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

Bakalářská práce

Vytvoření generického kosterního modelu dolních končetin

Plzeň 2022 David Šavel

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**David ŠAVEL**
Osobní číslo:	**A20B0239P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Vytvoření generického kosterního modelu dolních končetin**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s interním projektem Muscle Wrapping 2.0 zaměřeným na modelování muskuloskeletálního systému
2. Popište proces tvorby muskuloskeletálního modelu a možnosti automatizace tohoto procesu nástrojem STAPLE
3. Seznamte se s principem metody přímého multi-morphingu z již existující implementace
4. Analyzujte dostupné datové sady a navrhněte nástroj pro jejich automatizované sloučení za účelem dosažení generického kosterního modelu dolních končetin
5. Navržený nástroj implementujte a ověřte jeho funkčnost na dostupných datových sadách
6. Dosažené výsledky zhodnoťte a navrhněte doporučení pro další práci

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce

Vedoucí bakalářské práce: **Doc. Ing. Josef Kohout, Ph.D.**
Nové technologie pro informační společnost

Datum zadání bakalářské práce: **3. října 2022**
Termín odevzdání bakalářské práce: **4. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 25. října 2022

Prohlášení

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

V Plzni dne 4. května David Šavel

Abstract

This bachelor thesis focuses on implementing an algorithm that will fully automatically merge input models into one generic skeletal model of human lower limbs. Input models are combined with the direct multi-morphing method, which Petr Kellnhofer came up with in his diploma thesis. Some steps, of the Kellhofer algorithm, were amended in this thesis. Next, joint coordinate systems were generated for each bone model with the Staple tool. With these coordinates, bone models were transformed, so they were arranged in space the way, they are in the human body. The algorithm was tested during implementation on bone models obtained from medical images from real-life patients. Test results shown that algorithm generates merged models well, but it also revealed some input configurations, for which algorithm does not work quite right.

Abstrakt

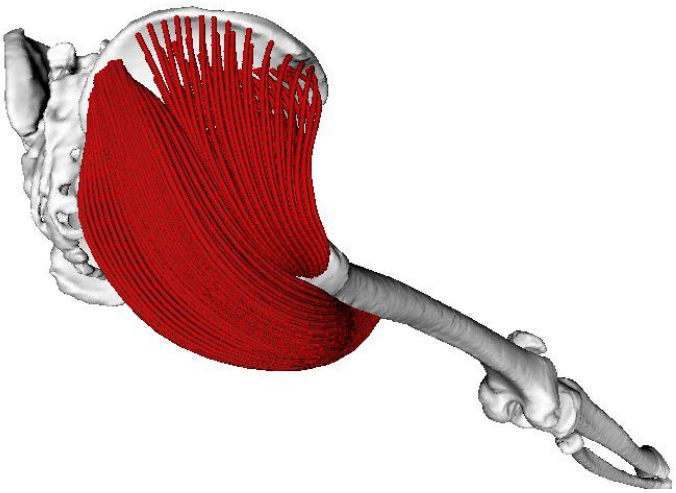
Cílem práce je vytvoření algoritmu, který bude plně automaticky slučovat vstupní modely do jednoho generického kosterního modelu dolních končetin člověka. Jednotlivé kosterní modely jsou slučovány metodou přímého multi-morphingu, navrženou Petrem Kellnhoferem v jeho diplomové práci. Některé kroky Kellnhoferova algoritmu byly v této práci pozměněny. Dále byly pomocí nástroje Staple vygenerovány kloubní souřadné systémy kostních modelů. Pomocí těchto dat byly kostní modely transformovány tak, aby byly v prostoru uspořádány stejně jako je tomu v lidském těle. Algoritmus byl během implementace testován na kosterních modelech získaných z lékařských snímků od reálných pacientů. Výsledky testů ukázaly, že algoritmus zvládá dobře generovat průměrné modely, avšak byly odhaleny i vstupní konfigurace pro které nefunguje zcela správně.

Obsah

1. Úvod 7
2. Tvorba muskuloskeletálního modelu 9
3. Metoda přímého multi-morphingu 11
4. Datové sady 15
5. Návrh nástroje pro automatické sloučení modelů 19
6. Implementace 22
6.1 Slučování kostí 22
6.1.1 Načítání vstupních dat 22
6.1.2 vtkMeshRegister 23
6.1.3 Postup slučování 23
6.1.4 Zvolený přístup ke slučování více modelů 27
6.2 Spojování modelů 29
6.2.1 Využití nástroje Staple ke spojení kostí 29
6.2.2 Transformace získaných JCS 30
7. Testování a zhodnocení výsledků 33
8. Závěr 36

1. Úvod

V medicíně se pro lékařské analýzy využívají muskuloskeletální modely, které se ukázaly jako efektivní nástroj pro studium svalové funkce. Vytvoření komplexního spojitého modelu, který by věrně reprezentoval chování lidského těla v každém bodě, je v dnešní době stále ještě nemyslitelné. Proto existují podstatně jednodušší modely pro různé aplikace. Ve všech modelech jsou nějakým způsobem reprezentovány kosti, typicky jako povrchové trojúhelníkové sítě. V modelech jsou definovány klouby a jejich stupně volnosti. To jsou body, okolo kterých se modely kostí mohou otáčet, a stupně volnosti udávají o kolik stupňů se může model kosti maximálně otočit. V modelech mohou být i svalově šlachové jednotky, reprezentované jako spojnice úponových oblastí definovaných na kostech, typicky zvané jako svalová vlákna. Ukázka takového modelu, i se svalovými vlákny je na obrázku 1.1.



Obrázek 1.1: Ukázka muskuloskeletálního modelu

Muskuloskeletální modely dělíme na generické a specifické, tedy obecné modely platné pro skupinu subjektů (např. muž ve věku 20–30 let) a modely generované pro konkrétní osoby. V případech, kdy potřebujeme vygenerovat specifický model pro konkrétního pacienta, je třeba vysoce přesné znázornění jeho anatomie muskuloskeletálních systémů. Vygenerování personalizovaných muskuloskeletálních modelů je nezbytné například v ortopedii pro plánování operací nebo navrhování chirurgického vybavení pro specifického pacienta. V současné praxi se pro tvorbu specifického modelu využívají dvě metody.

První z nich je metoda přizpůsobení generického modelu konkrétnímu pacientovi. Snahou je model co nejvíce přizpůsobit subjektu změnou měřítka kostí. Aby toto přizpůsobování bylo zcela správné, bylo by potřeba upravit i další parametry, jako třeba umístění svalových připojovacích bodů. Některé z těchto parametrů bohužel nelze jednoznačně přizpůsobit. Druhou metodou je vytvoření modelu z medicínských snímků konkrétního pacienta. Problém této metody je, že současná technologie neumí dostatečně

přesně získat některé parametry důležité pro model. V praxi se tento problém řeší tak, že se některé parametry určí z generických modelů.

Tato práce se zabývá generováním kosterního modelu dolních končetin. Hlavním výsledkem práce by měl být program, který na vstup dostane modely potřebných kostí a na výstupu vygeneruje model dolních končetin lidského těla, tedy všechny kosti od pánve směrem dolů spojené klouby. Výsledný model bude ve formátu OpenSim. OpenSim je software pro vytváření a zobrazování počítačových 3D modelů muskuloskeletálních systémů. Využívá se zejména pro analýzu a simulaci dynamiky pohybu lidí nebo zvířat. Software je postaven na knihovnách softwaru SimTK, který se využívá pro modelování dynamiky těles obecně. OpenSim nabízí knihovnu generických modelů, které může svými funkcemi dále upravovat. Funkce poskytované nástrojem *Scale Tool* v OpenSimu upravují měřítko generického modelu tak, aby odpovídal konkrétnímu subjektu [1].

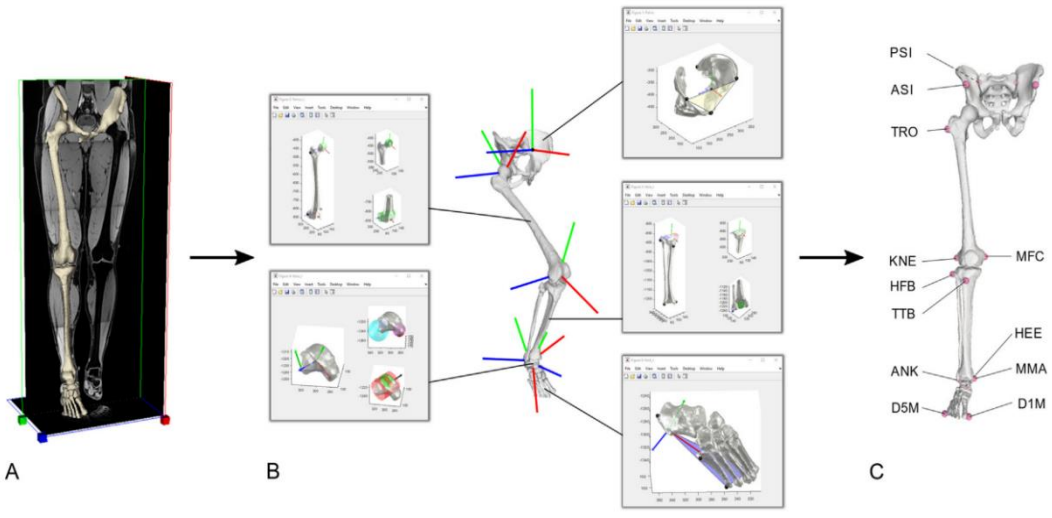
Automatizovaným vytvářením generických muskuloskeletálních modelů dolních končetin se zabývá i výzkum Muscle Wrapping 2.0, který je veden na Fakultě aplikovaných věd Západočeské univerzity v Plzni [2]. V rámci výzkumu byly vyvinuty dvě hlavní aplikace *AttachmentEstimation* a *OsimMuscleGeneratorTool*. *AttachmentEstimation* je nástroj pro odhad svalových úponů na kostech. Na vstupu dostane dva 3D modely kostí, z nichž u jedné oblast svalových úponů již bude označena. Tento nástroj pak vypočítá a na výstupu zvýrazní úponovou oblast na druhé kosti. Na kostech bude po běhu programu tedy vyznačené místo, kde by se k oběma kostem přichytili svalové úpony, kdyby se spojovali. *OsimMuscleGeneratorTool* je plugin pro OpenSim, který generuje libovolný počet svalových vláken pro každý kosterní sval muskuloskeletálního modelu, která dostal na vstupu. Výstupem jsou délky jednotlivých vláken a jejich ramena momentu síly pro odpovídající souřadnice, které se mění podle pohybu.

V následující teoretické kapitole se práce dále zabývá automatickým generováním muskuloskeletálního modelu. Konkrétně tím, jak modely generuje nástroj STAPLE.

2. Tvorba muskuloskeletálního modelu

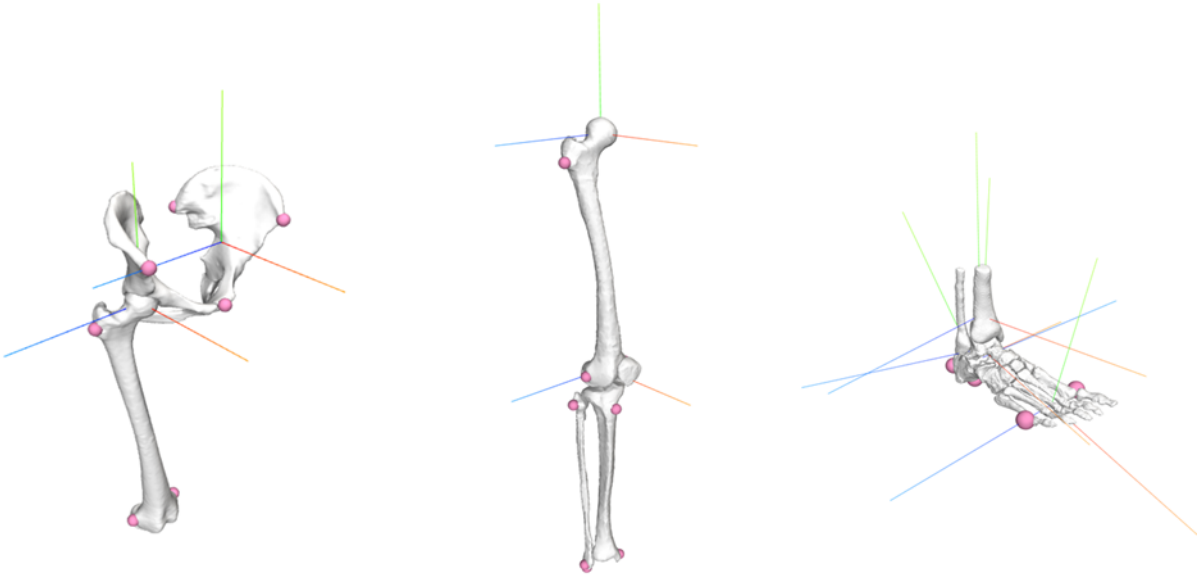
Muskuloskeletální modely generované pro specifického pacienta se v současnosti používají pouze v menších klinických aplikacích. Je tomu tak hlavně proto, že vygenerování personalizovaného kosterního modelu je časově náročná úloha vyžadující práci trénovaných operátorů. V roce 2021 přišli pánové Luca Modenese a Jean-Baptiste Renault s nástrojem pro plně automatické generování kosterních modelů dolních končetin, pomocí kosterních geometrií, získaných segmentací z lékařských snímků [3]. Tento nástroj s názvem Staple (Shared Tools for Automatic Personalised Lower Extremity modelling), je implementovaný ve skriptovacím jazyce Matlab a během několika vteřin bez potřeby zásahu obsluhy, vygeneruje kosterní model, který je okamžitě použitelný v kinetické nebo kinematické analýze. Nástroj implementuje sekvenci operací, běžně prováděných při manuálním generování modelu, avšak tyto operace provádí v zanedbatelném výpočetním čase. Modely vygenerované nástrojem Staple byly porovnány s manuálně vygenerovanými modely a byla nalezena pozoruhodná shoda ve vypočítaných počátcích souřadnicových systémů (rozdíly mezi 0.5-5.9 mm) a vygenerovaných kloubních osách (1-11°).

