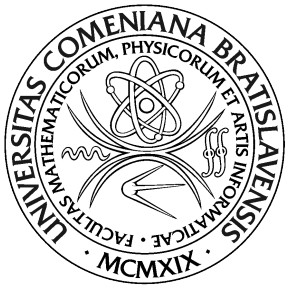
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

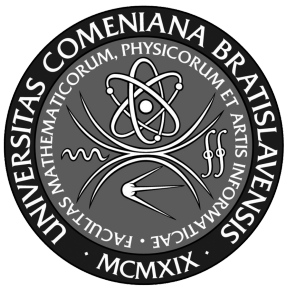


DETEKCIA 3D PÓZY RUKY ZO STEREOSKOPICKÉHO ZÁZNAMU

Diplomová práca

2020 Bc. Tomáš Bordáč

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



DETEKCIA 3D PÓZY RUKY ZO STEREOSKOPICKÉHO ZÁZNAMU

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. Viktor Kocur

Bratislava, 2020 Bc. Tomáš Bordáč

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Tomáš Bordáč
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Detekcia 3D pózy ruky zo stereoskopického záznamu
3D hand pose estimation based on stereo vision

Anotácia: Spoľahlivá detekcia 3D pózy ruky je dôležitým predpokladom pre mnohé aplikačné oblasti. Jednou z nich interakcia ľudského učiteľa s robotom. Cieľom tejto práce je navrhnúť a implementovať vhodnú metódu pre tento účel za využitia RGB stereozáznamu.

Cieľ: 1. Spracovanie prehľadu súčasných možností detekcie 3D pózy ruky z RGB obrazu s využitím jednej a viacerych kamier.
2. Porovnanie vybraných metód pre aplikačnú oblasť interakcie robota s ľudským učiteľom.
3. Implementácia a vyhodnotenie systému na detekciu 3D pózy ruky učiteľa zo stereoskopického záznamu.

Vedúci: Ing. Viktor Kocur
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 17.09.2018

Dátum schválenia: 03.10.2018 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa.

. . . . . .

Bratislava, 2020 Bc. Tomáš Bordáč

Poďakovanie

Podakovanie

Abstrakt

Táto práca sa venuje problematike ...

Kľúčové slová: odhadovanie pozy ruky, sledovanie ruky

Abstract

This thesis deals with ...

Keywords: hand pose estimation, hand tracking

Obsah

Úvod

Kapitola 1

Prehľad problematiky

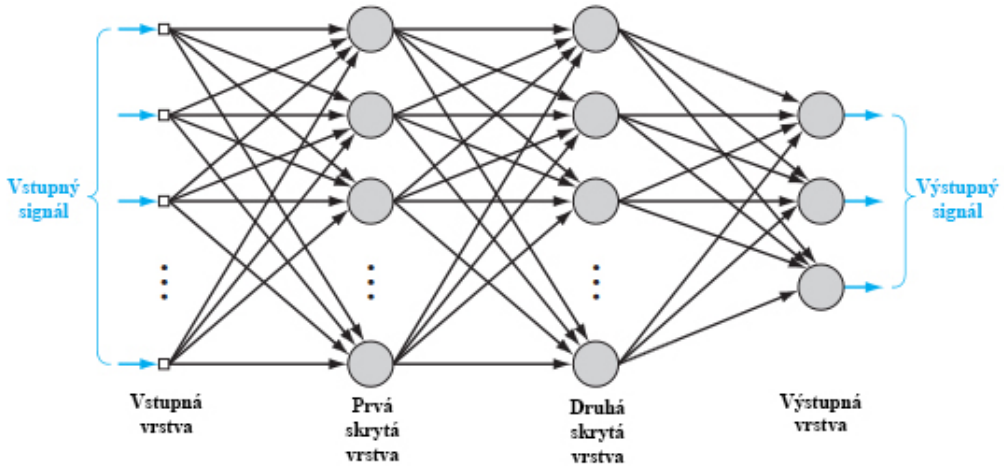
Prístup na určovanie pozície ruky môže byť rôzny v závislosti na vstupných dátach. Sú to obrázky buď z jednej kamery, RGB z viacerých zosynchronizovaných kamier alebo RGB-D. Keďže sa zaoberáme oblasťou interakcie robota s ľudským učiteľom, informácia v 3D priestore má prínos pre širšie vnímanie prostredia robotom. Táto informácia je poskytnutá iba pri RGB-D dátach a pri ostatných vstupoch je potrebné informáciu o hĺbke dopočítať. Týmto prístupom sa venovalo niekoľko prác, ktoré prichádzajú s riešeniami k problémom.

Táto časť obsahuje základné informácie, ktoré sú potrebné na pochopenie a prácu v danej problematike. Na úvod popíšeme základný princíp neurónových sietí, konvolučných sietí a ich niektorých vrstiev a popíšeme architektúru hourglass použitú v praktickej časti. Ďalej vysvetlíme 3 rôzne pohľady prístupov na riešenie našej problematiky. V závere tejto kapitoly uvedieme informácie o datasetoch a kamere použitej v praktickej časti.

1.1 Neurónové siete

Myšlienka neurónových sietí vychádza z princípu výpočtov mozgu, ktorý funguje inak ako bežné počítače. Mozog je v podstate komplexný, nelineárne, paralelný počítač a má schopnosť udržať konzistentnú štruktúru neurónov. Dokáže vykonať isté výpočty mnohonásobne rýchlejšie ako najrýchlejší počítač.

Úlohou umelej neurónovej siete je modelovať riešenie problémov spôsobom podobným mozgu. Sieť je obvykle implementovaná elektronickými komponentami alebo je simulovaná softvérom. Je zložená z umelých neurónov a množstvom prepojení medzi nimi, často označovanými ako synaptické váhy. Druh sietí z ktorého vychádzajú aj konvolučné neurónové siete popísané v sekcii 1.2 využíva proces učenia na to, aby boli ich výsledky použiteľné. Učenie je v tejto oblasti reprezentované funkciou, ktorá upravuje váhy siete tak, aby minimalizovala rozdiel medzi požadovaným výstupom a výstupom z neurónovej siete. Takáto sieť je ilustračne znázornená na obr. 1.1, kde sú čiernymi šípkami znázornené synaptické váhy.



Obr. 1.1: Graf siete s architektúrou zvanou viacvrstvový perceptrón [4].

V tejto časti popíšeme jednoduchú sieť na klasifikáciu. Architektúra siete je ilustračne znázornená na obr. 1.1, kde vstupný signál je vektor príznakov, ktorý chceme klasifikovať. Výstupný signál predstavuje vektor ohodnotení, ktorý je vypočítaný zo vstupného signálu a aplikovaním dopredného výpočtu. Vektor ohodnotení vyjadruje určenie triedy, do ktorej patrí vstupný signál. Po určení triedy je vypočítaná chyba. Pri klasifikácii sa určí chyba podľa správnosti klasifikácie vstupného signálu ako počet správne klasifikovaných príkladov vydelený počtom všetkých príkladov. Výsledok vyjadruje chybu v percentách. Následne je aplikovaný proces učenia. Tento postup použitia sietí je nazývaný ako učenie s učiteľom [4].

Dopredný výpočet prebieha zo vstupného signálu cez všetky vrstvy a vypočíta výstupný signál. Smer výpočtu je znázornený čiernymi šípkami na obr. 1.2. V tejto fáze sú váhy pevne dané a neupravujú sa. Vrstvy sú poprepájané tak, že výstup predchádzajúcej vrstvy je vstupom nasledujúcej vrstvy. Na každej vrstve prebieha nasledovný výpočet (1.1), kde x_i je vstupom do vrstvy, w_{ij} je váha medzi vstupom a neurónom, a f je aktviačná funkcia ako napríklad $sigmoid, tanh, ReLU$.

$$f\left(\sum x_i \cdot w_{ij}\right) \tag{1.1}$$

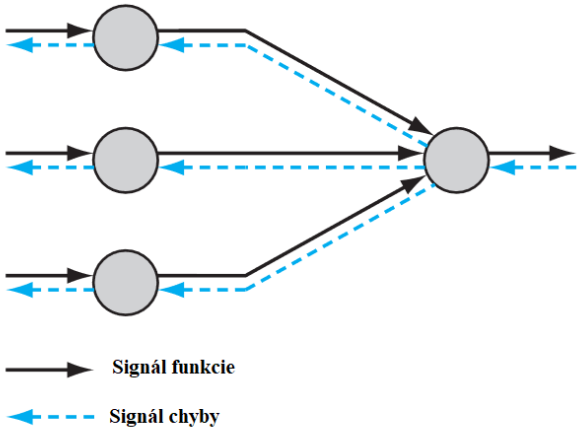
Proces učenia využíva učiaci algoritmus, ktorý pracuje na princípe spätného šírenia chyby zvanom **back propagation** [4]. Tento postup je smerovaný z výstupnej vrstvy, cez všetky skryté vrstvy až ku vstupnej. Smer je znázornený na obr. 1.2 modrými šípkami. Váhy sú upravené na každej vrstve podľa nasledovného výpočtu [4]:

$$w_{ji}(n) = w_{ji}(n) - \eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{1.2}$$

Kde η je rýchlosť učenia, parciálna derivácia vyjadruje gradient a mínus preto, aby sme zmenšovali chybu, teda posunuli výsledok proti smeru gradientu. Chyba je vyjadrená priemerným rozdielom medzi očakávaným a predikovaným výstupom pre jednu dávku vstupných príkladov. Matematicky zapísané rovnicou:

$$E(n) = \frac{1}{n} \sum_{i=1}^{n} l(f(x) - y) \tag{1.3}$$

Kde uvažujeme funkciu f, ktorá na základe váh vypočíta výstupnú hodnotu a funkciu l, ktorá definuje meranie vzdialenosti medzi očakávaným a predikovaným výstupom. Zvyčajne je počítaná ako euklidovská, prípadne tzv. Manhattanská vzdialenosť. Táto metóda sa nazýva **stochastic gradient descent** [1].



Obr. 1.2: Ilustrácia dopredného (výpočet funkcie) a spätného (výpočet chyby) šírenia signálu [4].

1.2 Konvolučné siete

Konvolučné siete [3] tiež nazývané konvolučné neurónové siete alebo CNN sú špeciálnym typom neurónových sietí pre spracovanie dát, ktoré majú známu mriežkovú topológiu. Príkladom takýchto dát sú dáta v časovej sérii, ktoré sú reprezentované vo vektore s časovou informáciou. Ďalším príkladom sú obrázky reprezentované v matici (tenzore) pixelov. Názov naznačuje, že v takej sieti je použitá matematická operácia konvolúcia. Sú to teda neurónové siete, ktoré používajú konvolúciu pri všeobecnom maticovom násobení aspoň na jednej vrstve neurónovej siete. Konvolúcia je matematická operácia medzi dvoma funkciami s argumentami hodnôt na reálnych číslach. Môže byť definovaná [3] spojitá (1.4) alebo diskrétna (1.5) konvolúcia, kde sú x a w funkciami a a, t hodnotami reálnych čísel. Označuje sa hviezdičkou 1.6.

$$s(t) = \int x(a)w(t-a)da \tag{1.4}$$

$$s(t) = \sum_{a=-\inf}^{\inf} x(a)w(t-a) \tag{1.5}$$

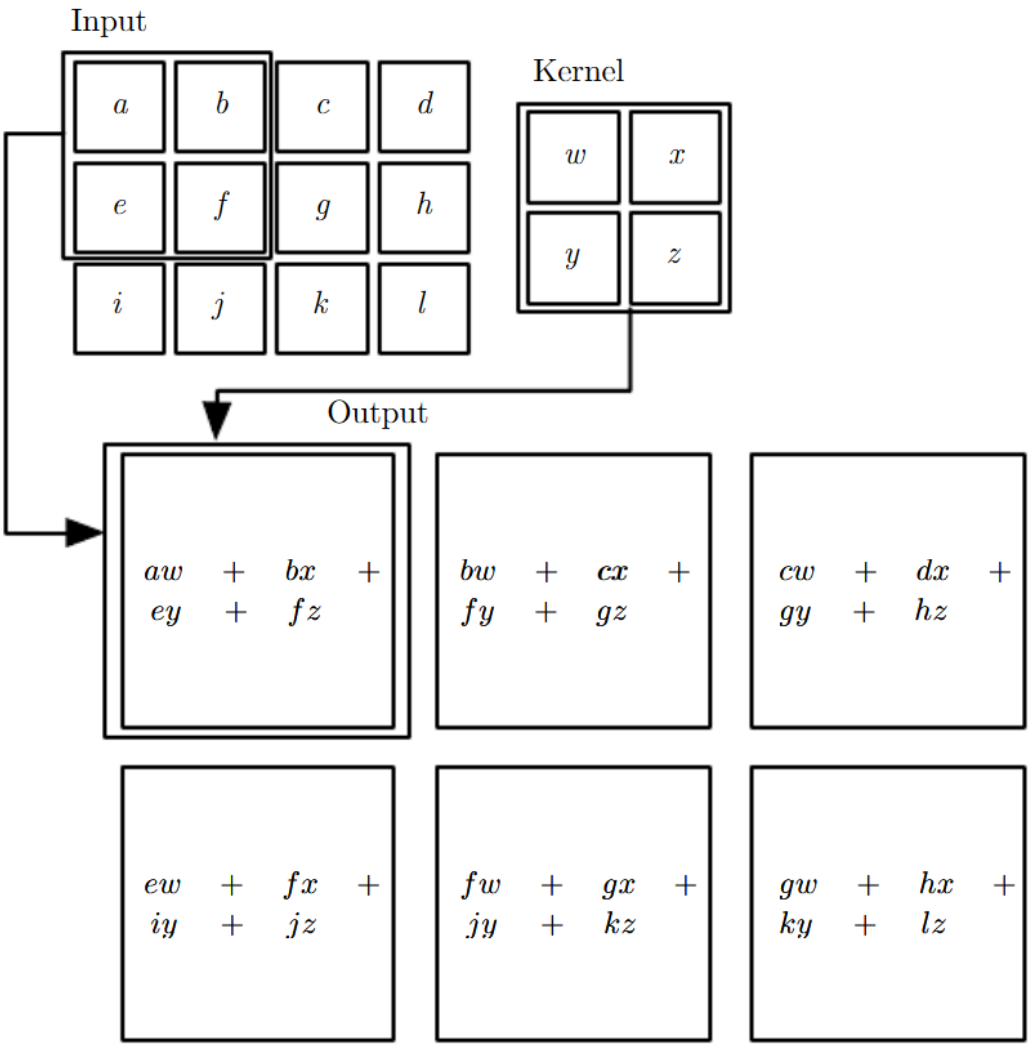
$$s(t) = (f * g)(t) \tag{1.6}$$

Kde f a g sú funkcie a t je premenná na množine reálnych čísel. V terminológii konvolučných neurónových sietí je prvý argument (v tomto prípade f) konvolúcie označovaný ako vstup a druhý argument (g) ako **jadro**. Výstup z konvolúcie je označením pre **mapu príznakov**. Vstup je obvykle viacrozmerné pole dát a kernel viacrozmerné pole parametrov, ktorý sa upravuje počas učenia. Viacrozmerné polia nazývame tenzor.

