešnosť najviac 24% a hodnota mediánu bola taktiež veľmi vysoká. Úprava formulácie úlohy na odhad pozície bodov priniesla veľké zlepšenie. Ak bol vstup jeden obrázok, sieti sa podarila dosiahnuť presnosť 84%, pre vstup z troch pohľadov na model sa to zvýšilo na 93% a medián 13°. Porovnanie výsledkov všetkých experimentov je na obrázku 5.1.

Zobrazenie modelov rotovaných podľa skutočnej rotácie a podľa odhadu siete sa nachádza na obrázku 5.2. Tieto vizuálne výsledky sú prevzaté z experimentu s metódou s tromi pohľadmi a röntgenovým zobrazením. Modely sú rendrované obyčajne pre lepšiu prehľadnosť.

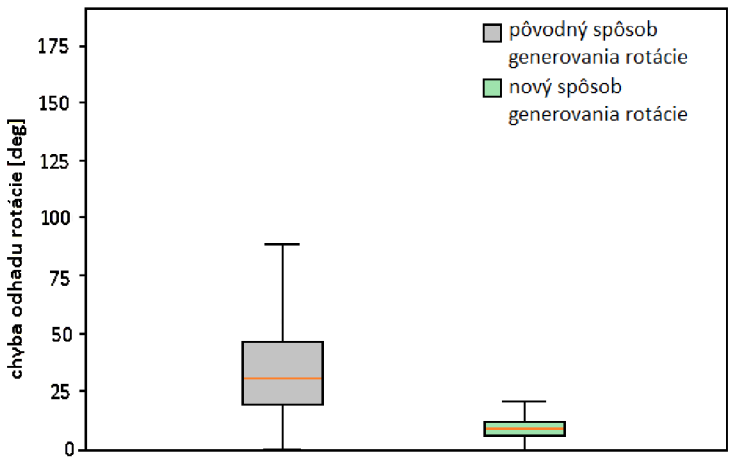
Zlepšenie výsledkov oproti pôvodnej metóde generovania rotácie zobrazujú grafy 5.3.

5.3 Metóda spracovávajúca model reprezentovaný grafom

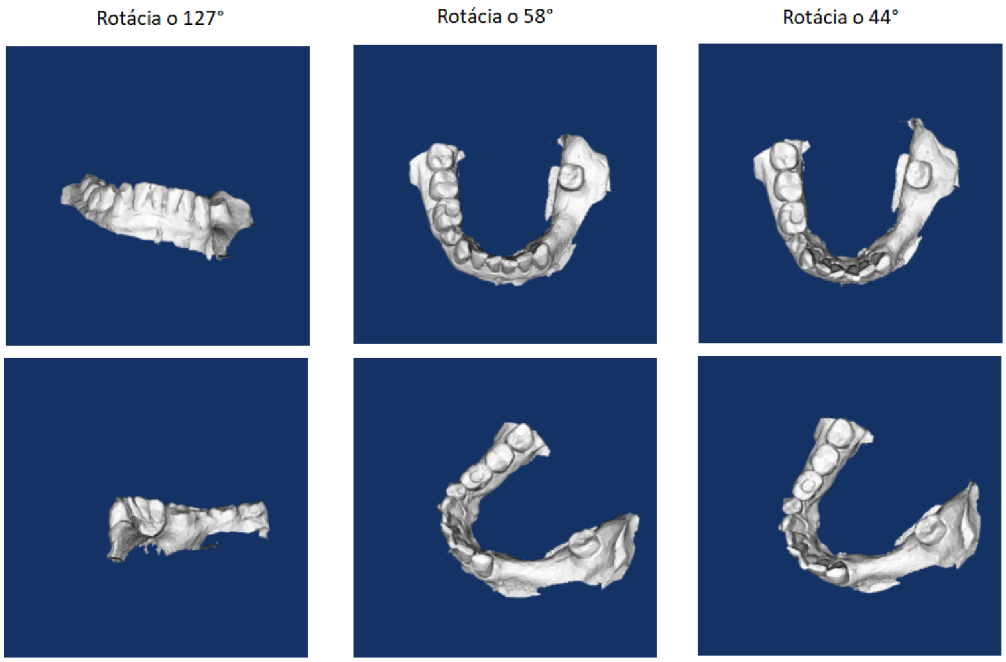
Pri experimente s úlohou definovanou ako regresia sa ukázalo, že sieť má veľký problém s pretrénovaním. Na trénovacích dátach bola schopná dosiahnuť úspešnosť 85%, ale na testovacích iba 36%. Medián na testovacích dátach bol 40°. Boxplot výsledkov na testovacích dátach je zobrazený na grafe 5.7 spolu s porovnaním oproti ostatným metódam. Príklady rozdielu rotácie od skutočnosti sú zobrazené na obrázku 5.4.