Před spuštěním Staplu musí uživatel získat trojrozměrné kosterní geometrie z lékařských snímků magnetické rezonance (MRI) nebo výpočetní tomografie (CT). Staple pak na vstup dostane modely všech kostí, které chceme ve výsledném modelu zahrnout. Spolu s modely musí být na vstupu specifikováno, který model reprezentuje jakou kost. Poté se automaticky zpracují vstupní modely k určení geometrických parametrů potřebných k vygenerování anatomických souřadnicových systémů (ACS) a kloubních souřadnicových systémů (JCS). Těmito geometrickými parametry mohou být například hlavní osy setrvačnosti kosti, analýza komponent nebo podélného řezu kosti. Záleží na konkrétní kosti a použitém algoritmu. V rámci STAPLE projektu bylo získáno několik výpočetních metod pro automatickou definici ACS z modelů kostí. Pro každou kost byl použit trochu jiný algoritmus, některé byly převzaty, některé reimplementovány a některé byly vytvořeny nově (viz [3]).



Obrázek 2.1: Graficky znázorněný postup generování dat nástrojem Staple [3]

Automaticky pak spočítá JCS, BL a BCS ke každému modelu a výstupem programu pak je jeden výsledný model ve formátu OpenSim. BL (BodyLandmarks) jsou pro každý model nějaké specifické body na kosti. Na obrázku 2.2 jsou to fialově zvýrazněné body na modelech kostí. BCS (Body Coordinate Systems) je struktura, která obsahuje nějaké geometrické informace o modelu kosti. Kde je počáteční bod jeho souřadného systému, kde se nachází jeho těžiště, matice setrvačnosti a podobné údaje. JCS (Joint Coordinate Systems) je pak struktura, která obsahuje informace o kloubních systémech. Pro stehenní kost například, jsou ve struktuře JCS informace o kolenním a kyčelním kloubu. Pro konkrétní kloub obsahuje struktura mimo jiné i informace o tom, kde se na kosti nachází počáteční bod, kolem kterého se kost v rámci kloubu může otáčet. Na obrázku 2.2 z těchto bodů vycházejí barevně vyznačené osy. JCS obsahuje navíc třeba údaje o tom, kde se nachází počátek souřadného systému předka nebo potomka kloubu. Na obrázku 2.1 je



Obrázek 2.2: Grafická ukázka dat, generovaných nástrojem Staple [5]

znázorněn postup generování dat Staplem. A označuje vstupní modely, B označuje kompletní model s vyznačenými JCS vygenerovaným pro jednotlivé modely kostí a C označuje kompletní model s vyznačenými body BL.

Staple umí vytvářet i dílčí modely dolní končetiny, které jsou podmnožinou kompletního modelu. Na vstupu musí program dostat kombinaci modelů kostí, ze kterých lze daný dílčí model vymodelovat, například modely stehenní lýtkové a holenní kosti pro vygenerování modelu kolenního kloubu. Další funkcionalitou nástroje je identifikace a export kloubních ploch kosti. STAPLE umí i sloučit specifický kosterní model s generickým.

Pokud člověk pohne svojí stehenní kostí, zvedne tím koleno a změní souřadnice všech kostí, které se nacházejí pod úrovní kolene. Aby výsledný model mohl správně simulovat pohyb člověka, je třeba správně určit pro každý kloub rodičovské klouby a potomky. Jako počáteční bod souřadnicového systému se volí model kosti, která se ve vzpřímeném lidském těle nachází nejvýše. Pokud by byl zadán kompletní model dolních končetin, byla byl by počátek souřadného systému zvolen na modelu pánve.

3. Metoda přímého multi-morphingu

Následující teoretická kapitola se zabývá metodou přímého multi-morphingu, tedy spojení více modelů stejného objektu do jednoho výsledného generického modelu. Konkrétně bude popsána metoda přímého multi-morphingu, která byla navržena v diplomové práci Petra Kellnhofera [4]. Základem této metody je nerigidní registrace.

Problém registrace znamená, že máme více modelů stejného objektu, se kterými potřebujeme pracovat, ale neznáme jejich vzájemnou polohu a natočení. Během registrace se najde správná transformace, která tyto modely co nejpřesněji srovná dohromady, tedy transformuje zdrojový model na cílový. Dvě hlavní skupiny registračních metod se odlišují podle typu transformace, kterou se pokouší najít.

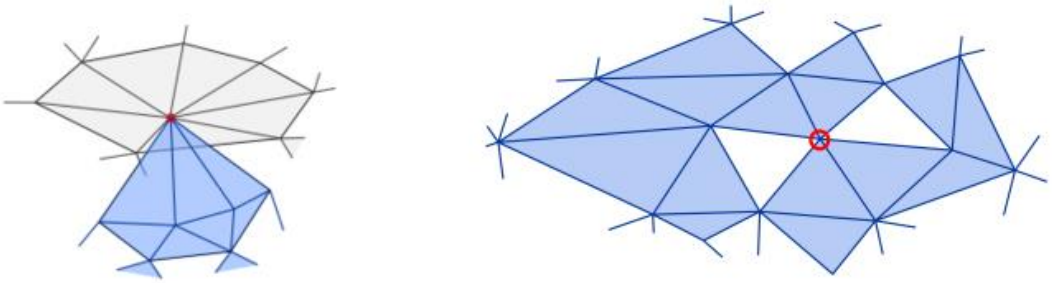
Rigidní metody hledají jednu afinní transformaci, která se aplikuje na celou síť, zatímco nerigidní metody hledají rigidní transformaci pro každý bod objektu. Rigidní metody předpokládají objekty stejného tvaru ve stejné poloze, které se po transformaci dobře překryjí. Za splnění tohoto předpokladu bude k registraci stačit translace a rotace objektu. V reálných situacích odchylky způsobují to, že dokonalá registrace nemůže být provedena pouze rigidní transformací. Nerigidní metody nespoléhají na to, že modely představují objekt shodného tvaru ve stejné poloze. Proto musí transformovat nejen celou síť, ale i jednotlivé body, aby kompenzovali počáteční deformaci modelu.

Zatímco při registraci se považuje jeden model za zdrojový a druhý za cílový, u morfování tomu tak být nemusí. Při morfování ze dvou vstupních modelů vzniká nový model, jehož geometrie je dána váženou kombinací dvou vstupních modelů. Dá se tedy parametrizovat ke kterému ze vstupních modelů bude mít výsledek blíže. Pokud bychom parametry nastavili u jednoho modelu na maximum a u druhého na minimum, jednalo by se o registraci. Přesto morfování obvykle předpokládá určitou úroveň registrace před jeho spuštěním, což předchází tomu, aby se morfovali navzájem odlišné části modelu. V této práci bude dále zmiňována pouze metoda přímého morfování (direct morphing), která parametrizování neřeší vůbec. Pokud na vstupu máme více než dva modely hovoříme o tzv. multi-morphingu.

Petr Kellnhofer ve své diplomové práci navrhl dvě varianty algoritmu morfování, parametrizovanou a přímou. V tomto odstavci bude stručně popsána Kellnhoferova metoda přímého morfování (direct on-mesh morphing). Tento algoritmus se skládá ze 4 základních kroků. Před prvním krokem má algoritmus kosti reprezentované trojúhelníkovými sítěmi.

Většina pozdějších částí algoritmu očekává manifoldní trojúhelníkové sítě, musí se sítě v prvním kroku upravit. Trojúhelníková síť je manifoldní, pokud platí, že každá hrana je součástí dvou trojúhelníků, každý vrchol je sdílen nejméně dvěma trojúhelníky a žádné trojúhelníky se nepřekrývají. Znamená to, že lze bezpečně procházet po povrchu sítě. Tento požadavek bohužel v praxi často nebývá splněn kvůli nízkému rozlišení lékařských snímků. Z toho důvodu se v prvním kroku algoritmu musí sítě upravit.

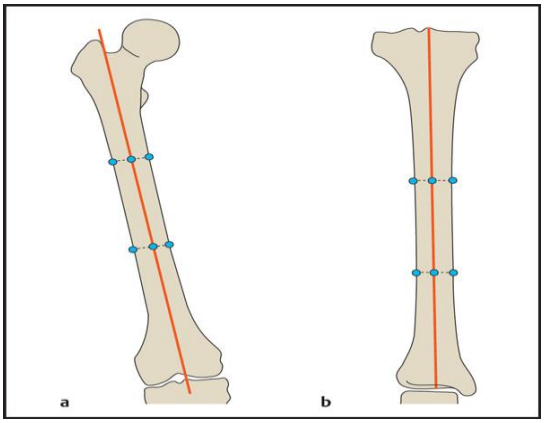
Nejprve se ze sítí odstraní ty hrany, které mají více než dva přilehlé trojúhelníky. Ani poté nemusí být splněn předpoklad manifoldní sítě. K síti mohou stále být připojeny komponenty, které se sítí však nesdílejí žádnou hranu, ale sdílejí pouze vrchol. Proto budou odstraněny i některé vrcholy, včetně všech trojúhelníků s daným vrcholem, pro které platí, že jsou přes něj k síti připojeny komponenty, které tam být nemají. Například na obrázku 3.1 vlevo je vrchol, který odstraněn bude, zatímco vpravo je vrchol, který v síti zůstane.



Obrázek 3.1: Ukázka dvou vrcholů v trojúhelníkové síti [4]

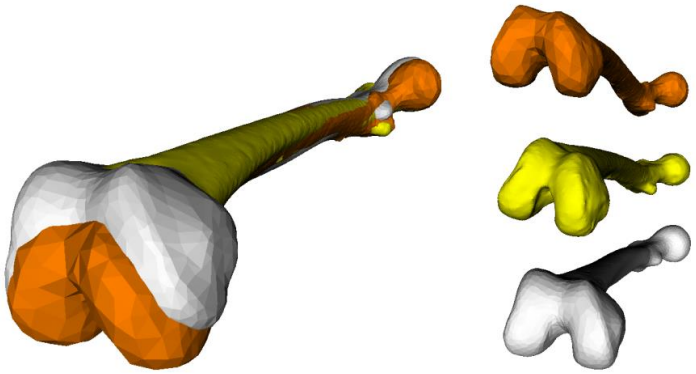
Odstranění některých hran a vrcholů může způsobit odizolování některých komponent sítě. Algoritmus odstraní všechny odizolované komponenty kromě té největší, kterou nechá jako základ nové upravené sítě. Odstraňování hran a vrcholů pravděpodobně způsobí také vznik děr. Jelikož sít se v tuto chvíli skládá už pouze z jedné komponenty, neměl by být problém tuto komponentu projít, najít oblast každé díry a zaplnit díru trojúhelníky. Byla k tomu použita metoda *vtkFillHolesFilter*, implementovaná v rámci softwaru VTK (The Visualization Toolkit). To je software pro vyobrazení a práci s vědeckými daty. Umožňuje vykreslování a interakci se 2D i 3D modely.

Ve druhém kroku algoritmu je cílem zarovnat sítě tak, že podobné části kostí budou ležet blízko u sebe. Cílovým stavem po druhém kroku jsou tedy sítě ležící na přibližně stejném místě, se stejným měřítkem a jejichž kartézské osy směřují stejným směrem. Nejprve se v kosti najde hlavní osa. Ukázka hlavních os některých kostí je znázorněna na obrázku 3.2.



