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

Diplomová práce

Generování kostry objektů reprezentovaných trojúhelníkovými sítěmi

Plzeň, 2015 Michal Žák

Prohlášení

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

V Plzni dne 21. června 2015

Michal Žák

Abstract

This diploma thesis deals with automatic generation of animation skeleton for humanoid objects which are represented by triangle meshes. After research focused on existing similar methods, new original method was invented, implemented and afterwards tested on input data set consisted of meshes representing humanoids. Generated animation skeleton keeps desired topological structure and contains movement limits in every joint. In addition to the skeleton generation, we implemented testing application which allows user to animate generated skeleton and original mesh by using inverse kinematics.

Abstrakt

Tato diplomová práce se zabývá automatickým generováním animační kostry pro humanoidy reprezentované trojúhelníkovými sítěmi. Na základě existujících příbuzných metod byl vytvořen a naimplementován vlastní postup, který byl poté testován na množině vstupních dat, modelů reprezentujících humanoidy. Výsledná animační kostra dodržuje předem danou topologickou strukturu a obsahuje definice omezení volnosti pohybu v jednotlivých kloubech. Kromě samotného generování byla naimplementována také testovací aplikace, ve které uživatel může interaktivně rozhýbat vzniklou kostru a původní model užitím inverzní kinematiky.

Poděkování

Chtěl bych poděkovat panu doc. Ing. Josefu Kohoutovi, Ph.D. za cenné rady a připomínky při vedení diplomové práce.

Obsah

1 Úvod

V počítačové grafice se při řešení problémů střetávají lidé různých zaměření, ostatně obor sám se nachází na pomezí informatiky, matematiky a estetiky. V řadě aplikací zejména konzumního charakteru nepostačuje pouze její technická funkčnost, uživatelé podvědomě kladou důraz na výtvarnou stránku. Na vývoji pak pracují vedle programátorů a analytiků také výtvarníci, kteří tvoří samotný grafický obsah. K tomu jim pomáhají různé komplexní nástroje pro modelování, kresbu, úpravu fotografií atd.

Největší překážkou je pak samotné přenesení vytvořeného grafického obsahu do cílové aplikace. Tyto úpravy často nepožadují žádnou invenci nebo estetické cítění, nicméně mohou být časově náročné. Takto tráví výtvarník čas řemeslnou prací, místo níž by mohl produkovat další originální obsah. Pro programátory v oboru počítačové grafiky se tak naskytuje poměrně široké pole pro tvorbu nástrojů, které usnadní produkci grafického obsahu.

Jedním z typů dat, které jsou užívány při vizualizacích, jsou trojúhelníkové sítě. V této podobě jsou objekty reprezentovány pomocí svého povrchu, respektive aproximací povrchu trojúhelníky, které mohou sdílet své hrany a vrcholy.

Problém nastává u rozpohybování (animace) sítí. Historicky byl omezujícím faktorem výpočetní výkon, používaly se primitivní metody jako například pohybující se hierarchie neměnných objektů či animace *per-vertex*, tedy definování pohybu pro každý vrchol sítě zvlášť. S nárůstem výkonu se prosadily tzv. kosterní animace, kdy pohyb povrchu tělesa řídí kostra značně jednodušší než původní síť. Zřejmou výhodou je úspora ukládaných dat, neboť stačí zachytit stav kostry v čase a přiložit statickou podobu trojúhelníkové sítě. Animátor navíc dostává řídicí mechanismus, kterým může objekt intuitivněji ovládat oproti animaci *per-vertex*. Na našeho výtvarníka však spadá také nevděčná tvorba této kostry. Uvědomme si, že pro řadu modelů má kostra podobný charakter, zejména pak u modelů zvířat nebo lidí, kdy apriorně známe například počet končetin, jejich přibližné rozložení a poměry délek. Pro specifická užití navíc musí výtvarník do kostry doplnit informaci o omezení volnosti v kloubech a další pomocné údaje.

Diplomová práce porovnává již vzniklé metody pro automatickou tvorbu kostry.

Na základě průzkumu pak vytvoříme nástroj, který pro humanoida ztvárněného trojúhelníkovou sítí, u něhož uplatníme apriorní znalosti, vytvoří jeho animační kostru s omezeními volnosti pohybu v kloubech. Výsledek bude předveden v aplikaci, která umožní iteraktivně deformovat model pomocí inverzní kinematiky užité na vygenerovanou kostru.

2 Metody extrakce kostry

K automatické extrakci kostry vznikla řada metod, které jsou založeny na různých přístupech. Práce se zabývá kostrami složenými ze křivek *(curve skeletons)*, které lze definovat jako jednorozměrnou strukturu zjednodušující původní objekt, přičemž zachovávají topologickou a geometrickou informaci.

Kostra se skládá z dílčích křivek nazývaných *kosti*, zpravidla uspořádaných do hierarchie. Některé metody produkují kostru složenou pouze z úseček. Při cílovém užití vzniklé kostry ji obvykle rozšíříme do podoby parametrizovaného Frenetova trojhranu (kvůli jednoznačnému určení lokální soustavy souřadnic).

Objekt bývá obvykle reprezentován povrchově pomocí trojúhelníkové sítě, které kromě pozic vrcholů a jejich spojení v trojúhelníky obsahují informaci o sousednostech. V tomto případě nastupují geometrické a topologické metody tvorby kostry, v případě objemové (voxelové) reprezentace metody pracující v diskrétním prostoru.

2.1 Medial axis transform

Harry Blum v roce 1967 [6] prvně formalizoval střední osu objektu. Z jeho definice v podstatě všechny novější postupy vycházejí, nicméně přidávají další kritéria a upouštějí od přesného vyjádření.

Obrázek 2.1: Ukázka střední osy dvourozměrného objektu (vyznačena přerušovaně).

Střední osa objektu *(medial axis)* je definována jako množina bodů Q majících stejnou vzdálenost k alespoň dvěma bodům hranice objektu, aniž by se jiný bod náležící hranici nacházel blíže k Q.

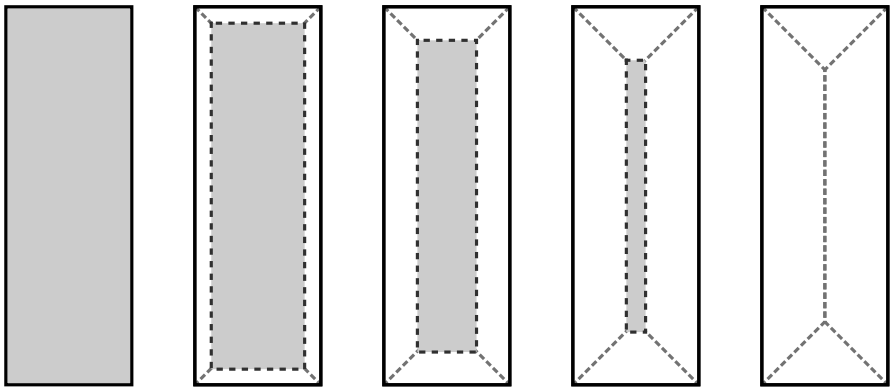
Formálně lze definici zapsat matematickou formulací:

$$\forall X \in E^3 : X \in \mathrm{MA}(O) \iff \exists A_1, A_2 \in \mathrm{surface}(O) : \|A_1X\| = \|A_2X\| \tag{2.2}$$

$$\wedge \nexists B \in \mathrm{surface}(O) : \|BX\| < \|A_1X\| \tag{2.3}$$

kde O značí vstupní objekt, $\mathrm{MA}(O)$ formálně střední osu objektu O a $\mathrm{surface}(O)$ povrch objektu. Příklad střední osy objektu je ukázán na obrázku 2.1.

Blum ve svém článku použil definici pomocí propagace události v čase: Na počátku vznikne na hranici objektu událost, která se dál homogenně šíří prostorem. Pro příklad si můžeme objekt představit jako louku a událost jako šířící se požár založený na její hranici. Daným místem prostoru může událost projít pouze jednou, při použití naší analogie lze říct, že požár nepokračuje zpět na spáleniště. Bod, ve kterém se v jednom čase setká více událostí pocházejících z odlišných zdrojů (jiných bodů hranice), tvoří střední osu objektu. Proces je znázorněn na obrázku 2.4.



Obrázek 2.4: Medial axis transform, ilustrace propagace události v čase. Vlevo se nachází původní objekt, vpravo objekt s výslednou střední osou. V místech nespojitosti vzniká střední osa objektu.

V souvislosti se střední osou objektu se hovoří o tzv. *medial axis transform*. Jedná se o transformaci, při které povrchem reprezentovaný objekt O nahradíme množinou maximálních prázdných hyperkoulí S_i tak, že:

$$\bigcup_{\forall i} S_i \;=\; \bar{O} \tag{2.5}$$

Hyperkoule se mohou vzájemně protínat, povrch objektu O však nesmí být obsažen uvnitř objemu libovolné hyperkoule, pouze na jejím povrchu (pro středy hyperkoulí platí formulace 2.2). Tato transformace plně zachovává původní tvar objektu, spolu se střední osou musíme však uchovat i poloměr vzniklých hyperkoulí.

Sluší se poznamenat, že ačkoli disponujeme elegantní definicí střední osy, využití její surové podoby není příliš žádoucí. Zaprvé, hledání střední osy je výpočetně náročný úkon, zadruhé, výsledná křivka je obtížně parametrizovatelná. Jako poslední bod zmiňme, že tvorba střední osy je vysoce citlivá na šum na povrchu objektu – sebedrobnější nuance tvoří nové větve střední osy.

Střední osu lze aproximovat pomocí po částech lineárním skeletem tvořeného vnitřními póly Voroného diagramů. Techniku detailně rozebírá Amenta et al. [3] za účelem rekonstrukce povrchu z navzorkovaných bodů. Mezi další aplikace střední osy patří například strukturní popis objektů.

2.2 Metody založené na geometrii a topologii

V tomto případě nijak neměníme charakter vstupních dat, během výpočtu využíváme vstupní trojúhelníkovou síť. Díky tomu můžeme ušetřit výpočetní čas, který bychom museli věnovat případné konverzi reprezentace dat.

2.2.1 Smrštění modelu s následnou redukcí geometrie

Řešení problému, které navrhuje Au et al. [4], se sestává z několika fází. Nejprve iterativně smršťujeme trojúhelníkovou síť, po dosažení určeného objemu přichází na řadu redukce geometrie, přičemž vzniká jednorozměrná kostra objektu. Na závěr provedeme úpravu pozic vrcholů kostry.

Smrštění sítě

Během smršťování sítě dochází k vyhlazování detailů a posunu vrcholů proti směru vypočtené normály. Řešíme přeurčenou soustavu rovnic 2.6 pro neznámý

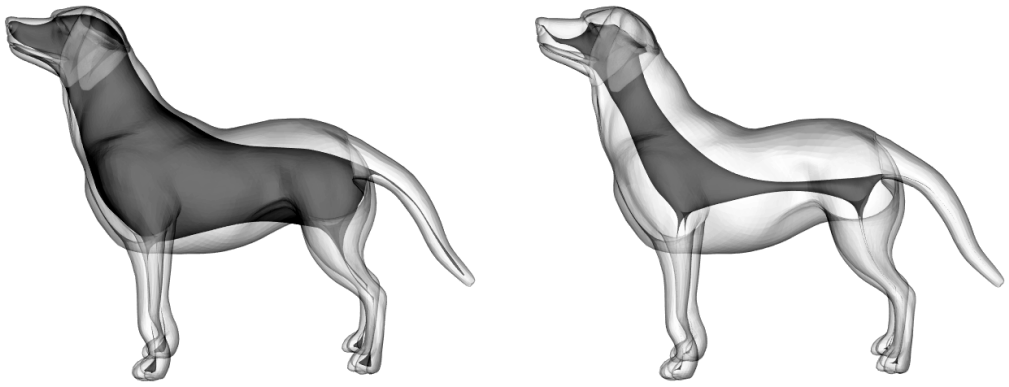
vektor pozic V', po jeho nalezení přiřadíme $V \leftarrow V'$ a výpočet opakujeme, dokud není dosažena zastavující podmínka.

$$\begin{bmatrix} \mathbf{W}_L\mathbf{L} \\ \mathbf{W}_H \end{bmatrix} \mathbf{V}' = \begin{bmatrix} 0 \\ \mathbf{W}_H V \end{bmatrix} \tag{2.6}$$

Popišme si význam jednotlivých členů soustavy. Vektor $\mathbf{V}$ značí všechny aktuální pozice vrcholů sítě, $\mathbf{V}'$ pozice nové, hledané. Diagonální matice $\mathbf{W}_L$ (vynucující setrvání v pozici) a $\mathbf{W}_H$ (zesilující kontrakci) zanášejí do výpočtu potřebné váhy. Matice $\mathbf{L}$ provádí diskrétní laplaceovské „vyhlazování“ proti směru normály a je definována následovně:

$$\mathbf{L}_{ij} = \begin{cases} \omega_{ij} = cotg\,\alpha_{ij} + cotg\,\beta_{ij} & \text{je-li } (i,j) \text{ hranou mezi vrcholy } v_i \text{ a } v_j \\ \sum_{(i,k)\in E} -\omega_{ik} & \text{je-li } i = j \\ 0 & \text{v jiném případě} \end{cases} \tag{2.7}$$

přičemž úhly α_{ij} a β_{ij} jsou protilehlé úhly vůči hraně (i,j). Jak již bylo řečeno, řešení soustavy musíme opakovat, jedna iterace k řešení nestačí. Za zastavovací podmínku autoři článku volí dosažení ϵ-násobku původního objemu, případně překročení maximálního možného počtu kroků (řešení soustavy rovnic).



Obrázek 2.8: Smršťování modelu. Konektivita původní geometrie je zachována.

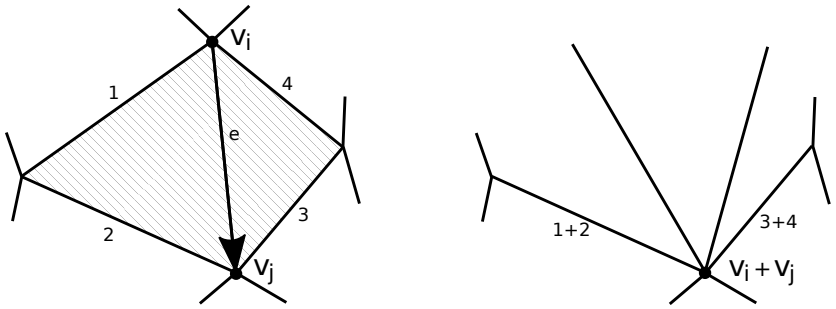
V článku doporučují váhy na počátku nastavit na hodnoty $\mathbf{W}_H^{(0)} = 0$ a $\mathbf{W}_L^{(0)} = 10^{-3}\sqrt{A}$. Úprava vah probíhá po každém kroku:

$$\begin{aligned}&\mathbf{W}_L^{(t+1)} \leftarrow s_L \mathbf{W}_L^{(t)}, \text{ přičemž doporučená hodnota } s_L = 2\\ &\mathbf{W}_{H,i}^{(t+1)} \leftarrow W_{H,i}^{(t)} \sqrt{\frac{A_i^{(0)}}{A_i^{(t)}}}, \text{ kde } A_i \text{ představuje plochu one-ring kolem vrcholu } i\end{aligned} \tag{2.9}$$

Průběh kontrakce sítě je znázorněn na obrázku 2.22, pro názornost byly vynechány dílčí iterace.

Redukce geometrie

Smrštěná síť si zachovává svoji původní konektivitu, k dosažení jednorozměrné kostry použijeme redukci geometrie pomocí *half-edge collapse*. Tento postup lze popsat tak, že vybereme nejméně významnou hranu (i, j), kterou ze sítě odstraníme (viz obrázek 2.10). Vrchol $\mathbf{v}_i$ je poté ztotožněn s vrcholem $\mathbf{v}_j$ a dojde k odstranění stěn incidujících s hranou (i, j). Hrany odebíráme opakovaně do dosažení zastavovací podmínky.