Obr. 5.2: Vizuálne výsledky baseline metódy. V hornom rade sa nachádzajú skutočné rotácie, v dolnom odhady siete.



Obr. 5.3: Porovnanie výsledkov baseline metódy pre pôvodný spôsob generovania rotácie trénovacieho a testovacieho datasetu (graf vľavo) a upravený spôsob generovania rotácie (graf vpravo).



Obr. 5.4: Zobrazenie rozdielu medzi skutočnou rotáciou a odhadom siete. Vrchný riadok predstavuje skutočnú rotáciu, spodný riadok odhad siete.

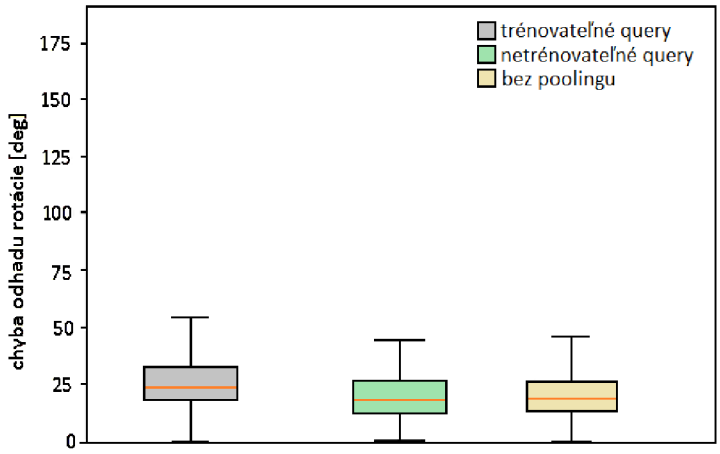
Skúsila som sa zbaviť pretrénovania tým, že som zmenšila počet parametrov siete. Pôvodná mala približne 5 miliónov parametrov. Pretrénovanie zmizlo až keď som redukovala parametre na približne 5000. Úspešnosť sa mi však nijako nepodarilo zlepšiť, naopak si myslím, že sieť mala príliš malý počet parametrov na to, aby sa naučila dobre odhadovať rotáciu.

Skúsila som úlohu preformulovať na klasifikáciu. Mojím predpokladom bolo, že architektúra sa nemusí správať dobre pri regresii. Toto som urobila tak, že som rotáciu rozdelila do intervalov po 30° a tie predstavovali kategórie. Celkovo ich teda bolo 12. Tu som ako metriku zvolila top-1 presnosť. Pri zvýšení parametrov siete sa ukázalo, že už nedochádza k takému veľkému pretrénovaniu, ale dosiahnutá presnosť siete bola veľmi malá. Na trénovacích dátach to bolo 13%, na testovacích 11%.

Sieť bola pôvodne trénovaná na rozoznávanie rôznych tvarov modelu na datasete ModelNet40. Je možné, že architektúra nie je prispôsobená na odhad rotácie a preto nedosahuje dobré výsledky. Ďalším problémom môže byť, že prevod polygoniálneho modelu na graf nie je pre túto sieť vhodný.

5.4 Metóda spracovávajúca model ako sekvenciu

S touto architektúrou som opäť vykonala niekoľko experimentov a výsledky niektorých z nich sa nachádzajú v tabuľke 5.3. Experiment, v ktorom som použila len vrstvy so self-attention ukázal, že táto sieť má veľké pamäťové nároky. Preto som navrhla zmenu popísanú v 4.3.3. S touto zmenou som vykonala dva rôzne experimenty. V prvom som náhodne vytvorené query nechala po celý čas trénovania siete rovnaké, v druhom som ich nastavila ako trénovateľné parametre. Výsledky ukazujú, že sieť sa naučila o trochu lepšie odhadnúť rotáciu v prípade, že sa query vektory nemenili. Pri porovnaní experimentu bez poolingu



Obr. 5.5: Porovnanie experimentov pre sieť s attention. Graf vľavo patrí experimentu s trénovateľnými query vektormi, prostredný experimentu s netrénovateľnými query a pravý graf patrí experimentu bez poolingu.