Obrázek 3.2: Ukázka hlavních os stehenní a holenní kosti [13]

K této ose se najdou dva nejdelší ortogonální vektory, které jsou i navzájem ortogonální. To budou zbylé dvě kartézské osy. V případě třeba stehenní kosti povede hlavní osa podél kosti od hlavice až kolennímu kloubu. Dva nejdelší ortogonální vektory pak budou uvnitř hlavice kosti. Tyto osy se nejdou pro každý model stejné kosti a ty budou poté transformovány tak, aby byly zarovnány jako je tomu na obrázku 3.3. Aby došlo k takovému zarovnání, je kromě transformace provést takzvaný *scaling*, neboli změnu měřítka jedné kosti tak, aby se přizpůsobila délce kosti druhé.



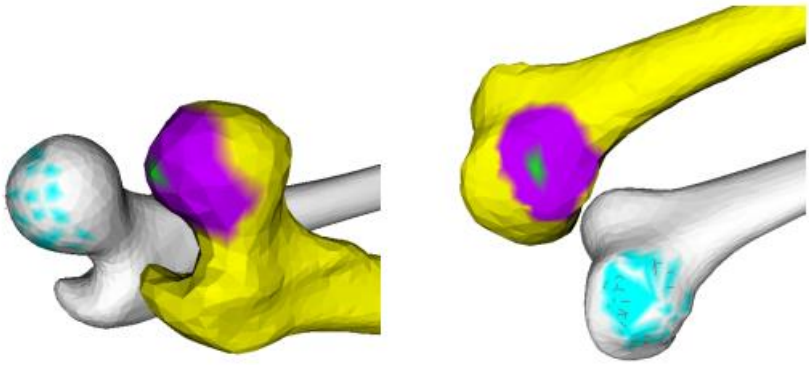
(a) Aligned (b) Moved apart

Obrázek 3.3: Ukázka úspěšného překrytí tří stehenních kostí po zarovnání [4]

V dalším kroku se tohoto počátečního zarovnání využívá k pokročilé registraci. Výsledkem tohoto kroku by měly být mírně zdeformované překrývající se sítě s minimálními rozdíly povrchů a velmi dobrým vzájemných zarovnáním. V tomto kroku je jedna síť považována za zdrojovou a jedna za cílovou. Registrace se skládá ze dvou kroků, nejprve se provede rigidní registrace ICP a poté nerigidní registrace ICP.

ICP (Iterative Closest Point) je jednou z nejpoužívanějších metod registrace. V prvním kroku metody rigidní ICP se pro každý vrchol z jedné sítě vybere jeden vrchol z druhé sítě. Výběr se provede na základě nejmenší Euklidovské vzdálenosti. Dále metoda hledá takovou transformaci, po které budou všechny vrcholy první sítě co nejblíže vrcholů, které jim byly vybrány z druhé sítě. Transformace probíhá ve dvou krocích. Nejprve se najdou těžiště obou sítí a sítě se posunou tak, aby těžiště byly na stejném místě. Poté se provede rotace, po které bude součet Euklidovských vzdáleností všech vrcholů s jejich vybranými vrcholy z druhé sítě co nejmenší.

Následuje nerigidní ICP registrace, která využívá výběru takzvaných regionů. Ve zdrojové síti je náhodně vybráno několik vrcholů, kolem kterých je vybrán region. Vrcholů, kolem kterých se bude určovat region je vybráno vždy 10 % z celkového počtu vrcholů, avšak nesmí jich být více než 500. Jako region Kellnhofer vybírá všechny vrcholy do topologické vzdálenosti 3 od vrcholu pro který je region vybírán. Pro body z tohoto regionu se pak vybírají nejbližší vrcholy z cílové sítě. Ukázka vybraného regionu je znázorněna na obrázku 3.4. Na obrázku je žlutou barvou označena zdrojová síť a na ní je



Obrázek 3.4: Ukázka regionu (fialová) vybraného pro konkrétní bod (zelená) [4]

fialovou barvou znázorněn region vybraný pro bod zvýrazněný zeleně. Bílá kost je cílová a jsou na ni tyrkysově vyznačeny nejbližší vrcholy pro fialový region. Vybrané regiony se registrují na cílovou sít pomocí transformací, které se však neprovádí pro celou síť, ale pouze pro daný region. V posledním kroku už se zarovnané sítě po registraci použijí jako vstup pro morfování. Výsledkem je model, který kombinuje rysy všech vstupních modelů.

Implementace Petra Kellnhofera byla zakomponována do projektu Muscle Wrapping 2.0 [2], avšak byla silně modifikována. V Kellnhoferově implementaci se při nerigidní transformaci volil pro několik vrcholů okolní region. Kellnhofer tento region volil fixně, což fungovalo dobře na konkrétních datech, avšak pro obecná data už algoritmus nefungoval. Pro projekt Muscle Wrapping 2.0 [2] byl algoritmus upraven tak, že se transformace vždy provádí na oblasti, která nemá fixní velikost, avšak spočte se pro každou síť podle její velikosti. Zvolí se minimální počet vrcholů v jednom regionu. Celkový počet vrcholů sítě se vydělí zvoleným minimem. Výsledek určuje počet regionu, avšak je nastavena nějaká spodní hranice velikosti regionu. Pokud by vyšla velikost regionu menší než stanovená hranice, bere se stanovená hranice jako velikost regionu.

4. Datové sady

V následující kapitole budou představena a popsána vstupní data, která jsou dostupná na githubu projektu STAPLE [5]. Jedná se celkem o 9 datových sad obsahujících snímky kostí uložených ve formátu mat. To jsou matlabovské binární soubory (.mat), do kterých se běžně ukládají hodnoty proměnných. Tyto soubory obsahují header, který je zabírá 128 bytů a je následován minimálně jedním datovým blokem. Datový blok obsahuje tag velký osm bytů. Tento tag specifikuje, kolik bytů má datový blok a jak by jeho data měla být interpretována.

Soubory datových sad, které jsou popsány v této kapitole obsahují vždy dva datové bloky. První datový blok obsahuje souřadnice všech vrcholů trojúhelníkové sítě. Druhý datový bok obsahuje trojice indexů do pole vrcholů, které určují, jaké vrcholy budou tvořit trojúhelník ve výsledném 3D modelu.

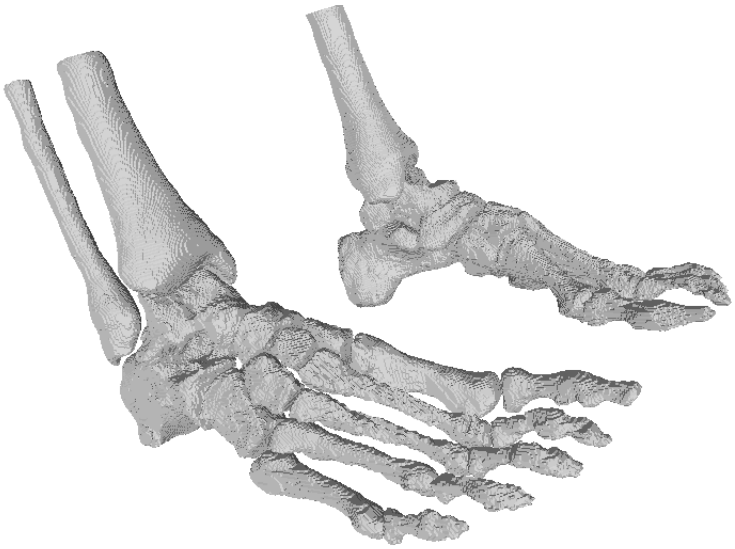
Abych si modely mohl zobrazit a podívat se, jak vypadají, musel jsem je převést na 3D Object formát (.obj). Nejprve jsem si z Matlabu vygeneroval textové soubory, které obsahovaly po trojicích zapsané nejprve souřadnice všech vrcholů a poté indexy vrcholů tvořící v síti trojúhelník. Poté jsem napsal krátky program v jazyce C++, který dostal na vstupu zmíněné textové soubory a vygeneroval z nich na výstup požadovaný soubor.

ICL_MRI

Datová sada ICL_MRI [3] obsahuje celkem 12 modelů kostí. Nachází se zde pouze modely kostí pravé dolní končetiny a pánve. Pravá noha je kompletní včetně snímků čéšky, paty nebo hlezenní kosti. Jediná část nohy, která chybí jsou prsty. Kosti pánve, tedy dvě pánevní kosti a křížová kost, jsou zde zastoupeny více modely. Kromě jednotlivých modelů každé z kostí pánve, se v datech nachází kompletní model všech tří kostí dohromady, a ještě model dvou kostí pánevních dohromady bez křížové kosti. Podobně tomu je s lýtkovou kostí, která je v jednom samostatném modelu a v jednom modelu už dohromady s holenní kostí. V datech se nevyskytují žádné větší nedostatky nebo nepřesnosti. Povrchy kostí jsou poměrné hladké. Data byla získána od pacienta v Královské univerzitě v Londýně (Imperial College London), proto je datová sada pojmenována ICL.

JIA_ANKLE_MRI

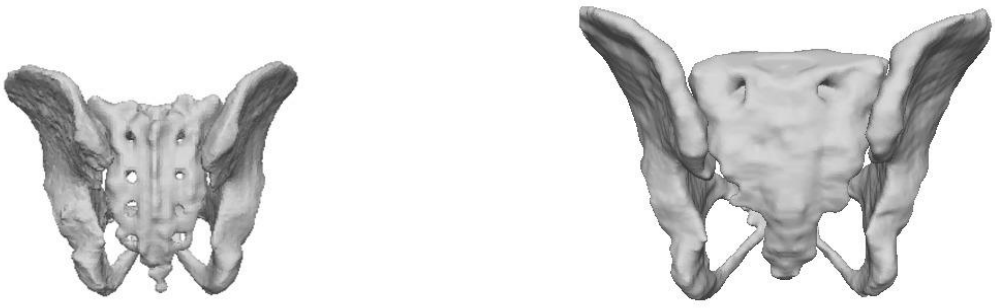
Datová sada JIA_ANKLE_MRI [6] obsahuje pouze 8 modelů kostí. Nachází se zde chodidla obou dolních končetin, avšak oproti sadě ICL_MRI, zde už jsou i modely prstů. Ke každé noze je zde ještě model s koncem lýtkové i holenní kosti. Tyto modely nemají příliš hladký povrch. Kromě toho kosti neobsahují žádné nepřesnosti či nedostatky. Ukázka je na obrázku 4.1. Data z této a následující datové sady byla získána od dětských pacientu trpících juvenilní idiopatickou artritidou (Juvenile Idiopathic Arthritis), proto se v názvu těchto sad objevuje zkratka JIA.



Obrázek 4.1: datová sada JIA_ANKLE_MRI

JIA_MRI

Datová sada JIA_MRI [7] obsahuje 14 modelů, které dohromady tvoří model obou dolních končetin včetně pánve. Je zde na první pohled viditelný rozdíl mezi modely některých kostí. Modely čéšky, stehenní, lýtkové a holenní kosti nemají tak hladký povrch jako zbytek snímků. Pánev je zde opět zastoupena více modely. Na jednom modelu je kompletní pánev a na druhém jsou pánevní kosti bez křížové. Tyto data nejsou tolik přesná a je to vidět zejména u pánve, hlavně u kosti křížové. Část křížové kosti zde chybí a například většina otvorů, které jsou v křížové kosti zde vůbec není.



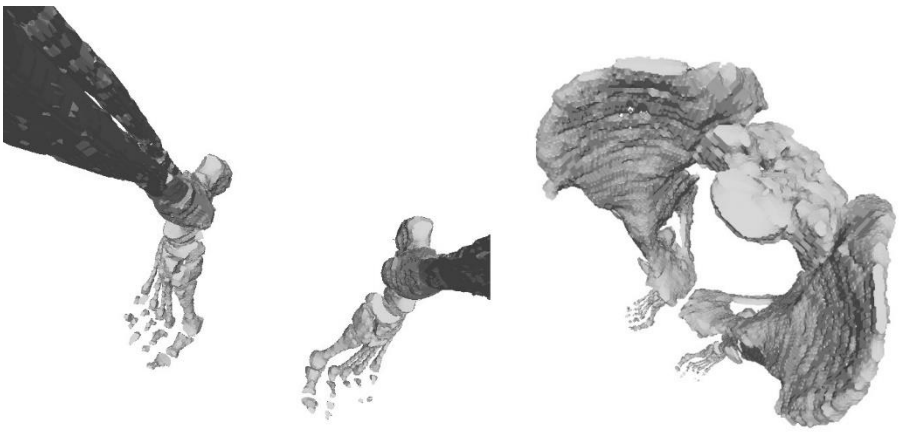
Obrázek 4.2: Porovnání pánví ze sady ICL_MRI a JIA_MRI. Vpravo je pánev ze sady JIA_MRI

LHDL_CT

V datové sadě LHDL_CT [8] je pouze sedm modelů. Jsou zde modely kostí pouze pravé dolní končetiny bez prstů a pánve, avšak bez křížové kosti. Lýtková kost je zde v jednom modelu i s holenní kostí a v jednom modelu samostatně. Povrchy kostí jsou velmi hladké.

