Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

Diplomová práce

Rekonstrukce pózy lidské ruky z hloubkového obrazu

Autor:

Milan Herbig

Vedoucí práce:

Ing. Zdeněk Krňoul, Ph.D.

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 30. dubna 2016

Podpis: _____

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Ing. Zdeňku Krňoulovi, Ph.D. za odborné vedení, podnětné konzultace a ochotnou pomoc při vypracování této práce.

Děkuji virtuální organizaci MetaCentrum VO za přístup ke gridovému výpočetnímu prostředí, ve kterém probíhaly veškeré výpočty a bez kterého by tato práce nemohla vzniknout. MetaCentrum VO je součástí programu "Projects of Large Research, Development, and Innovations Infrastructures"(CESNET LM2015042).

Abstrakt

Téma práce je zaměřeno na aplikaci state-of-the-art konvolučních neuronových sítí za účelem rekonstrukce pózy lidské ruky z hloubkových dat pořízených kamerou Microsoft Kinect v2. V práci se věnuji současným metodám a přístupům k rekonstrukci pózy, problémům spojeným s trénováním neuronových sítí, tvorbě trénovacích dat, jejich normalizaci a konečně návrhu samotné architektury neuronové sítě pro účely regrese pózy lidské ruky. V závěru diskutuji dosažené výsledky.

Výstupem práce je jednak navržená architektura konvoluční neuronové sítě a jednak natrénovaný model. Zároveň jsem vytvořil modul pro snímání a segmentaci hloubkových dat z kamery včetně modulu pro vizualizaci výsledků rekonstrukce. Celý systém běží v reálném čase s využitím výpočtů na grafické kartě.

Klíčová slova: regrese, odhad parametrů, rekonstrukce pózy, 3D model, strojové učení, konvoluční neuronová síť

Abstract

This thesis explores possibilities and contributions of state-of-the-art convolutional neural networks to hand pose estimation problem. Hand pose is estimated from depth images recorded by Microsoft Kinect v2 depth camera. Current hand estimation methods and neural networks in general are discussed. The focus is set on problems during neural network training, data creation and normalization. Whole chapter is devoted to regression convolution neural network model design. Finally, observed results are discussed in the last chapter.

This work produces both proposed and trained neural network model. Also, for real-word testing and demonstration purposes, both segmentation and visualization modules were developed. It is worth noting that whole pipeline runs in real time on GPU.

Keywords: regression, parameter estimation, pose reconstruction, 3D model, machine learning, convolutional neural network

Obsah

1 Úvod

Zájem o umělou inteligenci neboli o napodobení duševní činnosti člověka sahá historicky hluboko do minulosti. Samostatný vznik výzkumné oblasti umělé inteligence však můžeme pozorovat až s příchodem prvních počítačů během první poloviny minulého století. První vizionářské pokusy o napodobení funkce mozku, přesněji řečeno doslova o jeho nadrátování, hořce selhaly, nicméně daly impuls pro vznik úplně nových oborů. S uplynulými léty, jak postupoval vývoj výpočetní techniky, rozšiřovala se i oblast umělé inteligence. Od prvních programů určených k jedné konkrétní činnosti až po samoučící se entity schopné konkurovat člověku. Od prvních šachových partií až po vědomostní soutěže. Kombinace narůstajícího výpočetního výkonu a raketově rostoucího množství digitálních dat vedla k nastartování fenoménu zvaného strojové učení. Jak dnes ukazuje současný vývoj, napodobení mozku se zdá být tou správnou cestou. V posledních letech tak oboru umělé inteligence vévodí neuronové sítě. Složité neuronové struktury zpracovávají veliké množství dat nejrůznějšího charakteru. Zájem veřejnosti už dávno přerostl akademickou sféru, přičemž nejnovější poznatky nachází uplatnění v mnoha odvětvích průmyslu a obchodu.

Účelem této práce je blíže se seznámit s těmito novými trendy, prozkoumat nové metody strojového učení založené na neuronových sítích a získané poznatky posléze aplikovat v praxi za účelem odhadu a rekonstrukce pózy lidské ruky z hloubkových dat. Práce je rozčleněna na teoretickou a praktickou část. V teoretické části se věnuji stručné historii rekonstrukce pózy, současným metodám a neuronovým sítím. V praktické části se zabývám přípravou dat, návrhem neuronové sítě a trénováním modelu. Nakonec navržený model testuji na reálných datech z 3D kamery a diskutuji získané výsledky. V textu se místy vyskytují cizojazyčné termíny, které buďto nemají ustálený český překlad, nebo je jejich cizojazyčná varianta natolik zažitá, že překlad do češtiny by působil spíše na škodu.

2 Motivace

Úloha přesné a robustní rekonstrukce pózy ruky dodnes stále představuje výzvu pro skupiny výzkumníků po celém světě. Přitom obecnou rekonstrukcí pózy z obrazu se výzkumníci zabývají už dlouhá léta. Příkladem dnes již běžně fungujícího prostředku pro odhad polohy a pohybu částí těla je zařízení Microsoft Kinect, jehož komerční úspěch tkví v ovládání jednoduchých videoher pohybem celého těla. Problémovou částí však nadále zůstávají ruce. Ty bývají kvůli zjednodušení nebo nedostatečnému rozlišení snímače modelovány velice zjednodušeně, či vůbec. Skelet ruky totiž oproti skeletu těla skrývá daleko více stupňů volnosti. Zároveň je ruka nepoměrně menší vůči zbytku těla, tedy veškerá informace je zakódovaná v podstatně menším množství pixelů. Jednotlivé části ruky se navíc často překrývají a vzájemně vykazují velkou lokální podobnost. Návrh systému pro přesnou a robustní rekonstrukci tak představuje nelehký úkol, při jehož řešení vzniklo v průběhu let mnoho publikací popisující různé přístupy.

Příchod nového typu hloubkových snímačů, jejich miniaturizace a finanční dostupnost zájem o rekonstrukci pózy ruky rozvířil. Schopnost rozpoznat či zrekonstruovat pózu ruky otevírá nové možnosti ovládání nejrůznějších zařízení na dálku, například televize. Dlouholetou motivací je bezpochyby úloha rozpoznávání znakové řeči nebo emocí. Aktuálním trendem je rychle rostoucí perspektivní odvětví rozšířené a virtuální reality. Zde se vyžaduje naprosto přesná a věrohodná rekonstrukce celého těla do nejmenších detailů, navíc v reálném čase tak, aby se výsledný zážitek co nejvíce přiblížil skutečnosti. Jistě bychom vymysleli další příklady využití.

3 Historie rozpoznávání gest

Zájem o rozpoznávání gest ruky sahá poměrně hluboko do novodobé počítačové historie. Na konci 70. let se objevují první prototypy speciální tzv. datové rukavice, která prostřednictvím fyzických snímačů dokáže měřit sevření dlaně, rozevření dlaně i pohyby jednotlivých prstů. Na veřejnost se první modely dostávají až po několika letech a tím se koncem 80. let i přes velmi omezené výpočetní kapacity otevírají možnosti výzkumu zcela nového rozhraní pro ovládání strojů a jiných zařízení.

První datové rukavice využívaly dva základní fyzikální principy pro detekci ohybu v kloubu [1]. První z nich měřil deformaci rukavice jako změnu napětí na fotočlánku, respektive změnu intenzity dopadajícího světla. Konstrukce rukavice byla tvořena optickými kanály, do nichž LED dioda vysílala světelný signál. Ohnutím prstu dochází k ohybu optického kanálu. V daném místě tak světelný paprsek dopadá na stěnu kanálu pod mnohem větším úhlem, což má za následek zhoršený odraz měřitelný fotočlánkem na konci kanálu. Druhým základním principem hojně využívaným v prvních datových rukavicích je změna elektrických vlastností materiálu (odpor, piezoelektrický jev) pod působící deformační silou. Oba zmíněné principy však v raných fázích měly problém s přesností detekce. Situace se zlepšovala s vývojem v oblasti snímačů v průběhu 90. let. Software u pozdějších modelů umožnil snímání polohy všech kloubů ruky rovnou v 3D souřadnicích. Zatímco omezené možnosti pohybu byly časem odstraněny zavedením bezdrátové technologie, díky své pořizovací ceně a nezbytné kalibraci zůstávají tyto rukavice výsadou specializovaných pracovišť.

Spolu s vývojem datových rukavic se v průběhu 90. let objevují první pokusy

Obrázek 3.1: Vlevo raný prototyp datové rukavice Z^{TM} Glove (Zimmerman, 1982), vprostřed model jednoduché barevné rukavice a vpravo ukázka metody založené na hledání markerů. Zdroj: [1]

o rozpoznávání gest z vizuálních dat. Kvůli nedostatku výpočetního výkonu a vysoké koncentraci obrazového šumu zůstaly však metody segmentace založené na barevném modelu kůže pouze na papíře. Namísto toho se využívalo hledání dobře detekovatelných bodů (tzv. markerů) nebo barevných oblastí (barevné rukavice). První metody tak spoléhaly pouze na polohu těchto klíčových segmentů, přičemž pro celkovou přesnost detektoru byla naprosto nezbytná jejich stálá viditelnost. Až v průběhu času narůstající výpočetní výkon umožnil aplikaci metod klasifikace na holé ruce, například pomocí nalezených hran, siluety, apod. Tak tedy se dlouhodobým cílem stal návrh systému například pro rozpoznávání běžné znakové řeči bez využití pomocných markerů nebo rukavic.

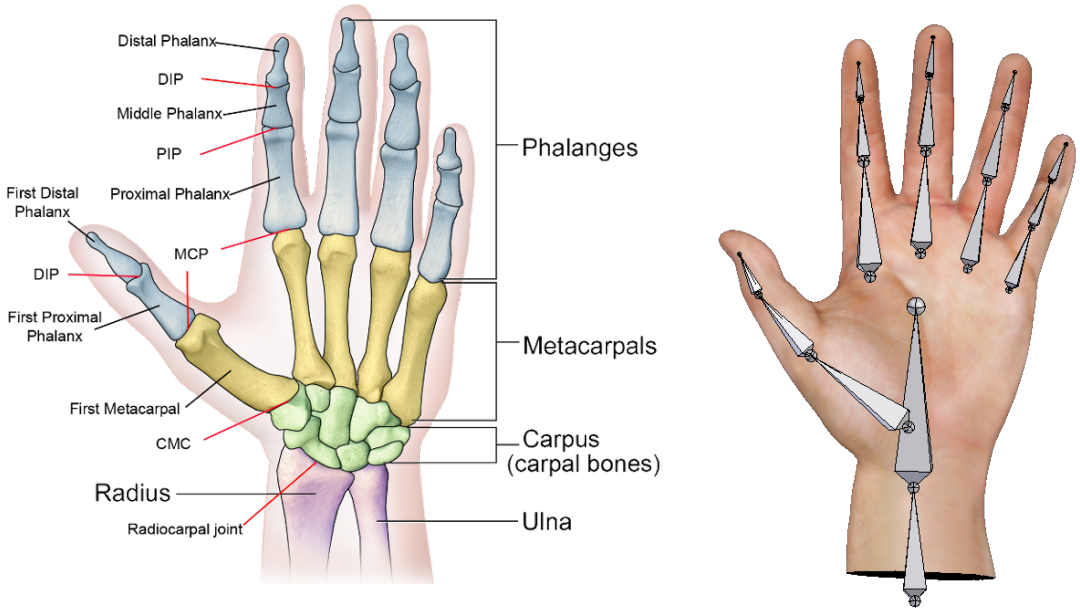
Pro celkovou rekonstrukci pózy však stále nějaká informace chyběla. Pokročilé algoritmy skládající snímky z více kamer sice dokázaly demonstrovat své schopnosti, jenomže jejich výpočetní náročnost zcela vylučovala veškeré ambice pro nasazení systému v reálném čase. Zatímco již existovaly metody zvládající rekonstrukci a tracking v reálném čase za pomoci markerů nebo barevných rukavic, experimenty na holé ruce stále čekaly na určitý průlom. Tím byl až příchod hloubkových snímačů, díky nimž se dramaticky zjednodušila a zrobustnila zejména segmentace, čímž se upustilo od potřeby nošení rukavic nebo markerů. S klesající cenou našly hloubkové snímače cestu i mezi běžné uživatele. Barevná data tak v posledních letech nahradila data hloubková. Snímání pohybu za pomoci markerů v kombinaci s hloubkovými daty je však dodnes považováno za nepřesnější řešení a běžně využíváno například filmovými studii.

4 Rekonstrukce pózy ruky

Úloha rekonstrukce pózy ruky v sobě skrývá hned několik komplikací. Podíváme-li se na fyziologický model ruky, už jen z počtu pohyblivých kůstek lze usoudit, že tvorba věrohodného modelu nebude jednoduchá záležitost. Motorická dokonalost vnitřního uspořádání je totiž vykoupena kompromisy. Vazivová zakončení jsou místy přichycena na více místech, tudíž vědomý pohyb určitou částí ruky s sebou často nese vynucený nebo omezený pohyb částí jinou. Pohyb jednotlivých prstů tedy není tak úplně nezávislý, jak bychom mohli předpokládat. Celková motorická zdatnost je navíc u každého jedince jiná, čímž se proces generalizace modelu ještě zesložiťuje.

Úlohou tvorby věrohodného počítačového 3D modelu se zabývali výzkumníci z Torontské univerzity [2]. Jejich cílem bylo navrhnout takový 3D model ruky, aby maximálně podchytili variabilitu jednotlivých částí a zároveň zachovali validitu vůči reálnému fyziologickému modelu. Jejich zjednodušený model disponuje vůči pevnému bodu v prostoru celkem 27 stupni volnosti (27 DOF). Čtyři stupně volnosti uvažované pro každý prst zahrnují ohyb ve třech kloubech (flexe) a posun celého prstu do strany (abdukce). Pět stupňů volnosti zahrnuje pohyb palce a tři stupně volnosti pohyb zápěstí. Poslední tři stupně volnosti udávají polohu ruky v prostoru. Autoři při tvorbě modelu uvažují nezbytná zjednodušení. Kupříkladu každý článek se může pohybovat samostatně nezávisle na ostatních, jeho rotace/poloha nezávisí na rotaci/poloze zápěstí a zároveň je zanedbán pohyb vnitřních kůstek. Tedy například té části spojující zápěstí a malíček. Posléze autoři za pomoci infračervených kamer a vhodně umístěných markerů nasnímali ruku při hře na kytaru, aby sevřené gesto následně pomocí inverzní kinematiky zrekonstruovali. Naměřená data posloužila pro porovnání přirozenosti mezi zrekonstruovaným a reálným gestem.

Díky vysoké variabilitě pohybu ruky vyvstává další problém, a to překryvy jednotlivých částí. Pro srovnání, při snímání a odhadu pózy těla je úhel mezi tělem a snímacím zařízením většinou fixní. Tedy postava míří čelem ke kameře, přičemž k zákrytu celých částí těla téměř nedochází. Dochází nanejvýš k částečnému překryvu. Naproti tomu umístění kamery pro snímání rukou je variabilní v závislosti na typu úlohy. Zatímco pro rozpoznávání gest je žádoucí tzv. pohled třetí osoby, pro snímání rukou do virtuální reality je nutný pohled z první osoby. Zvláště v druhém jmeno-



Obrázek 4.1: Vlevo skelet reálné ruky a vpravo použitý 27 DOF počítačový model. Zdroj: [3]

vaném případě je problém zákrytu markantní, protože zápěstí většinu času zakrývá polohu prstů. Estimátor si tak doslova musí tipnout polohu neviděných částí. V těchto případech tak může docházet k velké nepřesnosti a nestabilitě rekonstrukce.

Specifikum při rozpoznávání ruky přináší vysoká vzájemná lokální podobnost jednotlivých částí. Kupříkladu pokud bychom srovnali vedle sebe výřez snímku s prostředníkem a prsteníkem, najít klíčový rozdíl, podle kterého bychom tyto dva prsty od sebe odlišili, je nemožné. Tudíž nemůžeme nahlížet na problém rekonstrukce pouze lokálně okénko po okénku, ale potřebujeme zachytit i globální vazby detekovaných částí. Bez nich by rekonstruktor mohl snadno zaměňovat pořadí prstů na ruce a vzniklý výsledek by zcela odporoval fyziologii ruky. Nakonec pro věrohodnou rekonstrukci nesmíme zapomínat na možnost kolize jednotlivých článků nebo jejich fyziologické omezení z hlediska rozsahu pohybu.

4.1 Snímání a segmentace obrazu

Základním předpokladem pro vyřešení úlohy je bezchybná segmentace ruky. Segmentace je proces předzpracování, při němž se snažíme z obrazu vyextrahovat pouze objekt našeho zájmu. Segmentace byla ještě před několika lety nezbytnou součástí většiny úloh počítačového vidění. První větší úspěchy konvolučních neuronových sítí však ukázaly, že mnohdy problematická segmentace již pro většinu úloh rozpoznávání objektů není nezbytná [4]. Zatímco však náš mozek zvládá analyzovat obraz bez explicitního předzpracování, pro účely rekonstrukce pózy je bohužel segmentace

stále nezbytnou součástí, a to kvůli maximálnímu zjednodušení už tak komplikované úlohy.

4.1.1 RGB obraz

V případech, kdy máme k dispozici pouze barevný obraz z tradiční video kamery, se používají standardní metody segmentace počítačového vidění. V drtivé většině případů totiž máme apriorní znalost barvy, tedy například barevného modelu lidské kůže. Úloha se tak zjednodušuje na triviální kombinaci prahování a post-processingu. Následuje lokalizace objektu ruky, například metodou mean-shift, výpočet souřadnic a velikosti nalezeného objektu. Vysegmentovaný snímek pak dále postupuje k dalšímu zpracování. Alternativně lze pro segmentaci využít některou z pokročilých datově orientovaných metod založených na strojovém učení, například metodu Random Forest [5] či konvoluční neuronovou síť [4].

Je důležité zmínit, že podmínkou přesné segmentace je předpoklad barevné odlišitelnosti ruky a pozadí. Tato podmínka pro reálné nasazení bohužel nebývá často splněna, a proto se musí sáhnout k použití nejrůznějších barevných rukavic. Barva těchto rukavic bývá volena tak, aby nekorespondovala s žádnými běžně se vyskytujícími objekty, a tedy ji bylo snadné přesně vysegmentovat z obrazu. Příkladem funkčního a později i komercionalizovaného řešení je pestrobarevná rukavice dvojice Wang a Popović [6]. Nošení rukavice je však pro běžného uživatele velmi nepraktické. Výsledek je navíc stále závislý na intenzitě a úhlu okolního osvětlení. Proto po příchodu finančně dostupných hloubkových snímačů ustoupil zájem o RGB obraz do pozadí.

4.1.2 Hloubkový obraz

První infračervené snímače pracující na principu strukturovaného světla umožnily velice snadno a levně měřit tak chybějící hloubkovou informaci. Princip měření vzdálenosti je založen na infračerveném zářiči, jenž vysílá přes vhodně perforovanou mřížku světelné body, přičemž celou projekci snímá infračervená kamera. Prostorové uspořádání těchto bodů je známé (perforace mřížky), a tudíž lze z nasnímané deformované 2D projekce bodů zrekonstruovat vzdálenostní rozdíly jednotlivých bloků obrazu. Výstupem je hloubková mapa s přesností v řádu centimetrů, ideálně až v řádu milimetrů. Díky popsanému principu závisí rozlišovací schopnost přímo na vzdálenosti objektu. S rostoucí vzdáleností od zářiče rostou vzdálenosti mezi promítanými body, čímž se vytrácí schopnost zachytit drobné objekty nebo jejich texturu. Konkrétně řečeno, zatímco v blízkosti snímače jsme schopni zachytit jednotlivé prsty ruky nebo části lidské tváře, s rostoucí vzdáleností se schopnosti omezují na zachycení polohy zápěstí a hlavy. Ze stejného důvodu dochází mimo jiné k šumu na rovných

hladkých hranách. V neposlední řadě je třeba uvážit materiály, které špatně nebo vůbec neodráží světlo (černá barva). Ty způsobí slepá místa v obrazu, díry chcete-li, a proto je nutné zvážit určitý post-processing hloubkových snímků. Tyto vlastnosti citelně zúžily okruh použití těchto snímačů.

Nastupující nová generace hloubkových snímačů sice tyto neduhy odstranila, objevily se však jiné. Tyto snímače pracují na léty prověřeném a dobře známém principu měření doby letu signálu. Několikrát za sekundu vysílač vyšle světelný puls, přičemž se měří uplynulá doba, v níž vyslaný světelný signál odrazem dorazí zpět na plochu snímače. Z rychlosti světla posléze snadno vypočítáme uraženou vzdálenost. Rozlišovací schopnost v tomto případě závisí podobně jako u běžné video techniky na rozlišení a kvalitě snímače. Spotřební modely využívají typicky jako zářič LED diodu, přičemž k sejmutí scény dochází naráz. To dovoluje mnohonásobně vyšší snímací frekvenci při horším rozlišení, protože celá scéna se musí vejít na plochu snímače. Naproti tomu profesionální hloubkové snímače využívají laserový paprsek plynule plující po scéně. V jednom okamžiku je tak na plochu snímače sejmuta velmi malá část scény, čímž je dosaženo precizní přesnosti na úkor výpočetní náročnosti a snímkovací frekvence. Obecný problém tohoto typu snímačů je chybějící odraz od objektů, kde je úhel dopadu velmi malý, či odraz od reflexních a lesklých materiálů. V těchto případech vznikají v obrazu díry, protože se paprsek světla odrazí jinam. U reflexních a lesklých materiálů navíc může docházet ke vzniku šumu nebo duchů, a to zejména kolem objektu v popředí. Důvodem jsou pravděpodobně bloudící paprsky, které se vlivem špatného odrazu vrací z jiného místa.

Zásadní výhodou hloubkových snímačů je (téměř) naprostá invariance vůči nasvícení scény nebo na barvě pozadí, i když podobně jako u běžné videotechniky i zde stále představuje problém ostré sluneční světlo. Díky doposud chybějící informaci o prostorovém uspořádání jsme schopni snadno a věrohodně vysegmentovat pouze objekt našeho zájmu, tedy v tomto případě postavu člověka, nebo pouze její části. Postava totiž často stojí přímo před kamerou, přičemž má za sebou i pár metrů volného prostoru. Jako naprosto nejjednodušší metoda segmentace se tak nabízí jednoduché prahování. Předpokládáme například, že náš objekt zájmu se nachází v určité vzdálenosti od kamery. Všechna ostatní data přesahující náš interval zahodíme. Toto řešení však postrádá potřebnou robustnost a už vůbec nezaručuje, že vysegmentovaný objekt je postava člověka a ne například stojací lampa, křeslo, atp. Otevřely se tak možnosti pro návrh nových sofistikovaných metod segmentace založených na strojovém učení. Jmenovitě lze opět uvést populární metodu Random Forest [7].

Obrázek 4.2: Ukázka robustní klasifikace hloubkových snímků těla na jednotlivé části metodou Random Forest. Zdroj: [7]

4.2 Klasifikace částí těla

První metody spočívaly na hledání nejlepší shody vstupního snímku s databází [6]. Jejich aplikace pak striktně závisela na velikosti této databáze, respektive schopnosti tuto databázi udržet v paměti a provádět na ní dotazy v rozumném čase alespoň přibližujíc se 25 snímkům za sekundu. Přes veškeré optimalizační techniky byly tyto metody odsouzeny pouze pro jednodušší úlohy, jako například detekce celých paží, nohou, torza, hlavy, nebo pro použití off-line.

Úspěch zaznamenalo až zařízení Microsoft Kinect, k němuž společnost Microsoft dodala software zvládající rekonstrukci zjednodušeného skeletonu těla [7]. Inženýři Microsoftu dokázali jako první přijít s metodou, která zvládla nalézt jednotlivé části těla a zrekonstruovat celkovou pózu z jednoho jediného hloubkového snímku, a to vše v reálném čase. Jako jedni z prvních implementovali v té době už dobře známou metodu Random Forest pro per-pixel segmentaci na hloubkových datech. Protože jde primárně o úlohu strojového učení, bylo třeba nejprve nashromáždit množství anotovaných trénovacích dat. V porovnání s RGB daty je variabilita hloubkových dat menší, čehož autoři využili, aby natrénovali velice robustní systém pouze z uměle vygenerovaných dat. Za pomoci počítačového 3D modelu nasimulovali 15 různých postav lišících se vzájemně stavbou těla (výška, tloušťka). Postavu člověka rozdělili celkem na 31 částí, z nichž v každé se nachází jeden kloub. Pro simulaci reálného pohybu modelů využili existující mo-cap[1] nahrávku. S minimálními náklady tak získali velmi variabilní a přesně anotovaná data. Natrénovaný Random Forest klasifikátor pak v každém snímku určuje hustotu pravděpodobnosti pro každý pixel, tedy kon-

[1]Mo-cap neboli Motion-Capture je systém snímající pohyb objektů ve 3D nejčastěji pomocí setu infračervených kamer a markerů.

krétně s jakou pravděpodobností je daný pixel součástí pozadí, části paže, nohy, atd. Díky dobře škálovatelným výpočtům, kvalitním výsledkům při rozumných paměťových nárocích a relativně snadné implementaci se tato metoda s oblibou používá pro segmentaci a klasifikaci hloubkových dat dodnes.