Architektúra	% odhadov s rozdielom menej ako 30°	Medián
Bez poolingu	87%	18°
S poolingom bez trénovania query	**88%**	**18°**
S poolingom s trénovaním query	82%	20°

Tabuľka 5.3: Zhrnutie výsledkov metódy, ktorá reprezentovala polygoniálny model ako sekvenciu a spracovávala ju pomocou attention mechanizmu. *Dataset1* predstavuje dataset, v ktorom bola chyba generovania rotácií, *Dataset2* predstavuje upravený dataset.

a s poolingom bez trénovateľných query vidno, že úspešnosť siete sa po pridaní poolingu zlepšila len veľmi málo. Zato sa znížili pamäťové nároky siete. Zatiaľ čo v sieti bez poolingu som bola schopná použiť batch maximálne veľkosti 4, v sieti s poolingom to bolo až 8.

Boxplot pre najlepší výsledok siete s poolingom bez trénovania náhodne inicializovaných query vektorov sa nachádza na obrázku 5.7. Porovnanie najlepšieho výsledku s ostatnými experimentmi sa nachádza na obrázku 5.5.

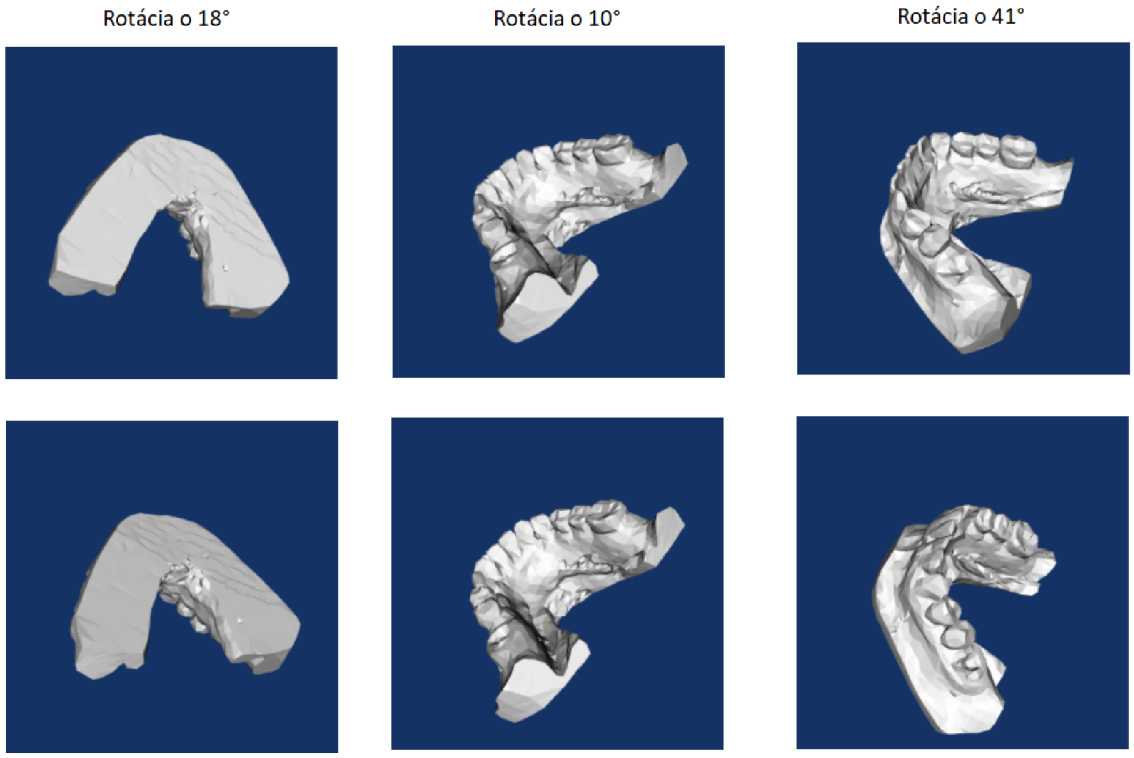
Vykonala som aj ďalšie experimenty s tým, že som po takomto natrénovaní sietí zmenšila learning rate z 0.001 na 0.00001. Zlepšenie to však neprinieslo, okrem toho trénovanie sa tak spomalilo, že by trvalo dni, možno až týždne, kým by sa niečo zmenilo. Preto som v tomto experimente nepokračovala.