1.2.1 Motivácia

Využtie operácie konvolúcie namiesto bežného maticového násobenia, ako je to napríklad v prípade plne prepojenej siete, prináša tri zásadné výhody. Neuróny v sieti nie sú všetky navzájom poprepájané, teda je v sieti riedka interakcia medzi neurónmi, aplikuje sa zdielanie parametrov a zabezpečuje ekvivarianciu voči translácii.

Tradičné vrstvy neurónových sietí používajú maticové násobenie medzi maticou parametrov danej vrstvy a parametrami popisujúcimi interakciu medzi vstupom a prislúchajúcim výstupom. Konvolučné siete majú **riedku interakciu** pri použití kernelu menšieho ako vstup. Napríklad, pri spracovaní obrázka môže mať vstup veľkosť tisíce alebo milióny pixelov, ale môžeme detegovať malé významné vlastnosti ako napríklad hrany s kernelom o veľkosti desiatkov pixelov. Teda stačí uchovať menej parametrov čo zníži nárok na potrebnú veľkosť pamäte pre model. Tiež z toho vyplýva, že na výpočet výstupu stačí menej operácií. Ak máme m vstupov a n výstupov, na maticové násobenie je potrebných $m * n$ parametrov. Ak určíme limit počtu prepojení výstupu na k, potom stačí $k * n$ parametrov.



Obr. 1.3: Príklad 2D konvolúcie s nastaveným jadrom [3], ktorý sa celý nachádza v obrázku. Niekedy nazývaná ako "valid" konvolúcia. Štvorce so šípkami zobrazujú, ako bol vypočítaný horný ľavý element výstupného tenzoru po aplikovaní jadra na príslušný horný ľavý región vstupného tenzora.

V neurónových sieťach je každý element matice váh použitý práve raz pre výpočet výstupu danej vrstvy. Konvolučné neurónové siete majú **zdielané parametre**, čo odkazuje na použitie toho istého parametra vo viac ako jednej funkcii modelu. Namiesto toho, aby sa vypočítala množina parametrov

pre každú pozíciu kernelu (každý pixel kernelu je použitý na každom pixeli vstupu), je počítaná jedna množina parametrov pre všetky pozície kernelu.

Forma zdielanaia parametrov pridáva vrstve vlastnosť nazývanú **ekvivariancia**. Funkcie sú ekvivariantné práve vtedy, keď zmenou vstupu sa rovnakým spôsobom zmení aj výstup. Ak máme funkciu $f(x)$ ktorá je ekvivariantá k funkcii $g(x)$, potom musí platiť $f(g(x)) = g(f(x))$. Pre konvolúciu platí: nech g je akákoľvek funkcia, ktorá upraví daný vstup posunom, potom konvolučná funkcia je ekvivariantná ku g. Napríklad, nech máme funkciu I, ktorá vráti jas obrázka v koordináoch celých čísel. Nech máme funkciu g, ktorá mapuje jednu obrázkovú funkciu na inú, teda $I' = g(I)$ je funkcia s $I'(x, y) = I(x - 1, y)$. Takto je posunutý každý pixel obrázka o jednu pozíciu do prava. Ak aplikujeme túto zmenu na I a potom aplikujeme konvolúciu, výsledok bude rovnaký ako pri aplikácii v opačnom poradí, teda konvolúcia na I' a potom zmenu g na výstup konvolúcie. Konvolúcia tvorí teda 2-D mapu výskytu istých vlastností (napr. hrany objektov na obrázku) vstupu. Ak posunieme objekt na vstupe o nejakú hodnotu, výstup bude tiež posunutý o tú istú hodnotu. Pri spracovaní obrázka je napríklad vhodné detegovať hrany na prvej vrstve konvolučnej siete. Rovnaké hrany sa totiž budú vyskytovať v celom obrázku, teda je vhodné zdielať tento parameter po celom obrázku. Môže nastať prípad, keď nie je vhodné zdielať takýto parameter. Príkladom je spracovanie vystrihnutého obrázka a vycentrovaného na tvár, pravdepodobne budeme chcieť rôzne príznaky na rôznych miestach.

Konvolúcia nieje prirozdene ekvivariantná voči niektorím zmenám ako napríklad škálovanie alebo otočenie obrázka. Preto je potrebné aplikovať iné techniky, ako napríklad tzv. augmentácia, na zabezpečenie invariancie voči týmto zmenám. Pri augmentácii ide o úpravu vstupných obrázkov spomínanými zmenami. Takto upravené obrázky sú pridané do vstupného datasetu.

1.2.2 Poolingová vrstva

Typický modul konvolučnej siete pozostáva z troch stupňov. Prvým je vrstva vykonávajúca paralelne niekoľko konvolúcií a výstupom je množina lineárnych aktivácií. Následne v druhom stupni je na každú lineárnu aktiváciu aplikovaná nelinárna aktivácia ako napríklad usmerňujúca lineárna aktivačná funkcia. Poolingová funkcia na úpravu výstupu vrsty na ďalšie spracovanie je tretí stupeň.

Poolingová funkcia nahrádza výstup vrstvy na istých miestach štatistickým zhrnutím susedných výstupov (pixelov v prípade spracovania obrazu). Používanými funkciami sú **max pooling**, ktorá aplikuje funkciu nájdenia maximálnej hodnoty medzi susedmi v obdĺžniku daných rozmerov, **average pooling**, ktorá na výstupnú pozíciu uloží priemernú hodnotu susedov v obdĺžniku, **L2 norma** susedov v obdĺžniku alebo **váhovaný priemer** na základe vzdialenosti od centrálneho pixelu.

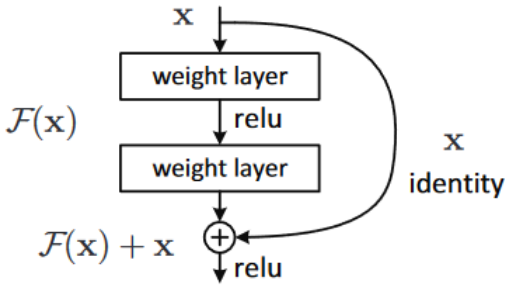
Na poolingovej vrste sa zvolí jedna z vyššie popísaných funkcií, veľkosť obdĺžnika pre oblasť susedov a krok s ktorým sa obdĺžnik posúva. Daná funkcia je aplikovaná pre každý pixel vstupného obrázka a každý kanál priložením centra obdĺžnika na každý pixel a výpočtom funkcie. Výsledok je uložený príslušnému kanálu na pozíciu centra obĺžnika. Táto vrstva zachováva hĺbku vstupu a zmenšuje šírku a výšku obrázka v závislosti na kroku posunu obdĺžnika.

Vďaka tejto vrstve je reprezentácia invariantná k malým zmenám vstupu. Teda ak sa vstup zmení o malú hodnotu, výstup z poolingovej vrstvy zostáva nezmenený. Invariancia voči lokálym zmenám je použiteľná v prípade, že nás zaujíma výskyt nejakého prízaku a nie jeho umiestnenie v obrázku. Napríklad keď potrebujeme zistiť, či sa na obrázku nachádza tvár, stačí vedieť, že sa oko vyskytuje na ľavej a na pravej strane tváre. [3]

1.2.3 Reziduálne učenie

Kaiming He a kol. [5] uvažovali hypotézu, že je jednoduchšie optimalizovať tzv. reziduálnu funkciu než pôvodnú funkciu. Ako extrémny prípad, keď je mapovanie identity optimálne, tak je jednoduchšie potlačiť reziduálnu funkciu k nule ako použiť niekoľko nelineárnych funkcií aproximujúcich mapovanie identity.

Uvažujme $H(x)$ ako výstup z niekoľkých po sebe idúcich vrstiev, kde x označuje vstup do prvej vrstvy. Ak existuje hypotéza, ktorá viacerými nelineárnymi vrstvami aproximuje zložitú funkciu, tak potom existuje ekvivalentná hypotéza, ktorá aproximuje reziduálnu funkciu $H(x) - x$. Pričom má vstup aj výstup rovnaké dimenzie. Teda namiesto aproximácie $H(x)$ bude aproximovaná reziduálna funkcia $F(x) := H(x) - x$. Pôvodnú funkciu dostávame výpočtom $F(x) + x$.



Obr. 1.4: Reziduálny blok [5]

Formulácia $F(x) + x$ môže byť realizovaná doprednov neurónovou sieťou s pridaním "skratky"(Obr. 1.4). Spojenia skratkami môžu vynechať jednu alebo viac vrstiev. Vo vetve skratky je použité mapovanie identity, no môže obsahovať ďalšie vrstvy. Výstupy vrstiev sú následne sčítané. He a kol. výskumom ukázali, že siete s reziduálnymi blokmi dosiahli vyššiu presnosť rýchlejšie ako základný model a zvýšením hĺbky siete bola presnosť ešte vyššia.

1.2.4 Architektúra Hourglass

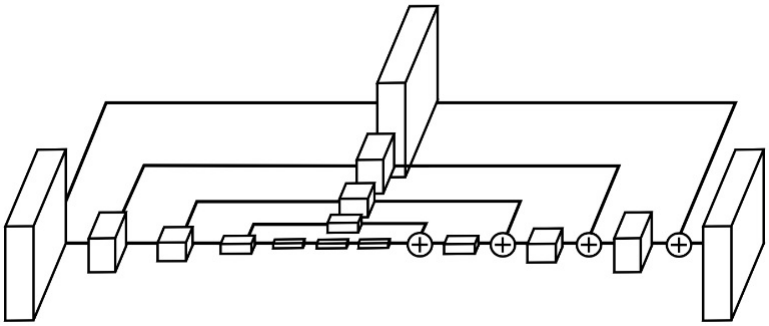
Návrh hourglass [11] je motivovaný potrebou zachytenia informácií v každej veľkosti vstupných obrázkov. Niektoré rozmery vrstiev rozpoznávajú príznaky ako napríklad jednotlivé časti tela človeka. Stretávame sa aj s úlohami, kde je potrebné vedieť širší kontext. Napríklad orientácia človeka, usporiadanie končatín a vzťahy medzi susedními kĺbmi. Takéto súvislosti sú najlepšie rozpoznateľné na rôznych škálach obrázka. Hourglass je jednoduchá sieť s minimalistickým návrhom, ktorý má kapacitu zachytiť všetky príznaky a navzájom ich pospájať. Výstupom z takejto siete je pixelová predikcia pravdepodobností.

Pri tejto architektúre je potrebné efektívne spracovať a spojiť príznaky zo všetkých škál. Niektoré prístupy to riešia tak, že nezávisle spracujú obrázok v rôznych škálach a neskôr výstupné príznkay skombinujú. Vďaka reziduálnym vrstvám v hourglass je možné zachovať priestorovú informácia aj pri spracovaní obrázka vo všetkých škálach postupne za sebou.

Postupnosť vrstiev v hourglass je nasledovná: Konvolučné a max pooling vrstvy sú použité na spracovanie príznakov až k najnižšiemu rozlíšeniu obrázka. Na každom max pooling kroku sa sieť rozvetví a aplikuje viac konvolučných vrstiev na obrázok s rozlíšením pred poolingovou vrstvou. Po dosiahnutí najnižšieho rozlíšenia nasleduje upsampling a kombinovanie príznakov na všetkých škálach. Spájanie výstupov z dvoch susedných vrstiev s rôznym rozlíšením je popísané nasledovným postupom podľa Tompon a kol. [15]. Metóda upsampling podľa najbližšieho suseda je aplikovaná na obrázok s menším rozlíšením. Výsledok sa sčíta po pixeloch s pôvodne väčšou vrstvou, ktorá bola vytvorená poolingovími vrstvami zo vstupu. Topológia siete je symetrická. Teda pre každú vrstvu, ktorá sa zmenšuje existuje vrstva, ktoré je vytvorená zväčšením.

Po dosiahnutí výstupného rozlíšenia zo siete sú ešte dve konvolučné vrstvy s jadrom 1x1. Vyprodukujú finálnu predikciu siete formou heatmáp, ktoré reprezentujú pravdepodobnosť výskytu klúčových bodov v každom pixeli. Celý modul je ilustrovaný na Obr. 1.5

Nevell a kol. [11] navrhli implementáciu pri ktorej dospeli k zisteniam, aké možnosti nastavenia vrstiev sú vhodnejšie ako iné. Práca ukázala, že redukčný krok s konvolúciou 1x1 a použitie menších filtrov napomohli k zachyteniu väčšieho priestorového kontextu. Napríklad bolo vhodnejšie použiť dva 3x3 filtre ako jeden 5x5.