Obrázek 2.10: Half-edge collapse. Vrchol $\mathbf{v}_i$ je ztotožněn s $\mathbf{v}_j$, hrana $\mathbf{e}$ je odstraněna spolu s incidujícími trojúhelníky. Slučování hran je popsáno v obrázku pomocí číslování v obrázku.

Ohodnocení hrany stanovíme jako vážený součet dílčích funkcí:

$$F(i,j) \;=\; w_a F_a(i,j) + w_b F_b(i,j) \quad \text{doporučeno: } w_a = 1,\; w_b = 0.1 \tag{2.11}$$

Funkce $F_a(i,j)$ vyjadřuje míru chyby vzniklé odstraněním hrany (i,j):

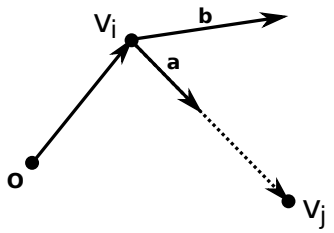
$$F_a(i,j) \;=\; F(\mathbf{v}_j, i) + F(\mathbf{v}_j, j) \tag{2.12}$$

přičemž:

$$F(\mathbf{v}_j, i) \;=\; \mathbf{v}_j^T \sum_{(i,j)\in E} (\mathbf{K}_{ij}^T \mathbf{K}_{ij})\mathbf{v}_j \tag{2.13}$$

$$\mathbf{K}_{ij} \;=\; \begin{bmatrix} 0 & -a_z & a_y & -b_x \\ a_z & 0 & -a_x & -b_y \\ -a_y & a_x & 0 & -b_z \end{bmatrix} \tag{2.14}$$

kde $\mathbf{a}$ je normalizovaným vektorem hrany (i,j) a $\mathbf{b} = \mathbf{a} \times \mathbf{v}_i$ a E představuje množinu hran sítě (viz obrázek 2.15). Předpokládáme, že $\mathbf{v}_i$ neleží v počátku soustavy souřadnic, jinak vyjde vektor $\mathbf{b}$ nulový.



Obrázek 2.15: Znázornění vektorového součinu $\mathbf{b} = \mathbf{a} \times \mathbf{v}_i$, bod $\mathbf{O}$ představuje počátek soustavy souřadnic. Vektor $\mathbf{b}$ je kolmý k $\mathbf{a}$ i $\mathbf{v}_i$.

Funkce $F_b(i,j)$ pokutuje délku hrany (i,j):

$$F_b(i,j) \;=\; \|\mathbf{v}_i - \mathbf{v}_j\| \sum_{(i,k)\in E} \|\mathbf{v}_i - \mathbf{v}_k\| \tag{2.16}$$

Během redukce udržujeme vazbu redukované geometrie na původní objekt. Pro počáteční stav je triviální jej nastavit, v každém vrcholu $\mathbf{v}_i$ vytvoříme seznam obsahující právě tentýž vrchol $(\mathbf{v}_i)$. Při redukci hrany $\mathbf{v}_i\mathbf{v}_j$ seznamy vrcholů sloučíme a uložíme do $\mathbf{v}_j$. Takto postupně tvoříme mapování, které přiřazuje vrcholu kostry množinu příslušných vrcholů původní geometrie.

Závěrečná úprava pozic

Z předchozího kroku jsme získali podmnožinu vrcholů reprezentující kostru objektu, jejíž pozice však musíme dodatečně upravit. Vrcholy kostry se totiž nemusejí nutně nacházet uvnitř modelu. Proto dochází k záverečné úpravě, jež přesune vrcholy kostry směrem doprostřed objemu tvořeného sítí. Využijeme mapování kostry na síť, které vzniklo v předchozím kroku.

K vrcholu kostry $\mathbf{u}$ s pomocí mapování získáme množinu příslušných vrcholů $\mathbf{v}_i$, která je ohraničena N hranicemi ξ_k. Korekce pozice vrcholu kostry $\mathbf{u}$ je dána jako:

$$\mathbf{u} = \mathbf{u} - \sum_{k}^{N} \frac{\mathbf{d}_k}{N} \tag{2.17}$$

$$\mathbf{d}_k = \frac{\sum_{i \in \xi_k} l_{k,i}(\mathbf{v}'_i - \mathbf{v}_i)}{\sum_{i \in \xi_k} l_{k,i}} \tag{2.18}$$

kde $\mathbf{d}_k$ představuje dílčí příspěvek daný vrcholy hranice ξ_k, $\mathbf{v}'_i$ pozici vrcholu po smrštění sítě a $l_{k,i}$ součet délek hran incidujících s vrcholem $\mathbf{v}_i$, které musí zároveň patřit do hranice ξ_k.

Autoři přiznávají, že ne pro každou síť závěrečná úprava pozic skutečně přemístí vrcholy kostry dovnitř modelu. V článku nezmiňují, o kolik je zvolený postup lepší než použití obyčejného centroidu.

Klady a zápory metody

Na metodě oceníme zejména robustnost vůči šumu, neboť během kontrakce sítě pomocí diskrétního laplaciánu dochází k jejímu vyhlazování. Tvorba kostry je netečná vůči rotaci objektu (jako jeden z důvodů uveďme, že pracujeme s původní sítí, ne s diskretizovanou reprezentací v podobě voxelové mřížky).

Citelnou nevýhodou je výpočetní náročnost, ostatně při výpočtu opakovaně řešíme rozsáhlou přeurčenou soustavu rovnic. Právě tento krok způsobuje, že výsledná výpočetní algoritmická složitost je $O(I_C\, N^3)$, kde N značí počet vrcholů a I_C počet iterací. Nastavitelné parametry se musí volit ručně dle sítě, která se nachází na vstupu – článek nepopisuje vhodnou automatickou volbu, doporučuje „osvědčené“ hodnoty.

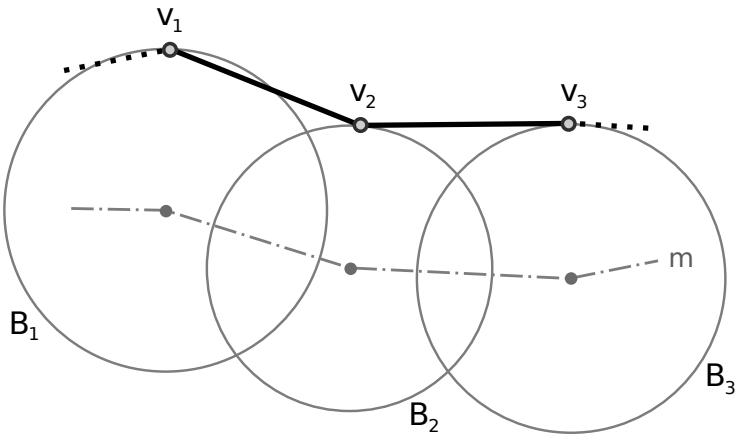
2.2.2 Tvorba kostry spojováním významných bodů

Článek od Jaehwan Ma a Sunghee Choi [11] se zaobírá tvorbou kostry přímo pro animační účely, při generování vznikají záměrně dlouhé kosti, nikoliv sada drobných (na rozdíl od metody od Au et al. popsané v kapitole), které bychom museli dodatečně na závěr spojovat.

Shrňme si nejdříve princip v několika větách. Nejprve jsou nalezeny význačné vrcholy sítě, říkejme jim kandidáti na listy kostry. Mezi nimi se vyskytují i takové body, které vznikly kvůli šumu na povrchu sítě, proto dochází k filtrování kandidátů. Poté započne generování kostí směrem od listů k prozatím neznámemu kořenu a jejich spojování v kostru. Následují závěrečné úpravy: přidání dodatečných vnitřních bodů kostry a umístění kořene kostry.

Předzpracování dat

Autoři použili svoji vlastní metodu hledání aproximace střední osy objektu (Ma et. al, 2012 [10]) k nalezení maximální tečných koulí *(tangent balls)* vůči trojúhelníkové síti ve všech jeho vrcholech (viz obrázek 2.19). Velikost tečné koule příslušící vybranému vrcholu je použita ve výpočtu později.



Obrázek 2.19: Ukázka tečných koulí B_i na části sítě (vrcholech $\mathbf{v}_i$), m značí odhad střední osy. Střed koule leží na v příslušném vrcholu střední osy (který odpovídá pólu Voronoi diagramu), poloměr je volen tak, aby vrchol $\mathbf{v}_i$ ležel na jejím povrchu. Pro názornost znázorněno ve dvourozměrné analogii.

Hledání význačných vrcholů

Význačné vrcholy jsou chápány jako lokálně nejvzdálenější vrcholy vzhledem k ostatním v síti. Intiutivně bychom je mohli popsat jako „zakončení modelu“. Pohybovat se můžeme pouze po hranách sítě, tím pádem se jedná o grafovou úlohu.

Nejprve zvolíme libovolný vrchol sítě v_r, ze kterého provedeme část známého Dijkstrova algoritmu pro hledání cesty (algoritmická složitost při použití binární haldy a předpokladu rovinného grafu, ve kterém počet hran lineárně závisí na počtu vrcholů, je $O(Nlog(N))$, kde N značí počet vrcholů), tj. vzdálenostní ohodnocení vrcholů. Nejvýše ohodnocený v_0 se stává prvním kandidátem na význačný vrchol. Jako další krok opět spustíme Dijsktrův algoritmus, ale tentokrát začneme ve vrcholu v_0. Kandidáty se stávají všechny vrcholy, pro které platí, že jejich vzdálenostní ohodnocení je vyšší než ohodnocení jejich libovolného souseda.

Získané body se mohou nacházet mezi kandidáty například kvůli šumu na povrchu sítě. Vrchol odstraníme ze seznamu kandidátů v případě, že šumem způsobený výčnělek sítě, kterému vrchol náleží, je mělčí než zvolený práh. Pokud se v jednom výčnělku nachází více kandidátů, akceptujeme z nich jen takový vrchol, kterému přísluší nejmenší tečná koule.

Tvorba kostry

Kostru tvoříme opakováním dvou kroků: odřezáváním a slučováním větví vznikající kostry (viz níže). Dílčí kroky si ve stručnosti představíme.

Od všech význačných vrcholů v_i začneme pseudoparalelně tvořit větve grafu kostry. To provedeme tak, že z každého vrcholu v_j spustíme Dijkstrův algoritmus, čímž vypočteme ostatním vrcholům sítě vzdálenost od v_j. Poté uniformně navzorkujeme izočáry, které představují množinu bodů stejně vzdálených od v_i. V metodě byl použit odhad, kdy místo přesných izočar byly použity jejich odhady (vzorkování bylo provedeno na hranách sítě a body stejné vzdálenosti byly propojeny do po částech lineární uzavřené křivky). Izočáry c oindexujeme podle vzdálenosti od význačného vrcholu a označíme je c_j. Pokud platí, že $|c_{j+1}| > |c_j|$, kde $|c_j|$ značí počet samostatných izočar stejné úrovně, pak došlo k topologickému rozdělení. V těchto místech se navíc velmi rychle mění charakteristika izočáry (střední průměr a rozptyl).

Právě zde vytvoříme nový vrchol, který propojíme s význačným. Příslušnou větev kostry tímto považujeme za dokončenou. Může dojít ke sloučení s jinou větví, je-li nalezena izočára s podobnou charakteristikou (pro tento účel je vytvořen nový vrchol). V opačném případě výpočet pokračuje v generování izočár jiné větve modelu. Za zmínku stojí, že se v průběhu algoritmu upřednostňují „tenké“ větve modelu (mající nejmenší střední průměr izočar) a že velmi důležitou roli v algoritmu hraje hustota vzorkování izočar (příliš velká klade vyšší výpočetní nároky, příliš nízká znemožňuje správný chod algoritmu).

Závěrečné úpravy

Každá větev sítě obsahuje nyní pouze jednu kost, která se pne od význačného vrcholu ke spoji s jinými kostmi (sourozenci a nadřazený předek). Do kostry vkládáme vrcholy střední osy objektu mající od ní největší odchylku. Tento proces opakujeme, dokud vkládání způsobuje změny vyšší, než byl stanoven práh.

Posledním krokem je určení pozice kořene kostry. Veškeré dílčí větve jsou v tento moment odřezány, zbyla nám množina vrcholů (v článku označené jako *root region*), z jejichž pozic vypočteme vážený centroid, přičemž jako váhy použijeme poloměry příslušných tečných koulí.

Klady a zápory metody

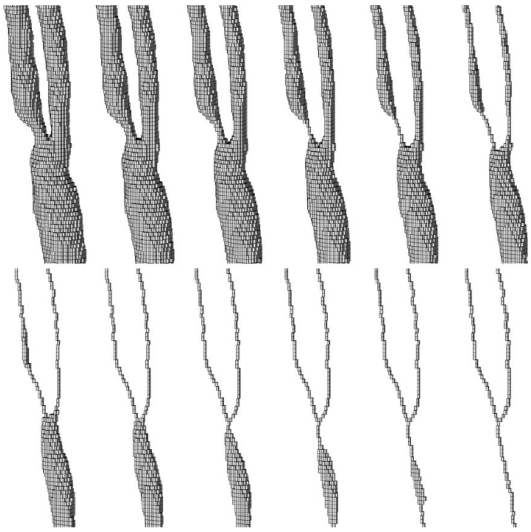
Algoritmus produkuje kostru, která je dostatečně jednoduchá, abychom ji mohli použít k animaci objektu. Stejně jako u výše popsané metody od Au et al. (viz sekce 2.2.1) je tvorba netečná vůči rotaci objektu či jeho póze. Vzhledem k tomu, jak je tvořena hierarchie kostí (od listů směrem ke kořeni), má výsledná kostra „hvězdicovitou“ podobu, kosti se sbíhají do kořene. Sami autoři článku přiznávají, že to může bránit praktickému využití algoritmu pro generování koster humanoidů.

Podle autorů největší výpočetní zátěž tvoří právě stavba kostry (nikoliv prvotní hledání význačných bodů). Výpočetní složitost algoritmu závisí nejen na počtu vrcholů, ale i na použité hustotě vzorkování a topologické složitosti modelu (kde dochází k více slučování dílčích větví). Vyčíslená složitost $O(q\,c(i)\,N\,logE)$ odpovídá procházení rovinného grafu, kdy v každém kroku i srovnáváme měnící se počet izočar c, N značí počet vrcholů, E počet hran, q hustotu vzorkování.

Za předpokladu, že délky hran jsou srovnatelné, můžeme zvolit parametr q tak, abychom navzorkovali izočárou každou hranu přibližně jednou. Potom můžeme výpočetní složitost vyjádřit jako $O(E\,N\,logE)$. Hrany povrchu sítě však tvoří rovinný graf, jejich počet je tedy lineárně závislý na počtu vrcholů N, což nám umožní závěrečné zjednodušení složitosti: $O(N^2\,logN)$. Nezapomeňme však, že k tomuto předpisu jsme dospěli po zmíněném zjednodušujícím předpokladu.

2.3 Volumetrické metody

Před použitím volumetrických metod dochází k diskretizaci objektu: nahradíme původní objem voxely (obecným rozšířením pixelu pro prostor), které představují plný prostor. Vzniklý diskretizovaný objekt ztenčujeme odebíráním hraničních voxelů. Pořadí jejich odebírání určuje prioritní funkce, která se v různých metodách liší. Stejně tak se rozchází definice hraničních voxelů ku příkladu z důvodu použití rozdílné sousednosti (6-okolí apod.). Jako příklad ztenčování uveďme metodu od Ma et al. [9]. Nevýhodou metod založených na ztenčování je nemožnost práce se surovými vstupními daty, nutnost volby hustoty mřížky a vysoká citlivost na šum, z vytvořené kostry musíme odstraňovat bezvýznamné větve.



Obrázek 2.20: Ukázka iterací ztenčování modelu reprezentovaného voxely. Zdroj: `http://csvision.swan.ac.uk/index.php?n=Site.BloodFlowModelling`, cit. 6. 2015.