Vizuálne zobrazené výsledky rozsahu chýb siete sú zobrazené na obrázku 5.6.

Tabuľka 5.3 zhŕňa výsledky s touto architektúrou.

5.5 Porovnanie s baseline metódou

V tejto sekcii porovnávam metódy pracujúce priamo nad polygoniálnym modelom s baseline metódou, ktorá spracováva rasterizovanú reprezentáciu modelu.



Obr. 5.6: Rozdiel medzi skutočnou rotáciou a odhadom. Horný riadok predstavuje skutočnú rotáciu, dolný odhad siete.

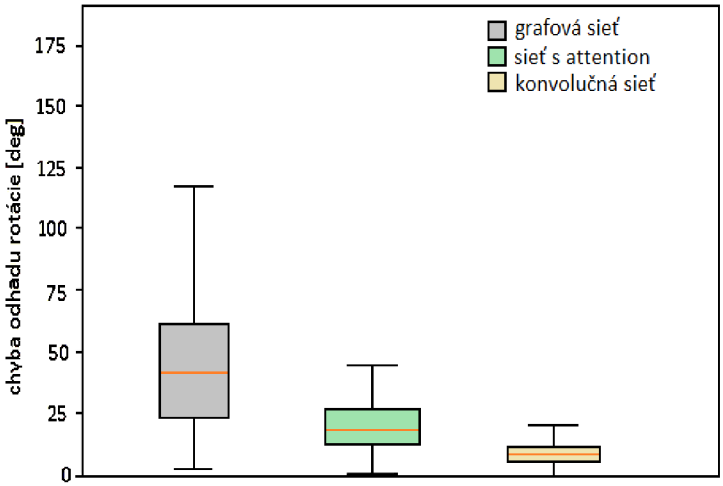
5.5.1 Porovnanie baseline metódy a grafovej siete

Metóda využívajúca grafové konvolučné siete oproti ostatným metódam mala veľký problém s pretrénovaním. Hlavne preto dosiahla oveľa horšie výsledky ako baseline metóda. Počet trénovateľných parametrov bol približne 5 miliónov, čo je porovnateľné s baseline metódou. Ako už bolo popísané v 5.3, zníženie trénovateľných parametrov síce viedlo k zníženiu problému pretrénovania, ale na druhej strane sa zhoršili výsledky siete. Pri trénovaní som nezaznamenala žiadne problémy so zvýšenými nárokmi na pamäť tak, ako pri attention sieti. Oproti baseline metóde sa však zvýšila doba trénovania jednej epochy. Trvala približne 2 hodiny. Čo sa týka inferencie, pre grafovú sieť bola doba 8s pre jeden vstup, pre baseline metódu 7 sekúnd.