S popularizací konvolučních neuronových sítí se pozornost soustředila na nové metody. Konvoluční neuronové sítě se sice ukázaly jako nové state-of-the-art pro segmentaci a klasifikaci RGB dat [4], nicméně pro aplikaci na hloubkových datech zdá se kvalitativně nenabídly mnoho nového [8].

4.3 Regrese a tracking

Je-li cílem naší práce klasifikace, výstupem systému bude konkrétní hypotéza ohodnocena určitou mírou jistoty. Tedy máme-li nasbíranou databázi jednotlivých gest ruky, nalezením nejlepší shody mezi testovaným snímkem a databází můžeme rozhodnout, o jaké gesto se jedná. Popřípadě uvážíme-li množinu kandidátů na nejlepší shodu, nabízí se možnost naši hypotézu podpořit hustotou pravděpodobnosti. Tím tedy naše práce končí. Úkolem regrese je však určit parametry úlohy na základě nějakého pozorování, respektive za pomoci navrženého modelu aproximovat neznámý reálný model úlohy. Úlohu samotnou tak lze dělit na část řešící návrh vhodného modelu a na část řešící návrh metody pro odhad parametrů. Tedy konkrétně pro úlohu rekonstrukce ruky je aproximativním modelem zjednodušený model skeletonu ruky s určitým počtem stupňů volnosti, který chceme pomocí neuronové sítě implicitně generalizovat z hloubkových dat. V tomto případě navíc známe i apriorní model úlohy. Ten nám může pomoci zamezit případům, kdy regresor vrací nesmyslné výsledky, tedy například nesmyslné hodnoty rotace odporující fyziologickému modelu.

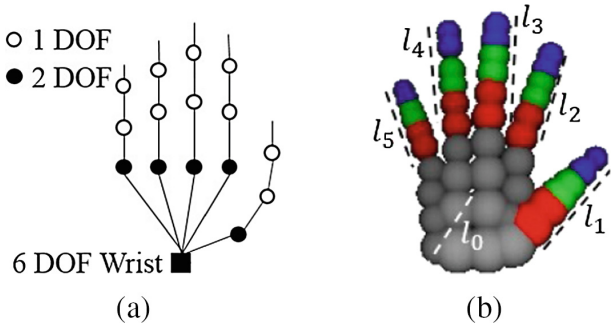
Na naši ruku můžeme nahlížet i jako na obecný dynamický systém, který v průběhu času mění svůj stav. Tvar či polohu ruky nelze měnit nekonečně rychle, tudíž změna stavu systému je vždy spojitá. Tohoto faktu lze pochopitelně využít jak pro účely rekonstrukce, tak i pro účely trackingu ruky. Jestliže známe historii předchozích stavů, lze nový stav velmi věrohodně odhadnout, čímž ulehčíme našemu estimátoru. Tedy namísto toho, aby estimátor v každém kroku prohledával celý stavový prostor, zúžíme mu prostor hledání na minimum, čímž dramaticky zredukujeme výpočetní náročnost úlohy, a umožníme tak nasazení tohoto estimátoru v reálném čase. Této znalosti využívají mnohé současné metody.

5 Současné metody a přístupy

Vývoj v oblasti výpočetní techniky a obrazových snímačů dovolil upustit od datových rukavic, aby se pozornost výzkumníků mohla soustředit na zcela novou disciplínu počítačového vidění. Detekované hrany a silueta ruky představují vysokou dimenzi dat, která vzhledem k variabilitě zkrátka v té době stále není upočítatelná. Největší problém se tak skýtá v hledání dobře detekovatelných unikátních a zároveň stabilních příznaků. Proto nejsnadnějším řešením bylo sáhnout k použití tzv. markerů, tedy objektů (bodů), které by byly v obraze snadno detekovatelné. Všechny tyto metody však pro dosažení přesnosti na složitějších gestech vyžadují viditelnost ideálně všech těchto markerů, což v praxi znamená nasazení zkalibrovaného setu více kamer. I přes značné omezení volnosti v rozsahu pohybu, pohled zblízka a takřka laboratorní podmínky nejsou tyto metody schopny nabídnout zpracování v reálném čase. Obecně se však objevují dva odlišné druhy přístupů k řešení úlohy. Ty začala odborná veřejnost rozdělovat na generativní a diskriminativní. Protože obě metody mají své pro a proti, jako třetí kategorie se nabízí kombinace obou. Tyto metody jsou v praxi označovány jako hybridní.

5.1 Generativní metody

Generativní metody, nebo také často nazývané jako metody modelově založené, jsou postaveny na principu testování hypotézy. Metody vychází částečně z teorie lineárních systémů, tedy že neznámý systém reprezentovaný lidskou rukou se nachází v každém časovém okamžiku vždy v určitém stavu. My se tento systém snažíme aproximovat naším vlastním modelem o určitých parametrech, přičemž máme k dispozici pozorování tohoto reálného systému. Změnou parametrů našeho modelu a následnou simulací se snažíme přiblížit výstup našeho modelu k získanému pozorování. Jedná se de facto o rekonstruktor stavu. V praxi to stručně řečeno znamená, že máme k dispozici snímek ruky z videokamery a obecný počítačový 3D model ruky. Na našem 3D modelu ruky provedeme simulaci odhadnutých parametrů . Následně porovnáme vygenerovaný snímek s reálným snímkem a vyhodnotíme naši hypotézu vyčíslením podobnostní funkce. Pokračujeme tak dlouho, dokud nebude vygenerovaný snímek



Obrázek 5.1: Ukázka zjednodušeného 3D modelu ruky složeného ze základních geometrických útvarů. Podobný model umožňuje daleko rychlejší výpočet per-pixel vzdálenosti (podobnosti) než tradiční polygonový model. Zdroj: [10]

totožný s reálným snímkem.

Už z tohoto popisu je jasné, že v naivním případě bychom museli prohledávat vždy celý stavový prostor. Protože se ale stav naší ruky mění spojitě, můžeme jako startovní bod hledání použít předchozí stav a hledat jen v jeho okolí. V praxi tak tyto metody vyžadují prvotní inicializaci, která probíhá typicky umístěním ruky do určitého stavu a vyčkáním, dokud algoritmus ruku nenalezne. Následně už tracking funguje na základě informace z předchozího snímku. Je tedy patrné, že v momentě, kdy tracking selže a ztratí ruku, už ji nedokáže v reálném čase najít a musíme buďto najet rukou zpět do známého posledního stavu nebo provést inicializaci. Podobně pokud systém vybere špatnou hypotézu, která způsobí zdánlivě malý rozdíl mezi reálným a zrekonstruovaným snímkem (lokální extrém), tracker se už nemusí z tohoto stavu vrátit. Z uvedeného vyplývá, že generativní metody jsou určeny pouze pro tracking na spojitém videozáznamu, nikoliv pro jednotlivé nenavazující snímky.

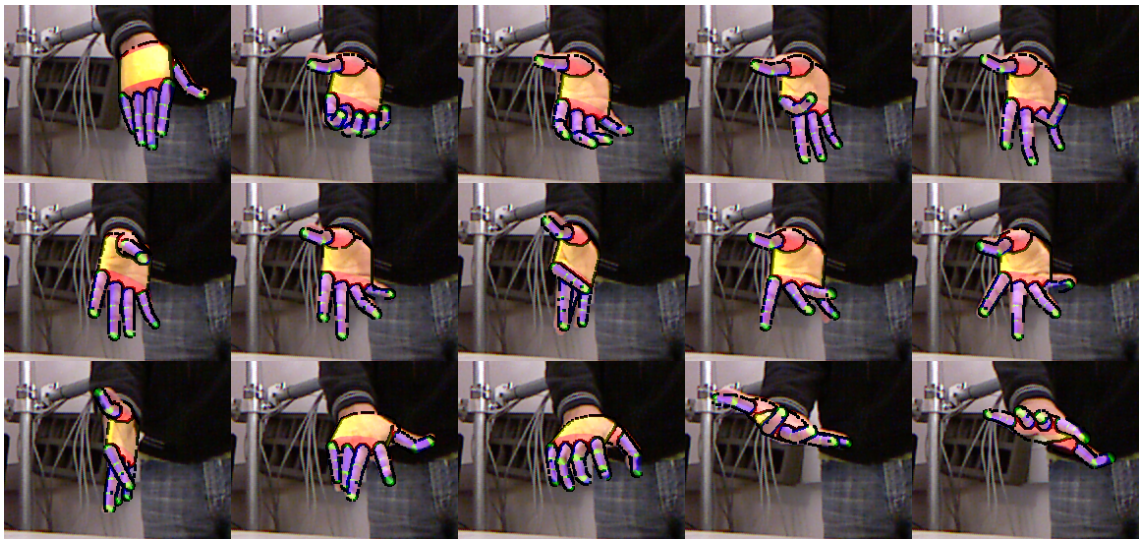
Protože porovnání dvou barevných obrázků je velice komplikované kvůli variabilnímu osvětlení ruky, odstínu barvy kůže, apod., metody vyhodnocující přesnost hypotézy pracují většinou ve 3D. Ještě před nedávnem bylo běžné využití dvou a více videokamer, jejichž přesnou kalibrací byla dosažena rekonstrukce 3D struktury obrazu. Z ní bylo možné číselně vyhodnocovat podobnost reálného pozorování a výstupu simulace v prostorových souřadnicích. Velkou část výpočetní energie však pojmul právě proces 3D rekonstrukce a porovnání obrazu. Tyto metody tak byly odsouzeny pouze pro off-line použití. Hloubkové snímače později 3D snímání výrazně zjednodušily, a tím umožnily větší soustředění výpočetního výkonu na vyřešení samotné úlohy.

S významnou metodou využívající princip Particle Swarm Optimization (PSO) pro hledání správných hypotéz parametrů z hloubkového obrazu přišli řečtí výzkumníci [9]. Úlohu rekonstrukce/trackingu pózy definují jako problém optimalizace, respektive minimalizace rozdílu mezi reálným a vygenerovaným snímkem ruky (ztrá-

tová funkce). Jako jedni z prvních tak navrhli přesnou metodu, která pracuje s hloubkovými snímky holé ruky bez potřeby nošení markerů či rukavic. Algoritmus funguje nezávisle na úrovni osvětlení scény a díky implementaci na grafické kartě dosahuje snímkovací frekvence 15 snímků za sekundu. Jejich 3D model je definován 27 parametry, přičemž namísto přirozeného polygonového modelu použili model složený z jednoduchých geometrických útvarů (koule, válce, elipsoidy). Důvodem je zjednodušení a zrychlení výpočtů. Pro nalezení minima ztrátové funkce modifikovali metodu PSO. Jednotlivé částice reprezentují celou pózu, respektive stav ruky definovaný 27 parametry, přičemž velikosti dílčích parametrů jsou omezeny vzhledem k fyziologickým omezením ruky (apriorní model). Obecně jsou u PSO na začátku algoritmu částice umístěny náhodně po celém stavovém prostoru. V tomto případě, jak již bylo uvedeno, jsou částice rozmístěny pouze v okolí inicializačního stavu. Následně, zjednodušeně řečeno, je v každém iteračním kroku pro každou částici vyhodnocena ztrátová funkce, přičemž společným cílem částic je pohybovat se směrem k těm částicím, které si vedou nejlépe. Ty totiž reprezentují hypotézu, respektive pózu, která nejvíce odpovídá získanému pozorování. Ve své podstatě je tak algoritmus velice triviální. Nejnáročnější část algoritmu tvoří vyhodnocení ztrátové funkce. Protože je však tento výpočet pro každou částici nezávislý, lze jej výhodně paralelizovat s využitím například grafické karty, jež je vybavena výrazně větším počtem výpočetních jednotek oproti běžnému procesoru. Na této metodě dále staví spousta dalších prací, v nichž se pracnou optimalizací algoritmu podařilo dosáhnout i více jak 30 snímků za sekundu.

Nabízí se otázka volby proporcí 3D modelu ruky. Napříč populací nalezneme ruce různé velikosti a tloušťky. Zřejmě nejmarkantnější rozdíl je mezi dospělými a dětmi. Bylo by tedy vhodné mít možnost adaptivně volit proporce modelu. To pochopitelně zvyšuje náročnost úlohy v podobě dalších parametrů, nicméně jak ukázali výzkumníci z univerzity v San Diegu [10], použitím aditivního modelu lze dosáhnout snímkovací frekvence 16 snímků za sekundu. Je ale důležité zmínit, že jejich algoritmus běží pouze na čtyřjádrovém procesoru. Lze tak předpokládat, že vhodnou implementací kódu na grafické kartě by bylo dosaženo ještě vyšší snímkovací frekvence.

Jak již bylo zmíněno, nejnáročnější částí algoritmu je vyhodnocení podobnosti mezi snímky, přičemž poté vždy následuje rozptyl polohy horších částic do okolí těch nejlepších částic. Kdybychom lépe věděli, kudy se mají částice vydat, místo abychom jejich polohu náhodně rozptýlili, dosáhli bychom nezanedbatelné výpočetní úspory. Už jen třeba proto, že bychom si vystačili s menším počtem částic. Z jiných optimalizačních úloh dobře známe metodu gradient descent, která neznámé parametry vždy tlačí směrem proti gradientu chyby. Problémem zde je však nespojitost našeho pozorování ve 3D v kombinaci s šumem a s překrývajícími se částmi ruky. Navíc

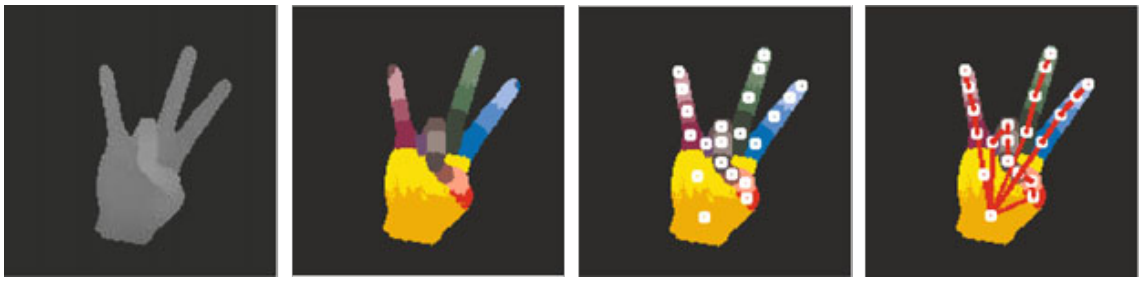


Obrázek 5.2: Ukázka rekonstrukce (tzv. nafitovaní) 3D modelu ruky metodou PSO. Zdroj: [9]

gradient descent má mnohdy tendenci uvíznout v lokálním optimu, tedy funguje výborně, pouze pokud je inicializován blízko výsledku. Zajímavý přístup tak zvolila skupina německých a finských výzkumníků [11]. Pro popis reálného snímku a výstupu z modelu použili směs Gaussovských funkcí. Konkrétně hloubkový snímek rozdělili na homogenní oblasti a každou z nich popsali Gaussovskou směsí. Získali tak hladký diferencovatelný popis snímku. Stále se jedná o variantu algoritmu PSO, přičemž pro vyhodnocení podobnosti používají komplexní výpočet energie (podobnost Gaussovských funkcí, jejich projekce v hloubce, kolize, apod.). Jejich cílem je nalézt takovou pózu, která minimalizuje tuto energii, přičemž v tomto případě je její výpočet snadno diferencovatelný. Zároveň kombinují informaci z Random Forest klasifikátoru, který slouží jako robustní reinicializátor, aby algoritmus neuvízl v lokálním minimu.

5.2 Diskriminativní metody

Diskriminativní metody, jinak řečeno metody založené na vzhledu, dokáží vyřešit úlohu nalezením přímého mapování vstupních dat na výstupní parametry. Nevytváříme tedy žádný model explicitně apriori, nýbrž naším cílem je implicitní generalizace modelu na základě předložených trénovacích dat. Jedná se primárně o metodu založenou na strojovém učení s učitelem, přičemž pro natrénování modelu je typicky zapotřebí nemalé množství dat. Ve fázi běhu se poté systému předloží na vstup snímek, ten se zpracuje algoritmem a na výstupu modelu se objeví přímo hledané parametry. Metoda není nijak závislá na předchozím stavu systému. Aby byl však systém dostatečně robustní, je v závislosti na požadavcích úlohy nutno sesbírat a



Obrázek 5.3: Ukázka klasifikace ruky z hloubkového snímku na jednotlivé části. Pomocí metody mean-shift je posléze nalezen střed oblasti a prohlášen za pozici kloubu. Můžeme pozorovat, že nalezené středy oblastí spíše než s reálnou polohou kloubů korespondují s polohou jednotlivých kostí. Zdroj: [12]

zpracovat desetitisíce až miliony snímků, přičemž během trénování musíme dbát na generalizaci modelu, abychom předešli snadnému přetrénování. Z důvodu výpočetních nároků a nedostatku trénovacích dat se tento typ metod dostal do popředí až nedávno s uvedením vysoce paralelních ryze diskriminativních metod, jako jsou například Random Forest a konvoluční neuronové sítě. Ty nahradily dřívější databázově orientované metody (např. kNN).

První pokusy o vytvoření systému na způsob databáze se objevují už v raných začátcích počítačového vidění. Algoritmus ve fázi běhu hledá nejlepší shodu mezi vstupním snímkem a snímky v databázi. Klíčem k upočítatelnosti je velikost databáze a unikátní, ba stručný popis obrazu. Problém s nalezením unikátních stabilních příznaků byl dlouho efektivně řešen použitím barevné rukavice. Tým Wang a Popović [6] přišel s originálním grafickým návrhem textury rukavice, kde jednotlivé segmenty ruky mají vždy danou barevnou kombinaci vzhledem k sousedním polím. Rukavici autoři rozdělili celkem na dvacet polí, jež jsou vybarveny jednou z deseti unikátních barev. Mezi barvami jsou jasné a syté odstíny, aby se maximálně zjednodušila jejich segmentace. Posléze je vysegmentovaný snímek ruky zmenšen na 40x40 pixelů a prohledána databáze za účelem nalezení nejlepší per-pixel shody. Vzhledem k velikosti databáze čítající 100 tisíc snímků nebylo možné hledání shody brutální silou, proto implementují aproximační techniku k nalezení nejlepších kandidátů. Až mezi nimi je hledána konečná shoda tradičním způsobem. Nalezený výsledek je poté podroben inverzní kinematice, čímž dojde k plné rekonstrukci 3D pózy a k vyhlazení jitteru[1]. Zároveň autoři diskutují kompromis mezi velikostí databáze a možnostmi rekonstrukce. Tedy jinak řečeno, pózy, které nejsou v databázi, nejsme schopni zrekonstruovat a naopak bohatou databázi nejsme schopni prohledat v reálném čase, nebo vůbec nahrát do paměti.

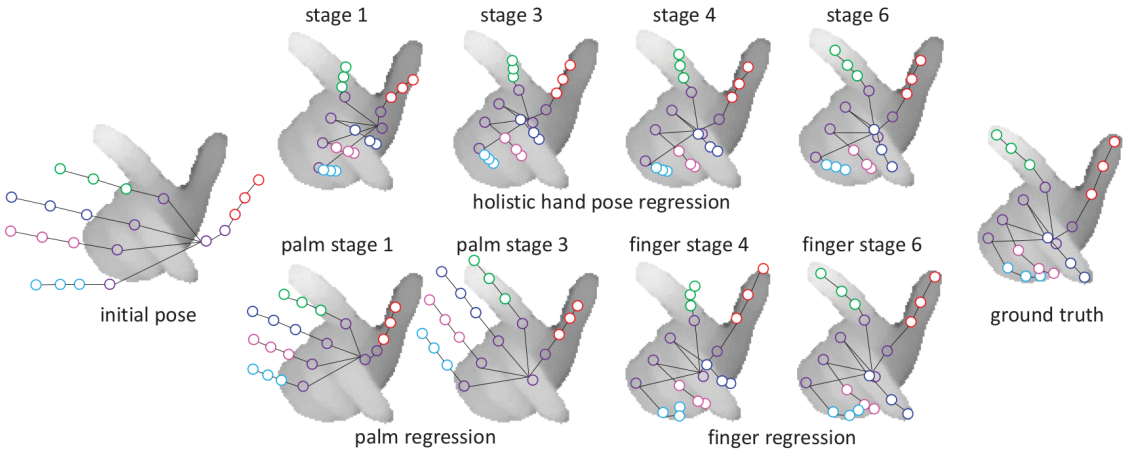
Jak rostly nároky na velikost databáze, vyvíjely se současně další algoritmy pro

[1]Jako jitter se zde označuje třepetání či chvění částí ruky způsobené například šumem.

zrychlení prohledávání. Vstupní dotaz byl mnohdy podroben sérii diskriminativních binárních rozhodnutí na způsob rozhodovacího stromu, aby došlo k přibližné lokalizaci shluku potencionálních kandidátů na nejlepší shodu. Zatímco zprvu našly uplatnění diskriminativní metody pouze jako jednoduchá pomocná informace generativní metodě o orientaci ruky, poloze palce či špiček prstů, později po demonstraci per-pixel schopností metody Random Forest [5] a možností efektivní paralelní implementace [7] se pozornost uchýlila právě k těmto metodám. Na úspěchy per-pixel klasifikace na hloubkových datech [7] navázali turečtí výzkumníci [12]. Ti aplikovali analogický postup pro rekonstrukci pózy ruky. Nejprve vygenerovali množství umělých trénovacích snímků, na nichž nechali natrénovat Random Forest klasifikátor. Každému pixelu snímku byla přiřazena hustota pravděpodobnosti vyjadřující příslušnost daného pixelu ke konkrétní části ruky. Pro 21 různých oblastí tak získaly 21 různých map, přičemž velikost hodnoty udává velikost pravděpodobnosti, že daný pixel patří do konkrétní oblasti. Metodou mean-shift nakonec lokalizovali v každé mapě shluk, jehož střed prohlásili za polohu kloubu.

Možnosti rekonstrukce se odvíjí od variability podchycené v trénovacích datech. Bohužel s narůstajícím trénovacím datasetem rostou komplikace s trénováním Random Forest klasifikátoru. Větší klasifikátor spotřebuje více prostředků a narůstá pravděpodobnost chybné klasifikace. Ve své další práci tak autoři [13] implementují sadů expertů, kteří mají rozhodnout, jaký klasifikátor použít pro rekonstrukci konkrétního snímku na vstupu. Používají tedy množinu Random Forest klasifikátorů, přičemž každý z nich se hodí pro jiný typ gesta. Jejich trénování je tak jednodušší a ve finále mohou běžet paralelně nezávisle na sobě, přičemž jejich výsledek bude na konci váženě zprůměrován (bagging).

O zachycení hierarchie ruky se pokusili výzkumníci z londýnské univerzity [14]. Jejich metoda nazvaná Latent Regression Forest provádí inferenci pózy od středu ruky (těžiště) směrem ven, přičemž ruka je hierarchicky dekomponována na oblasti s respektem k obecnému modelu ruky (pořadí částí). Podobnou kaskádní regresi představil tým složený z výzkumníků Microsoftu a čínských univerzit [15]. Pro inferenci používají množinu Random Forest klasifikátorů sériově v daném pořadí. Tedy nejprve dojde k inferenci polohy zápěstí, poté palce, prvních kloubů a nakonec článků prstů. Jak diskutují autoři práce, poněvadž si jednotlivé lesy předávají (mezi)informaci z předchozího kroku v sérii mezi sebou, je jejich trénování podstatně jednodušší a proces inference rychlejší, nežli použití monolitické architektury. Na druhou stranu zde ale hrozí riziko akumulace chyby. Udělá-li tedy první regresor vážnější chybu, další odhady v pořadí v sobě kumulují tuto chybu. Oproti dosavadním diskriminativním metodám tato kaskádní metoda [15] umožnuje rekonstruovat zcela volný pohyb ruky včetně velkých rozsahů rotace zápěstí. To vše při pozoruhodné snímkovací frekvenci přesahující 300 snímků za sekundu na jednom vlákně



Obrázek 5.4: Ukázka kaskádní rekonstrukce pózy. Zdroj: [15]

procesoru (bez inverzní kinematiky).

Úspěch konvolučních neuronových sítí znamenal průlom pro celou oblast umělé inteligence. Podobně jako Random Forest pracují konvoluční sítě s nepředzpracovaným obrazem (tzv. per-pixel), přičemž jejich grafová struktura je jako stvořená pro masivní paralelizaci. První aplikace konvolučních neuronových sítí pro úlohu rekonstrukce pózy týmem z newyorské univerzity [16] se principiálně velice podobá prvním výše zmíněným Random Forest metodám. Autoři zvolili tzv. multi-scale konvoluční architekturu. Vstupní snímek je transformován na další dva snímky vždy o poloviční velikosti (96, 48 a 24 pixelů). Díky tomu lze stejnou velikostí konvolučního okénka zachytit jinak velikou oblast, a tak pochytit včetně detailních lokálních vazeb i vazby globální. Podle závěru se tato varianta oproti monolitické architektuře liší jak v dosažené přesnosti, tak v době potřebné pro natrénování. Na vstup sítě je vždy přiveden vysegmentovaný snímek ruky. Konvoluční část sítě extrahuje příznaky a fully-connected část provede mapování extrahovaných příznaků do 14 map, které podobně jako u Random Forest klasifikátoru reprezentují pravděpodobnost pozice kloubu, respektive segmentu ruky. Z 2D map je extrahováno celkem 14 bodů v místech s maximální hodnotou. Následně je 2D bodům přiřazena hloubka na základě vstupního snímku. Rozdílem je, že autoři nehledají přímo pozici kloubů, nýbrž jakési pomocné body (cca. středy kostí), pomocí nichž je v další fázi zrekonstruována póza (IK + PSO). Přesnost polohy konečných 3D bodů je velmi limitována velikostí 2D mapy (18x18 pixelů). Zároveň může docházet k přiřazení stejné hloubky dvěma překrývajícím se bodům. Metoda běží implementovaná na grafické kartě včetně inverzní kinematiky okolo 30 snímků za sekundu.