Obdobně nerobustní chování (podle Au et al. [4]) mají i metody založené na distančním poli (*distance field methods*), u kterých každému voxelu přiřadíme hodnotu

odpovídající vzdálenosti od hranice objektu. Lokální maxima reprezentují kandidáty, jejichž spojením získáme aproximaci střední osy objektu.

Při porovnání s geometrickými metodami, které byly rozebrány výše, trpí volumetrické metody několika nepřehlédnutelnými problémy:

1. Geometrická data vyžadují převod do voxelové reprezentace, nejsou-li již na vstupu v této podobě.
2. Při konverzi musíme vhodně zvolit hustotu voxelové mřížky.
3. Voxelová reprezentace je zpravidla paměťově náročnější oproti geometrické.

2.3.1 Generování kostry snímaného člověka v reálném čase

Straka et. al [14] snímali člověka několika kamerami a v reálném čase byli schopni vypočítat jeho kostru. Snímáním získali přibližnou volumetrickou reprezentaci osoby, ze které metodou *volume scooping* [13] vypočítali přibližnou podobu střední osy objektu. Na první pohled se může zdát, že jsme se ocitli v oblasti strojového vidění, principy extrakce kostry však zůstávájí stejné a můžeme je využít pro data libovolného původu.

Střední osu není možné použít přímo jako kostru objektu, neboť je zatížena nepřesnostmi – oproti požadovanému stavu obsahuje nadbytečné kosti. Algoritmu proto dodáme tzv. vzorovou kostru S, podle níž se dočasně vypočtená kostra upraví. Vzor velmi zjednodušeně reprezentuje člověka, například celá paže včetně předloktí je reprezentována pouhými dvěma kostmi se společným kloubem v lokti.

Vzorová kostra S i dočasně vypočítaná kostra X jsou obě vlastně grafem, pro který určíme distanční matici, která poslouží v následujících výpočtech jako metrika pro výpočet chyby přiřazení dvojic vrcholů z X a S. Následně se zaměříme na koncové vrcholy $E(X)$ a $E(S)$ (listy grafu), přičemž chceme nalézt přiřazení:

$$f(E(S)) \rightarrow f(E(X)) \tag{2.21}$$

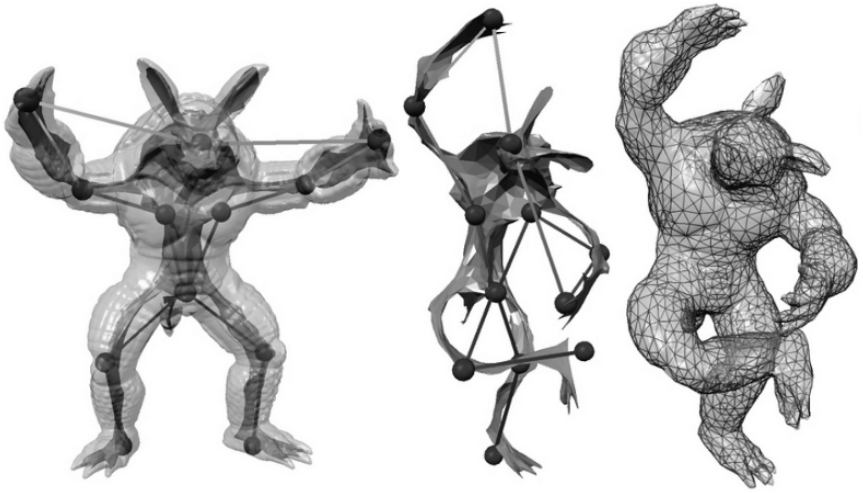
takové, že výsledná chyba je minimální. Autoři jej hledají kombinací *dynamic time*

warping (*DTW*, použita kvůli možnosti rozdílného počtu koncových vrcholů) a *mad'arské metody* (přiřazení dvojic s nejmenší globální chybou). Obdobným způsobem jsou spárovány vnitřní klouby. Díky tomu můžeme vzorovou kostru vložit do skenovaného modelu a provést už jen závěrečné úpravy pozic (viz původní článek).

Nesporná výhoda, která se v metodě ukrývá, je znemožnění tvorby nesmyslné topologie kostry (přídavné končetiny apod.). Algoritmus může selhat a přiřadit k sobě špatné dvojice kloubů z koster S a X, požadovaná topologie je však vždy zachována. Ačkoliv autoři svoji metodu použili pro volumetrická vstupní data, nic nebrání nasazení vstupu v podobě trojúhelníkové sítě, tj. nahradit krok tvorby dočasné kostry (odhadu střední osy).

2.4 Příbuzné oblasti

Metoda od Yoshizawa et al. [16] se zabývá generováním kostry v poněkud jiné podobě než doposud zmíněné přístupy. Kostru představuje odhad střední osy sítě vypočítaný pomocí pólů Voroného diagramu, které jsou spojeny topologicky stejně jako původní trojúhelníková síť [3]. Původní síť byla v podstatě zdeformována do podoby střední osy, k žádné redukci na jednorozměrný graf zde nedochází. Pro potlačení šumu doporučují autoři vstupní model nejprve zjednodušit.



Obrázek 2.22: Ukázka výsledků metody Yoshizawa et al. Uvnitř modelu se nachází vygenerovaný skelet, který přenáší deformaci ručně vytvořené jednorozměrné kostry na povrch tělesa.

Po aplikaci libovolných *free-form* deformací na vzniklou síť střední osy můžeme zrekonstruovat původní objekt se zachováním lokální tloušťky, což vyplývá z vlastností *medial axis transform*. Byla-li původní síť před transformací zjednodušena, dochází k opětovnému zesložitění s užitím diskétních diferenciálních souřadnic.

Skelet, kterým se řídí tvarování objektu, nevyužijeme přímo, neboť je složen ze stejného počtu vrcholů, který měla původní síť, jen se nacházející blízko střední osy. Nedosáhli bychom tak žádného zjednodušení. Autoři proto používají ručně vytvořenou (jednorozměrnou) kostru, kterou řídí deformaci generovaného skeletu.

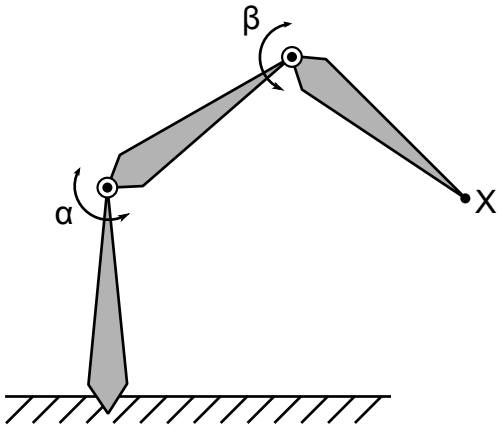
3 Přímá a inverzní kinematika

V předchozí kapitole jsme popsali metody, které slouží ke generování kostry. Obohatíme-li vzniklou kostru o vhodná omezení pohybu v kloubech (viz níže), můžeme ji rozhýbat a animovat původní model. Jeden ze způsobů animace je využití tzv. inverzní kinematiky.

Kostra objektů představuje otevřenou hierarchickou segmentovou strukturu, ve které bod položený nejvýše v hierarchii bývá pevně ukotven. Volné listy v hierarchii nazveme koncové efektory.

Polohu volného tělesa v prostoru lze určit pomocí zvoleného souřadnicového systému a šesti čísel, takzvanými stupni volnosti. Uvažujeme-li systém těles, počet stupňů volnosti roste s jejich přidaným počtem. Takový systém může ovšem obsahovat různá omezení pohybu (vazby mezi objekty, klouby), která naopak sníží výsledný počet stupňů volnosti. Na obrázku 3.1 je znázorněn příklad systému se dvěma stupni volnosti, jehož celkový stav lze popsat vektorem (α, β). Délka vektoru odpovídá počtu stupňů volnosti soustavy, přičemž poloha koncového efektoru je určena stavovým vektorem jednoznačně.

Inverzní kinematika se zabývá určením stavového vektoru na základě zadané pozice koncového efektoru a sady omezení pohybu. O základech této problematiky pojednává např. Žára et al. [5], podrobněji Goliáš ve své diplomové práci [8].



Obrázek 3.1: Příklad segmentové struktury se dvěma stupni volnosti. Koncový efektor je označen písmenem X.

3.1 Přímá kinematika

Přímá kinematika slouží k výpočtu pozice koncového efektoru na základě zadaného vstupního stavového vektoru θ. Vztah lze vyjádřit rovností:

$$X = f(\theta) \tag{3.2}$$

kde X značí pozici koncového efektoru a f použité zobrazení. Přímá kinematika se nehodí k interaktivním animacím, nasazuje se v případech, kdy známe časový průběh stavového vektoru θ. Jako příklady využití přímé kinematiky, které souvisejí s animační kostrou, uveďme:

- *Motion capture*, tj. zaznamenání pohybu reálného tělesa v místech blízkých jednotlivým kostem a následné přepočítání stavového vektoru θ.
- *Mesh-skinning*, tj. potažení kostry trojúhelníkovou sítí, kdy na základě stavového vektoru θ určujeme výsledné pozice povrchu.

3.2 Inverzní kinematika

S pomocí inverzní kinematiky, jak již bylo výše řečeno, se snažíme o určení stavového vektoru θ na základě zadané pozice X koncového efektoru. Formálně bychom tuto představu mohli zapsat vztahem:

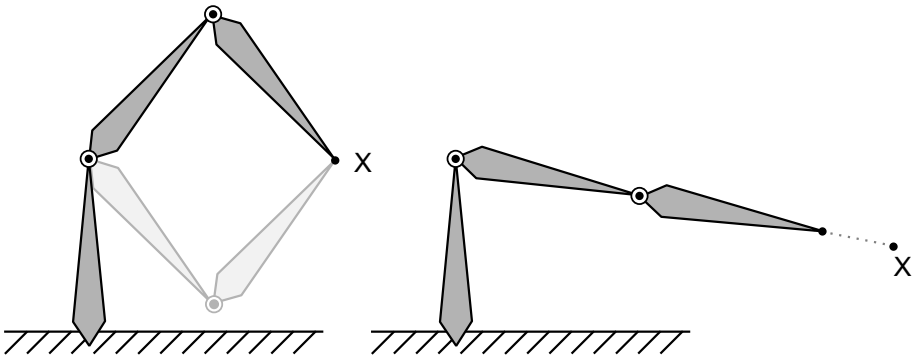
$$\theta = f^{-1}(X) \tag{3.3}$$

Na rozdíl od přímé kinematiky však nemusí existovat jednoznačné řešení, v některých případech dokonce žádné řešení neexistuje (viz obrázek 3.5). Právě z tohoto důvodu se zavádí omezení volnosti pohybu v kloubech, abychom náležitě zjednodušili podobu stavového prostoru. I tak se nevyhneme singularitám, navíc zobrazení f není lineární a analytické řešení rovnosti prakticky nelze realizovat.

3.2.1 Linearizace pomocí jakobiánu

Problém tedy řešíme v linearizované podobě za pomoci jakobiánu. V takovém případě můžeme předpokládat, že v dostatečně malém okolí bodu X platí:

$$\Delta X \;=\; J_f(\theta)\Delta\theta \;=\; \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_n} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \cdots & \frac{\partial f_2}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \theta_1} & \frac{\partial f_m}{\partial \theta_2} & \cdots & \frac{\partial f_m}{\partial \theta_n} \end{bmatrix} \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_n \end{bmatrix} \tag{3.4}$$



Obrázek 3.5: Singulární případy. Vlevo dvě řešení, vpravo úloha neřešitelná.

Vektor ΔX je nám známý – jedná se o rozdíl požadované a aktuální pozice. Výpočet změny $\Delta\theta$ lze formálně zapsat:

$$\Delta\theta \;=\; J_f(\theta)^{-1}\Delta X \tag{3.6}$$

Matice J_f zpravidla nebude čtvercová, klasická inverze tudíž ani nemůže být použita – přikročíme k tzv. pseudoinverzi obdélníkové matice, kterou můžeme realizovat například pomocí metody SVD (singular value decomposition) nebo vyjádřením (podle Mereditha a Maddocka [12])

$$J_f(\theta)^{-1} = J_f(\theta)^T(J_f(\theta)J_f(\theta)^T)^{-1}. \tag{3.7}$$

Místo pseudoinverze lze užít také transpozici jakobiánu, v takovém případě

realizujeme vlastně numerické řešení vztahu metodou největšího spádu. Transpozice klade nižší nároky na výkon v krocích iterace, na druhou stranu výpočet ztrácí na numerické stabilitě.

Výsledný algoritmus tak či onak pracuje iterativně, dokud není dosaženo kýžené pozice koncového efektoru:

1. Vypočti jakobián $J_f(\theta)$.
2. Vypočti pseudoinverzi jakobiánu $J_f(\theta)^{-1}$.
3. Vypočti nové ΔX jako rozdíl cílové a aktuální pozice koncového efektoru.
4. Vypočti změnu vnitřního stavu $\Delta\theta = J_f(\theta)^{-1}\Delta X$.
5. Přiřaď pro novou iteraci vnitřní stav $\theta_{+1} = \theta + \alpha\,\Delta\theta$, kde α představuje řídicí konstantu rychlosti konvergence.

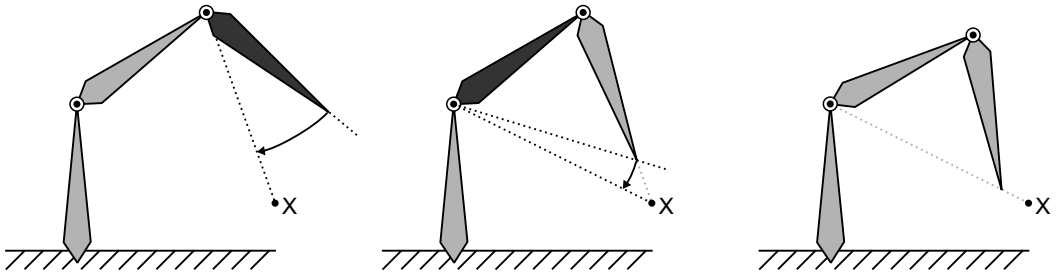
3.2.2 Cyclic coordinate descent

Minimalizaci chyby, tj. vzdálenosti koncového efektoru od jeho požadované pozice, lze realizovat také metodou cyclic coordinate descent (CCD). Tento způsob řešení úlohy inverzní kinematiky navrhli Wang a Chen [15].

Algoritmus stejně jako předchozí metoda pracuje iterativně, v každé iteraci prochází hierarchii segmentů od koncového efektoru směrem ke kořenu. Každému segmentu přísluší kloub, který ji propojuje s nadřazeným segmentem. U kloubu provedeme změnu hodnot popisující jeho konfiguraci, která minimalizuje vzdálenost koncového efektoru od zadané pozice (viz obrázek 3.8). Lokální úprava poté končí a přesuneme se k nadřazenému segmentu. V aplikacích dochází k ukončení výpočtu po dosažení prahové hodnoty chyby nebo po stanoveném počtu iterací, řešení totiž nemusí existovat a uživatel by marně čekal na odezvu.

Metoda se vyznačuje poměrně rychlou konvergencí v řádu stovek iterací, avšak hrozí uváznutí v lokálním optimu. I přes zavedení omezení pohybu v kloubech může metoda vygenerovat na rozdíl od výpočtu pomocí jakobiánu velmi nepřirozený pohyb. Toto chování je možné redukovat, pokud nedovolíme změnu hodnot představující konfiguraci kloubu větší než stanovený práh (v takovém případě upravujeme konfiguraci

o prahovou hodnotu), což na druhou stranu může zpomalit konvergenci algoritmu. Oproti výpočtu pomocí jakobiánu se CCD vyznačuje jednodušší implementací.



Obrázek 3.8: Ukázka jedné iterace metody cyclic coordinate descent.