5.5.2 Porovnanie baseline metódy a siete s attention mechanizmom

Priame spracovanie polygoniálneho modelu pomocou attention mechanizmu dosiahlo približne o 10% horšie výsledky než baseline metóda pri počítaní odchýlky menšej ako 30°. Architektúra v baseline metóde však obsahovala niekoľko miliónov trénovateľných parametrov, zatiaľ čo táto architektúra len približne 50 000. Stačilo to na dosiahnutie pomerne dobrého výsledku s touto metódou a neviedlo to k takmer žiadnemu pretrénovaniu. Počet trénovateľných parametrov by sa dal zvýšiť tým, že pri poolingu nastavím náhodne inicializované query ako trénovateľné. Experimenty ukázali, že to k zlepšeniu výsledkov nevedie. Je však možné, že keby som sieť trénovala oveľa dlhšie, tak by sa rozdiel na výsledku prejavil a celková úspešnosť sa zvýšila ešte o pár percent. Oproti baseline metóde táto architektúra mala oveľa vyššie pamäťové nároky. Je to spôsobené tým, že sa v attention sieti počíta násobenie veľmi veľkých matíc (3500x3500). Kvôli tomu som nebola schopná použiť batche väčšie než 8 prvkov. Toto bol jeden z dôvodov, prečo sa čas na jednu epochu trénovania výrazne zvýšil. Pri baseline metóde a rovnakom počte trénovacích dát, trvala jedna epocha 13 minút, pri tejto architektúre 1 až 1.5 hodiny. Inferencia siete pre jeden vstup trvala 15 sekúnd, pre baseline metódu to bolo 7 sekúnd. Hoci metóda s attention mechanizmom nedosiahla tak dobré výsledky ako baseline metóda a mala väčšie pamäťové a časové nároky, oproti metódam, ktoré pracujú s voxelizovanými modelmi, nepoužívam náročnú operáciu 3D konvolúcie. Vyskytuje sa tu síce násobenie matíc s veľkými rozmermi, ale tie sa mi podarilo redukovať navrhnutou operáciou poolingu, ktorá dokonca dosiahla najlepšie výsledky zo všetkých variácií, s ktorými som robila experimenty. Preto celkovo túto metódu hodnotím ako veľmi dobrú a úspešnú na priame spracovanie polygoniálnych modelov.

Porovnanie boxplotov pre všetky tri metódy je na obrázku 5.7.



Obr. 5.7: Porovnanie výsledkov grafovej siete a baseline metódy.

Kapitola 6

Záver

V tejto práci som sa zaoberala spracovaním polygoniálnych modelov pomocou neurónových sietí. Úlohou, ktorú som riešila, bolo odhadnúť rotáciu 3D modelu čeľuste človeka. Základná metóda, ktorú som zvolila pre túto úlohu, bola zobraziť si model ako 2D obrázok a ten spracovávať pomocou neurónovej siete.

Navrhla som 3 rôzne metódy, ktoré pracovali s rôznou reprezentáciou modelu. Prvá metóda rasterizovala model a zobrazovala ho do 2D obrázka. Na rôznych experimentoch sa ukázalo, že je lepšie úlohu formulovať ako dohad pozície troch bodov a nie ako priamy odhad rotácie (popísané v 4.1.1). Taktiež sa ukázalo lepšie použiť röntgenové zobrazenie modelu oproti obyčajnému. Výsledná architektúra pracovala s tromi pohľadmi na rotovaný model. Tri pohľady opäť zlepšili úspešnosť siete oproti jednému pohľadu na model.

Druhá metóda pracovala s grafovou reprezentáciou modelu. Na jej spracovanie som využila grafové konvolučné siete.

Tretia metóda pracovala s modelom reprezentovaným ako sekvencia. Navrhnutá architektúra využívala attention mechanizmus a bola založená na transformer architektúre. Pri experimentoch sa ukázalo, že táto metóda je pamäťovo veľmi náročná. Na zmenšenie tejto pamäťovej náročnosti som navrhla pooling operáciu, ktorá umožňovala zmenšiť veľkosť spracovávanej sekvencie.

Na vyhodnotenie experimentov s navrhnutými architektúrami som použila metriku, ktorá počítala v koľkých testovacích prípadoch sa odhad siete od skutočnej rotácie líšil o menej ako 30°. Metóda, ktorá pracovala s rasterizáciou a tromi pohľadmi na model dosahovala najlepšie výsledky, v 99% testovacích prípadov sa odhadnutá rotácia líšila od skutočnej o menej ako 30°.

Metóda pracujúca s attention mechanizmom dosiahla 88%, pričom zavedenie poolingu prinieslo o trochu lepšie výsledky ako bez neho. Metóda síce nedosiahla také dobré výsledky ako baseline metóda, ktorá spracovávala model vygenerovaný ako röntgenový obrázok, ale pracovala lepšie ako metóda, ktorá spracovávala obyčajný renderovaný model. Tá dosiahla úspešnosť 84%. V prípade attention mechanizmu by sa dali vyskúšať ďalšie zlepšenia, napríklad by sa dal vymyslieť mechanizmus, ktorý by umožňoval po každej pooling vrstve zvýšiť počet trénovacích parametrov, podobne ako to je v konvolučných sieťach, kde sa typicky po poolingu zvýši počet filtrov. Skúsila som zvýšiť počet trénovateľných parametrov zvýšiť tak, že som nastavila náhodný query vektor, ktorý som pri pooling vrstve generovala, ako trénovateľný. Toto však prinieslo zhoršenie výsledkov.