Obr. 1.5: Návrh architektúry hourglass [11]

1.3 Detekcia pózy ruky konvolučnými sieťami

V tejto časti popíšeme možnosti pohľadov na riešenie detekcie pózy ruky. Môžeme sa na tento problém pozrieť z pohladu detekcie objektov alebo ako klasifikačný alebo regresný problém. Implementáciou a bližším popisom konkrétnych metód sa budeme venovať v kapitole 2.

1.3.1 Detekcia objektov

Vo všeobecnosti ide pri detekcii objektov o algoritmus, ktorý vytvorí zoznam kategórií objektov nachádzajúcich sa na obrázku a označí všetky nájdené objekty bounding boxom. [13]

Konvolučná neurónová sieť sa naučií pravdepodobnostnú distribúciu jednotlivých kĺbov. Na výstupe z natrénovaného modelu sú heat mapy, ktoré obsahujú diskrétne pravdepodobnostné hodnoty kĺbových pozícií. Presnosť odhadovania kĺbov je teda obmedzená rozlíšením heat mapy [17]. Zoznam kategórií je v tomto prípade reprezentovaný jednotlivými kĺbmi.

1.3.2 Regresný prístup

Regresia je typ úloh, ktorý predikuje nejakú numerickú hodnotu pre daný vstup. Na vyriešenie tejto úlohy je použitý učiaci algoritmus s funkciu pre výstup $f : \mathbb{R}^n \mapsto \mathbb{R}$. Ak $y = f(x)$, tak model určí vstupu popísanému vektorom x predikovanú číselnú hodnotu y. [3]

Natrénovaná sieť predikuje pozíciu ruky, ktorá je definovaná niekoľkými kĺbmi ruky. Reprezentácia predikcie pre dvojrozmerný vstup je heat mapa pre každý kĺb a pre trojrozmerný vstup je predikcia reprezentovaná relatívnymi 3D koordinátmi ku zápästnému kĺbu. [9]

1.3.3 Klasifikačný prístup

Klasifikácia odpovedá na otázku, do ktorej kategórie patria vstupné dáta. Riešením takejto úlohy je obvykle použitie učiaceho algoritmu s funkciou $f : \mathbb{R}^n \mapsto \{1, ..., k\}$. Ak $y = f(x)$, tak model určí vstupu popísanému vektorom x kategóriu popísanú číselným kódom y. Inou možnosťou je popísať výstup pravdepodobnostnou hodnotou pre každú triedu. [3]

Postupom pri klasifikácii je vytvoriť diskrétny priestor H všetkých možných konfigurácií polohy ruky. Vytvorí sa referenčná databáza D, ktorá obsahuje vybrané pózy ruky $\{h_1, h_2, ...\} \subseteq H$. Každý vstup do databázy D je dvojica konfigurácie ruky h a vstupného obrázku I prislúchajúcemu tejto konfigurácii: $D = \{(h_1, I_1), (h_2, I_2), ...\}$. Problém je zjednodušený týmto spôsobom na klasifikáciu vstupu, ktorý sa s najväčšou pravdepodobnosťou zhoduje s obrázkom v databáze D. Výstupom je potom konfigurácia pózy ruky prislúchajúca tomuto obrázku. [12]

1.4 Datasety

Na trénovanie neurónovej siete pre nájdenie kľúčových bodov ruky môžeme uvažovať tri možnosti datasetov. Existuje ich niekoľko už vytvorených, ktoré obsahujú obrázky aj s anotáciou klúčových bodov ruky na obrázku. Obrázky sú buď skutočné snímky napr. NYUhands [16], alebo vytvorené modely ľudí s anotovanými rukami napr. RenderedHandposeDataset [19]. Treťou možnosťou je vytvorenie vlastného datasetu. V našom prípade kamerou popísanou v časti 1.5 uložiť obrázky a priradiť im anotácie. Anotácie v datasetoch sú uvádzané v súradnicovom systéme obrázka tzv. 'uvd', kde u a v sú súradnice bodu v pixeloch a d vzdialenosť bodu od ohniska kamery, alebo v kamerovom súradnicovom systéme označovanom ako 'xyz'.

Dataset NYUhands [16] obsahuje 72 757 trénovacích obrázkov s hĺbkovou mapou. Každý snímok je zachytený troma Kinect kamerami, ktoré sú nasmerované tak, aby snímali osobu z predu, z ľavej a pravej strany. Obrázky zachytávajú jednu osobu. Testovacie dáta obsahujú 8 252 obrázkov dvoch osôb, jedna je rovnaká ako na trénovacích dátach a druhá iná, ktorú sieť ešte nevidela. Všetky obrázky trénovacej aj testovacej množine sú ano-

tované formou uvd a xyz súradníc.

Dataset RenderedHandpose [19] obsahuje 41 258 trénovacích a 2 728 testovacích obrákov. V datasete sú RGB obrázky, hĺbková mapa, maska segmentov, 21 kľúčových bodov ruky a nastavenia kamery. Maska segmentov vyjadruje triedy pre pozadie, osobu, tri triedy pre každý prst a jednu pre zápästie. Anotácia kľúčových bodov je reprezentovaná formou uv koordinátov, xyz koordinátov a indikátorom viditeľnosti.

1.5 Kamera IntelRealsense d435i

Kamera poskytuje stereoskopický záznam s hĺbkovou mapou a vďaka malým rozmerom je možná jednoduchá integrácia pre množstvo projektov.

Kamera zaznamenáva hĺbkovú mapu výpočtami so stereoskopického záznamu. Implementácia tohoto systému pozostáva z ľavej a pravej IR (infračervenej) kamery a IR projektora. IR vysiela neviditeľný IR lúč, čím zvyšuje hĺbkovú presnosť v obraze. Ľavý a pravý snímač zaznamenávajú scénu a dáta odosielajú na vision procesor, kde je vypočítavaná hodnota hĺbky pre každý pixel v obrázku koreláciou bodov ľavého, pravého snímku a ich vzájomnom posune. Hodnoty hĺbkových pixelov vytvoria hĺbkovú mapu pre jeden snímok.

Hĺbkový stereozáznamový modul má dva kamerové senzory *OmniVision OV2740* s rovnakým nastevením. Poskytuje maximálne rozlíšenie 1920x1080 v pomere 16:9 alebo 1280x800 v pomere strán 8:5. Minimálna vzdialenosť je 280mm, pre ktorú je možné zaznamenať hĺbku s rozlíšení 1280x720. So znižovaním rozlíšenia minimálna vzdialenosť klesá.

Farebný senzor zaznamenáva RGB obrázok. Pridáva informáciu o textúre hĺbkovému senzoru. Použitím textúry s hĺbkovými dátami je vytvorené farebné mračno bodov použité pri rekonštrukcii 3D modelu zo záznamu.

Kapitola 2

Predchádzajúce riešenia

Existuje mnoho prác, ktoré sa zaoberajú sledovaním ruky v obraze. Je veľké množstvo metód, ktoré sa používajú v závislosti na vstupných dátach a dostupných hardvérových prostriedkoch. Väčšina prác uvažuje obraz, na ktorom je ruka ako prvý a najväčší objekt. Postupom času sa výskum rozšíril na vytvorenie systému, ktorý je schopný detegovať pozície niekoľkých rúk na obraze. Veľký pokrok priniesli aj do tejto oblasti konvolučné neurónové siete. Vďaka nim je možné presnejšie určovať pozíciu jednej a viacerých rúk. Ďalším pokrokom bolo rozlišovanie medzi prstami, teda je možné presne párovať kĺb ruky na súradnice v obraze.

V tejto kapitole popíšeme tri práce, ktoré pristupujú k určovaniu pozície ruky rôznymi spôsobmi. Výber týchto prác je vhodný aj preto, že používajú vstupné dáta, ktoré sú v podobnej forme ako dáta z kamery použitej v našej práci. Systémy navrhnuté v týchto troch riešeniach by bolo možné použiť ako učenie robota s ľudským učiteľom, čo prispieva k vyhodnoteniu najlepšieho prístupu a porovnanie pre riešenie v tejto oblasti. Taktiež tieto práce poskytujú implementované riešenia a bolo možné zreprodukovať ich dosiahnuté výsledky.

2.1 Sledovanie ruky z RGB-D dát bez CNN

Sledovanie a určovanie polohy ruky na RGB-D dátach je možné aj bez použitia konvolučných neurónových sietí. Stan Melax a kol. [7] použili vo svojej práci hĺbkové dáta, na ktoré mapovali syntetickú ruku. Použitím správneho predspracovania dát dosiahli úspešné odhadovanie polohy ruky.

Sledovací algoritmu je možné používať v reálnom čase. Sleduje plne definovaný 3D model ruky. Systém využíva predošlú pozíciu ruky, ktorú na ďalšom snímku videa upraví. Syntetický model ruky je definovaný ako konvexné pevné teleso, čo umožňuje priblížiť sa k zachytenej ruke na obraze.

Mračná bodov získané z hĺbkového senzora kamery obsahujú tisícky bodov.Väčšinou sú tieto body namerané správne, ale senzor niekedy zaznamená nesprávnu informáciu. Pri rýchlych pohyboch však nie je možné a ani potrebné, aby bola pozícia modelu upravená podľa každého nameraného bodu. Preto zredukovali tieto dáta na minimálny počet, ktorý jednoznačne definuje pozíciu ruky v obraze. Pre niektoré pixely však vypočíta a vráti nesprávnu vzdialenosť od kamery, čo spôsobuje uloženie ruky v neprirodzených polohách. Odstránením šumu a týchto nesprávnych bodov boli získané lepšie výsledky a sledovací algoritmus bol rýchlejší. Toto zlepšenie bolo dosiahnuté implementáciou pod-vzorkovania voxelov do mriežky. Pre každý voxel bolo vypočítané ťažisko, ktoré reprezentuje skupinu hĺbkových dát.

Model ruky má na začiatku prednastavenú prirodzenú polohu. Využíva vlastnosti pevných telies pri zmene polohy kľúčových bodov na ruke, vďaka čomu sa udrží v prirodzenej póze. Hĺbkové dáta z kamery vplývajú na model tak, že poloha modelu je upravená podľa ľudskej ruky pred kamerou. Pozícia vyššie spomínaného ťažiska voxelu je použitá ako kľúčový bod ovplyvňujúci jednotlivé časti modelu ruky. Úprava syntetickej ruky má magnetické správanie, teda najbližší kľúčový bod syntetickej ruky je posunutý k najbližšiemu

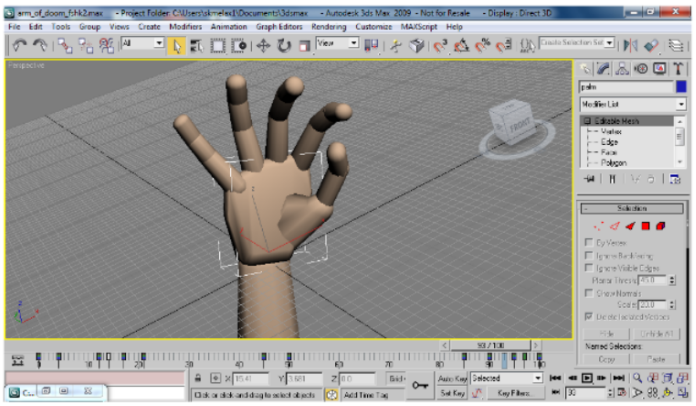
ťažisku.

So senzoru sú získavané aj informácie o miestach, kde sa objekt nenachádza. Tieto oblasti označili ako priestor kolíznych hraníc 3D modelu ruky a zabránili posunu model do týchto oblastí. Pretože malé objekty ako prsty sa môžu pohybovať relatívne rýchlo k ich veľkosti, nepridávali im blokovací kolízny priestor, keďže požiadavkou bol voľný pohyb k dosiahnutiu čo najvyššej presnosti.

Pri tomto systéme občas dochádza k nesprávnej identifikácii prstov. Používateľ môže napríklad ukazovať iba jedným prstom (ukazovákom), ale systém namapuje na hĺbkové dáta iný prst, napr. prostredník alebo palec. Môže nastať aj prípad, že z mračna bodov je potenciál na viac zdvihnutých prstov ale lokálna príťažlivosť modelu na tieto body zabraňuje zdvihnúť alebo schovať nadbytočný prst. Na vyriešenie tohoto problému Stan Meax a kol. [7] vychádzali z predošlého najlepšie priradeného modelu. Pretože najlepšie mapovanie modelu na hĺbkové dáta môže trvať niekoľko snímkov, výber prstov pri taktomto rozhodovaní sa vykoná každú pol sekundu.

2.1.1 Syntetický model ruky

Model ruky vymodelovali z 3 častí pre každý prst a jednej časti dlane. Model je zobrazený na obr. 2.1. Každá časť bola definovaná ako pevné teleso a teda zachováva aj vlastnosti a výpočet kolízií pevných telies. Taktiež majú pevné telesá pridelené kosti, ktoré im upravujú pohyb. Model je reprezentovaný súborom exportovaným z programu 3DS Max. Dôležitá je veľkosť modelu, ktorá bola zvolená na takú, aby systém fungoval s väčšinou bežných dospelých ľudských rúk teda na dĺžku 20cm. Existujú techniky, ktoré automaticky preškálujú model podľa potreby, čo ale nie je cieľom tejto práce a prednastavená jedna veľkosť modelu je dostačujúca pre tento výskum.