3.2.3 Obohacení o posuvná spojení

Posuvná spojení (také posuvné klouby, anglicky nazývané *prismatic joints*) mimo jiné umožní realizovat *pružné segmenty* kostí. Tímto obohatíme segmentovou strukturu o další stupeň volnosti. Uvažme však, že po formální stránce zůstává celý výpočet stejný, můžeme použít výpočty popsané výše, pokud zaneseme posuvná spojení do stavového vektoru θ náležícího segmentové struktuře. To je možné například použitím DH-notace (Denavit–Hartenbergovy parametry [7], popsáno např. v [5], podkapitola 18.2.2).

4 Návrh vlastní metody

Jak již bylo řečeno v úvodu práce, zaměřili jsme se na generování koster humanoidů pro účely animace. Díky tomu lze vydedukovat některé požadavky na vlastnosti kostry.

- Kostra topologicky odpovídá stavbě humanoida, vstupuje do končetin a hlavy. Předpokládáme humanoida se dvěma nohama i pažemi, v místě hrudního koše je zachycena pouze páteř.
- Počet kostí není zbytečně vysoký. To by výtvarníkovi znesnadňovalo rozanimování modelu, vyžadovalo ukládání většího množství dat a kladlo vyšší nároky na výkon při přehrávání animace.
- Kostra leží alespoň přibližně uvnitř vstupní sítě.
- Kostra obsahuje omezení na volnost pohybu v kloubech.

4.1 Generování kostry

Žádná z metod prozkoumaných v kapitole 2 přímo neodpovídá naší situaci. Vygenerovat rozumná omezení na volnost pohybu v kloubech čistě geometrickou interpretací kostry a sítě je velmi obtížný, ne-li nemožný úkol. Jako vhodnější se jeví využít znalosti, že vstupní síť reprezentuje humanoida, a připravit vzor obsahující přibližnou podobu kostry včetně omezení v kloubech. S přihlédnutím k seznamu požadavků a postřehů navrhněme koncept řešení, který kombinuje vybrané metody z kapitoly 2:

1. Připravíme si referenční kostru humanoida, která bude obsahovat i omezení na volnost v kloubech.
2. Vygenerujeme kostru smrštěním trojúhelníkové sítě s následnou redukcí na 1D graf (Au et al., viz podkapitola 2.2.1). Vzniklá prozatímní kostra je však příliš detailní.

3. Užitím *dynamic time warping* a *mad'arské metody* spárujeme vrcholy (klouby) referenční a prozatímní kostry. Jedná se o část metody od autorů Straky et al. popsané v podkapitole 2.3.1.

4. Referenční kostru vložíme dovnitř sítě. Pozice kloubů kostry je ovlivněna „partnerskými“ klouby prozatímní kostry a například poměry délek incidujících kostí.

5. Do vzniklé kostry zkopírujeme omezení volnosti pohybu v kloubech obsažené v referenční kostře. Předpokládáme, že vstupní trojúhelníková síť představuje humanoida v klidové póze (rozpažené ruce, plochy dlaní směřující vpřed, spatný stoj).

4.2 Interaktivní ukázka

Vlastnosti vzniklé kostry se projeví až při jejím vlastním užití, tedy při uvedení do pohybu. Z tohoto důvodu na výše zmíněné generování, které bychom mohli označit jako předzpracování (preprocessing), naváže interaktivní vizualizace kostry.

V pozicích koncových efektorů budou umístěny kontrolní body, které uživatel bude moci přetáhnout na jiné místo. V tu chvíli dojde k aplikaci inverzní kinematiky, která se vynasnaží přiblížit odpovídající koncový efektor kýžené pozici. Z experimentálních důvodů budou naimplementovány dvě verze algoritmu:

1. Inverzní kinematika s pevnými kostmi neměnné délky.

2. Inverzní kinematika s pružnými kostmi, které se mohou libovolně natahovat.

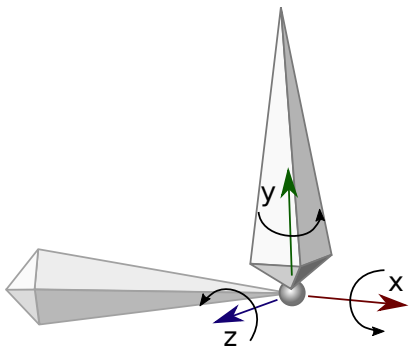
Obě varianty budou respektovat omezení volnosti pohybu v kloubech, která byla zkopírována z referenční kostry. Při změně konfigurace kostry dojde k odpovídající deformaci modelu humanoida.

5 Podrobný rozbor metody

V kapitole 4 byl navržen postup generování kostry a následná demonstrace výsledku pomocí interaktivní vizualizace. Nyní si rozebereme dílčí kroky podrobněji. Popíšeme si algoritmy, jejich parametry a charakter dat, se kterými se na dané úrovni pracuje.

5.1 Vstupní data

Model humanoida, který načteme na počátku celého procesu, je reprezentován povrchově pomocí trojúhelníkové sítě. Navržená metoda tvorby kostry požaduje, aby síť byla manifoldní (blízké okolí libovolného bodu náležícího síti je homeomorfní vůči euklidovskému prostoru E^2). Díky tomu například můžeme užít jednodušší datové struktury či korektně spočítat objem uzavíraný sítí. Předpokládáme, že humanoid ve scéně stojí, je rozpažený a dívá se ve směru osy z.

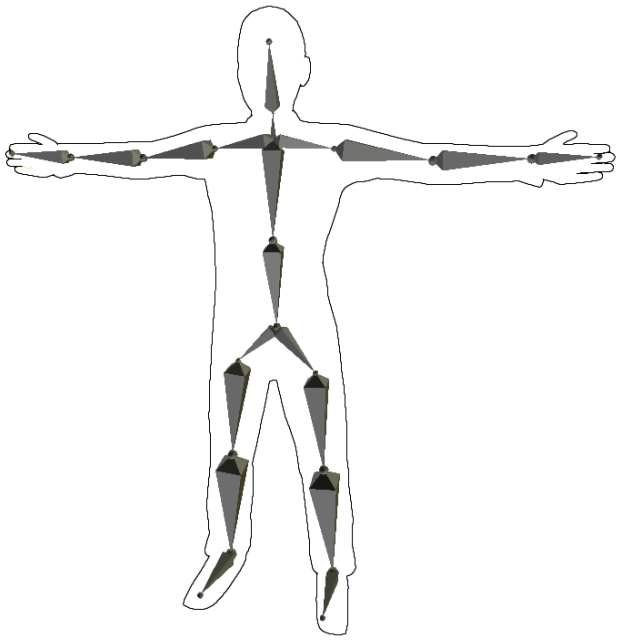


Obrázek 5.1: Znázornění rotací kosti podle os x, y, z. Použitý lokální systém je pravotočivý, směr osy y je ztotožněn se směrem kosti.

Referenční kostra (označme si ji S_R) popisuje pozice jednotlivých kloubů a jejich spojení kostmi. Dá se na ni pohlížet jako na graf umístěný v prostoru, ve kterém klouby představují uzly a kosti hrany. Tohoto pozorování využijeme během generování animační kostry – je možné použít grafové algoritmy. Podle výše uvedených parametrů jsme schopni kosti chápat také jako úsečky v prostoru. Tím je dána orientace kosti pouze v jednom směru, proto ke každé kosti připojíme lokální souřadnicový systém, směr osy y ztotožníme se směrem kosti. Mimo to známe omezení její rotace vůči

klidové póze kolem os lokálního souřadnihového systému (viz obrázek 5.1), což je ekvivalentní s limitací pohybu v nadřazeném kloubu.

Podoba referenční kostry je velmi zjednodušenou podobou kostry člověka (viz obrázek 5.2. V první řadě se oprošťuje od anatomie, neobsahuje např. hrudní koš nebo zdvojené kosti v končetinách – u dolních končetin jsou kost lýtková a holenní sloučeny do jedné, podobně u horních sloučíme kost loketní a vřetenní. Tato podoba kostry pokazí vzhled modelu pouze při extrémních zkrutech či ohybech. V druhé řadě nepostihujeme drobné detaily, jimiž jsou jednotlivé prsty končetin a čelisti, čímž odpadá možnost otevírat ústa.



Obrázek 5.2: Referenční kostra, pro ilustraci znázorněn i obrys člověka.

5.2 Generování prozatímní kostry

Z trojúhelníkové sítě vygenerujeme prozatímní kostru S_G s využitím metody autorů Au et al. podrobně popsané v kapitole 2.2.2, tj. nejprve smršťujeme trojúhelníkovou síť laplaceovským vyhlazováním a následně opakovaně odstraňujeme technikou edge-collapse nadbytečné hrany, než se zbavíme všech trojúhelníků sítě. Vzniklé jednorozměrné struktuře upravíme pozicování v prostoru.

5.3 Vytvoření animační kostry

Připravenou referenční kostru S_R humanoida budeme nyní vkládat dovnitř modelu. Tato část metody vychází z metody popsané v kapitole 2.3.1. Hledáme prosté zobrazení $M : U(S_R) \rightarrow U(S_G)$, které jednoznačně přiřadí uzlům kostry S_R uzly z prozatímní kostry S_G. Pro existenci takového zobrazení musí platit nerovnost:

$$|U(S_R)| \leq |U(S_G)|. \tag{5.3}$$

Výsledná animační kostra je grafovým isomorfismem referenční kostry S_R, jejíž uzly U jsou umístěny na pozice $M(U)$.

5.3.1 Přípravné kroky a přiřazení hlavy

Předpokládáme, že koncový efektor představující hlavu se nachází nejvýše, tj. jeho souřadnice y je maximální (osa y míří vzhůru). Tím určíme pozici hlavy v prozatímní kostře.

V obou grafech (referenční i prozatímní kostře) spočteme vzdálenost mezi všemi dvojicemi vrcholů, což lze provést například oblíbeným Floyd-Warshallovým algoritmem, a následně seřadíme vrcholy podle geodetické vzdálenosti od uzlu představujícího hlavu (např. pomocí quick sort nebo merge sort). Řazení je nezbytné pro správnou funkci dalších kroků.

Určíme nejdelší cestu v obou grafech, její délka bude sloužit jako měřítko velikosti kostry. Poté provedeme přeškálování referenční kostry tak, aby velikostně odpovídala prozatímní. Zvětšení s odpovídá vztahu

$$s = \frac{L_{temp}}{L_{ref}}, \tag{5.4}$$

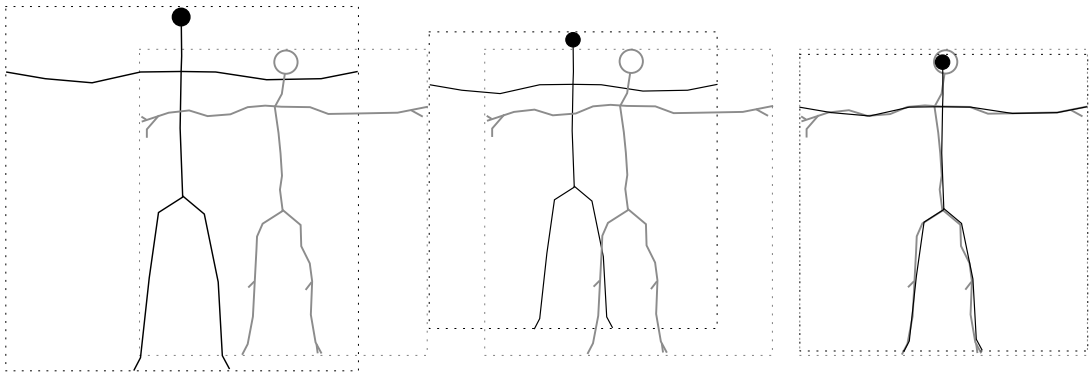
kde L_{temp} představuje délku nejdelší cesty v grafu prozatímní kostry a L_{ref} délku nejdelší cesty v grafu kostry referenční. Vypočtené délky cest mezi dvojicemi vrcholů přenásobíme stejným faktorem.

Jako poslední krok je nutné vyrovnat odlišné umístění koster – referenční kostra se může v souřadnicovém prostoru nacházet na odlišné pozici než kostra prozatímní. U obou koster zjistíme rozměry axis-aligned bounding boxu B_{temp} a B_{ref} (obalové těleso o podobě kvádru orientované souhlasně s osami). Posun referenční kostry $\mathbf{T}(x, y, z)$ je pak určen jako:

$$\mathbf{T}(x, y, z) = \begin{bmatrix} B_{temp}^{maxx} - B_{temp}^{minx} - B_{ref}^{maxx} + B_{ref}^{minx} \\ B_{temp}^{maxy} - B_{temp}^{miny} - B_{ref}^{maxy} + B_{ref}^{miny} \\ B_{temp}^{maxz} - B_{temp}^{minz} - B_{ref}^{maxz} + B_{ref}^{minz} \end{bmatrix}. \tag{5.5}$$

Proces zarovnání koster je znázorněn na obrázku 5.6.

Výpočetní složitost předzpracování kostry je dána jejím nejnáročnějším krokem, kterým je výpočet vzdálenosti mezi všemi dvojicemi uzlů. Floyd-Warshallův algoritmus, který dokáže problém efektivně řešit, nabývá složitosti $O(N^3 + M^3)$, kde N představuje počet vrcholů referenční kostry a M počet vrcholů kostry prozatímní.



Obrázek 5.6: Znázornění sjednocení velikosti a pozice referenční a prozatímní kostry. Referenční kostra má vyznačenou hlavu černě, prozatímní bíle. Tečkovaně jsou vyznačeny obalující boxy.

5.3.2 Přiřazení listů

V této části se zabýváme přiřazením listových uzlů. Hlava je v tuto chvíli už přiřazena, a proto ji z tohoto procesu vyjmeme. Z listových vrcholů utvoříme seznamy, formálně si je označíme: pro referenční kostru $L_{ref} = (r_1, r_2, \ldots, r_n)$, pro prozatímní

kostru pak $L_{temp} = (t_1, t_2, \ldots, t_m)$. Obecně nelze říci, že by byl počet listů stejný, naopak předpokládáme, že $m \geq n$. Je nutné poznamenat, že respektujeme pořadí vrcholů vzniklé řazením podle vzdálenosti od hlavy.

Listy obou grafů k sobě přiřadíme užitím maďarské metody [2], která minimalizuje globální cenu (penalizaci) přiřazení. Pro vyčíslení ceny $E_{i,j}$ přiřazení dvojice (t_i, r_j) použijeme vztah:

$$E_{i,j} \;=\; c_{DWT} \cdot DWT(d_{t_i}, d_{r_j})^2 + c_{dist} \cdot dist(t_i, r_j)^2, \tag{5.7}$$

Předpis ceny je váženým součtem dvou členů, neprve si objasníme výraz funkce *dist*. Ta vrací euklidovskou vzdálenost mezi předanými vrcholy, c_{dist} představuje násobicí konstantu (váhu). Funkce DWT pak spočítá hodnotu *dynamic time warping* dvou sekvencí d_{t_i} a d_{r_j}. V nich jsou obsaženy geodetické vzdálenosti listů $(t_1, t_2, \ldots, t_m)$ a $(r_1, r_2, \ldots, r_n)$ od vrcholu t_i, respektive r_j (vzdálenosti už byly vypočteny během přípravného kroku Floyd-Warshallovým algoritmem). Konstanta c_{DWT} je váhou funkce DWT.

Výpočetní složitost přiřazení listů se skládá z několika částí. Výpočet hodnoty DWT pro dvě sekvence, z nichž má první délku N a druhá M, je obecně složitosti $O(M\,N)$. Tento výpočet však provádíme pro všechny prvky matice velikosti $M \times N$, tudíž složitost samotné přípravy matice pro maďarskou metodu bude $O(M^2\,N^2)$. Následuje výpočet optimálního přiřazení (maďarskou metodou) o složitosti $O(M\,N^2)$.