MC22

Datová sada MC22 [9] obsahuje pouze 8 modelů, které však tvoří kompletní model obou dolních končetin včetně pánve. Pánev má v sadě dva modely, jeden s křížovou kostí a jeden bez. Na pohled obsahují tyto data asi nejvíce nepřesností. Například u chodidel jsou prsty na obou nohách jiné, a navíc velké části prstů chybí, takže některé prsty vypadají pouze jako malé kousíčky vzdálené od nohy. Povrchy kostí nejsou příliš hladké a na některých místech, zejména u pánevních kostí, jsou dokonce díry skrz kost. Ukázka datové sady je na obrázku 4.3. Data z této sady byla získána od ženy v pokročilém věku.



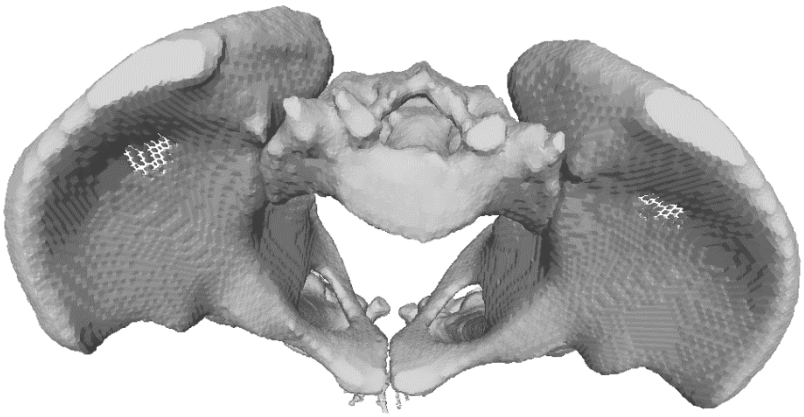
Obrázek 4.3: Ukázka datové sady MC22

TLEM2_CT

Tato datová sada TLEM2_CT [10] se 14 modely obsahuje opět celý kosterní model obou dolních končetin včetně pánve, která je opět zastoupena dvěma modely, jedním s křížovou kostí a jedním bez. Povrchy kostí jsou velmi hladké a žádné větší části kosterního modelu nechybí.

TLEM2_MRI

Datová sada TLEM2_MRI [10] je velmi podobná sadě TLEM2_CT. Obsahuje stejné modely jako datová sada TLEM2_CT, avšak jejich kvalita je jiná. Tato sada nemá tak hladké povrchy kostí, a navíc jsou zde díry v pánevní kosti, které jsou vidět na obrázku 4.4. Data byla získána v rámci výzkumu na univerzitě v Twente. TLEM je zkratka pro Twente Lower Extremity Model.



Obrázek 4.4: Díry v pánevních kostech v sadě TLEM_MRI

VAKHUM_CT

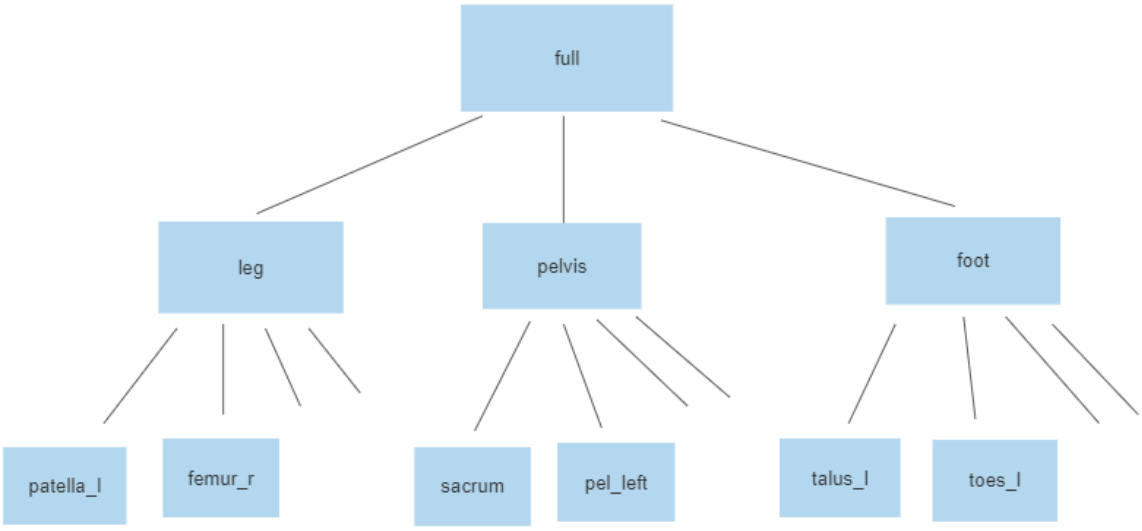
Sada VAKHUM_CT [11] obsahuje pouze pět modelů. Je zde model pánve, bez křížové kosti, poté model stehenní kosti, čéšky a hlezenní kosti. V posledním model je lýtková a holenní kost dohromady. Je to jediná datová sada, kde kosti nejsou spojeny, tedy nejsou správně rozmístěny v prostoru. Kosti mají poměrně hladký povrch, avšak kvůli rozmístění se prolínají na některých místech, na kterých by neměly. Například hlavice holenní kosti vstupuje do pánevní kosti a její zadní část kvůli tomu není vidět. Data byla získána v rámci projektu VAKHUM, což je zkratka odvozená z Virtual animation of the kinematics of the human.

5. Návrh nástroje pro automatické sloučení modelů

V této kapitole bude popsán návrh nástroje, který automaticky sloučí několik kosterních modelů do jednoho výsledného kosterního modelu dolních končetin. Vstupem nástroje tedy bude libovolný počet kosterních modelů, jejichž přesný formát zadávání na vstup bude popsán v následujícím odstavci. Výstupem nástroje bude pouze jeden generický model vygenerovaný ze vstupních modelů.

Protože uživateli bude umožněno zadávat jeden kosterní model dolních končetin po více částech nebo po jednotlivých kostech, bylo by poměrně nekomfortní chtít po uživateli zadávat na vstup každý soubor. Z toho důvodu bude nástroj vyžadovat pouze jeden konfigurační soubor v textové podobě, ve kterém budou specifikovány všechny potřebné informace.

Struktura konfiguračního souboru bude taková, že zde budou specifikovány jednotlivé modely oddělené prázdnými řádky. Každý model pak bude zadán pomocí parametrů. Každá kost dolních končetin má svůj parametr, například femur_r pro pravou stehenní kost nebo sacrum pro křížovou kost. Hierarchie parametrů je znázorněna na obrázků níže. Všechny kosti jsou rozděleny do tří kategorií pelvis leg a foot. Do kategorie pelvis patří pouze obě pánevní kosti a kost křížová. Do kategorie leg patří všechny kosti od pánve až



Obrázek 5.1: Hierarchie a rozdělení parametrů konfiguračního souboru

k chodidlům. Kosti od hlezenní kosti včetně až ke špičkám prstů patří do kategorie foot. Modely v konfiguračním souboru musí být rozdělen do těchto kategorií. Každý řádek bude začínat parametrem, který určí o jako kategorii se jedná, a následovat budou všechny zadávané kosti z dané kategorie. Pokud by uživatel měl všechny kostí jedné kategorie v jednom modelu, může ho zadat místo všech kostí. Pro co nejlepší a nejpřesnější výsledky je lepší zadávat modely po jednotlivých kostech, pokud je to možné. Při specifikaci modelu nemusí být využito všech tří parametrů. V případě, že by měl model bez chodidel, může použít pouze parametry pelvis a leg, foot může vynechat a nástroj nebude hledat soubor s modelem chodidel. Pokud by uživatel zadal odkaz na neexistující soubor

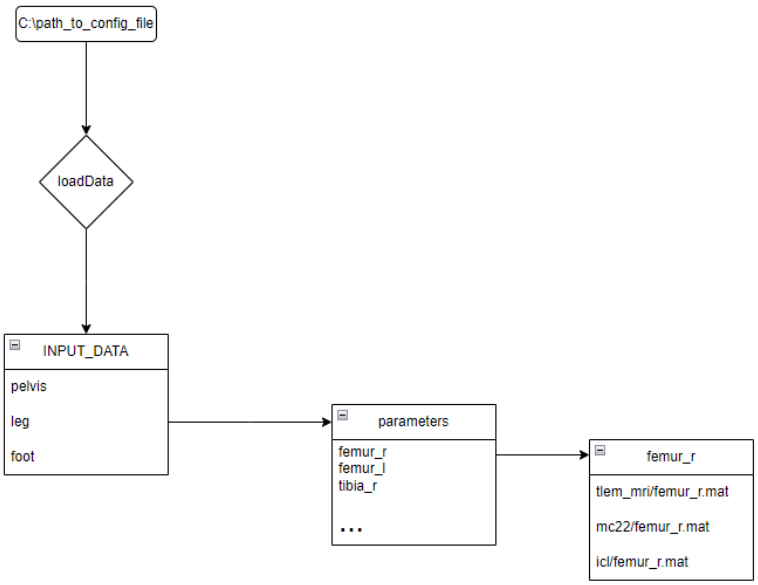
nástroj tento model vynechá a uživatele o tom informuje. Na obrázku 5.2 je ukázán příklad konfiguračního souboru, který načte 3 datové sady. V první datové sadě jsou obě stehenní kosti, ve druhé je pouze pravá stehenní kost a v poslední datové sadě je pánevní kost, obě stehenní kosti, levá lýtková kost s holenní kostí a levá hlezenní kost.

```
leg -femur_r tlem_mri/femur_r.mat -femur_l tlem_mri/femur_l.mat

leg -femur_r mc22/femur_r.mat

pelvis -pelvis  icl/pelvis.mat
leg -femur_r icl/femur_r.mat -femur_l icl/femur_l.mat -tibia_l icl/tibia_l.mat
foor -talus_l icl/talus_l.mat
```

Obrázek 5.2: Ukázka struktury konfiguračního souboru

Podle konfiguračního souboru si nástroj najde modely jednotlivých kostí a seskupí si modely stejných kostí. Poté začne slučovat jednotlivé kosti. Po sloučení všech kostí je spojí do tří modelů kostí z kategorií z konfiguračního souboru. Po tomto kroku budou tedy 3 modely, model chodidel, model pánve a model stehenní kosti spojené s holenní a lýtkovou kostí, včetně čéšky. To proto kdyby uživatel zadal kosti nějaké kategorie jedním modelem. V tom případě by se zadaný model sloučil s jedním ze tří dosud vygenerovaných generických modelů. V posledním kroku už se jen spojí tyto tři modely do jednoho výsledného celkového modelu.

Modely jednotlivých kostí se budou slučovat postupně, tedy nejdřív se sloučí dva modely a vzniklý generický model se zase sloučí s dalším modelem. Slučování jednotlivých modelů bude probíhat metodou přímého multi-morfingu popsanou ve třetí kapitole. Nástroj modely nejprve posune a zvětší či zmenší tak, aby byly co nejvíce zarovnané. Následuje nerigidní registrace pomocí metody ICP, která byla popsána ve třetí kapitole.



Obrázek 5.3: Postup uložení dat z konfiguračního souboru

Vybere se tedy zdrojová a cílová kost, poté se na zdrojové kosti určí regiony a na těchto regionech se provede registrace. Jako zdrojová kost bude zvolena ta s větším počtem vrcholů.

Po provedení nerigidní registrace bychom měli dostat poměrně dobře zarovnané sítě, které už je třeba jen spojit do jedné. Tedy ze dvou množin vrcholů a hran nějakým způsobem udělat jen jednu množinu. To se provede tak, že pro každý vrchol cílové sítě se se vybere jeden trojúhelník na zdrojové síti. Jejich poloha se zprůměruje a poloha vrcholu cílové sítě se změní podle vypočítaného průměru. Tím se změní poloha vrcholů a hrany mezi nimi mohou zůstat. V tomto případě bude jako cílová kost zvolena ta s větším počtem vrcholů. Poté co se lehce změní poloha jejích vrcholů bude moct být tato cílová kost považována za výsledek sloučení. Výsledný model bude na stejné pozici jako model, který byl zvolen jako cílový. Na jeho místo se totiž zaregistruje zdrojový model předtím, než se začne měnit poloha jeho vrcholů. Výběr cílového modelu tedy ovlivní polohu výsledného modelu, nikoli však jeho vzhled. Vůči sobě budou vrcholy modelů posunuty stejně, ať už se jako zdrojový zvolí kterýkoliv model.