Obr. 2.1: Model ruky v 3DS Max

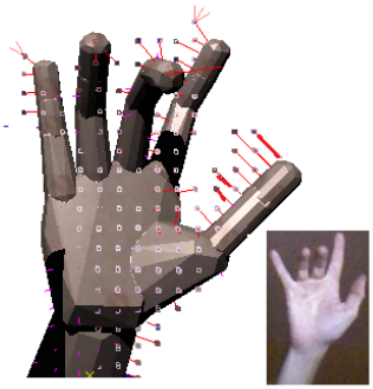
2.1.2 Výpočet chyby

Stan Meax a kol. [7] použili meranie vzdialenosti medzi množinou bodov na modeli ruky a príslušnému bodu na používateľovej ruke. Nakoľko hĺbkové dáta znázorňujúce používateľovu ruku nie sú jednoznačné, nedá sa určiť, ktoré prislúchajú ktorým bodom syntetickej ruky. Známe je, že ak existuje bod v hĺbkových dátach P a je daná vzdialenosť od povrchu sledovaného modelu B, potom model musí byť posunutý aspoň o vzdialenosť, ktorá súhlasí s aktuálnym stavom používateľovej ruky. Vzdialenosť chyby je zobrazená červenými úsečkami na obr. 2.2. Definovanie chyby vychádza z tohoto predpokladu. Teda pre každé pevné teleso B_i v modeli B je nájdená minimálna vzdialenosť o ktorú musí byť posunutý model pre najlepšiu zhodnú pozíciu s hĺbkovými dátami. V hĺbkových dátach je vybrané mračno bodov P_i, ktoré sú najbližšie k povrchu pevného telesa B_i. Priebeh výpočtu je nasledovný:

$$P_i = \{p : p \in P, min\{|p - b_i| : b_i \in B_i\} = min\{|p - b| : b \in B\}\} \qquad (2.1)$$

Následne sa z tejto množiny vyberie najvzdialenejší bod p. Vzdialenosť je potom vypočítaná medzi bodom p a k nem najbližšiemu bodu b z pevného telesa B_i, ku ktorému bolo vybrané príslušné mračno bodov P_i. Celý výpočet tejto chyby môžeme zapísať takto:

$$errFitt_i = max\{min\{|p - b_i| : b_i \in B\} : p \in P_i\} \tag{2.2}$$



Obr. 2.2: Ovplyvnenie 3D modelu ruky hĺbkovými dátami. Vybrané body p z mračna bodov sú znázornené tmavými bodkami, z ktorých vychádzajú červené úsečky.Tie vyjadrujú vzdialenosť medzi bodom p a najbližším bodom modelu ruky b. Túto vzdialenosť vypočítame ako na (2.2)

.

K chybe podľa (2.2) sa pričíta ešte chyba za nesprávne zvolenú pozíciu pevného telesa (2.3). Miesta pre penalizáciu sú tie, kde hĺbkové dáta nezachytili žiaden objekt, teda sa tam v skutočnosti nič nemôže nachádzať. Vzali centroid každej kosti, ktorý sa neprekrýva s pixelmi hĺbkových dát. Chyba je aspoň taká veľká ako polomer najväčšej vpísanej gule.

$$errOcc_i = \begin{cases} r, & \text{ak sa centroid } (B_i) \text{ prekrýva s pozadím} \\ 0, & \text{inak} \end{cases} \tag{2.3}$$

Celková chyba je potom vypočítaná ako súčet chyby a penalizácie každého pevného telesa (2.4). Kde $errFit$ je množina chýb a $errOcc$ je množina penalizácií, kde každá chyba alebo penalizácia prislúcha práve jednému pevnému telesu B_i z celého modelu ruky B.

$$errTotal = \sum err_i \in errFit + \sum err_i \in errOcc \tag{2.4}$$

2.1.3 Vyhodnotenie

Tento systém je schopný sledovať ruku na základe hĺbkových dát a každému kĺbu prideliť pozíciu v 3D priestore. Keďže hĺbkové dáta najbližšie ku kamere sú predpokladom na určenie pozície ruky, tak musí byť prvým objektom pred kamerou ruka aby systém fungoval, čo prináša isté obmedzenia.

Na jeho používanie a sledovanie ruky v reálnom čase stačí jeden procesor pre jednoduchšie pózy. So zvyšujúcim výkonom hardvéru je možné plynule sledovať zložitejšie polohy. Aby bolo možné sledovať viac rúk, systému by stačilo pridať rozdelenie hĺbkových dát do skupín napr. algoritmom ‘k-means clustering", čím by sa odlišovalo medzi rukami.

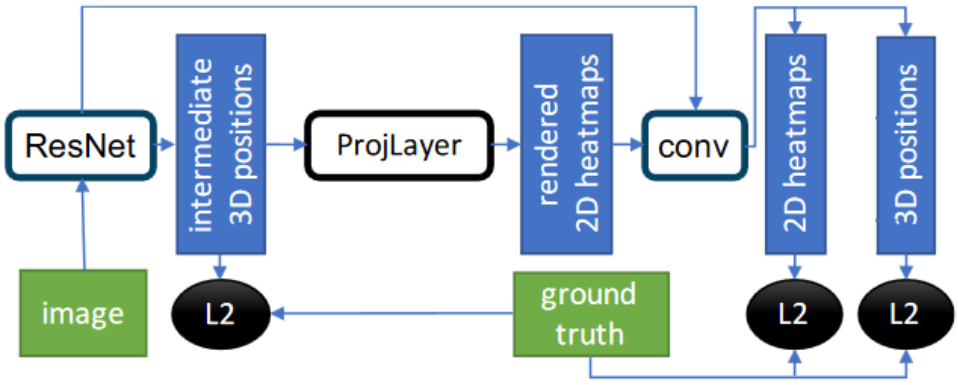
2.2 Sledovanie ruky z RGB dát s CNN

Najbežnejšie rozšírenými vstupnými dátami sú RGB obrázky. Vytvorenie takýchto snímkov je dnes jednoduché a dostupné takmer každému. Problémom je však určiť pozíciu konkrétnych bodov v 3D priestore, pretože nemáme informáciu o vzdialenosti k danému bodu. Navyše nie je možné túto vzdialenosť vypočítať z jedného obrázku. Myšlienkou ako vyriešiť tento problém a získať informáciu o vzdialenosti od kamery je pridanie syntetického modelu sledovaného objektu, ktorý je vložený do obrázka na základe bodov definovaných

v dvoch dimenziách. Nájdením pozície ruky v 3D priestore z jedného RGB obrázka sa zaoberala Franziska Mueller a kol. [9]

Vytvorili systém, ktorý v reálnom čase sleduje polohu ruky z RGB obrázku. Tok videa je posielaný do CNN, ktorá funguje na princípe regresie kĺbov ruky. Táto sieť *RegNet* predikuje 2D a 3D pozíciu 21 kĺbov. 2D pozícia je reprezentovaná tepelnými mapami a 3D pozície sú reprezentované relatívnymi 3D súradnicami voči zápästiu.

RegNet zobrazená na obr. 2.3 využíva reziduálnu sieť, ktorá pozostáva z 10 reziduálnych blokov odvodených z architektúry *ResNet50* [5]. Pre zlepšenie prepojenia 2D a 3D predikcie pridali projekčnú vrstvu *ProjLayer*. Myšlienka za touto vrstvou je vytvoriť tzv. ortografickú projekciu 3D predikcie, z ktorej je vytvorená 2D Gausovská tepelná mapa. Tieto mapy vplývajú na zvyšnú časť siete pre dosiahnutie finálnej 2D a 3D predikcie. V práci ukázali, že táto vrstva prispela ku zlepšeniu výsledkov.



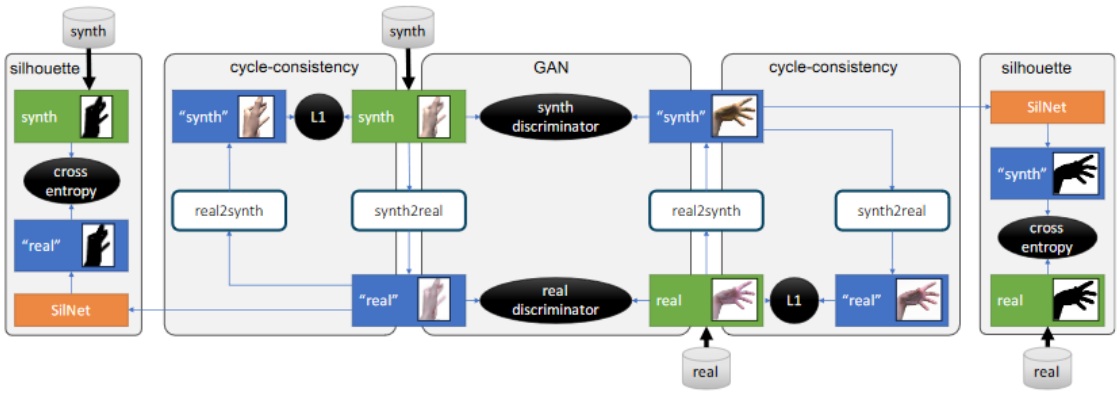
Obr. 2.3: Architektúra *RegNet*[9]. ResNet a conv sú trénovateľné, chyba sa spätne šíri cez ProjLayer. Vstupné dáta sú označené zelenou a generované dáta sieťou sú označené modrou farbou. Časti cenovej funkcie sú zobrazené v čiernych ováloch.

2.2.1 Dataset

Anotovanie rúk v 3D priestore pre skutočné fotografie je veľmi náročné, preto použili vygenerované obrázky rôznych polôh počítačom vytvorenej syntetickej ruky. Výhodou takých obrázkov je známa pozícia kĺbov. Rozdiely medzi skutočnou a syntetickou rukou však limitujú generalizačnú schopnosť CNN. Aby sieť vedela predikovať aj snímky so skutočného sveta, tak sú potrebné trénovacie obrázky aj z reálneho sveta. Dôležité je, aby syntetické a reálne obrázky navzájom nezobrazovali rovnaký snímok.

Aby použili reálne vyzerajúce obrázky a mali aj 3D pozíciu každého kĺbu, vytvorili generátor obrázkov na základe teórie podľa Goodfellow a kol. [3]. Využíva myšlienku teórie hry, v ktorej musí generátor súťažiť voči diskriminačnej sieti. Generovaný je priamo vzorový obrázok a diskriminačná sieť určí pravdepodobnosť, že tento obrázok je skôr skutočným trénovacím vstupom než vygenerovaným obrázkom. Cieľom je maximalizovať pravdepodobnosť, že generovaný obrázok je skutočným vstupom.

Sieť na generovanie obrázkov, ktorej teoretický základ je popísaný vyššie, navrhli [9] tak, aby zo syntetického obrázka vytvorila reálny a opačne. Vstupom do siete sú dvojice syntetickej a reálnej ruky na bielom pozadí. Pre každý obrázok je aplikovaná funkcia na vytvorenie syntetického alebo reálneho páru pre vstup. Kontrolovanie konzistencie prebieha spätným vytvorením pôvodného obrázka a porovnaním L1 normou. Upravený obrázok je vstupom do binárneho klasifikátora, ktorý určuje pre každý pixel, či patrí pozadiu alebo ruke. Takto vytvorená silueta je porovnaná so siluetou pôvodného obrázka krížovou entrópiou. Táto sieť je trénovaná na SynthHand [10] datasete a vlastnom datasete rúk, ktoré sú zachytené na zelenom pozadí. Architektúra je ilustrovaná na obr. 2.4.



Obr. 2.4: Architektúra siete na generovanie obrázkov [9]. V bielych obdĺžnikoch sú trénovateľné časti, čiernou sú označené stratové funkcie, zelenou sú obrázky z datasetu, modrou sú označené vygenerované obrázky a oranžovou binárny klasifikátor.

Natrénovaná sieť je potom použitá na vytvorenie reálne vyzerajúcich obrázkov z rôznych póz vymodelovanej ruky so známimi 3D súradnicami. Syntetickým obrázkom menili pozadie náhodnými textúrami a aj úplne náhodnými farbami. Výstupom zo siete je RGB obrázok s 3D pozíciami kĺbov reálnej ruky.

2.2.2 Model ruky a jeho umiestnenie v obrázku

Získaním predikcie 2D tepelných máp a 3D súradníc z CNN boli schopní [9] umiestniť kinematickú kostru pre tieto dáta. Predikované pozície sú také, že umiestnenie kostry podľa nich zobrazuje polohu, ktorú človek dokáže napodobniť.

Syntetický model ruky je zložený zo zápästia reprezentovaného základným kĺbom (ostatné kĺby majú relatívnu polohu k tomuto kĺbu v 3D súradniciach) a 20 kĺbmi prstov, vrátane končekov prstov, ktoré majú nulový stupeň voľnosti. Ostatných 15 kĺbov prstov má stupeň voľnosti jeden alebo

dva. Absolútna poloha kĺbu je určená ako $\Theta = (t, \mathbf{R}, \theta)$, kde $t \in \mathbb{R}^3$, a $\mathbf{R} \in SO(3)$ je globálna pozícia a rotácia zápästného kĺbu, a $\theta \in \mathbb{R}^{20}$ je vektor uhlov desiatich kĺbov so stupňom voľnosti jedna a piatich kĺbov so stupňom voľnosti dva. Ako globálny súradnicový systém brali do úvahy súradný systém kamery.