Obě výše zmíněné nevýhody odstraňují další práce [17] a [18]. Zde se na výstupu sítě objevuje rovnou 3D souřadnice jednotlivých kloubů. V obou pracích došli autoři shodně k závěru, že multi-scale architektura pracuje lépe než monolitická a že pozi-

tivní vliv na výsledek má omezení variance výstupu aplikací tzv. hrdla, tedy zúžení počtu neuronů před výstupní vrstvou. Efekt zúžení je principiálně podobný metodě Principal Component Analysis (PCA). Víme totiž, že parametry modelu ruky vzájemně velmi korelují a díky omezení variance výstupu lze předcházet outlierům.

5.3 Hybridní metody

Hybridní metody kombinují to nejlepší z generativních a diskriminativních metod. Generativní metody jsou velice přesné a robustní. Na druhou stranu vyžadují počáteční inicializaci a i rychlejší pohyb může způsobit selhání trackingu, z něhož se už nedokáží vzpamatovat. Oproti tomu diskriminativní metody jsou schopny ruku velice rychle nalézt kdekoliv v obraze a odhadnout její pózu nezávisle na předchozím snímku. Odhad však mnohdy ignoruje detaily, tíhne k přetrénování a pózy, které se v trénovací sadě nikdy neobjevily, nedokáže zrekonstruovat. Nabízí se zde tedy symbióza obou metod. Zatímco diskriminativní metody dokáží velice rychle odhadnout gesto řekněme z 95% správně, generativní metoda využije odhad jako inicializační stav a dokonverguje k výsledku. Hybridní metoda tak využije benefity obou metod – rychlost, robustnost a přesnost.

V nedávné práci výzkumníci Microsoftu [19] představili velice robustní hybridní systém pro rekonstrukci pózy nezávislý na poloze kamery. Jiné systémy totiž většinou předpokládají, že uživatel sedí přímo před kamerou. Rozsah pohybu je díky tomu značně omezen. Metoda publikovaná v [19] si však poradí s nejrůznějšími úhly pohledu (zpředu, ze země, přes rameno aj.) a to právě díky kombinaci robustního diskriminativního reinicializátoru (Random Forest) a následné generativní metody (PSO).

6 Neuronové sítě

Neuronové sítě zaznamenaly v posledních zhruba pěti letech ohromný boom. Principy známé často i několik desetiletí prošly renesancí. Za tím vším stojí jak vývoj výpočetní techniky, tak stále narůstající množství dostupných digitálních dat. Pro účely této práce byla jako rekonstruktor zvolena právě neuronová sít. V době, kdy jsme začali plánovat vznik této práce, nebyla uveřejněna žádná publikace aplikující konvoluční neuronovou síť za účelem rekonstrukce ruky. Rozhodli jsme se proto tyto možnosti prozkoumat. Krátce na to byla uveřejněna první práce [16] potvrzující naše předpoklady, že konvoluční neuronová síť může být mimo klasifikace využita i pro regresi tak komplikovaného a variabilního objektu, jakým je lidská ruka.

6.1 Historie neuronových sítí

Historie umělých neuronových sítí sahá už do počátku 20. století. Spolu s biologickým výzkumem mozku bylo vyvíjeno úsilí aplikovat získané poznatky v praxi. Začátkem 40. let neurofyziolog Warren McCulloch a matematik Walter Pitts publikovali práci o fungování neuronů [20]. V ní modelují neuron pomocí jednoduchého elektrického obvodu. Tento model váženě sčítá výstupy ostatních neuronů. Pokud výsledná hodnota přesáhne určitý práh, dojde k aktivaci neuronu a k vyslání elektrického pulzu. Kombinace těchto neuronů tak mohla provádět například základní logické operace AND, OR nebo jejich negované varianty.

$$f(x) = \begin{cases} 1, & \text{pokud } \omega \cdot x + b > 0 \\ 0, & \text{jinak} \end{cases} \tag{6.1}$$

Na konci 40. let psycholog Donald Hebb publikuje své závěry popisující dopad procesu učení na neuronové struktury v mozku [21]. Jeho hypotéza, že ve fázi učení vznikají nové nebo jsou posilovány existující synapse korespondujících neuronů, měla zásadní dopad na další výzkum v tomto oboru. Vzniká tak úplně první pravidlo aplikované pro učení umělých neuronových sítí. Koncem 50. let je vyvinut první veřejně úspěšný neuro-počítač Mark I Perceptron, jehož úkolem bylo rozpoznávání příznaků. Neuro-počítač je tvořen perceptrony, tedy novými umělými modely neuronu, za ni-

miž stojí Frank Rosenblatt [22]. Mark I Perceptron jako lineární systém tvořen pouze dvěma vrstvami perceptronů bohužel dokázal řešit pouze lineárně separabilní úlohy.

V těchto letech se tak odborná veřejnost začíná dělit dvěma směry. Zatímco jedni oslavují objevení ultimátního klíče k umělé inteligenci a předpovídají fantastické věci, druzí poukazují na zřetelné nedostatky perceptronů, jako neschopnost řešit lineárně neseparabilní úlohu nebo modelovat logickou funkci XOR. Skeptici přirovnávají dosavadní vývoj ke konci slepé uličky. Problémem je totiž neexistence algoritmu učení, který by zvládl natrénovat vícevrstvou architekturu. Vlna zájmu opadá a pozornost se soustřeďuje primárně na budoucí (dnešní) mikropočítače. Následují léta útlumu, přičemž zlomek výzkumníků ve své práci pokračuje. Právě oni pokládají základy budoucí renesanci. Až po několika desetiletích se nakonec ukazuje, že prvotní koncept byl správný. V roce 1975 Paul Werbos představuje svůj algoritmus později nazývaný backpropagation fungující na principu optimalizační metody gradient descent, přičemž klíčem k řešení je použití spojitých diferencovatelných aktivačních funkcí (sigmoida, hyperbolický tangens). Konečně tak lze natrénovat vícevrstvé architektury schopné řešit XOR problém i problém nelineárně separabilních tříd.

Bohužel proces trénování trvá velmi dlouhou dobu, natrénovat výrazně hlubší architektury je takřka nemožné, a tak se pozornosti neuronovým sítím stále nedostává. Reálným aplikacím stále vévodí statistické a diskriminativní algoritmy strojového učení (Bayes, SVM, aj.). Relativně nedávné úspěchy v rozpoznávání číslic na datasetu MNIST [23] byly jedním z prvních signálů, že si konvoluční neuronové sítě zaslouží pozornost. V roce 2012 výzkumníci z univerzity v Torontu publikují pozoruhodné výsledky klasifikace na veřejném datasetu LSVRC-2010 ImageNet pomocí konvoluční neuronové sítě [24]. Do té doby nejlepší dosažené výsledky dokázali předčit o přibližně 10 procent, respektive chybu snížili o více jak třetinu. Zároveň přestavili nový typ aktivační funkce, díky níž si mohli dovolit mnohem hlubší architekturu. Pootevřeli tak dveře do zcela nového oblasti výzkumu nazvané Deep Learning.

6.2 Trénování neuronových sítí

Stejně jako jiné metody strojového učení neuronové sítě umožňují dvě varianty trénování – s učitelem a bez učitele. Metody bez učitele se ve většině případů snaží naučit schopnost odlišit mezi dvěma nepodobnými vzorky a mezi dvěma podobnými vzorky na základě nějaké podobnostní míry. Naopak metody trénování s učitelem mají informaci o podobnosti k dispozici od učitele, přičemž se v rámci trénování snaží nalézt taková rozhodovací pravidla, která minimalizují chybu rozhodování. V případě

regrese nehledáme binární rozhodovací pravidla, nýbrž systém se učí interpolovat mezi trénovacími daty.

6.2.1 Backpropagation

Naprosto nejrozšířenějším algoritmem pro trénování dopředných neuronových sítí metodou s učitelem je algoritmus backpropagation, jinak známý jako algoritmus zpětného šíření chyby. Základní princip spočívá v propagaci chyby z výstupu nazpět sítí, přičemž jednotlivé váhy jsou aktualizovány v závislosti, jakou mírou se podílely na vzniku této chyby.

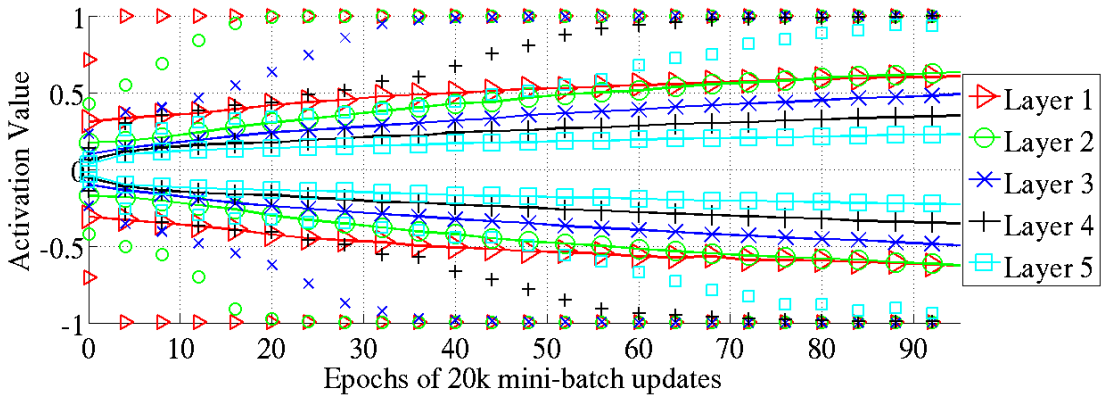
$$E = \frac{1}{2} \sum_i (f(x_i) - y_i)^2 \tag{6.2}$$

$$w'_{ij} = w_{ij} - c \cdot \frac{\delta E}{\delta w_{ij}} \tag{6.3}$$

Rovnice 6.2 a 6.3 popisují algoritmus backpropagation pro případ, kde je jako ztrátová funkce zvolena suma nejmenších čtverců (zjednodušená L2 norma). Celková velikost chyby E je dána sumou rozdílu aktivační hodnoty neuronu $f(x_i)$ a očekávané hodnoty y_i. Výsledná suma je podělena dvěma, abychom po zderivování obdrželi jednoduší tvar. Nakonec je pro každou váhu vypočten gradient chyby, tedy podíl jakým se tato váha podepsala na výsledné chybě. Velikost gradientu je zmenšena přenásobením konstantou učení $c < 1$, čímž dojde k útlumu dlouhých skoků a oscilací. Váha je upravena na novou hodnotu w'_{ij} a algoritmus pokračuje skrz všechny skryté vrstvy sítě směrem od výstupu ke vstupu. Jednoduchým zřetězením výpočtů tak docílíme průchodu sítí, který se podobá tomu dopřednému.

6.2.2 Problémy trénování neuronových sítí

Přestože výzkumníci měli k dispozici trénovací algoritmus backpropagation, který se aktivně používá dodnes, nedokázali efektivně trénovat hlubší architektury sítě, které by byly schopné řešit komplexní a silně nelineární úlohy. Sítě tak spíše rostly do šířky, přičemž se svou funkcí velmi podobaly kernelovým metodám. Velmi širokou a mělkou architekturu s jednou skrytou nelineární vrstvou následovanou lineární vrstvou (RBF síť) můžeme totiž principiálně přirovnat k metodě Support Vector Machines [25]. Právě s touto metodou byly neuronové sítě nejčastěji porovnávány, přičemž metoda SVM se přímočaře trénovala a oproti neuronovým sítím dokázala skvěle generalizovat. Na druhé straně však neuronové sítě dokázaly daleko rychleji rozhodnout, poněvadž průchod takovouto sítí lze zobecnit na pouhé násobení vektorů.



Obrázek 6.1: Ukázka jevu zvaného gradient vanishing. Obrázek ilustruje saturaci aktivačních funkcí v jednotlivých vrstvách neuronové sítě. Můžeme pozorovat, že saturace, respektive útlum gradientu, směrem ke vstupní vrstvě způsobí, že tyto neurony téměř nebudou měnit své váhy. Dochází tedy k trénování pouze hlubších vrstev. Zdroj: [26]

Jak se později výzkumníci shodli, problém s trénováním hlubších architektur je způsoben nevhodnou inicializací sítě. Na začátku trénování musí být totiž váhy jednotlivých neuronů inicializovány náhodně kvůli rozbití symetrie. Typicky se volí malá náhodná čísla z normálního rozdělení. Podíváme-li se na tvar používaných aktivačních funkcí (sigmoida, tangh), tyto funkce mají oboustranně omezený obor hodnot. Tedy ve své kladné respektive záporné polorovině asymptoticky konvergují vždy k reálné hodnotě. Zde se tak pro velká kladná a záporná čísla jejich derivace přibližuje k nule. Pro backpropagation to pak znamená, že je-li chyba na výstupu veliká, postupným šířením chyby sítí nazpět dochází k postupnému úplnému útlumu gradientu [26]. Zatímco tedy vrstvy blízké té výstupní mají snahu své váhy aktualizovat, vrstvy blízké té vstupní své váhy téměř vůbec nemění. Částečným řešením je použití velmi malé konstanty učení, to však vede k rychlé konvergenci do lokálního extrému. Lepším řešením je zvolit takový způsob inicializace vah, aby k tomuto jevu na začátku trénování nedocházelo.

$$f(x) = max(0, x) \tag{6.4}$$

Jedním ze způsobů je tzv. předtrénování sítě. Nejprve se provede trénování neuronové sítě metodou bez učitele. Jednotlivé neurony se samostatně naučí extrahovat informativní příznaky. V druhé fázi se síť inicializovaná předtrénováním dotrénuje již standardně metodou trénování s učitelem. Takto dosažené výsledky předčily například tehdejší state-of-the-art v rozpoznávání číslic na datasetu MNIST [23]. Další variantou je uvážení jiného tvaru aktivační funkce. Například aktivační funkce Rectified Linear Unit (ReLU), kterou popisuje vztah 6.4, byla použita v úspěšné architektuře AlexNet [24], kde díky své lineární části dokáže předcházet útlumu gradientu.

Nezávisle na velikosti operandu aktivační funkce je její derivace buďto nulová nebo konstantní nenulová. Mezi oblíbené techniky patří rovněž využití některé z inicializačních metod [26] nebo [27], které respektují počet vstupů a výstupů neuronu.

6.2.3 Optimalizace trénování neuronových sítí

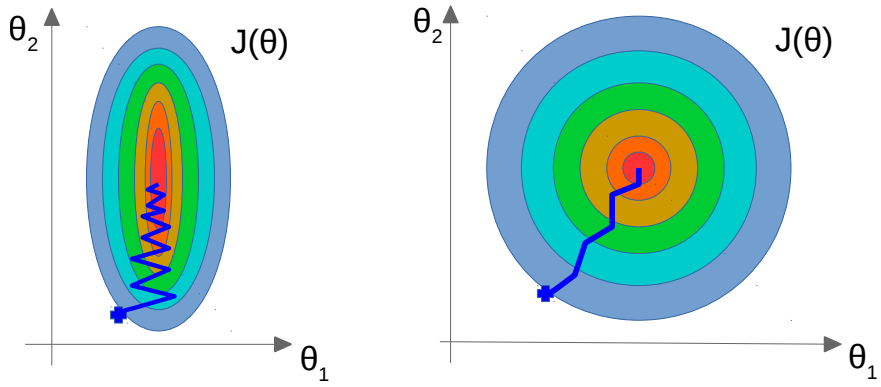
Jak bylo zmíněno na případu výše, použití pravidla backpropagation vůbec nezaručuje konvergenci ke globálnímu optimu. Nicméně existuje několik obecně doporučených postupů a rad [28], jak zlepšit svou šanci na natrénování neuronové sítě.

Dávkové trénování

K aktualizaci vah neuronů nedochází pro jednotlivé trénovací vzorky, nýbrž dávkově (full-batch). Síť tedy zpracuje typicky všechny trénovací vzorky naráz, chybu zprůměruje a poté dochází k aktualizaci vah. Díky průměrování lze zamezit vzniku oscilací, tedy takových případů, kdy jeden snímek nutí síť aktualizovat váhy jedním směrem a další snímek směrem úplně jiným. Protože jsou v dávce obsaženy všechny trénovací vzorky, aktualizace vah probíhá tak, aby došlo k uspokojení všech, respektive k minimalizaci chyby v rámci celého datasetu. Dávkovým trénováním lze rovněž dramaticky snížit výpočetní nároky. To ale platí pouze pro velmi omezenou velikost datasetu (typicky < 10000 vzorků).

Vzhledem ke své kapacitě mají neuronové sítě silnou tendenci k přetrénování, a proto je zapotřebí k natrénování modelu řádově daleko větší množství dat. Namísto tradičního full-batch trénování se tak trénovací dataset rozdělí na menší dávky (mini-batch), přičemž velikost dávky se volí v závislosti na úloze tak, aby byl nalezen ideální poměr mezi dobou trénování, paměťovými nároky a poklesem chyby. Oscilacím vah předejdeme, když zajistíme, že informační bohatost celého datasetu bude podchycena v každé trénovací dávce. Například pro klasifikaci do tříd je vhodné, aby každá dávka obsahovala vzorky od každé třídy. V praxi je tedy většinou dataset uniformě zamíchán a až poté rozdělen na malé dávky. Na tomto principu funguje dnes nejrozšířenější metoda Stochastic Gradient Descent (SGD).

Výskyt oscilací nemusí vždy nutně znamenat špatnou věc [28]. Díky nim se lze například vyhnout lokálním minimům či vyskočit z koridoru. Trénování s menší dávkou je zároveň daleko rychlejší, protože síť musí zpracovat v každé iteraci diametrálně menší množství dat. Na druhou stranu, aby síť viděla všechna data datasetu, musí proběhnout daleko více iterací. Nevýhodou oscilací nadále zůstává neschopnost dokonvergovat do lokálního (popř. globálního) minima. Proto je v závěru trénování (finetunning) vhodné dávku zvětšit v kombinaci s menší konstantou učení.



Obrázek 6.2: 2D projekce parabolického error surface $J(\theta)$ pro různě veliké parametry θ_1 a θ_2. Vlevo můžeme pozorovat, že kvůli rozdílné škále obou parametrů potrvá konvergence delší dobu, než-li v případě, kdy jsou parametry normalizovány na jednotné měřítko.

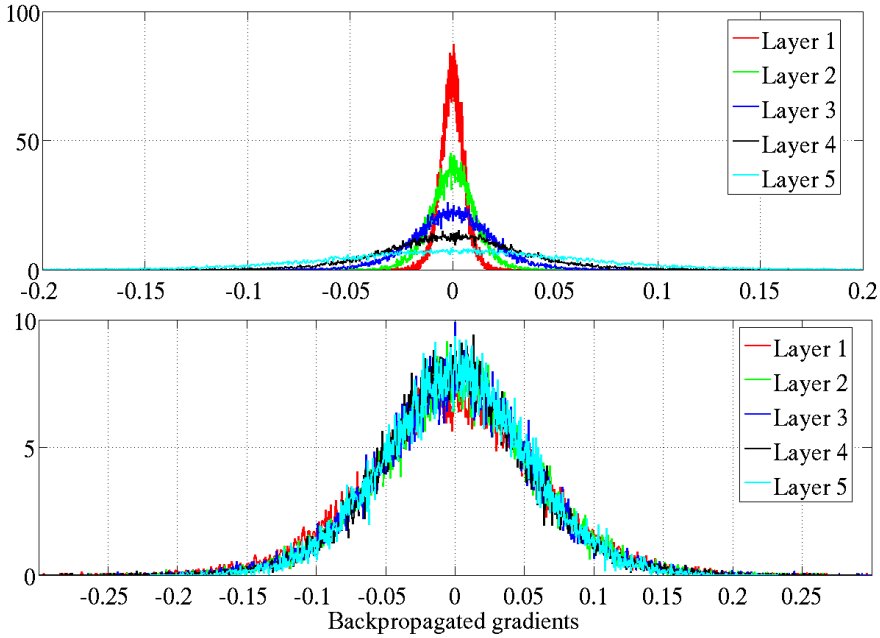
Normalizace a dekorelace vstupu

Z ostatních metod strojového učení víme, že u gradientních metod normalizace vstupu výrazně pomáhá konvergenci chyby k jejímu minimu. Pokud si představíme vícedimenzionální konvexní problém, přičemž jednotlivé dimenze budou obsahovat data o různých škálách, vzniklý error surface způsobí, že směr gradientu chyby bude více záležet na datech o větší škále. Například bude-li jedna složka vektoru v řádu tisíců a ostatní složky v řádu jednotek či zlomků, rozhodující slovo bude mít vždy výrazně větší složka. Je tedy vhodné data normalizovat způsobem, že jejich střední hodnota bude nulová a jejich kovariance jednotková, či alespoň na diagonále konstantní.

V případě, kdy máme na vstupu nekorelovaná data, hodnota váhy w_1 minimalizuje chybu nezávisle na hodnotě váhy w_2 [28]. V opačném případě je v hodnotách vah í-tého neuronu zakódovaná vzájemná závislost, která musí být během trénování naučena. To zbytečně komplikuje řešení problému. Pro výše uvedený případ s normalizací škály to navíc znamená, že ostatní váhy mají daleko menší důležitost a nic moc s problémem nezmůžou oproti jedné řádově důležitější váze.

Inicializace vah neuronů

Jak již bylo zmíněno u jevu útlumu gradientu v předchozí podkapitole, velice důležitým krokem u hlubších architektur je prvotní inicializace vah sítě. V práci [26], kde se autoři zabývají příčinou útlumu gradientu, nalezneme doporučenou inicializační metodu později označovanou jako Xavier. Ta nám říká, jak volit velikost náhodných čísel tak, aby byla zachována konstantní vstupně výstupní variance nezávisle na počtu vstupních neuronů. Mají-li tedy například vstupní data jednotkovou varianci, je zajištěna taková inicializace, která napříč celou sítí zajistí její zachování.



Obrázek 6.3: Histogram velikosti gradientů v jednotlivých vrstvách pro síť s běžnou náhodnou inicializací (nahoře) a pro síť inicializovanou metodou Xavier (dole). Zdroj: [26]

Tato metoda by navržena původně pro sigmoidní aktivační funkce. Existuje rovněž upravená varianta určená pro ReLU aktivační funkce, která se pracovně nazývá MSRA [27].

Analýzou vlivu změny střední hodnoty a kovariance jednotlivých vrstev na průběh trénování se zabývá práce [29]. Zde výzkumníci Googlu dokázali, že podmíněním nulové střední hodnoty a jednotkové kovariance pro všechny vrstvy sítě lze dramaticky snížit nutný počet iterací k natrénování srovnatelného modelu. Proces učení jednotlivých vrstev totiž autoři přirovnávají k odhadu parametrů náhodného rozdělení. V průběhu učení se však díky aktualizaci vah v předcházejících vrstvách parametry tohoto neznámého rozdělení posouvají (Internal Covarince Shift). To způsobí, že hlubší vrstvy se musí neustále přizpůsobovat této změně namísto toho, aby se soustředily pouze na samotný odhad parametrů. Díky zafixování rozdělení hrozí mimo jiné i menší riziko saturace, a proto si můžeme dovolit větší konstantu učení. Zároveň síť údajně lépe regularizuje, tudíž není nutné aplikovat tzv. Dropout.

Konstanta učení

Volba velikosti konstanty učení přímo ovlivňuje dosažený výsledek trénování. Velikost konstanty se pohybuje v řádu desetin, setin až tisícin. Velká konstanta učení umožní skokovou aktualizaci vah, což může vést k rychlejšímu trénování, ale ve většině případů to rovněž vede na oscilace a nestabilitu. U omezených aktivačních funkcí velká změna vah často způsobí saturaci neuronů. Ta vede k zamezení pro-

pagace gradientu chyby, a to poté k nevyhnutelnému zastavení procesu učení nebo naopak k nestabilitě. Malá konstanta učení na druhou stranu zajistí hladký průběh učení, nicméně díky malým krokům hrozí, že algoritmus uvízne v lokálním minimu vzdáleném od globálního minima. Často se tak volba konstanty učení přirovnává k alchymii.

Sympatickým řešením je volba velikosti konstanty dynamicky v závislosti na vývoji derivace vah pro každý neuron. Výpočet nové hodnoty by tak závisel například na velikosti druhé derivace váhy, tedy na sklonu (tvaru) error surface. V každé iteraci by docházelo k volbě hodnoty proporcionálně ke druhé derivaci váhy tak, aby se v globálu učily všechny neurony stejným tempem. Z důvodu výpočetní náročnosti se však používají pouze metody aproximující druhou derivaci. Velikost konstanty je určena například na základě historie gradientu v posledních iteracích (plovoucí průměr, shodná orientace gradientu, aj.).