Ještě je třeba určit jakým způsobem se budou provazovat vrcholy, tedy podle čeho se budou pro vrcholy cílové sítě vybírat trojúhelníky ze sítě zdrojové. Po provedení nerigidní registrace můžeme každému vrcholu sítě přiřadit jeden trojúhelník z druhé sítě na základě Euklidovské vzdálenosti. Vrchol se pak zprůměruje s každým vrcholem trojúhelníku. Tím získáme tři nové vrcholy a ty zprůměrujeme do jednoho výsledného.

6. Implementace

6.1 Slučování kostí

První problém, kterým jsem se při implementaci zabýval, bylo slučování více kosterních modelů do jednoho modelu průměrného. Cílem bylo vymyslet algoritmus, který na vstupu dostane několik datových sad a na výstupu vrátí pouze jednu datovou sadu, která bude kombinací všech datových sad ze vstupu.

Nástroj, který automaticky slučuje kosterní modely, jsem nazval GenericGenerator. Pro implementaci tohoto nástroje jsem zvolil programovací jazyk *C++* hlavně proto, že v tomto jazyce je napsán projekt Muscle Wrapping 2.0 [2], ze kterého tato práce vychází a jehož některé knihovny nástroj také používá. Z podobného důvodu byla zvolena i knihovna VTK [12] pro práci s modely kostí a pro jejich zobrazení.

V této podkapitole bude nejprve podrobně vysvětlen celý algoritmus slučování kosterních modelů. Bude vysvětleno proč a jak se dělají všechny kroky algoritmu a také jaký je rozdíl mezi tímto algoritmem a algoritmem, který ve své diplomové práci navrhl Petr Kellnhofer. V druhé části bude vysvětleno, jakými způsoby se dá přistoupit k problematice slučování několika modelů a jaký způsob byl nakonec zvolen a proč.

6.1.1 Načítání vstupních dat

Jak již bylo zmíněno dříve v kapitole o návrhu nástroje, datové sady jsou specifikovány v jednom konfiguračním souboru, ve kterém jsou uvedeny cesty ke všem modelům kostí, které bude chtít uživatel slučovat. Struktura konfiguračního souboru odpovídá předchozímu popisu, tedy kosti jednoho modelu rozděleny do tří kategorií, pelvis, leg a foot. Modely kostí jsou takto děleny, aby bylo uživateli umožněno zadávat i modely, ve kterých je již obsaženo více kostí. Tedy aby bylo možné sloučit například model celé pravé dolní končetiny s několika modely pravé stehenní kosti a několika modely kosti lýtkové. To by probíhalo tak, že by se nejprve pomocí nástroje GenericGenerator vygenerovaly průměrné modely stehenní a lýtkové kosti, poté by se tyto modely pomocí Staplu [5] spojily do jednoho modelu, a tento model by se průměroval s modelem celé dolní končetiny opět pomocí nástroje GenericGenerator. To by vyžadovalo poměrně složitý skript, který by pro každý konfigurační soubor zařídil správné střídání volání nástrojů GenericGenerator a Staplu.

To bohužel výsledná implementace neumožňuje, protože během hlubší analýzy nástroje Staple se ukázalo, že nástroj vstupní modely v kloubech nespojuje. Předpokladem bylo, že Staple dostane na vstup několik modelů na různých místech v prostoru a automaticky je přesune na správná místa tak, jako je tomu v lidském těle. Staple ale modely nepřesouvá vůbec, protože je očekává na vstupu již správně uspořádané. Staple pak pouze vygeneruje souřadnice kloubů, stupně volnosti a spoustu dalších dat, která pak společně s modely uloží do výsledného modelu ve formátu OpenSim. Z toho důvodu se spojování modelů v kloubech bude muset provést jiným způsobem.

V konečném řešení se proto slučují pouze modely jednotlivých kostí. Toto omezení ale nevadí, protože jak je patrné z analyzovaných datových množin, není obvyklé, aby v jednom vstupním modelu bylo obsaženo více kostí.

6.1.2 vtkMeshRegister

Tato třída je součástí projektu Muscle Wrapping 2.0 [2] a slouží ke slučování kostních modelů. Funguje na základě metody přímého multi-morphingu popsané v jedné z předchozích kapitol. Uvádím ho zde, protože celou řadu jeho funkcí využívám ve své implementaci.

Do této třídy se přidají modely, z nichž jeden se zvolí jako cílový. Modely jsou ve *vktMeshRegisteru* uloženy jako instance třídy *ExtendedMesh*, která slouží pro práci s trojúhelníkovými sítěmi. Poté se spustí algoritmus, který nejprve všechny sítě zkontroluje a odstraní některé jejich části tak, aby byly uzavřené a skládaly se pouze z jedné komponenty. Poté provede počáteční zarovnání na cílovou síť, pak rigidní registraci, pak nerigidní, poté parametrizaci, a nakonec samotné slučování. Před spuštěním, lze kterýkoli z těchto stavů nastavit jako koncový. Před spuštěním algoritmu lze také nastavit *cachování*. To v případě této třídy znamená uložení všech modelů, se kterými třída momentálně pracuje do souboru, v takovém stavu, v jakém se nacházejí v mezikroku, pro který bylo *cachování* nastaveno. Pokud by se tedy nastavilo *cachování* například pro mezikrok počátečního zarovnání, všechny vstupní modely by se počáteční zarovnání uložily do *cache* adresáře. Tato funkce umožňuje i to, že pokud bychom měli již zarovnané modely, mohly bychom je dát do složky *cache,* nastavit *cachování* pro mezikrok počátečního zarovnání, a *vktMeshRegister* by si automaticky zarovnané modely načetl a algoritmus spustil až od mezikroku počátečního zarovnání.

6.1.3 Postup slučování

V následující kapitole je popsán přesný algoritmus, jakým ve své implementaci slučuji modely kostí. Bude popsán algoritmus sloučení dvou modelů. Při slučování více modelů je logika algoritmu stejná, avšak jeho kroky se uplatňují na více modelů zároveň. Pokusím se také objasnit, jaký je rozdíl mezi Kellnhoferovou metodou přímého multi-morphingu [4], kterou jsem již popsal v jedné z dřívějších kapitol, a mým algoritmem.

Počáteční kroky se provádějí tak, jak je popsáno ve třetí kapitole. Z modelů se odstraní nežádoucí komponenty tak, aby byly manifoldní, provede se počáteční zarovnání, rigidní registrace a následně nerigidní registrace.

Aby se sítě mohly zarovnat je třeba jednu z nich zvolit jako cílovou. Teoreticky je jedno, který z modelů bude označen jako cílový, to ovlivní pouze polohu a rotaci výsledného modelu, nikoli však jeho vzhled, jak již bylo vysvětleno v předchozí kapitole. Dalším krokem algoritmu je rigidní registrace, následována nerigidní registrací. Všechny předchozí kroky jsem ve svém programu implementoval využitím nástroje *vtkMeshRegister*.

Jeden z modelů je označen jako zdrojový. To je ten s větším počtem vrcholů. Právě tento model bude algoritmem vrácen jako výsledný, avšak předtím se provede translace jeho vrcholů podle průměru. Ke každému vrcholu zdrojového modelu se najde jeden nejbližší trojúhelník na druhém modelu. Pro nejbližší trojúhelník platí, že součet vzdáleností jeho vrcholů, od vrcholu z druhého modelu, je co nejnižší. K tomu využívám knihovnu *vtkCellLocator*. Do *vtkCellLocatoru* se metodou *SetDataSet* přidá model, na které budeme hledat nejbližší trojúhelník. Ten se hledá pro každý vrchol metodou *FindClosestPoint,* přičemž bod, pro který trojúhelník hledáme se předává jako parametr metody.

Po nalezení všech těchto dvojic se můj algoritmus vrací o krok zpět, tedy od nerigidně registrovaných modelů zpět k těm registrovaným pouze rigidně. Dvojice vrcholu z jedné kosti a trojúhelníku z druhé však zůstanou tak, jak byly nalezeny po nerigidní

registraci. K návratu k rigidně registrovaným modelům jsem využil funkci *cachování* ve *vtkMeshRegisteru*. Rigidně registrovaný model zdrojové kosti se načte jako nový model ze složky *cache*. *Cachování* bylo ve *vtkMeshRegisteru* implementováno pouze pro stav po počátečním zarovnání nebo po nerigidní registraci, proto jsem musel *cachování* po rigidní registraci do knihovny přidat.

V posledním kroku algoritmu se spočítají nové souřadnice pro každý vrchol zdrojového modelu. Pro každý vrchol se nové souřadnice spočítají jako průměr se všemi vrcholy trojúhelníku, který byl vrcholu přiřazen v předchozím kroku. Nejprve se počítá průměr pro každý vrchol trojúhelníku zvlášť a tyto tři výsledky se dále průměrují do jediného výsledného vrcholu. Všechny vrcholy zdrojového modelu se přesunou na nově vypočítané souřadnice, ale hrany mezi vrcholy se zachovají.

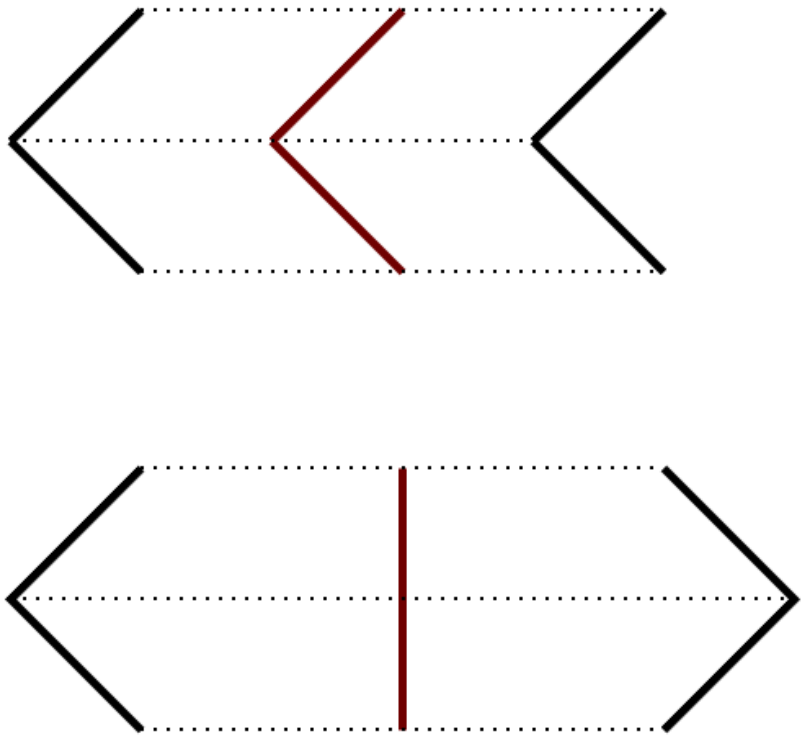
Obrázek 6.1: Výsledek zprůměrování nezarovnaných modelů

Hlavním rozdílem mezi mou a Kellnhoferovou metodou je krok ve kterém se vracím k rigidně registrovaným modelům, zatímco Kellnhofer proved morfování na nerigidně registrovaných modelech. Dalším podstatným rozdílem je to, že Kellnhofer jako výsledný model vytváří novou supersíť, do které přidává nové hrany a vrcholy, zatímco já pouze přesouvám vrcholy jednoho ze vstupních modelů.

V rámci vývoje jsem se pokusil implementovat i postup při kterém by se poslední krok algoritmu aplikoval na vstupní modely rovnou tak jak byly zadány na vstup, tedy na modely před jakoukoliv transformací. V tomto případě by se registrace prováděla jen kvůli nalezení dvojic vrcholů a trojúhelníků. Ukázalo se však, že mezikrok rigidní registrace je nezbytný pro dosažení rozumného výsledku. Jinak by při docházelo při velké odlišnosti modelů, například zrcadlovém natočení, k degeneraci dat, což je vidět na obrázku 6.2. I v případě, že odlišnost není veliká, výsledek nemusí být akceptovatelný, jako je tomu na obrázku 6.1.

Obrázek 6.2: Výsledek zprůměrování zrcadlově natočených modelů

Kdyby byly modely vzájemně pouze posunuty, ničemu by to nevadlo. V případě, že jsou ale kosti natočeny vzájemně jiným směrem, výsledný model bude zdeformovaný. Když jsou vstupní modely vzájemně natočeny zrcadlově, výsledný model se podobá čáře. Celou situaci a důvod, proč dochází k deformaci při natočení kostí, jsem znázornil ve 2D na obrázku 6.3.