Pre správne umiestnenie kostry je úlohou minimalizovať chybu definovanú ako $E(\Theta) = E_{2D}(\Theta) + E_{3D}(\Theta) + E_{limit}(\Theta) + E_{temp}(\Theta)$. Trénovaním 2D tepelnej mapy je účelom minimalizovať vzdialenosť medzi pozíciou kĺbu projektovanú v obraze a maximálnou hodnotou v prislúchajúcej tepelnej mape. Matematicky zapísané na rovnici (2.5).

$$E_{2D}(\Theta) = \sum_j \omega_j ||\Pi(\mathcal{M}_j(\Theta)) - u_j||_2^2 \tag{2.5}$$

Kde $u_j \in \mathbb{R}^2$ označuje maximálnu hodnotu v tepelnej mape pre j-ty kĺb, $\omega_j > 0$ je skalár označujúci váhu odvodenú z tepelnej mapy a $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ je projekcia z 3D priestoru na 2D obrázkovú plochu. Túto projekciu ovplyvňuje nastavenie kamery. Výstup z 3D regresie je relatívnou polohou k zápästiu.

2.2.3 Vyhodnotenie

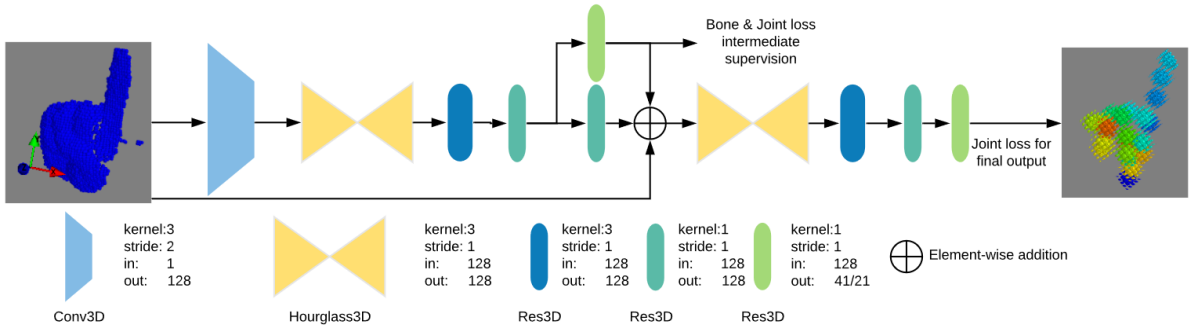
Na vyhodnocovanie presnosti predikcie a porovnanie z inými prístupmi k predikcii ruky v obraze sa počítali percentá správne určených kĺbov. Správne určenie kĺbu je vtedy, ak jeho predikovaná pozícia leží v kružnici pre 2D a v guli pre 3D prípad so stredom v súradniciach očakávanej pozície kĺbu a priemerom 50 mm. Vyhodnocovaním týmto spôsobom na čisto syntetických dátach dosiahli presnosť 55%. Pridaním reálne vyzerajúcich obrázkov a rozdelením trénovacej a testovacej množiny dosiahol presnosť 96.5% na Stereo datasete [18].

2.3 Detekcia 3D polohy ruky s použitím architektúry Hourglass

Ďalšou možnosťou ako pristupovať k problému hľadania polohy ruky v obraze je vytvoriť vhodnú architektúru neurónovej siete tak, aby na vstupe mala jeden hĺbkový obrázok, pre ktorý určí pozície kĺbov ruky v 3D priestore. Tento spôsob využil Fuyang a kol. [6], ktorých návrh a implementáciu popisuje táto podkapitola, pričom architektúru siete vytvorili podľa siete Hourglass na základe výsledkov riešenia podobného problému určovania polohy človeka, ktorým sa zaoberal Newell a kol. [11].

2.3.1 Návrh metódy

Navrhli systém založený na regresnom prístupe (podsekcia 1.3.2), ktorý pracuje s hĺbkovým obrázkom. Vstupom do systému je normalizovaný binárny voxel a na výstupe je 3D tepelná mapa kĺbov v pirestore voxelov. Vstupný obrázok je najprv premietnutý do súradnicového priestoru kamery ako mračno bodov. Následne je toto mračno prevedené na diskrétne binárne voxely normalizované podľa veľkosti mračna bodov. Trénovacím výstupom je 3D Gaussovo rozdelenie pravdepodobností výskytu kĺbov ako tepelná mapa pre každý kĺb. Súčasne je trénovaná sieť na vytváranie tepelnej mapy 3D kostí. Vďaka nej je možné zahrnúť závislosť medi kĺbmi na ruke a kosťami, ktoré tieto kĺby prepájajú. Tento návrh je zobrazený na Obr. 2.5 [6].



Obr. 2.5: Návrh systému s použitím 2 modulov architektúry siete Hourglass

2.3.2 Predspracovanie dát

Vstupné dáta do modulu Hourglass sú upravené do objemovej reprezentácie a sú premietnuté do priestoru kamery podľa vnútorných parametrov kamery. Vytvárajú mračno bodov, ktoré je zjednodušené na mračno bodov ohraničené kvádrom. Na vytvorenie takého ohraničenia sa v tejto metóde používa geometrický stred bodov mračna a rozmery ako najvzdialenejšie body v osiach x, y, z. Potom je mračno bodov prevedené do diskrétneho priestoru binárnej mriežky voxelov. Ak je voxel označený jednotkou, obsahuje hĺbkové dáta inak je označený nulou. Očakávané súradnice pre kĺby vyjadrujú iba jeden bod v priestore. Teda je distribúcia týchto bodov veľmi riedka pre tréning. Zväčšenie množstva týchto bodov je dosiahnuté 3D Gaussovou distribúciou v okolí súradníc kĺbu so stredom v jeho súradniciach a smerodajnou odchýlkou dĺžky mriežky. Táto distribúcia vyjadruje tepelnú mapu očakávaných súradníc. Príklad takejto mapy je znázornený na obr. 2.5.

2.3.3 Štruktúra siete

Základným modulom je Hourglass sieť, ktorej architúru sme popísali v 1.2.4. Podľa návrhu má táto sieť pracovať s 3D dátami, a preto vrstvy v tomto module sú upravené tak, aby používali 3D konvolučné vrstvy. Výpočty týchto vrstiev je náročnejšie vykonať a viac údajov sa musí uložiť do pamäte. Preto je znížené rozlíšenie vstupných voxelov na 64x64x64 a výstupne rozlíšenie z modulu hourglass na 32x32x32. Na vybudovanie siete boli použité dva takéto moduly, kde výstup z prvého modulu je použitý aj na vygenerovanie tepelnej mapy kostí a kĺbov. Táto časť je tvorená z tzv. Res3D modulu s jadrom 3x3 a dvoma Res3D modulmi s jadrom 1x1. Vstupom do druhého hourglass modulu je súčet pôvodného vstupu, výstupu z prvého hourglass modulu a z tepelnej mapy kostí a kĺbov. V reziduálnych blokoch bolo použitých 128 filtrov. Po doprednom behu cez všetky moduly je tepelná mapa kĺbov predikovaná dvoma za sebou spojenými konvolučnými vrstvami s jadrom 1x1x1 tak, aby mala sieť na výstupe požadované rozmery.

2.3.4 Trénovanie

Tréning siete prebieha metódou učenia s učiteľom. To znamená, že je k vstupným dátam prislúchajú očakávané súradnice kĺbov a na výstupe zo siete je možné vypočítať rozdiel medzi skutočnými a predikovanými súradnicami za použitia tzv. stratovej funkcie. Podľa návrhu siete sa okrem učenia súradníc kĺbov sieť učí aj polohu kostry. Výsledná stratová funkcia je ovplyvnená oboma naučenými časťami a počíta sa ako ich súčet, teda $\mathrm{L} = L_j + L_b$. Priebeh výpočtu stratovej funkcie kĺbov (L_j) a funkcie kostí (L_b) je popísaný nižšie.

Učenie polohy kĺbov

Na predikovanie tepelnej mapy voxelov pre každý kĺb bolo potrebné vytvoriť očakávané mapy v rovnakom formáte. V použitých datasetoch je iba jedna očakávaná hodnota pre každý kĺb a preto upravili vstupné dáta spôsobom popísaným v 2.3.2. Takto vytvorené mapy cieľových súradníc sú použité na porovnanie s predikovanými tepelnými mapami.

Chyba predikovaných súradníc sa vypočíta stratovou funkciu priemerných štvorcov (MSE) v závislosti od mapy očakávaných súradníc vytvorených postupom popísaným vyššie. Konkrétny výpočet stratovej funkcie je definovaný nasledovne:

$$L_j = \sum_{s=1}^{S} \sum_{n=1}^{J} \sum_{i,l,k}^{R} |H_{s,n}^{j}(i,l,k) - \bar{H_n^j}(i,l,k)|^2 \tag{2.6}$$

kde $H_{s,n}^{j}(i,j,k)$ vyjadruje predikovanú hodnotu na pozícii i,j,k tepelnej mapy pre n-tý kĺb z s-tého hourglass modulu. $\bar{H_n^j}$ je očakávaná tepelná mapa pre n-tý kĺb. Písmeno S vyjadruje počet hourglass modulov, J počet kĺbov a R rozmer výstupnej tepelnej mapy.

Vzťah medzi kostrou a kĺbmi

Ku zlepšeniu presnosti určovania polohy ruky pomohlo pridanie časti, ktorá upravuje predikciu aj podľa vplyvu kostry. Sieť je navrhnutá tak, aby okrem tepelnej mapy pre kĺby vygenerovala aj tepelnú mapu kostí. Počet kostí je stanovený stromovou štruktúrou ľudskej ruky , teda napríklad ako v datasete NYUHand je 36 kĺbov spojených 35 kosťami. Tepelná mapa kostí je tvorená rovnakým spôsobom ako mapa kĺbov, ktorej postup je popísaný vyššie, s tým rozdielom, že smerodajná odchýlka má polovičnú veľkosť dĺžky mriežky. Výhodou je, že tento vzťah medzi kosťou a kĺbom sa môže naučiť navrhnutá sieť.

Učenie tohoto vzťahu prebieha medzi dvoma hourglass modulmi a stratová funkcia je definovaná nasledovne:

$$L_b = \sum_{s=1}^{S-1} \sum_{n=1}^{B} \sum_{i,j,k}^{R} |H_{s,n}^b(i,j,k) - \bar{H}_n^b(i,j,k)|^2 \tag{2.7}$$

kde $H_{s,n}^b(i,j,k)$ vyjadruje predikovanú hodnotu na pozícii i,j,k tepelnej mapy pre n-tú kosť z s-tého hourglass modulu. $\bar{H}_n^b$ je očakávaná tepelná mapa pre $n-t$ kosť. Písmeno S vyjadruje počet hourglass modulov, B počet kostí a R rozmer výstupnej tepelnej mapy.

2.3.5 Spracovanie výstupu

Vstupné dáta boli prevedené do diskrétneho priestoru v ktorom priemerný rozmer voxelu 32x32x32 predstavuje 8mm. Ak by boli vybrané najpravdepodobnejšie súradnice, chyba spôsobená prevedením do diskrétneho priestoru by mohla byť až $\sqrt{4^2+4^2+4^2} = 6.92mm$. Preto použili jednoduchú metódu na obnovenie pôvodnej pozície kĺbu nasledovným výpočtom (2.8):

$$J_n = \sum_{i=1}^{K} w_i^{(n)} j_j^{(n)} \tag{2.8}$$

$$w_i^{(n)} = \frac{H_n^j(j_i^{(n)})}{\sum_{i=1}^{K} H_n^j(j_i^{(n)})} \tag{2.9}$$

Pre n-tý kĺb je teda výsledná pozícia určená ako vážený priemer prvých K odpovedajúcich voxelov tepelnej mapy n. $j_i^{(n)}$ označuje pozíciu prvých i odpovedajúcich voxelov pre n-tý kĺb a $w_i^{(n)}$ váhu každého voxelu, ktorého súčet je normalizovaný na interval $< 0,1 >$ príslušnou hodnotou $H_n^j(j_i^{(n)})$ v tepelnej mape n-tého kĺbu. Vďaka tejto úprave je chyba spôsobená prevodom do diskrétneho priestoru približne 0.3mm.

2.3.6 Vyhodnotenie

Tento systém vyhodnocovali na dvoch datasetoch MSRA [14] a NYU [16]. Vstupné obrázky do systému boli výrezy jednej ruky, MSRA dataset obsahuje takéto obrázky a z obrázkov v NYU boli tieto výrezy vytvorené.

Ako vyhodnocovaciu metriku použili priemernú vzdialenosť medzi očakávanými a predikovanými súradnicami v 3D priestore. Na datasete NYU sieť predikovala s priemernou chybou 8.9 mm a na datasete MSRA bola táto priemerná chyba 7.4 mm. Táto metóda bola schopná predikovať polohy ruky pre približne 10 obrázkov za jednu sekundu.

2.4 Porovnanie metód

V tejto kapitole sme popísali 3 rôzne metódy, ktorými sa dá pristupovať k určovaniu polohy ruky. V tejto problematike existuje veľa ďalších prác, ale hlavné základy prístupov sú obsiahnuté v týchto popísaných prácach. Výhodou použitia CNN je možnosť nájsť ruku kdekoľvek na obrázku, inak je nutné aby bola prvým objektom pred kamerou.

V prvej práci používali hĺbkové dáta, na ktoré mapovali syntetickú ruku. Výhodou tohto prístupu bez CNN je nezávislosť fungovania od predchádzajúcich obrázkov a nutnosti poznať súradnice kĺbov pred určením polohy.

Druhá práca používala generovanie obrázkov syntetických a reálne vyzerajúcich pomocou CNN. Z RGB obrázka vygenerovali syntetickú ruku so známymi 3D súradnicami kĺbov. Týmto prístupom dosiahli presnosť predikcie do vzdialenosti 25 mm od skutočnej pozície kĺbu.

Tretia práca využila architektúru Hourglass a 3D konvolúciu. Predikovala tak súradnice v 3D a vďaka vhodnej architektúre dosiahli presnosť predikcie do 8.9 mm.