Momentum

Velmi atraktivní a hojně používaná metoda, která implementuje aktualizaci váhy nikoliv přímo z vypočteného gradientu, ale nejprve jako aktualizaci jakéhosi rychlostního vektoru V_t pomocí gradientu $\nabla L(W_t)$. Až posléze dochází k aktualizaci váhy z tohoto vektoru, jak popisují vztahy 6.5 a 6.6. Na daný neuron o váhách W_t totiž nahlížíme jako na částici, která se v rámci trénování snaží dostat v error surface do globálního minima. Gradient chyby udává částici, respektive vektoru V_t, směr a rychlost. Jestliže jsou gradienty v rámci posledních několika iterací přibližně jednotného směru, částice díky plovoucímu průměru nabere rychlost a řítí se prostorem. Díky své kinematické energii tak může podstatně rychleji překonat plochy s malým sklonem, překonat díry v podobě lokálních minim a ignorovat ojedinělé gradienty, které hlasují pro změnu směru (oscilace).

$$V_{t+1} = \mu V_t - c\nabla L(W_t) \tag{6.5}$$

$$W_{t+1} = W_t + V_{t+1} \tag{6.6}$$

Chování můžeme přirovnat ke kuličce vhozené do misky. Z tohoto připrovnání je bohužel patrné, že částice může snadno přestřelit změnu směru koridoru, který vede k výsledku. Typický příklad nežádoucího chování lze pozorovat pro koridor ve tvaru sedla (údolí). Namísto toho, aby částice docestovala na dno sedla a posléze se vydala jedním ze dvou směrů, v průběhu sestupu nabere rychlost a díky své setrvačnosti bude chvíli oscilovat po stěnách sedla.

Dropout

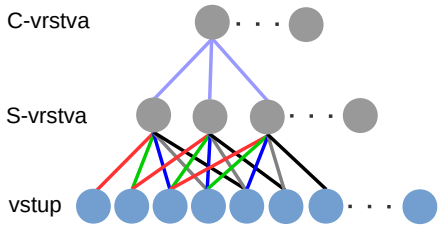
Neuronové sítě díky své vysoké kapacitě (počtu parametrů) během učení silně tíhnou k přetrénování. Tedy pro málo bohatá data je pro síť daleko jednodušší si tato data zapamatovat, než-li se snažit o jejich generalizaci. Namísto toho, aby se schopnosti sítě rozložily napříč vrstvami, určité neurony se adaptují pouze na určité neurony z nižší vrstvy, které se předtím adaptovaly na určitou hodnotu vstupu. V případě, kdy se na vstupu objeví mírně pozměněné data (fáze testu) nebo konkrétní neuron deaktivujeme, tato vazba přestane fungovat a síť díky tomu nedokáže nová data správně zpracovat, i když na trénovacích datech podávala skvělé výsledky. Tento stav je označován jako přetrénování. Z toho je myslím patrné, že toto pravděpodobně nebude způsob, jakým náš mozek pracuje.

Tento problém řeší technika nazvaná Dropout. Každému neuronu je přidělena určitá pravděpodobnost (typicky 50%), že bude v současné trénovací iteraci deaktivován. Trénování probíhá stále stejným způsobem, přičemž nyní je v každé trénovací iteraci určitá část neuronů deaktivovaná. Jinými slovy v každé iteraci obdržíme jiný model sítě, a tedy napříč tisícem iterací fakticky trénujeme (průměrujeme) tisíc různých architektur. Ve fázi testu se už do práce zapojují všechny neurony. Každý neuron se tak v rámci trénování naučí využívat maximálně informace z neuronů z předchozí vrstvy (weak learners), protože se v průběhu trénování nemohl spoléhat, že v další iteraci budou všechny neurony k dispozici. Tato technika je svou funkcí přirovnávána k extrémní formě baggingu.

6.2.4 Konvoluční neuronové sítě

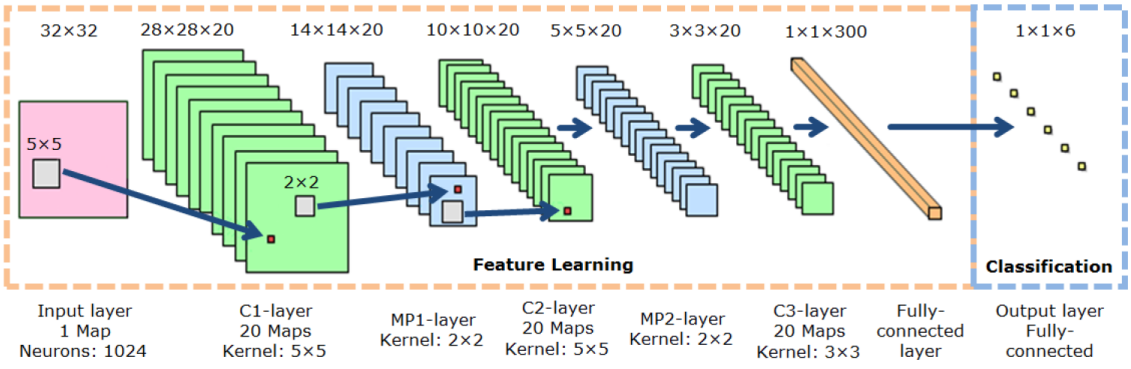
Klíčovým přínosem konvolučních neuronových sítí je schopnost poradit si s vysokou dimenzí dat. Autoenkodéry se bez problému popasovaly s rozpoznáváním vysegmentovaných číslic o dimenzi několika desítek pixelů, nicméně barevné obrázky o tisících pixelech už představují problém. Řešením je omezit pohled neuronu jen na určitou oblast snímku. Jednotlivé neurony jsou uspořádány hierarchickým způsobem tak, že reagují pouze na určitý velmi malý zlomek obrazu - okénko (angl. receptive field), přičemž se okénka sousedních neuronů částečně překrývají. Inspirací této architektury byly výsledky biologického výzkumu zrakového centra mozku savců v jejich rané fázi života. Studie odhalila, že na konkrétní tvary objektů (stimuly) reagují vždy konkrétní shluky neuronů. Fungují tedy jako detektory.

Základním prvkem konvoluční sítě jsou S-buňky (simple cell) a C-buňky (complex cell), jejichž způsob propojení ilustruje obrázek 6.4. Extrakci příznaků provádí S-buňky nastavením svých vah (vizuální stimul). Váhy konkrétního neuronu jsou provázány přes celou vrstvu, respektive feature mapu. Tím je docíleno, že na každý pixel je aplikováno stejné konvoluční jádro. Úkolem je natrénovat síť tak, aby jed-



Obrázek 6.4: Ilustrace struktury konvoluční neuronové sítě. Barevné vazby demonstrují společné hodnoty vah jednotlivých neuronů.

notlivé neurony S-vrstvy reagovaly pouze na určitý příznak v obraze (hranu, roh, atp). Tato vrstva posléze tvoří vstup komplexní vrstvy tvořené C-buňkami. Aplikuje se zde tzv. pooling. Každá C-buňka je propojena pouze s několika S-buňkami a její aktivační hodnota je v případě max-poolingu rovna maximální hodnotě na vstupu neuronu. To zajišťuje jistou formu invariance vůči posunu a zároveň vnáší nelinearitu. Dojde-li totiž k posunu obrazu, dojde rovněž k posunu příznaků, avšak nadřazená C-buňka bude stále aktivní. Zároveň tak dochází k výrazné redukci dimenze dat díky zakódování informace z několika podřazených neuronů do jednoho nadřazeného. Svým způsobem tedy kombinace S-vrstvy a C-vrstvy slouží jako filtr, jenž filtruje z obrazu pouze podstatné informace. Po extrakci příznaků následuje tradiční fully-connected architektura, jejíž funkci lze interpretovat jako transformaci lineárně neseparabilních příznaků do prostoru lineárně separabilního. Klíčovým faktorem stále zůstává návrh vhodné architektury pro danou úlohu. Na to bohužel neexistuje univerzální pravidlo.



Obrázek 6.5: Grafické znázornění konkrétní struktury konvoluční neuronové sítě. Za každou konvoluční vrstvou následuje max-pooling vrstva. Výstup konvoluční sítě je nakonec přiveden na vstup fully-connected sítě. Zdroj: [30]

7 Návrh rekonstruktoru pózy

Návrhu systému pro odhad parametrů ruky z hloubkového snímku lze rozdělit na dvě stejně důležité části. V první části je nutné sesbírat dostatečné množství anotovaných trénovacích dat a zvolit vhodnou formu jejich reprezentace. Ve druhé části přichází na řadu návrh samotného estimátoru, tedy návrh architektury a trénování konvoluční neuronové sítě.

7.1 Příprava trénovacích dat

Pro diskriminativní metody je proces tvorby datasetu velice důležitou fází vývoje celého systému. Nezáleží totiž pouze na kvantitě dat, nýbrž na jejich kvalitě. Sebelépe bohatá data nejsou platná, pokud nejsou náležitě normalizovaná nebo obsahují chyby. S rostoucím množstvím dat zejména druhý zmíněný problém nabývá na důležitosti. Pro rozsáhlé datasety totiž není možno data manuálně zkontrolovat, a tak se musí použít pro kontrolu různé ad-hoc metody vycházející z nějaké apriorní znalosti o úloze. Například máme-li hloubkové snímky ruky, víme, že těžiště ruky bude kolem středu obrázku, velikost kontury bude v určitém rozsahu, stejně tak hodnoty hloubky a jejich variance. Kontrolou těchto hodnot lze už ve fázi přípravy odhalit chyby, které by později během trénování způsobily problémy.

7.1.1 Umělá versus reálná data

Spousta úloh díky svému charakteru umožňuje využít pro své trénování uměle vygenerovaná data, jinak řečeno data vygenerovaná algoritmem simulující nějaký reálný proces. Nejlepší variantou je použít data reálná, jenomže jejich pořízení může být velmi pracné, komplikované, drahé, či téměř nemožné (neopakovatelná data). Naproti tomu u umělých dat nejsme limitováni jejich množstvím a ani variabilitou. Důležité je pochopitelně zajištění věrohodnosti vygenerovaných dat vůči reálným datům.

V kapitole mapující současné metody rekonstrukce byl zmíněn tým stojící za softwarem pro rekonstrukci pózy těla v zařízení Microsoft Kinect. Zde výzkumníci chytře využili omezení variability úlohy a natrénovali celý systém na vygenerovaných umě-

Obrázek 7.1: Ilustrace tří základních metod pro pořízení anotovaných snímků ruky - mo-cap, datové rukavice, nebo polygonový 3D model ruky. Zdroj: [3]

lých datech. Pokud by totiž chtěli získat reálná anotovaná data, dost pravděpodobně by museli využít nějaký mo-cap systém podobný těm, které využívají nejrůznější filmová studia. K podchycení variability by navíc bylo potřeba více herců různé tělesné konstituce. Stačí už jen podotknout, že za tím skrývající se náklady jsou nemalé. Ve srovnání s tím jsou náklady na pořízení umělých dat zanedbatelné.

U navrhovaného systému pro rekonstrukci ruky je problém ještě trochu jiný. Anotace těla pomocí markerů je vzhledem k poměru velikosti markeru a těla relativně snadná. Málokdy se stane, že je marker zcela zakryt. I když k tomu dojde, díky jejich redundantnímu počtu a chytré softwarové interpolaci se moc neděje. U ruky tomu je jinak. Ruka je totiž vůči tělu nepoměrně menší a variabilnější. Čím menší marker, tím hůře je viditelný a naopak čím větší marker, tím méně přirozeně může ruka artikulovat. Navíc díky reflexivní barvě v místě markeru vzniká v hloubkovém snímku díra, kterou je vzhledem k velikostním poměrům daleko těžší věrohodně interpolovat. Druhou variantou anotace je použití již zmíněných datových rukavic. Po přesném zkalibrování jsou rukavice schopny zaznamenat převážnou většinu gest. Protože si ale přejeme provádět estimaci na holé ruce, datová rukavice nám díky svému tvaru a svým datovým vývodům zanese do hloubkové mapy zkreslení. Existují ještě další metody anotace, načež většina z nich využívá nějaký specifický hardware. Naproti tomu principiálně jednoduchý postup zvolili výzkumníci z newyorské univerzity [16]. Za pomoci setu několika barevných kamer zrekonstruovali neúplný 3D model ruky a později pomocí offline generativní PSO metody napasovali počítačový 3D polygonový model na nahraná data. Tím získali ke každému snímku jeho poměrně přesnou anotaci.

Alternativním přístupem a zřejmě i tím nejjednodušším je použití věrohodného polygonového modelu pro generování umělých dat. Zřetelnou výhodou je možnost vygenerovat takřka jakoukoliv pózu ruky snímanou z libovolného úhlu. Rovněž díky své variabilitě lze nasnímat různé velikosti a proporce ruky. Překvapivou nevýhodou je počítačová přesnost těchto dat. Data je nutné zatížit šumem, aby se kvalitou přiblížila výstupu nepřesnému hloubkovému snímači. Druhým ne zcela zřejmým proble-

mem při generování umělých snímků ruky je zachycení apriorní pravděpodobnosti polohy jednotlivých článků (prior). Jinak řečeno, pokud bychom měřili například ohyb jednotlivých článků ruky během přirozené gestikulace nebo i během znakování, z pozdější analýzy bychom zjistili, že různé články ruky se pohybují často jen v omezených rozsazích. Získali bychom tak jakýsi histogram stavů. S ohledem na tento histogram by měla být obsažena trénovací data v datasetu. Kupříkladu měřením bychom zjistili, že ohyb špiček prstů se nejčastěji pohybuje jen v rozsahu 0 až 45°, přestože maximální rozsah ohybu činí přibližně 0 až 90°. Toto zjištění by se tedy mělo odrazit v trénovacích datech. Pokud toto rozložení pravděpodobnosti neznáme, řešením je použít a vyrenderovat nahrávku přirozeného pohybu z mo-cap respektive z datové rukavice.

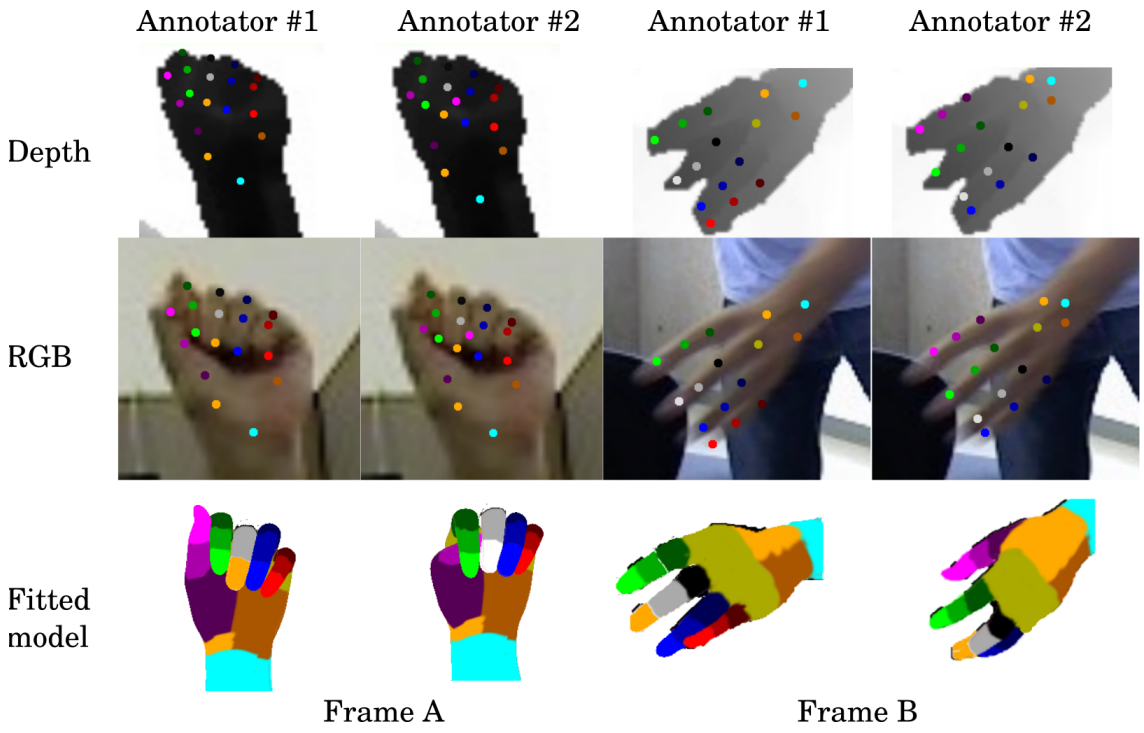
7.1.2 Existující datasety

Před více jak rokem a půl, kdy začala vznikat tato práce, nebyl k dispozici žádný kvalitní dataset pokrývající rozmanitost lidské ruky. Proto jsme se rozhodli prozkoumat možnosti trénování neuronové sítě z umělých dat. Zanedlouho poté se začaly objevovat práce orientované na použití diskriminativních metod a s nimi první datasety. V nich autoři řešili podobný problém s nedostatkem dat, přičemž zatímco jedni vsadili na data umělá [17], jiní nastřádali vlastní reálná data. Jako příklad uvedu dnes již známé datasety NYU [16] a ICL [14]. Za zmínku určitě stojí i novější dataset MSRA [15]. Každý z těchto datasetů má svá specifika a nevýhody. Ať už se jedná o nepřesnou anotaci, malou variabilitu artikulace, omezený rozsah pohledů, či nedostatek herců, v jednom si můžeme být jisti, a to že právě nyní vznikají další a další datasety.

Ze zmíněných specifik bych rád podtrhl nepřesnou, nebo spíše nejednotnou anotaci. Výsledek anotace se totiž odvíjí od rozhodnutí anotátora. Zatímco se nám může zdát anotace kloubů ve snímku jako banální úkol, jak potvrzuje analýza datasetu ICL na obrázku 7.2, není tomu tak. Například na jednom a tom samém gestě se mohou animátoři dopustit chyby v rámci kloubu i více jak několik milimetrů. Přestože tedy konečný rekonstruktor pracuje velice přesně, výsledná (zdánlivá) chyba na takovém gestě může být v součtu i přes dva centimetry [31].

7.1.3 Generování syntetických dat

Ke generování syntetických dat jsem použil open-source modelovací program Blender [32], který umožňuje snadný přístup ke svému API pomocí jazyka Python. Jako model ruky jsem využil realistický open-source polygonový model ruky z



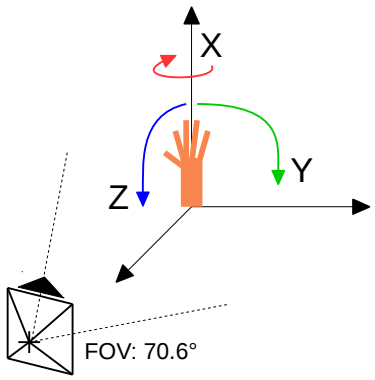
Obrázek 7.2: Ukázka rozdílné anotace stejného gesta od dvou anotátorů. Jak lze pozorovat na zrekonstruovaných gestech, každý milimetr vedle hraje roli. Zdroj: [31]

knihovny LibHand [33], který krom OGRE[1] varianty existuje i pro Blender. Model ruky je složen celkem z 21 kostí, přičemž každá kost může být rotována ve třech osách. To je více než dostatek stupňů volnosti pro modelování přirozených gest. Celou ruku lze navíc díky jednoduchému API ovládat programově.

Jak jsem již zmínil, variabilitě 3D modelu se meze nekladou. Proto jsem uvažoval i variabilní velikost a proporce ruky. V průběhu vývoje jsem navíc zkoušel dvě upravené varianty 3D modelu – variantu s předloktím a variantu zcela bez předloktí. Zatímco u počítačového modelu změna v datech zabere maximálně desítky minut, pozdější renderování maximálně jednotky hodin, u reálných dat bych takovou volnost návrhu neměl, protože každá změna by si vyžádala novou nahrávku a spolu s ní absolvování související procedury. Zároveň však beru na vědomí závěry mnohých publikací, které se shodují na tom, že samotná syntetická data nejsou dostatečná, zato se perfektně hodí pro předtrénování systému, přičemž pro konečné dotrénování systému pak stačí mnohem menší množství reálných dat.

Jako pomůcka při modelování různorodých gest mi posloužil slovník znakové řeči. Včetně neutrální pózy jsem na začátku namodeloval 53 různých gest, jež z většiny pokrývají variabilitu ruky. V modelovacím prostředí jsem vytvořil kameru simulující svými parametry zařízení Microsoft Kinect v2. Bohužel API Blenderu ne-

[1] OGRE je open-source 3D grafický engine.



Obrázek 7.3: Ilustrace os rotace a umístění kamery v Blenderu.

umožňuje přímý přístup k hloubkovým datům (tzv. Z-Buffer), nicméně toto omezení se mi podařilo obejít. Pomocí nástrojů Blenderu jsem extrahovanou hloubkovou informaci ze scény spojil s barevnou informací z rendereru do RGBD snímku (RGB + hloubka). Protože Kinect v2 měří hloubku obrazu v milimetrech v rozsahu 500 až 4500 milimetrů, informaci o hloubce jsem zakódoval na dva bajty (0 až 65535). Díky standardu PNG jsem tedy mohl zkompletovaný RGBD snímek uložit do 16 bitového formátu. Nakonec jsem vždy ruku pro každé gesto nasnímal v předem zvoleném rozsahu pohledů a data uložil jako obrázky pro další zpracování. Ke každému obrázku jsem ještě uložil stejnojmenný JSON[2] soubor obsahující informace o rotaci a poloze jednotlivých kostí.

Rozsah rotace kamery

Rozsah rotace kamery udává, z jakých směrů bude umělá ruka nasnímaná a po jak velkém kroku. To pochopitelně souvisí s tím, v jakých rozsazích vůči kameře si přejeme ruku rekonstruovat. V tom nejjednodušším případě stojíme čelem ke kameře s rukou vzpřímenou vzhůru. Obecně zde platí přímá úměra – čím větší rozsah rotace, tím větší volnost pohybu v zápěstí a v předloktí, ale zároveň tím větší množství trénovacích dat. Kompromisem je hledání velikosti kroku rotace. Nakonec jsem zvolil 15° krok, přičemž jsem bral v úvahu, že velikost vstupního obrázku neuronové sítě je menší jak 100 pixelů. Tudíž pro menší krok se změna v obrázku téměř neprojeví, ale za to se dramaticky nafoukne dataset.

Rozsah rotace vůči kameře jsem v průběhu experimentů několikrát upravoval. Prvotní optimistické rozsahy rotace jsem zredukoval na pohled zpředu, jak ilustruje obrázek 7.3. Po prvních experimentech s reálnou kamerou Kinect v2 jsem si totiž uvědomil, že vzhledem k rozlišení snímače musí být ruka maximálně kolem jednoho metru ve vzdálenosti od kamery, a tudíž některá prve modelovaná natočení ruky

[2]JSON neboli JavaScript Object Notation je datový formát nezávislý na počítačové platformě.

nejsou přirozenou cestou proveditelná.

Vzdálenost od kamery

Blender počítá vzdálenost objektů od kamery v interních jednotkách. Aby bylo možné tuto vzdálenost použít, bylo nejprve nutné transformovat jen určitý konečný vzdálenostní rozsah na interval 0 až 1. K tomu jsem použil interní nástroje Blenderu. Vzdálenost objektů v blízkosti kamery se blíží k nule a naopak vzdálenost objektů na pozadí k jedničce. Takto extrahovanou informaci jsem uložil ve formě Alpha[3] kanálu spolu s barevnou informací do PNG obrázku.

Podobně jako rozsah rotace kamery jsem i vzdálenost objektu ruky od kamery v průběhu experimentů několikrát měnil. Zpočátku jsem ruku umístil do blízkosti kamery na vzdálenost jen několika centimetrů. Výsledkem sice byla velice detailní hloubková mapa, avšak zcela jsem zanedbal dopady perspektivní projekce. Tedy že části ruky blíže kameře se jeví větší než zbytek. Po prvotních experimentech s kamerou Kinect jsem si uvědomil, že přestože umělá i reálná kamera mají stejné FOV, bylo by vhodné ruku umístit přibližně stejně daleko od kamery pro oba případy. Ruku jsem proto v Blenderu nakonec umístil 80 cm od kamery. Abych simuloval rozlišení Kinectu, extrahoval jsem pouze hloubkovou oblast 0 až 4.5 metru.

7.1.4 Perturbace parametrů

Protože neuronová síť svými schopnostmi tíhne k přetrénování, rozhodl jsem se hodnoty rotací jednotlivých kostí perturbovat tak, aby nebyly odstupňovány v pevném kroku, a tím aby nedocházelo k neustálému opakování hodnot. Obával jsem se totiž, že si síť všimne tohoto opakujícího vzorce v anotačních datech. Například že ruka je na jednotlivých snímcích vždy pootočena přesně o 15°, přičemž ostatní parametry ruky se nemění. Při generování každého snímku jsem tedy ke všem parametrům ruky přičetl náhodnou hodnotu z normálního rozdělení. Pro natočení ruky to byla hodnota typicky větší z $\mathcal{N}(0, \sigma = 3)$ a pro zbytek parametrů hodnota o něco menší z $\mathcal{N}(0, \sigma = 2)$.