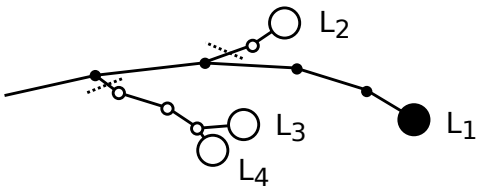
Jak již bylo řečeno výše, předpokládáme, že počet listů prozatímní kostry je větší nebo roven počtu listů referenční kostry, tj. $M \geq N$. Některé listy prozatímní kostry tak zůstanou nepřiřazeny (viz dále). Není-li dodržena podmínka $M \geq N$, algoritmus přiřazení listů selže, neboť není možné nechat jediný list referenční kostry, kterou hodláme ovládat model, nepřiřazený.

Po dokončení této fáze se zaměříme na vnitřní uzly.

5.3.3 Přiřazení vnitřních uzlů

Před samotným přiřazováním vnitřních uzlů projdeme graf prozatímní kostry od každého přiřazeného listu směrem ke kořeni. Po průchodu mohou zbýt některé vrcholy

nenavštívené, jedná se o vrcholy spadající do větví nepřiřazených listů. Chápeme je jako nežádoucí a z grafu je odstraníme (viz obrázek 5.8).



Obrázek 5.8: Prořezávání větví dočasné kostry. Nepřiřazené listy L_2, L_3, L_4 budou včetně celé větve z dočasné kostry odstraněny.

Vnitřní uzly přiřazujeme pouze na základě optimalizace lokální chyby, tedy bez ohledu na penalizaci ostatních vrcholů – pro definici chyby využijeme hodnoty, které jsou na přiřazení nezávislé (viz níže). Drobnou výjimkou je, že již přiřazený vrchol nesmí být použit opětovně (to lze provést například označením vrcholu). Vnitřní vrcholy náležící dočasné kostře označíme u_i, vnitřní vrcholy náležící referenční kostře pak v_j. Pro každý u_i tedy procházíme všechny nepoužité uzly v_j, přičemž hledáme takové přiřazení, které je nejméně penalizováno.

Pro každý vrchol lze vyjádřit tyto dílčí penalizace:

1. **Rozdíl vzdáleností k listům.** Známe přiřazení listů, v každém grafu taktéž geodetickou vzdálenost k příslušnému listu. Tato vzdálenost se v obou grafech bude nejspíše lišit. Sečtením kvadrátů všech rozdílů dostáváme chybu, kterou označíme E_L.

2. **Rozdíl stupně vrcholů.** Kvadrát rozdílu stupně vrcholů u_i a v_j, označíme jej E_D.

3. **Vzdálenost vrcholů.** Kvadrát euklidovské vzdálenosti vrcholů u_i a v_j, označíme jej E_P.

Výsledná chyba odpovídá jejich váženému součtu:

$$E \;=\; c_{E_L} \cdot E_L + c_{E_D} \cdot E_D + c_{E_P} \cdot E_P. \tag{5.9}$$

Tato hodnota není závislá na přiřazení jiných vnitřních vrcholů, proto můžeme výslednou konfiguraci hledat s pomocí optimalizace lokální chyby. Složitost pomocí

naivní implementace je pak rovna $O(M\,N)$, N znamená počet vnitřních vrcholů vzorové kostry, M počet vrcholů prozatímní.

Momentálně jsme referenční kostru umístili do vstupní trojúhelníkové sítě, čímž jsme vytvořili animační kostru. Kosti jsou v tuto chvíli reprezentovány pouze pomocí úseček (hrany grafu), pro interaktivní demo deformující síť musíme určit orientaci lokálních souřadnicových systémů.

5.3.4 Zkopírování lokálních souřadnicových systémů

Pro lokální souřadnicový systém platí, že osa y je ztotožněná s orientací kosti $\overrightarrow{\mathbf{B}}$. Dále víme, že animační kostra si je s referenční velmi podobná (vzhledem k charakteru vstupních dat). Označme si orientaci os lokálního systému animační kostry $\mathbf{A}_x, \mathbf{A}_y, \mathbf{A}_z$ a u referenční kostry $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$. Pak platí:

$$\mathbf{A}_y = \frac{\overrightarrow{\mathbf{B}}}{\|\overrightarrow{\mathbf{B}}\|} \tag{5.10}$$

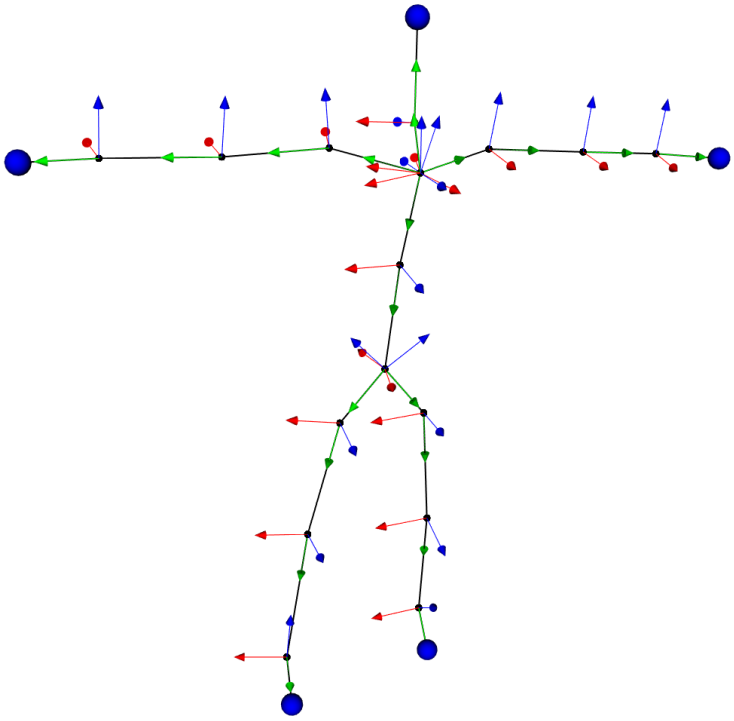
$$\mathbf{A}_z = \mathbf{R}_x \times \mathbf{A}_y \tag{5.11}$$

$$\mathbf{A}_x = \mathbf{A}_y \times \mathbf{A}_z \tag{5.12}$$

Tento krok má viditelně lineární složitost $O(N)$. Příklad možného výsledku se nachází na obrázku 5.13.

5.3.5 Zkopírování limitace pohybu v kloubech

Hodnoty omezení v kloubech jsou do výsledného modelu jednoduše zkopírovány. Předpokládáme, že jsou oba modely v klidové póze a je díky tomu možné použít stejné hodnoty.



Obrázek 5.13: Znázornění lokálních souřadnicových systémů kostí.

5.4 Interaktivní deformace

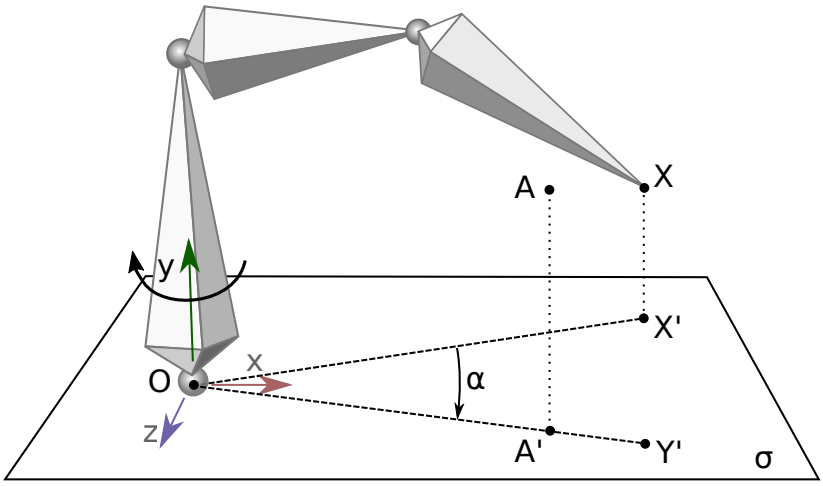
Animační kostra nyní obsahuje veškeré informace nutné pro interaktivní manipulaci. V místech koncových efektorů se objeví prvky, které uživatel může přesouvat v prostoru a zadá tak novou pozici příslušného koncového efektoru, který se k ní přiblíží s využitím inverzní kinematiky.

5.4.1 Inverzní kinematika

Metody řešící úlohu inverzní kinematiky byly popsány v kapitole 3. Pro výpočet v interaktivní ukázce byl zvolen přístup *cyclic coordinate descent* (CCD), která konverguje zpravidla během několika set iterací a umožňuje řešit úlohu přímo v globálním souřadnicovém systému.

Limity rotací v kloubech jsou zadány jako maximální hodnoty rotací kolem lokální osy x, y a z. Krok algoritmu, kdy optimálně natáčíme právě vybranou kost, aby vzdálenost koncového efektoru X od zadané pozice Y byla minimální, rozložíme na

tři individuální rotace podle lokálních os. Překročí-li rotace zadané omezení, je její hodnota upravena oříznutím na povolený interval.



Obrázek 5.14: Znázornění rotace kolem lokální osy y. Bod X představuje pozici koncového efektoru, A nově zadanou pozici, O počátek lokální soustavy souřadnic. Rovina σ znázorňuje rotační rovinu, α hledaný úhel rotace. X' je průmětem koncového efektoru X do roviny rotace, Y' průmětem koncového efektoru po provedení rotace.

Rotace kolem jedné z os je znázorněna na obrázku 5.14. Provedeme ortogonální projekci koncového efektoru X do roviny rotace σ, kterou vyjádříme pomocí bodu O a normálového normalizovaného vektoru $\mathbf{n}$ (souhlasně orientovaným se zvolenou osou rotace):

$$X' \;=\; X - (\overrightarrow{OX} \cdot \mathbf{n})\,\mathbf{n}. \tag{5.15}$$

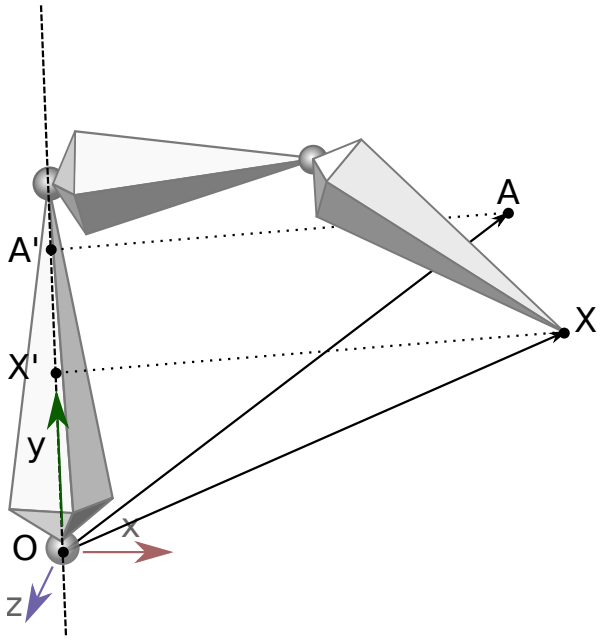
Obdobně promítneme do roviny σ i cílový bod Y. Takto jsme získali vektory $\overrightarrow{OX'}$ a $\overrightarrow{OY'}$, které svírají hledaný úhel ideální rotace α, jehož velikost vypočteme podle předpisu:

$$\alpha \;=\; arccos\left(\frac{\overrightarrow{OX'}}{\|\overrightarrow{OX'}\|} \cdot \frac{\overrightarrow{OY'}}{\|\overrightarrow{OY'}\|}\right). \tag{5.16}$$

Výsledná hodnota se nachází v intervalu $\langle 0, \pi \rangle$, neboť se jedná pouze o absolutní velikost. Naštěstí znaménko můžeme snadno určit užitím vektorového součinu, výsledný vztah potom nabyde tvaru:

$$\alpha = arccos\left(\frac{\overrightarrow{OX'}}{\|\overrightarrow{OX'}\|} \cdot \frac{\overrightarrow{OY'}}{\|\overrightarrow{OY'}\|}\right) \cdot sgn\left((\overrightarrow{OX'} \times \overrightarrow{OY'}) \cdot \mathbf{n}\right), \tag{5.17}$$

kde *sgn* představuje matematickou funkci *signum*, která pro záporné číslo vrací hodnotu -1, pro kladné $+1$ a pro nulu 0.



Obrázek 5.18: Znázornění prodloužení kosti podél lokální osy y. Bod X představuje pozici koncového efektoru, A nově zadanou pozici, O počátek lokální soustavy souřadnic. X' je průmětem koncového efektoru X na lokální osu y, A' průmětem zadané pozice.

Inverzní kinematika s pružnými kostmi

V případě, že kosti lze natahovat, přidáme kromě tří rotačních kroků ještě změnu délky kosti. Tento krok bude proveden pouze podél osy y, s níž je kost souhlasně orientovaná. Situace je zachycena na obrázku 5.18, na němž se používá stejné značení jako u rotace. Hledaná změna délky kosti odpovídá orientované úsečce $\overrightarrow{X'A'}$, body

X' a A' určíme pomocí průmětu na lokální osu y:

$$X' = O + (\overrightarrow{OX} \cdot \mathbf{b})\, \mathbf{b}, \tag{5.19}$$

$$A' = O + (\overrightarrow{OA} \cdot \mathbf{b})\, \mathbf{b}. \tag{5.20}$$

kde $\mathbf{b}$ je normalizovaný vektor souhlasně orientovaný s lokální osou y. Abychom zabránili singularitám, nedovolíme zkrácení kosti pod určitý práh (například nedovolíme, aby kost byla kratší než v klidovém stavu).

Výpočetní složitost

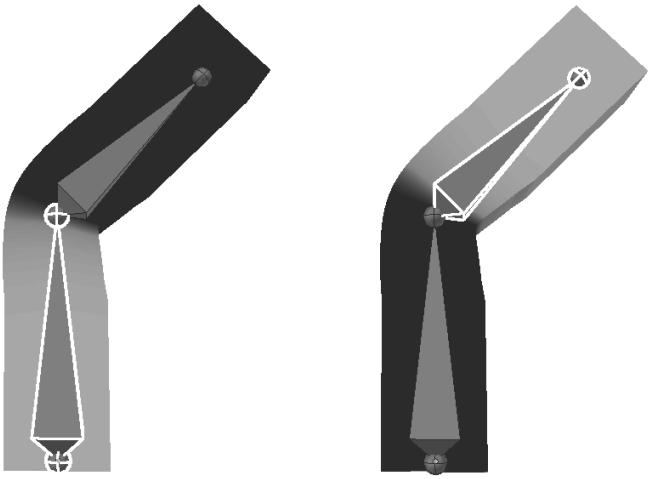
Složitost výpočtu úlohy inverzní kinematiky metodou CCD záleží na počtu kostí P, ze kterých se skládá segmentová struktura. Dále se mění počet nutných iterací I_{CCD} v závislosti na počáteční a cílové konfiguraci struktury. Výslednou výpočetní složitost lze formálně vyjádřit předpisem $O(I_{CCD}\, P)$.

5.4.2 Mesh-skinning

Změna animační kostry vyžaduje patřičnou úpravu trojúhelníkové sítě. Proto použijeme tzv. mesh-skinning, techniku deformace sítě, kdy se trojúhelníková síť přizpůsobuje kostře, která je jí přiřazena. Pro demonstrační aplikaci byla naimplementována varianta nazývaná linear blend skinning (LBS).

Linear blend skinning (LBS) deformuje síť pomocí lineární kombinace transformačních matic definovaných v kostech. Jedná se o nejzákladnější přístup k problematice, a přesto často používaný například v počítačových hrách zejména kvůli jednoduché implementaci a nízkým nárokům na cílový hardware.