Kapitola 3

Návrh a implementácia

Prvou časťou tejto kapitoly bude návrh systému. Popíšeme z ktorých metód budeme vychádzať, ako ich upravíme a uvedieme predpoklady, prečo by naša metóda mala byť lepšia a vhodná pre úlohu vnímania ľudského učiteľa robotom. Popíšeme riešenie na určovanie 3D pozície kĺbov a navrhneme postup trénovania siete. V druhej časti popíšeme implementáciu. Bude sa venovať vytvoreniu vlastného datasetu z výstupu našej kamery, spracovaniu vstupných dát ako úprava datasetu pre CNN a výstupu z našej kamery. Zaoberať sa bude architektúrou siete, kde popíšeme jednotlivé moduly, z ktorých sa sieť skladá, a nastavenia hyperparametrov. Ďalej uvedieme postup trénovania, teda akými fázami prebiehalo učenie a aké boli pri tom použité výpočty. Na záver tejto kapitoly popíšeme spôsob vyhodnocovania presnosti nášho systému.

3.1 Návrh modelu

Návrh nášho nového systému bude založený na regresnom prístupe. Podobne ako v predchádzajúcom riešení 2.3 s tým rozdielom, že použijeme 2D konvo-

lúcie a naša sieť bude predikovať 2D tepelné mapy. 2D konvolúcie sú jednoduchšie na výpočet a má menšie nároky na kapacitu pamäte. Teda by sme znížili hardvérové požiadavky na chod systému, zatiaľ čo presnosť predikcie by sa nemusela znížiť.

Budeme predikovať 11 kĺbov na ruke. Sú to dva kĺby na každom prste a zápästie. Vďaka dvom kĺbom vieme určiť pozíciu prsta v obraze a tiež aj smer, ktorým prst ukazuje. Výstupná informácia z nášho systému bude preto vhodná aj pre úlohy, kde robotovi ukážeme na predmet a on bude vedieť označiť takto zvolený predmet.

Pridanie hĺbkovej informácie pre predikovaný kľúčový bod bude robiť mapovaním tepelnej mapy s hĺbkovou mapou, ktorú nám poskytuje výstup z kamery 1.5 Inel RealSense D430i.

Vytvoríme CNN zloženú z dvoch hourglass 1.2.4 modulov. Názorne ilustrovaný návrh je na obr. 3.2. Výstup prvého modulu bude vstupom do druhého hourglass modulu, ktorého výstup vytvorí pre každý kĺb jednu tepelnú mapu. Preto vytvoríme zo vstupných súradníc kĺbov tepelnú mapu pre každý kĺb. Mapa bude obsahovať Gausovu distribúciu kladných hodnôt so stredom v súradniciach kĺbu a štandardnou odchýlkou. Výber hodnoty odchýlky bude popísaný ďalej v tejto kapitole. Trénovanie bude potom reprezentované výpočtom euklidovskej vzdialenosti medzi očakávanými a predikovanými tepelnými mapami. Pre každých 32 vstupných obrázkoch bude vypočítaná priemerná hodnota tejto vzdialenosti, podľa ktorej sa upravia váhy v CNN.

Na trénovanie siete použijeme dataset NYUHands popísanom v 1.4 a najlepší model následne dotrénujeme na vlastnom datasete. Súradnice v našich obrázkoch určíme ručne. Vytvoríme si jednoduchý program, ktorý vezme pozíciu kliknutia v obrázku a zaznamená súradnice kĺbu. Na zvýšenie presnosti predikcie kĺbov pre naše konkrétne prostredie bude stačiť menšie množstvo

obrázkov, keďže sieť už bude vedieť predikovať nejaké pozície ruky.

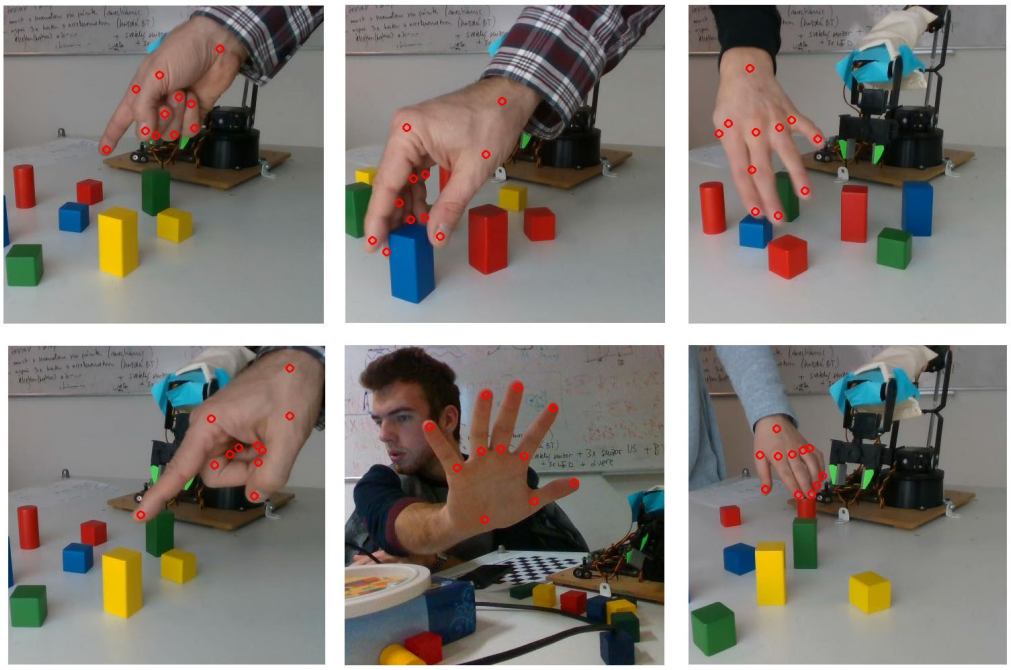
3.2 Implementácia

Systém budeme implementovať v jazyku Python. Tento jazyk ponúka vhodné knižnice na prácu s konvolučnými sieťami a tiež na načítanie výstupu z našej kamery. Ku kamere výrobca vytvoril knižnicu s názvom 'librealsense 2' v jazyku Python, ktorú používame na spracovanie údajov z kamery. Na vytvorenie konvolučnej neurónovej siete použijeme knižnicu 'Keras'. Vytvoríme vlastný dataset a na jeho tvorbu napíšeme skript v jazyku Python s využitím vyššie spomínanej knižnice na načítanie obrázkov z kamery. Na vykreslenie obrázka v ktorom určíme súradnice použijeme knižnicu 'openCV'.

3.2.1 Vlastný dataset

Prínosom pre predikciu súradníc v našom prostredí je doučiť sieť, ako vyzerá ruka v našom prostredí. Taktiež je vhodné na vstup do siete použiť rôzne polohy ruky. Na vytvorenie vlastného datasetu sme vytvorili skript v jazyku Python. Načítali sme dáta z kamery metódami knižnice 'librealsense 2' a postupne sme prechádzali jednotlivými obrázkami z videa. Vybrali sme niekoľko scén a v nich zopár obrázkov zobrazujúcich reprezentatívne polohy rúk. Na načítaných obrázkoch sme ručne zvolili body so súradnicami predstavujúcimi kĺby, ktoré má sieť predikovať. Každý bod, na ktorý sme klikli uložil svoju polohu súradnicami x, y tak, že bod $[0, 0]$ je v ľavom hornom rohu. Výhodou tohoto vlastného určovania pozície je, že sme súradnice ukladali vo forme, ktorú používame pri načítaní datasetu v našej sieti. Náš dataset obsahoval 100 obrázkov s označenými súradnicami kĺbov. Príklady polôh ruky v našom datasete sú znázornené na obr. 3.1, kde sú červenými krúžkami označené

nami určené súradnice. Tie však nie sú priamou súčasťou obrázka a vstup do siete je taký obrázok bez označenia súradníc.



Obr. 3.1: Príklady obrázkov v našom datasete.

Štruktúru datasetu sme robili podľa vzoru NYUHands, aby sme pri dotrénovaní nášho datasetu zmenili iba cestu a nemuseli meniť kód. Názov obrázku sme zvolili s prefixom 'rgb_1_', infix bol sedem miestne poradové číslo zhodné so súradnicami a sufixom '.jpg'. Súradnice sme ukladali do súboru vo formáte pre 'MatLAB' ako štvorrozmerný tenzor (P, I, J, C). Pričom P je index kamery (v našom prípade sme mali jednu, teda bol index 0), I predstavuje index obrázka (určujúci obrázok príslušný k daným súradniciam), J index kĺbu (určujúci pozície kĺbu) a C súradnice x, y príslušnému kĺbu.

3.2.2 Predspracovanie

Použili sme dáta z datasetu NYUhands ako vstup do našej siete a vybrali sme reprezentáciu obrázka v RGB. Tieto obrázky majú rozmer 640x480 pixlov a súradnice kĺbov tiež prislúchajú obrázkom vo forme u,v,d pre túto veľkosť. Keďže používame komplexnú sieť, je vhodné tieto dáta zmenšiť, aby sme urýchlili proces učenia a znížili nároky na veľkosť pamäte.

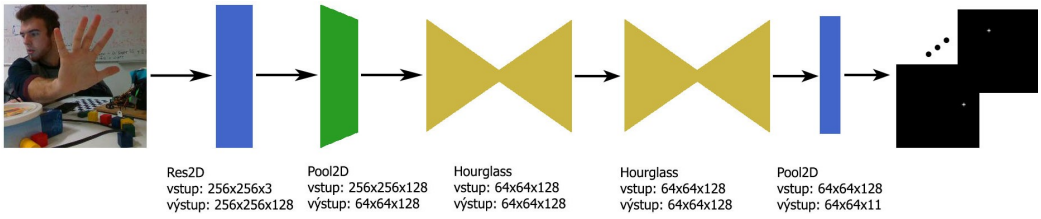
Obrázky sme upravili tak, aby mali štvorcový rozmer a následne znížili rozlíšenie na 256x256. Poloha ruky na všetkých obrázkoch v datasete sa nachádza na ľavej časti obrázka, teda pri upravení na štvorcový rozmer sme odstránili 160 pixlov z pravej strany. Po tejto úprave nám zostali všetky súradnice kĺbov nezmenené, nakoľko súradnicová sústava má bod [0,0] v ľavom hornom rohu.

Navrhnutá sieť bude predikovať tepelné mapy. Očakávané súradnice sú v datasete reprezentovaný jedným bodom v obrázky. Preto sme na základe týchto súradníc vytvorili tepelné mapy. Na ich vytvorenie sme použili Gausovu distribúciu so stredom súradníc kĺbu a štandardnou odchýlkou 13 pixlov. Hodnotu štandardnej odchýlky sme empiricky hľadali tak, aby na vytvorenej tepelnej mape pokrývala dostatočnú časť okolia bodu.

Tepelné mapy obsahujú hodnoty v intervale $[0, 1]$, kde nenulová hodnota vyjadruje pozíciu kĺbu. Počas trénovania by teda rozdiel medzi tepelnou mapou a predikciou mapy obsahujúcej samé nuly bol veľmi malý. Preto sme každú hodnotu prenásobili hodnotou sto a teda vývoj chyby počas trénovania začal rádovo v stovkách nad nulou a klesal smerom k nule.

3.2.3 Architektúra siete

Konvolučnú sieť sme zložili z dvoch reziduálnych častí, do ktorých vstupuje obrázok. Výstup prechádza cez poolingovú vrstvu do prvého modulu 'Hourglass'. Zloženie vrstiev v tomto module popíšeme neskôr. Pripojili sme druhý takýto modul priamo za predošlí. Výstupom z druhého Hourglass modulu boli tepelné mapy, z ktorých sme vyberali súradnice kĺbov. Rozmery výstupu sú štyrikrát menšie, teda sme interpoláciou zväčšili ich rozmer na rozmer vstupných obrázkov. Táto architektúra je ilustračne znázornená na obr. 3.2.



Obr. 3.2: Architektúra siete, ktorú sme navrhli a implementovali. Vstupný obrázok je reprezentovaný maticou s rozmermi 256x256. Prvý blok vrstiev podvzorkuje za použitia poolingových vrstiev vstupnú maticu na rozmer 64x64. Tento ďalej vstupuje do modulu hourglass. Predikovaným výstupom je tepelná mapa s rozmerom 64x64 a ten ďalej upravíme časti úpravy výstupu.

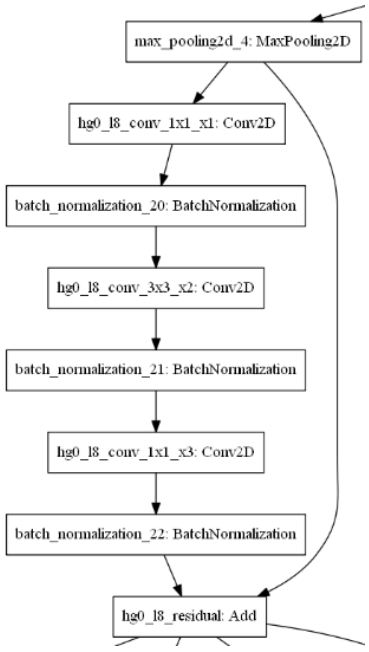
Hourglass implementácia

Modul hourglass sme implementovali v jazyku Python a použili pri tom knižnicu Keras ako sme už spomínali aj v úvode kapitoly. Ako základ pre tento modul sme vychádzali podľa Newell a kol. [11], ktorý vytvorili tento modul pre predikovanie kostry človeka.