7.1.5 Předzpracování umělých dat

Z vyrenderovaných obrázků je dále nutno extrahovat hloubkovou informaci spolu s anotací pózy. Extrahovaný snímek má zakódovanou hloubku na 16 bitech, přičemž měřítko je známo. Jednoduchým přepočtem jsem daný 16 bitový rozsah převedl na milimetry. Kinect měří vzdálenost pouze v jednotkách milimetrů, tudíž jsem umělou

[3] Alpha kanál udává u PNG obrázku průhlednost jednotlivých pixelů.

hloubkovou mapu zaokrouhlil na celá čísla. Nyní už jen zbývalo aplikovat šum kvůli rozbití monotónních hodnot.

Měření šumu hloubkového snímače Kinect v2

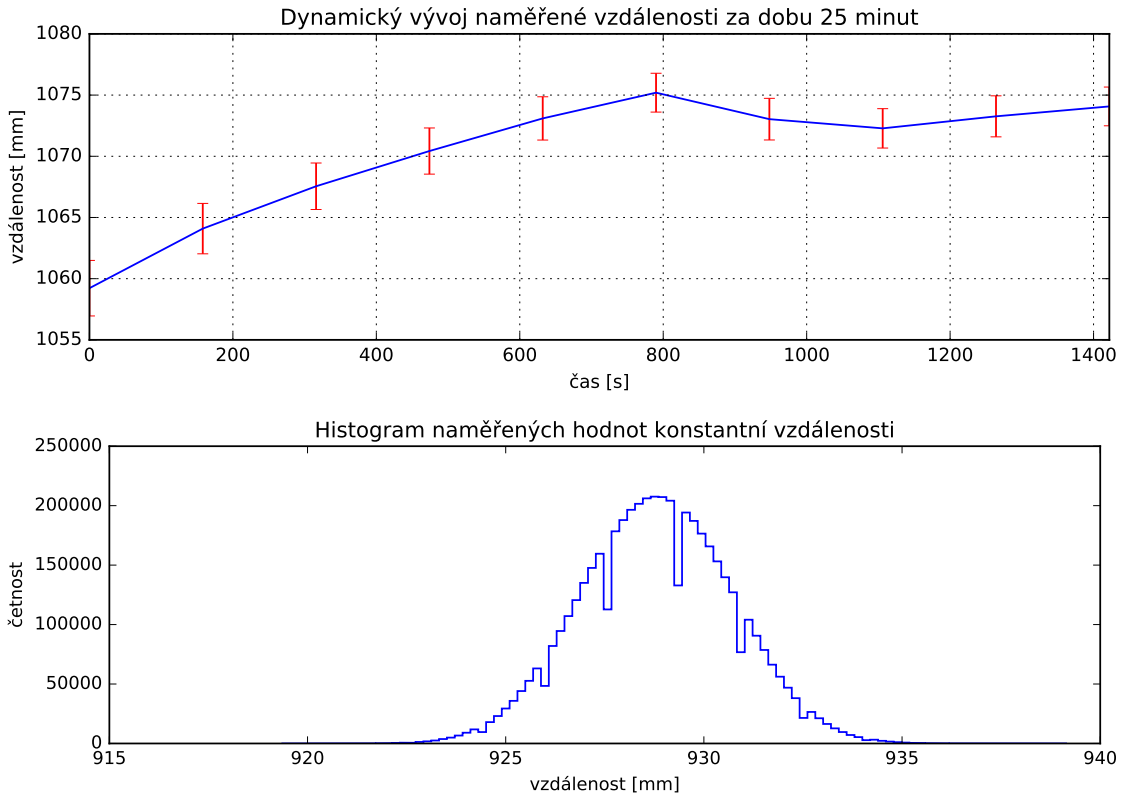
Protože Microsoft údaje o šumu své hloubkové kamery neuvádí, provedl jsem vlastní měření, abych zjistil parametry náhodného rozdělení šumu. Na základě závěrů v publikaci věnované analýze druhé generace snímače Kinect [34] jsem se rozhodl ověřit jeho dynamické vlastnosti. Zde totiž autoři mimo jiné uvádějí, že v průběhu času se hodnota naměřené hloubky mění. Na vině je pravděpodobně zahřívání elektroniky snímače. Provedl jsem tedy měření, abych toto zjištění potvrdil a zároveň abych odhadl parametry náhodného rozdělení šumu.

Při prvním měření jsem zjišťoval změnu hodnoty hloubky v závislosti na čase. Kameru jsem umístil do pevné vzdálenosti od hladké zdi okolo pracovního bodu 100 centimetrů. Pomocí vodováhy a svinovacího metru jsem zařízení nastavil maximálně kolmo na zeď. Jednoduchým programem jsem z každého snímku uložil malý výřez 10x10 pixelů. Měření jsem prováděl každých cca 158 milisekund po dobu 25 minut. Výstupem tedy bylo téměř 10^4 snímků. Z grafu 7.4 můžeme pozorovat vývoj hodnoty hloubky v čase. Přibližně od patnácté minuty je patrné, že se změna ustaluje. Domnívám se, že důsledkem je spuštění vestavěného ventilátoru v kameře.

Před druhým měřením jsem tedy kameru nejprve uvedl do pracovní teploty, abych minimalizoval dopad zahřívání. Provedl jsem podobný záznam, byť kratší, jako v předchozím případě, ale tentokrát pro snímkovací interval 35 milisekund. Naměřené snímky jsem spojil do jednoho vektoru, vypočítal histogram a směrodatnou odchylku. Z grafu 7.4 je patrné, že náhodné rozdělení šumu můžeme bez větší chyby aproximovat normálním rozdělením o nulové střední hodnotě (pracovní teplota) a směrodatné odchylce $\sigma = 1.9$.

7.1.6 Normalizace dat

Nyní, když máme stejný formát umělých a reálných dat, zbývá navrhnout způsob normalizace tak, aby měla umělá vstupní data vhodný formát pro trénování neuronové sítě a zároveň bylo možné stejným způsobem normalizovat i reálná data. Cílem je, aby vygenerované a reálné snímky jednoho gesta byly před vstupem do sítě téměř totožné. U reálných dat jsem tedy musel uvážit proměnou vzdálenost ruky od kamery. Blíže umístěná ruka bude totiž větší a s rostoucí vzdáleností se bude zmenšovat, přičemž gesto zůstává neměnné. Objekt ruky tak musím nejen posunout v hloubce na jednotný interval, ale zároveň přizpůsobit škálu obrázku.



Obrázek 7.4: Měření dynamických vlastností a parametrů šumu u kamery Microsoft Kinect v2.

Posun v hloubce

Způsobů, jak určit velikost posunu, se nabízí několik. Jedním z nich je určení hodnoty hloubky v místě těžiště ruky. Následným odečtením této hodnoty od celého snímku dojde k ustředění do okolí nuly. U této metody jsem však narazil na dva problémy. Pro model ruky s předloktím (viz obrázek 8.6) se těžiště divoce mění v závislosti jak velká část předloktí je součástí vysegmentovaného snímku. Rovněž může nastat případ, kdy je poloha vypočteného těžiště těsně mimo objekt ruky. K tomu dochází například při bočním pohledu na gesta formující pomyslné písmeno C. V takovém případě není ruka ustředěna do okolí nuly, tudíž toto dato můžeme považovat vůči ostatním za outlier. Taková data mohou ve větším počtu během trénování způsobit zbytečné oscilace.

Druhou možností, jak zajistit ustředění, je odečíst průměrnou hodnotu hloubky v každém snímku. Nejen že tím nezávisle na šumu dosáhneme stejného ustředění pro reálná i umělá data, ale zároveň dosáhneme přibližně nulové střední hodnoty pro celý dataset, což je jeden z doporučených kroků před trénováním neuronových sítí.

Konstantní velikost ruky

S měnící se vzdáleností od kamery dochází i ke změně velikosti objektu ruky. Umělá data jsou ale generovaná pouze v konstantní vzdálenosti od kamery. Generování snímků v různých vzdálenostech by jen zbytečně navýšilo množství trénovacích dat. Namísto toho je vhodné změnit velikost obrázku dle průměrné vzdálenosti ruky od kamery, jak popisuje vztah 7.1. Tím lze zajistit, že plocha kontury bude přibližně stejně velká v různých vzdálenostech ruky od kamery.

$$novaVelikost = \frac{prumVzd}{1000} \cdot puvodVelikost \tag{7.1}$$

Například bude-li průměrná vzdálenost objektu ruky od kamery rovna 800 milimetrů, původní obrázek o velikosti 500 pixelů bude zmenšen na novou velikost 400 pixelů.

7.1.7 Augmentace datasetu

Augmentace je proces, kdy ke každému snímku z našeho datasetu aplikací určité transformace vytvoříme N dalších. Z často používaných transformací u obrázků lze jmenovat rotaci, translaci, zrcadlení, přičtení šumu, změnu pozadí, atd. Variant augmentace lze vymyslet jistě spousty a liší se často dle typu úlohy. Vždy je ale třeba najít rozumný poměr mezi reálným přínosem a narůstajícím počtem dat. Cílem augmentace je totiž zlepšit robustnost a generalizaci neuronové sítě. Je obecně známo, že máme-li malé množství dat, neuronová síť díky své kapacitě má tendenci si data spíše pamatovat, nežli je generalizovat. Bohužel často nelze s jistotou říci, kolik dat je zapotřebí, protože to vše se odvíjí od charakteru dat a architektury sítě.

Augmentace jednoduše zabrání síti data si zapamatovat tím, že jí předhodí další data, která ještě neviděla. Je to podobné, jako bychom si prohlíželi album o 10 fotografiích, přičemž bychom nevěděli, na co konkrétně se máme ve snímku zaměřit. Hravě si je ale dokážeme zapamatovat. Pokud by však v albu bylo 1000 fotografií, které se sice liší, avšak na všech nalezneme obličej člověka, dokážeme tento společný rys odhalit, protože jiným způsobem si nejsme schopni obsah fotografií zapamatovat.

Poněvadž zejména v pozdější části experimentů dataset nepříjemně nabobtnal, volil jsem metody augmentace opatrně, abych nalezl optimum mezi přínosem a nárůstem počtu dat. Objekt ruky je svým těžištěm zarovnán na střed obrázku, přičemž poloha těžiště se může díky segmentaci v rámci pár pixelů měnit. Na segmentaci závisí i výpočet průměrné hloubky. Při ustředění ruky do oblasti nuly tak může docházet k milimetrovému posunu. Jako základní metody augmentace jsem tedy aplikoval drobný posun ruky ve třech osách, tedy posun do stran a posun v hloubce. Zpočátku jsem ještě přičítal normální šum, jenomže později jsem přičítání šumu

přemístil do procesu generování umělých dat. Celkově tedy aplikací augmentace z jednoho snímku vzniknou čtyři snímky.

7.1.8 Ground truth data

Protože trénuji neuronovou síť metodou s učitelem, spolu s každým snímkem ruky síti předkládám ground truth informaci, konkrétně hodnoty rotace jednotlivých kostí v kvaternionech. Kvaternion je čtyřdimenzionální vektor popisující rotaci v prostoru $\mathbb{R}^3$. Za zmínku stojí zajímavá vlastnost, a to že k jedné prostorové rotaci přísluší dva kvaterniony $\overrightarrow{q}$ a $-\overrightarrow{q}$.

$$\begin{aligned} \overrightarrow{q} = [q_w, q_x, q_y, q_z] &= [-q_w, -q_x, -q_y, -q_z] = -\overrightarrow{q} \\ q_w, q_x, q_y, q_z &\in \langle -1; 1 \rangle \\ \| \overrightarrow{q} \| &= 1 \end{aligned} \tag{7.2}$$

V počátcích experimentů jsem tuto vlastnost neuvážil, což mělo za následek zhoršené výsledky doprovázené divokými oscilacemi. Později jsem znaménka upravil tak, abych zajistil, že složka $q_w \geq 0$.

7.1.9 Tvorba datasetu

Zbývá nasbíraný dataset zamíchat a rozdělit na trénovací a testovací část. Pokud bych totiž trénoval neuronovou síť pouze na trénovacím datasetu bez validace na neviděných datech, nebyl bych schopen kvalitativně posoudit průběh trénování, a tak například včas odhalit přetrénování sítě. Alternativně lze rozdělit dataset na tři sady – trénovací, validační a testovací. Až na testovací sadě se provede finální vyhodnocení. Aby bylo zajištěno, že získaný výsledek není jen šťastnou náhodou, data se před rozdělením zamíchají a použije se technika cross validation. Tedy náhodné rozdělení na tři sady se provede opakovaně a výsledky se například zprůměrují. Poměry dělení se různí. Jedním z používaných poměrů je 60% trénovacích dat, 30% validačních a 10% testovacích.

7.2 Návrh architektury neuronové sítě

S rostoucí popularitou neuronových sítí se před několika málo lety začaly objevovat Open Source frameworky pro snadný návrh a trénování neuronových architektur. Při vývoji těchto frameworků akademická sféra spojila své síly s tou průmyslovou (Google, Facebook, Microsoft), přičemž se podařilo upoutat i pozornost výrobců hardwaru. Brzy poté inženýři společnosti NVIDIA přiložili ruku k dílu a mimo CUDA implementace určené pro akcelerované výpočty na grafických kartách vyvinuli specializované knihovny ušité na míru neuronovým sítím (cuDNN). Mezi tři

nejpopulárnější frameworky patří Caffe [35], Torch [36] a Theano [37]. Rozhodl jsem se použít framework Caffe, a to z těchto důvodů:

- na vývoji se intenzivně pracuje
- za vývojem stojí aktivní otevřená komunita lidí
- implementovány jsou čerstvě publikované novinky z konferencí
- spolupráce se společností NVIDIA
- moduly jsou naimplementované jak pro výpočty na procesorech, tak pro akceleraci pomocí grafických karet (CUDA + cuDNN)
- návrh architektury probíhá velice jednoduše v textovém protobuf formátu
- existuje oficiální Python wrapper

7.2.1 Ztrátová funkce

Nejčastěji používanou ztrátovou funkcí pro úlohy regrese je funkce inspirovaná euklidovskou vzdáleností nazývaná střední kvadratická chyba (MSE). Kvadratická funkce má sympatický konvexní tvar, tedy budeme-li se pohybovat proti směru gradientu chyby, dorazíme do minima funkce.

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_i - y_i)^2 \tag{7.3}$$

Při porovnávání dvou rotačních vektorů zde ale narazíme. Pokud totiž budeme porovnávat dva vektory v eulerovských souřadnicích (úhly XYZ), musíme uvážit jeden dobře známý problém. Mějme například dvě hodnoty rotace $X_{r1} = 0°$ a $X_{r2} = 359°$ nebo například $X_{r1} = 180°$ a $X_{r2} = -179°$. V obou případech se jedná o téměř totožné úhly svírající vzájemně 1° úhel. Pokud bychom ale dosadili tyto hodnoty do výše uvedeného vzorce, vypočetli bychom velikou chybu. Gradient descent algoritmus by se pak snažil tyto dva vektory otočit tou horší (delší) cestou. Totiž namísto 1° chyby bude algoritmus napravovat 359° chybu.

Reprezentace úhlů v kvaternionech tuto nespojitost sice řeší, nicméně objevuje se jiná komplikace. Pomineme-li již zmíněnou vlastnost, že $\overrightarrow{q} = -\overrightarrow{q}$, během trénování jsem odhalil ještě jeden problém. Rotace je v kvaternionu zakódovaná ve čtyřech složkách a normalizována tak, aby $\|\overrightarrow{q}\| = 1$. Jenomže na výstupu sítě se mohou objevit vektory, kde tato podmínka splněna není. Po ručním znormalizování jsem

tedy zjistil, že dva zdánlivě rozdílné vektory jsou si fakticky velice podobné. Mějme například tyto dva kvaterniony:

$$\begin{aligned} \overrightarrow{q_1} &= [0.296 - 0.149 - 0.011 - 0.033] \\ \overrightarrow{q_2} &= [0.881 - 0.455 - 0.024 - 0.124] \\ MSE(\overrightarrow{q_1}, \overrightarrow{q_2}) &= 0.111 \end{aligned} \tag{7.4}$$

Na první pohled je vidět, že se jedná o úplně odlišné vektory. Z pohledu prostorové rotace tomu tak ale není. Důležitý je totiž poměr mezi složkami vektoru, nikoliv jeho délka. Po znormalizování tak získáme následující vektory:

$$\begin{aligned} \overrightarrow{q_1} &= [0.889 - 0.446 - 0.032 - 0.099] \\ \overrightarrow{q_2} &= [0.881 - 0.455 - 0.024 - 0.124] \\ MSE(\overrightarrow{q_1}, \overrightarrow{q_2}) &= 0.001 \end{aligned} \tag{7.5}$$

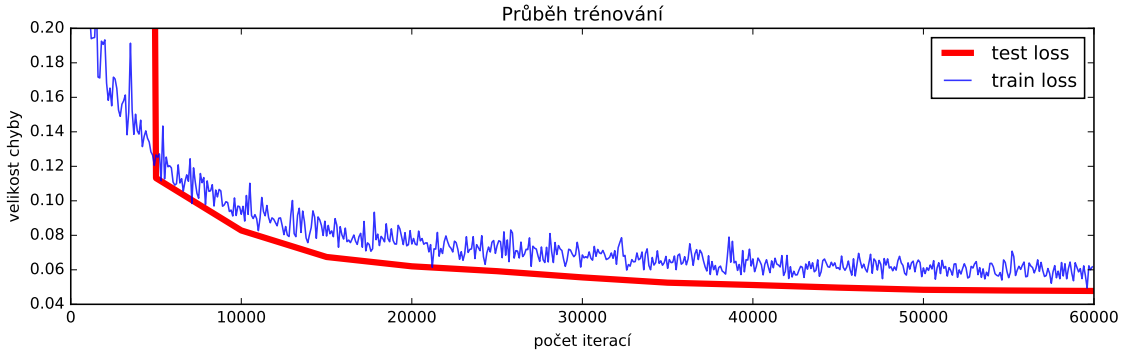
Rozdíl v chybě jsou minimálně tři řády. Přitom se jedná o velice podobné rotační vektory. Drobnou nevýhodou zde je, že si těžko z desetinných čísel uděláme představu, o jak moc podobné vektory se jedná. Proto jsem později hledal jiný způsob reprezentace chyby, který by byl na první pohled informativní. Protože se jedná o dva rotační vektory, domnívám se, že jednou z vhodných reprezentací podobnosti je hodnota úhlu, kterou je zapotřebí k přerotování jednoho z vektorů tak, aby se rovnal tomu druhému. Velice zjednodušeně si to lze představit jako úhel, který je těmito vektory svírán, viz vztah 7.6.

$$\theta = cos^{-1}(2(\overrightarrow{q_1} \cdot \overrightarrow{q_2})^2 - 1) \tag{7.6}$$

Po dosazení dvou předešlých znormalizovaných kvaternionů obdržíme chybu zaokrouhlenou na celé stupně rovnou $\theta = 3°$. Nyní si tak lze udělat lepší představu, jak moc si jsou tyto dva vektory blízké. Pro trénování neuronové sítě jsem však použil již implementovanou MSE ztrátovou funkci bez explicitní normalizace vektorů, a to i kvůli podstatně jednodušší derivaci. Výpočet tohoto úhlu jsem však používal při vyhodnocení modelu a dále je v textu označován jako chyba theta.

7.2.2 Analýza modelu konvoluční neuronové sítě

Abych byl schopen určitým způsobem vyhodnocovat chování navržené architektury, použil jsem celkem tři způsoby analýzy neuronové sítě. Tím prvním triviálním je sledování průběhu trénování a validace neuronové sítě v reálném čase. Jedná se o základní metodu, pomocí které lze odhalit, že nastal problém a trénování neprobíhá tak, jak se očekává. Už v počátcích jsem tedy schopen zjistit, že síť uvízla a odmítá se trénovat, že jsem zvolil příliš malou nebo naopak velkou konstantu učení, zda dochází k poklesu oscilací v závislosti na změně velikosti dávky nebo například zda



Obrázek 7.5: Vizualizace průběhu trénování a validace na testovací (validační) sadě.

Obrázek 7.6: Porovnání nejlepších a nejhorších dosažených výsledků pro konkrétní model neuronové sítě. Jako metrika je použita chyba theta uvedená ve vztahu 7.6.

již dochází k přetrénování modelu. Z uvedeného plyne, že tento způsob kontroly je naprosto nezbytný.

Druhým způsobem vyhodnocení natrénovaného modelu je analýza nejhorších výsledků. Po dokončení trénování jsem na síti vyhodnotil celý validační dataset a za pomoci Blenderu nejhorší výsledky nechal vyrenderovat do PNG souborů. Vedle sebe jsem posléze srovnal očekávanou a zrekonstruovanou pózu. Právě díky této analýze jsem v počátcích odhalil problémy s použitou MSE metrikou popsané v kapitole výše.

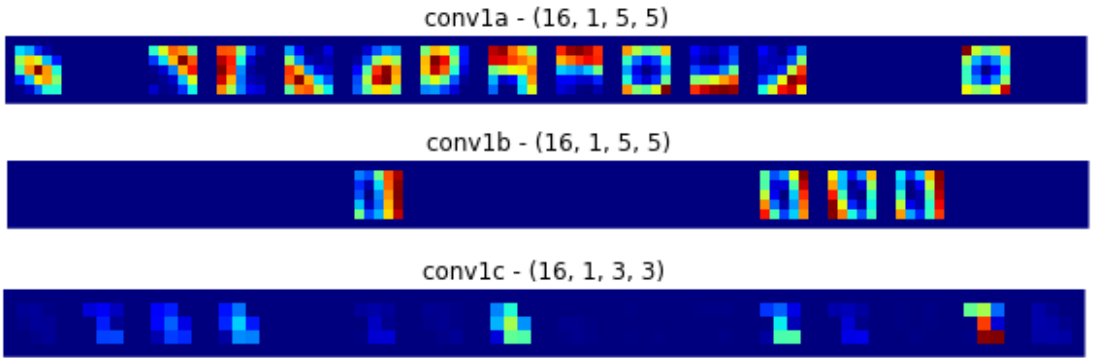
Třetím způsobem je analýza natrénovaných konvolučních jader a analýza odezvy na vstupní snímek v jednotlivých konvolučních vrstvách. Pohledem na jednotlivá konvoluční jádra lze identifikovat rysy, na které se síť soustředila, ověřit, že síť využila všechna dostupná konvoluční jádra a u paralelní multi-scale architektury zjistit, kterou z paralelních větví fully-connected vrstva preferuje. Díky analýze odezvy na vstupní snímek lze pozorovat, co jednotlivá jádra se snímkem provádí a zda výstupní informace má nějakou relevantní hodnotu.

Jako příklad závěru takové analýzy mohu uvést případ, kdy v první konvoluční

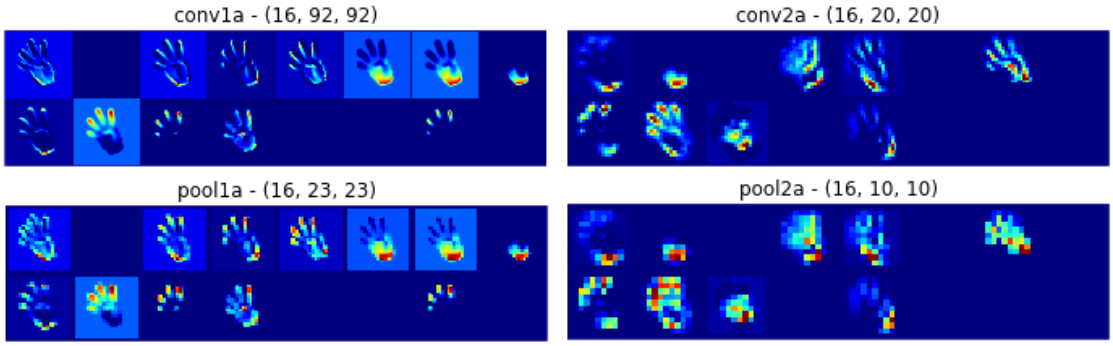
vrstvě vidíme druh odezvy, který si můžeme vysvětlit tak, že zde síť odhalila oblast prstů, oblast palce a zápěstí. Jestliže se pak tato na pohled srozumitelná informace v další konvoluční vrstvě namixuje do mapy velikosti pár pixelů připomínající šum, kde mimo jiné dojde ke ztrátě relativně přesné informace o poloze, jedním z takových intuicí řízených rozhodnutí může být odebrání této poslední konvoluční vrstvy. Vždy ale musí následovat další experiment, který potvrdí zlepšení výsledku, zhoršení nebo bohužel dopadne úplně stejně.