Obrázek 6.3: Ukázka průměrování bodů ve 2D

V předchozím odstavci jsem popisoval metodu, kterou se počítají nové souřadnice pro body zdrojové sítě. V původní implementaci se nové souřadnice vždy počítaly jako průměr mezi daným vrcholem a těžištěm trojúhelníka, který mu byl přiřazen z cílové sítě. Pro všechny vrcholy trojúhelníka se spočítal průměr neboli těžiště a s těžištěm se zprůměroval bod zdrojové sítě. To dá úplně stejný výsledek jako kdyby se spočítal průměr nejprve s každým vrcholem trojúhelníka a z těchto tří výsledků by se opět spočítal průměr, který by se použil jako výsledný bod. Ve finální implementaci se průměr mezi vrcholem ze zdrojové sítě a trojúhelníkem z cílové sítě počítá jiným způsobem. Využívají se k tomu barycentrické souřadnice.

Barycentrické souřadnice jsou hodnoty, které vyjadřují polohu bodu vzhledem k trojúhelníku jako vážený průměr vrcholů trojúhelníků. Bod P by se pomocí barycentrických souřadnic x, y, z vyjádřil vzhledem k trojúhelníku s vrcholy A, B, C jako $P = xA + yB + zC$. V našem případě je to trojice čísel v rozmezí 0 až 1, jejichž součet je roven jedné.

K jejich získání jsem v implementaci využil metodu *BarycentricTriangle3D* z knihovny *MyMath*. Barycentrické souřadnice získávám pro nerigidně registrované modely, zatímco průměr stále počítám pro rigidně zaregistrované modely. Ve finální implementaci se pak poloha nového bodu spočítá tak, že se spočítá průměr mezi vrcholem zdrojové sítě a každým vrcholem trojúhelníku a z těchto tří výsledků se spočítá vážený průměr, kde váhy jsou právě získané barycentrické souřadnice. Na obrázku 6.4 je snímek modelu, který je výsledkem zprůměrování 4 různých vstupních modelů pomocí barycentrických souřadnic. Výsledek stejného průměrování bez použití barycentrických souřadnic je pak na obrázku 6.5 vlevo nahoře. Na první pohled je patrné, že při použití barycentrických souřadnic jsou výsledné modely mnohem hladší a více se podobají lidským kostem.

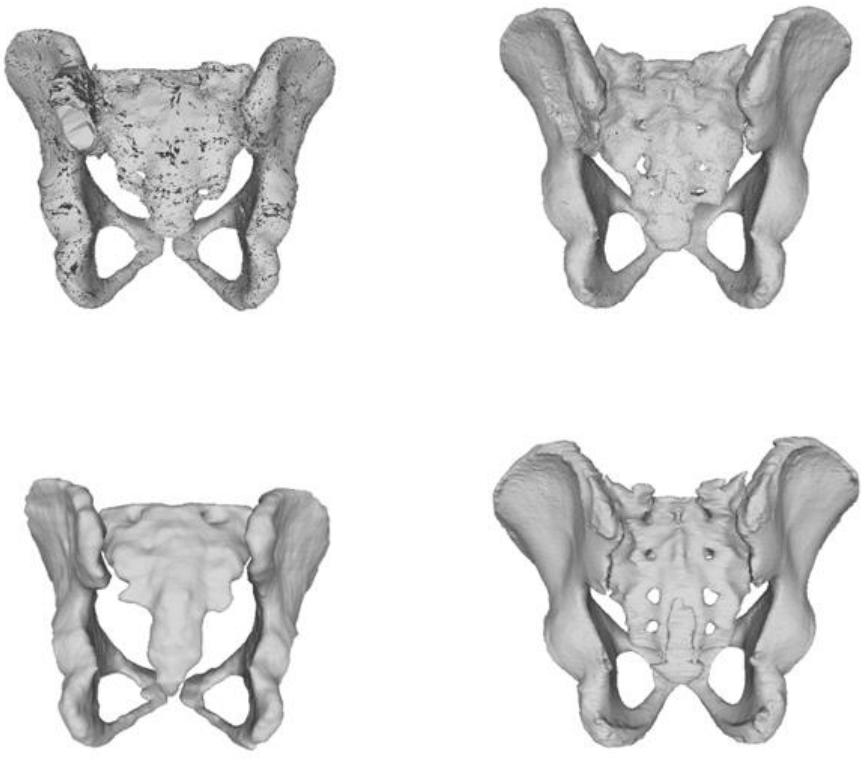
Obrázek 6.4: Výsledek po průměrování s využitím barycentrických souřadnic

6.1.4 Zvolený přístup ke slučování více modelů

V případě, že uživatel zadá na vstup několik datových sad, dá se k jejich slučování přistoupit více způsoby. Prvním možností je sloučit všechny modely najednou, tedy spočítat průměr pro všechny modely naráz. V tomto případě nezáleží na pořadí, v jakém byly modely zadány na vstup. V případě, že by byl program spuštěn dvakrát po sobě se stejnými kosterními modely, avšak zadanými v opačném pořadí, výstup programu by měl být v obou případech stejný. Při zvolení tohoto přístupu mají všechny vstupní modely stejně velký vliv na výsledný model.

Druhým možným přístupem je postupné slučování. Nejprve se sloučí první dva modely a vznikne nový průměrný model, ten se sloučí s dalším a tímto způsobem se pokračuje, dokud se nesloučí všechny modely. Při tomto přístupu záleží na pořadí, v jakém jsou modely zadávány na vstupy. Zprůměrovaný model se bude nejvíce podobat modelu, který byl na vstup zadán jako poslední. Váha každého modelu roste při postupném slučování exponenciálně, každý model bude mít dvakrát větší váhu, než měl model předchozí. Zprůměrujeme dva modely a oba mají na výsledek stejný vliv. Průměrný model zprůměrujeme s dalším modelem, kde další model bude mít stejnou váho jako mají oba modely, ze kterých vznik průměrný model, dohromady.

Druhou zmíněnou metodu jsem zkusil implementovat jako první. Modely se slučují v cyklu, který běží n-1krát, kde n je počet vstupních modelů dané kosti. Při první průchodu cyklem se sloučí první dva modely ze vstupu. Při druhém průchodu se sloučí výsledek prvního průchodu cyklem s následujícím modelem ze vstupu. Výsledek po



Obrázek 6.5: Výsledky postupného slučování pánevních kostí

posledním průchodu cyklem je pak finální zprůměrovaný model. Jako zdrojový model se bere vždy ten s větším počtem vrcholů. Kvůli tomu není výsledný model ovlivněn čistě pořadím, ale i tím, kolik má který model vrcholů.

Abych otestoval, jak výsledný model ovlivní změna pořadí modelů na vstupu, spustil jsem program pro 4 modely pánve. Při první konfiguraci jsem zadal modely z datových sad TLEM2_CT [10], MC22 [9], ICL_MRI [3] a JIA_MRI [7] v tomto pořadí.

Při této konfiguraci se výsledný model nejvíce podobal modelu z datové sady JIA_MRI [7]. Poté byl program spuštěn se stejnými modely, ale v opačném pořadí, největší váhu měl tedy model z datové sady TLEM2_CT [10]. Model pánve z datové sady TLEM2_CT [10] má v křížové kosti viditelné díry, zatímco model z datové sady JIA_MRI [7] žádné viditelné díry v křížové kosti nemá. Kvůli zřejmé odlišnosti obou modelů jsem je zvolil jako ty s největší vahou při spouštění programu. Rozdíl ve výsledcích po spuštění obou konfigurací je ukázán na obrázku 6.5. Nahoře jsou výsledky konfigurací a dole pak vždy model pánve, který byl zadán jako poslední. Modely vlevo patří k první konfiguraci a ty vlevo k druhé. Rozdíl je patrný jednak na tvaru celé pánve, tak například na zachování děr v křížové kosti anebo na velikosti výsledného modelu. Model pánevní kosti vlevo nahoře na sobě má několik tmavých míst. To jsou místa, kde po zprůměrování došlo k překlopení trojúhelníků. K překlopení došlo kvůli tomu, že tento test jsem prováděl ještě předtím, než jsem do implementace přidal počítání průměru pomocí barycentrických souřadnic.

Později jsem se pokusil implementovat i metodu sloučení všech kostí najednou. Tato metoda kopíruje postup slučování postupné metody, avšak provádí kroky na více modelech najednou. Všechny modely se nejprve nerigidně zaregistrují na jeden model s nejméně vrcholy. Následně se vybere ten s největším počtem vrcholů a pro každý se najde na každém modelu nejbližší trojúhelník. Tím je myšlena trojice vzájemně spojených vrcholů v trojúhelníkové síti, která leží k vrcholu nejblíže. Poté se všechny modely vrátí do stavu po rigidní registraci. V posledním kroku se každý vrchol modelu s největším počtem vrcholů zprůměruje s vrcholy každého nejbližšího trojúhelníku nalezeného v předchozím kroku.

Metodu jsem implementoval tak, že jsem pouze zkopíroval funkci pro postupné slučování kostí a pozměnil některé její kroky. Výsledné modely tvarem připomínaly kost, kterou měly, jejich povrch však nebyl vůbec hladký. Kód jsem několikrát prošel a na důvod toho, proč jsou výsledné modely, po spuštění této metody zubaté, jsem zatím nepřišel. To je jeden z důvodů, proč byl ve finální implementaci ponechán přístup postupného slučování.

Obrázek 6.6: Výsledek po sloučení všech vstupních modelů najednou

6.2 Spojování modelů

V prvním části algoritmu, se vzájemně sloučily modely různých kostí, do modelů průměrných. Výstupem program má být jeden model, ve kterém budou všechny modely různých kostí spojeny. Modely se samozřejmě musí spojit tak, jak je tomu v lidském těle. Tedy aby v kloubní jamce pánve byla zasazena stehenní kost, pod stehenní kostí začínala správně nasazená lýtková kost spojená s holenní a tak dále. Program modely, které jsme mu dali na vstup, vnímá jako různé trojúhelníkové sítě, bez jakékoliv souvislosti. Proto jsem se v druhé části zabýval tím, jak nejlépe transformovat zprůměrované modely, tak aby co nejvíce připomínaly kompletní model dolních končetin člověka.

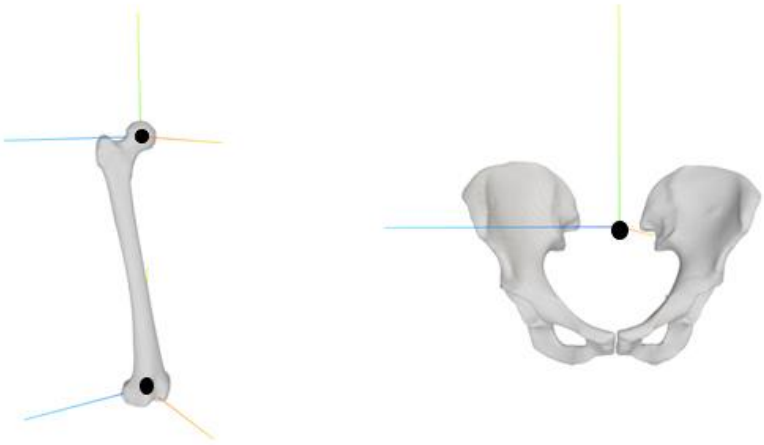
6.2.1 Využití nástroje Staple ke spojení kostí

Při návrhu výsledného nástroje jsem předpokládal, že zprůměrované modely kostí dám na vstup Staplu a jako výstup dostanu spojený model dolních končetin. To byl chybný předpoklad, Staple totiž s modely vůbec nehýbe, na výstupu jsou přesně na stejných místech jako byly na vstupu. Staple přepokládá, že modely datových sad na vstupu budou vždy modely získané z lékařských snímků jednoho pacienta, tím pádem budou v prostoru na správných místech a není důvod provádět transformaci.

To poměrně zkomplikovalo celý proces vývoje. Musel jsem vymyslet, jak pomocí údajů, které generuje Staple, tedy struktur BL, BCS a JCS, provést správné transformace tak, aby na výstupu byly modely umístěny co nejpřesněji.

Staple generuje pro každou kost pouze omezený počet BL, typicky tři až čtyři. Staple vygeneruje na modelu pánve počáteční bod jeho souřadné soustavy a na modelu stehenní kosti vygeneruje body, kde by se nacházely kolenní a kyčelní kloub. Z takových dat se nedá jednoznačně určit, jak přesunout stehenní kost, tak, aby kyčelní kloub zapadl do kyčelní jamky v pánvi.

Poté mě napadlo, že úplně nejjednodušším řešením by bylo, kdyby uživatel zadal na vstup alespoň jeden model, který už je spojený. V tom případě by se na již spojený model všechny další modely zarovnaly pomocí *vtkMeshRegisteru*. Toto by bylo teoreticky ideální řešení, ale pouze v případě, že všechny vstupní sady by měly zhruba stejně velké kosti. Kdyby kosti byly jinak dlouhé, mohly by se ve výsledném modelu překrývat, nebo v opačném případě by mezi kostmi byly moc velké mezery. S tím že kosti



Obrázek 6.7: Ukázka bodů JCS vygenerovaných na modelu stehenní a pánevní kosti

v datových sadách jsou zhruba stejně velké počítat samozřejmě nemůžeme, proto jsem toto řešení také vyloučil.