V tejto architektúre sme využili reziduálne prepojenia (ich princíp sme ukázali v kapitole 1 odseku 1.2.3) také, že jedna vetva obsahuje konvolučnú vrstvu a vrstvu dávkovej normalizácie (nazývanú BatchNorm) trikrát za se-

bou a v druhej vetve nie je pridaná žiadna vrstva. Ilustrácia implementovaného reziduálneho prepojenia je na obr. 3.3. V časti zmenšovania vstupu sme použili pred každým reziduálnym prepojením poolingovú vrstvu s funkciou max. Takto sme zmenšovali vstup na polovicu po rozmer 8x8. Tu sme následne použili dve za sebou prepojení vyššie spomínané reziduálne prepojenia a ďalej sme pripravili časť zväčšovania. Výstup z vrstvy s menší rozmerom prešiel cez tzv. vrstvu UpSampling. Výstupom z nej bol rozmer dvojnásobný, teda rovnaký ako predchádzajúca vrstva. Matice na výstupe z týchto vrstiev boli sčítané a ich súčet bol upravený reziduálnym spojením. Tento postup sa opakoval až kým sme mali výstupný rozmer 64x64.

Posledné dve vrstvy, ktoré sme ešte pridali boli konvolučné s jadrom 1x1. Výstupná konvolučná vrstva mala počet filtrov rovnaký ako počet tepelných máp ktoré sme chceli predikovať. Tento počet sme implementovali konfigurovateľný, čo bolo vhodné pre testovanie a úpravu hyperparametrov.



Obr. 3.3: Reziduálne prepojenie v module hourglass.

3.2.4 Trénovanie

Trénovanie sme rozdelili do dvoch častí. V prvej sme trénovali sieť na datasete NYUhands a v druhej časti sme použili natrénovanú sieť na nájdenie súradníc kĺbov pre obrázky vo vlastnom prostredí. Na spustenie trénovania takejto komplexnej siete je potrebný veľký hardvérový výkon. Využili sme službu 'Google Cloud', kde sme vytvorili virtuálny počítač na serveroch od Google.

Celý proces trénovania bol spustený na virtuálnom počítači. Aby sme mohli konzolové okno zavrieť a náš počítač vypnúť, vytvorili sme terminálovú reláciu pomocou príkazu 'screen'. Vo vytvorenej relácii tak môže byť spustený proces a naša konzola môže byť zavretá. Taktiež sme potrebovali presunúť dataset na náš virtuálny počítač a natrénovaný model znovu späť na náš lokálny počítač. Na tieto presuny sme použili službu 'bucket' v pro-

stredí Google Cloudu. Virtuálnemu počítaču sme povolili zápis a čítanie na tento 'bucket' a nástrojom 'gsutil' sme presúvali súbory medzi 'bucketom' a virtuálnym počítačom.

Sieť je veľmi citlivá na inicializáciu. Pri nevhodnom počiatočnom nastavení váh nie sieť schopná zlepšovať predikciu súradníc. Všetky vrstvy sme inicializovali Xavierovým [2] uniformným rozdelením. Trénovanie prebiehalo v troch fázach, pričom v každej z nich sme zvýšili počet tepelných máp, pre ktoré sme počítali chybu. Postupovali sme tak, že najprv sme počítali chybu pre konček palca. Po niekoľkých epochách, keď vedela sieť predikovať pozíciu končeka palca nad 60%, pridali sme počítanie chyby pre všetky končeky prstov. Podobne ako v prvej fáze sme nechali trénovať sieť tak, aby každú mapu predikovala s presnosťou aspoň 60%. V tretej fáze sme nastavili počítanie chyby pre všetky kĺby. Matematický výpočet chyby je popísaný rovnicou (3.1). Pre každú fázu sme menili množinu M, ktorá obsahovala tepelné mapy príslušné k vyššie popísaným kĺbom.

Trénovanie na NYUHands

Vstupné dáta sme rozdelili na trénovaciu množinu s 80% obrázkov a validačnú množinu so zvyšnými 20% obrázkov určenými na trénovanie. Stav siete je upravený metódou spätného šírenia chyby (popísanou v kapitole 1 v časti 1.1). Pre každý obrázok v trénovacej množine sa vypočíta chyba, teda rozdiel medzi predikovanými a očakávanými tepelnými mapami. Matematicky vyjadrené nasledujúcou rovnicou:

$$L2 = \sqrt{\sum_{b \in B, i \in H, j \in W, m \in M} \left(y_{b,i,j,m} - \overline{y}_{b,i,j,h}\right)^2} \tag{3.1}$$

Kde B je množina obrázkov a k nim príslušným tepelných máp v jednej

dávke. H označuje počet pixlov v stĺpci a W počet pixlov v riadku tepelnej mapy. M je množina tepelných máp pre obrázok b.

Spätné šírenie je pre dávku obrázkov. Veľkosť dávky je prínosná pre rýchlosť učenia siete. Množstvo obrázkov v sieti je potrebné vhodne zvoliť, pretože s jej zvyšujúcim počtom narastá požiadavka na veľkosť pamäte. Nastavili sme veľkosť dávky na 32 obrázkov, teda najväčšiu možnú veľkosť, ktorá sa ešte zmestí do pamäte počítača.

Vypočíta sa priemerná hodnota zo všetkých chýb pre každý obrázok v dávke. Podľa nej sú potom upravené váhy v sieti. Takto sa postupuje pokiaľ sa vyhodnotia všetky obrázky v trénovacej množine. Tento postup nazývame priebeh epochy. Po každej epoche zistíme presnosť nášho modelu tak, že ako vstupné dáta zvolíme obrázky z validačnej množiny. V tejto časti trénovania sieť neupravuje hodnoty váh a slúži iba na vyhodnotenie kvality modelu.

Dotrénovanie na vlastných dátach

Inicializovali sme našu sieť váhami nastavenými podľa najlepšieho modelu, natrénovaného na datasete NYUhands. Výpočet chyby sme nastavili pre všetky kĺby podľa rovnice (3.1). Vstupné dáta sme načítali z našej kamery a sieť sa učila lepšie predikovať súradnice kĺbov pre obrázky v našom prostredí.

3.2.5 Vyhodnocovanie systému

Pri vyhodnocovaní systému sme chceli sledovať presnosť určenia súradníc kĺbov ruky. Taktiež nás zaujímala presnosť jednotlivých kĺbov. Rozhodovací systém, či sú súradnice správne využíval prahovú hranicu, ktorá určovala maximálnu vzdialenosť medzi predikovanými a očakávanými súradnicami. Takto sme určili kružnicu s polomerom prahu a stredom v očakávaných súradniciach. Podobne ako v kapitole 2 časti 2.2, kde prahová hranica určovala

priemer kruhu so stredom v očakávaných súradniciach, v ktorom sa musí predikovaný bod nachádzať, aby boli súradnice vyhodnotené ako správne.

Sledovali sme dve prahové hranice nastavené na 2mm a 5mm pre určenie presnosti súradníc všetkých kĺbov a pre určenie presnosti každého kĺbu samostatne sme použili prahovú hranicu 5mm. Presnosť sme určovali v percentách ako pomer medzi správne určenými a všetkými súradnicami.

Výpočet vzdialenosti bol ovplyvnený okrem predikcie aj zmenou veľkosti interpoláciou. Na minimalizovanie tejto chyby sme vydelili vypočítanú vzdialenosť konštantou, ktorú sme dostali nasledovným postupom. Tepelné mapy v pôvodnej veľkosti sme interpoláciou zmenšili na výstupný rozmer z našej siete (64x64). Tie sme spolu s príslušným obrázkom vložili na vstup pre určenie presnosti predikcie. Vedeli sme o správnosti všetkých kĺbov, a teda sme konštantu menili tak, aby systém vyhodnotil presnosť na 100%.

Kapitola 4

Výsledky

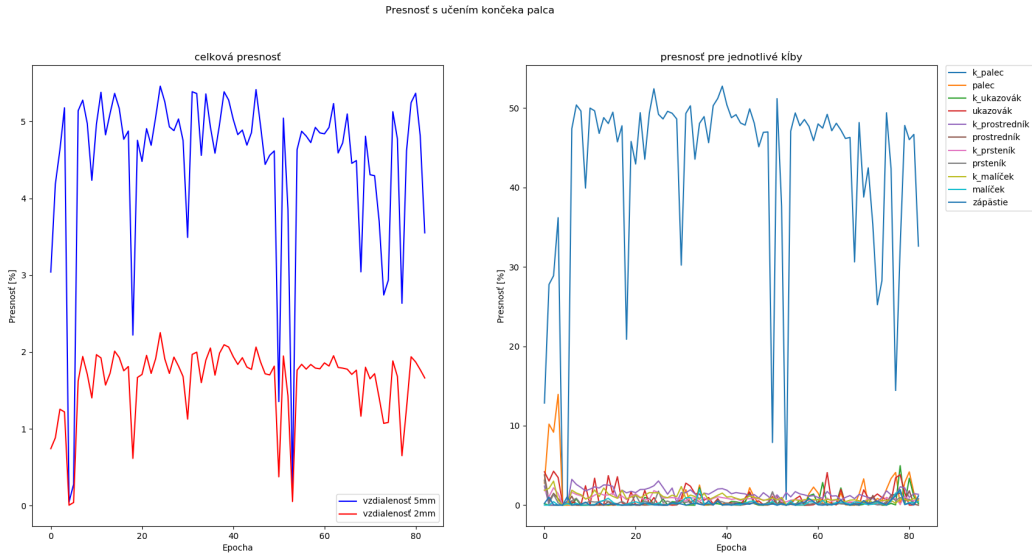
V tejto kapitole uvedieme dosiahnuté výsledky namerané spôsobom popísaným v predošlej kapitole 3. Obsahuje záznam o zmene presnosti trojfázového trénovania na datasete NYUHands. Ďalej uvedieme ako sa menila presnosť počas trénovania na našom datasete.

Pre všetky vyhodnocovania sme vytvorili dva grafy v dvoch stĺpcoch, kde graf na ľavej strane zobrazuje celkovú presnosť, teda priemernú hodnotu z presností všetkých kĺbov. Červenou je znázornená presnosť pre tie kĺby, ktorých predikované súradnice boli vzdialené od očakávaných súradníc do 2mm a modrou do 5mm. Uvádzané výsledky sú v tolerancii 5mm. Graf na pravej strane zobrazuje presnosť predikcie pre jednotlivé kĺby. Na tomto grafe sme použili označenie k_prst a znamená, že dané hodnoty prislúchajú končeku daného prsta.

Náš systém predikuje 2D súradnice a preto sme vyhodnocovali predikciu pre RGB obrázky a súradnice v 2D. Na vyhodnocovanie prvej časti trénovania na NYUHands datasete sme používali obrázky z validačnej množiny.

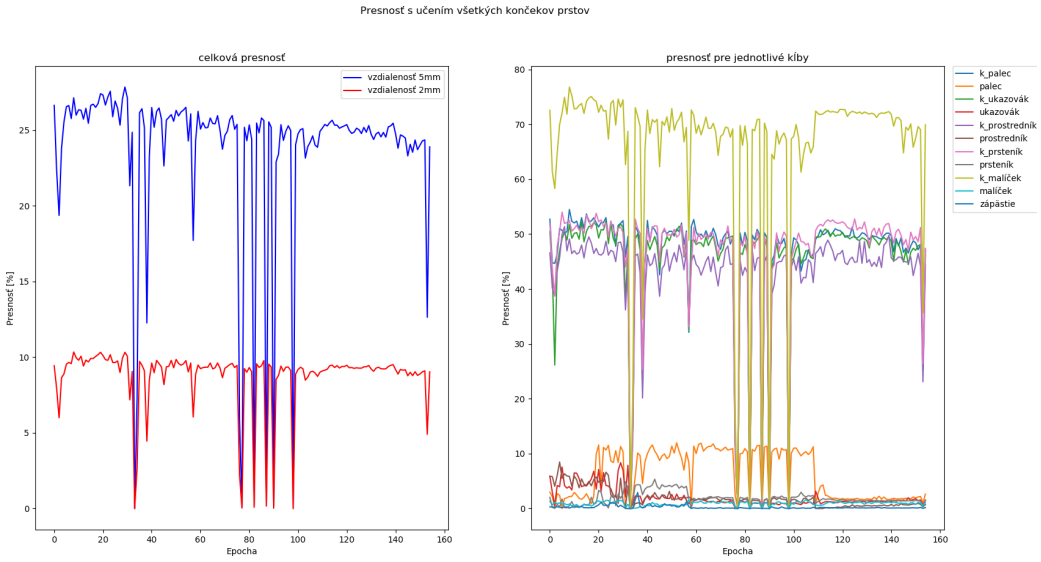
V prvej fáze trénovania sme nastavili počítanie chyby pre konček palca. Dosahovaná presnosť pre každú epochu je zobrazená na Obr. 4.1. V tejto

fáze učenia je teda vidieť zlepšenie predikcie pre konček palca a presnosť predikcie pre ostatné kĺby je blízka nule. Do druhej fázy sme vybrali také nastavenie váh ktoré dosiahlo najvyššiu presnosť podľa predikcie palca. Vrchol bol dosiahnutý v 39. epoche s presnosťou 52,74%.