Na vstupu máme trojúhelníkovou síť s m vrcholy $\mathbf{v}_i$ a kostru tvořenou uspořádanou množinou n kostí $\mathbf{b}_j$. Pro každou kost $\mathbf{b}_j$ je definována matice afinní transformace $\mathbf{M}_j$. Každý vrchol $\mathbf{v}_i$ obsahuje n definovaných vah příspěvku kosti w_{ij} (viz obrázek 5.21). Výstupem je deformovaná množina vrcholů, přičemž původní topologie sítě zůstává zachována. Deformace jednoho vrcholu se provádí podle vztahu 5.22.



Obrázek 5.21: Vizualizace vah kostí (ovlivnění vrcholu danou kostí). Světlá barva znamená velké ovlivnění, černá žádné. V místě kloubu mají vliv obě kosti.

Algoritmus je vysoce paralelizovatelný, vrcholy lze zpracovat naprosto nezávisle.

$$\mathbf{v}'_i = \sum_{j=0}^{n} w_{ij} \left(\mathbf{M}_j \, \mathbf{v}_i\right) = \left(\sum_{j=0}^{n} w_{ij} \, \mathbf{M}_j\right) \mathbf{v}_i \tag{5.22}$$

Jednoduchost LBS se podepisuje na viditelných artefaktech, mezi něž patří:

- ztráta objemu (zborcení sítě) zejména v místě velkého ohybu či zkrutu,
- sebeprotínání sítě v místech velkého ohybu či zkrutu.

Výpočet vah mesh-skinningu

Spolu s automaticky vygenerovanou kostrou bohužel nedostáváme rovnou váhy určené pro mesh-skinning. Tyto hodnoty zpravidla tvoří výtvarník (animátor) ve vhodném nástroji, lze je však pro naše účely poměrně dobře odhadnout.

K výpočtu použijeme síť ve stavu, v jakém se nacházela při generování kostry po poslední iteraci laplaceovského smršťování (tj. před zahájením decimace sítě). Od vstupní sítě předané aplikaci ke zpracování se liší pouze pozicemi vrcholů, které se ale vyskytují vhodně blízko vygenerované kostry.

Na základě tohoto pozorování určíme váhu kosti $\mathbf{b}_i$ pro vrchol $\mathbf{v}_j$ podle vztahu:

$$w_{ij} = \frac{1}{\varepsilon + dist(\mathbf{b}_i, \mathbf{v}_j)^2} \tag{5.23}$$

kde ε značí vhodné malé číslo (v aplikaci zvoleno $\varepsilon = 10^{-5}$, čím vyšší, tím plynulejší bude změna vlivu kostí v okolí kloubů), funkce *dist* vzdálenost vrcholu $\mathbf{v}_j$ od kosti $\mathbf{b}_i$, kterou při výpočtu vah chápeme jako úsečku. Získané váhy posléze musíme normalizovat:

$$\hat{w}_{ij} = \frac{w_{ij}}{\sum_i w_{ij}} \tag{5.24}$$

Z výkonnostních důvodů se obvykle používá stanovený počet vah, neboť valná většina je téměř rovna nule a kost má tak na deformovaný vrchol sotva zanedbatelný vliv. Demonstrační aplikace provádí mesh-skinning s užitím čtyř nejvýznamějších vah na jeden vrchol.

6 Implementace demonstrační aplikace

Kapitola se zabývá implementací metody generování kostry a následné interaktivní deformace modelu. Popisuje rozhodnutí, k nimž došlo během vývoje, specifikuje přesný formát vstupních dat a zabývá se architekturou vzniklé aplikace.

Demonstrační aplikace je spouštěna jako konzolová aplikace, která posléze vytvoří okno pro vykreslování scény. Uživatel tak zadává vstup skrze parametry programu a následně (po dokončení preprocessingu) ovládá scénu interaktivně myší a klávesnicí skrze vykreslovací okno.

6.1 Použité knihovny a jazyk

Demonstrační aplikace byla naprogramována v jazyce `C++` verze 2011 (`C++11`) a závisí na frameworku *VTK (Visualization Toolkit)* a knihovnách *libhungarian* a *Eigen*. Mimo tyto knihovny třetích stran je pro překlad vyžadována standardní knihovna `C++` opět verze 2011.

VTK je open source projektem společnosti Kitware, která jej aktivně vyvíjí a poskytuje k němu podporu. Tento rozsáhlý framework mimo jiné nabízí třídy umožňující vizualizace, zpracování obrazu, ale i abstrakci cílové platformy. V demonstrační aplikaci má na starosti veškeré vykreslování obsahu, zpracování vstupů od uživatele a pohyb ve scéně. Kitware poskytuje *VTK* v době psaní práce (2015) pod benevolentní BSD licencí.[1]

Knihovna *libhungarian* poskytuje implementaci maďarské metody v jazyce `C`, řeší tedy již zmiňované optimální párování s nejmenší ztrátou. Jejím autorem je Cyrill Stachniss.

Eigen je knihovna pro výpočty z oblasti lineární algebry. Kromě operací s maticemi a vektory disponuje například řešením soustav lineárních rovnic. Knihovna je zaštítěna licencí MPL2 [2].

[1] Oficiální licenční informace lze nalézt na adrese `http://www.vtk.org/licensing/`.

[2] Lze nalézt na adrese `http://eigen.tuxfamily.org/index.php?title=Main_Page#License`.

6.2 Formát vstupních dat

Abychom ukázkovou aplikaci nezatěžovali dalšími pomocnými knihovnami, bylo nutné zvolit takový vstupní formát dat, jehož načítání je podporováno již v rámci frameworku *VTK* nebo jeho vlastní implementace není obtížná. Druhým požadavkem je, abychom do použitého formátu snadlo exportovali existující data, a to opět existujícím řešením nebo vlastní implementací.

6.2.1 Vzorová kostra

Pro uchování vzorové kostry, která musí navíc uchovat omezení pohybu kloubech a lokální trojhrany, bohužel nebyl nalezen vhodný a zároveň jednoduchý formát. Kostra byla připravena v modelovacím a animačním programu Blender, který je naštěstí rozšiřitelný uživatelskými skripty v jazyce Python, pro něž poskytuje API v podstatě k veškeré své funkcionalitě [3]. Díky tomu mohl vzniknout exportovací skript do vlastního textového formátu, který obsahuje pouze námi vyžadované informace.

Soubor je uvozen nejprve číslem určujícím uložený počet kostí, které po něm následují. Každé kosti je dle pořadí přiřazen index, přičemž začínáme číslem nula. Zapsané souřadnice a matice jsou spadají do světových souřadnic, nejsou podřízeny jiné transformaci. Hodnoty v souboru jsou odděleny libovolným množstvím mezer, tabulátorů nebo odřádkování. Následuje strukturovaný popis formátu s vysvětlivkami.

Hlavička

`[počet kostí]`	Soubor je uvozen nejprve číslem určujícím uložený počet kostí, které po něm následují.

Pro každou kost

`[index nadřazené kosti]`	Index kosti nadřazené v hierarchii, -1 značí, že nadřazená vůči aktuální neexistuje.

[3] Viz Blender/Python Documentation:
`http://www.blender.org/api/blender_python_api_2_74_release/`

`[headx] [heady] [headz]`	Souřadnice počátku kosti zapsané v desetinných číslech.
`[tailx] [taily] [tailz]`	Souřadnice konce kosti zapsané v desetinných číslech.
`[rotx-min] [rotx-max]`	Limity rotace kosti kolem osy x v radiánech.
`[roty-min] [roty-max]`	Limity rotace kosti kolem osy y v radiánech.
`[rotz-min] [rotz-max]`	Limity rotace kosti kolem osy z v radiánech.

`[m11] [m12] [m13] [m14]`
`[m21] [m22] [m23] [m24]`
`[m31] [m32] [m33] [m34]`
`[m41] [m42] [m43] [m44]`
	Matice definující lokální prostor kosti.

Limitace rotace kosti, která vlastně představuje omezení pohybu v nadřazeném kloubu, určuje maximální rotaci kolem dané osy (procházející nadřazeným kloubem) z klidové pózy.

6.2.2 Trojúhelníková síť

Pro vstupní trojúhelníhovou síť reprezentující humanoida jsem zvolil formát *Wavefront OBJ*. Mimo trojúhelníkové sítě poskytuje uchovávání Bézierových plátů, přiřazení materiálů plochám atd. Aplikace nicméně předpokládá, že vstupní soubor obsahuje pouze trojúhelníky. Shrnující informace o formátu lze nalézt například v [1].

Formát *Wavefront OBJ* je podporován drtivou většinou programů pracujících s trojrozměrnými modely, které jej dokáží importovat i exportovat. V naší aplikaci jej snadno načteme užitím třídy `vtkOBJReader`, kterou poskytuje *VTK*.

6.3 Generování kostry

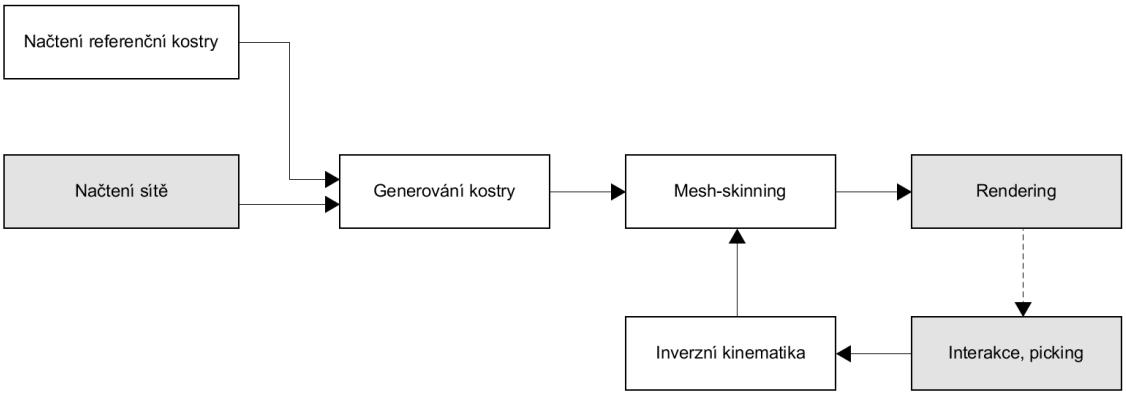
Kostra je v programu reprezentována třídou `CSkeleton`, která dokáže vykonat nad kostrou potřebné úkony, mimo jiné načíst ji ze souboru (využito u referenční kostry) nebo přiřadit odpovídající vrcholy prozatímní a referenční kostry a tak vytvořit výslednou animační kostru.

Tvorba prozatímní kostry (viz sekce 5.2) byla naprogramována děděním abstraktní třídy `vtkPolyDataAlgorithm`, která poskytuje možnost zpracování libovolných primitiv (přesněji vrcholů a jejich spojení v grafická primitiva). Dílčí části procesu byly rozděleny do dvou tříd: `MeshLaplaceFilter` zajišťuje smršťování modelu užitím laplaceovského vyhlazování, `EdgeCollapseFilter` provádí redukci geometrie a závěrečnou úpravu pozic.

Výsledná animační kostra, jak již bylo zmíněno, je vytvořena jednou z metod třídy `CSkeleton` – statickou metodou `CSkeleton::Match`, která přebírá přes argumenty referenční a prozatímní kostru. Mechanismus tvorby byl popsán v sekci 5.3.

6.4 Ukázková aplikace

V rámci diplomové práce byla kromě generování animační kostry naimplementována také demostrační aplikace, která dovoluje vzniklou kostru rozhýbat uživatelem, deformovat tak vstupní síť s použitím mesh-skinningu a výsledek zobrazovat. Načtení sítě ve formátu OBJ, rendering a zpracování vstupu byly realizovány pomocí tříd nabízených frameworkem VTK (`vtkPolyData`, `vtkPolyDataMapper`, `vtkActor` aj.). Znázornění pipeline se nachází na obrázku 6.1.



Obrázek 6.1: Znázornění pipeline demonstrační aplikace. Šedě jsou vyznačeny části implementované VTK, bíle vlastní implementace.

6.4.1 Inverzní kinematika

Přikročíme k interaktivnímu ovlivňování scény. Na pozice koncových efektorů umístíme pomocné prvky, jejichž přetažením dojde přepočítání konfigurace kostry – řešení úlohy inverzní kinematiky. Pomocné prvky jsou vybírány myší, pro výběr objektu ve scéně byla využita již naimplementovaná třída `vtkCellPicker` z frameworku VTK.

Inverzní kinematika realizovaná pomocí CCD byla naimplementována v pouhých několika funkcích, neboť se jedná o demonstrační ovlivnění scény nevelkého rozsahu. Funkce `ProcessIK`, po jejímž zavolání dojde k výpočtu, má dva parametry: první určuje, který koncový efektor byl přetažen na jiné místo, druhý je příznak, jestli jsou kosti pevné či pružné.

6.4.2 Mesh-skinning

V demonstrační aplikaci je skinning realizován jako třída `CSkinnedMesh`, která si udržuje stav kostry v klidové a současné póze a trojúhelníkovou síť v klidové póze. Tato data jsou uchována v podobě hluboké kopie, což umožňuje připravovat další snímek paralelně. Třída dědí od `vtkPolyDataAlgorithm`, jedná se o modul poskytovaný frameworkem VTK, který disponuje vstupními a výstupními datovými porty. Díky tomu dokáže předat výslednou síť standardní cestou k dalšímu zpracování nebo vizualizaci. Při změně konfigurace kostry dochází pouze ke změně pozic vrcholů, indexace sítě zůstává zachována.

6.4.3 Nastavení

Nastavení kostant a názvů souborů používaných aplikací je shomážděno centrálně ve třídě `TAppSettings` včetně přednastavených hodnot, které byly určeny empiricky na základě výsledků testování. `TAppSettings` je naimplementována jako struktura, což způsobuje, že všechny její atributy jsou defaultně veřejné.

Po spuštění aplikace nic nebrání dodatečné úpravě hodnot například parametry předanými skrze příkazovou řádku. Přesně tak lze měnit vybrané konstanty v demonstrační aplikaci.

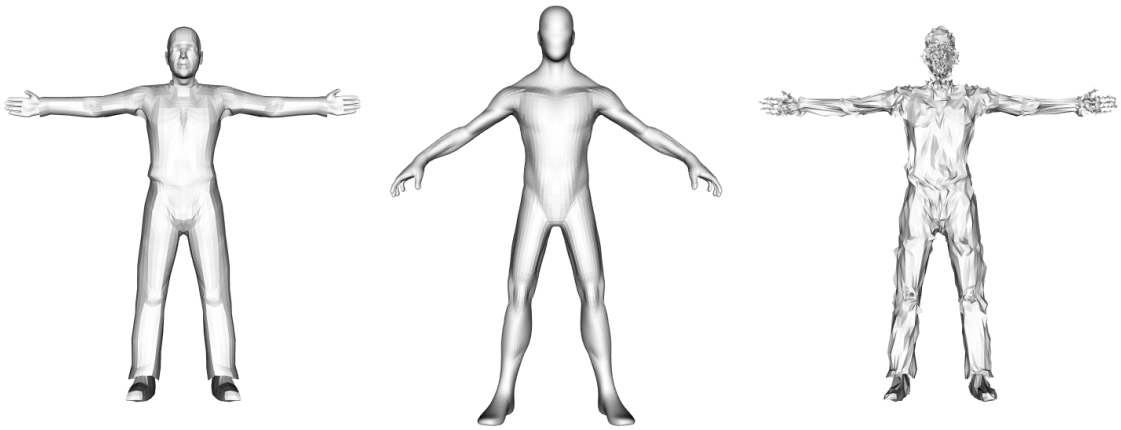
7 Výsledky

Metoda byla po implementaci otestována na několika zkušebních modelech. Jedná se o manifoldní trojúhelníkové sítě představující více či méně humanoidní charaktery.

Mezi modely byly záměrně zařazeny modely, které by mohly činit potíže, například se nenacházejí přesně v klidové póze (což může ovlivnit generování kostry, ale i fungování inverzní kinematiky, která pro omezení v kloubech předpokládá stále stejné nastavení klidové pózy modelu).