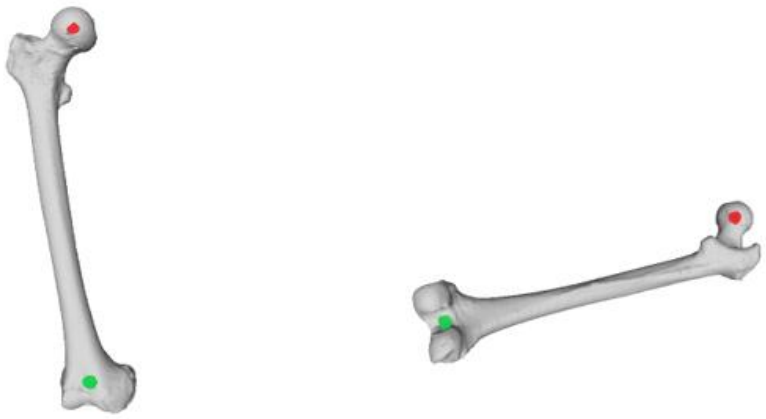
Dále jsem zkusil zkombinovat nápady z obou předchozích úvah. Nějak využít JCS, které generuje Staple a zároveň využít toho, že nějaké z datových sad zadané uživatelem mají už spojené některé kosti. Pokud se spustí Staple s datovou sadou, kde jsou již spojené kosti, získáme JCS, ze kterých můžeme vypočítat, na jaké pozici má být kyčelní kloub stehenní kosti, vzhledem k počátku souřadné soustavy pánve. Poté můžeme provést sloučení všech kostí a výslednou stehenní kost přesunout tak, aby byla vzhledem k pánvi ve stejné vzdálenosti jako na začátku v již spojeném modelu.

Základem tohoto postupu je spuštění Staplu na datových sadách ještě před jejich slučováním. To s sebou přináší dvě výhody. První výhodou je to, že Staple zvládá načíst na vstupu soubory ve formátu *mat*. Stejné soubory pak ukládá do výstupní složky, avšak ve formátu *obj*. Všechny datové sady, na kterých jsem nástroj při vývoji testoval, jsem převzal z gitlabu Staplu [5] a ty jsou právě ve formátu *mat*. Jedná se o datové sady, které jsem popisoval ve čtvrté kapitole. Pokud by měly tyto modely být vstupem nástroje pro slučování, musely by se nejprve převést z formátu *mat* do formátu *obj*. K tomu bohužel neexistuje v C++ žádná volně dostupná knihovna, proto bych musel program, který tuto akci provede implementovat sám. Tím, že se před slučováním spustí Staple tento problém odpadá.

Druhou výhodou spuštění Staplu před slučováním je to, že nám bude stačit pouze jedna datová sada, kde budou modely kostí spojeny dohromady, abychom podle jejich JCS dokázali spojit i výsledné sloučené modely. Staple totiž dokáže například spočítat na stehenní kosti bod, kde bude kyčelní kloub, aniž by k tomu potřeboval model pánevní kosti. Tím pádem dostaneme JCS pro všechny vstupní modely, ale stačí nám pouze jedna datová sady, kde budou modely spojeny, abychom podle jejích JCS dokázali spočítat transformaci, kterou je nutno provést na zprůměrované kosti.

6.2.2 Transformace získaných JCS

Protože kloubní souřadné systémy (JCS) se budou ve Staplu generovat ještě před slučováním kostí, je třeba na body JCS aplikovat stejné transformace, jako byly aplikovány na modely během slučování. Pokud máme například dva modely stehenní kosti a pro oba známe souřadnice kyčelního kloubu, potřebujeme vypočítat souřadnice kyčelního kloubu pro výsledný model. To spočítáme tak že oba body JCS jednoduše zprůměrujeme, stejně jako se průměrují vrcholy modelů kostí při slučování. Protože při slučování průměrujeme



Obrázek 6.8: Ukázka transformace JCS bodů na stehenní kosti

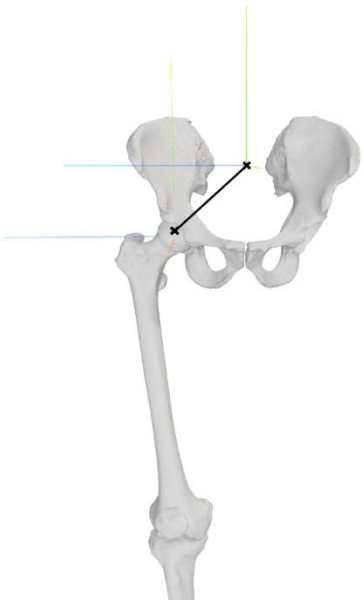
vrcholy rigidně zaregistrovaných kostí, musíme body JCS vygenerované ve Staplu transformovat společně s kostmi.

Když vtkMeshRegister provádí slučování, transformuje modely dvakrát, jednou při počátečním zarovnání a jednou při rigidní registraci. V obou případech k tomu využívá funkci *Transform* knihovny *ExtendedMesh*. Tato funkce projde všechny body trojúhelníkové sítě, která reprezentuje model kosti, a každý vynásobí transformační maticí o velikost 4x4. Transformační matici funkce přebírá jako parametr od *vtkMeshRegisteru*. Přidal jsem do *vtkMeshRegisteru* funkce, které vrací tyto transformační matice. Jednu funkci pro transformační matici použitou počátečním zarovnání a druhou pro transformační matici použitou při rigidní registraci. Těmito maticemi pak vynásobím body získané z JCS pro danou kost. Teoreticky by měly tedy body JCS každého modelu ležet na podobném místě, když se na podobné místo díky rigidní registraci přesunuly i modely kostí. Má to však komplikaci.

V této části implementace fungovalo slučování kostí tak, že se vzaly dva první modely, ten s více vrcholy se přesunul na místo, kde byl ten s méně vrcholy a na tomto místě se sloučily do výsledného modelu. Výsledný model se poté vzal společně s dalším modelem a provedla se stejná operace. Vybral se ten s menším počtem vrcholů a na jeho pozici se druhý model přesunul. To lehce komplikuje transformaci bodů získaných z JCS. Při transformaci každého modelu, bychom museli sledovat, zda je tento model již tvořen více sloučenými modely, a pokud ano, transformovat společně s modelem i body z JCS všech modelu, které tvoří model, který aktuálně přesouváme. To však také není tak snadné, protože ve výsledném modelu má každý vstupní model jinak velkou váhu neboli jinak velký vliv na pozici jeho vrcholů. To by se dalo vyřešit tak, že by se po každém sloučení dvou modelů vypočítal z jejich JCS bodů jeden průměrný.

To by však celý problém spojování výsledných modelů vyřešilo pouze částečně. Představme si situaci, kdy se slučuje několik datových sad. Ve všech datových sadách jsou modely na začátku správně spojeny, avšak každá datová sada se nachází na jiném místě, navíc s jinou rotací. Spustíme Staple se všemi datovými sadami na vstupu. Tím získáme pro každou datovou sadu vlastní JCS. U každé datové sady tedy víme, o kolik je posunuta stehenní kost vůči pánevní nebo o kolik je posunuta holenní vůči stehenní a tak dále. Zároveň máme jistotu, že modely v jednotlivých datových sadách jsou vzájemně správně natočeny. Dále se pro každou kost provede sloučení jejích modelů ze všech datových sad. Výsledné modely jsou vždy na stejném místě, jako byl vstupní model

s nejmenším počtem vrcholů. To ale při dosavadní implementaci dopředu nevíme, který model bude. Výsledný model pánevní kosti skončí na místě, kde byl vstupní model z první datové sady. Výsledný model stehenní kosti skončí na místě, kde byl vstupní model z datové sady druhé. Protože společně s modely jsme transformovali i body JCS, víme nyní kdy se nachází JCS bod výsledného modelu pánevní kosti, kde se nachází JCS bod pro kyčelní kloub výsledného modelu stehenní kosti a také víme, jak byly tyto body o sebe vzdáleny na vstupních modelech. Stačilo by provést translaci, která zajistí, aby tato vzdálenost odpovídala. Jak jsou však výsledné modely vzájemně natočené netušíme a nevíme jak máme provést rotaci.



Obrázek 6.9: Vzdálenost bodů JCS ve vstupní sadě TLEM2_CT [10]

Jako nejrozumnější řešení celého problému spojování výsledných modelů mi proto přijde postup, při kterém nebudeme muset rotaci modelů vůbec provádět Využívám toho, že ve *vtkMeshRegisteru* se dá určit, který z modelů je cílový. Aby toto řešení správně fungovalo musí být splněna jedna podmínka. Uživatel musí na vstup zadat alespoň jednu datovou sadu, která obsahuje korektně spojené modely všech kostí, které mají být spojeny i na výstupu. Právě modely této datové sady budou při každém slučování voleny jako ty cílové. Tím se docílí toho, že výsledné modely budou na zhruba správných místech hned po sloučení, a navíc budou správně natočené. Akorát mezi nimi budou příliš velké mezery, nebo se naopak budou překrývat. Nemusí se tedy provádět žádná rotace, jen translace. Ta se udělá velice jednoduše podle JCS. Další výhodou toho, že od začátku víme, který model bude cílový, je to, že na jednotlivé JCS každého modelu stačí provést jen jednu transformaci.

Ve finální implementaci je sada, která obsahuje na vstupu nejvíce modelů, volena jako cílová. Postup slučování modelů se tedy oproti předchozí implementaci musel pozměnit. Původně se modely slučovaly podle pořadí, v jakém byly zadány, s tím, že cílový byl vždy ten s menším počtem vrcholů. Ve finální implementaci se v první iteraci vezme model kosti z datové sady, která byla označena za cílovou a sloučí se s prvním modelem v pořadí. Výsledek sloučení se pak v další iteraci opět bere jako cílový, aby výsledný model po sloučení všech vstupních modelů byl v zhruba stejné poloze, jako model z cílové datové sady.

7. Testování a zhodnocení výsledků

Program jsem testoval pro nejrůznější vstupy během celého vývoje. Výsledky některých testů jsem již ukázal v předchozích kapitolách. V následující kapitole budou ukázány výsledky některých dalších testů a bude poukázáno na modely pro které výsledný program nefunguje optimálně či vůbec. Modely kostí, které jsem měl pro testování k dispozici napříč všemi sadami byly model pánevní kosti, potom modely stehenní, lýtkové, holenní a hlezenní kosti. Potom ještě patní kosti, čéšky a prstů. Pro všechny modely kostí platí, že byly jak z nohy levé, tak pravé. Modely lýtkové a holenní kosti byly vždy zahrnuty v jednom modelu.

Před ukázáním výsledků testování je třeba upozornit na některé modely, které byly z implementace vynechány, jinými slovy pro ně program nefunguje a nejde je do něj zadat na vstup. Jde o modely patních kostí a prstů u nohou. Důvodem je to, že některé modely těchto kostí se nepodařilo zaregistrovat pomocí *vtkMeshRegisteru*. Dochází tam k chybě kvůli tomu, že modely jsou tvořeny mnoha oddělenými komponentami. Při prvním kroku se *vtkMeshRegister* totiž snaží ze sítí odstranit oddělené komponenty. V tomto případě by ze sítí skoro nic nezbylo.

Další modely, na které je dobré upozornit jsou modely čéšky. Pro ty Staple negeneruje body JCS, proto jsou na výstupu ukládány do vlastních souborů, a ne souboru s výsledným modelem. Důvodem je to, že pro ně neznáme body JCS a mohly by na výsledném modelu ujíždět a kazit celkový dojem.

Jako první jsem během testování vyzkoušel sloučení datových sad, co obsahovaly různé modely pánevních, stehenních a holenních kostí. Výsledek jsou na obrázku 7.1 i s ukázkou obsahu konfiguračního souboru. Výsledný model připomíná dolní končetiny člověka i tvarem jednotlivých kostí i jejich rozložením v prostoru. Je vidět, že model levé stehenní kosti je o něco delší než model pravé stehenní kosti, proto trochu zajíždí do modelu pánevní kosti, avšak pouze minimálně. Výsledek testu bych hodnotil jako uspokojivý.