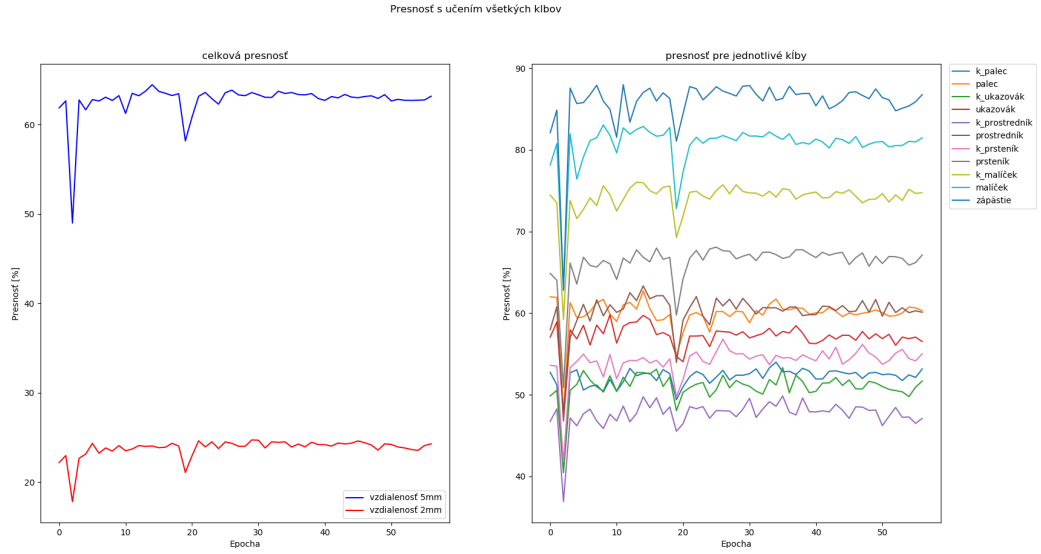
Obr. 4.1: Priebeh učenia prvej fázy s počítaním chyby pre konček palca.

Druhá fáza mala nastavené počítanie chyby pre všetky končeky prstov. Priebeh dosahovaných presností je zobrazený na Obr. 4.2. Keďže chceme náš model aj čo najlepšie generalizovať a nie ho iba sústrediť na najlepšie určenie jedného kĺbu, tak sme vybrali najvhodnejšie nastavenie váh podľa najvyššej priemernej presnosti z trénovaných kĺbov. Tá bola dosiahnutá v ôsmej epoche s priemernou presnosťou určenia končekov prstov 56,91%.



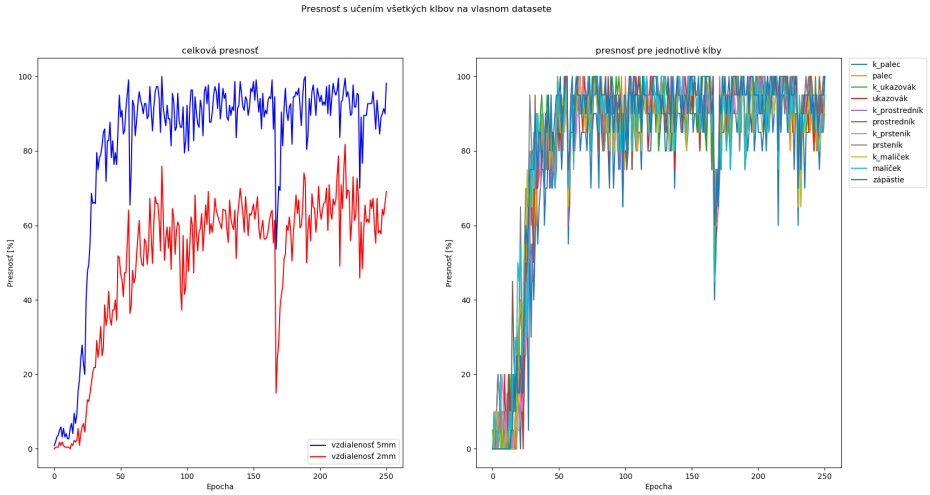
Obr. 4.2: Priebeh učenia druhej fázy. Počítanie chyby pre končeky prstov.

Pre tretiu fázu sme inicializovali váhy z ôsmej epochy predošlého kroku. Vývoj presnosti počas učenia je zobrazený na obr. 4.3. Celkovú presnosť predikcie na NYUhands náš systém dosiahol 64,48% v štrnástej epoche.



Obr. 4.3: Priebeh učenia tretej fázy s počítaním chyby pre všetky kĺby.

Náš dataset sme dotrénovali na sieti s inicializáciou váh zo 3trn8stej epochy predošlého trénovania. Vývoj presnosti je na obr. 4.4. Keďže v ňom máme veľmi málo obrázkov, rýchlo sa sieť naučila predikovať celkom presne súradnice kĺbov. Preto sme pre výber brali do úvahy väčší dôraz na presnosť s odchýlkou 2mm a vybrali sme taký model, ktorý viac kĺbov určil s presnosťou do tejto vzdialenosti. Taký výsledok bol dosiahnutý v 219. epoche s presnosť na 2mm bola úspešnosť 81.81% a na 5mm 99.54%.



Obr. 4.4: Priebeh učenia predikcie pozície kĺbov na našom datasete.

Kapitola 5

Diskusia

Uvedieme hlavné myšlienky, ktoré napomohli k zlepšeniu učenia. Bolo to najmä vďaka nastaveniu parametrov ako pri generovaní tepelných máp, tak aj pri trénovaní a trojfázovému trénovaniu. Ukážeme výsledky na konkrétnych príkladoch. Použijeme obrázky s takou polohou ruky, kedy náš systém správne predikuje súradnice kĺbov, ale aj také, kde je predikovaná pozícia zlá. Vyhodnotíme množstvo a rôznorodosť nášho datasetu pre dosahované výsledky v oblasti problematiky ľudského učiteľa pre robota.

5.1 Dolaďovanie trénovania

Tvorba tepelných máp zo súradníc využívala parametre ako škálu a normu. Výstupný rozmer tepelných máp zo siete je menší ako pôvodné obrázky, ku ktorým prislúchajú súradnice. Preto sme aj vstupné súradnice zmenšili na výstupný rozmer 64x64 pixlov z pôvodných rozmerov 256x256. Škála je vyjadrená číslom predstavujúcim koľko násobne sme tepelnú mapu zmenšili. Táto zmena prispieva ku odchýlke a posunu súradníc. Teda pri spätnom zväčšení obrázkov nedostávame rovnaký pixel s rovnakými súradnicami. To

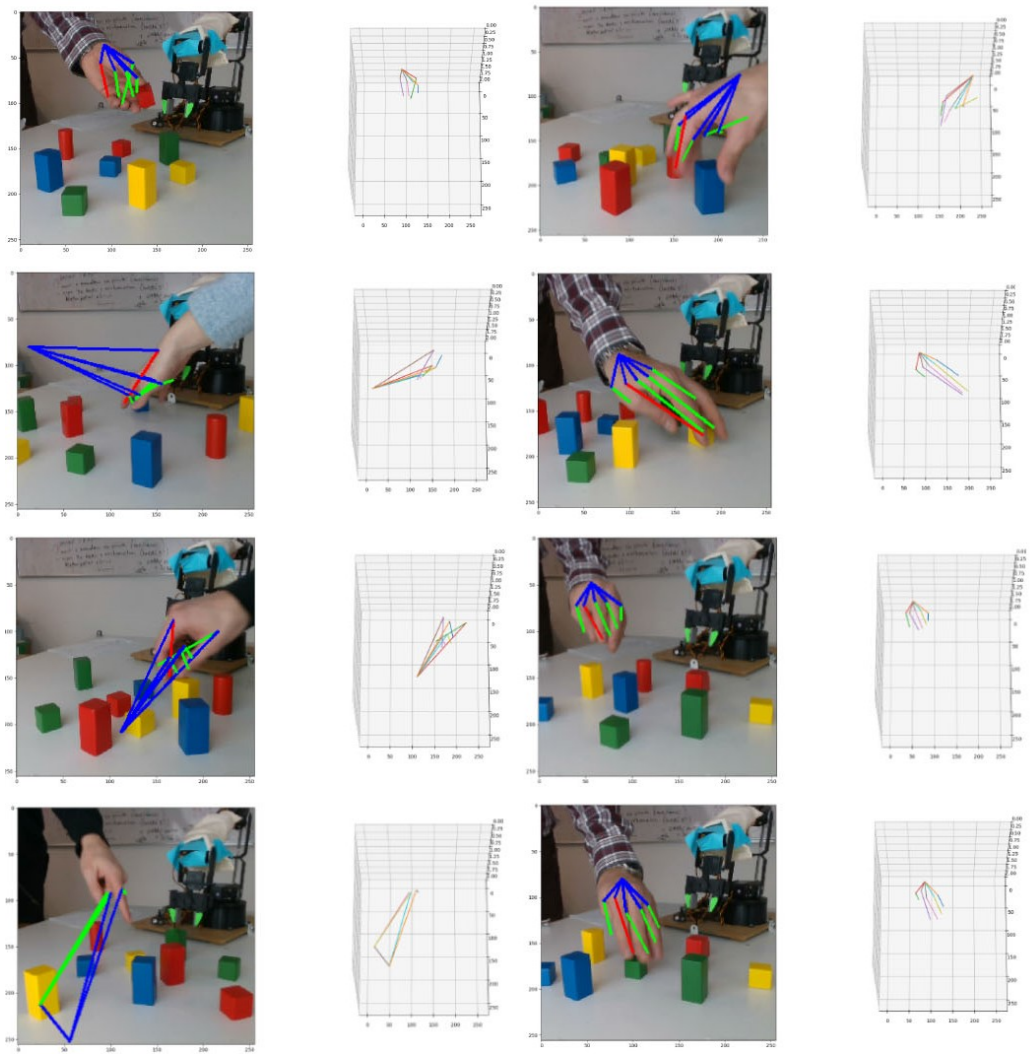
bol dôvod pre zvolenie normy spomínanej v úvode odseku. Norma je reprezentovaná jedným číslom, ktoré vyjadruje rozdiel medzi pôvodnou tepelnou mapou a tepelnou mapou opäť zväčšenou z rozmerov výstupu CNN tak, aby sme vyhodnotili zhodu súradníc medzi mapami.

Sieti sme nastavovali viacero parametrov. Počet filtrov pre reziduálne vetvy v module 'Hourglass', ktorý vplýval na rýchlosť učenia. Z rastom počtu filtrov sa čas učenia zvyšoval. Pri nastavení malého počtu filtrov nebola sieť schopná naučiť sa a správne predikovať súradnice. Skúšali sme teda niekoľko možností a najvhodnejšie bolo zvoliť 128. Pri tomto počte sme sieť natrénovali. Pri nastavení 256 filtrov nebolo vidieť prínos pri učení, iba bol čas učenia dlhší. Naopak pri menšom počte ako 64 sa sieť nedokázala naučiť správne predikovať súradnice v rovnakom čase ako s použitím 128 filtrov. Ďalej sme nastavili pre každú vrstvu aktivačnú funkciu 'ReLU' a pre výstupné plne prepojené vrstvy funkciu 'sigmoid'.

Najväčším prínosom pre správne predikovanie súradníc bolo rozdelenie trénovania na tri fázy. Pôvodný postup trénovania bol s použitím stratovej funkcie pre všetky kĺby. Mali sme inicializované váhy náhodne 'Xavierovskou' distribúciou. Po spustení trénovania výsledky predikcie sa nezlepšovali ani po vyše 100 epochách. Neskôr sme zistili, že ak zvolíme predikciu iba jednej tepelnej mapy, sieť sa dokáže naučiť správne predikovať tepelnú mapu. Preto sme zvolili tréning tak, aby sme stále predikovali všetky tepelné mapy, ale váhy menili iba podľa výsledkov z jednej mapy. Vždy keď sieť zmenila váhy tak, že vedela aspň polovicu prípadov správne zvoliť súradnice kĺbov, podľa ktorých menila váhy, prešli sme na ďalšiu fázu.

5.2 Vyhodnotenie na obrázkoch

Natrénovanú sieť na našom datasete sme použili na predikovanie súradníc kĺbov ruky na rôznych videách. Z týchto videí sme nepoužili žiaden obrázok v našom datasete. Výstupné obrázky s predikovanými súradnicami je na obr. 5.1, kde sme vykresľovali spojenia medzi týmito súradnicami.



Obr. 5.1: Príklady obrázkov s predikovanými súradnicami. Ukazovák je zvýraznený červenou úsečkou, ostatné prsty zelenou a spojenia prstov so zápästím sú označené modrou úsečkou.

Záver

Literatúra

[1] Léon Bottou. Stochastic gradient tricks. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks, Tricks of the Trade, Reloaded*, Lecture Notes in Computer Science (LNCS 7700), pages 430–445. Springer, 2012.

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[4] S. Haykin and S.S. Haykin. *Neural Networks and Learning Machines*. Number zv. 10 in Neural networks and learning machines. Prentice Hall, 2009.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] Fuyang Huang, Ailing Zeng, Minhao Liu, Jing Qin, and Qiang Xu. Structure-aware 3d hourglass network for hand pose estimation from single depth image. *CoRR*, abs/1812.10320, 2018.

[7] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3d skeletal hand tracking. *CoRR*, abs/1705.07640, 2017.

[8] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. *CoRR*, abs/1711.07399, 2017.

[9] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time hand tracking under occlusion from an egocentric RGB-D sensor. *CoRR*, abs/1704.02201, 2017.

[11] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016.

[12] Iason Oikonomidis. *Efficient Tracking of the 3D Articulated Motion ofHuman Hands*. PhD dissertation, University of Crete, 2015.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015.

[14] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[15] Jonathan Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *CoRR*, abs/1406.2984, 2014.

[16] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics*, 33, August 2014.

[17] Xiaokun Wu, Daniel Finnegan, Eamonn O'Neill, and Yong-Liang Yang. Handmap: Robust hand pose estimation via intermediate dense guidance map supervision. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[18] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 3d hand pose tracking and estimation using stereo matching. *CoRR*, abs/1610.07214, 2016.

[19] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. Technical report, arXiv:1705.01389, 2017. https://arxiv.org/abs/1705.01389.

Zoznam obrázkov