7.2.3 Počet konvolučních vrstev

Začal jsem se sériovou architekturou, kterou mi poskytl kolega Ing. Pavel Campr, Ph.D. Tato architektura byla použita původně pro klasifikaci snímků mikroskopických organismů. Mně ale posloužila jako šablona, ze které jsem se snažil pochopit závislosti mezi jednotlivými parametry. První problém jsem spatřil v počtu konvolučních vrstev. Například pro vstupní snímek velikosti 64x64 pixelů byla výstupem třetí konvoluční vrstvy mapa velikosti 2x2 pixely. V rámci dalších experimentů a studia konvolučních sítí jsem došel k závěru, že hlubší konvoluční vrstvy sice uchovávají vyšší stupeň informace, nicméně díky poolingu dochází ke ztrátě prostorové informace. To se perfektně hodí pro účely klasifikace. Tedy hlubší vrstva nám může říct, že se ve snímku nachází prsty nebo palec. Bohužel ale ztrácíme informaci, kde konkrétně se nacházejí a v jakém pořadí. Odstranil jsem proto třetí vrstvu a dále jsem si vystačil už jen s maximálně dvěma konvolučními vrstvami. Odstranění třetí vrstvy vedlo ke zlepšení výsledku. Později jsem se dozvěděl, že doposud všechny existující práce na téma rekonstrukce pózy používají rovněž maximálně dvě konvoluční vrstvy.



Obrázek 7.7: Vizualizace natrénovaných konvolučních jader z prvních konvolučních vrstev multi-scale architektury uvedené na obrázku 8.7. Můžeme pozorovat, že nedošlo k natrénování všech jader, a proto lze sáhnout k redukci jejich počtu.



Obrázek 7.8: Vizualizace průchodu vstupního snímku první větví multi-scale konvoluční sítě uvedené na obrázku 8.7. Jednotlivá políčka reprezentují odezvu na konkrétní konvoluční jádro. Prázdná políčka znamenají, že zde je konvoluční jádro nulové. Jinými slovy se síť v průběhu trénování rozhodla, že jej nepotřebuje.

7.2.4 Velikost konvolučních jader a jejich počet

Stejně jako u filtrace obrazu i zde platí, že čím větší okénko, tím větší oblast jsme sice schopni zachytit, ale zároveň tím větší detail ztrácíme. Vhodná velikost okénka závisí na velikosti vstupních dat, respektive na velikosti objektu, a na výstupu, který od konvoluční vrstvy očekáváme. Zatímco větší okénka dokáží identifikovat těžiště a orientaci ruky, pro detekci například prstů je nutné použít menší okénko. Během návrhu jsem si pro získání lepší představy vždy vstupní snímek otevřel v grafickém editoru a posléze porovnával různé velikosti oken vůči jednotlivým částem ruky.

Ve většině případů jsem si vystačil s okénky velikosti 2x2 až 5x5 pixelů. U větších jader jsem zjistil, že poněvadž dochází k většímu vyhlazení v první vrstvě (ztrátě detailu), méně kernelových jader v další vrstvě se natrénuje. Rovněž velikost vstupního obrázku 64x64 pixelů se zdála být pro úlohu odhadu rotace (natočení) ruky dostatečná. Až poté, co jsem úlohu rozšířil na odhad všech parametrů ruky, zjistil jsem, že na vstupu postrádám potřebný detail. Zvětšil jsem proto vstupní data na 96x96 pixelů. Nyní jsem sice získal větší detail, jenomže díky malé velikosti jader jsem pro změnu ztratil prostorovou informaci, tedy jakýsi pohled z výšky. Nakonec poté, co [16] zveřejnili svou paralelní multi-scale architekturu, jsem se rozhodl rovněž rozčlenit síť na paralelní větve, jejichž vstupem je vždy snímek různé velikosti. U tohoto typu architektury jsem nakonec setrval.

7.2.5 Velikost poolingu

Klíčovou vlastností poolingu krom redukce dimenze dat je určitá robustnost vůči drobné translaci v předchozí vrstvě. Obecný přínos lze opět nejlépe ilustrovat na případu klasifikace. Namísto toho, abychom uchovávali kompletní informaci z oblasti

o velikosti 4x4 pixelů, uchováme pouze maximum (max-pooling). To nám může pouze říkat, že v této oblasti se s určitou maximální věrohodností nachází palec. Jenomže tímto zároveň ztrácíme informaci, kde přesně se palec nachází a vědomě se tak dopouštíme nepřesnosti. To je pochopitelně u regrese problém. Neznamená to ovšem, že bychom se měli úplně poolingu vyhnout. Naopak pooling je z důvodu redukce dat nezbytný.

Velikost poolingového okénka tedy podobně jako u kernelových jader musí být volena s ohledem na očekávaný výsledek. Jestliže například mám detektor špiček prstů na velkém snímku, zvolím větší velikost okénka. Očekávám totiž, že detekované špičky budou mít mezi sebou velký odstup daný například tloušťkou prstů, kdežto já si přeji extrahovat pouze skelet ruky.

7.2.6 Šířka a hloubka fully-connected vrstvy

Zatímco úkolem konvolučních vrstev je extrahovat příznaky, následná plně propojená síť musí z těchto příznaků vyčíst, respektive rekonstruovat, něco rozumného. U klasifikace se tento proces mnohdy popisuje jako transformace ze silně nelineárního prostoru lineárně neseparabilních tříd do prostoru lineárně separabilního. U regrese bychom tento proces mohli popsat jako funkci mapující příznaky na výsledné parametry. Klíčovými neznámými jsou šířka a hloubka fully-connected vrstev.

Ani zde neexistuje konkrétní recept, avšak v průběhu studia různých publikací jsem si všiml, že šířka fully-connected vrstvy by měla alespoň přibližně odpovídat šířce výstupu konvoluční vrstvy. Platí zde totiž jednoduchá úměra. Čím více kernelových jader, tím více dat na výstupu konvoluční vrstvy. Tyto počty v případě paralelní architektury naskakují velice rychle. Mějme například tuto jednoduchou architekturu:

Vrstva:	**Počet jader:**	**Velikost:**	**Celkem:**
Vstup		96x96	1x1x96x96
C1	8x	5x5	1x8x92x92
P1		4x4	1x8x23x23
C2	16x	4x4	1x16x20x20
P2		2x2	1x16x10x10
—			
Výstup:			1600 neuronů

Máme tedy zdánlivě miniaturní architekturu. V první vrstvě extrahujeme jen 8 příznaků, ve druhé pak 16. Přesto máme na výstupu vektor dimenze 1600. Vezmemeli v úvahu, že toto je jen jedna větev paralelní struktury, pro start budeme potřebovat opravdu širokou fully-connected architekturu. Tím dramaticky naroste počet trénovaných parametrů neuronové sítě i doba trénování. Je tedy třeba volit parametry obou částí sítě tak, aby se vzájemně respektovaly.

Hloubka neuronové sítě se mnohdy přirovnává ke schopnosti extrahovat komplexní informaci. Jako příklad mohu uvést práce [16, 17, 18], kde na výstupu rekonstruktoru nalezneme polohu jednotlivých částí ruky. Zde si autoři vystačili pouze se dvěma fully-connected vrstvami. Toto si vysvětluji tak, že hledaná 2D/3D poloha přímo souvisí polohou jednotlivých detekovaných částí ruky (pixelů) v obrazu. Jinými slovy detekuje-li neuronová síť špičku prstu nebo zápěstí, má téměř vyhráno, protože pouhým přičtením offsetu v závislosti na umístění počátku soustavy souřadnic získá řešení. Natrénování této vazby je proto jednodušší, kdežto mnou navržená neuronová síť se musí ještě naučit mapování mezi polohou nalezených částí a jejich rotací. Proto jsem později začal používat více jak dvě fully-connected vrstvy, což vedlo ke zlepšení výsledku.

7.2.7 Inicializace vah

Během trénování jsem se řídil radami uvedenými v kapitole věnované trénování neuronové sítě. Přesvědčil jsem se, že například nulová střední hodnota datasetu spolu s vhodnou inicializací vah dokáží nezanedbatelně ovlivnit dobu trénování. Prvotní model neuronové sítě inicializoval na začátku trénování své váhy náhodnými malými čísly z normálního rozdělení, avšak po nastudování práce [26] jsem se rozhodl vyzkoušet techniku inicializace zvanou Xavier. Dopady této změny byly takřka nepostřehnutelné, poněvadž v této rané fázi experimentů zmítaly mými daty různé chyby, na které jsem s postupem času přicházel. Rovněž jsem vyzkoušel novější metodu MSRA [27] vhodnou pro ReLU aktivační funkce, avšak neshledal jsem větších rozdílů. Daleko větší dopad na výsledek mělo odstranění zmíněných chyb z datasetu nebo jeho normalizace na nulovou střední hodnotu.

7.2.8 Aktivační funkce

Od samého začátku jsem používal ReLU aktivační funkci, a to jak u konvoluční, tak i u fully-connected sítě. U tohoto typu aktivační funkce totiž nehrozí riziko saturace neuronu jako u sigmoidálních funkcí. Ze zvědavosti jsem po dobu několika experimentů zkoušel tradiční logistickou aktivační funkci i hyperbolický tangens. Brzy jsem se ale na vlastní kůži přesvědčil o problémech se saturací neuronů a nestabilitou. Nakonec jsem se vrátil zpět k ReLU funkci, u které jsem setrval.

7.2.9 Konstanta učení

V rámci studia neuronových sítí jsem se seznámil s různými technikami trénování. Téměř všechny tyto techniky vychází z metody backpropagation, přičemž každá z nich se snaží svým způsobem adresovat určitý problém. Tou nejjednodušší a zároveň nejpoužívanější je Stochastic Gradient Descent v kombinaci s metodou Momentum. Základními parametry, které je třeba zvolit, jsou velikost konstanty učení a parametr momentum. Momentum viditelně pomáhá urychlit trénování, zvláště pak v jeho počátcích. V konečné fázi, kdy dochází k citlivému dotrénování vah, je naopak momentum spíše na škodu. Po většinu trénování jsem proto volil hodnotu 0.9, a v pozdějších fázích jsem pak snižoval na hodnoty okolo 0.5.

Volbu konstanty učení považuji za správnou alchymii. Základní poučka říká, že na začátku trénování je vhodné zvolit co největší konstantu učení. Konkrétně volíme takovou hodnotu, která ještě zajišťuje konvergenci, často z intervalu kolem 10^{-2} až 10^{-1}. S postupem času je vhodné konstantu snižovat, nejlépe v momentě, kdy jednoznačně vidíme, že velikost chyby přestala klesat. Ke snižování konstanty může docházet buďto postupně (harmonicky) nebo skokově.

Mně se nejlépe osvědčil přístup postupného systematického trénování. Totiž na začátku jsem vždy zvolil co největší možnou konstantu učení. Díky tomu chyba během několika iterací dramaticky klesne, přičemž asymptoticky dokonverguje na pevnou hodnotu, kolem které už jen osciluje. Jakmile jsem tedy usoudil, že hodnota chyby již neklesá, přerušil jsem trénování a snížil hodnotu konstanty učení o polovinu či více. Takto jsem postupoval až do hodnot okolo 10^{-6}, kdy už viditelně k žádnému trénování nedocházelo. Alternativním přístupem je vsadit na některou z poloautomatických optimalizačních technik, které volí konstantu učení dynamicky například dle vývoje velikosti gradientu. Není třeba si tak lámat hlavu s vhodnou volbou konstanty učení. Nicméně většina těchto metod má své vlastní volitelné parametry, tudíž laborování s volbou konstanty učení nahrazuje laborování s volbou těchto parametrů. Navíc ve všech případech jsem dosáhl horšího výsledku oproti pečlivému ručnímu trénování.

7.2.10 Dropout

Techniku Dropout jsem již popsal v kapitole věnované neuronovým sítím. Důvodem, proč jsem použil tuto metodu, byla snaha zabránit přetrénování sítě. Totiž od určitého pokročilého okamžiku, kdy už trénovací i validační chyba výrazně klesla, začíná validační chyba zpátky pomalu narůstat. Jedná se o jasný signál přetrénování. Důvodem tohoto chování je, že náš model poskytuje daleko větší kapacitu, než je pro tuto úlohu zapotřebí. Nejjednodušším řešením je zastavit v tomto místě

trénování, protože lepšího výsledku už nedosáhneme. Pochopitelně můžeme upravit architekturu modelu tak, abychom snížili jeho kapacitu. Zejména fully-connected vrstvy mohou být zdrojem tohoto problému. Tato úprava ale bývá pracná, protože nevíme přesně, kde a kolik ubrat. Proto se nabízí jiné řešení.

Varianty, jak tomuto jevu předejít, jsou minimálně dvě. Buďto vytvořit augmentovaný dataset, nebo použít techniku zvanou Dropout. Augmentace však výrazně nafoukne počet trénovacích dat, což může být neúnosné. Na druhé straně Dropout krom drobného prodloužení doby trénování nestojí nic a přitom funguje velmi dobře. Díky 50% Dropoutu se mi podařilo nejen zabránit přetrénování sítě, ba co víc výsledek ještě zlepšit.

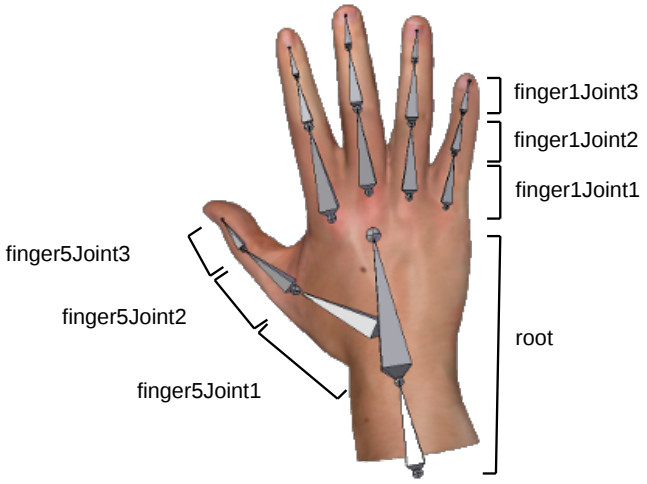
8 Experimenty na umělých datech

Většina poznatků uvedená v předchozích kapitolách vychází z mnou provedených experimentů. Namísto nesourodého chronologického popisu jsem se proto rozhodl získané poznatky uvést uceleně v samostatné kapitole a v kapitole věnované experimentům zachytit spíše postup při práci na úloze. Nejprve jsem začal s jednodušší úlohou odhadu rotace ruky, tedy její orientace a natočení (tzv. root). Cílem bylo přesvědčit se, že je síť schopná regrese, a zároveň najít takovou architekturu sítě, na které by se později dalo stavět. Prvotní dataset obsahoval 53 gest a ruka byla pro dané gesto nasnímaná s krokem 15°. S postupem času, jak se dařilo snižovat hodnotu chyby, jsem se nakonec rozhodl do úlohy připojit zbylé části ruky. Konečný systém odhaduje rotaci celkem v 16 kloubech, respektive 16 kvaternionů, viz obrázek 8.1.

8.1 Rekonstrukce rotace ruky

Hlavní náplní prvotního experimentu bylo jednak se seznámit s technikami návrhu ve frameworku Caffe a jednak nalézt takovou architekturu, jejímž vylepšováním by docházelo k viditelnému zlepšení výsledků. Problém návrhu konvoluční sítě je, že nikde neexistuje univerzální postup, neexistuje žádná kuchařka, která říká, jaký parametr změnit, kde a jakou vrstvu přidat. Úloha je o to složitější, protože se věnuji regresi na datech a nikoliv klasifikaci. Zatímco pro klasifikaci již existuje množství publikací, ze kterých čtenář může načerpat základní paradigma návrhu, regresi řeší minimum prací. Návrh architektury neuronové sítě lze nadneseně přirovnat ke kompozici hudby. Noty známe, ale zcela záleží na jejich pořadí. V počátcích návrhu mi tak nezbývalo, než se řídit intuicí a metodou pokus-omyl.

Začal jsem se sériovou architekturou tvořenou nejprve třemi, později dvěma konvolučními vrstvami, a dvěma fully-connected vrstvami. Zkoumal jsem vliv změny jednotlivých parametrů na výsledek predikce. Během analýzy výsledků jsem přišel například na již zmiňovaný fakt, že $\overrightarrow{q} = -\overrightarrow{q}$. Pozoruhodné na tomto zjištění bylo, že se tuto vazbu síť dokázala naučit. Nicméně po zavedení podmínky $q_w \geq 0$ došlo ke snížení oscilací a ke zlepšení výsledků. Celkově se mi v rámci experimentů podařilo snížit velikost průměrné chyby theta z prvotních 14.3° na konečných 5.44°, což



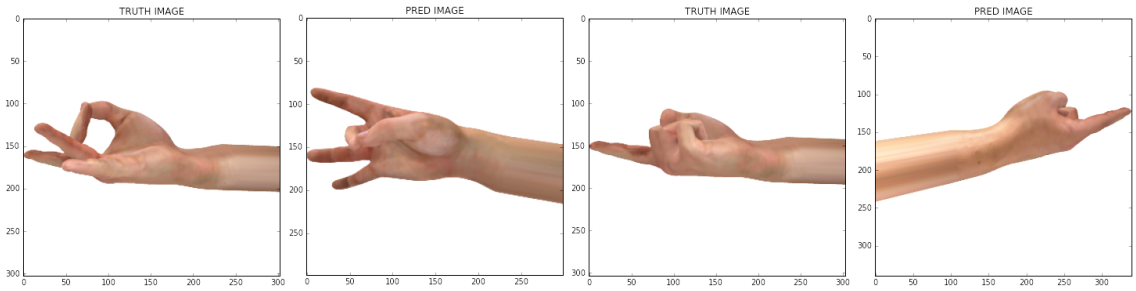
Obrázek 8.1: Popis jednotlivých článků 3D modelu ruky.

je vzhledem k velikosti obrázku 64x64 pixelů lidským okem takřka nepozorovatelný rozdíl. Alternativním způsobem vyjádření poklesu chyby je sledování relativního počtu dat, jejichž chyba theta je menší nebo rovna 15°. Zde lze pozorovat zlepšení z původních 91.04% na konečných 96.27%. Po pozdější redukci rozsahu rotace kamery na úhly uvedené v tabulce 8.1 a zvětšení vstupního snímku na 96x96 pixelů se mi podařilo průměrnou chybu ještě snížit až na 4.8° (97.78%).

Po celou dobu experimentů s odhadem rotace ruky jsem se potýkal s jednou neustále se opakující chybou. Při pohledu na nejhorší výsledky se mezi chybami nacházely zrekonstruované pózy, kde sice síť skvěle odhadla orientaci ruky, jenomže ve finále přerotovala ruku o téměř 180°. Tato ruka byla posazena na pomyslnou osu tvořenou předloktím, avšak podél této osy byla buďto otočena tak, že palec byl na druhé straně, nebo celá ruka směřovala opačným směrem. Lépe celou situaci ilustruje obrázek 8.2. Vysvětlení tohoto chování shledávám následovně. Protože úkolem sítě je pouze odhad natočení ruky nezávisle na kterémkoliv z 53 gest, síť se naučí při predikci koukat spíše na předloktí, které udává hlavní osu orientace. To je dle mého názoru zcela pochopitelné chování. Jenomže z předloktí už síť není schopna určit rotaci ruky kolem této osy, a protože se gesta ruky snímek od snímku velmi mění, má síť tendenci tuto změnu ignorovat podobně jako šum, protože neví, čeho se zde chytit. Má se chytit palce nebo prstů? Kterého prstu? Nebo se chytit dlaně? Ta ale není vždy vidět. Vše se totiž v průběhu trénování vůči zápěstí a předloktí mění.

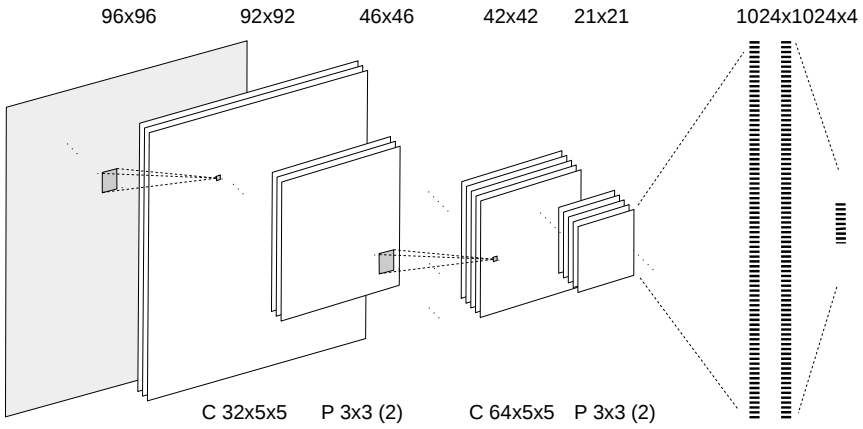
Osa:	X	Y	Z
Rotace:	-45° .. 45°	-45° .. 45°	-90° .. 90°

Tabulka 8.1: Rozsah rotace ruky (rootu) vůči kameře.



Obrázek 8.2: Ukázka chybného přetáčení odhadu rotace ruky. V obou případech můžeme pozorovat, že systém odhadl hlavní osu orientace, avšak ruku už nenatočil správně.

Přišel jsem proto s řešením k úloze predikce krom rotace celé ruky ještě přidat rotaci palce s vírou, že si síť uvědomí, že palec je vždy v určitém vztahu k rotaci ruky. Jinými slovy palec je vždy na jedné straně, a tedy pomocí této znalosti lze zamezit těmto případům. Výsledky však dopadly neprůkazně. Jak se později ukázalo při rekonstrukci pózy celé ruky, myšlenka byla správná. Zde si tedy neúspěch vysvětluji malým rozlišením vstupního snímku spolu s nevhodnou velikostí konvolučních okének vůči velikosti palce. Zatímco tedy pro úlohu rekonstrukce natočení ruky byly parametry sítě vhodné, pro rekonstrukci palce nikoliv. Další experimenty už neprokázaly žádné zlepšení výsledku, a proto jsem se vrhl na rekonstrukci celé ruky včetně všech prstů. Finální architektura neuronové sítě, na které bylo dosaženo nejlepšího výsledku, je uvedená na obrázku 8.3.



Obrázek 8.3: Finální sériová architektura neuronové sítě pro odhad rotace ruky.

8.2 Rekonstrukce pózy celé ruky

8.2.1 Globální a lokální vazby

Rotace kostí v jednotlivých kloubech 3D modelu ruky existují ve dvou variantách. Hodnoty rotací mohou být uvedeny v lokálních nebo globálních souřadnicích. V lokálních souřadnicích je hodnota rotace uvedena ve vztahu ke své rodičovské (předchozí) kosti. Naproti tomu hodnoty v globálních souřadnicích jsou uvedeny ve vztahu k pevnému bodu v prostoru. Obě varianty jsem vyzkoušel, přičemž obě mají svá pro i proti. Začal jsem s odhadem rotací v globálních souřadnicích. Intuitivní výhodou je, že hodnota rotace nezávisí na zbytku pózy, nýbrž pouze na pixelech ve svém okolí. Určitá hodnota rotace má tedy pouze jednu vizuální reprezentaci. Zároveň nedochází ke kumulaci chyby, protože je-li natočení ruky odhadnuto s chybou, ostatní kosti mohou být zrekonstruovány správně. Poněvadž ale póza svírající jedno gesto při každém svém pootočení nabývá jiných hodnot, není možné v rámci trénování zobecnit apriorní model pro jednotlivé kosti. Zatímco se tedy náš prst může ohýbat pouze v rozsahu 0 až 90 stupňů vůči rodičovské kosti, protože je rotace uváděna v globálních souřadnicích, v průběhu trénování nabývá téměř všech možných hodnot. Při rekonstrukci tak často dochází k případům, kdy jsou špatně viditelné části ruky zrekonstruovány do nereálné pózy. Výsledek tak zcela odporuje fyziologickým omezením reálné ruky.

Abych zamezil těmto případům, pokusil jsem se o omezení variance výstupu vytvořením hrdla těsně před výstupní vrstvou neuronů. Vývoj hodnot parametrů ruky je totiž vcelku korelovaný proces, jak jsem se mimo jiné přesvědčil po aplikaci PCA analýzy. Rozhodl jsem se proto vyzkoušet metodu aplikovanou [17, 18]. Za poslední skrytou vrstvu jsem připojil vrstvu neuronů užší, než je vrstva výstupní. Za tímto hrdlem následuje opět široká fully-connected vrstva následovaná vrstvou výstupní. V případě lineární aktivační funkce se tak jedná o variantu PCA. Například vrstva šířky 4096 neuronů je zkomprimována na 48 neuronů, dekomprimována zpět

Obrázek 8.4: Ukázka chybné rekonstrukce v globálních souřadnicích. Rekonstruovaná póza odporuje fyziologii reálné ruky.

na 4096 a nakonec přivedena na výstupní vrstvu šířky 64 neuronů. Nelineární aktivační funkce možnosti komprese ještě posouvají. Protože jsou tedy data korelovaná, síť bude nadále schopna rekonstruovat původní pózy, avšak mělo by dojít k zamezení náhlých změn úhlů mezi kostmi. Jinými slovy svíraná póza by měla ve většině případů vykazovat monotónní charakter. Maximální zúžení hrdla, které se mi podařilo natrénovat, aniž by docházelo k významnému zkreslení výsledků, se pohybuje mezi 16 až 20 neurony. Jak ukázaly experimenty, problémové případy však úplně nevymizely, ačkoliv lze pozorovat zlepšení. Cenou takového modelu je ale dlouhé a komplikované trénování.