Posledná metóda, ktorej som sa venovala boli grafové konvolučné siete. Po vykonaní experimentov som však videla, že sieť sa veľmi pretrénovala a celkovo nebola vhodná na odhad rotácie.

Aj napriek tomu, že attention sieť dosiahla dobré výsledky, stále je priestor na zlepšovanie a dosiahnutie aspoň takej úspešnosti, akú som získala z experimentov s baseline metódou. Je možné, že iné reprezentácie polygoniálneho modelu, napríklad octree by boli schopné dosiahnuť lepšie výsledky, alebo by boli schopné efektívne definovanými operáciami poolingu a konvolúcie aspoň dosiahnuť menšiu výpočetnú náročnosť ako v prípade attention alebo grafovej siete.

Z metód priameho spracovania polygoniálneho modelu dosiahla najlepšie výsledky sieť a attention mechanizmom a zavedenou operáciou poolingu. Vzhľadom na to, že pri procese návrhu implantátu je potrebná pomerne veľká presnosť odhadu rotácie, tieto metódy ešte nie sú vhodné na použitie. Avšak navrhnutá baseline metóda spracovania modelu ako röntgenového snímku dosiahla veľmi presné výsledky a dala by sa v tomto procese použiť a čiastočne ho automatizovať.

Literatúra

[1] *3D rotation matrices* [http://motion.pratt.duke.edu/RoboticSystems/3DRotations.html#3D-rotation-matrices]. [Online; accessed 28-December-2019].

[2] *Deep Learning Book Series · 2.8 Singular Value Decomposition* [https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.8-Singular-Value-Decomposition/]. [Online; accessed 26-December-2019].

[3] *DICOM to Surgical Guide* [https://www.youtube.com/watch?v=47KtOmCEFQk]. [Online; accessed 12-May-2020].

[4] *Step-by-step: How to digitally plan implants and design surgical guides* [https://www.dentalproductsreport.com/lab/article/step-step-how-digitally-plan-implants-and-design-surgical-guides?page=0,0]. [Online; accessed 14-May-2020].

[5] *Understanding Quaternions* [https://www.3dgep.com/understanding-quaternions/]. [Online; accessed 28-December-2019].

[6] Abdullatif, H. *Understanding SVD (Singular Value Decomposition)* [https://towardsdatascience.com/svd-8c2f72e264f]. [Online; accessed 26-December-2019].

[7] Ausawalaithong, W., Thirach, A., Marukatat, S. a Wilaiprasitporn, T. Automatic Lung Cancer Prediction from Chest X-ray Images Using the Deep Learning Approach. In: *2018 11th Biomedical Engineering International Conference (BMEiCON)*. 2018, s. 1–5.

[8] Bahdanau, D., Cho, K. a Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. 2014. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. Dostupné z: http://arxiv.org/abs/1409.0473.

[9] Brock, A., Lim, T., Ritchie, J. M. a Weston, N. Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. *CoRR*. 2016, abs/1608.04236. Dostupné z: http://arxiv.org/abs/1608.04236.

[10] Charles, R., Su, H., Kaichun, M. a Guibas, L. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In:. Júl 2017, s. 77–85.

[11] Choi, E., Schuetz, A., Stewart, W. F. a Sun, J. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association : JAMIA*. 2017, roč. 24, s. 361 – 370.

[12] Defferrard, M., Bresson, X. a Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In: Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I. a Garnett, R., ed. *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, s. 3844–3852. Dostupné z: http://papers.nips.cc/paper/6081-convolutional-neural-networks-on-graphs-with-fast-localized-spectral-filtering.pdf.

[13] Deng, X., Zhang, Y., Yang, S., Tan, P., Chang, L. et al. Joint Hand Detection and Rotation Estimation Using CNN. *IEEE Transactions on Image Processing*. 2018, roč. 27, č. 4, s. 1888–1900.

[14] Ezuz, D., Solomon, J., Kim, V. a Ben Chen, M. GWCNN: A Metric Alignment Layer for Deep Shape Analysis. *Computer Graphics Forum*. August 2017, roč. 36, s. 49–57.