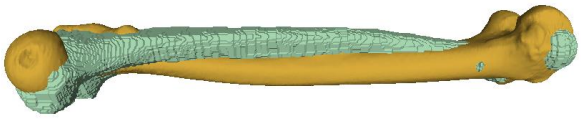
```
leg -femur_r tlem_mri/femur_r.mat
    -femur_l tlem_mri/femur_l.mat
    -tibia_r tlem_mri/tibia_r.mat
    -tibia_l tlem_mri/tibia_l.mat

pelvis -pelvis mc22/pelvis.mat
leg    -femur_r mc22/femur_r.mat
       -femur_l mc22/femur_l.mat
       -tibia_r mc22/tibia_r.mat
       -tibia_l mc22/tibia_l.mat

pelvis -pelvis  icl/pelvis.mat
leg    -femur_r icl/femur_r.mat
       -tibia_r icl/tibia_r.mat
```

Obrázek 7.1: Výsledek prvního testu

Dalším testem, který jsem provedl bylo sloučení několika modelů stehenní kosti. Výsledkem tohoto testu byl model kosti, který se možná zdá na první pohled optimální, avšak není tomu tak. Obě hlavice modelu stehenní kosti si byly dost podobné, avšak u normální lidské kosti jsou oba konce kosti odlišné. Na jednom konci kosti je jasně zakulacený kyčelní kloub. Na obrázku 7.2 je vidět, že u výsledného modelu tohoto testu tomu tak není. Po analýze problému jsem zjistil, že se modely kostí na sebe zaregistrovaly

Obrázek 7.2: Výsledek druhého testu

opačně, než by měly. Na obrázku 7.3 je vidět, že první model má po rigidní registraci kyčelní kloub tam kde má druhý model kloub kolenní a naopak. Po několika spuštění jsem se zjistil to, že k tomu dochází pouze u některých dvojic modelů, záleží na vstupních datech a jejich pořadí. Chybu jsem lokalizoval ve *vtkMeshRegisteru* v metodě *UnifyPositions*. Tato metoda ke vstupním trojúhelníkovým sítím vytvoří nové hrubší sítě, na kterých bude hledat transformaci pro nejlepší zarovnání. Velikosti těchto hrubších sítí jsou defaultně nastaveny na 300 vrcholů, zatímco vstupní sítě mívají desítky tisíc vrcholů. Poté metoda provede analýzu hlavních komponent a zkoumá možnosti, jak natočit model podle hlavních os, tak, aby byly vzdálenosti mezi vrcholy co nejmenší. Tím, že stehenní kost má celkem souměrný tvar a vstupní modely jsou od různých pacientů, tak pravděpodobně pro nesprávnou transformaci vyjde o něco menší celkový rozdíl vzdáleností mezi vrcholy. Jelikož na body JCS se uplatňují stejné transformace, jako na model kosti, je

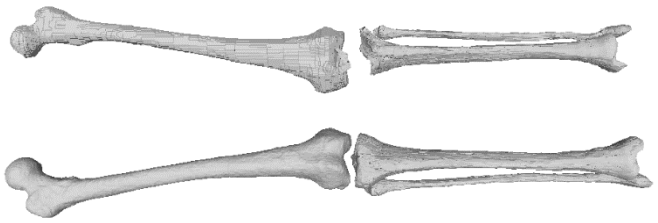


Obrázek 7.3: Výsledek rigidní registrace modelů stehenní kosti

model špatně posunut i ve výsledném modelu, pokud k této situaci dojde. U většiny kombinací modelů stehenních kostí k tomuto problému nedochází, avšak stále existuje šance, že když uživatel zvolí náhodnou kombinaci a pořadí modelů stehenních kostí tak k němu doje. Proto nemohu tento test hodnotit jako zcela kladný. Je ale dobře, že test tento nedostatek odhalil a víme o něm.

Jako poslední jsem udělal test sloučení modelů bez zadání pánevní kosti, abych se přesvědčil, že se modely umí dobře spojit i když model pánve nebude zadán na vstup.

Výsledek je vidět na obrázku 7.4. Jde vidět, že mezera mezi modely na levé straně těla je o něco větší než na pravé straně. To však zřejmě není chyba posunutí, ale toho, že model levé lýtkové kosti je u kolene zdeformován.



Obrázek 7.4: Výsledek třetího testu

Celkově bych výsledky testování i celé práce hodnotil pozitivně. Hlavní cílem bylo vytvoření nástroje, který automaticky sloučí vstupní modely do jednoho průměrného. To nástroj dělá, a pro většinu dat to dělá velmi dobře. Jako pozitivum bych hodnotil například i to, že vzhled výsledného modelu se dá ovlivnit pořadím zadání modelů. Uživatel může zkusit různá pořadí a z nich si vybrat nejlepší výsledek. Nevýhodou je však to, že tento model není skutečně průměr, ale vážený průměr. Aby byl výsledný model opravnu průměrný, musely by se modely sloučit najednou. To je prostor pro zlepšení při další práci. Naimplementovat metodu, která sloučí kosti tak aby výstup byl stejný při jakémkoliv pořadí zadání modelů. Jinými slovy, aby každý vstupní model měl na výsledek stejný vliv. V ideálním případě by byly v implementaci ponechány obě metody, a uživatel by si mohl vybrat.

Další věc, kterou by šlo zlepšit při další práci je slučování modelů stehenních kostí. Myslím tím opravení chyby, která vzniká při počátečním zarovnání modelů u některých konfigurací. Chybu jsem odhalil až během testování a myslím že při další práci by mohla být odstraněna. Spojování modelů v kloubech bych také hodnotil spíše pozitivně. Algoritmus spojování funguje dobře a jednoduše, avšak za cenu toho, že uživatel musí vždy zadat jeden již spojený model.

Při další práci by bylo také vhodné pozměnit logiku načítání vstupních dat. Bylo by například šikovné dovolit uživateli zadávat do programu i příkazy a nechat ho s programem komunikovat. Jedním příkazem by se zadávaly modely, dalším by se spouštělo slučování kostí a jiným by se třeba vypisovalo jaké dosud byly zadány do programu modely. Uživatel by měl větší přehled, co se v programu děje a byla by v něm jednodušší orientace. Také by bylo vhodné umožnit uživateli vyžádat si na výstup modely průběžně během slučování.

8. Závěr

V rámci této práce byl navržen a následně implementován nástroj pro automatické slučování kosterních modelů do generických muskuloskeletálních modelů. V prvním kroku algoritmu se modely jednotlivých kostí sloučí do průměrných. Bylo vyzkoušeno několik přístupů k problematice slučování kosterních modelů, reprezentovaných trojúhelníkovými sítěmi. Výsledný algoritmus vychází z metody přímého multi-morphingu, avšak s několika upravenými kroky. Celý princip funguje na základě posouvání vrcholů jednoho ze vstupních modelů tak, aby výsledný model byl průměrem všech vstupních. Jako nejlepší řešení pro získání průměrného modelu vyšlo průměrování rigidně zaregistrovaných modelů. Ukázalo se, že bez této registrace by průměrování nemohlo fungovat.

Průměr se počítá vždy pro vrchol z jedné sítě a trojúhelník z druhé sítě. K tomu byla využita metoda, ve které se využíval čistě průměr, i metoda, ve které se využíval vážený průměr spočítaný pomocí barycentrických souřadnic. Výsledné modely měly hladší povrch při použití druhé metody, proto byla ponechána ve finální implementaci. Dalším klíčovým faktorem při slučování je pořadí, v jakém jsou kosti zadávány na vstup. Modely se slučují postupně a výsledný model se nejvíce podobá vždy tomu, který byl do průměrného přidán jako poslední.

Druhým krokem algoritmu je posunutí zprůměrovaných modelů tak, aby byly vůči sobě ve stejné poloze jako jsou v lidském těle. K tomu bylo využito nástroje Staple. Po původním nastudování nástroje Staple bylo v plánu celý druhý krok udělat pomocí tohoto nástroje. Během hlubší analýzy se však ukázalo, že Staple posun modelů neprovádí, protože je očekává na vstupu již posunuté správně. Proto se posun provádí pomocí souřadnic kloubů, které Staple generuje. Základním principem je vygenerování souřadnic kloubů na správně spojeném modelu na vstupu. Na výstupu se pak posunou modely tak, aby vzdálenosti kloubů odpovídali vzdálenostem na vstupním modelu.

To je poměrně jednoduché a dobře funkční řešení, avšak jeho nedostatkem je to, že se na vstupu očekává minimálně jeden již spojený model, který kombinuje všechny modely kostí, které budou zahrnuty ve výsledném modelu. Vzhledem k tomu, že modely jsou správně spojeny skoro ve všech vstupních datových sadách, to není problém, pouze věc, na kterou se musí myslet při sestavování vstupní konfigurace.

Ve všech testovacích scénářích připomínaly výsledné modely lidské kosti a vždy dobře kombinovaly rysy vstupních modelů. Jediné výsledky, které nelze nehodnotit jako zcela kladné jsou některé výsledky sloučení stehenních kostí, kdy došlo ke špatnému počátečnímu zarovnání modelů.

Práci lze uzavřít tím, že nástroj je schopen dobře slučovat vstupní modely do průměrných i je přesouvat v prostoru tak, aby připomínaly lidské dolní končetiny. Spojení lze však provést pouze pokud uživatel zadá na vstup jeden již spojený model.

Literatura

[1] *Overview of OpenSim Workflows.* [Online] [Citace: 10. 10 2022.] https://simtk-confluence.stanford.edu:8443/display/OpenSim/Overview+of+OpenSim+Workflows#OverviewofOpenSimWorkflows-CommonPre-ProcessingSteps.

[2] Doc. Ing. Josef Kohout, Ph.D. Muscle Wrapping 2.0. *gitlab.com.* [Online] [Citace: 10. 10 2022.] https://gitlab.com/besoft/muscle-wrapping-2.0.

[3] Luca Modenese, Jean-Baptiste Renault. Journal of Biomechanics. *Automatic generation of personalised skeletal models of the lower limb from three-dimensional bone geometries.* [Online] 12. 2 2021. [Citace: 10. 10 2022.] https://www.sciencedirect.com/science/article/pii/S0021929020306102.

[4] Kellnhofer, Petr. *Non-rigid Transformations for Musculoskeletal Model.* 2012.

[5] Luca Modenese, Jean-Baptiste Renault. *modenaxe/ msk-STAPLE.* [Online] [Citace: 18. 11 2022.] https://github.com/modenaxe/msk-STAPLE.

[6] Erica Montefiori, Luca Modenese, Robert Di Marco, Silvia Magni-Manzoni, Clara Malattia, Maurizio Petrarca, Anna Ronchetti, Laura Tanturri de Horatio, Pieter van Dijkhuizen, Anqi wang, Stefan Wesarg, Marco Viceconti, Claudia Mazza. *Linking Joint Impairment and Gait Biomechanics in Patients with Juvenile Idiopathic Arthritis.* [Online] 20. 5 2019. [Citace: 8. 12 2022.] https://link.springer.com/article/10.1007/s10439-019-02287-0#article-info.

[7] *An image-based kinematic model of the tibiotalar and subtalar joints and its application to gait analysis in children with Juvenile Idiopathic Arthritis.* [Online] 6. 3 2019. [Citace: 8. 12 2022.] https://www.clinicalkey.com/#!/content/playContent/1-s2.0-S0021929019300016?returnurl=https:%2F%2Flinkinghub.elsevier.com%2Fretrieve%2Fpii%2FS0021929019300016%3Fshowall%3Dtrue&referrer=.

[8] Marco Viceconti, Gordon Clapworthy, Serge Van Sint Jan. The Journal of Physiological Sciences. *The Virtual Physiological Human — A European Initiative for in silico Human Modelling* . [Online] 21. 10 2008. [Citace: 8. 12 2022.] https://www.jstage.jst.go.jp/article/physiolsci/58/7/58_7_441/_article.

[9] Erica Montefiori, Barbara M. Kalkman, William H. Henson, Margaret A. Paggiosi, Eugene V. McCloskey, Claudia Mazza. *MRI-based anatomical characterisation of lower-limb muscles in older women.* [Online] 1. 12 2020. [Citace: 8. 12 2022.] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0242973

[10] V. Carbone, R. Fluit, P. Pellikaan, M.M. van der Krogt, D. Janssen, M. Damsgaart, L. Vigneron, T. Feilkas, H.F.J.M. Koopman, N. Verdonschot. Journal of Biomechanics. *TLEM 2.0 – A comprehensive musculoskeletal geometry dataset for subject-specific modeling of lower extremity.* [Online] 2015, 2022. 18 3. https://www.clinicalkey.com/#!/content/playContent/1-s2.0-S0021929014006885?returnurl=https:%2F%2Flinkinghub.elsevier.com%2Fretrieve%2Fpii%2FS0021929014006885%3Fshowall%3Dtrue&referrer=.

[11] Jan, S. Van Sith. *The VAKHUM project: virtual animation of the kinematics of the human.* [Online] 8. 8 2006. [Citace: 8. 12 2022.] https://www.tandfonline.com/doi/abs/10.1080/14639220412331529591.

[12] Inc., Kitware. VTK: The Visualisation Toolkit. [Online] 12. 4 2022. https://vtk.org/.

[13] D., Paley. Principles of deformity correction. *Radiology Key.* [Online] 2001. [Citace: 20. 1 2023.] https://radiologykey.com/lower-limb-alignment/.