U lokálních souřadnic lze naopak snadno generalizovat v průběhu trénování silně apriorní model. V průběhu predikce by se díky tomu nikdy nemělo stát, že odhadnutá rotace odporuje fyziologickým omezením reálné ruky. Toto chování považuji za nesmírnou výhodu. Totiž části ruky, které síť nevidí dostatečně nebo vůbec, jsou rekonstruovány do statisticky nejpravděpodobnější pózy. Pokud tedy bude v našich trénovacích datech dobře podchycen apriorní model reálné ruky, tedy apriorní pravděpodobnost jednotlivých hodnot rotací, lze očekávat věrohodný a stabilní odhad i v případech, kdy jsou některé části zakryty. Intuitivní nevýhodou v tomto případě se zdá, že například pro sto různých pootočených variant jednoho gesta po síti požadujeme vyjma rootu stejný výstup. Z toho jsem usuzoval, že výsledky na lokálních souřadnicích budou horší. Už však po prvních experimentech se ukázalo, že neuronová síť rekonstrukci pózy v lokálních souřadnicích síť zvládá daleko lépe, než rekonstrukci v souřadnicích globálních. Primárně jsem se proto soustředil právě na lokální souřadnice.

8.2.2 Multi-scale architektura

Z experimentů, v nichž jsem se přidáním palce snažil zlepšit odhad rotace ruky, jsem došel k závěru, že pro rekonstrukci celé pózy budu muset architekturu sítě výrazně modifikovat. Inspirovaný prací [16] jsem navrhl vlastní paralelní multi-scale architekturu. Vstupem každé ze tří větví byl velikostně odstupňovaný obrázek o rozměrech 96, 48 a 24 pixelů. První výsledky byly zprvu nepřesvědčivé, nicméně brzy jsem si uvědomil, že je zapotřebí věnovat pozornost konečnému dotrénování. Tedy přestože se pro svůj exponenciální průběh může zdát, že už k žádnému trénování nedochází, analýza konvolučních jader mne vyvedla z omylu. Jen díky pečlivé manuální volbě konstanty učení v závislosti, zda chyba klesá, nebo stagnuje, se mi podařilo na stejném modelu snížit průměrnou chybu theta i o více jako deset stupňů. Například drobnou úpravou počtu konvolučních jader, šířky a hloubky fully-connected vrstvy spolu s optimalizací trénování jsem docílil snížení prvotní průměrné chyby 78.30° na hodnoty menší jak 10°. Po přechodu na lokální souřadnice jsem se z nejlepší

průměrné chyby 9.45° na globálních souřadnicích dostal na nejlepší výsledek 3.99° dosažený na lokálních souřadnicích. Ke zlepšení došlo místy o více jak řád. Uvážíme-li velikost obrázku, respektive velikost jednotlivých částí ruky, které jsou mnohdy tvořeny jen několika málo pixely, dojdeme k názoru, že už velikost chyby menší jak 10° lze považovat za uspokojivý výsledek. Není rovněž žádným překvapením, že k největší chybě dochází u malíčku a obecně u posledních článků prstů.

8.2.3 Přirozená mezigesta ruky

První experimenty s modelem trénovaným pro 53 gest na reálných datech potvrdily mé obavy, že síť nedokáže zrekonstruovat odlišná gesta a raději se bude snažit hledat nejpodobnějšího kandidáta z viděných gest. Toto chování vyplývá jak ze základních vlastností diskriminativních metod, tak z důsledku zobecněného apriorního modelu. Jestliže se například v trénovacích datech pohybuje ohyb článku jen v rozsahu 0 až 45 stupňů, nelze od sítě očekávat, že zvládne rekonstruovat gesta, kde je ohnut do pravého úhlu. Je tedy patrné, že složíme-li naší ruku do nějakého mezigesta, protože máme síť natrénovanou jen pro 53 diskrétních gest, s rekonstrukcí tohoto mezigesta si síť patrně neporadí správně. V našich datech jsme totiž zdaleka nepokryli variabilitu lidské ruky, a tak se nyní nelze divit důsledkům.

Poněvadž jsem neměl k dispozici nahrávku přirozeného pohybu ruky z mo-cap, nebo z datové rukavice, rozhodl jsem se zlepšit pokrytí variability v umělých datech tak, že jsem sloučením dvou gest vytvořil v trénovacích datech nové gesto (mezigesto). Nejprve jsem k existujícím 53 gestům jsem namodeloval ještě pár dalších. Z celkového počtu 59 gest jsem jejich sloučením metodou každý s každým vytvořil 1770 mezigest. To je bohužel nepřijatelně vysoké číslo, zejména uvážíme-li, že pro každé gesto vzniknou stovky až tisíce variant s různou rotací. Proto jsem se rozhodl vzniklá gesta nashlukovat tak, aby došlo k vyřazení vzájemně podobných mezigest. V rámci shlukování jsem u každého gesta hodnoty rotací zaokrouhlil do pomyslného 45° intervalu. Tím se z doposud podobných gest stala gesta stejná. Zbylo mi tak již jen 866 unikátních kombinací. S tímto množstvím jsem se smířil a mezigesta ponechal v datasetu.

Poněvadž došlo k dramatickému nárůstu počtu gest, musel jsem znovu zredukovat rozsah rotace ruky. Mimo jiné jsem si totiž při prvních experimentech před kamerou uvědomil, že zkrátka některá natočení nejsou kvůli omezením segmentace přirozeně proveditelná, a tak došlo k úpravě na přirozenější hodnoty rozsahu na-

Osa:	X	Y	Z
Rotace:	-75° .. 15°	-45° .. 45°	-90° .. -15°

Tabulka 8.2: Finální rozsah natočení ruky vůči kameře pro odhad všech parametrů.

Obrázek 8.5: Na obrázku jsou vyobrazeny různé velikostní varianty 3D modelu. Vlevo menší silnější ruka, veprostřed neutrální velikost a vpravo hubenější ruka s delšími prsty.

točení, viz tabulka 8.2. Přestože došlo ke zvýšení průměrné chyby rekonstrukce o více jak dva stupně na hodnoty 6-7°, síť nyní díky mezigestům výrazně lépe zvládá rekonstrukci reálného přirozeného pohybu ruky. Jsem však nadále přesvědčen, že ještě lepších výsledků by bylo dosaženo použitím mo-cap nahrávky volného pohybu nebo znakování.

8.2.4 Různé proporce a velikosti ruky

Velikost ruky se jedinec od jedince může velmi lišit, a proto jsem se rozhodl estimátor navrhnout tak, aby byl vůči těmto změnám robustní. Částečně sice svou práci odvede přidání šumu během generování dat a regularizace v průběhu trénování, přesto jsem se ale rozhodl namodelovat další dvě velikostní varianty. Modelování ruky jsem nikterak neposouval do extrému. Vědomě jsem například neuvážil natolik dramatické velikostní rozdíly, jaké mohou být mezi dospělou pánskou a malou dětskou rukou. Namísto toho jsem se rozhodl pouze pro variantu hubené ruky s dlouhými prsty a pro variantu silnější ruky s kratšími prsty. Spolu s neutrální velikostí jsem tedy systém natrénoval celkem na tři velikostně různé varianty 3D modelu.

Místo abych tento robustní model trénoval od samého počátku na trojnásobně velkém datasetu, použil jsem jako startovní bod již precizně natrénovaný model určený pro neutrální ruku. Díky tomu si síť jen drobně poupravila parametry tak, aby byla vůči těmto velikostním nuancím robustní. Toto dotrénování tedy zabere výrazně méně času, než by bylo jinak zapotřebí. Nakonec se mi podařilo dosáhnout stejné chyby, jako u neutrální velikosti ruky. Při pohledu na konvoluční jádra z obou modelů jsem nepozoroval žádné větší změny. Nikde nepřibylo nebo neubylo žádné konvoluční jádro. Na několika kernelech lze sice pozorovat drobnou změnu hodnot, jedná se však o téměř nepostřehnutelnou změnu. Předpokládám tedy, že nejvíce změn se odehrálo ve fully-connected části, poněvadž konvoluční jádra slouží primárně jako extraktory příznaků.

Obrázek 8.6: Ukázka 3D modelů ruky s různou délkou předloktí. Vlevo je model, který by použit při odhadu rotace ruky v prvních fázích experimentů, veprostřed originální model z knihovny LibHand a vpravo finální model s uříznutým předloktím.

8.3 Nejlepší dosažené výsledky

V rámci návrhu systému jsem provedl veliké množství dílčích experimentů, jejichž výsledky jsem se pokusil shrnout v předchozích kapitolách. Nejlepších výsledků se mi obecně podařilo dosáhnout na paralelní multi-scale architektuře. Experimentoval jsem snad se všemi možnými parametry neuronové sítě, přičemž žádný z experimentů nepřekoval výsledky dosažené architekturou na obrázku 8.7. Rovněž varianty s více či méně paralelními větvemi nevedly ke zlepšení.

Po absolvování prvních reálných experimentů jsem se rozhodl zkrátit 3D model ruky přesně po zápěstí viz obrázek 8.6, ačkoliv jsem se obával, že dojde ke zhoršení odhadu natočení ruky. Důvod k tomuto zásahu byl takový, že, abych zmenšil počet trénovacích dat, rozhodl jsem se nemodelovat ohyb v zápěstí. Jenomže před kamerou jsem si uvědomil, že přes veškerou snahu při provádění gest se nelze ohybu v zápěstí vyvarovat. Oříznutí ruky po zápěstí toto omezení odstranilo, čímž jsem získal takřka neomezenou volnost ohybu v zápěstí, přičemž nedošlo k obávanému zhoršení výsledků při odhadu natočení ruky.

V tabulce 8.3 jsou uvedeny nejlepší dosažené výsledky pro úlohu odhadu všech parametrů ruky. Vizualizaci jednotlivých kostí lze nalézt na obrázku 8.1. Jedná se o model sítě natrénovaný na třech velikostních variantách ruky. Z výsledků si můžeme všimnout, že k největší chybě dochází na prvním článku palce. Tato kost je totiž bohatě obalena hmotou, přičemž z ruky nikterak zvlášť nevyčnívá. Rekonstrukce tak probíhá z relativně velké plochy pixelů, která je velmi zatížena šumem, a z polohy druhého článku palce. Přesný odhad natočení je v tomto případě zcela nemožný. Uvedená chyba 8.93° je natolik nevýznamná, že na podobu výsledného gesta takřka nemá vliv. Velkou roli v tomto případě rovněž hraje apriorní model, poněvadž funkční rozsah této kosti je velmi omezený. Překvapivě nejlépe si vede od-

finger1joint1	finger1joint2	finger1joint3	finger2joint1
6.70°	5.56°	6.55°	5.18°

finger2joint2	finger2joint3	finger3joint1	finger3joint2
5.48°	7.45°	5.46°	5.50°

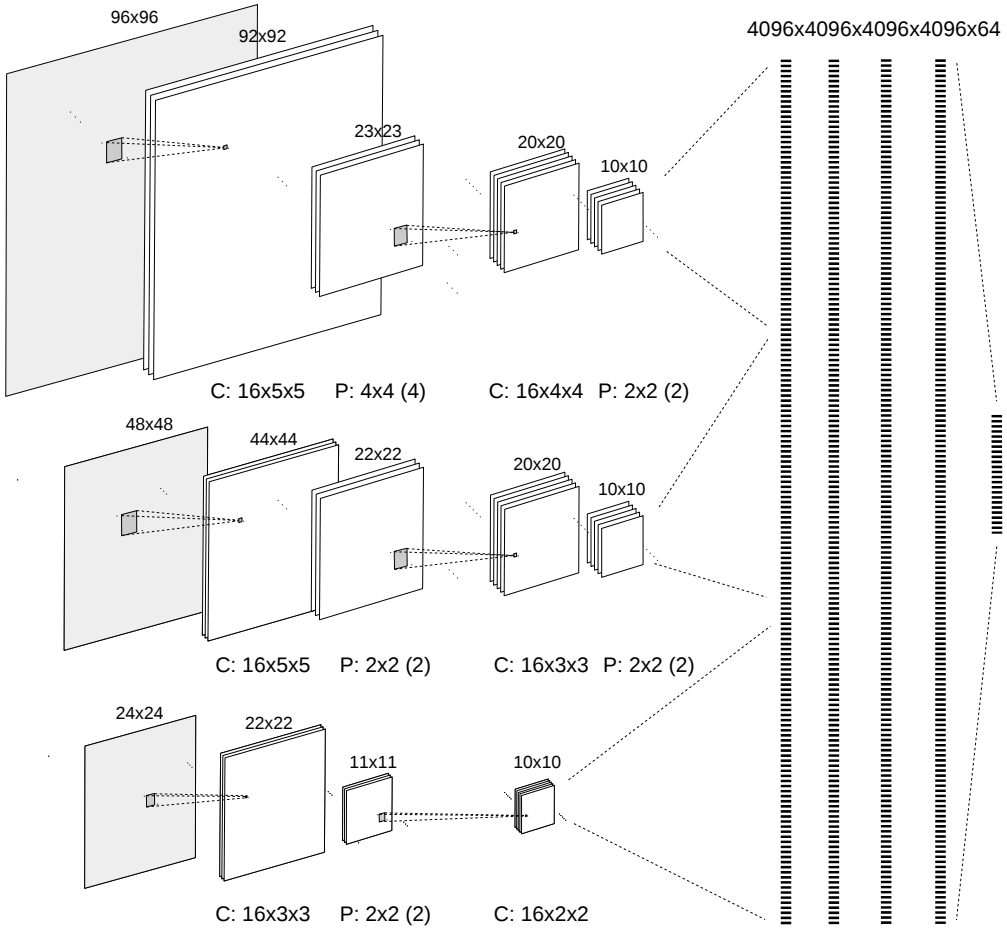
finger3joint3	finger4joint1	finger4joint2	finger4joint3
7.00°	6.30°	5.15°	4.96°

finger5joint1	finger5joint2	finger5joint3	root
8.93°	5.83°	6.74°	4.66°

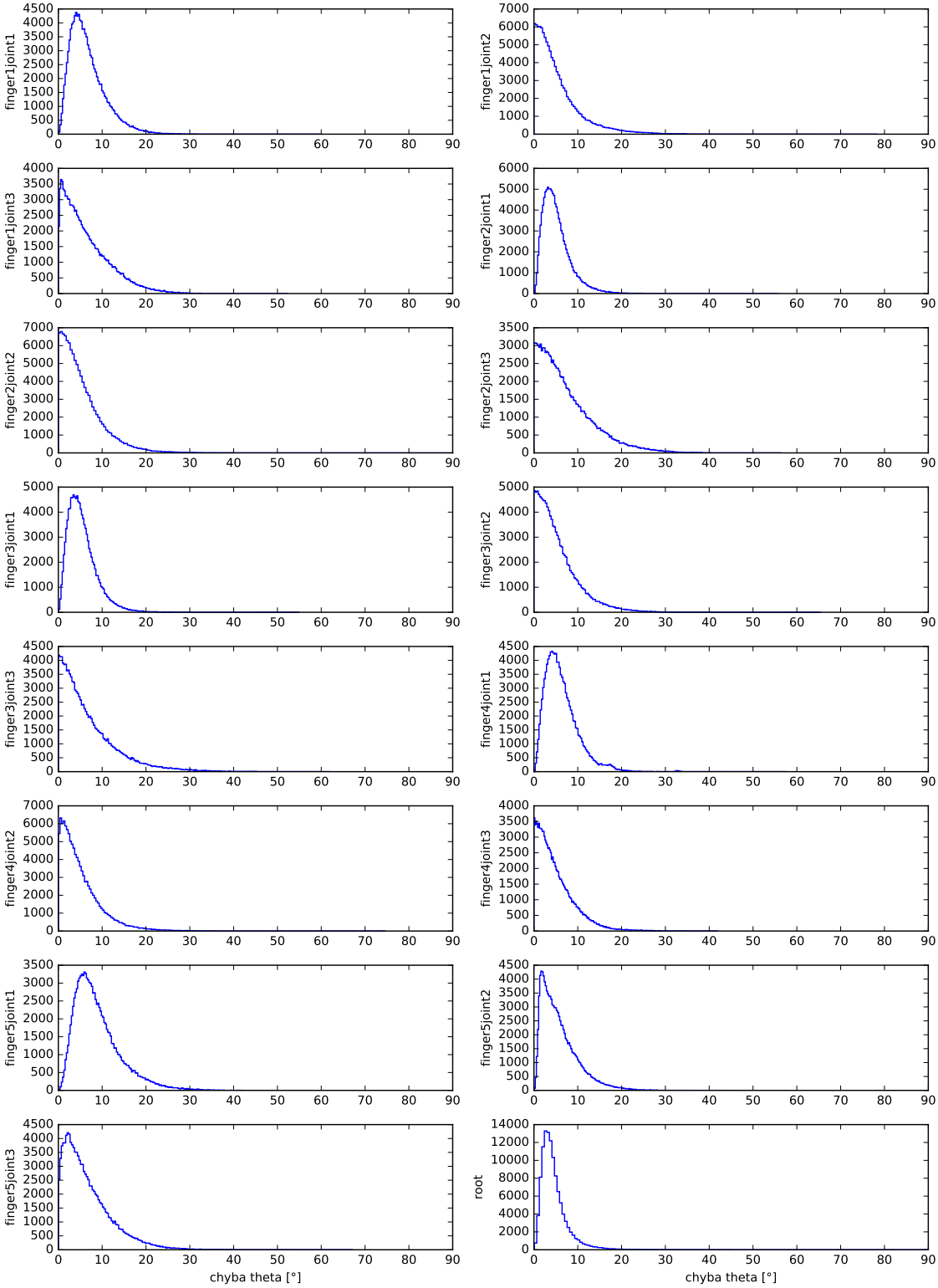
průměrná hodnota
6.09°

Tabulka 8.3: Tabulka s nejlepšími dosaženými výsledky pro konkrétní model neuronové sítě.

had rotace ruky. Průměrná hodnota chyby 4.66° je fantastický výsledek, který bych v počátcích experimentů vůbec neočekával. Jsem přesvědčen, že tomuto výsledku dopomohla informace o poloze ostatních článků ruky, protože již nadále nedochází k případům s přerotováním, které jsem popsal v předchozí kapitole.



Obrázek 8.7: Výsledná navržená multi-scale architektura podávající nejlepší výsledky.

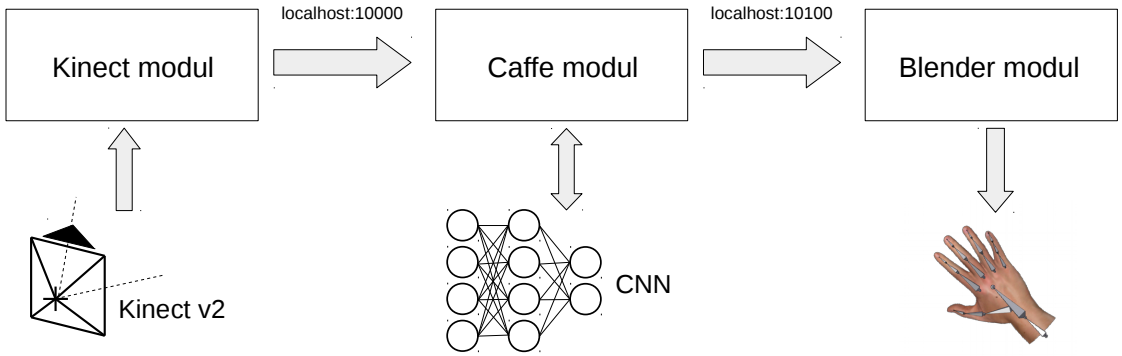


Obrázek 8.8: Histogram chyby theta pro jednotlivé články ruky.

9 Experimenty na reálných datech

Konečným cílem této práce je rekonstrukce pózy ruky z reálných dat, respektive z hloubkových snímků pořízených kamerou Microsoft Kinect v2. Jako třetí, testovací, část datasetu mi proto poslouží reálné snímky pořízené touto kamerou. Bohužel nemám k dispozici žádnou anotační metodu, která by poskytla ground-truth informaci. Musím se tak spokojit pouze s vizuálním vyhodnocením. Tedy zkoušením různých gest musím ověřit, že síť je schopna jejich rekonstrukce. Abych toto vyhodnocení mohl provézt, musel jsem nejprve vytvořit nezbytné programy pro pořízení snímků a pro vizualizaci výsledků. Navržený systém jsem nakonec dekomponoval na tři zcela samostatné moduly. Moduly mezi sebou vzájemně komunikují přes rozhraní TCP/IP, a nemusí proto běžet na stejném počítači.

Motivace stojící za návrhem modulárního řešení je poměrně jednouchá. Každý modul je z určitého důvodu napsaný v jiném programovacím jazyce. Překlenout tento neduh v rámci jedné aplikace je sice technicky možné, avšak zbytečně komplikované. Samostatné moduly běží zcela paralelně v oddělených procesech. Rovněž paralelizaci lze řešit v rámci jedné monolitické aplikace, nicméně v případě Pythonu je to poněkud zapeklité, protože vlákna neběží zcela paralelně. S tím trochu souvisí i třetí bod, a tím je přístup k hardwarovým prostředkům systému. Neuronová síť totiž běží na grafické kartě mnohdy i o více jak řád rychleji než na procesoru. Ideálně by tak celá aplikace musela vždy běžet na počítači s grafickou kartou, nebo alespoň



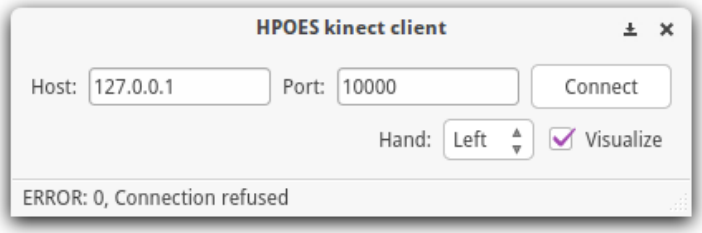
Obrázek 9.1: Blokové schéma jednotlivých komponent systému.

se silným procesorem. Modulární implementace naproti tomu umožňuje řešení typu tenký a tlustý klient. Tedy náročná predikce poběží na výkonném desktopovém počítači (server), kdežto vizualizace může běžet třeba na televizní obrazovce připojené k notebooku (klient). V neposlední řadě díky zvolenému univerzálnímu TCP/IP rozhraní mohou jednotlivé moduly běžet jak na různých počítačích, tak i na různých platformách. Totiž například framework Caffe oficiálně podporuje pouze systémy Linux a Mac OS, kdežto zařízení Microsoft Kinect v2 oficiálně podporuje pouze operační systémy Windows. Přitom zpoždění způsobené TCP/IP komunikací je v rámci jednoho počítače (localhost) neměřitelné a v rámci lokální sítě zanedbatelné ($<$ 1 ms po kabelu).

Ve všech případech můj navržený komunikační protokol postavený nad TCP/IP vyžaduje potvrzení zaslané zprávy. Důvod je prostý. Odesílatel si totiž nemůže být jist, zda příjemce už zprávu dostal a zpracoval. Pokud by totiž příjemce narazil na problém nebo se zdržel, odesílatel by v nevědomí vesele posílal další data. Zamezím tím potencionálnímu přetečení komunikačního bufferu. Pokud by totiž odesílatel posílal data daleko rychleji, než je příjemce schopen zpracovávat, mohlo by dojít z důvodu naplnění bufferu k nekonzistenci dat. Mezi daty totiž není žádný oddělovač a v případě, že by buffer zahodil stará data nebo přetekl, dojde k porušení konzistence. Po rekonstrukci získaného bajtového pole by namísto snímku ruky vznikl naprostý nesmysl. Zároveň tvoří-li některý z modulů hrdlo, nemá smysl, aby ostatní moduly pracovaly naplno.

9.1 Návrh Kinectového modulu

Úkolem tohoto modulu je stáhnout z kamery Microsoft Kinect v2 hloubkový snímek, provézt segmentaci ruky a poslat vysegmentovanou ruku do modulu s neuronovou sítí. Oficiální ovladače od výrobce existují pouze pro Windows, avšak podobně jako u první verze zařízení Kinect i tentokrát komunita vytvořila vlastní perfektně funkční open-source ovladače libfreenect2 pro všechny tři majoritní desktopové platformy. Rozhraní prozatím existuje pouze pro jazyk C++. Jelikož libfreenect2 nabízí multiplatformní využití, rozhodl jsem se rovněž svůj C++ modul napsat pro více platforem. K tomu jsem využil fenomenální C++ framework Qt [38], který v sobě implementuje nejrůznější multiplatformní knihovny, například pro vývoj uživatelských aplikací, síťovou komunikaci, databáze, multimédia, atd. Pomocí Qt jsem vytvořil jednoduchou uživatelskou aplikaci, která se skládá celkem ze čtyř modulů – GUI formuláře, Kinect modulu, segmentačního modulu a TCP/IP modulu.



Obrázek 9.2: Snímek jednoduchého uživatelského rozhraní Kinectového modulu.

9.1.1 GUI formulář

Provádí inicializaci uživatelského rozhraní z knihovny Qt. Rovněž inicializuje ostatní moduly aplikace do samostatného vlákna, aby se dala aplikace z uživatelského rozhraní ovládat. Zároveň hlavní formulář informuje uživatele o chybách v průběhu inicializace kamery a TCP/IP modulu, viz obrázek 9.2.