7.1 Testovací modely

Následuje představení použitých modelů a jejich charakteristických vlastností. Jedná se o modely poskytnuté zdarma v různých databankách, které musely být mírně upraveny, aby byly manifoldní.



Obrázek 7.1: Ukázka vstupních modelů, zleva human, human-low, human-high.

- **human** – Trojúhelníková síť představující člověka připravená v různých rozlišeních (pro testování vlivu na výkon). Hraniční verze human-high obsahuje čtyřikrát více trojúhelníků, verze human-low naopak čtyřikrát méně než human. Na základě tohoto modelu byla vytvořena referenční kostra. Model se nachází vlevo na obrázku 7.1.

- **human2** – Trojúhelníková síť představující člověka. Model nestojí přesně v klidové póze, což může znesnadnit generování kostry (viz uprostřed na obrázku 7.1).

- **human-noise** – Jedná se o zašuměnou podobu modelu human (viz vpravo na obrázku 7.1), tj. pozice vrcholů modelu jsou mírně náhodně odchýleny.

- **armadillo** – Síť představující stylizovaného humanoidního pásovce. Model nestojí přesně v klidové póze, má ocas a dlouhé uši na hlavě (viz vlevo na obrázku 7.2).

- **minotauro** – Minotaur bez rohů, model má vyšší počet trojúhelníků ve srovnání s ostatními (viz uprostřed na obrázku 7.2). [1]

- **stickman** – Velmi zjednodušená podoba humanoida, tělo je velmi tenké (viz vpravo na obrázku 7.2).

Obrázek 7.2: Ukázka vstupních modelů, zleva armadillo, minotauro, stickman.

[1] Autor: `https://www.youtube.com/user/blendermodels4free/about`.

7.2 Nastavení konstant generování kostry

Pro výše představené modely byla vygenerována animační kostra. Empiricky byly určeny optimální hodnoty konstant, které byly popsány v kapitole 5.2 a 5.3. Pro fázi smršťování:

- maximální počet iterací $I_C = 6$
- počáteční hodnota $W_H = 0.5$
- zesilovací koeficient $s_L = 2.5$
- práh $V_{min} = 10^{-5}$

Pro fázi redukce geometrie:

- penalizace křivosti $C_{shape} = 0.1$
- penalizace délky kostí $C_{sampling} = 1$

Pro fázi přiřazování listů kostry:

- váha výsledku DWT $c_{DWT} = 1$
- váha vzdálenosti od pozice vrcholu $c_{dist} = 0.01$

Pro fázi přiřazování vnitřních uzlů kostry:

- váha rozdílu vzdáleností k listům $c_{E_L} = 1$
- váha rozdílu stupně vrcholu $c_{E_D} = 0.1$
- váha vzdálenosti k vrcholu $c_{E_P} = 0$

Přesto se ukázalo, že u některých modelů je nutné některé parametry přenastavit na jiné hodnoty.

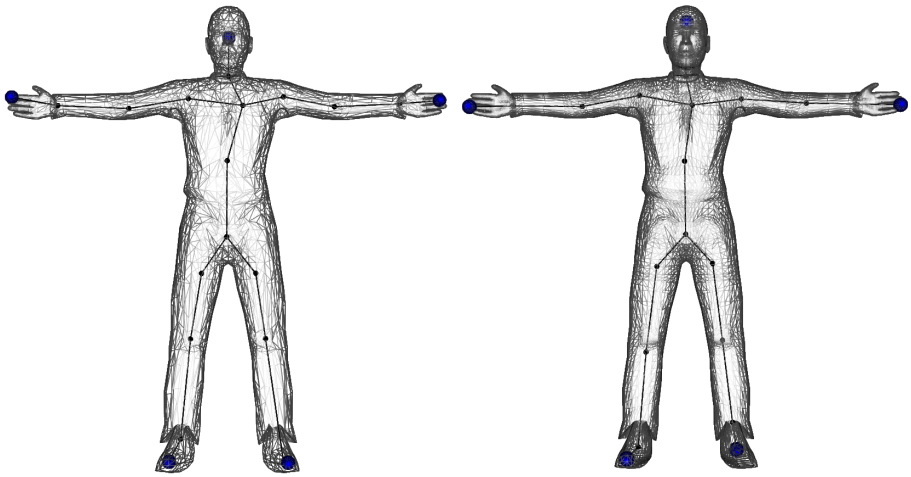
U modelu armadillo docházelo k nedostatečnému smršťování sítě, proto byl počet iterací laplaceovského vyhlazování I_C zvýšen na hodnotu 9 a zesilovací koeficient s_L na 3.

Model human2 poskytoval nejlepší výsledky po pouhých třech iteracích, vyšší počty zhoršovaly podobu kostry v oblasti ramen.

Model stickman poskytoval nejlepší výsledky po pouhých třech iteracích a nastavení zesilovacího koeficientu s_L na hodnotu 1.5.

7.3 Výsledná podoba kostry

Ukažme si výsledné animační kostry vygenerované naší metodou. Následující seznam popisuje vzniklé vady, výsledky jsou mimo to zachyceny na obrázku 7.4 a 7.5.



Obrázek 7.3: Model human ve dvou různých rozlišeních v drátěné podobě včetně vygenerované kostry.

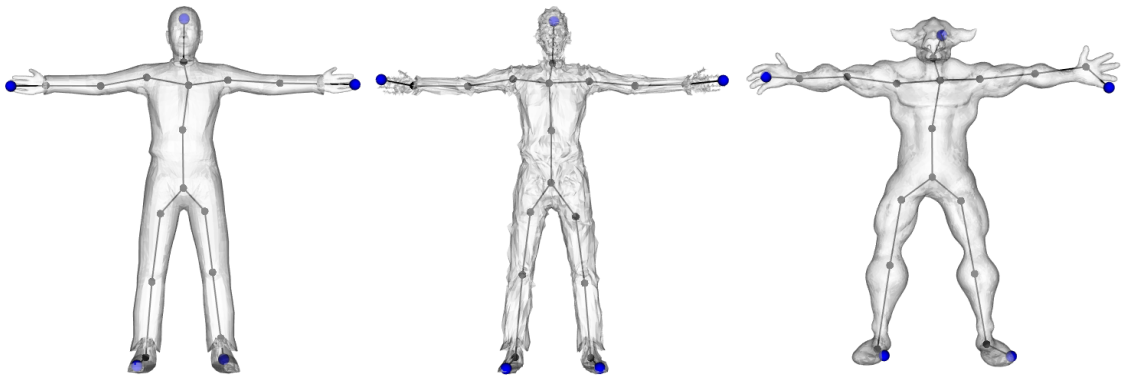
- **human, human-noise** – Kvalitativně výborný výsledek nezávisle na hustotě sítě či jejím zašumění (viz obrázek 7.3), kostra místy nepatrně vychází z vnitřku sítě.

- **minotauro** – Dobrý výsledek s mírným odchýlením pozic vrcholů (ruka z našeho pohledu vpravo, noha vlevo). Kostra vychází z vnitřku sítě.

- **stickman** – Dobrý výsledek, nesymetrické umístění loketních, kyčelních a ramenních kloubů, což lze tolerovat vzhledem k jiným proporcím oproti vzorové kostře. Kostra místy vychází z vnitřku sítě. Vstupní síť nerespektuje přesně klidovou pózu (dlaně směřují dolů), omezení volnosti pohybu v kloubech tak postrádá na správném účinku.

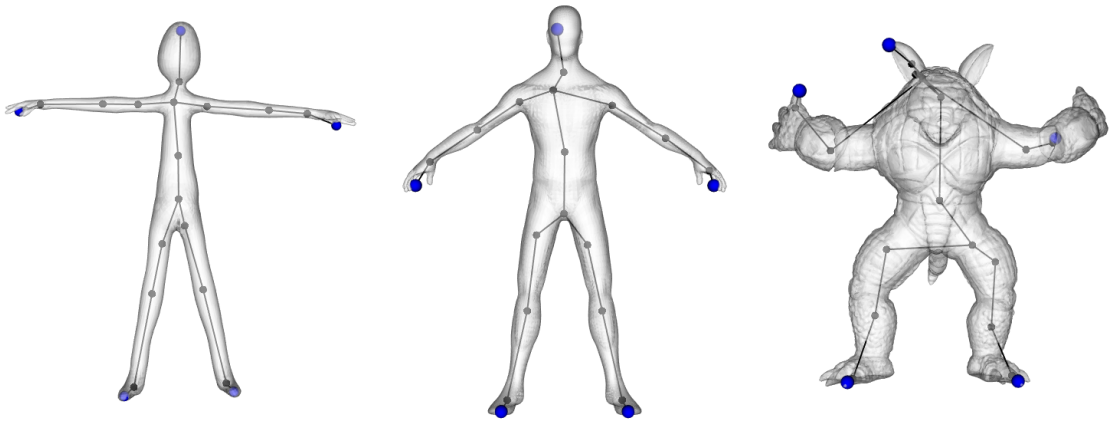
- **human2** – Vstupní model nerespektuje klidovou pózu. Výsledná kostra je přesto vygenerována správně (až na omezení volnosti pohybu v kloubech), mírně vyčnívá z vnitřku sítě.

- **armadillo** – Špatný výsledek, zmiňme zejména ramenní klouby, které jsou chybně umístěny do vnitřku hlavy. Problém bude patrně v dlouhých uších, z nichž je jedno metodou vybráno jako celá hlava. Navzdory špatnému výsledku v horní části těla lze dobře interagovat s dolními.



Obrázek 7.4: Výsledné animační kostry modelů human, human-noise, minotauro (zleva).

Z výsledků plyne následující pozorování. Aby byla animační kostra vygenerována správně, musí vstupní model stát přesně v klidové póze a jeho proporce musí co nejvíce souhlasit s proporcemi referenční kostry. Patrně by bylo vhodné vytvořit více referenčních koster s různými proporcemi, přičemž by se výpočet provedl s užitím každé z nich a jako výsledek by byl vybrán ten, který je zatížen nejmenší chybou. Ta je ostatně vyčíslována již nyní v rámci dílčích kroků přiřazování vrcholů kostry.



Obrázek 7.5: Výsledné animační kostry modelů stickman, human2 a armadillo (zleva).

7.4 Doba generování kostry

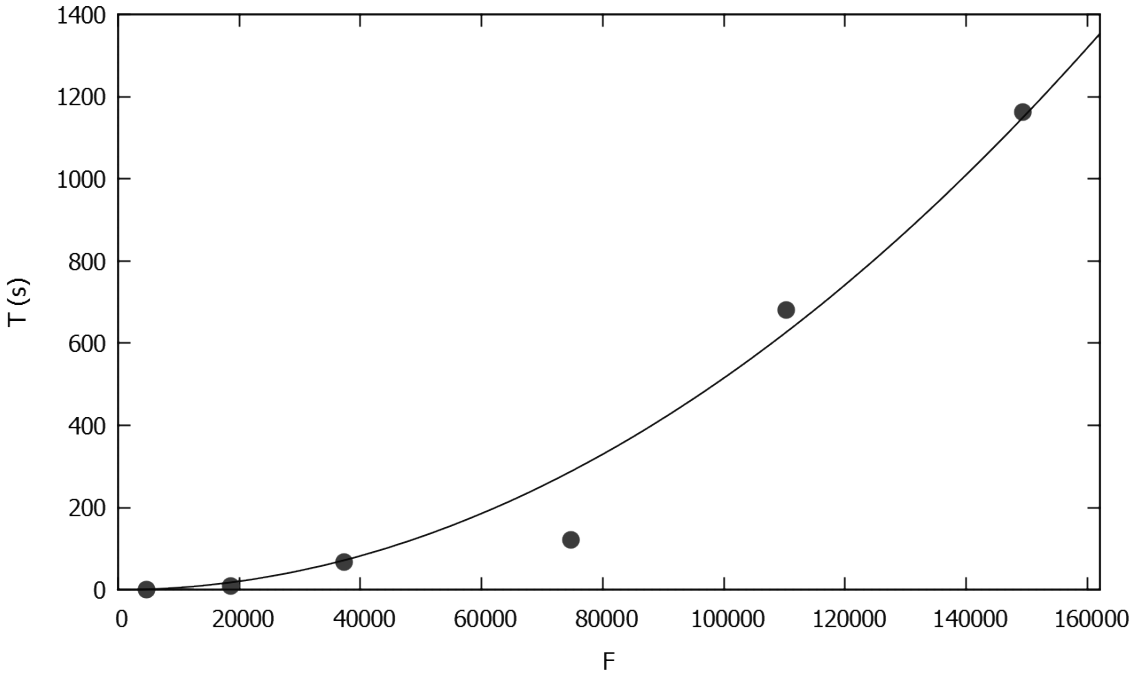
Doba generování animační kostry pro vybrané modely byla změřena na referenčním stroji s procesorem Intel®Core™ i5-3350P o frekvenci 3.10 GHz a 8 GB RAM. Měření bylo provedeno opakovaně, v potaz byla brána nejlepší naměřená hodnota (předpokládáme, že horší výsledek je způsobený například preemptivním chováním plánovače operačního systému, který rozhodl v neprospěch našeho programu). Nejprve ukažme vliv počtu trojúhelníků modelu na dobu generování kostry, použijeme sítě různých rozlišení reprezentující tentýž model *human*.

Model	**Vrcholy** V	**Trojúhelníky** F	**Čas** T
human-low	2336	4668	656 ms
human	9338	18672	9110 ms
human-mid-high	18674	37344	67727 ms
human-high	37346	74688	120388 ms
human-high2	55160	110316	680934 ms
human-high3	74690	149376	1163010 ms

V následující tabulce prozkoumáme poměr $T/p(F)$, kde p značí polynomiální funkci, kterou se snažíme popsat výpočetní složitost. Poměry jsou normalizovány do srovnatelného řádu. Pokud naměřená data odpovídají svou závislostí zvolené funkci p, pak budou hodnoty ve sloupci oscilovat kolem konstanty.

Model	T/F	T/F^2	T/F^3
human-low	1.4053	3.0105	6.4493
human	4.8789	2.6130	1.3994
human-mid-high	18.1360	4.8565	1.3005
human-high	16.1188	2.1581	0.2890
human-high2	61.7258	5.5953	0.5072
human-high3	77.8579	5.2122	0.3489

Strmě rostoucí hodnoty sloupce T/F nenasvědčují lineární závislosti T na F, naopak pro kvadratickou závislost se mění poměr nejméně. Předpokládaná kubická složitost algoritmu příliš neodpovídá výsledkům – nejspíše kvůli tomu, že řešené soustavy lineárních rovnic, které měly kubickou výpočetní složitost způsobit, jsou ve skutečnosti velmi řídké. Výslednou závislost doby výpočtu T na počtu trojúhelníků F zaneseme do grafu (viz obrázek 7.6).



Obrázek 7.6: Naměřené doby výpočtu T v závislosti na počtu trojúhelníků F včetně kvadratické regresní funkce $T' = 0.0515435 \cdot 10^{-6} F^2$.

V další tabulce porovnáváme časy všech vstupních sítí představených v kapitole 7.1:

Model	**Vrcholy**	**Trojúhelníky**	**Iterace smršťování**	**Čas**
armadillo	34594	69184	9	325405 ms
human	9338	18672	6	9110 ms
human2	14812	29620	3	11601 ms
human-noise	9338	18672	6	9147 ms
minotauro	11832	23660	6	15978 ms
stickman	3798	7592	3	1056 ms

Nesmíme zapomenout, že celková doba generování je dána kromě počtu vrcholů a trojúhelníků vstupní sítě také jejím tvarem. Složitý tvar sítě vyžaduje více iterací pro smršťování laplaceovským vyhlazováním, což se projevilo na vysokém čase u modelu armadillo. Zvyšování počtu iterací má nicméně pouze lineární vliv na výslednou dobu výpočtu kostry (jedná se v podstatě o opakování jedné části algoritmu).