[15] Grabner, A., Roth, P. M. a Lepetit, V. 3D Pose Estimation and 3D Model Retrieval for Objects in the Wild. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, s. 3022–3031.

[16] Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S. et al. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics (TOG)*. ACM. 2019, roč. 38, č. 4, s. 90.

[17] Hoang, L., Lee, S.-H., Kwon, O.-H. a Kwon, K.-R. A Deep Learning Method for 3D Object Classification Using the Wave Kernel Signature and A Center Point of the 3D-Triangle Mesh. *Electronics*. Október 2019, roč. 8, s. 1196.

[18] Kanatani, K. Analysis of 3-D rotation fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1994, roč. 16, č. 5, s. 543–549.

[19] Kaufman, A., Cohen, D. a Yagel, R. Volume Graphics. *Computer*. Washington, DC, USA: IEEE Computer Society Press. júl 1993, roč. 26, č. 7, s. 51–64. Dostupné z: https://doi.org/10.1109/MC.1993.274942. ISSN 0018-9162.

[20] Langlois, J., Mouchère, H., Normand, N. a Viard Gaudin, C. 3D Orientation Estimation of Industrial Parts from 2D Images using Neural Networks. In: *International Conference on Pattern Recognition Applications and Methods*. Madeira, Portugal: [b.n.], Január 2018. Dostupné z: https://hal.archives-ouvertes.fr/hal-01681124.

[21] Lepetit, V., Moreno Noguer, F. a Fua, P. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*. Február 2009, roč. 81.

[22] Liu, X., Han, Z., Liu, Y. a Zwicker, M. Point2Sequence: Learning the Shape Representation of 3D Point Clouds with an Attention-based Sequence to Sequence Network. *CoRR*. 2018, abs/1811.02565. Dostupné z: http://arxiv.org/abs/1811.02565.

[23] Mahendran, S., Ali, H. a Vidal, R. 3D Pose Regression Using Convolutional Neural Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, s. 494–495.

[24] Masci, J., Boscaini, D., Bronstein, M. M. a Vandergheynst, P. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In: *ICCV Workshops*. IEEE Computer Society, 2015, s. 832–840. Dostupné z: http://dblp.uni-trier.de/db/conf/iccvw/iccvw2015.html#MasciBBV15. ISBN 978-1-4673-9711-7.

[25] Maturana, D. a Scherer, S. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In:. September 2015, s. 922–928.

[26] Sinha, A., Bai, J. a Ramani, K. Deep Learning 3D Shape Surfaces Using Geometry Images. In:. Október 2016, s. 223–240. ISBN 978-3-319-46465-7.

[27] Su, H., Maji, S., Kalogerakis, E. a Learned Miller, E. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. Washington, DC, USA: IEEE Computer Society, 2015, s. 945–953. ICCV '15. Dostupné z: http://dx.doi.org/10.1109/ICCV.2015.114. ISBN 978-1-4673-8391-2.

[28] Su, T.-C. B. *Seq2seq pay Attention to Self Attention: Part 1* [https://medium.com/@bgg/seq2seq-pay-attention-to-self-attention-part-1-d332e85e9aad]. [Online; accessed 30-December-2019].

[29] Szeliski, R. a Tonnesen, D. Surface Modeling with Oriented Particle Systems. September 1998, roč. 26.

[30] Umeyama, S. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* 1991, roč. 13, č. 4, s. 376–380. Dostupné z: http://dblp.uni-trier.de/db/journals/pami/pami13.html#Umeyama91.

[31] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is All you Need. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R. et al., ed. *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, s. 5998–6008. Dostupné z: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[32] Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y. a Tong, X. O-CNN: octree-based convolutional neural networks for 3D shape analysis. *ACM Trans. Graph.* 2017, roč. 36, č. 4, s. 72:1–72:11. Dostupné z: http://dblp.uni-trier.de/db/journals/tog/tog36.html#WangLGST17.

[33] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L. et al. 3D ShapeNets: A deep representation for volumetric shapes. In: *CVPR*. IEEE Computer Society, 2015, s. 1912–1920. Dostupné z: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#WuSKYZTX15. ISBN 978-1-4673-6964-0.

[34] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. et al. *A Comprehensive Survey on Graph Neural Networks*. 2019. Cite arxiv:1901.00596Comment: updated tables and references. Dostupné z: http://arxiv.org/abs/1901.00596.