9.1.2 Kinect modul

Pomocí knihovny libfreenect2 stáhne barevný a hloubkový snímek z kamery. Stažená data jsou převedena na univerzální datový typ Mat používaný knihovnou OpenCV [39].

9.1.3 Segmentační modul

Velice jednoduchý segmentační modul založený na prahování ručně specifikované barvy. Návrh segmentační metody nebyl předmětem této práce, proto jsem se uchýlil k jednoduchému hledání barvy v kombinaci s těmi nejjednoduššími technikami zpracování obrazu. Úspěšná segmentace tedy vyžaduje nošení barevné rukavice, to proto, aby se podařilo vysegmentovat ruku vždy jen s určitou částí předloktí nebo kompletně bez něj. Za ideálních světelných podmínek funguje segmentace velmi přesně a rychle (< 5 ms).

Protože lze pomocí libfreenect2 knihovny získat barevný snímek přesně synchronizovaný s hloubkovou mapou, stačí nalézt v obrazu barevnou rukavici a vyříznout z hloubkového snímku plochu kontury nalezené rukavice. V každém snímku tedy provedu barevné prahování. Kolem ruky se objevuje mnohdy i několikapixelový šum, proto jsem použil morfologickou operaci otevření se strukturálním elementem ve tvaru kříže, abych tyto otřepy zvláště v okolí prstů odstranil. Zároveň tím dochází k odstranění šumu ve zbytku binárního obrazu, což ulehčí práci metodě hledající kontury objektů. Po nalezení kontur si ponechám jenom tu největší, protože předpokládám, že se ruka nachází v blízkosti kamery a tedy plocha její kontury má

větší velikost, než zvolený práh. Pokud tomu tak není, v segmentaci dál nepokračuji a metoda vrací logickou hodnotu false značící, že segmentace selhala. V opačném případě pokračuji segmentací objektu ruky z hloubkového snímku za pomoci nalezené kontury. Z pixelů spočtu průměrnou hodnotu vzdálenosti ruky a na základě tohoto poměru vzdálenosti snímek s rukou zvětším nebo zmenším. Cílem je, aby ruka zůstala stejně velká, nachází-li se 80 nebo 120 centimetrů od kamery. Nakonec od těžiště ruky odměřím na každou stranu okénko, tuto oblast vyříznu a výsledný vysegmentovaný snímek ruky uložím do výstupní matice. Metoda vrací logickou hodnotu true, čímž dává signál TCP/IP modulu.

Segmentační modul je záměrně separovaný od zbytku kódu ve formě C++ třídy v odděleném souboru. Pro toto řešení jsem se rozhodl v naději, že jiný student vyvine takový robustní segmentační modul, u kterého nebude zapotřebí nosit barevnou rukavici nebo dbát na ideální světelné podmínky. Domnívám se totiž, že se jedná o úlohu, která svým rozsahem dokáže pokrýt bakalářskou či diplomovou práci.

9.1.4 TCP/IP modul

Před tím, než vůbec započne snímání a segmentace, se TCP/IP modul nejprve pokusí zkontaktovat se serverem. Navrhl jsem naprosto triviální komunikační protokol vzdáleně inspirovaný emailovým komunikačním protokolem SMTP. Na začátku se obě strany pozdraví, vymění si informace o datech a po zprávě DATA následuje tok dat, respektive jednotlivé snímky rozložené na pole bajtů. Komunikační protokol pro data typu float32 o velikosti 96x96 pixelů vypadá následovně:

Qt klient:	**Caffe server:**
HELLO	HELLO
WIDTH 96	OK
HEIGHT 96	OK
DTYPE float32	OK (test existence datového typu)
DATA	OK (výpočet velikosti v bajtech)
blok_dat	OK
...	

Komunikace je zjednodušena na nezbytné minimum. Nejsou logovány ani ošetřeny potencionální chybové stavy, jako například situace, kdy vypadne spojení. Přestože tedy aplikaci nic nebrání v komunikaci mimo lokální síť, primárně se předpokládá, že vše půjde hladce bez komplikací.

9.2 Návrh Caffe modulu

Caffe modul funguje jako serverová aplikace napsaná v jazyce Python 2. Aplikace vyžaduje natrénovaný model neuronové sítě, do kterého jsou posléze posílány obdržené snímky. Výstup tohoto modelu je posílán znovu přes TCP/IP do aplikace Blender.

Na začátku programu dojde k ověření a inicializaci modelu neuronové sítě. Při tomto procesu totiž může dojít s velkou pravděpodobností k nečekanému pádu aplikace, a proto se vykonává jako první. Framework Caffe je totiž napsaný v C++. V případě, že dojde k nečekanému problému, knihovna jednoduše havaruje. Důvodů k pádu může být mnoho, protože Caffe závisí na množství knihoven třetích stran. Zároveň komunikuje s grafickou kartou prostřednictvím proprietární CUDA knihovny, a tak zajištění bezproblémového chodu rozhodně není triviální záležitostí. Po úspěšné inicializaci následuje navázání komunikace s vizualizačním modulem. Znovu jsem implementoval jednoduchý komunikační protokol, který po pozdravu pošle modulu důležitou informaci o tom, která data patří jaké kosti.

Caffe klient:	**Blender server:**
HELLO	HELLO
[seznam kostí] (JSON)	OK (rekonstrukce seznamu)
SHAPE 64	OK
DTYPE float32	OK (test existence datového typu)
DATA	OK (výpočet velikosti v bajtech)
[blok_dat]	OK
...	

Následuje inicializace serveru a čekání na připojení klienta. Po připojení proběhne již popsaná komunikační procedura. Ze získaných informací o rozměrech snímku a jeho datovém typu modul vypočte, kolik bajtů odpovídá jednomu snímku. V každé iteraci se tedy očekávané množství bajtů přečte z bufferu a provede se rekonstrukce na odpovídající datový typ. Pokud rekonstrukce selže, spojení končí, což je zpráva pro ostatní moduly, že došlo k problému nebo k ukončení. Snímek se znormalizuje a projde neuronovou sítí. Výstupní vektor kvaternionů se nakonec odešle do Blenderu a algoritmus pokračuje další iterací.

9.3 Návrh Blender modulu

Ze všech tří modulů je implementačně nejsložitější ten vizualizační napsaný v jazyce Python 3 pro aplikaci Blender. Jediným způsobem, jak přistoupit k API Blenderu, je spustit v aplikaci Python skript. Bohužel z důvodu, jakým způsobem je spouštění skriptů v Blenderu implementováno, nelze tento skript nechat běžet na pozadí. Jakmile totiž dojde k exekuci, grafické rozhraní Blenderu zcela zmrzne, dokud skript svou práci nedokončí. Nedochází tedy k překreslování 3D modelu. Existuje však interní komponenta zvaná Operator, pomocí níž lze tento problém částečně vyřešit. Dokumentace je v tomto případě až dramaticky strohá, nicméně princip funkce si vysvětluji jako typ event filtru. Tedy instance se napojí do event loop a prochází skrz ni jak události vlastních komponent, tak obecné události aplikace. Jedná se o dobře známé chování z programování uživatelských aplikací.

Na začátku programu proběhne inicializace serveru a čekání na klienta. Po připojení klienta následuje již známá komunikační procedura. Skript končí registrací mého modálního operátoru, který reaguje pouze na událost vlastního časovače a na stisk klávesy Esc. Všechny ostatní události jsou ignorovány a přeposlány dál. Časovač zhruba každých 30 milisekund způsobí vyvolání události, která vede k přerušení event loop a k zavolání obslužné metody. To způsobí dočasné zmrznutí rozhraní Blenderu, protože skrz operátor nemohou proudit ke svým handlerům další události čekající ve frontě. Během tohoto přerušení je zavolána má metoda, která přečte data z komunikačního bufferu. Opět je znám datový typ a velikost dat, tudíž není problém dopočítat délku zprávy v bajtech. Data jsou následně zrekonstruována na pole kvaternionů. Každé kosti je nakonec přiřazena odpovídající rotace, metoda končí a event loop je opět průchozí. Protože došlo ke změně parametrů 3D modelu, Blender automaticky vyvolá událost, která zajistí překreslení modelu ruky.

9.4 Výsledky na reálných datech

Pro otestování schopností rekonstrukce pózy jsem zvolil model sítě, na kterém bylo dosaženo nejlepšího výsledku a který je uveden v předchozí kapitole. Test probíhal před zeleným nebo černým plátnem s modrou rukavicí. Důvodem bylo maximální zjednodušení barevné segmentace a minimalizace případné chyby způsobené neúplnou segmentací. Spustil jsem jednotlivé moduly aplikace, které po vzájemné inicializaci a navázání komunikace začaly posílat data. Všechny dílčí moduly běžely na jediném desktopovém počítači v reálném čase plynule bez viditelných příznaků trhání či zasekávání. Stručná sestava počítače je uvedená v tabulce 9.1. Oba náročné moduly segmentace a neuronové sítě dosahují výpočetních časů < 4 ms. Důležitou

Obrázek 9.3: Ukázka segmentace reálné ruky před černým plátnem. Vlevo je barevný obrázek synchronizovaný s hloubkovou mapou, veprostřed vidíme hloubkovou mapu a vpravo konečně vysegmentovaný objekt ruky.

roli přitom hraje mimo procesoru zejména grafická karta, která se stará o ty nejnáročnější výpočty. Kromě neuronové sítě (CUDA) totiž grafickou kartu využívá knihovna libfreenect2 pro rekonstrukci a transformaci hloubkové mapy (CUDA) a zároveň Blender pro vizualizaci 3D modelu (OpenGL). Když jsem tedy aplikaci testoval na jiném počítači vybaveném podstatně slabší grafickou kartou NVIDIA GTX 740 (762 GFLOPS v SP[1]), bylo již patrné drobné cukání, přičemž obraz byl stále relativně plynulý (odhadem okolo 20 FPS).

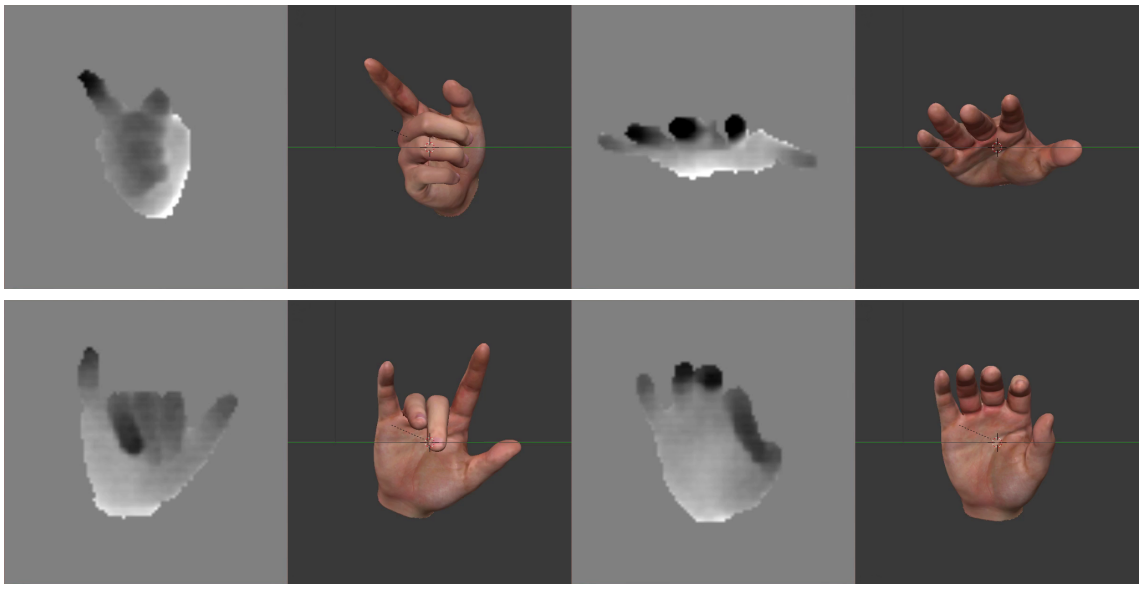
Procesor a RAM:
Intel Core i5 3570K @ 4.5 GHz, 8 GB DDR3 @ 1867 MHz
Grafická karta:
NVIDIA GTX970 @ 1316 MHz, 4 GB GDDR5 @ 6003 MHz (3494 GFLOPS v SP[1])

Tabulka 9.1: Sestava použitá pro online testování na reálných datech.

Celkem jsme rekonstrukci naživo otestovali já a vedoucí mé práce Ing. Zdeněk Krňoul, Ph.D. Oba jsme si vyzkoušeli jak volný pohyb v různých natočeních, tak tvorbu gest. Z pozorování mohu prohlásit, že přesnějších výsledků dosahoval kolega, jehož ruka je nepatrně silnější, než má. Domnívám se, že za to může Kinect a způsob, jakým měří hloubku v prostoru. V důsledku toho totiž nedokáže zachytit zaoblené kraje ruky, od nichž se infračervený paprsek odráží jen špatně, či spíše vůbec. Výsledkem je tedy štíhlejší ruka, zvláště prsty, než je tomu doopravdy. I když se jedná třeba jen o pár pixelů, vzhledem k velikosti ruky snímané metr od kamery to jistě má nezanedbatelný dopad.

Pokud bych měl rekonstrukci pózy ruky z reálných snímků ohodnotit kvalitativně, tak rekonstruktoru chybí smysl pro detail. Zkrátka póza je ve většině případů zrekonstruovaná dobře řekněme z 90%, avšak chybí schopnost dotáhnout rekon-

[0]Uvedená hodnota značí výpočetní výkon v tzv. Single Precision módu (např. float32 čísla).



Obrázek 9.4: První řádek ukazuje velice přesnou rekonstrukci pózy. Na druhém řádku nalezneme zrekonstruované pózy postrádající detail. Dochází tedy k nějaké menší chybě - vlevo je zvednutý prst navíc, vpravo chybí dotyk palce a ukazováku.

strukci do detailu. Jinými slovy například gesta tvořená dotykem bříšek nebo překřížení prstů představují problém. Vždy chybí pár milimetrů k přesné rekonstrukci. V tomto případě by se skvěle hodila generativní metoda, která by odhadnutou pózu pojala jako inicializační stav a následně dokonvergovala k přesné póze. Zároveň by došlo k odstranění jitteru, tedy drobného chvění způsobeného šumem kamery.

Odhad celkového natočení ruky funguje skvěle pro intervaly obsažené v trénovacích datech, viz tabulka 8.2. Na pohled není znát větší chyba. Pozoruhodné je, že neuronová síť zvládá i rekonstrukci zrcadlené ruky. Tedy systém je natrénován na levou ruku, přičemž v uživatelské aplikaci lze přepínačem zvolit, zda má segmentace ruku převrátit (zrcadlit) tak, aby byla možnost rekonstruovat i ruku pravou. I bez tohoto přepnutí však rekonstrukce z větší části funguje, aniž by síť odhadovala nesmysly nebo nereálné pózy. Podobný případ je pro natočení, která nejsou uvedena v trénovacích datech. Odhadnuté natočení ruky je sice chybné, avšak prsty na ruce mají snahu se ohýbat do správného gesta.

10 Závěr

Cílem této práce bylo prozkoumat nové metody strojového učení založené na neuronových sítích a získané poznatky aplikovat v praxi za účelem odhadu a rekonstrukce pózy lidské ruky z hloubkových snímků pořízených 3D kamerou. V teoretické části jsem se věnoval historickým i současným metodám rekonstrukce pózy. Podrobněji jsem nastudoval a rozebral aktuální state-of-the-art postupy včetně posledních pokusů o aplikaci neuronových sítí pro úlohu rekonstrukce (regrese) pózy. Nastudoval jsem problematiku trénování neuronových sítí spolu s technikami trénování konvolučních neuronových sítí. Vytvořil jsem vlastní dataset z umělých dat vygenerovaných počítačovým 3D modelem ruky. Navrhl jsem takové metody předzpracování a normalizace dat, abych normalizoval data umělá vygenerovaná a data pořízená 3D kamerou na stejnou úroveň. Samostatně jsem navrhl vlastní architekturu konvoluční neuronové sítě pro účely odhadu a rekonstrukce pózy. Po absolvování prvních experimentů s odhadem natočení ruky jsem došel k závěru, že tvar architektury má zásadní vliv na dosažený výsledek. Během prvních experimentů s odhadem parametrů celé ruky jsem usoudil, že sériová architektura konvoluční neuronové sítě nedostačuje k zachycení potřebných vazeb mezi jednotlivými částmi ruky. Později jsem proto sériovou architekturu sítě nahradil paralelní multi-scale architekturou, která dokázala daleko lépe zachytit jak lokální detail, tak globální vazby v celém obrazu. Pro natrénování nejlepšího modelu jsem použil metodu Stochastic Gradient Descent s metodou Momentum, přičemž přetrénování modelu jsem předcházel jak augmentací datasetu, tak pomocí techniky Dropout.

První experimenty jsem prováděl za účelem odhadu orientace a natočení ruky. Pomocí sériové architektury neuronové sítě se mi podařilo snížit chybu z prvotních 14.30° na konečných 4.80°. Pro odhad všech parametrů ruky již sériová architektura nepodávala dostatečné výsledky, a tak jsem navrhl paralelní multi-scale architekturu. V úloze odhadu všech parametrů ruky jsem dosáhl snížení celkové průměrné chyby z počátečních 78.30° na konečných 6.09°. Získanou minimální hodnotu považuji vzhledem k velikosti vstupního snímku 96x96 pixelů za velice dobrý výsledek přibližující se limitům diskriminativních metod.

Pro reálné experimenty jsem natrénoval model na třech velikostních variantách

ruky, abych rekonstrukci alespoň částečně zrobustnil vůči reálným podmínkám. Ověřil jsem, že rekonstrukce ruky funguje uspokojivě, i když byla pro trénování modelu použita pouze uměle vygenerovaná data. Rekonstrukce probíhá v reálném čase, velice plynule a s minimální odezvou. Jak jsem se osobně přesvědčil, míra plynulosti a velikost odezvy se zejména odvíjí od výpočetního výkonu grafické karty. Odhad natočení ruky funguje dobře pro intervaly, které jsou uvedeny v trénovacích datech. Ukázalo se, že rekonstrukce zbylých částí ruky funguje dokonce i pro natočení ruky, která nejsou v trénovacích datech. Celkově však odhad postrádá potřebný detail v řádu milimetrů, například pro dotyky prstů.

Vhodným řešením pro zvýšení přesnosti rekonstrukce by byla následná aplikace generativní metody. V závislosti na použitém hardwaru zde totiž zůstává určitý výkonnostní prostor. Jsem přesvědčen, že pomocí generativní metody by se podařilo zrekonstruovat velice detailní pózu. Domnívám se, že dalšího zlepšení by bylo rovněž dosaženo zahrnutím reálných snímků do trénovacích dat. Respektive model nejprve natrénovat na datech umělých a posléze provést dotrénování na datech reálných. Celkově je výsledek rekonstrukce mimo jiné velice závislý na přesnosti segmentace, které bylo v tomto případě docíleno díky barevné rukavici a pozadí.

Literatura

[1] PREMARATNE, Prashan. *Human computer interaction using hand gestures*. Springer Science & Business Media, 2014.

[2] ELKOURA, George; SINGH, Karan. *Handrix: animating the human hand*. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association, 2003. p. 110-119.

[3] WHEATLAND, Nkenge, et al. *State of the art in hand and finger modeling and animation*. In: Computer Graphics Forum. 2015. p. 735-760.

[4] GIRSHICK, Ross, et al. *Region-based convolutional networks for accurate object detection and segmentation*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2016, 38.1: 142-158.

[5] SCHROFF, Florian; CRIMINISI, Antonio; ZISSERMAN, Andrew. *Object Class Segmentation using Random Forests*. In: BMVC. 2008. p. 1-10.

[6] WANG, Robert Y.; POPOVIĆ, Jovan. *Real-time hand-tracking with a color glove*. ACM transactions on graphics (TOG), 2009, 28.3: 63.

[7] SHOTTON, Jamie, et al. *Real-time human pose recognition in parts from single depth images*. Communications of the ACM, 2013, 56.1: 116-124.

[8] NEVEROVA, Natalia, et al. *Hand segmentation with structured convolutional learning*. In: Computer Vision–ACCV 2014. Springer International Publishing, 2014. p. 687-702.

[9] OIKONOMIDIS, Iason; KYRIAZIS, Nikolaos; ARGYROS, Antonis A. *Efficient model-based 3D tracking of hand articulations using Kinect*. In: BmVC. 2011. p. 3.

[10] KANG, Byeongkeun; LEE, Yeejin; NGUYEN, Truong Q. *Efficient Hand Articulations Tracking Using Adaptive Hand Model and Depth Map*. In: Advances in Visual Computing. Springer International Publishing, 2015. p. 586-598.

[11] SRIDHAR, Srinath, et al. *Fast and robust hand tracking using detection-guided optimization*. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. p. 3213-3221.

[12] KESKIN, Cem, et al. *Real time hand pose estimation using depth sensors*. In: Consumer Depth Cameras for Computer Vision. Springer London, 2013. p. 119-137.

[13] KESKIN, Cem, et al. *Hand pose estimation and hand shape classification using multi-layered randomized decision forests*. In: Computer Vision–ECCV 2012. Springer Berlin Heidelberg, 2012. p. 852-863.

[14] TANG, Danhang, et al. *Latent regression forest: Structured estimation of 3d articulated hand posture*. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014. p. 3786-3793.

[15] SUN, Xiao, et al. *Cascaded hand pose regression*. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. p. 824-832.

[16] TOMPSON, Jonathan, et al. *Real-time continuous pose recovery of human hands using convolutional networks*. ACM Transactions on Graphics (TOG), 2014, 33.5: 169.

[17] RIEGLER, Gernot, et al. *A Framework for Articulated Hand Pose Estimation and Evaluation*. In: Image Analysis. Springer International Publishing, 2015. p. 41-52.

[18] OBERWEGER, Markus; WOHLHART, Paul; LEPETIT, Vincent. *Hands deep in deep learning for hand pose estimation*. arXiv preprint arXiv:1502.06807, 2015.

[19] SHARP, Toby, et al. *Accurate, robust, and flexible real-time hand tracking*. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, 2015. p. 3633-3642.

[20] MCCULLOCH, Warren S.; PITTS, Walter. *A logical calculus of the ideas immanent in nervous activity*. The bulletin of mathematical biophysics, 1943, 5.4: 115-133.

[21] HEBB, Donald Olding. *The organization of behavior: A neuropsychological approach*. John Wiley & Sons, 1949.

[22] ROSENBLATT, Frank. *The perceptron: a probabilistic model for information storage and organization in the brain*. Psychological review, 1958, 65.6: 386.

[23] POULTNEY, Christopher, et al. *Efficient learning of sparse representations with an energy-based model*. In: Advances in neural information processing systems. 2006. p. 1137-1144.

[24] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. *Imagenet classification with deep convolutional neural networks*. In: Advances in neural information processing systems. 2012. p. 1097-1105.

[25] BENGIO, Yoshua, et al. *Scaling learning algorithms towards AI*. Large-scale kernel machines, 2007, 34.5.

[26] GLOROT, Xavier; BENGIO, Yoshua. *Understanding the difficulty of training deep feedforward neural networks*. In: International conference on artificial intelligence and statistics. 2010. p. 249-256.

[27] HE, Kaiming, et al. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. In: Proceedings of the IEEE International Conference on Computer Vision. 2015. p. 1026-1034.

[28] ORR, Genevieve B.; MÜLLER, Klaus-Robert (ed.). *Neural networks: tricks of the trade*. Springer, 2003.

[29] IOFFE, Sergey; SZEGEDY, Christian. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. arXiv preprint arXiv:1502.03167, 2015.

[30] NAGI, Jawad, et al. *Max-pooling convolutional neural networks for vision-based hand gesture recognition*. In: Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on. IEEE, 2011. p. 342-347.

[31] SUPANCIC III, James Steven, et al. *Depth-based hand pose estimation: methods, data, and challenges*. arXiv preprint arXiv:1504.06378, 2015.

[32] *Blender - a 3D modelling and rendering package, 2015*. , Blender Institute, Amsterdam : Blender Foundation, http://www.blender.org.

[33] Šarić, Marin, 2011, *LibHand: A Library for Hand Articulation*. http://www.libhand.org.

[34] LACHAT, E., et al. *First experiences with kinect v2 sensor for close range 3d modelling*. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2015, 40.5: 93.

[35] JIA, Yangqing, et al. *Caffe: Convolutional architecture for fast feature embedding*. In: Proceedings of the ACM International Conference on Multimedia. ACM, 2014. p. 675-678.

[36] COLLOBERT, Ronan; KAVUKCUOGLU, Koray; FARABET, Clément. *Torch7: A matlab-like environment for machine learning*. In: BigLearn, NIPS Workshop. 2011.

[37] BERGSTRA, James, et al. *Theano: a CPU and GPU math expression compiler*. In: Proceedings of the Python for scientific computing conference (SciPy). 2010. p. 3.

[38] *Qt*, 2016. , Espoo, Finland : Qt Company.

[39] BRADSKI, Gary, et al. textitThe opencv library. Doctor Dobbs Journal, 2000, 25.11: 120-126.