7.5 Paměťová složitost generování kostry

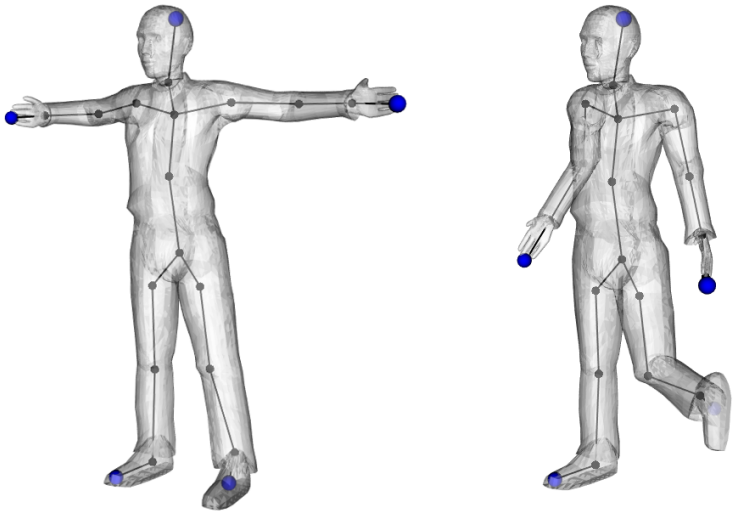
Z hlediska paměťové složitosti se nejedná o zajímavou úlohu, při použití řídkých matic je velikost alokované paměti pro výpočet lineárně úměrná velikosti vstupních dat (tj. počtu trojúhelníků). Pouze při tvorbě matice pro maďarskou metodu alokujeme paměť o velikosti $M \times N$, kde M a N však odpovídá pouze počtu listů prozatímní, respektive referenční kostry. Počet listů referenční kostry je však známý ($M = 5$), jedná se v podstatě o konstantu, a tak se opět dostáváme na lineární složitost.

U nejkomplexnějšího modelu armadillo nepřekročila veškerá paměť alokovaná programem hranici 50 MB během generování kostry, 150 MB během vykreslování.

7.6 Doba výpočtu inverzní kinematiky

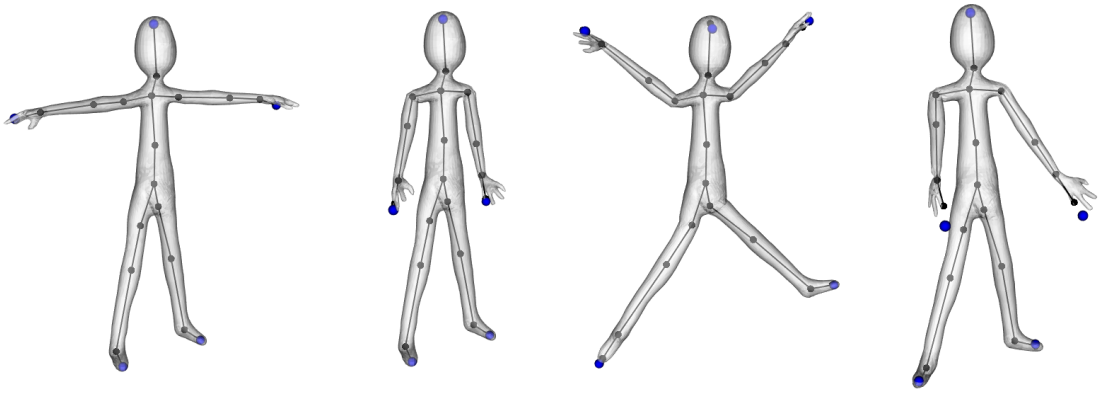
Výpočet úlohy inverzní kinematiky probíhal v reálném čase. Pro všechny vstupní modely byla doba výpočtu shodná, neboť používáme shodnou kostru.

Nástroje pro přesné měření času ze standardní knihovny jazyka C++ jsou bohužel stále naimplementovány pro platformu Windows s nedostatečným rozlišením, proto musely být pro meření užity funkce závislé na platformě: `QueryPerformaceCounter` a `QueryPerformaceFrequency`.



Obrázek 7.7: Inverzní kinematika aplikovaná na model human.

V případě, že byly zakázány pružné segmenty, trval výpočet méně než 10 ms, s pružnými kostmi pak 15 ms. Ukázky deformovaných modelů nalezne čtenář na obrázcích 7.7 a 7.8.



Obrázek 7.8: Inverzní kinematika aplikovaná na model stickman.

7.7 Srovnání s existujícími metodami

S ohledem na metody prozkoumané v kapitole 2 můžeme prohlásit, že vznikla unikátní, dosud neexistující metoda, která pro vstupní trojúhelníkovou síť humanoida generuje kostru určenou pro animaci, a to včetně definic omezení pohybu

v kloubech. V následujících několika odstavcích srovnáme vzniklou metodu a její výsledky s ostatními metodami popsanými v kapitole 2.

Metoda od Straka et al. (kapitola 2.3.1), podle které bylo vytvořeno přiřazování vrcholů referenční a prozatímní kostry, měla vytvořenou počáteční kostru na základě snímaných volumetrických dat a zpracování muselo probíhat v reálném čase. Na rozdíl od původní podoby byla naše část metody rozšířena o výpočet orientace lokálního souřadnicového systému a zkopírování definic omezení pohybu v kloubech.

Z oblasti geometrických metod jsme prozkoumali metodu *medial axis transform* a její odhad pomocí spojení pólů Voronného diagramu. Tento přístup je na rozdíl od naší metody citlivý na zašumění povrchových dat a produkuje velké množství prvků skeletu.

Metoda od Au et al., představená v kapitole 2.2.2, se stala součástí naší metody, slouží ke generování prozatímní kostry.

Tvorba kostry spojováním význačných bodů (metoda od Ma et al.), kterou jsme se zabývali v kapitole 2.2.2, na rozdíl od naší metody nemusí nikterak vstupní data iterativně smršťovat. Její složitost, nebereme-li v úvahu preprocessing v podobě generování tečných koulí, je beztak těžko vyjádřitelná a pro komplexní modely bude pravděpodobně časově srovnatelná s metodou Au et al. Přístup spojováním význačných bodů má tu výhodu, že produkuje poměrně malý počet kostí. Zůstává však otázkou, mají-li vzniklé kosti alespoň přibližně anatomické umístění pro animaci.

Naše metoda (podobně jako metoda od Straka et al.) se specializuje na tvorbu kostry humanoida, omezení volnosti pohybu v kloubech proto byla definována napevno u referenční kostry. Bez znalosti referenční kostry bychom omezení mohli aplikovat například poloautomatizovaně, kdy by uživatel zvolil u vzniklých kloubů jejich typ (kulový, válcový atd.) a limity by byly dopočítány testováním kolizí. Klouby by se zkrátka zkusmo ohnuly do svých nejzazších mezí, kde ještě nedochází ke kolizi obalových těles sousedních kostí. Nabízí se ale otázka, zda takto vzniklá kostra skutečně usnadní tvorbu animace, neboť se dá předpokládat, že bude vyžadovat dodatečné úpravy, které nakonec budou možná srovnatelně obtížné s manuální tvorbou celé animační kostry.

8 Závěr

Cílem práce bylo vytvořit metodu, která vygeneruje vhodnou animační kostru pro model představující humanoida. Na základě průzkumu a rozboru několika metod byl navržen vlastní postup pro automatické generování kostry humanoida z trojúhelníkové sítě. Vzniklá metoda využívá referenční kostru, ze které jsou přeneseny charakteristické vlastnosti do výsledné kostry: především počet kostí a jejich hierarchické uspořádání, orientace lokálních souřadnicových systémů a omezení volnosti pohybu v kloubech.

Metoda byla otestována na trojúhelníkových sítích různých hustot a podobností vůči člověku. Ukázalo se, že u modelů, které jsou blízké referenční kostře (tj. dodržují klidovou pózu a mají proporce člověka, což je ostatně současnou největší limitací našeho postupu), dosáhneme kvalitnějších výsledků. Vzniklá animační kostra tak potřebuje jen drobné ruční závěrečné úpravy.

Kromě samotného generování došlo na implementaci interaktivní deformace humanoida skrze inverzní kinematiku, což umožňuje snadno zjistit, zda kostra byla vygenerována správně. K řešení inverzní kinematiky byla užita vlastní implementace přístupu *cyclic coordinate descent*.

Výkonnostně poznamenává generování kostry fakt, že je během něho opakovaně řešena soustava lineárních rovnic. Na druhou stranu, jak již bylo naznačeno v úvodu, cílem práce bylo automatizovat část animátorského procesu – během výpočtu kostry se tak výtvarník může věnovat jiné činnosti. Demonstrační interakce s modelem oproti tomu probíhá v reálném čase.

Navržená metoda by mohla být rozšířena tak, aby vhodně volila z více referenčních koster, což by bylo možné realizovat hledáním případu, kdy se nejméně liší referenční a vygenerovaná kostra. Další možností vylepšení by byla úprava omezení v kloubech podle vlastností vstupní sítě, neboť současná metoda předpokládá neměnné hodnoty, případně korekce klidové pózy modelu pro správné přiřazení omezení kloubů.

Literatura

[1] Wavefront OBJ File Format Summary [online]. cit. 8.4.2015.
Dostupné z: `http://www.fileformat.info/format/wavefrontobj/egff.htm`

[2] Hungarian algorithm. *Wikipedia: the free encyclopedia* [online]. Wikimedia Foundation. 2001-. San Francisco (CA).
Dostupné z: `http://en.wikipedia.org/wiki/Hungarian_algorithm`

[3] Amenta, N.; Choi, S.; Kolluri, R. K.: The Power Crust, Unions of Balls, and the Medial Axis Transform. *Computational Geometry: Theory and Applications*, ročník 19, 2000: s. 127–153.

[4] Au, O. K.-C.; Tai, C.-L.; Chu, H.-K.; aj.: Skeleton extraction by mesh contraction. *ACM Transactions on Graphics*, ročník 27, č. 3, 2008: s. 1–1722.

[5] Beneš, B.; Žára, J.; Sochor, J.; aj.: *Moderní počítačová grafika*. Brno: Computer Press, první vydání, 2004.

[6] Blum, H.: A Transformation for Extracting New Descriptors of Shape. In *Models for the Perception of Speech and Visual Form*, editace W. Wathen-Dunn, Cambridge: MIT Press, 1967, s. 362–380.

[7] Denavit, J.; Hartenberg, R. S.: A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 1955.

[8] Goliáš, R.: *Inverzní kinematika. Diplomová práce*. Brno: Masarykova univerzita, 1999.

[9] Ma, C.-M.; Wan, S.-Y.; Lee, J.-D.: Three-dimensional topology preserving reduction on the 4-subfields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník 24, č. 12, 2002: s. 1594–1605, ISSN 0162-8828, doi:10.1109/TPAMI.2002.1114851.

[10] Ma, J.; Bae, S.; Choi, S.: 3D medial axis point approximation using nearest neighbors and the normal field. *The Visual Computer*, ročník 28, č. 1, 2012: s. 7–19, ISSN 0178-2789, doi:10.1007/s00371-011-0594-7.

[11] Ma, J.; Choi, S.: Kinematic skeleton extraction from 3D articulated models. *Computer-Aided Design*, ročník 46, 2014: s. 221–226.

[12] Meredith, M.; Maddock, S.: Real-Time Inverse Kinematics: The Return of the Jacobian. 2004.

[13] Rodriguez, A.; Ehlenberger, D. B.; Hof, P. R.; aj.: Three-dimensional neuron tracing by voxel scooping. *Journal of Neuroscience Methods*, ročník 184, č. 1, 2009: s. 169–175, ISSN 0165-0270.

[14] Straka, M.; Hauswiesner, S.; Rüther, M.; aj.: Skeletal Graph Based Human Pose Estimation in Real-Time. In *Proceedings of the British Machine Vision Conference*, BMVA Press, 2011, ISBN 1-901725-43-X.

[15] Wang, L.-C.; Chen, C.: A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *Robotics and Automation, IEEE Transactions on*, ročník 7, č. 4, Aug 1991: s. 489–499, ISSN 1042-296X.

[16] Yoshizawa, S.; Belyaev, A. G.; Seidel, H.-P.: Skeleton-based Variational Mesh Deformations. *Computer Graphics Forum*, ročník 26, č. 3, 2007: s. 255–264, doi:10.1111/j.1467-8659.2007.01047.x.

A Příloha: Obsah CD

Tato příloha popisuje adresářovou strukturu přiloženého CD.

binary	spustitelné soubory práce a potřebná data
build	EXE a DLL soubory
data	modely ve formátu Wavefront OBJ
run	připravené spouštěcí skripty
doc	diplomová práce ve formátu PDF
tex	zdrojové soubory bakalářské práce (LaTeX)
source	veškeré zdrojové soubory aplikace
blender-bones-script	exportovací skript kostry pro program Blender
program	zdrojové soubory samotné aplikace

B Příloha: Uživatelský manuál

Popišme si nyní ve stručnosti překlad zdrojových kódů aplikace a její ovládání.

B.1 Překlad

Aplikace byla naprogramována v jazyce C++ a sestavována pomocí programu CMake a Microsoft Visual Studio. Pro překlad je nutné dodržet následující kroky:

1. Stáhneme a nainstalujeme program CMake (`http://www.cmake.org/`).
2. Stáhneme framework VTK verze 6.0.0 (`http://www.vtk.org/`).
3. Za pomoci programu CMake vygenerujeme projektový soubor VTK pro Visual Studio, který umožní překlad a instalaci frameworku. Generování se provádí klasickou cestou – zvolí se kořenový adresář VTK, ve kterém se nachází soubor `CMakeLists.txt`.
4. Vzniklý projekt přeložíme ve Visual Studiu.
5. Za pomoci programu CMake vygenerujeme projektový soubor diplomové práce pro Visual Studio. Zdrojové soubory se nacházejí na CD v adresáři `source\program`.
6. Vzniklý projekt diplomové práce přeložíme ve Visual Studiu.

B.2 Parametry příkazové řádky

Po přeložení můžeme spustit aplikaci v příkazové řádce parametry. Formát je ve tvaru:

```
skeletonGenerator.exe "model.obj"[další možnosti]
```

Jako první parametr předáváme název souboru obsahující model ve formátu Wavefront OBJ. Další dobrovolné možnosti jsou předávány vždy po dvojicích; první je název, druhá hodnota parametru.

- `itercount N` – Smršťování sítě bude provedeno maximálně N-krát. Základní hodnota $N = 6$.
- `poscoeff p` – Vážení vlivu pozic při smršťování sítě. Základní hodnota $p = 0.5$.
- `areacoeff a` – Vážení vlivu obsahu trojúhelníků při smršťování sítě. Základní hodnota $a = 1.0$.
- `sl s` – Zesilování smršťování s každou iterací, každou iteraci zesíleno s-krát. Základní hodnota $s = 2.5$.
- `volumecoeff V` – Prahový objem V, po jeho dosažení jsou iterace smršťování ukončeny. Základní hodnota $V = 10^{-5}$.

B.3 Ovládání aplikace

Po dokončení generování kostry je uživateli dovolena interakce s modelem skrze vizualizační okno. Popišme si možnosti ovládání tlačítky klávesnice:

Tlačítko	Funkce
1	Skrytí/odkrytí vygenerované kostry.
2	Skrytí/odkrytí lokálních trojhranů.
4	Skrytí/odkrytí sítě.
5	Vizualizace kosti s největším vlivem (segmentace).
6	Skrytí/odkrytí smrštěné sítě.
7	Povolení pružných kostí inverzní kinematiky.
mezerník	Reset pózy modelu.
W	Zobrazit jako drátěný model.
S	Zrušit zobrazení drátěného modelu.

Tažením myši za stisku levého tlačítka lze rotovat s kamerou, pravé tlačítko ovládá přiblížení a prostřední pozici kamery. Myší lze rovněž přesouvat modré ovladače

(nacházejí se v listech kostry), což způsobí deformaci modelu s užitím inverzní kinematiky.

Do konzole jsou po celou dobu běhu programu vypisována informační hlášení.