[35] Xiang, Y., Schmidt, T., Narayanan, V. a Fox, D. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In:. Jún 2018.

[36] Zhang, Y. a Rabbat, M. A Graph-CNN for 3D Point Cloud Classification. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, Canada: [b.n.], 2018.

Príloha A

Obsah priloženého pamäťového média

`README.md`
`dataset_generation` - obsahuje skripty na generovanie datasetov
`attention` - obsahuje skripty na spustenie trénovania attention modelu
`graphcnn` - obsahuje skripty na spustenie trénovania grafového modelu
`rasterized` - obsahuje skripty na trénovanie konvolučných sietí na rasterizovanom modeli
`dp.pdf` - obsahuje text práce
`dp_src` - obsahuje zdrojové texty práce v LaTeXe

Príloha B

Manuál

Datasety

Datasety sa dajú stiahnuť na adrese https://nextcloud.fit.vutbr.cz/s/4igoFmdWoTHbXEN. Aby skripty fungovali správne, treba .zip súbory rozbaliť do priečinka dataset v adresári pre príslušnú metódu. Súbor `stlSamePosition.zip` obsahuje dataset pre metódu s attention, súbor `xray_multiview_dataset.zip` obsahuje dataset pre metódu s rasterizáciou a súbor `graph_data` obsahuje signály pre grafovú sieť. Samotné grafy sa v tomto priečinku nenachádzajú, kvôli veľkosti súborov. Dajú sa však vygenerovať z .stl modelov pomocou skriptu `generate_graphs.py` v priečinku `graphcnn`. V priečinku `models` sa nachádza niekoľko modelov, z ktorých sa dajú vygenerovať všetky datasety.

Trénovanie a vyhodnotenie

Na spustenie trénovania a vyhodnotenia metódy stačí spustiť skript `main.py` v priečinku príslušnej metódy.

Generovanie datasetu

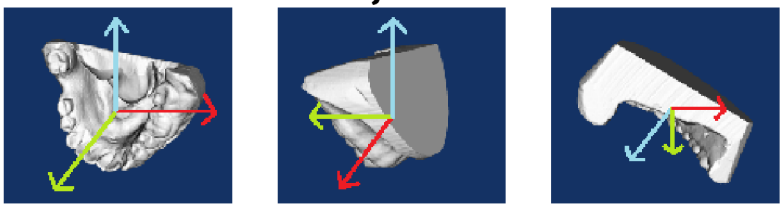
Priečinok `dataset_generation` obsahuje skripty na rôzne generovanie datasetov z .stl modelov. Skript `run_generating_script.py` spúšťa iné skripty na spracovanie modelu pre každý .stl model osobitne. Je to preto, že na zobrazenie modelu v 2D bola použitá knižnica VTK. Aby dobre fungovalo ukončenie renderovania jedného modelu a načítanie nového, tak musí byť proces ukončený. Tento skript je schopný spustiť skripty `generateDataset.py`, ktorý generuje obrázky renderovaného modelu bez röntgenového zobrazenia, `generateStlDataset`, ktorý generuje rotované .stl modely, `reduceFaces.py`, ktorý redukuje počet polygónov v modeli na 3500, `xray_display.py`, ktorý generuje 2D obrázky modelu ako röntgen a `stltovoxel.py`, ktorý slúži na voxelizáciu modelov a dá sa stiahnuť zo stránky https://github.com/cpederkoff/stl-to-voxel. Skripty `generateSequenceForAttention.py` a `generateSequenceForGraph.py` sa spúšťajú samostatne a generujú dataset pre attention model a sekvenciu signálov pre grafovú sieť.

ANALÝZA POLYGONIÁLNYCH MODELOV POMOCOU NEURÓNOVÝCH SIETÍ

AUTOR: MICHAELA DRONZEKOVÁ VEDÚCI: ING. OLDŘICH KODYM

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ÚLOHA: Odhad všeobecnej rotácie čeľuste



METÓDY:

1.

Rasterizácia modelu a zobrazenie pomocou röntgenu s tromi pohľadmi na model

2.

Využitie attention mechanizmu a spracovanie polygoniálneho modelu ako sekvencie

3.

Reprezentácia polygoniálneho modelu pomocou grafu a spracovanie grafovými konvolučnými sieťami

VÝSLEDKY: